Table of Contents

Introduction to Visual Paradigm

This chapter gives you an introduction about Visual Paradigm. The following topics will be covered:

Visual Paradigm product overview

A brief description of Visual Paradigm which outlines some of the key features that Visual Paradigm supports.

Editions

A summary of editions and their supported features.

Licensing

Visual Paradigm need to run with a key. This page shows you how to import different kinds of key into Visual Paradigm.

Software maintenance

Describes what software maintenance provides and tell you why you need it.

System requirements

A description of hardward requirements.

Visual Paradigm user's guide trademark information

Product names, logos, brands and other trademarks refer to within Visual Paradigm's products and services and within visual-paradigm.com are the property of their respective trademark holders. These trademark holders are not affiliated with Visual Paradigm International, our products, or our website. They do not sponsor or endorse our materials. Below is a partial listing of these trademarks and their owners. This list is subject to change without notice.

- ActionScript is a trademark of Adobe Systems Inc.
- Android is a trademark of Google Inc.
- ArchiMate is a trademark of The Open Group.
- BPMN, CORBA and UML are trademarks of the Object Management Group
- C#, C++, MS SQL Server, Microsoft Windows, Microsoft Visio, VB.NET and Visual Studio are trademarks of Microsoft Corporation.
- Cache is a trademark of InterSystems Corporation.
- Cloudscape, DB2, Informix and Rhapsody are trademarks of International Business Machines Corporation.
- Derby is a trademark of the Apache Software Foundation.
- Eclipse is a trademark of Eclipse Foundation, Inc.
- Erwin is a trademark of CA Inc.
- Firebird is a trademark of Firebird Foundation Inc.
- FrontBase is a trademark of FrontBase Inc.
- H2 is a trademark of H2, Inc.
- Hibernate is a trademark of Red Hat, Inc.
- HSQL is a trademark of the hsql Development Group.
- IntelliJ IDEA is a trademark of JetBrains.
- Ingres is a trademark of Actian Corporation.
- iPad and iPhone are trademarks of Apple Inc.
- Java, MySQL and Oracle are trademarks of Oracle Corporation.
- NetBeans is a trademark of Sun Microsystems, Inc.
- Objective-C is a trademark of Stepstone Corporation
- OpenEdge is a trademark of Progress Software Corporation.
- Perl is a trademark of The Perl Foundation.
- PHP is a trademark of The PHP Group.
- PostgreSQL is a trademark of the PostgreSQL Global Development Group.
- Python is a trademark of the Python Software Foundation
- SQLite is a trademark of Hipp, Wyrick & Company, Inc.
- Sybase ASE are Sybase SQL Anywhere are trademarks of Sybase Inc.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Visual Paradigm official website
- Contact us if you need any help or have any suggestion

Visual Paradigm product overview

Visual Paradigm is a powerful, cross-platform and yet easy-to-use visual UML modeling and CASE tool. Visual Paradigm provides software developers the cutting edge development platform to build quality applications faster, better and cheaper! It facilitates excellent interoperability with other CASE tools and most of the leading IDEs which excels your entire Model-Code-Deploy development process in this one-stop-shopping solution.

UML modeling

You can draw all kinds of UML 2.x diagrams in Visual Paradigm, which include:

- Class diagram
- Use case diagram
- Sequence diagram
- Communication diagram
- State machine diagram
- Activity diagram
- Component diagram
- Deployment diagram
- Package diagram
- Object diagram
- Composite structure diagram
- Timing diagram
- Interaction overview diagram

Requirement modeling

Capture requirements with SysML Requirement Diagram, Use Case Modeling, Textual Analysis, CRC Cards, and create screen mock-up with User Interface designer.

Database modeling

You can draw the following kinds of diagrams to aid in database modeling:

- Entity Relationship Diagram
- ORM Diagram (visualize the mapping between object model and data model)

You can model not only database table, but also stored procedure, triggers, sequence and database view in an ERD.

Besides drawing a diagram from scratch, you can reverse engineer a diagram from an existing database.

Apart from diagramming, you can also synchronize between class diagram and entity relationship diagram to maintain the consistency between them.

SQL generation ane execution feature is available for producing and executing SQL statement from model instantly.

Business process modeling

You can draw the following kinds of diagrams to aid in business process modeling:

- Business process diagram
- Data flow diagram
- Event-drive process chain diagram
- Process map diagram
- Organization Chart

Object-Relational mapping

Object-Relational Mapping enables you to access relational database in an object relational approach when coding. Visual Paradigm generates object-relational mapping layer which incorporates features such as transaction support, pluggable cache layer, connection pool and customizable SQL statement.

Team collaboration

For users that work as a team, team collaboration support lets you perform modeling collaboratively and concurrently with any one of the following tools or technologies:

- VPository
- VP Teamwork Server (Need to buy Visual Paradigm Teamwork Server separately)

Documentation generation

Share your design with your customers in popular document formats, including:

- HTML (report generation)
- HTML (project publisher)
- PDF
- Microsoft Word

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Visual Paradigm official website
- Visual Paradigm's UML modeling supports

Editions

Belows are the kinds of features that can be found in each edition of Visual Paradigm. For details, please visit: http://www.visual-paradigm.com/editions

	Enterprise	Professional	Standard	Modeler	Community
UML Modeling	●	●	●	●	●
Use Case Diagram	●	●	●	●	●
Flow of Events Editor	●	●			
Requirement Diagram	●	●	●	●	●
SoaML Modeling	●	●			
Glossary Grid	●	●	●	●	
Entity Relationship Diagram	●	●	●	●	●
Generate Hibernate Mapping	●	●	●		
Business Process Modeling	●	●	●	●	
Mind Mapping	●	●	●		
Visual Diff	●	●	●		
Design Animation	●	●			
Business Process Simulation	●				
Collaborative Modeling	●	●	●	●	
Instant Forward and Reverse Engineering	●	●	●		
Java and C++ Round Trip Engineering	●	●	●		
Impact Analysis	●	●			
PDF, HTML, Word Document Generation	●	●	●	●	
Doc. Composer	●	●	●		

The filled circle indicates the support of certain feature in certain edition

A summary of features supported by Visual Paradigm

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Full comparison of Visual Paradigm editions, from community edition to enterprise edition
- Overview of Visual Paradigm enterprise edition, with features and strength listed
- Overview of Visual Paradigm professional edition, with features and strength listed
- Overview of Visual Paradigm standard edition, with features and strength listed
- Overview of Visual Paradigm modeler edition, with features and strength listed

- Overview of Visual Paradigm community edition, with features and strength listed

Licensing

Visual Paradigm needs to run with a valid license key. The various licensing options listed in this page vary in price and functionality.

Various licensing options

Single seat license

Visual Paradigm's single-seat (team-member-based) license allows one licensee to install the software on a computer that belongs and provides sole access to the named user only. Since the license is team-member-based, the software must be used by the licensee only, without running more than one instance concurrently. The single-seat license only allows installation on a maximum of three computers.

Edition	Unit price	Unit price (with 1 year maintenance)
Enterprise	$1399.00	$1678.50
Professional	$699.00	$838.50
Standard	$299.00	$358.50
Modeler	$99.00	$118.50
Community	Free for non-commercial use	Not applicable
Viewer	Free	Not applicable

Price of single seat licenses (Prices are provided in US dollars)

Floating license

The floating license supports sharing of the pool of licenses among your team. Instead of purchasing a single-seat license for each team member, optimize your budget by purchasing floating licenses for the maximum number of simultaneous software users or access points. This approach allows greater flexibility in using our software. Users can then export the license files to a laptop to use the software offsite (to deliver a presentation, for example) and then import the license back to the server at a later time.

Edition	Unit price	Unit price (with 1 year maintenance)
Enterprise	$1818.50	$2182.00
Professional	$908.50	$1090.00
Standard	$388.50	$466.00
Modeler	$128.50	$154.00
Community	Not applicable	Not applicable
Viewer	Not applicable	Not applicable

Price of floating licenses (Prices are provided in US dollars)

In order to work with the floating license, installation of a VP Server that stores the license key file(s) and also automatically manages access requests from clients is required. The client must enable the connection to the license server when requesting access to the software.

For more information about floating licenses, please visit
http://www.visual-paradigm.com/shop/floatinglicense.jsp

Subscription license

Edition	Unit price (per month)
Enterprise	$69.00
Professional	$35.00
Standard	$15.00
Modeler	$5.00
Community	Not applicable
Viewer	Not applicable

Price of subscription licenses (Prices are provided in US dollars)

Academic license

Academic licenses are available for higher education, with the aim of providing site licenses for the teaching of software engineering. Educational institutions that join the Academic Partners Program are entitled to licenses for the Visual Paradigm Enterprise Edition, which can then be used solely for educational purposes. The academic license is not limited to use on campus, but can also be used at home by students and teachers.

For more information about academic licenses, please visit
http://www.visual-paradigm.com/partner/academic/ .

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Buy Visual Paradigm, the UML modeling software online in Visual Paradigm online store
- Use Visual Paradigm for educational purposes. Join Visual Paradigm Academic Partner Program (VPAPP)
- Wanna know more about how floating license can help you share license between users? Visit this page
- See the most complete, updated Visual Paradigm price details
- Can you switch to another edition after purchase? Yes, you can. See how to do
- Keep Visual Paradigm updated. Know more about Visual Paradigm's product maintenance policy

Software maintenance

The Visual Paradigm Software Maintenance package includes both version upgrades and technical support services for our customers. The following benefits are all included in the Visual Paradigm Software Maintenance package.

Version upgrades

Typically, Visual Paradigm produces two to three versions per year, with approximately five to ten major new features and enhancements per version. You are entitled to install any new versions of Visual Paradigm product that are released within your software maintenance period.

Technical support

You and your team members can submit technical support tickets to our Technical Support Team at http://www.visual-paradigm.com/support/

Our Technical Support Team will respond to your message within one working day. Normally, you will receive our response by email within a few hours.

Visual Paradigm is committed to deliver extraordinary technical support to our customers. Our Technical Support Team employs the following technologies to back up our products.

Email with text and screen shot attachments

In most cases, we can provide assistance by guiding you with the aid of screen shots.

Flash demo

Sometimes, a short movie is more descriptive than a thousand words. If the answer to your question is complex, we can prepare a short Flash demonstration to guide you in resolving your difficulty.

Secure online sessions

We can schedule an online meeting with you to take an interactive look at your issue. Online meetings are held using a secure Internet connection. During the meeting, our team can remotely access and operate your PC while speaking with you by telephone or while chatting with you using the built-in chat program.

Telephone

You can leave a callback request at the following URL. Our Technical Support Team will return your call as soon as possible.

Price

Software maintenance is purchased on an annual basis (e.g., Aug 20, 2014 to Aug 19, 2015).

If you decide to purchase the software maintenance package with your product or if you decide to extend a current maintenance contract, the yearly cost is 20% of the product list price. To take advantage of this 20% offer, you must extend your maintenance contract at least one week prior to its expiration date.

If you decide to purchase a software maintenance package separately, the yearly cost is 30% of the product list price.

You can purchase software maintenance to cover up to three years from the date of purchase.

Detailed software maintenance package pricing is listed below.

Single seat license

Prices are provided in US dollar

Edition	1 Year Maintenance (extend current maintenance)	1 Year Maintenance (buy maintenance separately)
Enterprise	$279.50	$419.50
Professional	$139.50	$209.50
Standard	$59.50	$89.50
Modeler	$19.50	$29.50
Community	not applicable	not applicable
Viewer	not applicable	not applicable

The above software maintenance contract prices are for 1 year only.

Price for single-seat license

Floating license

Prices are provided in US dollar

Edition	1 Year Maintenance (extend current maintenance)	1 Year Maintenance (buy maintenance separately)
Enterprise	$363.50	$545.50
Professional	$181.50	$272.50
Standard	$77.50	$116.50

Modeler	$25.50	$38.50
Community	not applicable	not applicable
Viewer	not applicable	not applicable

The above software maintenance contract prices are for 1 year only.

Price for floating license

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Buy Visual Paradigm, the UML CASE Tool online in Visual Paradigm online store
- Use Visual Paradigm for educational purposes. Join Visual Paradigm Academic Partner Program (VPAPP) for FREE
- See the most complete, updated Visual Paradigm price details
- Can you switch to another edition after purchase? Yes, you can. See how to do

System requirements

Hardware requirements

- Intel Pentium 4 at 2.0 GHz or higher
- Minimum 2.0 GB RAM, but 4.0 GB is recommended.
- Minimum 4GB disk space (NOT include project space).
- Microsoft Windows (XP/Vista/7/8,10), Microsoft Windows Server (2000/2003/2008/2012), Linux, Mac OS X 10.7.3 or above

IDE requirements (for IDE integration)

- Eclipse 3.5 or above
- IntelliJ IDEA 11.0 or above
- Android Studio 1.3 or above
- NetBeans 6.7 or above
- Microsoft Visual Studio 2010, 2012, 2013, 2015

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Download Visual Paradigm and try it FREE
- Download the community edition of Visual Paradigm - simply free for non-commercial use

Installing Visual Paradigm

This chapter covers mainly the installation of Visual Paradigm on various platforms, as well as the steps of switching between product editions and how to remove Visual Paradigm.

Installing Visual Paradigm on Windows

List the steps of installing Visual Paradigm on Microsoft Windows as well as the use of InstallFree version.

Installing Visual Paradigm on Mac OS X

List the steps of installing Visual Paradigm on Mac OS X as well as the use of InstallFree version.

Installing Visual Paradigm on Linux and Unix

List the steps of installing Visual Paradigm on Linux and Unix as well as the use of InstallFree version.

Uninstalling Visual Paradigm

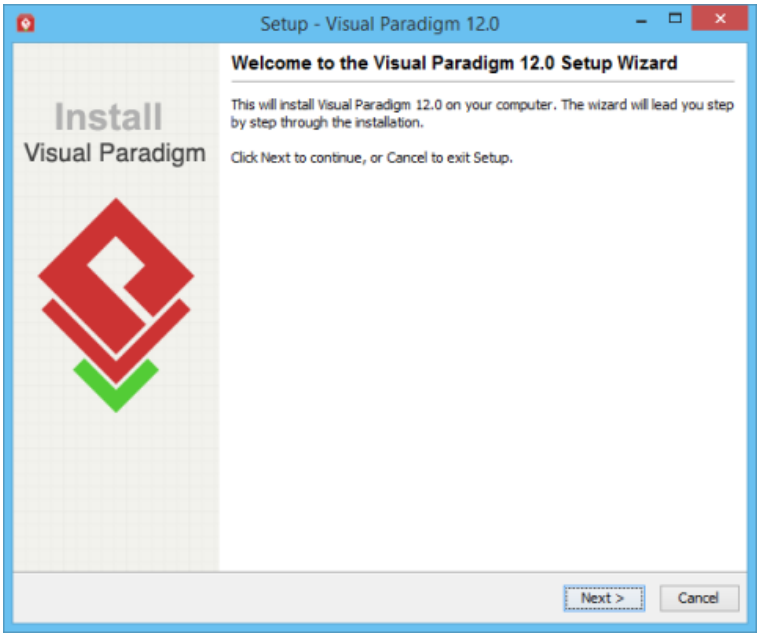
List the steps of uninstalling Visual Paradigm.

Installing Visual Paradigm on Windows

Having downloaded the installer of Visual Paradigm, execute it, run through the installation to install Visual Paradigm. If you are using the InstallFree version, you just need to unzip it and run Visual Paradigm directly. In this chapter, we will go through the installation of Visual Paradigm both with installer (.exe) and InstallFree (.zip).

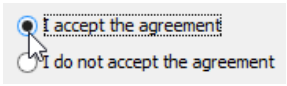
Using installer (.exe)

1. Execute the downloaded Visual Paradigm installer file. The setup wizard appears as below.



Visual Paradigm welcome screen

2. Click **Next** to proceed to the **License Agreement** page.
3. Read through the license agreement carefully. Make sure you accept the terms before continuing with the installation. If you accept the agreement, select **I accept the agreement** and click **Next** to proceed to the **Select Destination Directory** page.



The License Agreement

4. Specify the directory for installing Visual Paradigm. Click **Next** to proceed to the next page.
5. Specify the name of the Start Menu folder that will be used to store the shortcuts. Keep **Create shortcuts for all users** checked if you want the shortcut to be available in all the user accounts in the machine. Click **Next** to proceed.
6. In the **File Association** page, keep **Visual Paradigm Project (*.vpp)** checked if you want your system able to open the project file upon direct execution (i.e. double click). Click **Next** to start the file copying process.
7. Upon finishing, you can select whether to start Visual Paradigm or not. Keep **Visual Paradigm** selected and click **Finish** will run Visual Paradigm right away.

Using InstallFree version (.zip)

Decompress the downloaded zip file into a directory. This creates a subdirectory named "Visual Paradigm 12.2" where 12.2 is the version number. That's it. To start Visual Paradigm, execute **Visual Paradigm 12.2\Visual Paradigm.exe**.

Installation FAQ

Question: What is the difference between Installer and InstallFree Version?
Answer: Installer version creates shortcut and registers the menus that make the system more easy to use. We suggest user to use installer version for a long term usage. The InstallFree version is good for evaluation and testing the release candidate.
Question: I cannot complete the installation due to a file is missing when copying files. What can I do?
Answer: This can be caused by a corrupted installer file. Please download the installer file again with a different mirror site and run it again to solve the problem.
Question: I cannot start the application after installing the software. What can I do?
Answer: There are several possible causes of the problem. If you are sure that your installation was performed correctly, contact Visual Paradigm's support team for assistance. It is recommended to include the **vp.log** file in **%HOME-DIR%\visualparadigm** (e.g. *C:\Users\Peter\visualparadigm\vp.log*).
Question: I don't have administrator right, can I install the software?
Answer: Yes, you can.
Question: The installer file is detected to contain a virus. What can I do?
Answer: Our installer files are all packed by ourselves in a secure environment and are scanned for virus before releasing to public. If a virus is detected, please update to the latest virus profile first. After that, we recommend you to perform a full system scan, download the installer file from our official site and run the installation again. If the problem remains, please contact us or the virus scanner vendor for assistance.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

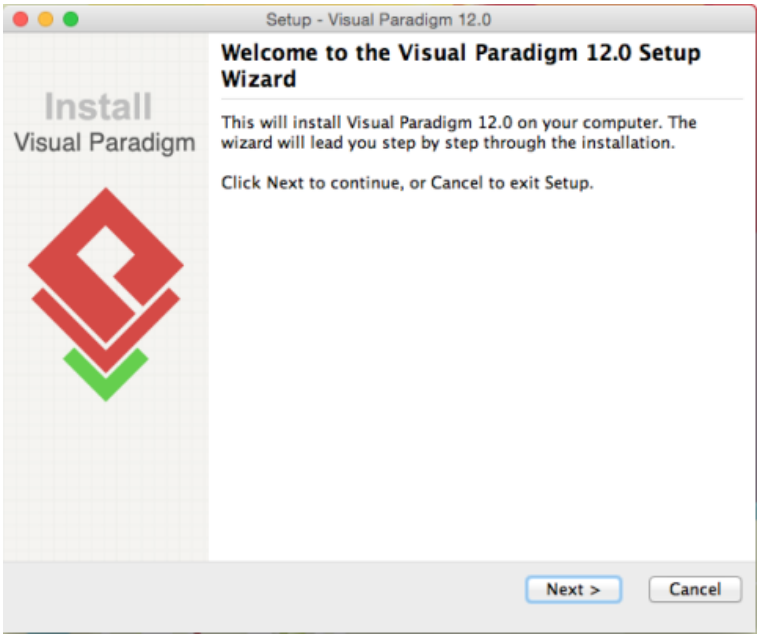
- Download Visual Paradigm and try it FREE
- Download the community edition of Visual Paradigm - simply free for non-commercial use
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Have difficulties when installing Visual Paradigm? Contact us. We will help you out

Installing Visual Paradigm on Mac OS X

Having downloaded the installer of Visual Paradigm, execute it, run through the installation to install Visual Paradigm. If you are using the InstallFree version, you just need to unzip it and run Visual Paradigm directly. In this chapter, we will go through the installation of Visual Paradigm both with installer (.dmg) and InstallFree (.tgz).

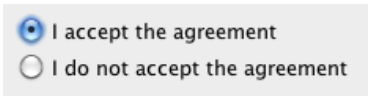
Using installer (.dmg)

1. Execute the downloaded Visual Paradigm installer file. The setup wizard appears as below.



Visual Paradigm welcome screen

2. Click **Next** to proceed to the **License Agreement** page.
3. Read through the license agreement carefully. Make sure you accept the terms before continuing with the installation. If you accept the agreement, select **I accept the agreement** and click **Next** to proceed to the **Select Destination Directory** page.



The License Agreement

4. Specify the directory for installing Visual Paradigm. Click **Next** to proceed to the next page.
5. In the **File Association** page, keep **Visual Paradigm Project (*.vpp)** checked if you want your system able to open the project file upon direct execution (i.e. double click). Click **Next** to start the file copying process.
6. Upon finishing, you can select whether to start Visual Paradigm or not. Keep **Visual Paradigm** selected and click **Finish** will run Visual Paradigm right away.

Using InstallFree version (.tgz)

Decompress the downloaded .tgz file into a directory. This creates a subdirectory named "Visual Paradigm 12.2" where 12.2 is the version number. That's it. To start Visual Paradigm, execute **Visual Paradigm 12.2\Visual Paradigm.app**.

Installation FAQ

Question: What is the difference between Installer and InstallFree Version?
Answer: Installer version creates shortcut and registers the menus that make the system more easy to use. We suggest user to use installer version for a long term usage. The InstallFree version is good for evaluation and testing the release candidate.
Question: I cannot complete the installation due to a file is missing when copying files. What can I do?
Answer: This can be caused by a corrupted installer file. Please download the installer file again with a different mirror site and run it again to solve the problem.
Question: I cannot start the application after installing the software. What can I do?
Answer: There are several possible causes of the problem. If you are sure that your installation was performed correctly, contact Visual Paradigm's support team for assistance. It is recommended to include the **vp.log** file in **%HOME-DIR%\visualparadigm** (e.g. *C:\Users\Peter\visualparadigm\vp.log*).
Question: The installer file is detected to contain a virus. What can I do?
Answer: Our installer files are all packed by ourselves in a secure environment and are scanned for virus before releasing to public. If a virus is detected, please update to the latest virus profile first. After that, we recommend you perform a full system scan, download the installer file from our official site, and run the installation again. If the problem remain, please contact us or the virus scanner vendor for assistance.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Download Visual Paradigm and try it FREE

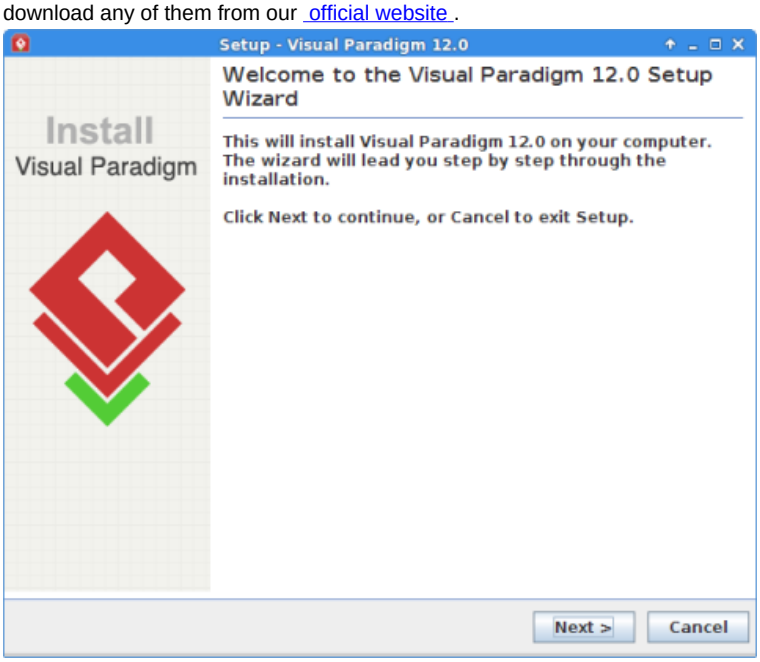
- Download the community edition of Visual Paradigm - simply free for non-commercial use
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Have difficulties when installing Visual Paradigm? Contact us. We will help you out

Installing Visual Paradigm on Linux and Unix

Having downloaded the installer of Visual Paradigm, execute it, run through the installation to install Visual Paradigm. If you are using the InstallFree version, you just need to unzip it and run Visual Paradigm directly. In this chapter, we will go through the installation of Visual Paradigm both with installer (.sh) and InstallFree (.tar.gz).

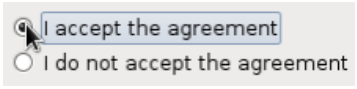
Using installer (.sh)

1. Execute the downloaded Visual Paradigm installer file.
   bash ./%Visual Paradigm-INSTALLER-FILENAME%
   The setup wizard appears as below. If you are prompted an error like "bin/unpack200: /lib/ld-Linux.so.2: bad ELF interpreter: No such file or directory. Error unpacking jar files. The architecture or bitness (32/64)", make sure you are executing the right installer - 64 bit / 32 bit. You can download any of them from our official website.



Visual Paradigm welcome screen

2. Click **Next** to proceed to the **License Agreement** page.
3. Read through the license agreement carefully. Make sure you accept the terms before continuing with the installation. If you accept the agreement, select **I accept the agreement** and click **Next** to proceed to the **Select Destination Directory** page.



The License Agreement

4. Specify the directory for installing Visual Paradigm. Click **Next** to proceed to the next page.
5. Select a folder for creating symlinks. You may uncheck **Create symlinks** if you do not want to. Click **Next** to the start file copying process.
6. Upon finishing, you can select whether to start Visual Paradigm or not. Keep **Visual Paradigm** selected and click **Finish** will run Visual Paradigm right away.

Using InstallFree version (.tar.gz)

Decompress the downloaded .tar.gz file into a directory: *tar -zxf %INSTALL-FREE-FILE.tar.gz% -C %DESTINATION-FOLDER%*

This creates a subdirectory named "Visual Paradigm 12.2" where 12.2 is the version number. That's it. To start Visual Paradigm, execute **Visual Paradigm 12.2\Visual Paradigm**.

Installation FAQ

Question: What is the difference between Installer and InstallFree Version?
Answer: Installer version creates shortcut and registers the menus that make the system more easy to use. We suggest user to use installer version for a long term usage. The InstallFree version is good for evaluation and testing the release candidate.
Question: I cannot complete the installation due to a file is missing when copying files. What can I do?
Answer: This can be caused by a corrupted installer file. Please download the installer file again with a different mirror site and run it again to solve the problem.
Question: I cannot start the application after installing the software. What can I do?
Answer: There are several possible causes of the problem. If you are sure that your installation was performed correctly, contact Visual Paradigm's support team for assistance. It is recommended to include the **vp.log** file in **%HOME-DIR%\visualparadigm** (e.g. *C:\Users\Peter\visualparadigm \vp.log*).
Question: The installer file is detected to contain a virus. What can I do?
Answer: Our installer files are all packed by ourselves in a secure environment, and are scanned for virus before releasing to public. If a virus is detected, please update to the latest virus profile first. After that, we recommend you to perform a full system scan, download the installer file from our official site and run the installation again. If the problem remain, please contact us or the virus scanner vendor for assistance.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Download Visual Paradigm and try it FREE
- Download the community edition of Visual Paradigm - simply free for non-commercial use
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Have difficulties when installing Visual Paradigm? Contact us. We will help you out

Activating/De-activating Visual Paradigm

Starting Visual Paradigm

List the steps of starting Visual Paradigm, with a brief description on 'workspace'.

Evaluating Another Edition

You can switch between editions without re-installation. This page shows you how to do.

Activating/Deactivating Visual Paradigm

Learn how to apply your license in Visual Paradigm, and how to deactivate Visual Paradigm.

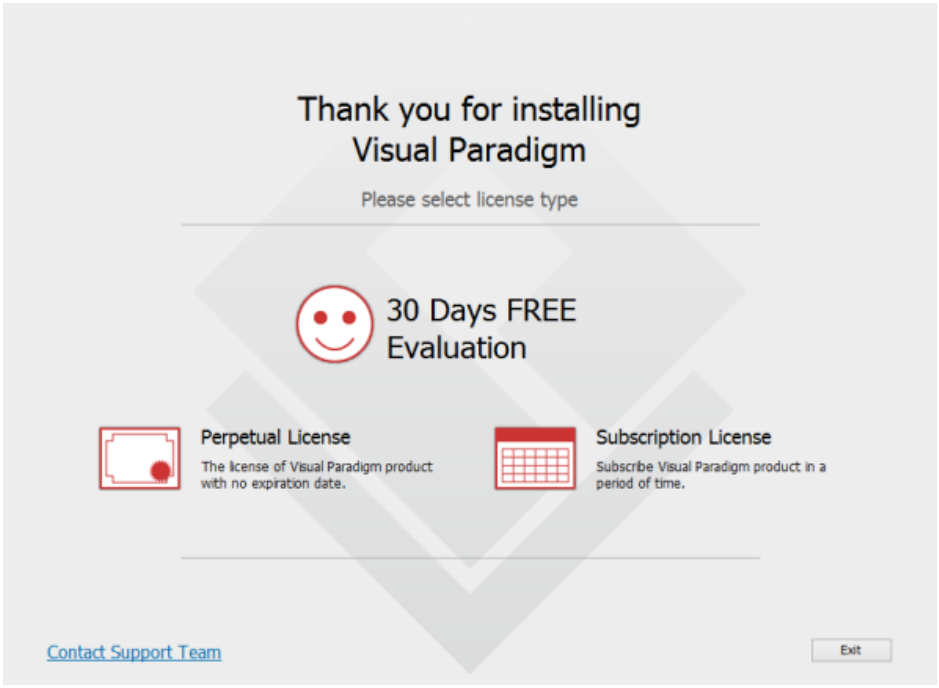
Starting Visual Paradigm

Ways of starting Visual Paradigm

Visual Paradigm can be started through accessing the **Start** menu for Windows 7 or earlier or **Start** screen for Windows 8. For Linux users, Visual Paradigm can be started through the shortcuts in desktop, created by the installer.

Activating Visual Paradigm the first time

When you start Visual Paradigm the first time, you are asked to select a way to activate Visual Paradigm.



Select a way to activate Visual Paradigm

30 Days FREE Evaluation

If you want to evaluate Visual Paradigm, click this. You will then be asked to select the edition of product to evaluate. Visual Paradigm features vary by product editions. For more details on the features supported by different editions, check the Edition Comparison page. Click on the **Evaluate** button to confirm your edition selection. Then, you can start your 30 days evaluation.

To evaluate the Enterprise Edition

Perpetual License

If you want to activate Visual Paradigm with a perpetual license, either a single seat license or a floating license, click **Perpetual License**.

If you hold a single seat license, enter your activation code, your name and email address and click **Activate**. You can obtain the code by visiting your customer account at our Customer Service Center. Alternatively, the licensee should have received our Email notification with activation code included. Note that you can activate Visual Paradigm on both your desktop computer and laptop, provided that you are the only user and at most one instance is started at a time.

If you use a floating license, expand the **Floating License** section, enter the connection settings of the host machine where the license is installed and click **Apply**.

If you use a site license, expand the Site License section, enter your activation code, your name and email address and click **Activate**. Note that the email address has to be under the organizational email domain entered when purchasing.

Subscription License

If you have subscribed to run Visual Paradigm on a time basis or if you are our academic partner, click **Subscription License**. Enter the login details of your customer account and click **Sign in** to continue.

For academic partners, enter the activation code and click **Activate**. Note that Internet connection is required for license verification.

Offline activation

When you try to activate Visual Paradigm without an active Internet connection, you will be prompted for offline activation. You will see a window with a URL in it. To activate Visual Paradigm, copy the URL, pass the URL to a machine that can access the Internet. Visit the URL to obtain a key code. Pass the key code back to the machine that requires activation. Paste the code at the bottom part of the popup window and click **OK** to finish the activation process.

Starting Visual Paradigm (for floating license client whose host is IP-4-enabled)

If you are a floating license client and if your host is IP-4 enabled, you need to start Visual Paradigm with a startup script in order to connect to the server. Here are the steps:

1. Copy **Visual Paradigm.bat** under the scripts folder of Visual Paradigm installation directory to become **Startup.bat**
2. Edit **Startup.bat**
3. Add -Djava.net.preferIPv4Stack=true to the script

```
pushd ..\bin
start ..\jre\bin\javaw -Xms256m -Xmx768m -XX:MaxPermSize=256m -cp ".;..\lib\
vpplatform.jar;..\lib\jniwrap.jar;..\lib\winpack.jar;..\ormlib\orm.jar;..\ormlib\
orm-core.jar;..\lib\xalan.jar;..\lib\lib01.jar;..\lib\lib02.jar;..\lib\lib03.jar;..\
lib\lib04.jar;..\lib\lib05.jar;..\lib\lib06.jar;..\lib\lib07.jar;..\lib\lib08.jar;..
\lib\lib09.jar;..\lib\lib10.jar" -Djava.net.preferIPv4Stack=true RV %*
popd
```

Editing the start up script

4. Save
5. From now on, execute **Startup.bat** to run Visual Paradigm

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Learn more about floating license - how to import, export, return a floating license
- Full Visual Paradigm edition comparison

Evaluate another edition

You may want to try out the different editions of Visual Paradigm during evaluation. To evaluate another edition, you don't need to download or reinstall Visual Paradigm. You just need to perform the "Change License" operation, re-select "Evaluation" and choose the edition to evaluate. Here are the steps:

1. Select **Windows > License Manager...** from the toolbar.
2. Click **Change License...** at bottom left.
3. Click on the smiley face.
4. Click **Evaluate** at the column of edition you want to switch to.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Full comparison of Visual Paradigm editions, from community edition to enterprise edition
- Overview of Visual Paradigm enterprise edition, with features and strength listed
- Overview of Visual Paradigm professional edition, with features and strength listed
- Overview of Visual Paradigm standard edition, with features and strength listed
- Overview of Visual Paradigm modeler edition, with features and strength listed
- Overview of Visual Paradigm community edition, with features and strength listed
- Can you switch to another edition after purchase? Yes, you can. See how to do

Deleting license

You can allow activating Visual Paradigm on both your desktop computer and laptop, provided that you are the only user and at most one instance is started at a time. The maximum allowable installations are three. If you want to de-activate a license, you have to go through the "Delete License" process. By deleting a license, your license will be removed from your machine, meaning that you will not be able to run Visual Paradigm until a valid license is provided again. To delete a license:

1. Open the **License Manager** by selecting **Windows > License Manager...** from the toolbar.
2. Click **Delete License** in the **License Manager** window.
3. Click **Yes** when you are prompted for confirmation. Visual Paradigm will then be exited.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

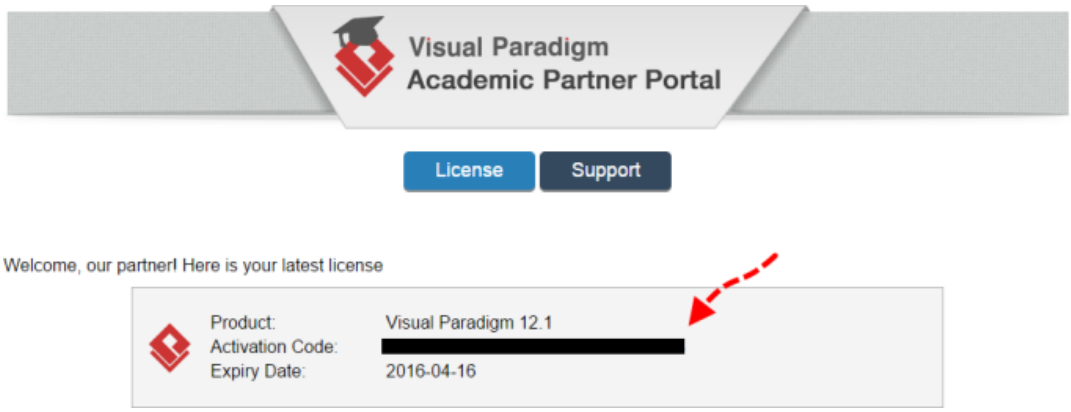
- Learn more about floating license - how to import, export, return a floating license
- Full Visual Paradigm edition comparison

Activating Visual Paradigm with Academic License

If you hold an academic license, please follow the steps below to activate Visual Paradigm.

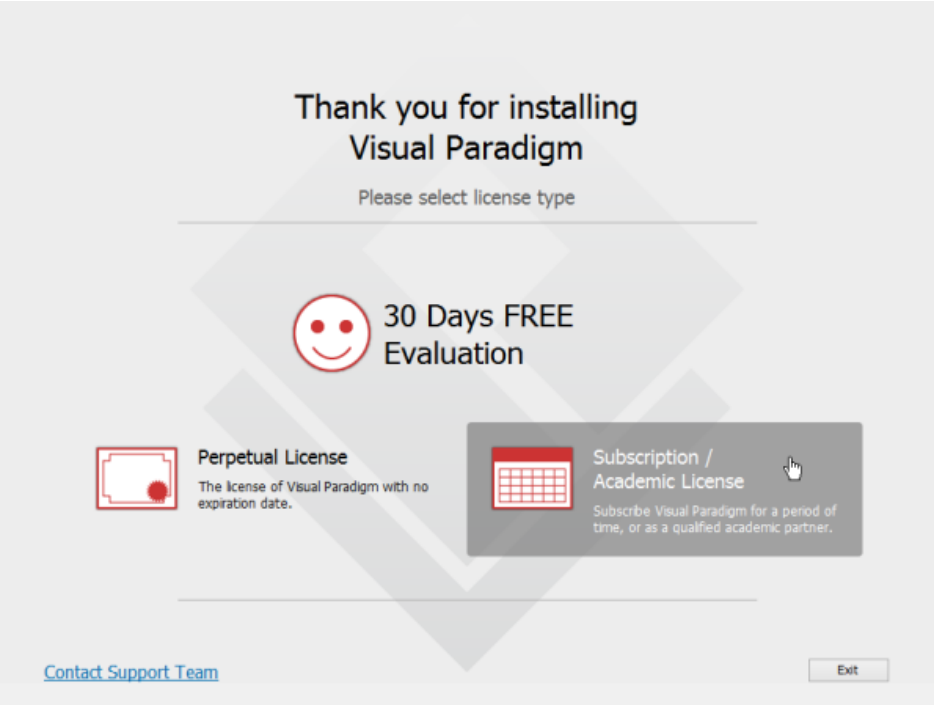
Activating Visual Paradigm

1. Visit the Academic Partner Portal in a web browser and copy the activation code from there. If you are a student and do not know the URL of the portal, please contact your teacher.



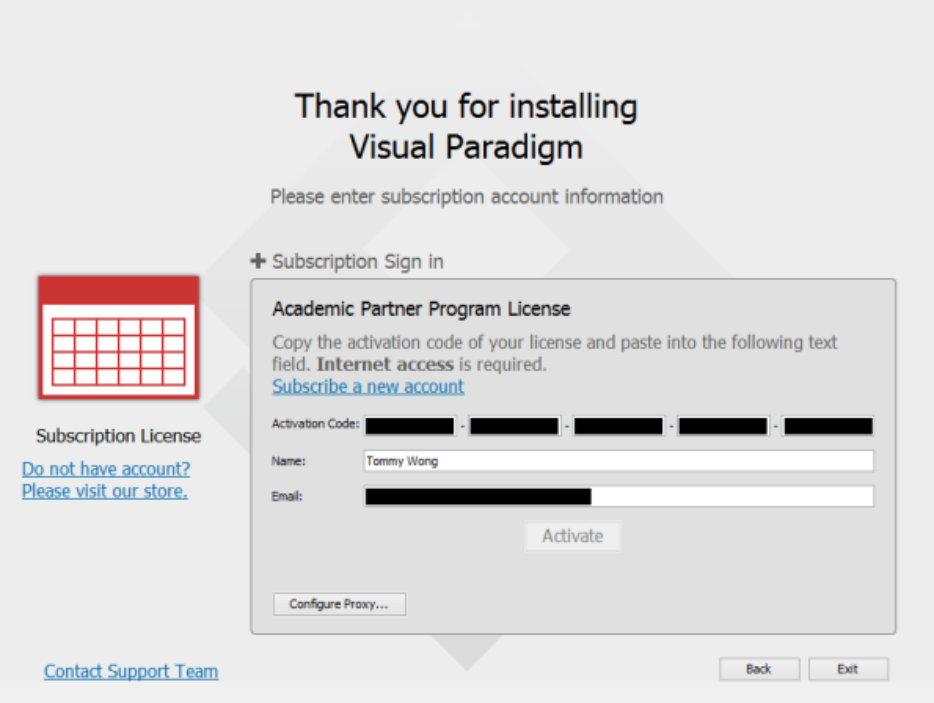
Copy the activation code from Academic Partner Portal

2. Install and start Visual Paradigm.
3. When you see the following screen, select **Subscription/Academic License**.



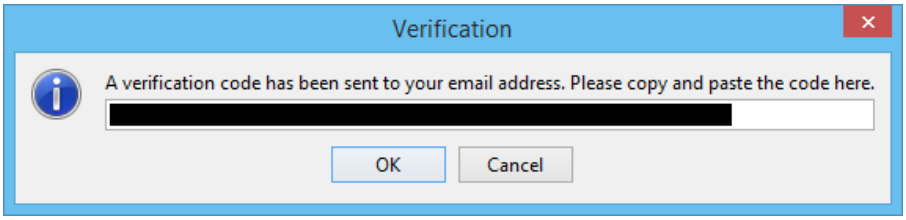
Select Subscription / Academic License

4. Expand **Academic Partner Program License**.
5. Paste the activation code there.
6. Enter your name and email address.



Activating Visual Paradigm

7. Click **Activate**. You are promted for activation code.
8. Check your email for the verification code.
9. Copy the verification code and paste to Visual Paradigm.



Pasting the verification code

10. Click **OK** to confirm. Visual Paradigm will be activated if the verification code is valid.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Learn more about floating license - how to import, export, return a floating license
- Join us now - Visual Paradigm Academic Partner Program
- Full Visual Paradigm edition comparison

Uninstalling Visual Paradigm

This chapter teaches you how to uninstall Visual Paradigm from your system.

Uninstalling Visual Paradigm

List the steps of uninstalling Visual Paradigm.

Uninstalling Visual Paradigm

If you want to remove Visual Paradigm from your system, you can perform an uninstallation.

Uninstalling Visual Paradigm will remove the files in your Visual Paradigm installation from system. If you have installed Visual Paradigm through installer, you can uninstall it by running the **uninstall** file right under the installation directory. If you are using the InstallFree version of Visual Paradigm, which means that the installation was produced by decompressing the zip file that contains the installation, you just need to delete the whole installation folder to have Visual Paradigm removed.

Note that uninstallation does not clear the setting files that are stored under the Home directory.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

User interface

This chapter walks through the various panes and components in user interface.

Interface overview

A summary of the user interface you can see when Visual Paradigm is started.

Main menu

The main menu enables you to access most of the core functions in Visual Paradigm.

Toolbar

The toolbar is a by default horizontal bar below the main menu bar which covers most of the core functions in Visual Paradigm.

Dockable user interface

Dockable user interface refers to the ability to drag out a pane and dock it to another part of the application screen.

Diagram navigator

Diagram navigator is a pane that lets you access and create diagrams.

Model explorer

Model explorer is a pane that lets you access and create models, and browse for their specifications.

Class repository

Class repository is a pane that lets you access and create classes, and browse for their specifications.

Logical view

Logical view is a pane that lets you organize diagrams with user-named views.

ORM pane

ORM pane serves two distinct purposes - to convert domain source code into UML persistable class model, and to convert database schema into entity models.

Property pane

Property pane is a pane that lets you read and edit chosen element(s) 's properties.

Diagram overview

Diagram overview is a thumbnail of diagram which enables you to navigate and zoom into a diagram.

Documentation pane

Documentation pane is a pane that lets you read and edit documentation of chosen element.

Stencil pane

Stencil pane is a pane that list stencil, and lets you drag out a shape to diagram.

Diagram specification

Diagram specification enables you to adjust some of the diagram settings

Perspective

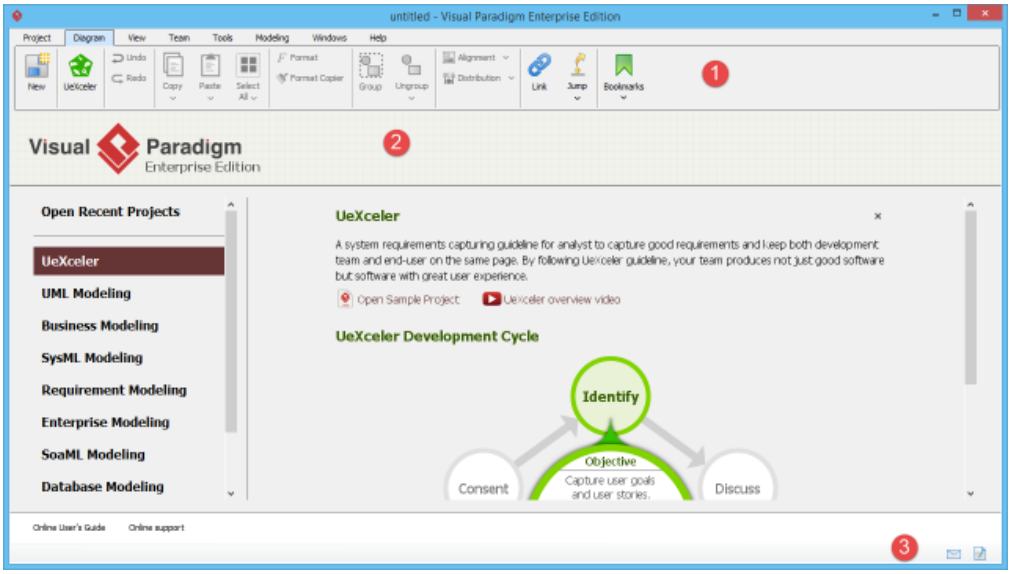
Perspective define way to position panes.

Model element specification

Properties of model elements can be set and read through the specification dialog box.

Interface overview

Visual Paradigm 's user interface comprises a toolbar, a diagram editor and a status bar.



User Interface of Visual Paradigm

No.	Name	Description
1	Toolbar	A tabbed toolbar that allows you to perform various operations in Visual Paradigm.
2	Diagram editor	The diagram will be displayed in diagram editor.
3	Message pane	Notifications are shown here. You can also open the message pane and description pane from the bottom right of the status bar.

Description of user interface

Related Resources

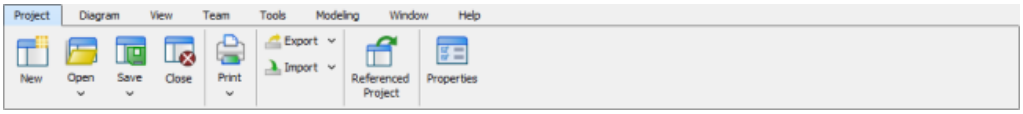
The following resources may help to you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Toolbar in Visual Paradigm

The toolbar appears at the top of the Visual Paradigm application window. It is a collection of commnoly used buttons and tools, categorized into several function tabs. The toolbar is shown by default, but you can collapse it by double clicking on any tab.

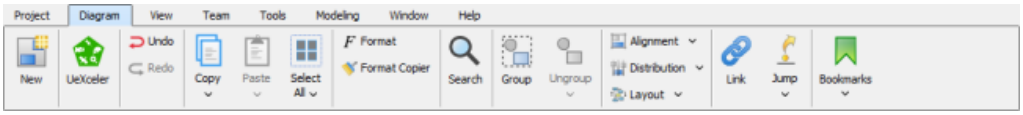
The Project tab



The Project tab

- New - Create a project
- Open - Open an existing project file either by selecting one from file chooser or from the list of recent opened project
- Save - Save the opening project
- Close - Close the opening project
- Print - Open the Print tool to configure the printing and print
- Export - Export the project to other supported formats
- Import - Import project data from an external source
- Referenced Projects - Add or remove referenced project
- Properties - Edit the basic project properties like name, author and description

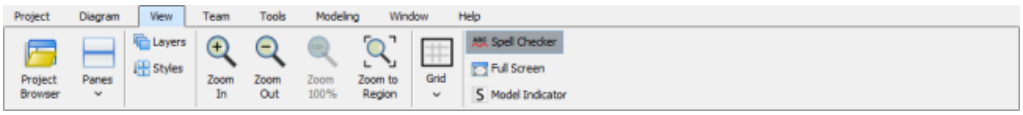
The Diagram tab



The Diagram tab

- New - Create a diagram
- UeXceler - Open the UeXceler tool, which provides the necessary toolset for requirements management with the use of use case and user story
- Undo - Roll back undesired changes
- Redo - Rerun an undone action
- Copy - Copy the selected shape(s)
- Paste - Paste the selected shape(s) to the active diagram
- Select All - Select everything on the active diagram
- Format - Open the Format tool for configuring the formatting properties of the selected (shapes). These properties include the color and line style of shape/connector.
- Format Copier - Click once to copy the format of the select shape so that you can apply the format to another shape by clicking on that shape. Double click to lock Format Copier so that you can paste the formatting to multiple shapes
- Search - Toggle the search bar for searching elements in active diagram.
- Group - Group the selected shapes
- Ungroup - Break a group
- Alignment - Align the selected shapes based on their top, bottom, left or right, or to set their width and/or height to be the same
- Distribution - Distribute the selected shapes evenly based on various criteria
- Layout - Quick rearrangement of shapes on the opening diagram
- Link - Toggle the Link box for you to paste a model element URL there and then visit the URL
- Jump - Instantly open a model element or diagram by providing its name, or part of its name
- Bookmarks - Bookmark a shape for quick accessing, or to manage the previously added bookmarks

The View tab



The View tab

- Project Browser - Opens the Project Browser for browsing and accessing the diagrams, model elements and model structure of the opening project.
- Panes - Several panes are available for accessing different kinds of project data. For instance, the Model Explorer can be used to view model hierarchy, while the Diagram Navigator can be used to find diagrams in ease. You can open a pane here.

- Layers - Create multiple diagram layers to better categorize the different kinds of shapes
- Styles - Define the style. Each style carries a set of formatting properties. You can assign a style to a shape to apply those properties to that shape
- Zoom In - Increase the magnification of the active diagram
- Zoom Out - Reduce the magnification of the active diagram
- Zoom 100% - Restore the zoom ratio to 100%, which means, no zooming
- Zoom to Region - Zoom the diagram to a specific region as set by you
- Grid - Show, hide and configure the appearance of grid lines on diagram
- Spell Checker - Enable or disable the spell checking feature, which is capable in verifying the correctness of shape names and descriptions and when a problem found, underline it with a red line
- Full Screen - Maximize the diagram editor by collapsing the toolbar and hiding the opened panes, if any
- Model Indicator - Show a small icon on shape body for each these situations: that shape has description filled, has a sub-diagram, has references, is an auxiliary view or is came from a referenced project

The Team tab

The Team tab

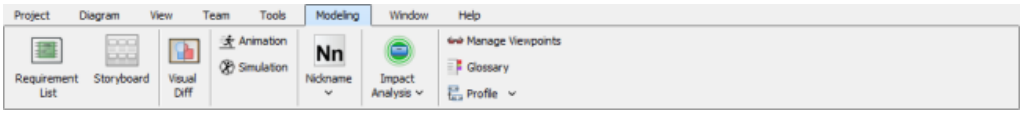
- Login - Login to a version controlling system like VPository and Teamwork Server
- Commit - Upload the local modifications of your project to server
- Update - Grab the modifications made by your teammates from server to local
- Teamwork Files - Togger the Teamwork Files pane for accessing the files that are kept with this project
- Tasifier - Open Tasifier, a task management tool, in Visual Paradigm
- PostMania - Read the posts made by teammates
- Visual History - Check out and restore the old revisions of diagrams with the Visual History tool
- Utilities - Perform operations like branching, tagging, exporting a specific revision, etc

The Tools tab

The Tools tab

- Doc. Composer - Produce a document with Doc. Composer, or access its management features.
- Publish Project - Produce web contents from project using Project Publisher.
- Doc - Produce HTML/PDF/Word document from project.
- DB - Generate database from ERD, reverse engineer ERD from database
- Hibernate - Generate Hibernate ORM code for database application development
- Code - Generate code from class diagram, reverse engineer UML from code. A number of programming languages are supported

The Modeling tab

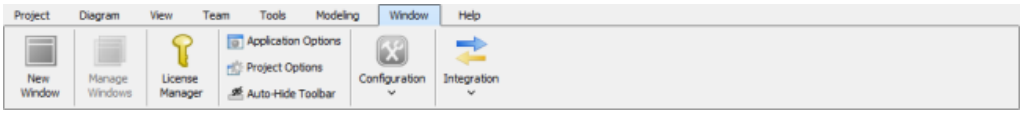


The Modeling tab

- Requirement List - Open a list of SysML requirements elements. You can view, edit and create requirements there
- Storyboard - Perform storyboarding by creating scenarios of wireframes
- Visual Diff - Compare two diagrams and know their differences
- Animation - Animate the active diagram. Only available when opening an Activity Diagram, Sequence Diagram or Business Process Diagram
- Simulation - Open the Simulation panel when the active diagram is a Business Process Diagram
- Nickname - The nickname feature allows you to define multiple name and description set for your model. You can configure and switch between nicknames here
- Impact Analysis - Create matrix or chart diagram
- Manage Viewpoints - Manage the ArchiMate viewpoint and stakeholders

- Glossary - Define project vocabularies with the use of the Glossary tool
- Profile - Create UML profile

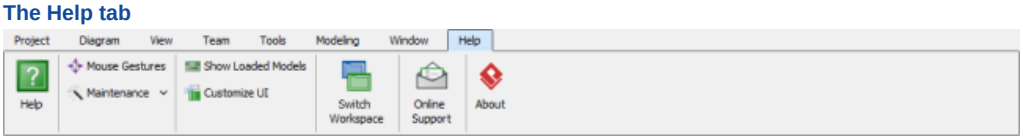
The Window tab



The Window tab

- New Window - Open a new application window
- Manage Window - Open another application window, or close non-used window(s)
- License Manager - Delete or replace your license
- Application Options - Configure Visual Paradigm application options
- Project Options - Configure the options for the opening project
- Auto-Hide Toolbar - Make the toolbar automatically collapsed when you are editing diagram
- Configuration - Configure stereotypes, requirements, project management look-ups, user story tags, etc
- Integration - Install the integration with IDEs such as Eclipse, NetBeans, IntelliJ IDEA and Visual Studio

The Help tab



The Help tab

- Help - Open the Help contents of Visual Paradigm
- Mouse Gestures - Check out the various kinds of supported mouse gesture
- Maintenance - Utility features for maintaining or repairing your project
- Show Loaded models - View the model elements loaded
- Customize UI - Hide away non-used UI components
- Switch Workspace - Restart Visual Paradigm in another workspace
- Online Support - Visit the Visual Paradigm Support page online to submit your support request
- About - Show the About window

Related Resources

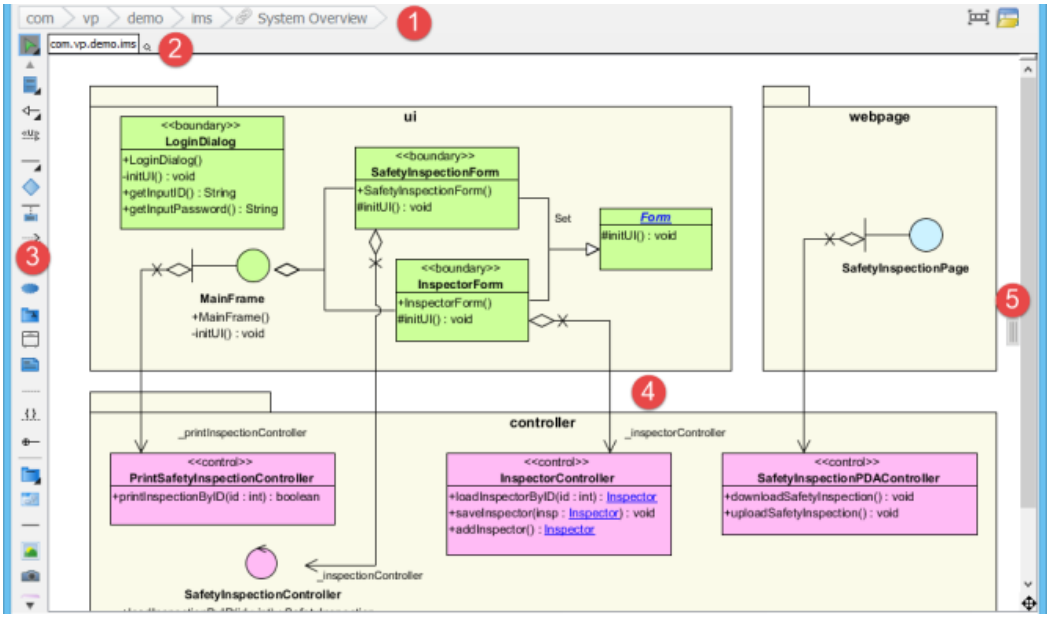
The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram editor

Diagram editor is where you can view and edit your diagram. In this page you will learn how to operate with the diagram editor.

Overview of diagram editor



Overview of diagram editor

Location	Name	Description
1	Navigation bar	Navigation bar consists of two parts. The left hand side contains a breadcrumb that shows the diagram name and the location of the diagram. The right hand side contains two buttons. One for switching to another diagram, another for opening the Project Browser.
2	Package header	The name of the package where the diagram resides. Package header is available only in some of the diagrams, but not all. You can double click on it to edit it. By entering the name of another package, the diagram will be moved to that package accordingly. You can click here to learn more about package header.
3	Diagram toolbar	The diagram toolbar provides you with the tools that you need to draw a diagram. Two kinds of tools can be found in the toolbar: • Diagramming utilities like the pan tool, sweeper and magnet tool. • Diagram -specific tools like actor in Use Case Diagram, BPMN pool in Business Process Diagram. For details about how to create a diagram with the toolbar, click here.
4	Diagram	The diagram to edit.
5	Action bar	Click to toggle the action bar that provides you with access to the tools based on your selection. For instance, if you have selected a Business Process Diagram, you can access tools like Working Procedure, Animation. If you have selected a use case, you can access tools like Use Case Details.

Description of diagram editor

The breadcrumb

The breadcrumb displays both the name and location of the opening diagram. Location means the place where the diagram resides within the model hierarchy. For example, if diagram "Bar" is put under model "Foo", you will see the breadcrumb structure *Foo* > *Bar*, where *Foo* is the name of the model and *Bar* is the name of the opening diagram. Similarly, if the diagram is placed in a model and that model is placed inside another model, you will see the breadcrumb structure *${name-of-grandparent}* > *${name-of-parent}* > *${name-of-diagram}*.

The segments of a breadcrumb is clickable. You can click directly on the name in a segment to open up the corresponding shape, model or model element.

A breadcrumb that shows the diagram name (i.e. System Overview) and the package where it resides (i.e. com.vp.demo.ims)

Using link

You may want to draw your teammates' attention to a specific diagram to discuss it. Instead of asking your teammates to read diagram "XYZ", you can simply share the link of that diagram with them.

A link is a unique address of an element in a project. It is a general element accessing mechanism available in Visual Paradigm. You can share a link with teammate, so to let him/her to locate a diagram, a shape or a model element quickly and accurately.

Obtaining the link of the opening diagram

You can obtain the link of the opening diagram from the breadcrumb. Next to the diagram name in the breadcrumb, there is a link icon. Click on it.

Clicking on the link icon

This opens the link box, with the link of the diagram in it. You can now copy the link by pressing **Ctrl-C**.

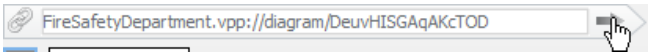
Link of the opening diagram

Visiting a link

Once you are given a link, no matter it's a diagram link, model element link or shape link, you can open the link in the diagram breadcrumb. Next to the diagram name in the breadcrumb, there is a link icon. Click on it.

Clicking on the link icon

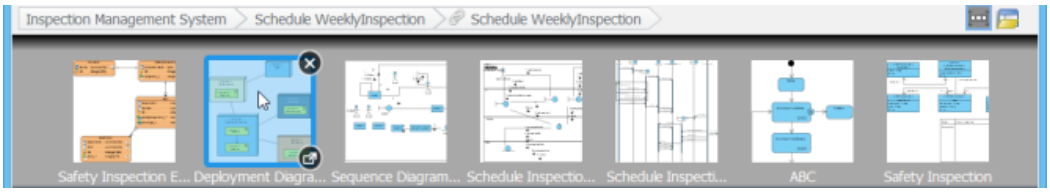
This opens the link box. Paste the link into the link box and click on the button on the right hand side to open it.



Open a link

Switching to another diagram

The diagram switch provides you quick access to diagrams opened within the active session. To switch to another diagram, click the **Switch Diagram** button on the right hand side of the navigation bar. Then, double click on the thumbnail of the desired diagram to open it.



Switch to another diagram

You can also press **Ctrl-Tab** in an active diagram to toggle this pane.

Opening Project Browser

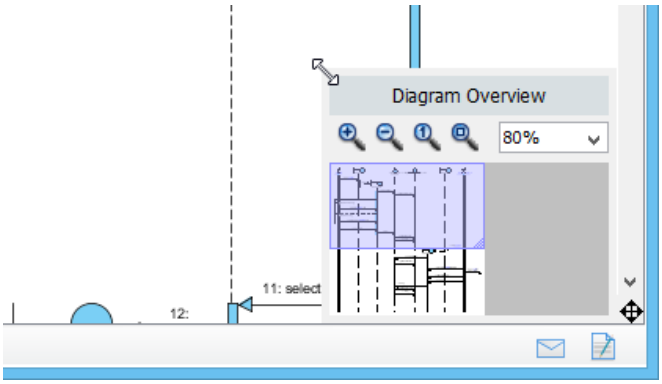
The **Project Browser** incorporates several 'views' of the project, such as the **Diagrams** view, **Model Structure** view, **My Recent** and **Team Recent**. These views provide different ways for realizing a project's model hierarchy as well as to find out the desired diagrams. The **Project Browser** supports view-based finding and project-level searching, which makes searching of project data much easier.

You can open the **Project Browser** by clicking on **Open Project Browser** on the right hand side of the navigation bar. Click here if you want to know more about the Project Browser.

Open Project Browser

Diagram overview

Diagram Overview is a pane where you can view and zoom in the opening diagram directly and quickly. You can open it only when the diagram size exceeds the viewing region. When that happens, you will see a button appear at the bottom right corner of the diagram. You can click on it to open the **Diagram Overview**.



Resizing the Diagram Overview

Having a quick view on a particular part of diagram

There is much truth in saying that viewing a large diagram is such an annoying task, especially a particular part of this large diagram is needed to focus on. In fact, a particular part of diagram can be navigated by moving the purple rectangle in **Diagram Overview** which represents the visible area of diagram in diagram overview.

Zooming in a particular part of diagram

Drag the diagonal of purple rectangle to zoom in a particular part of diagram. The smaller you drag the purple rectangle, the more the part of diagram will be magnified.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

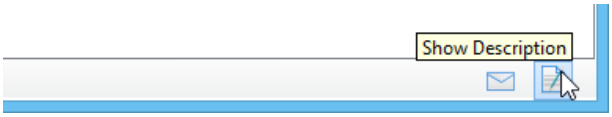
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Description Pane

Description Pane enables you to document project data such as model elements, shapes or diagrams either in written or verbal form. For written content, it can be a plain text or HTML text with formattings like bold, italic, font color, etc.

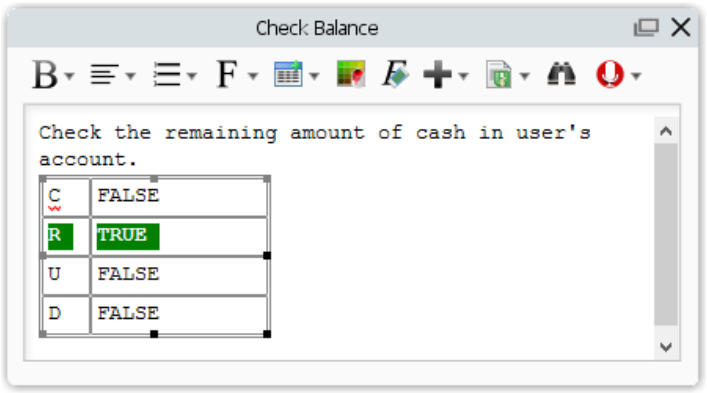
Opening and closing the Description Pane

Description Pane can be opened by clicking on Description button at the bottom right corner of the status bar. To close it, either click the same button again or press the **X** button at the top right corner of the **Description Pane**.



Show Description Pane

Overview of Description Pane



Description Pane

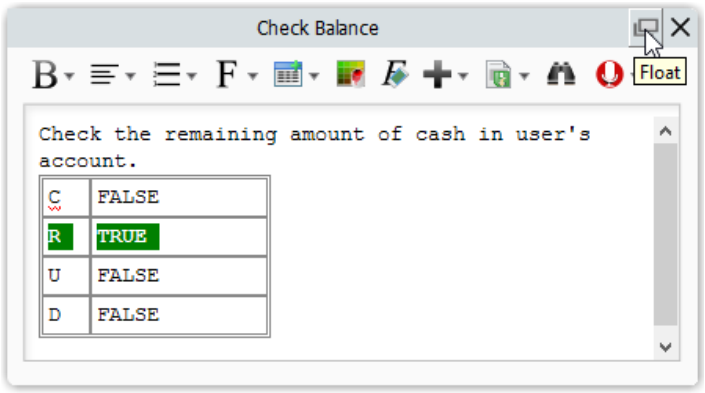
Button	Name	Description
B	Bold	Bold: Set the highlighted text to bold. Italic: Set the highlighted text to italic. Underline: Underline the highlighted text.
	Alignments	Set the alignment of highlighted text to the left, the center or the right.
	Ordered list	Ordered list: Add a numbered list. Un-ordered list: Add a list with bullet points.
F	Font	Font: Select the font family of highlighted text. Font size: Select the size of highlighted text. Font color: Select the color of highlighted text.
	Table:	Add a table.
	Background color	Select the background color of highlighted text.
	Clear Formats	Clear formats of the whole editor to convert the content to plain text.
	Add	Add Link: Add a hyperlink into description. Add Image: Add an image into description. Add Model Element: Add a model element link into description. Add Property Value: Add a property value into description. Add Diagram: Add a diagram link into description.
	Template	Save as Template: Save the current description as a template. Manage Template: Delete a template or set a template as default.
	Find	Search for text in the description.
	Record	Record: Record voice description.

> **NOTE:** The formatting toolbar is by default hidden for description pane shown in specification window. In order to show the formatting toolbar, simply click on the editor.

Floating mode

The **Description Pane** has a fixed position and size by default, which means that you cannot move it nor to resize it. However, you can change it to **Floating** mode so that the **Description Pane** can be moved and resized. You can even move it out of the application window.

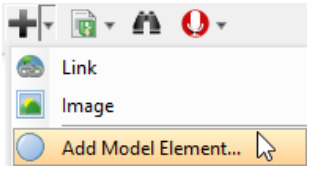
To change the **Description Pane** to **Floating** mode, click **Float** in the title bar of the **Description Pane**. Or you can simply drag the title bar of the pane to change it to Floating mode.



Change the Description Pane to Floating mode

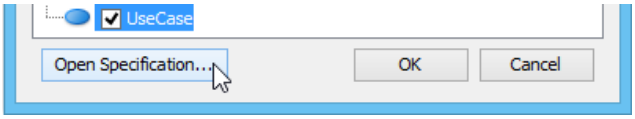
Adding model element link

1. Click **Add Model Element** button on the editor's toolbar after decided a place for inserting a model element.



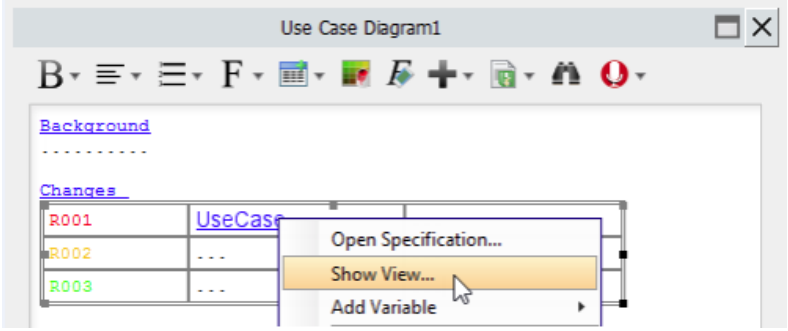
Click Add Model Element button

2. In **Select Model Element** window, select an existing model element on the list. If you want to modify the selected model element, you can click **Open Specification...** button.



Click Add Model Element button

3. Finally, click **OK** button to confirm.
4. Consequently, the name of inserted model element will be shown on the description pane with underline. If you want to preview the inserted model element, you can right click on its name and select **Show View...** from the pop-up menu. After the **Show View** window pops out, you can preview it in the **Preview** window. If you want to view the actual model element on the diagram, you can click **Go to View** button.



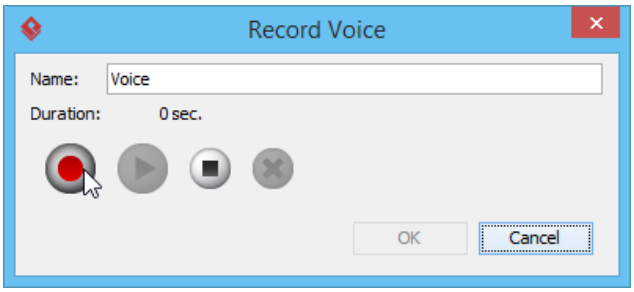
Click Show View... from the pop-up menu

Voice description

In addition to textual description for your model elements, you can record voice description or embbed audio files.

Recording voice

1. On top of the **Description Pane**, click on .
2. In the **Record Voice** window, click the **Record** button to start recording.



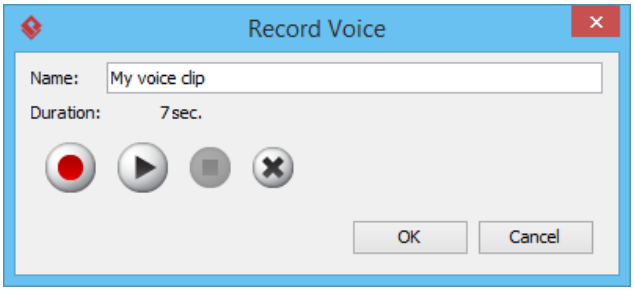
Start recording

> **NOTE:** Make sure your audio input device is active before operate voice description.

3. Click the **Stop** button when you want to end the recording.

> **NOTE:** Play the recorded voice by pressing the **Play** button; record again by pressing the **Clear** button and rerun the previous steps.

4. Enter the name for recorded voice clip in the text field of **Name**.

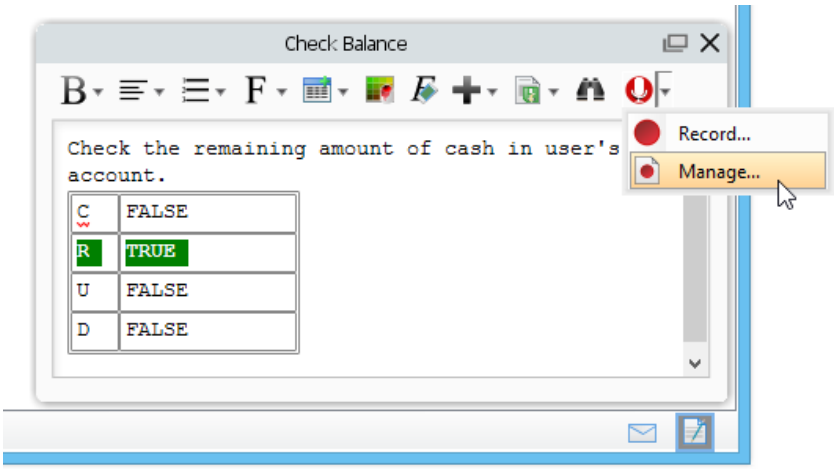


Name voice clip

5. Click **OK** button to confirm recording.

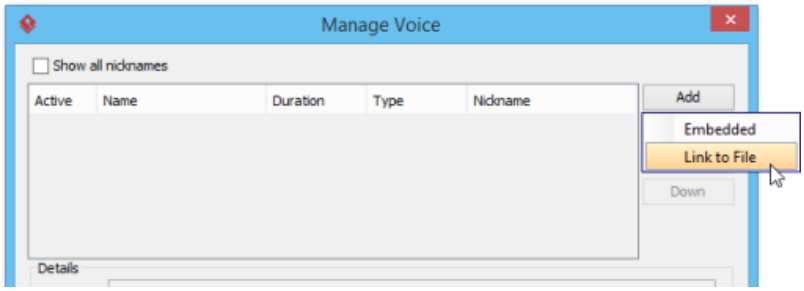
Managing audio clips

1. On top of the **Description Pane**, click on the down arrow next to .Then, select **Manage...** from the drop down menu.



*Select the **Manage...** menu*

2. In the **Manage Voice** window, click the **Add** button, and choose either **Embedded** or **Link to File**.



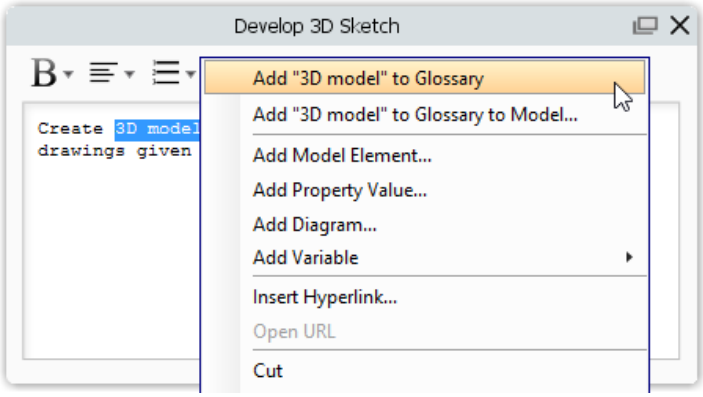
Try to add a link to an audio file

3. If you choose **Embedded**, record a voice clip when the **Record Voice** window pops out; if you choose **Link to File**, select an audio file when the **Open** dialog box pops out.
4. The voice clip can be renamed in the text field of **Name**.
5. You can also select an added audio clip and click **Delete** to remove it.
6. Click **OK** button to confirm the changes.

Defining a glossary item

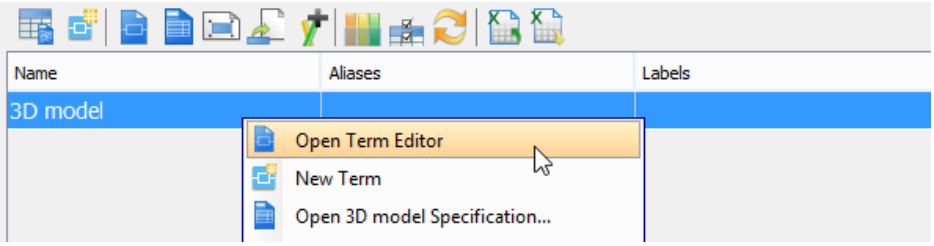
A word or a lexis can be defined as a glossary item for explication.

1. Highlight the word or the lexis you would like to be defined and then right click on it. Select **Add "[highlighted term]" to Glossary** from the pop-up menu to switch to **Glossary Grid**.



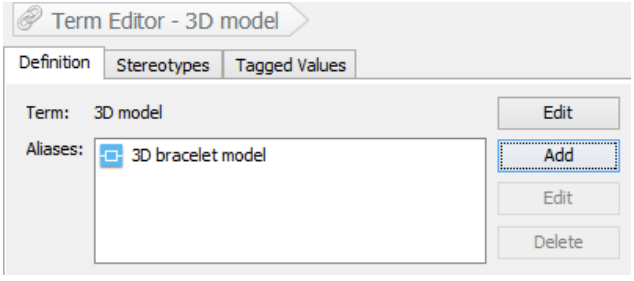
*Select **Add "premium" to Glossary** from the pop-up menu*

2. In **Glossary Grid**, click **Open Term Editor** from the pop-up menu in order to fill more details about the new item. Alternatively, right click on the term and select **Open Term Editor** from the pop-up menu.



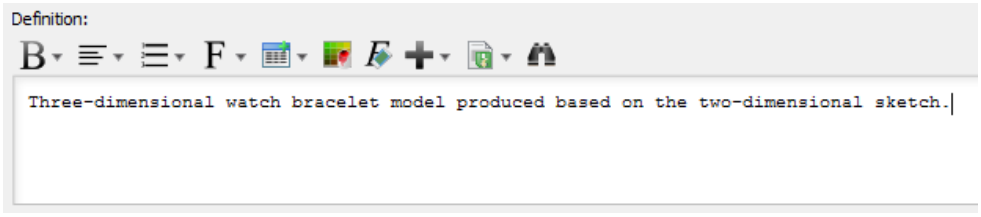
*Click **Open Term Editor** from the pop-up menu*

3. In **Term Editor** page, click **Add** button to type the alias(es) for the new item.



Type an alias for the new item

4. Further information about the new item can be given by typing in the space under **Definition**.



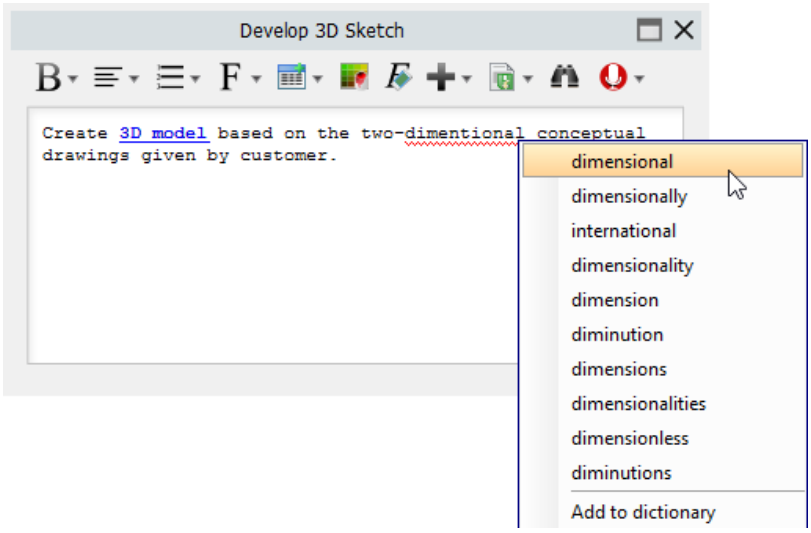
Enter term definition

NOTE: In Glossary Grid, you can jump to the source from which a term was defined by right clicking on the term and selecting **Transit From > %SOURCE_ELEMENT_NAME%** from the popup menu.

Spell checking

When you type an incorrect word carelessly, **Description Pane** can offer you a help.

For correction, right click on the incorrect word with a red curved line and select one out of the suggested words from the pop-up menu.



Select a correct word from the pop-up menu

Moreover, you can add a new word to the dictionary if the word you typed is a rare word or a new created word. Right click the new word and select **Add to dictionary** from the pop-up menu. When you type the word next time, it won't be marked as an incorrect word again.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Message Pane

The **Message Pane** appears at the bottom of the application window. It reports events that occur when using Visual Paradigm. Here are some cases in which you will receive a message:

- Failed to generate code from project
- Someone has created a post in PostMania to a diagram you followed
- Refactoring failed

Message Pane with a message in it

The **Message Pane** is hidden by default. You can turn it on by clicking on the **Show Message** button at the bottom right corner of the status bar. Likewise, you can turn it off by clicking the same button.

Messages are accumulated over time, untill you close the application or clear them manually. To clear message(s), right click on the **Message Pane** and select **Remove Selected Logs** or **Clear Logs** from the popup menu.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Project Browser

The **Project Browser** provides you with different perpectives in viewing your opening project. With **Project Browser**, you can view and open the diagrams in your project, browse model structure, check and open the recently modified diagram, etc.

Opening the Project Browser

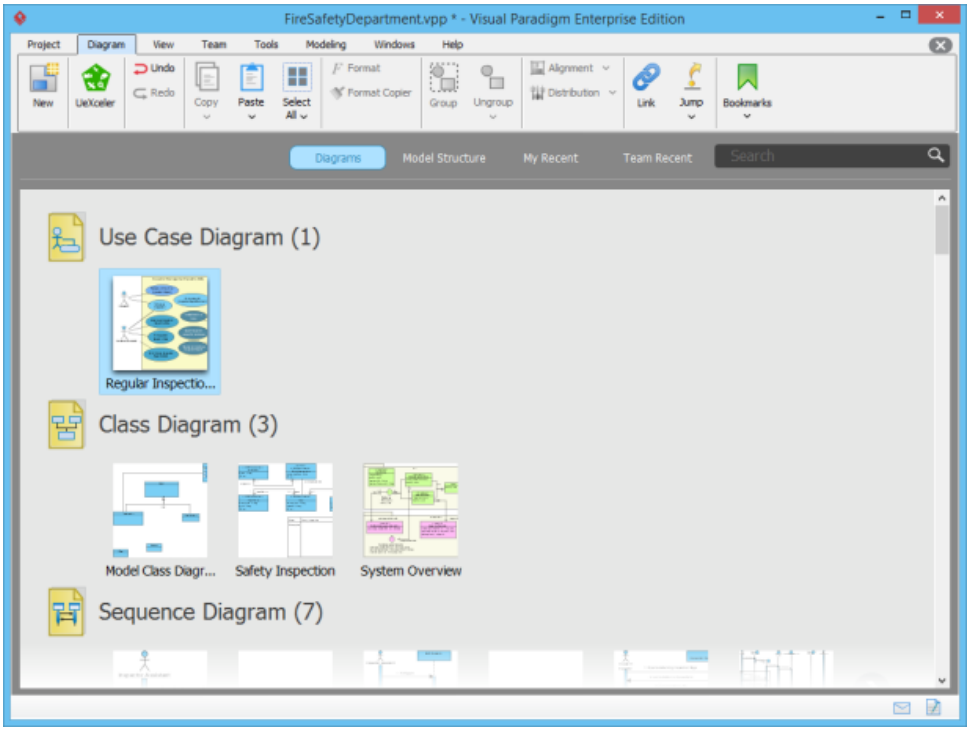
There are two ways you can take to open the **Project Browser**:

- Select **View > Project Browser** from the toolbar
- Click on the **Open Project Browser** on the right hand side of the navigation bar of any diagram

Open Project Browser from navigation bar

Diagrams view

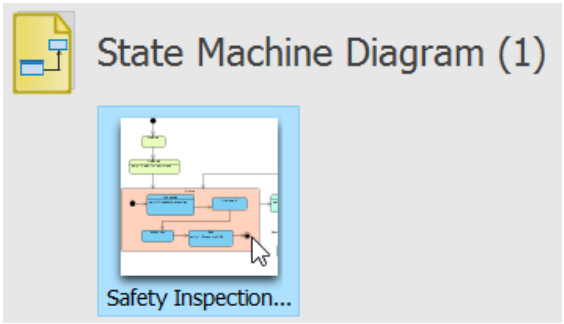
The **Diagrams** view enables you to browse the diagrams in the opening project. Diagrams are grouped by the categories they belong to. You can scroll up and down to view the thumbnails of diagrams.



Diagrams view of Project Browser

Opening a diagram

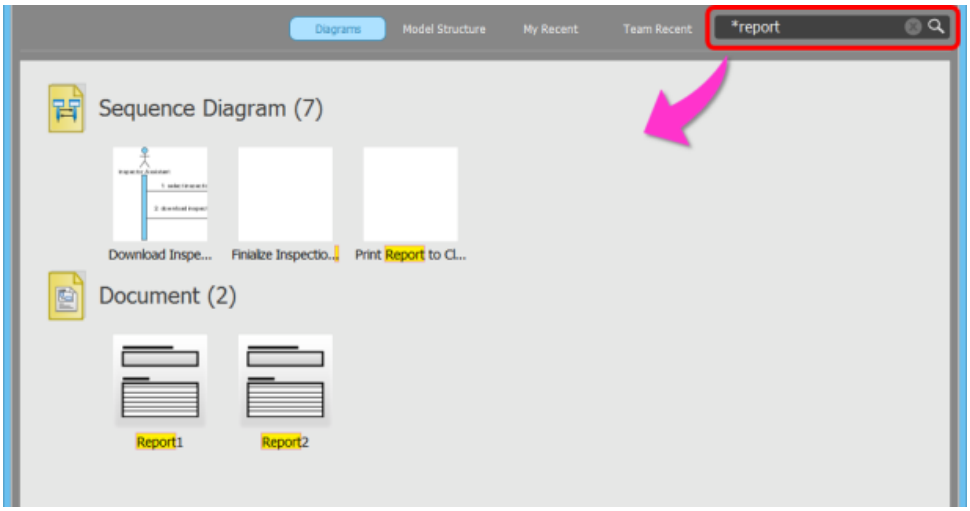
To open a diagram, double click on its thumbnail.



Opening a State Machine Diagram

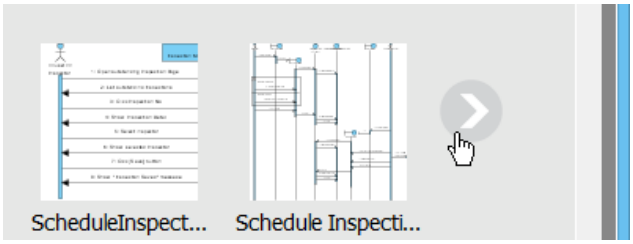
Filtering diagrams

if you want to locate a diagram with its name, or part of its name, enter the search text in the **Search** field. The **Diagrams** view will be updated to list only the diagrams that match with your search string. Note that you can use *, the wildcard character in filtering.



Filtering in Project Browser

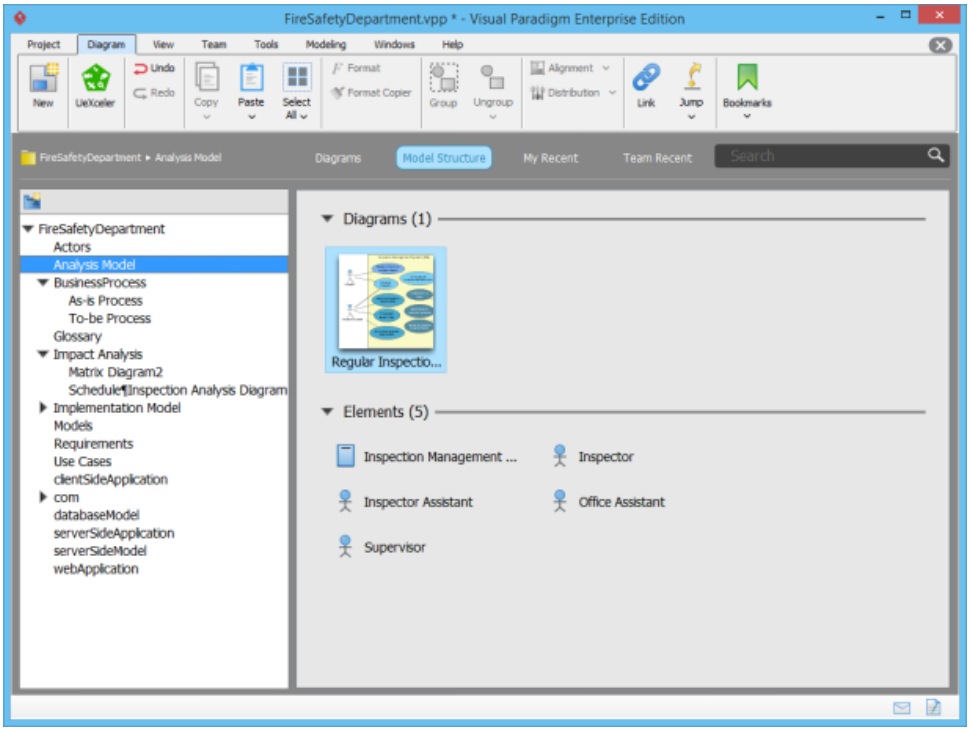
Navigating through diagrams
By moving your mouse pointer over the forward and backward button, you can view the remaining diagrams of a diagram category, if any.



Navigating through diagrams

Model Structure view

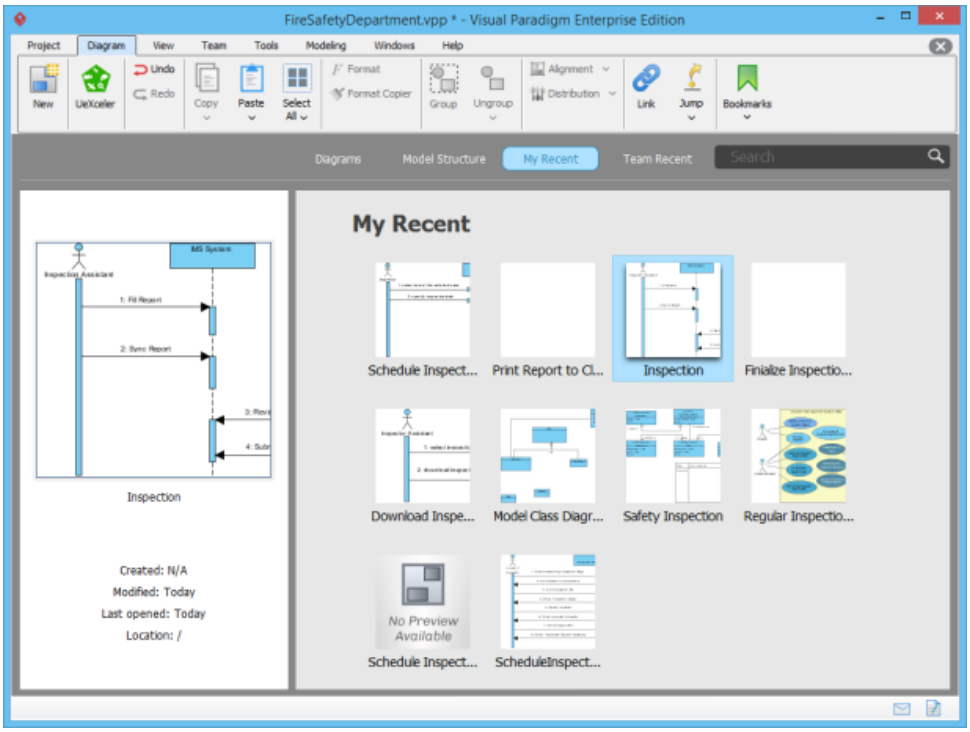
The **Model Structure** view enables you to browse the model structure and to access the diagrams and model elements in those models. It consists of three main parts. On the left hand side, there is a list that comprises of package and model of the opening project. On top of the list, there is a breadcrumb that shows the location of the selection package/model. When you select a package or model in the list, the right hand side of the screen lists the containing diagrams and model elements. You can double click on a diagram thumbnail to open a diagram or double click on a model element to open its specification.



Model Structure view of Project Browser

My Recent

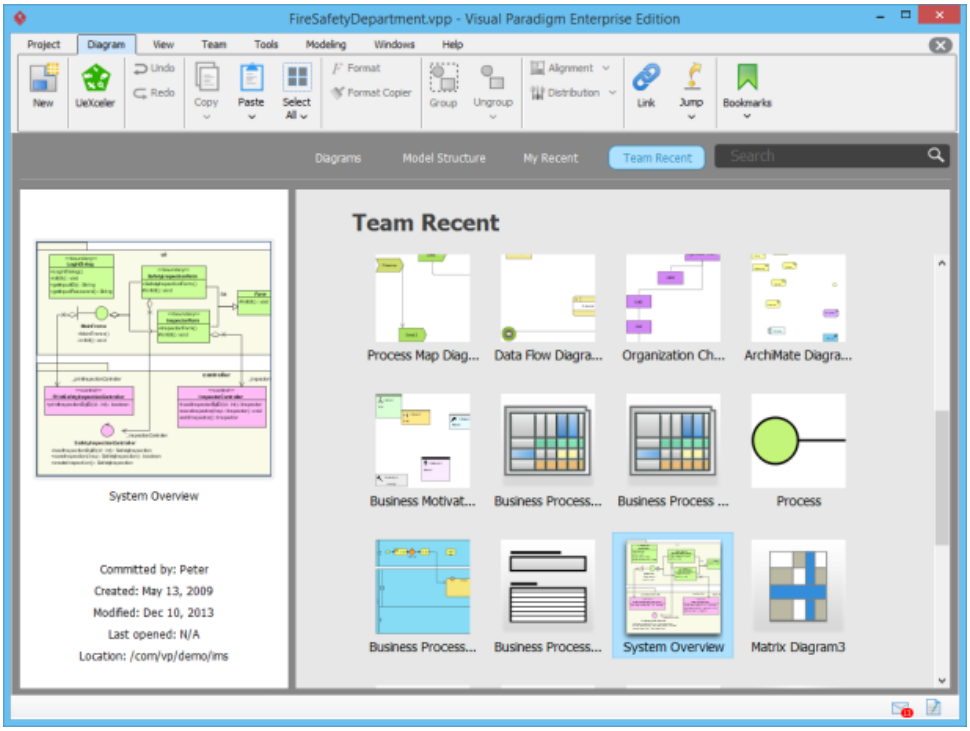
My Recent lists the diagrams you opened recently, except the one currently opened. You can open a diagram by double clicking on its thumbnail.



My Recent view of Project Browser

Team Recent

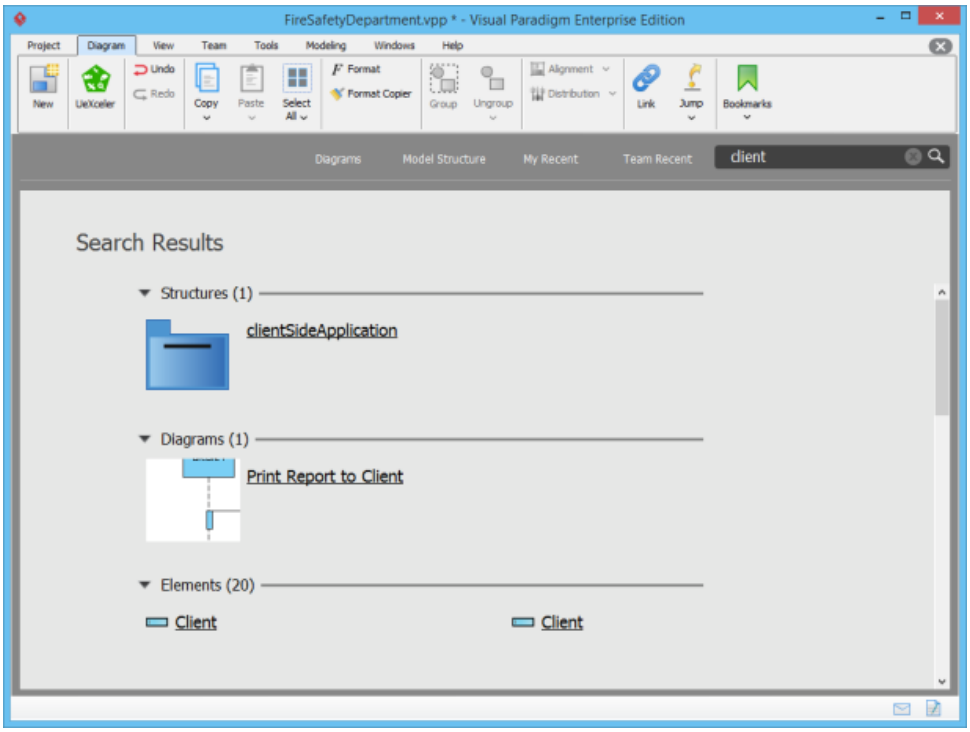
Team Recent lists the diagrams recently modified and committed by the team. You can open a diagram by double clicking on its thumbnail.



Team Recent view of Project Browser

Searching

The search feature allows you to find model elements, views and diagrams quickly. You can perform a searching by entering the search string in the **Search** field and then press **Enter**. When finished searching, search result will be displayed in **Project Browser** and you can open a diagram by double clicking on its thumbnail or view a model element's specification by double clicking on it directly.



Searching in Project Browser

Related Resources

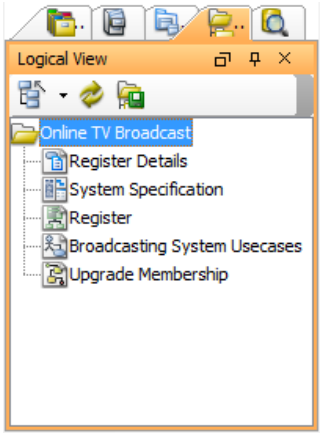
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Logical View

The logical view provides a hierarchical view of a project's structure. With the logical view, users can create and customize the diagrams in their project with meaningful categorization by adding domain specific view(s).

In addition, users can customize a default logical view for their preference, rather than re-creating a new logical view for every new project. The logical view can be exported to xml files which can be used in other projects or distributed among the development team. Different views, thereby, can be merged automatically through the Teamwork Server .



*The **Logical View***

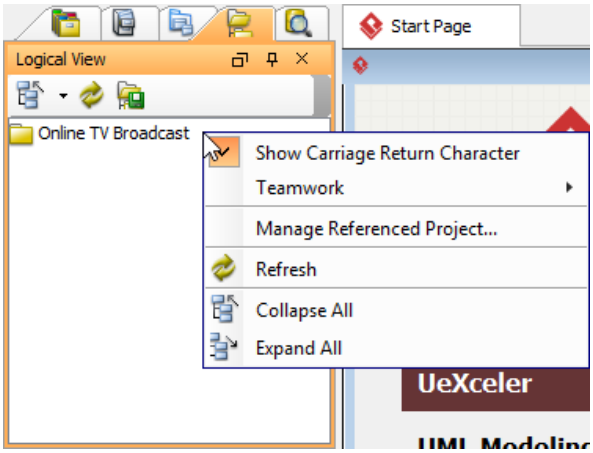
The toolbar

Name	Icon	Description
Collapse		To collapse the selected diagram.
Expand		To expand the selected diagram.
Refresh		To update the content of logical view.
Set Logical View Structure as Default		To set default structure for logical view in all projects.

The description of icons on Logical view

Pop-up menu

Pop-up menu of logical view

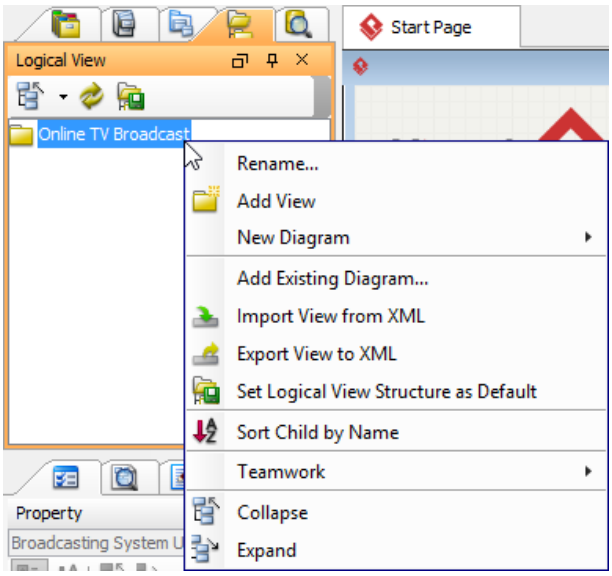


The pop-up menu of Logical View

Menu Title	Description
Show Carriage Return Character	Display line breaks of multi-lined diagram name as carriage return character.
Teamwork	Perform teamwork activities.
Manage Dependent Project...	Add or remove dependent project.
Refresh	Refresh Logical View content.

Collapse All	Collapse all tree nodes.
Expand All	Expand all tree nodes.

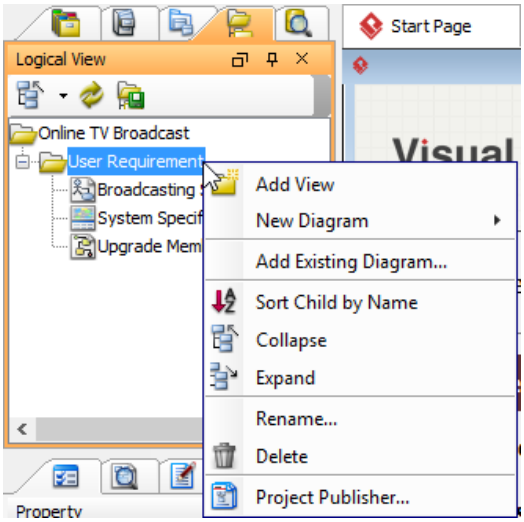
Pop-up menu of project



The pop-up menu of project node in Logical View

Menu Title	Description
Rename...	Rename the project.
Add View	Add a view under project.
New Diagram	Create a diagram under root view.
Add Existing Diagram...	Add an existing diagram under root view.
Import View from XML	Import logical view configuration file.
Export View to XML	Export logical view as configuration file.
Set Logical View Structure as Default	Set the current view structure as default so that another project that will be created under the same workspace will share the same structure.
Sort Child by Name	Sort the views by name.
Teamwork	Perform teamwork activities.
Collapse All	Collapse the project node.
Expand All	Expand the project node.

Pop-up menu of view



The pop-up menu of view in Logical View

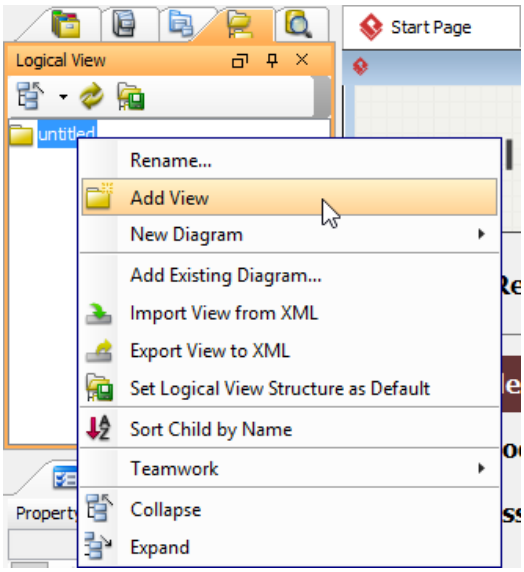
Menu Title	Description
Add View	Add a child view under the selected view.
New Diagram	Create a diagram under the selected view.
Add Existing Diagram...	Add an existing diagram under the selected view.
Sort Child by Name	Sort the views/diagrams by name.
Collapse	Collapse the selected view node.
Expand	Expand the selected view node.
Rename...	Rename the selected view.
Delete	Delete the selected view.

Closing and opening the Logical view

Logical view is opened by default. To close it, press the **X** button at the top right corner. On the other hand, it can be opened by selecting **View > Panes > Logical View** from the main menu.

Creating a new view node

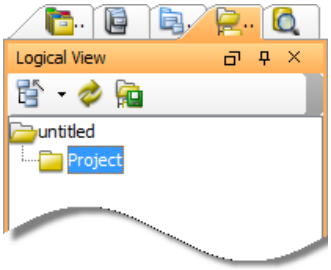
Right-click a root node on **Logical View** and select **Add View** from the pop-up menu.



*Click **Add View** from the pop-up menu*

You can enter the name for the new view node in the **Input** dialog box and then click **OK** button to confirm editing and close the dialog box.

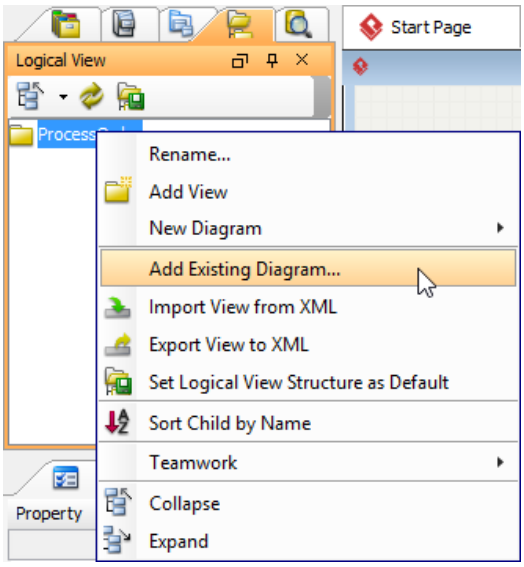
A new view node is, therefore, created under the chosen node.



Created new view node

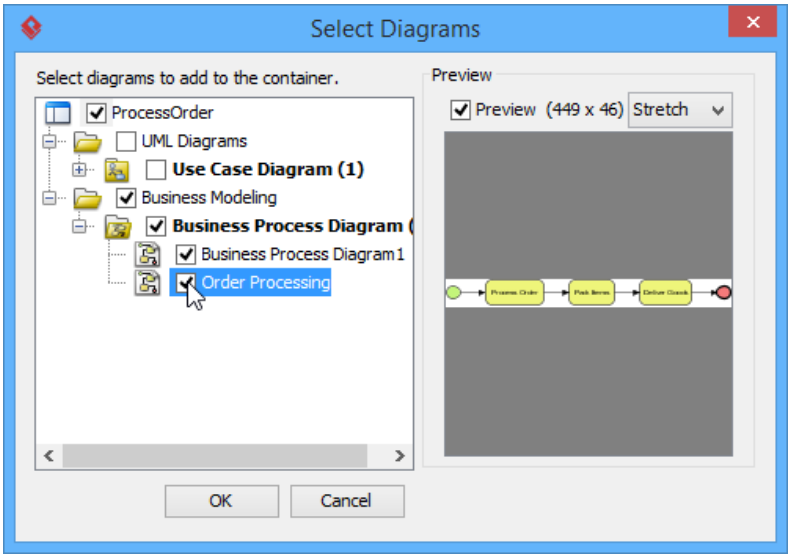
Adding diagram to view

After you create a few diagrams, right-click on a view node and select **Add Existing Diagram...** from the pop-up menu.



*Select **Add Existing Diagram...** from the pop-up menu*

In **Select Diagrams Dialog**, check the diagrams you would like to insert in the view node.

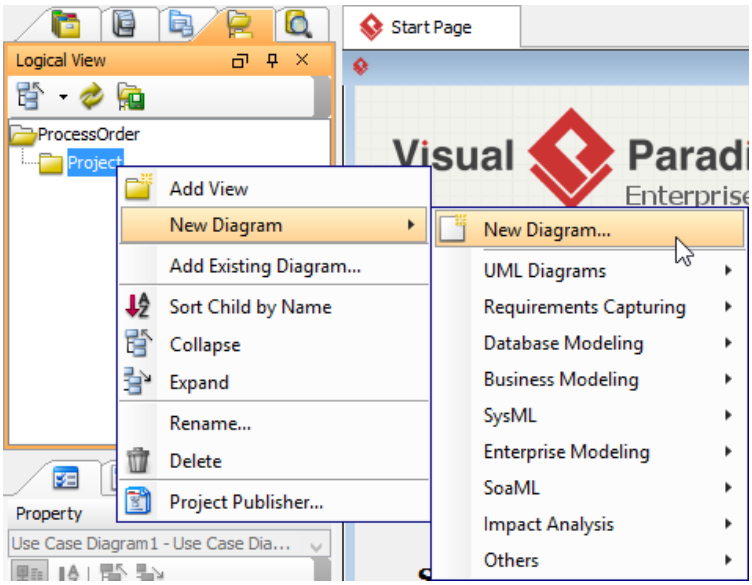


*Check diagrams in **Select Diagrams Dialog***

Click **OK** button to confirm the selection.

Creating a new diagram

Right click the newly created view node, select **New Diagram** from the pop-up menu and then select **New Diagram...** or a pre-defined diagram.



*Select **New Diagram...** from the pop-up menu*

A new diagram is, therefore, created under the view node.

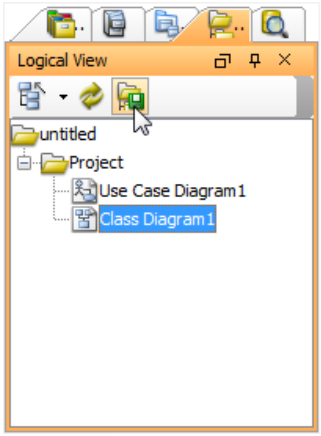
Opening a diagram

Double click on the diagram you want to view in the logical view.

Setting Default View Structure

Visual Paradigm provides a feature where you can set the current logical view structure as default, therefore, you may save your time and do not have to re-create the structure every time you create a new project.

Either click **Set Logical View Structure as Default** button on the top of logical view or right click a root node to select **Set Logical View Structure as Default**.

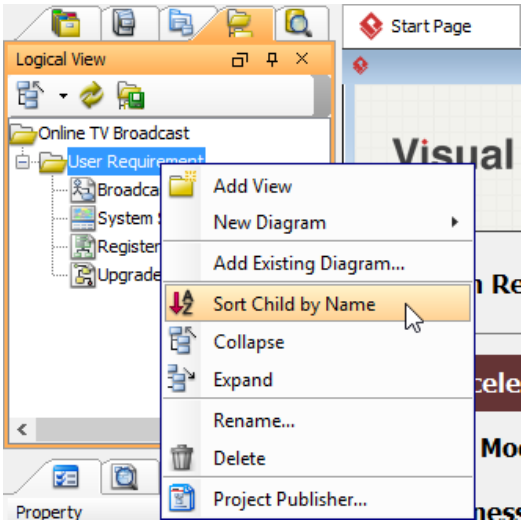


*Click **Set Logical View Structure as Default** button*

In the pop-up **Message** dialog box, click **OK** button. The logical view structure in the new project will then follow the default style you have just customized.

Sorting diagram by name

In logical view, diagram are listed under diagram nodes by default. You can sort child diagrams by their names as well.
To sort by name, right click on view node and select **Sort Child by Name** from the pop-up menu. The child diagrams will be listed by name, in alphabetical order.



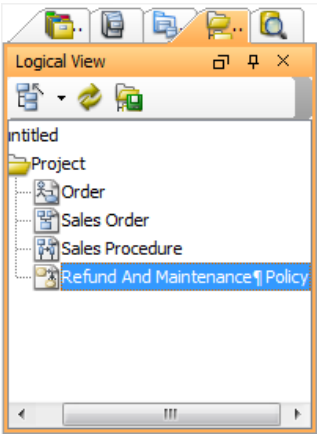
*Click **Sort Child by Name***

NOTE: The sort function applies to the entire logical view instead of the selected node.

Showing/hiding carriage return character

If it is the case that the name of the diagram is in multi-line, the character ¶ will be revealed.

When **Show Carriage Return Character** is selected, line break will be shown.



To show carriage return character

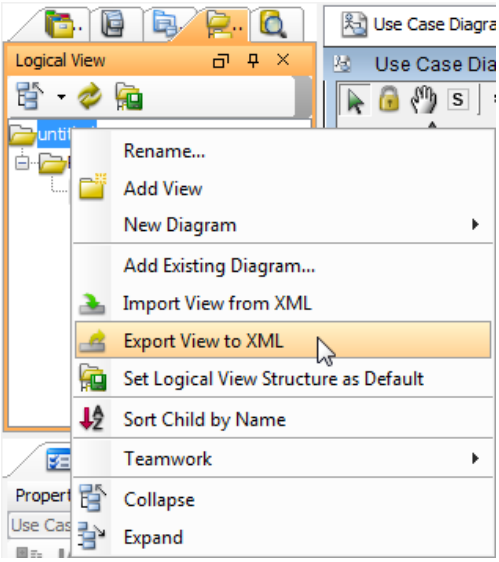
When off, the character ¶ is hidden.

If you want to hide it, uncheck **Show Carriage Return Character** and the character will automatically be unshown.

Exporting View Structure to XML

Visual Paradigm allows you to export the current Logical View Structure as an XML file and to re-use it again on other projects.

Right click the root node and select **Export View to XML** from the pop-up menu.



*Click **Export View to XML** from the pop-up menu*

Find a location for exporting the project and enter its file name in **Save** dialog box. At last, click the **Save**.

Importing View Structure from XML

Visual Paradigm also allows you to import the existing xml file in your new project.

Right click on the root node and select **Import View to XML** from the pop-up menu.

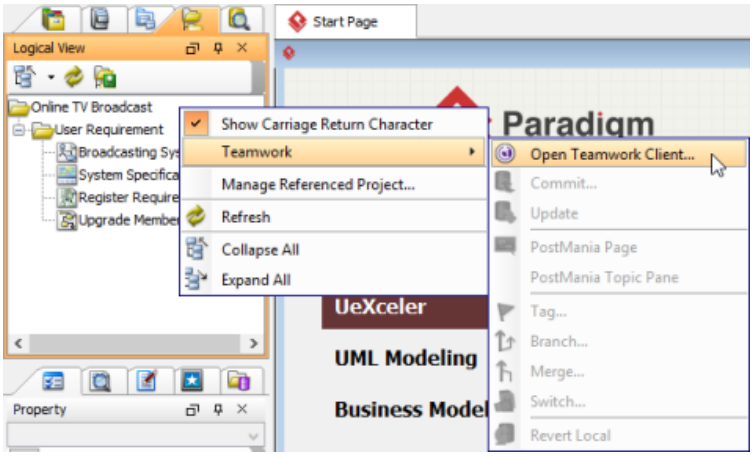
In **Open** dialog box, browse and select the xml file to be imported. You can choose one out of two following choices provided for importing a logical view structure:

1. **Append to existing structure**: the imported structure will be added to the current structure without deleting the old one.
2. **Replace existing structure**: the new imported structure will replace the current structure. Therefore, the current structure will be removed.

Connecting to server for team collaboration

Visual Paradigm's team collaboration support enables your team members work together on projects. To connect to server and perform related activities:

1. Right click on the logical view's background.
2. Select **Teamwork** and the action you want to perform from the pop-up menu.



Perform teamwork

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Working with projects

This chapter introduces how to create and save project. You will also see how to create model and organize diagrams with model.

Creating project

This page shows you how to create a new project in Visual Paradigm.

Saving project

This page shows you how to save a project.

Organizing diagrams by model explorer

You may create models in model explorer for organizing diagrams. This page tells you how to do this in detail.

Project dependency

Establishing dependency between projects for model sharing.

Maintaining backups

Backup files will be saved from time to time. This page tells you more about backup, and how to retrieve works from backup files.

Manage project properties dialog

Edit project properties like project name, author, company and project description.

Project template

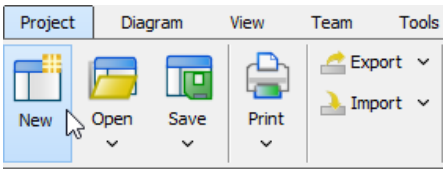
Project template enables you to specify the diagram to create by default when creating a new project.

Switch to diagram

You can open another diagram by double clicking on a tree node in Diagram Navigator. An alternative way is to open the Switch to Diagram dialog box, select diagram and click Activate Selected Diagram.

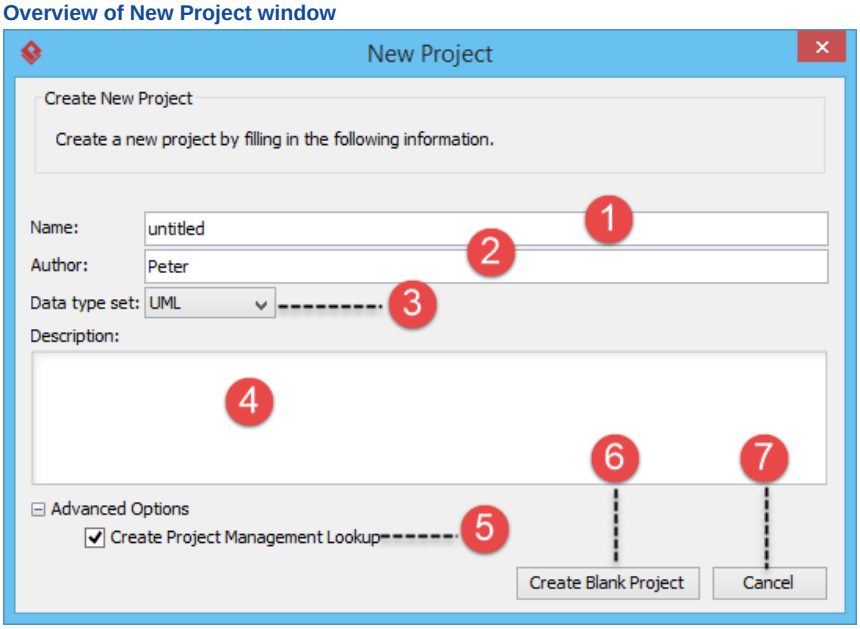
Creating project

Visual Paradigm stores information like model elements and diagrams in a project. Therefore, you need to create a project before performing modeling. To create a project, select **Project > New** from the toolbar. The **New Project** window appears.



Create a new project

Overview of New Project window



*An overview of **New Project** window*

No.	Name	Description
1	Name	The name of project.
2	Author	The person who create the project.
3	Data type set	Lets you select the programming/scripting language for the project. The language you selected mainly affects the class modeling. For example, the selectable visibilities and primitive types vary among languages.
4	Description	The project description. You can make use of the toolbar on top of the description pane to add formatted content.
5	Create Project Management Lookup	Check it to automatically fill the project management lookups such as iteration, version, etc with default lookup values.
6	Create Blank Project	Click to create the project.
7	Cancel	Click to cancel creating project, and close the window.

*Description of **New Project** window*

Related Resources

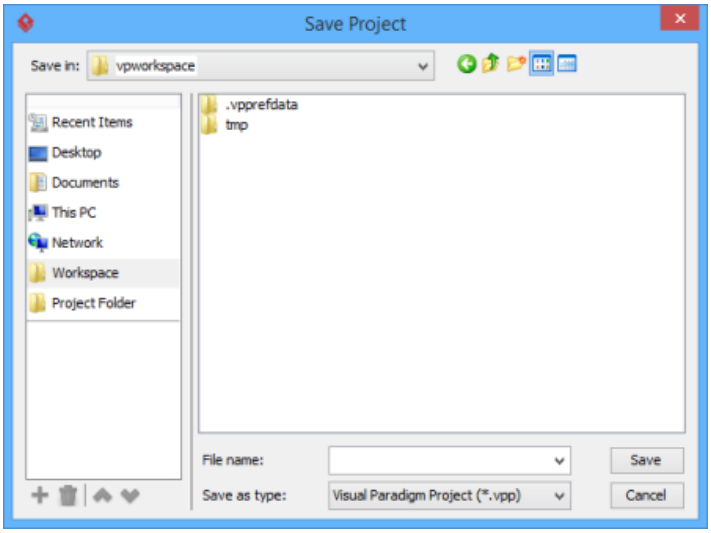
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Saving project

Visual Paradigm saves all project content to a single file, with file extension .vpp.

To save your work, select either **Project > Save** or **Project > Save as...** When you are saving a project for the first time, you will be asked to specify its location. If you have connected to VPository/VP Teamwork Server, you can directly import your project to the server now. Otherwise, you can save the file to a local directory.



Save Project

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Organizing diagrams by Model Structure view

For small scale project, it would be easy to use Diagram Navigator to manage it. However, for middle to large scale project which has considerable numbers of diagrams and model elements, it would be better to use Model Structure view to organize the project.

Visual Paradigm loads diagrams and model elements only when they are used. For example, opening a diagram will load all its diagram elements and opening the specification dialog box of a model element will cause it (and the model elements it referenced) to be loaded. Besides, selecting a tree node in the Model Structure view will cause the corresponding element to be loaded as well.

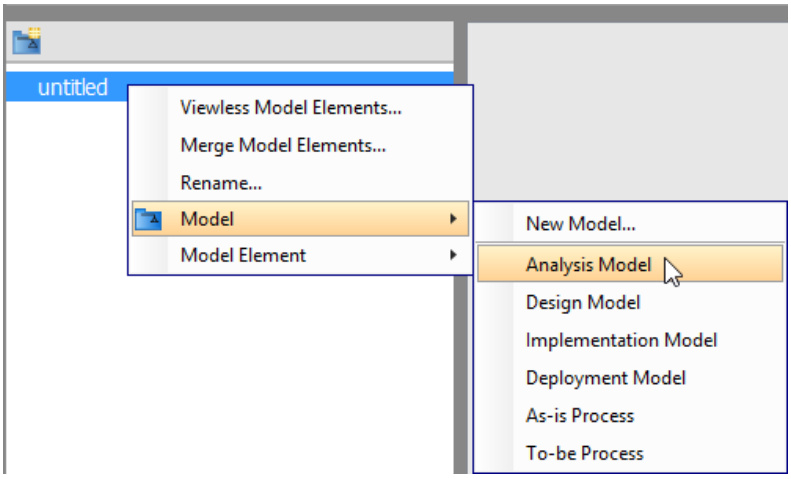
For this reason, we recommend you to group diagrams using **Model** instead of laying them flat in the project. This can avoid accidentally loading diagrams and model elements that you never use and thus can speed up project loading and saving.

To open the Model Structure View:

1. Select **View > Project Browser** from the toolbar to open the Project Browser.
2. In the Project Browser, select **Model Structure** view.

Creating model

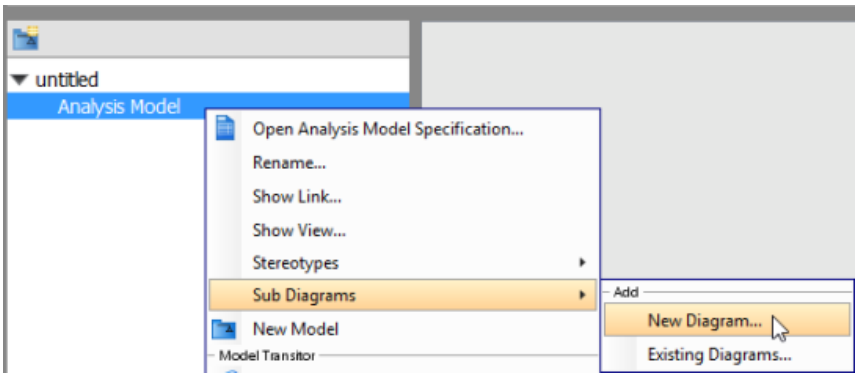
To create a Model, right-click on the project node in Model Structure view and select **Model** from the pop-up menu. You can either create a custom model by selecting **New Model...**, or create a pre-defined Model (e.g. Analysis Model) by selecting it in the list.



Create a model in Model Structure view

Creating diagram in model

To create diagram in model, right-click on the target model and select **Sub Diagrams > New Diagram**s... from the pop-up menu. In the **New Diagram** window, select the type of diagram to create, enter its name and click **OK**.



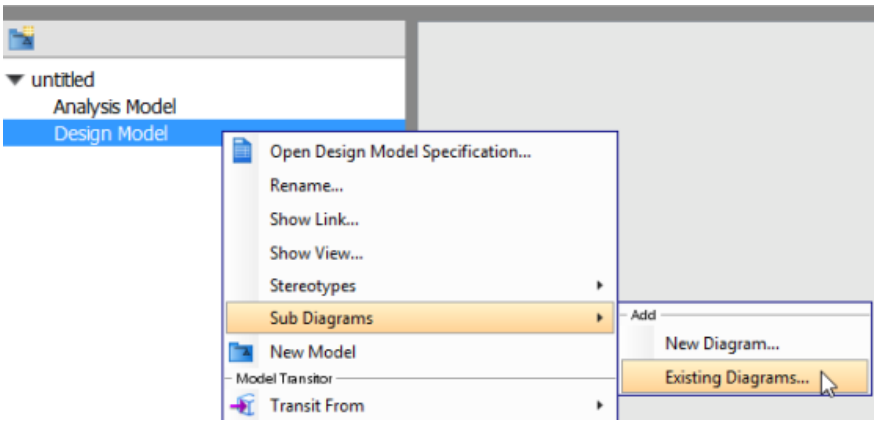
Create a diagram under Model

NOTE: When you draw a shape, its model element will be put under the same model as diagram.

Moving diagrams between models

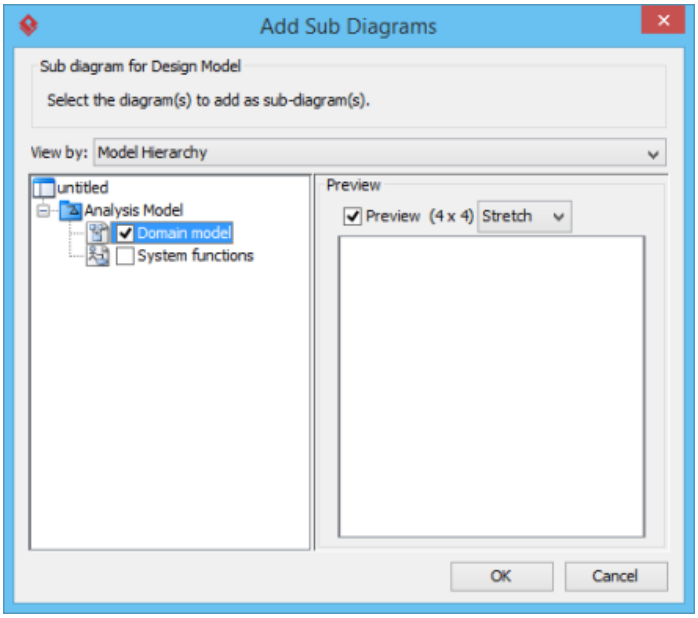
If you are not organizing project structure with model, you may want to do it now. You can move a diagram from root into a model or transfer a diagram from one model to another.

To move diagram from one model to another, right-click on the target model in **Model Structure view** and select **Sub Diagrams** > **Existing Diagrams...** from the popup menu.



Add existing diagram to Model

Select the diagrams you want to move in the **Add Sub Diagrams** window and click **OK**.



Select diagram(s) to move

The selected diagrams will be moved to the target model.

NOTE: If you move a diagram which has the master view of model element(s), the model element(s) will be moved together with the diagram to the new model.

Related Resources

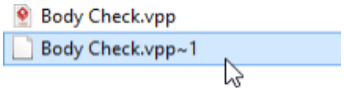
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Maintaining backups

Backup is a copy of project file. The major advantage of backup is to allow you to recover your work, in case you have made some undesired modification on your project. The backup file is usually put along with the project.

Backup is a default setting. After you save your project, it will be produced subsequently. The name of backup file is basically similar to the name of project file, but an extra ~ and a number are appended to it.

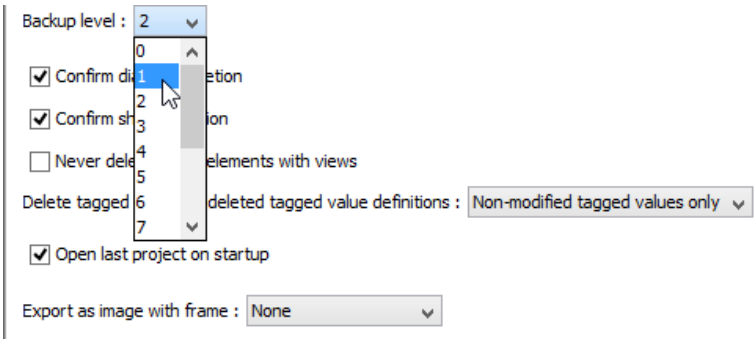


Backup copy is produced

Setting the backup level

You can set the amount of backup copy for your own reference. Select **Windows > Application Options...** from the toolbar.

In **Application Options** window, click **General** and select a number from the drop-down menu of **Backup level** under **Project** tab. The number of backup level represents the amount of backup copy that is produced after a project is saved.



*Select the amount of backup level in **Application Options** window*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Manage Project Properties window

In Visual Paradigm , you can specify the project name, main author of project, your company's name and a description of your project (in rich text format). With **Project Properties** dialog box, you can edit and review your project properties. For your convenience, when you create another new project, all the properties of previous project are set as default, except project name. However, you can modify those default properties in accordance with your preference.

1. Open **Project Properties** dialog box by selecting **Project > Properties...** from the toolbar.
2. Enter project name, author, company and project description. Click **OK** button to confirm and close the dialog box. Note that you can enter formatted text for project description.

Related Resources

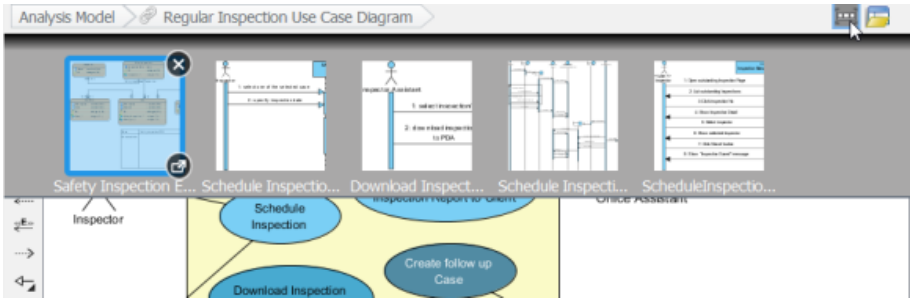
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Switch Diagram

After you've opened a few diagrams, you can switch to your target diagram easily with the feature of **Switch Diagram**. Moreover, you can close diagram(s) with this feature as well.

1. After you've opened a few diagrams, Click the **Switch Diagram** button on the right of the diagram pane.
2. Then, you can open a diagram by double clicking on its thumbnail.



Select a diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Use case diagram

Use case diagram lets you model system functions (i.e. goals) as well as the actors that interaction with those functions. In this chapter, you will learn how to draw a use case diagram, how to record the events behind use cases using use case details and flow of events as well as to have a quick look on actor and use case grid.

Creating use case diagram

Shows you how to create use case diagram and provides you with information about those frequently used use case diagram elements.

UML Use case diagram notations guide

Detailed description of use case diagram notations

Documenting use case details

Document the details of use case with the use case details editor.

Drawing use case diagrams

Use case diagram is a kind of UML diagram that enables you to model system functions (i.e. goals) as well as the actors that interact with those functions. You can draw use case diagrams in Visual Paradigm as well as to document the use case scenario of use cases using the flow-of-events editor. In this page, you will see how to draw use case diagram. Flow of events will be mentioned in coming pages.

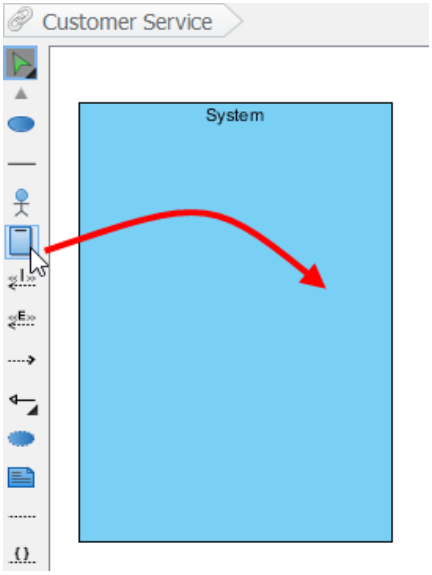
Creating a use case diagram

Perform the steps below to create a UML use case diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Use Case Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Drawing a system

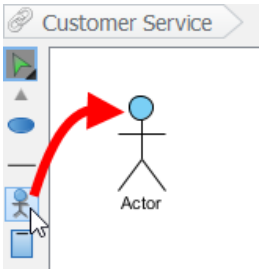
To create a system in use case diagram, select **System** on the diagram toolbar and then click it on the diagram pane. Finally, name the newly created system when it is created.



Create a system

Drawing an actor

To draw an actor in use case diagram, select **Actor** on the diagram toolbar and then click it on the diagram pane. Finally, name the newly created actor when it is created.

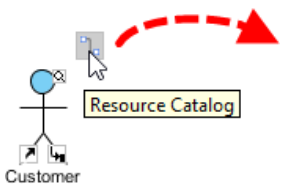


Create an actor

Drawing a use case

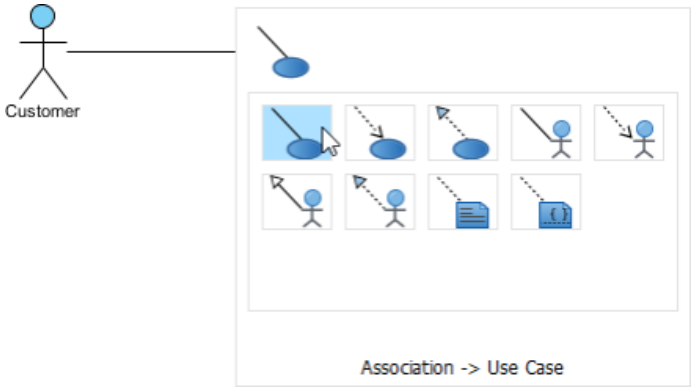
Besides creating a use case through diagram toolbar, you can also create it through Resource Catalog:

1. Move the mouse over a source shape (e.g. an actor).
2. Press on the **Resource Catalog** button and drag it out.



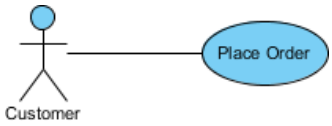
Resource Catalog

3. Release the mouse button until it reaches your preferred place.
4. Select **Association -> Use Case** from Resource Catalog.



To create a use case

5. The source shape and the newly created use case are connected. Finally, name the newly created use case.



Use Case created

Line wrapping use case name

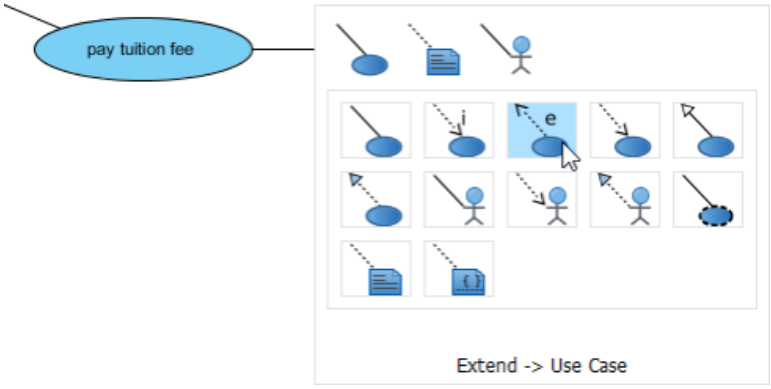
If a use case is too wide, you may resize it by dragging the filled selectors for a better outlook. As a result, the name of use case will be line-wrapped automatically.

Resize a use case

NOTE: Alternatively, you can press **Alt** + **Enter** to force a new line.

Drawing <<Extend>> relationship

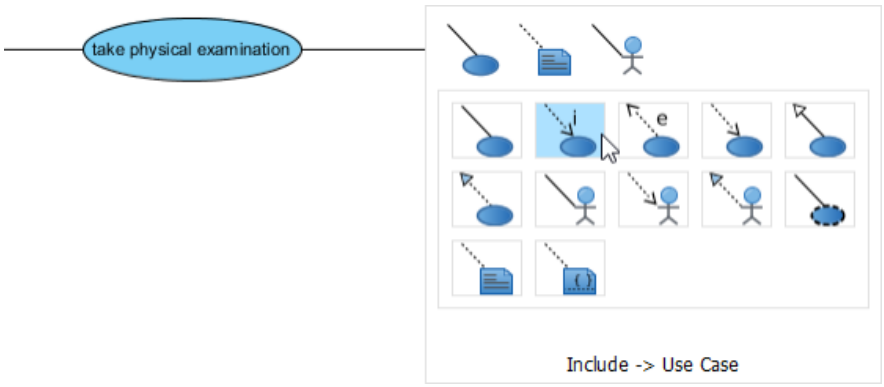
To create an extend relationship, move the mouse over a use case, press and drag out its **Resource Catalog** button. Then, release the mouse button at the preferred place and select **Extend -> Use Case** from Resource Catalog. The use case with extension points and a newly created use case are connected. After you name the newly created use case, you can name the extension point.



Create an extend relationship

Drawing <<Include>> relationship

To create an include relationship, move the mouse over a use case, press and drag out its **Resource Catalog** button. Then, release the mouse button at the preferred place and select **Include -> Use Case** from Resource Catalog. A new use case together with an include relationship is created. Finally, name the newly created use case.

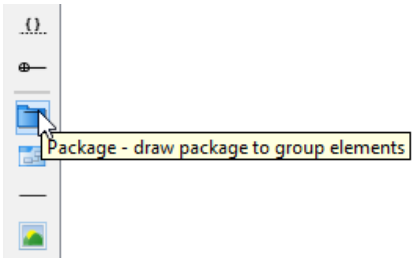


Include relationship is created

Structuring use cases with package

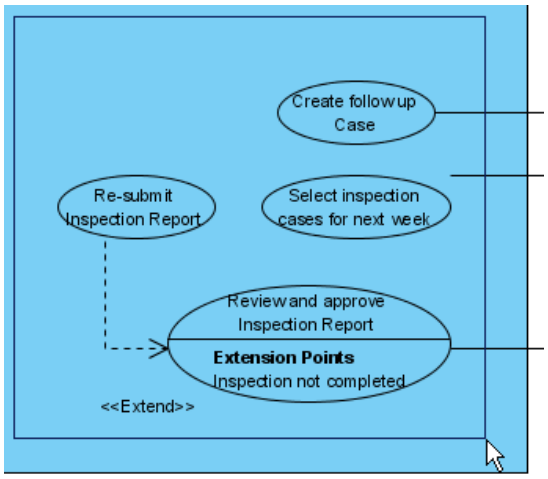
You can organize use cases with package when there are many of them on the diagram.

Select **Package** on the diagram toolbar.



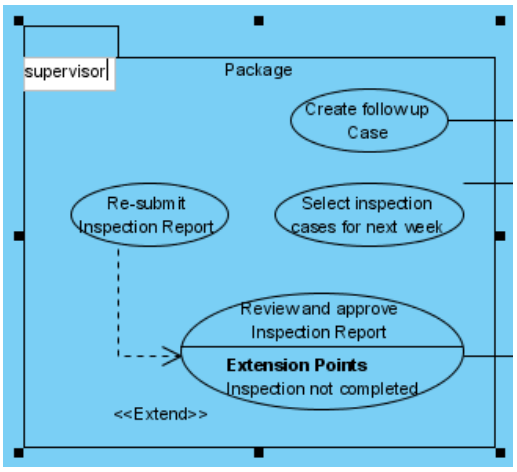
Create a package

Drag the mouse to create a package surrounding those use cases.



Surround use cases with package

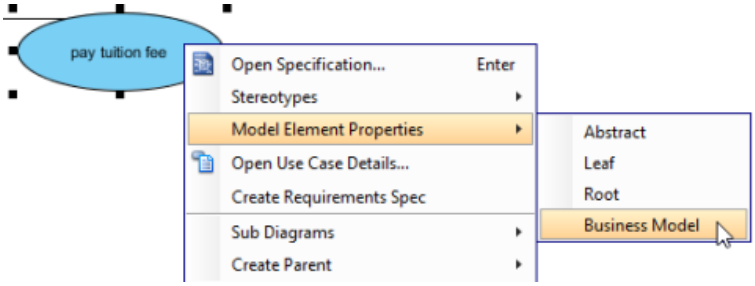
Finally, name the package.



Name the package

Drawing business use case

1. Right click on a use case and select **Model Element Properties > Business Model** from the pop-up menu.



Click Business Model

2. After selected, an extra slash will be shown on the left edge of the use case.

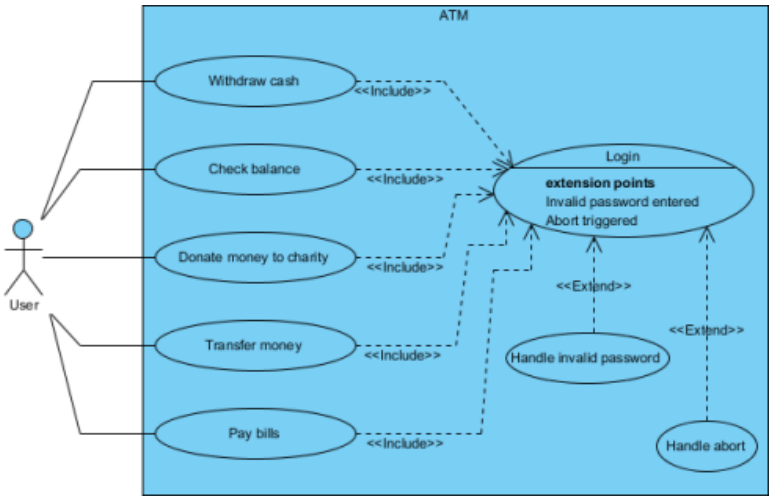
Business Model

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- The 10-tips to create a professional use case diagram
- A set of use case modeling tutorials
- User's Guide - Creating a use case diagram
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

UML Use case diagram notations guide



Sample UML use case diagram

Use case diagram is a kind of UML diagram. Here is a list of Unified Modeling Language (UML) notations supported in a UML use case diagram:

Icon	Name
	Use Case
	Association
	Actor
	System
	Include
	Extend
	Dependency
	Generalization
	Realization
	Collaboration

List of UML notations available in UML use case diagram

Use Case

UML use case

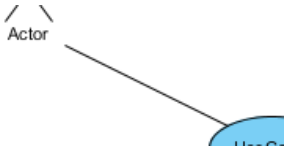
A use case represents a user goal that can be achieved by accessing the system or software application. In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case. You can also describe the use case scenario using the Flow of Events editor.

OMG UML Specification

What is a use case in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 606), use case is:

> A use case is the specification of a set of actions performed by a system, which yields an observable result that is typically of value for one or more actors or other stakeholders of the system.

Association



UML association

Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

OMG UML Specification

What is an association in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 36), association is:

> An association describes a set of tuples whose values refer to typed instances. An instance of an association is called a link. A link is a tuple with one value for each end of the association, where each value is an instance of the type of the end.
> ...
> An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type.
> An end property of an association that is owned by an end class or that is a navigable owned end of the association indicates that the association is navigable from the opposite ends; otherwise, the association is not navigable from the opposite ends.

Actor

UML actor

Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc.

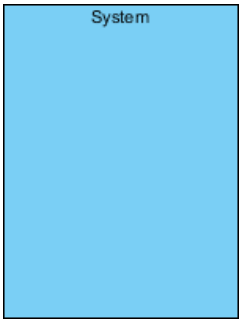
Note that actor represents a role that a user can play but not a specific user. So, in a hospital information system, you may have doctor and patient as actors but not Dr. John, Mrs. Brown as actors.

OMG UML Specification

What is an actor in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1), actor is:

> An actor specifies a role played by a user or any other system that interacts with the subject. (The term "role" is used informally here and does not necessarily imply the technical definition of that term found elsewhere in this specification.)
> ...
> An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data) but which is external to the subject (i.e. in the sense that an instance of an actor is not a part of the instance of its corresponding subject). Actors may represent roles played by human users, external hardware, or other subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e."role") of some entity that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and conversely, a given actor may be played by multiple different instances.

System



UML system

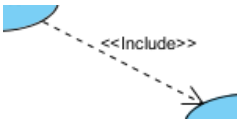
The scope of a system can be represented by a system (shape), or sometimes known as a system boundary. The use cases of the system are placed inside the system shape, while the actor who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.

OMG UML Specification

What is a system in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 608), system is:

> If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier.

Include



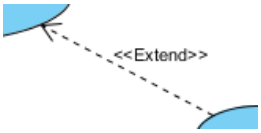
UML include

An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

OMG UML Specification
What is an include in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 604), include is:

> An include relationship defines that a use case contains the behavior defined in another use case.

Extend



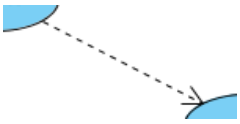
UML extend

An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.

OMG UML Specification
What is an extend in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 601), extend is:

> A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.
> ...
> This relationship specifies that the behavior of a use case may be extended by the behavior of another (usually supplementary) use case. The extension takes place at one or more specific extension points defined in the extended use case. Note, however, that the extended use case is defined independently of the extending use case and is meaningful independently of the extending use case. On the other hand, the extending use case typically defines behavior that may not necessarily be meaningful by itself. Instead, the extending use case defines a set of modular behavior increments that augment an execution of the extended use case under specific conditions.
> Note that the same extending use case can extend more than one use case. Furthermore, an extending use case may itself be extended.

Dependency



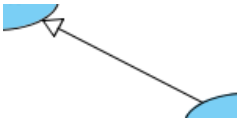
UML dependency

A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

OMG UML Specification
What is a dependency in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 61), dependency is:

> A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s).

Generalization



UML generalization

A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element share the same specification with. the more general the model element but carries more details in extra.

OMG UML Specification
What is a generalization in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 70), generalization is:

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier.

Realization

UML realization

A realization is a relationship between a specification and its implementation.

OMG UML Specification

What is a realization in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 131), realization is:

Realization is a specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client). Realization can be used to model stepwise refinement, optimizations, transformations, templates, model synthesis, framework composition, etc.

Collaboration

UML collaboration

OMG UML Specification

What is a collaboration in UML? According to the OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.1, page 174), collaboration is:

A collaboration describes a structure of collaborating elements (roles), each performing a specialized function, which collectively accomplish some desired functionality. Its primary purpose is to explain how a system works and, therefore, it typically only incorporates those aspects of reality that are deemed relevant to the explanation. Thus, details, such as the identity or precise class of the actual participating instances are suppressed.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

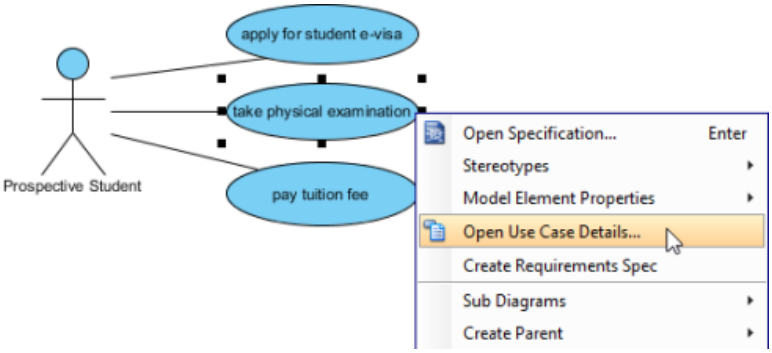
- The 10-tips to create a professional use case diagram
- A set of use case modeling tutorials
- User's Guide - Creating a use case diagram
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Documenting use case details

The feature of use case details refers to the basic information, flow of events , requirements and test plan of a use case. Documenting use case details is essential in recording meaningful and important information for a use case.

Opening use case details

To start editing and viewing use case details, right click on the target use case in use case diagram and select **Use Case Details...** from the pop-up menu.

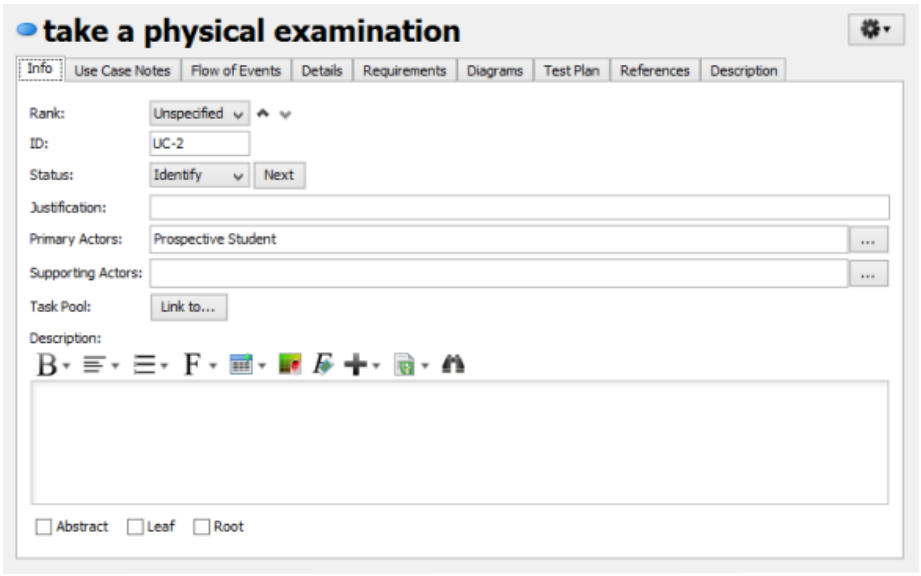


*Select **Open Use Case Details...***

Entering basic information

Basic information refers to all general information of a use case. Rank and justification determine the importance of a use case. Select a rank from the drop-down menu and enter the text in **Justification** text field.

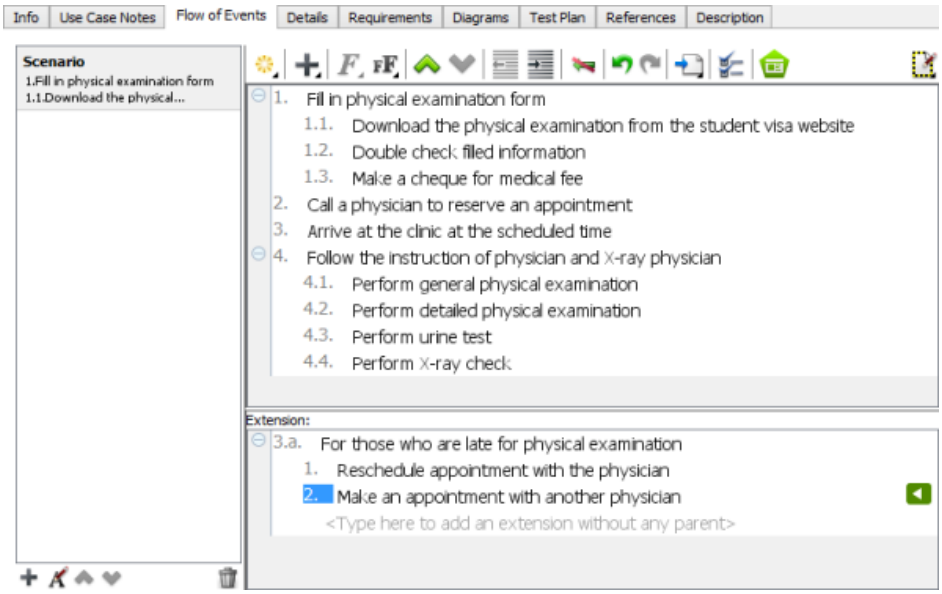
Primary actors list the actors being involved in a use case. Actors that are connected to a use case are automatically defined as primary actors. Supporting Actors are actors who are beneficial from the system but without direct interaction. Both primary and supporting actors can be added manually by pressing the **Plus** button and select the actors in the pop-up dialog box.



Basic information of use case

Entering flow of events

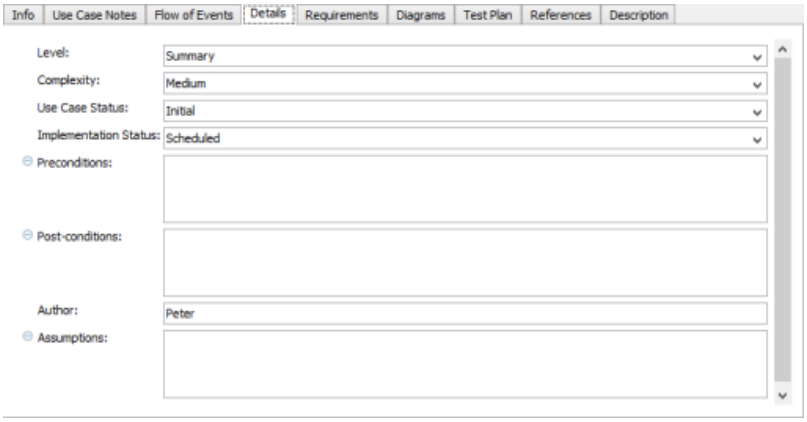
Flow of events refers to the steps required to go through and fulfill a use case. You may define multiple flows of events under a use case and add extension to an event as well. For more information about documenting flow of events, read the next chapter **Documenting Flow of Events**.



Flow of events of use case

Entering details

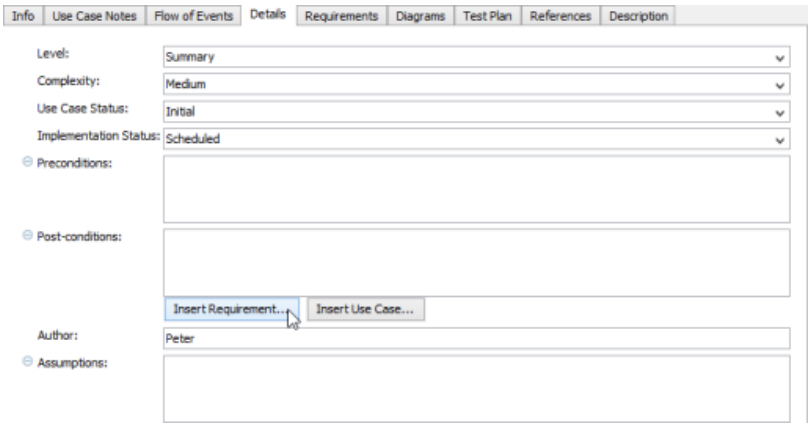
Details are predefined and detailed fields of a use case, which includes level, complexity, status, implementation status, preconditions and post-conditions, author and assumptions. Select an option for **Level**, **Complexity**, **Use Case Status** and **Implementation Status** from the drop-down menu.



Details of use case

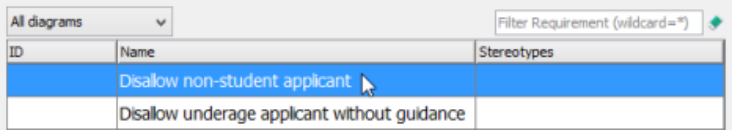
Inserting requirement links

1. Click in the text field where you want to insert a requirement link. Click the **Insert Requirement...** button when it pops out. Note that only fields which support multiple line are allowed to add requirement links.



*Click **Insert Requirement...** button*

2. When the **Select Requirement** dialog box pops out, select the requirement you want to link to and click **OK** to confirm. The searching scope of selecting requirement may be narrowed down if you find too many requirements in your project. Select a specific diagram from the drop-down menu at the top-left corner of dialog box or enter its name at the **Filter** field directly at the top right corner.

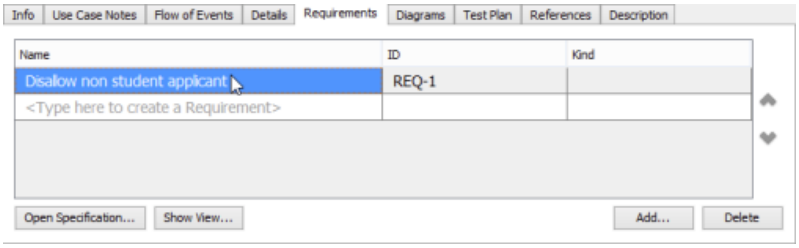


Select a requirement

3. Once the link is inserted in the text field, you can right click to navigate it through its pop-up menu.

Adding requirements

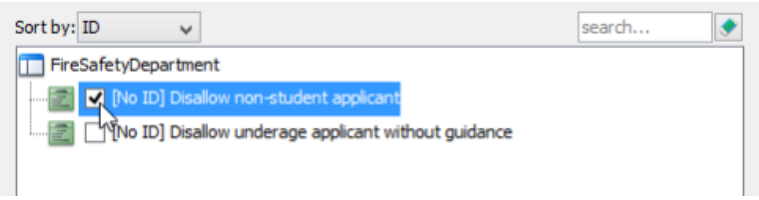
Requirements of a use case can be added in the **Requirements** page.



Requirements of use case

To add requirement(s) to a use case:

1. Click the **Add...** button at the bottom right of dialog box.
2. In the **Requirements** dialog box, look for and select the requirements to add and click **OK** to confirm the selection.

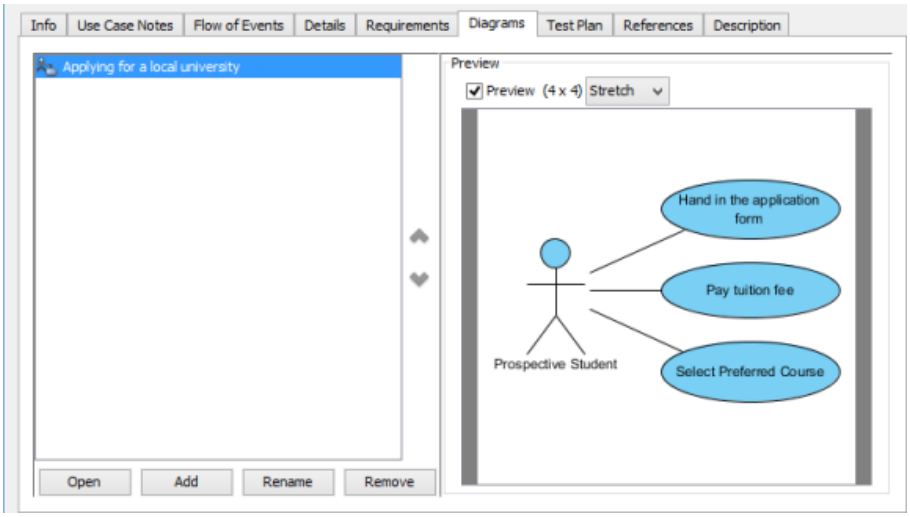


Select a requirement

> **NOTE:** The **Requirements** page is for adding existing requirements as requirements. If you want to define a new requirement, read the next section *Adding a sub-diagram*. Information about how to add a requirement diagram as sub-diagram and define the requirements in the diagram is provided. Requirements which are made in **Diagrams** page will be automatically added to the use case's requirements.

Managing sub-diagrams

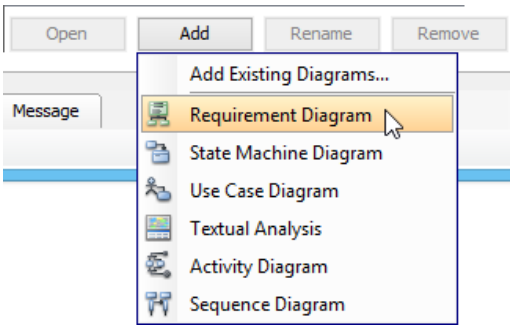
You can make use of another diagram for elaborating a use case. The **Diagrams** page enables you to add and open sub-diagrams of a use case. When you select a diagram on the list on the left, you may preview it on the right if **Show preview** is checked.



Diagrams of use case

Adding a sub-diagram

1. Click the **Add** button at the bottom of **Diagrams** page, select a type of diagram from the pop-up menu if you want to add a new diagram as sub-diagram. On the other hand, select **Add Existing Diagrams...** if you want to add an existing diagram in your current project.

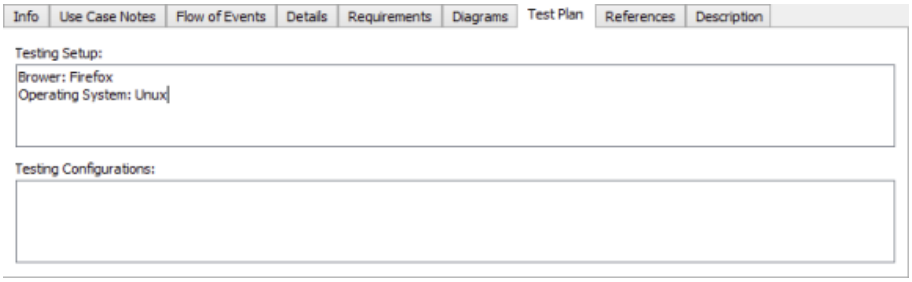


Add a sub-diagram

Opening a sub-diagram
Select a sub-diagram on the list to open and click the **Open** button at the bottom of **Diagrams** page.

Writing test plan

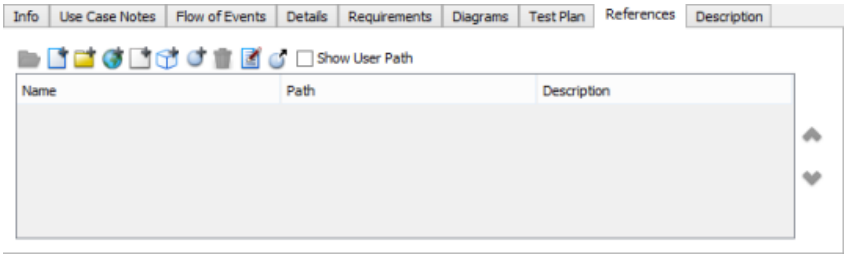
While the detailed testing procedure can be documented in flow of events, the testing setup and configurations can be documented in the **Test Plan** tab.



Test Plan of use case

Adding references

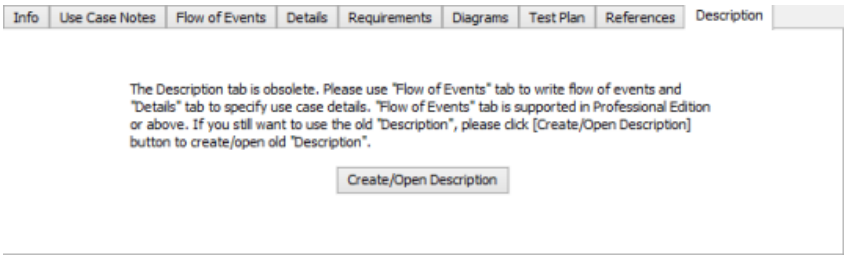
You may add references to both internal and external artifacts, such as shapes, diagrams, files, folders and URLs for describing the use case in various views.



References of use case

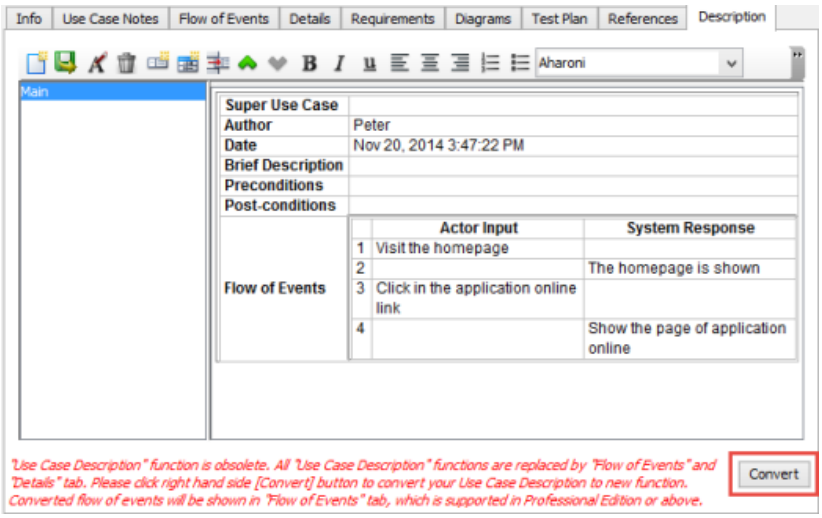
Opening obsolete use case description

Use Case Description was an obsolete feature removed in Visual Paradigm Suite version 4.1. We recommend users to use the flow of events tool to document the internal flow of use case rather than making use of use case description. But for the old version users, they can still activate the **Description** pane to access the saved data.



Obsolete use case description

You can convert the obsolete use case details to flow of events by clicking on the **Convert** button at the bottom right corner.



Convert use case description

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- A set of use case modeling tutorials
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Class diagram

A class diagram models the classes of a system/application going to develop. In this chapter, you will learn how to create class diagram.

Creating class diagram

Create and draw class diagram. You will learn how to create class, attribute, operation and other common class diagram constructs.

Drawing class diagrams

A class diagram is a kind of UML diagram that shows the objects that are required and the relationships between them. Since it provides detailed information about the properties and interfaces of the classes, it can be considered as the main model and regard the other diagrams as supplementary models.

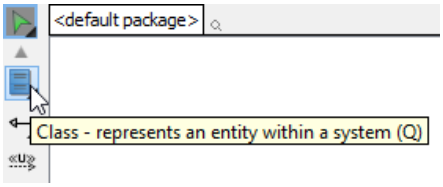
Creating class diagram

Perform the steps below to create a UML class diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Class Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

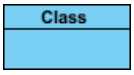
Creating class

To create a class in a class diagram, click **Class** on the diagram toolbar and then click on the diagram.



Create class

A class will be created.

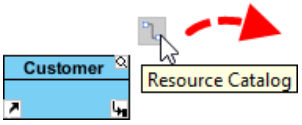


Class created

Creating association

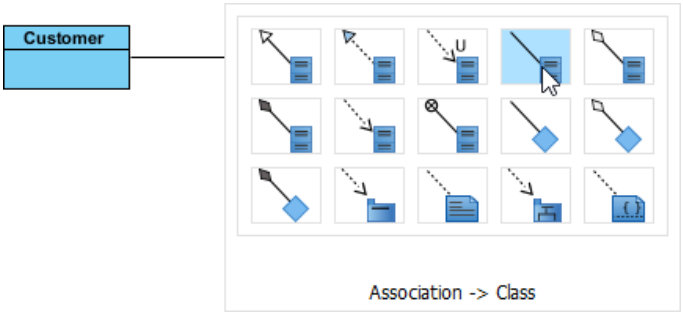
To create an associated class in a class diagram:

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.



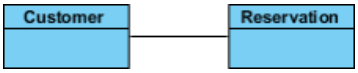
Using Resource Catalog

3. Release the mouse button at the place where you want the class to be created. If you want to connect to an existing class, drop at that class. Otherwise, drop an the empty space (either at the diagram background or container shape like package).
4. If you are connecting to an existing class, select **Association** from Resource Catalog. If you are creating a new class, select **Association -> Class** from Resource Catalog. If you want to create an aggregation or composition, select **Aggregation -> Class** or **Composition -> Class** instead.



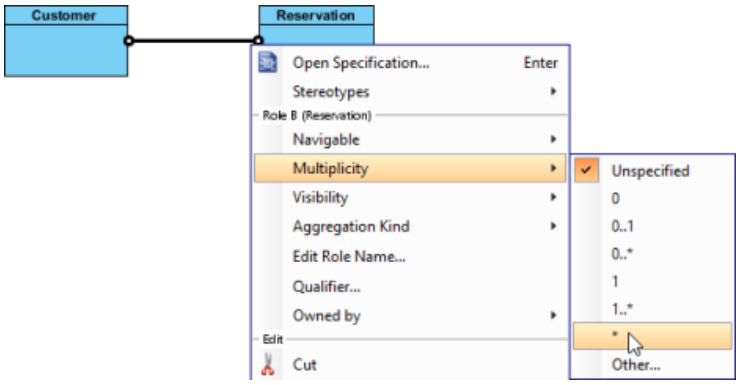
To create a class

5. If you are creating a new class, you should see the class now and it is connected to the source shape. Enter its name and press **Enter** to confirm editing.



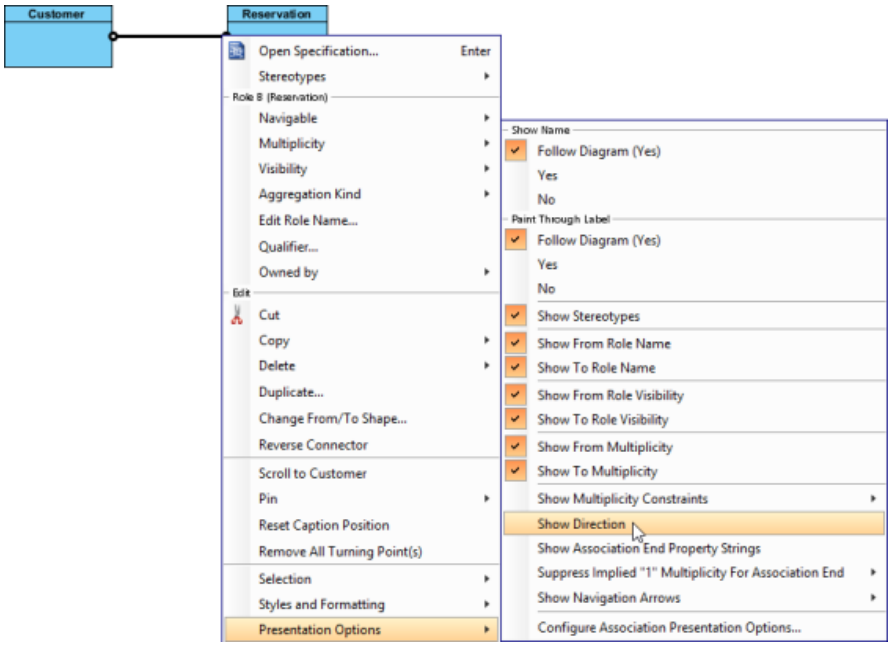
Associated class created

To edit multiplicity of an association end, right-click near the association end, select **Multiplicity** from the popup menu and then select a multiplicity.



Edit multiplicity

To show the direction of an association, right click on it and select **Presentation Options** > **Show Direction** from the pop-up menu.



Show direction

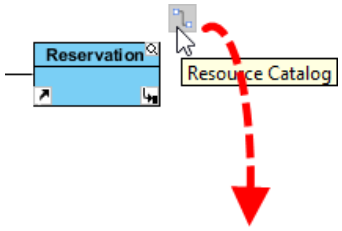
The direction arrow is shown beside the association.

Direction shown

Creating generalization

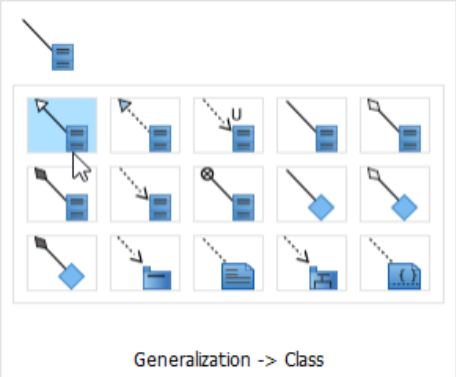
To create a subclass:

1. Move your mouse pointer over the superclass.
2. Press on the **Resource Catalog** button and drag it out.



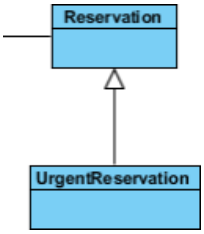
Using Resource Catalog

3. Release the mouse button at the place where you want the subclass to be created. If you want to connect to an existing class, drop at that class. Otherwise, drop an the empty space (either at the diagram background or container shape like package).
4. If you are connecting to an existing class, select **Generalization** from Resource Catalog. If you are creating a new class, select **Generalization -> Class** from Resource Catalog.



To create a subclass

5. If you are creating a new class, you should see the class now and it is connected to the source shape with a generalization. Enter its name and press **Enter** to confirm editing.



Subclass created

Creating dependency from/to attribute/operation

You can also add a dependency from and/or to an attribute or operation in class. To create such a dependency.

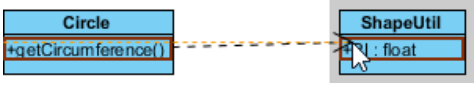
1. Select **Dependency** from the diagram toolbar.

Selecting Dependency

2. Press on the source shape or a class member.

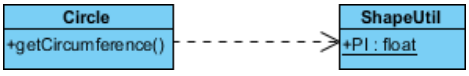
To press on the source operation

3. Drag to the target shape, or a class member.



Dragging to target attribute

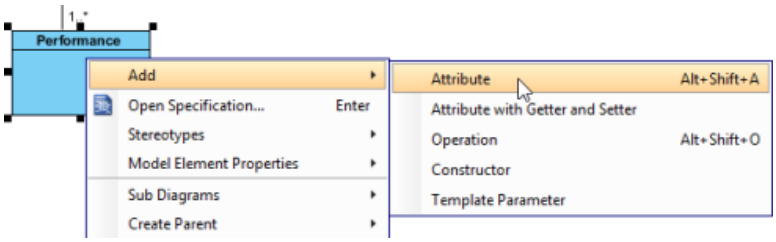
4. Release the mouse button to create the dependency.



Dependency created between an operation and a member

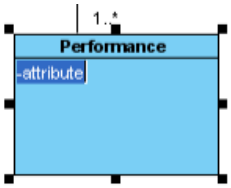
Creating attribute

To create attribute, right click the class and select **Add** > **Attribute** from the pop-up menu.



Create attribute

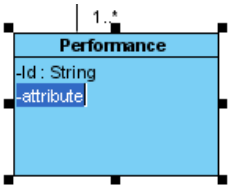
An attribute is created.



Attribute created

Creating attribute with enter key

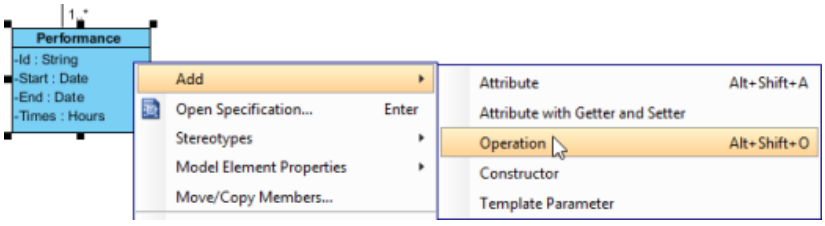
After creating an attribute, press the **Enter** key, another attribute will be created. This method allows you to create multiple attributes quickly and easily.



Create attribute with Enter key

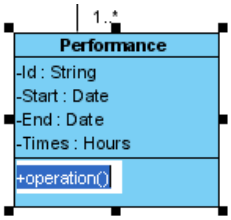
Creating operation

To create operation, right click the class and select **Add** > **Operation** from the pop-up menu.



Create operation

An operation is created.

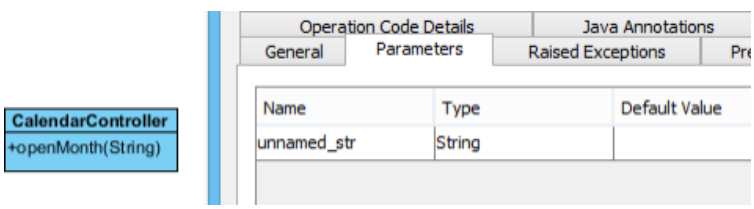


Operation created

Similar to creating attribute, you can press the **Enter** key to create multiple operations continuously.

Showing just a parameter's type

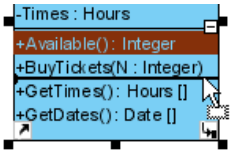
When the name of a parameter starts with "unnamed_", its name will not be displayed in the class shape, leaving the parameter type (if defined).



Unnamed parameter

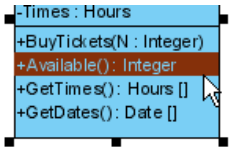
Drag-and-Drop reordering, copying and moving of class members

To reord a class member, select it and drag within the compartment, you will see a thick black line appears to indicate where the class member will be placed.



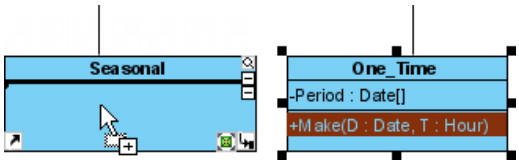
Reorder class member

Release the mouse button, the class member will be reordered.



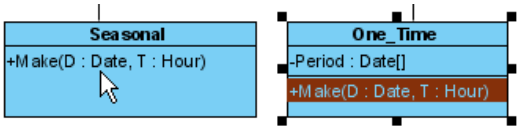
Class member reordered

To copy a class member, select it and drag to the target class while keep pressing the Ctrl key, you will see a thick black line appears indicating where the class member will be placed. A plus sign is shown beside the mouse cursor indicating this is a copy action.



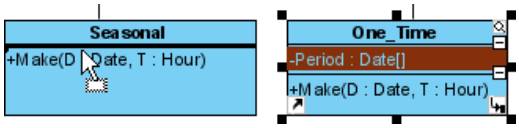
Copy class member

Release the mouse button, the class member will be copied.



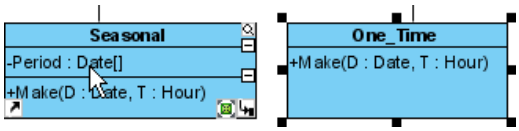
Class member copied

To move a class member, select it and drag to the target class, you will see a thick black line appears indicating where the class member will be placed. Unlike copy, do not press the Ctrl key when drag, the mouse cursor without the plus sign indicates this is a move action.



Move class member

Release the mouse button, the class member will be moved.



Class member moved

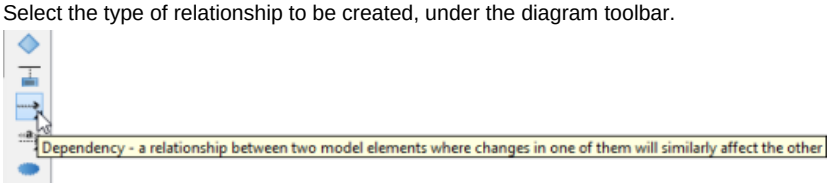
Selecting all class members

To select all members within a class, you can select any member first, and then press **Alt-A** to select the rest.

Relating class members

Relationships such as dependency and generic connectors can be added between attribute and operation of classes. To do this:

1. Select the type of relationship to be created, under the diagram toolbar.

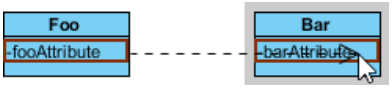


Selecting Dependency

2. Move the mosue pointer over the source member.

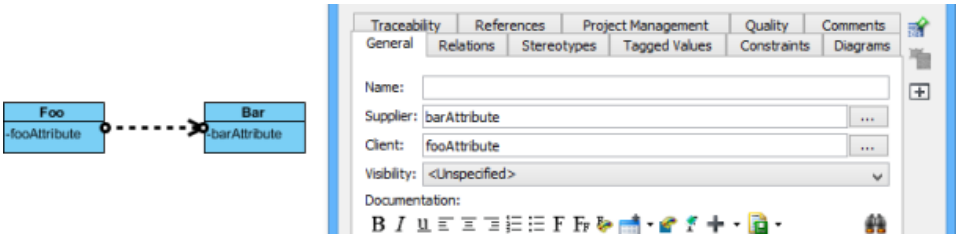
Create a relationship from a class member

3. Press on it and hold the mouse button.
4. Drag to the target member.



Release mouse button on target class member

5. Release the mouse button to create the connector. While it looks like the connector is connecting the classes but not the members, if you check its specification you can see that the connector is indeed connecting the members.



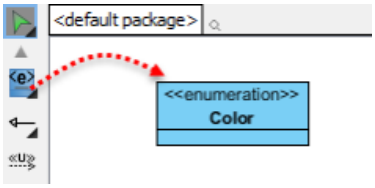
Relationship created

Creating enumeration and adding enumeration literal

An enumeration is a special data type that consists of a pre-defined set of values, known as enumeration literals. Here are some of the common examples:

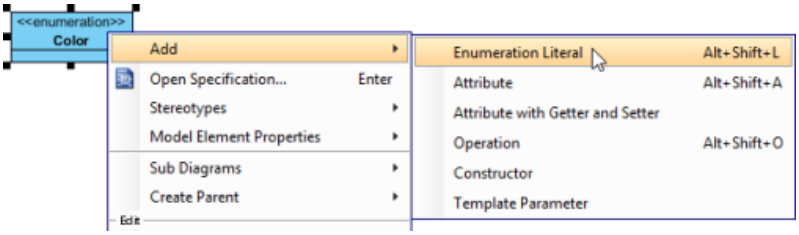
- Color (RED, GREEN, BLUE)
- Orientation (NORTH, SOUTH, EAST, WEST)
- Switch (ON, OFF)

To create an enumeration, select **Enumeration** from the diagram toolbar and click on the diagram to create one.



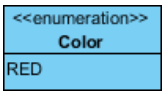
Create an enumeration

To add an enumeration literal, right click on the enumeration class and select **Add > Enumeration Literal** from the popup menu.



Add an enumeration literal

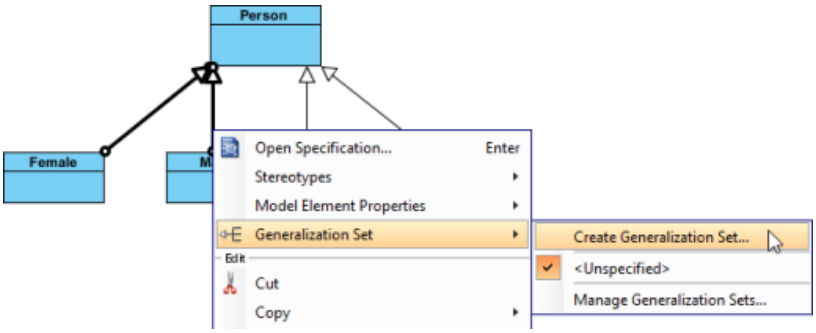
Then, enter the name of the literal and confirm editing.



Enumeration literal entered

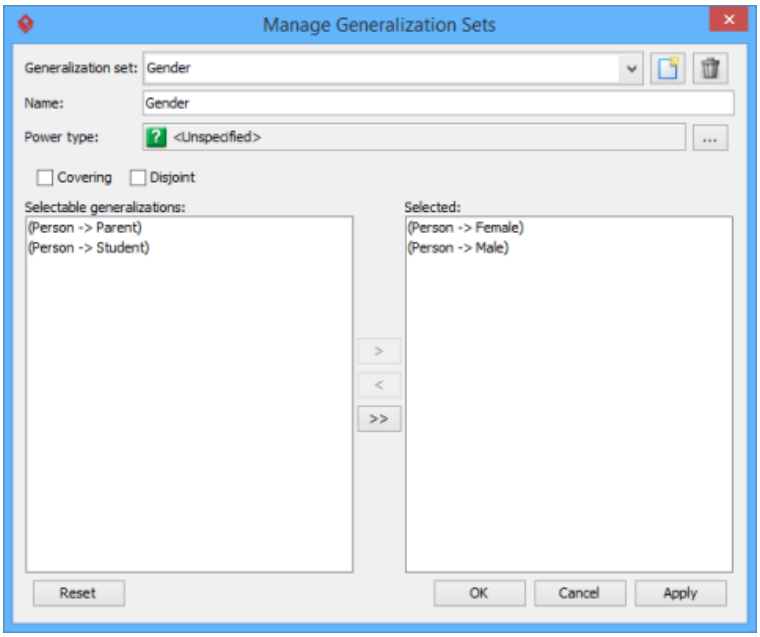
Generalization set

A generalization set defines a particular set of generalization relationships that describe the way
in which a general classifier (or superclass) may be divided using specific subtypes. To define a generalization set, select the generalizations to include, right click and select Generalization set > Create Generalization Set... from the popup menu.



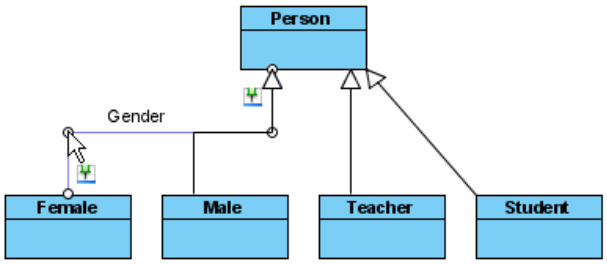
Create a generalization set

Name the set in the Manage Generalization Sets dialog box, and confirm by pressing OK.



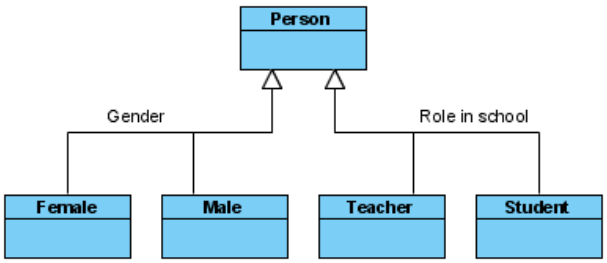
Name the generalization set

The selected generalizations are grouped. Adjust the connector to make the diagram tidy.



Adjust connector

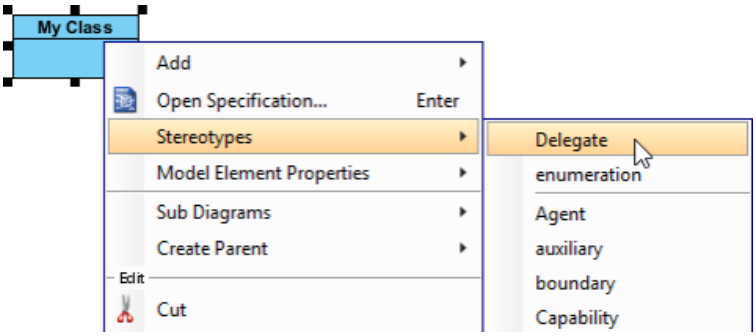
Repeat the steps for other generalizations.



Generalization sets defined

Defining delegate method for class

When project's programming language is set to be Visual Basic or C#, it is possible to define delegate method for classes. To define delete method, right click on the class and select **Stereotypes > Delegate** from the pop-up menu.



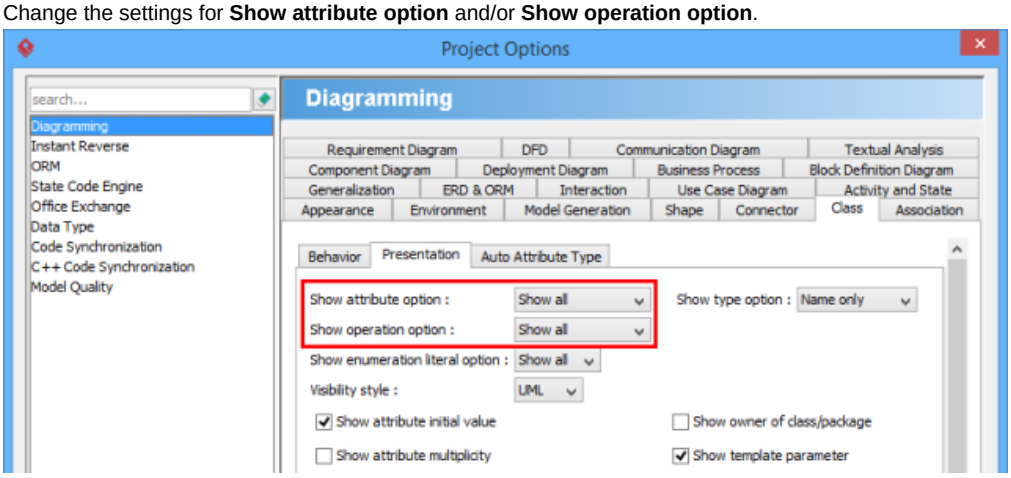
Stereotype class as Delegate

Hiding (and showing) attributes and operations

Per workspace

This applies to new classes that will be created in a project opened in specific workspace. To change the setting:

1. Select **Windows > Project Options** from the toolbar to open the **Options** dialog box.
2. Click **Diagramming** on the list.
3. Open the **Class** tab.
4. Click **Presentation** tab.
5. Change the settings for **Show attribute option** and/or **Show operation option**.

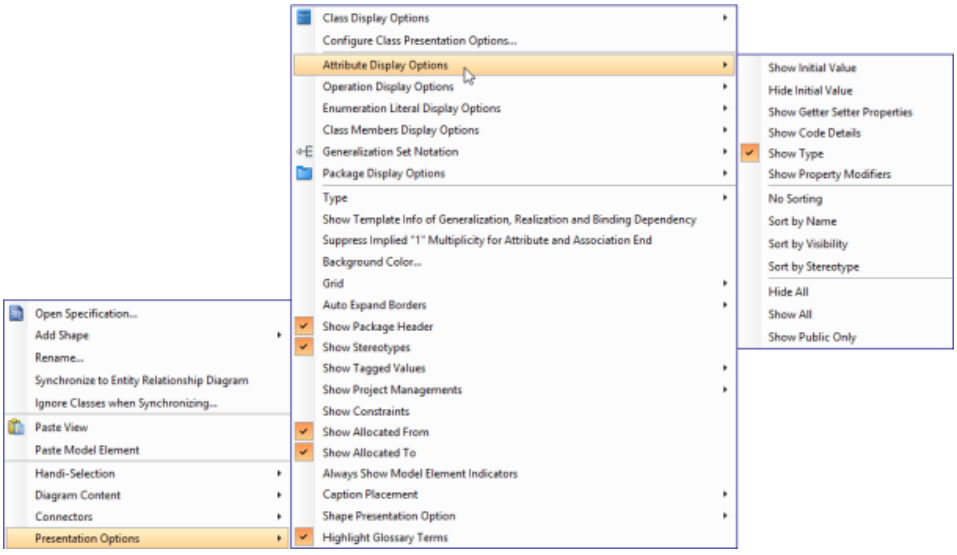


Show or hide operations

Per diagram

This applies to classes in specific diagram. To change the setting:

1. Right click on the class diagram to set the option.
2. Select **Presentation Options > Attribute Display Options / Operation Display Options** from the pop-up menu.
3. Select Hide All / Show All / Show Public Only.

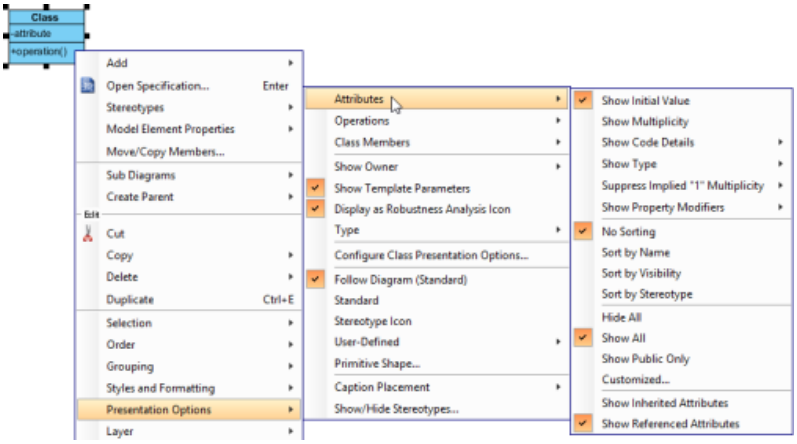


Change the operations' presentation options for classes in diagram

Per class

This applies to specific class. To change the setting:

1. Right click on the class to set the option.
2. Select **Presentation Options > Attributes / Operations** from the popup menu.
3. Select **Hide All** / **Show All** / **Show Public Only**.

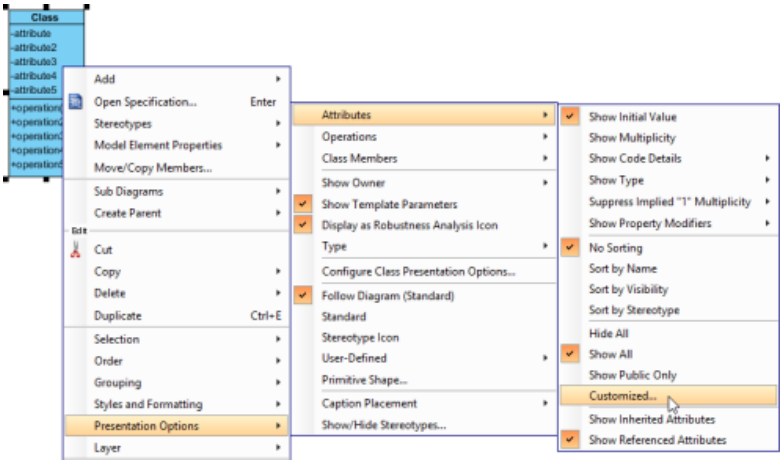


Change the operations' presentation options for a class

For specific attribute/operation

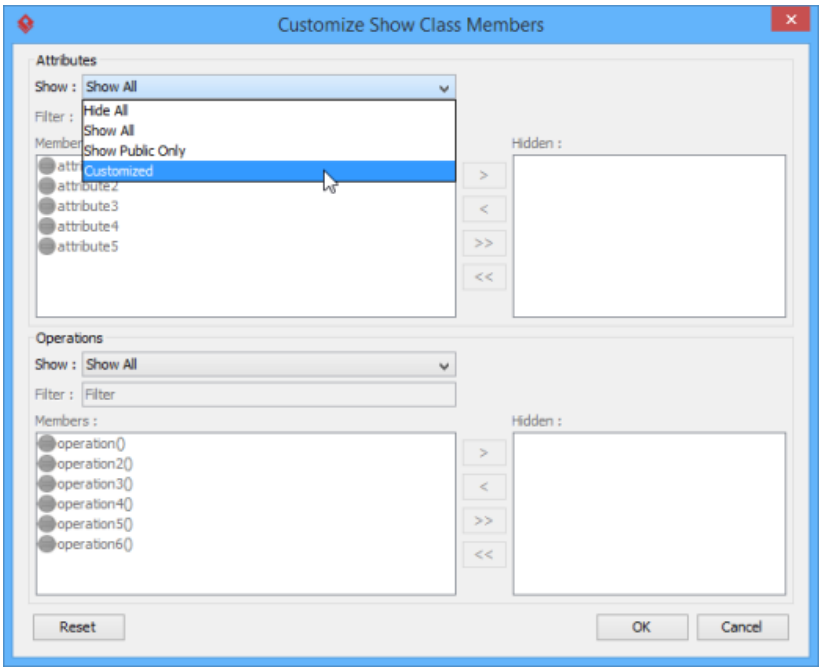
Instead of showing or hiding all members or public members, you may show/hide specific class member per class. To do this:

1. Right click on the class to set the option.
2. Select **Presentation Options > Attributes / Operations > Customized...** from the pop-up menu.



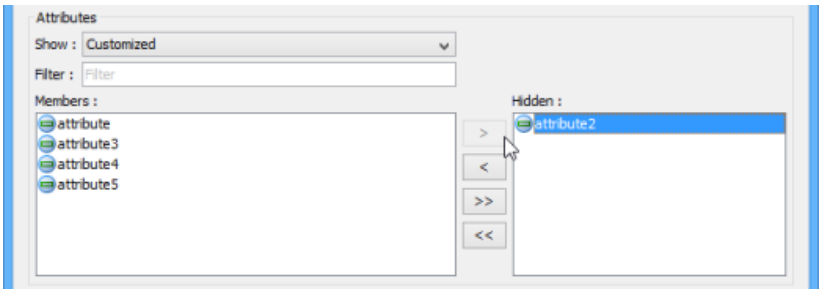
Show or hide specific class member

3. Select **Customized** under the drop down menu of **Show**.



Select Customized in dialog box

4. Select the member(s) to hide and click **>** to hide them.



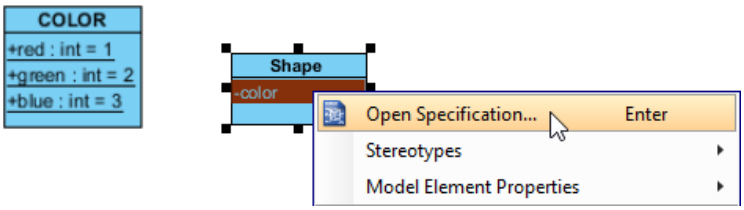
Select attributes to hide

5. Click **OK** button to confirm.

Setting initial (default) value for attribute

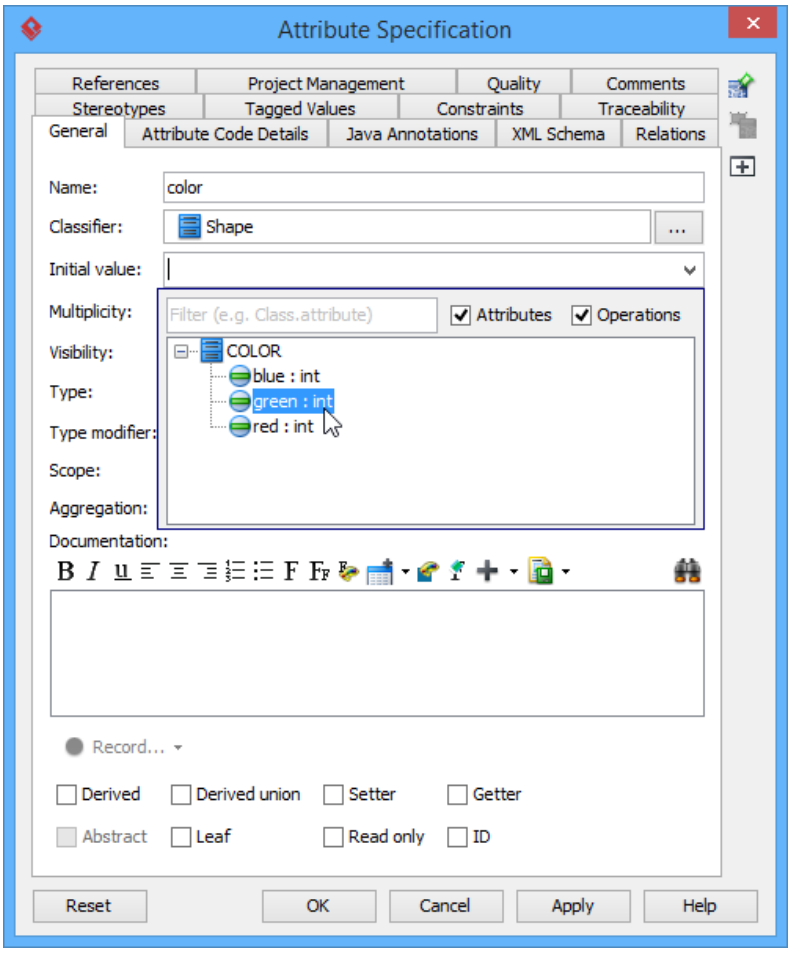
Initial value can be set to an attribute, indicating the default value of the attribute when the owning object is instantiated. You can give a text value for initial value, or select an attribute of another class. To set initial value to an attribute:

1. Open the specification dialog box of attribute by right clicking on the attribute and selecting **Open Specification...** from the popup menu.



Opening the attribute specification

2. In the **General** page of the specification dialog box, enter the initial value in initial value field if it is a text value, or popup the drop down menu to select a public and static field of any class to be the value.



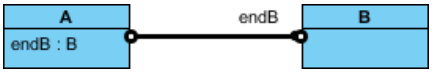
Selecting an initial value

NOTE: In order to select the attribute of another class to be the default value, make sure the attribute you want to select is static (i.e. set to be in classifier scope) and is public (so that other classes can access).

3. Click **OK** to confirm.

Setting the ownership of association end

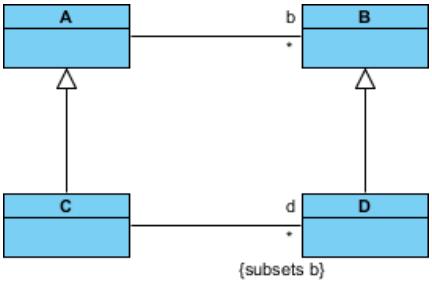
Ownership of association ends by an associated class may be indicated by a small dot. To set the ownership, right click at the association end where you want to set ownership, select **Owned by** in the popup menu, then select either the association or the class at the opposite end. By selecting class, the small dot will be shown.



Association end with ownership set

Subsetting on association end

Take a look at the sample below. The subset on d indicates that the collection d, which is an instance of class C, is a subset of the collection b, instance of class A.



Subsetting on association end

To define a subset on an association end:

1. Right click on the association (where the subset end exist) and select **Open Specification...** from the popup menu.

2. In the **General** tab, locate the association end where you want to define a subset. Click on **...** for the Role property of the association end.
3. In the **Association End Specification**, open the **Subsetted Association Ends** tab.
4. From the list on the left hand side, click on the role you want to define subset for. Click **>** to select it. If you do not see any role listing there, make sure your model respect the pattern similar to the class diagram above - The class of both association ends are subclasses, and there is an association connecting their superclasses.
5. Click **OK** to confirm and close the **Association End Specification** window.
6. Click **OK** to confirm and close the **Association Specification** window.
7. Right click on the association end and select **Presentation Options > Show Association End Property Strings** from the popup menu to show the subset.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Sequence diagram

A sequence diagram is used primarily to show the interactions between objects that are represented as lifelines in a sequential order. In this chapter, you will learn how to draw a sequence diagram.

Creating sequence diagram

Teaches you how to create sequence diagram through the diagram and through the editor at the bottom of diagram.

Drawing sequence diagrams

A sequence diagram is a kind of UML diagram that is used primarily to show the interactions between objects that are represented as lifelines in a sequential order.

Creating sequence diagram

Perform the steps below to create a UML sequence diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Sequence Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating actor

To create actor, click **Actor** on the diagram toolbar and then click on the diagram.

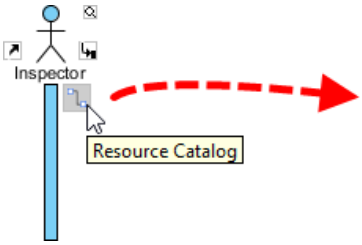
Create actor

Creating lifeline

To create lifeline, you can click **LifeLine** on the diagram toolbar and then click on the diagram.

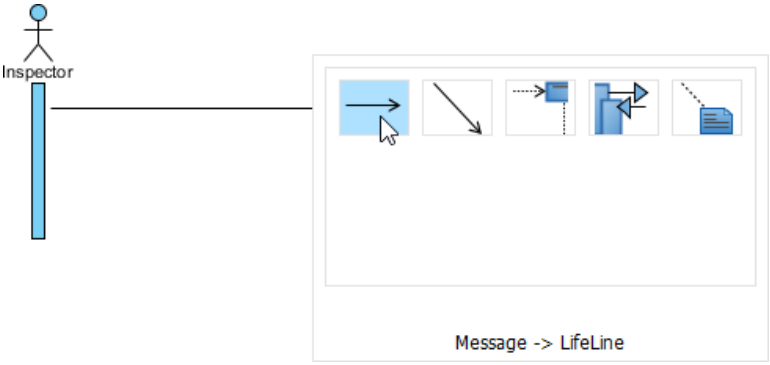
Alternatively, a much quicker and more efficient way is to use Resource Catalog:

1. Move your mouse pointer over the source lifeline.
2. Press on the **Resource Catalog** button and drag it out.



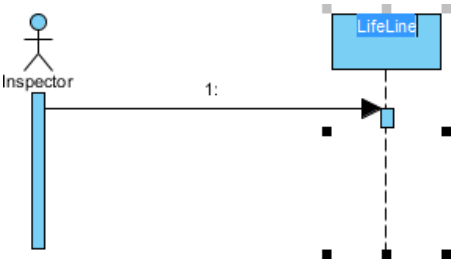
Using Resource Catalog

3. Release the mouse button at the place where you want the lifeline to be created.
4. Select **Message** -> **LifeLine** from Resource Catalog.



To create a lifeline

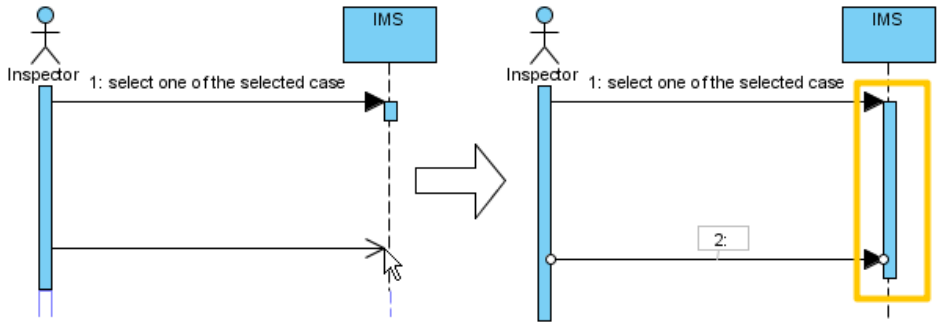
5. A new lifeline will be created and connected to the actor/lifeline with a message. Enter its name and press **Enter** to confirm editing.



Lifeline created

Auto extending activation

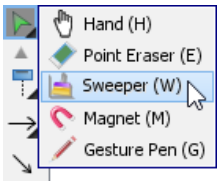
When create message between lifelines/actors, activation will be automatically extended.



Auto extending activation

Using sweeper and magnet to manage sequence diagram

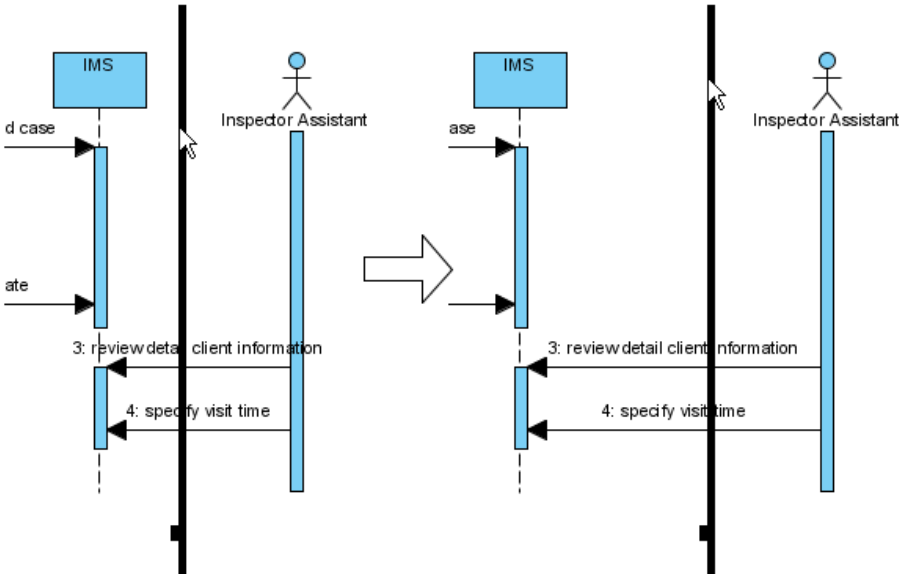
Sweeper helps you to move shapes aside to make room for new shapes or connectors. To use sweeper, click the **Selector** on the toolbar, then select **Sweeper**.



sweeper

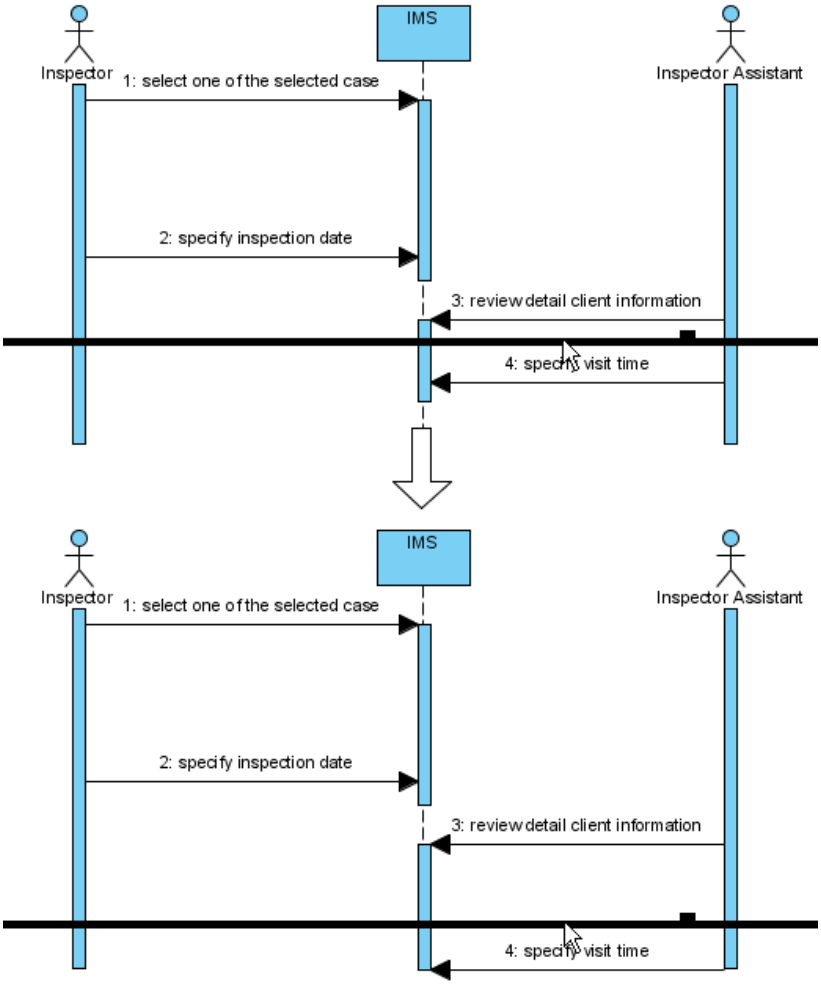
Click on empty space of the diagram and drag towards top, right, bottom or left. Shapes affected will be swept to the direction you dragged.

The picture below shows the actor *Inspector Assistant* is being swept towards right, thus new room is made for new lifelines.



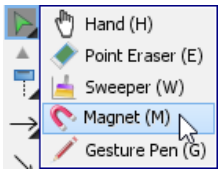
Sweep towards right

The picture below shows the message *specify visit time* is being swept downwards, thus new room is made for new messages.



Sweep downwards

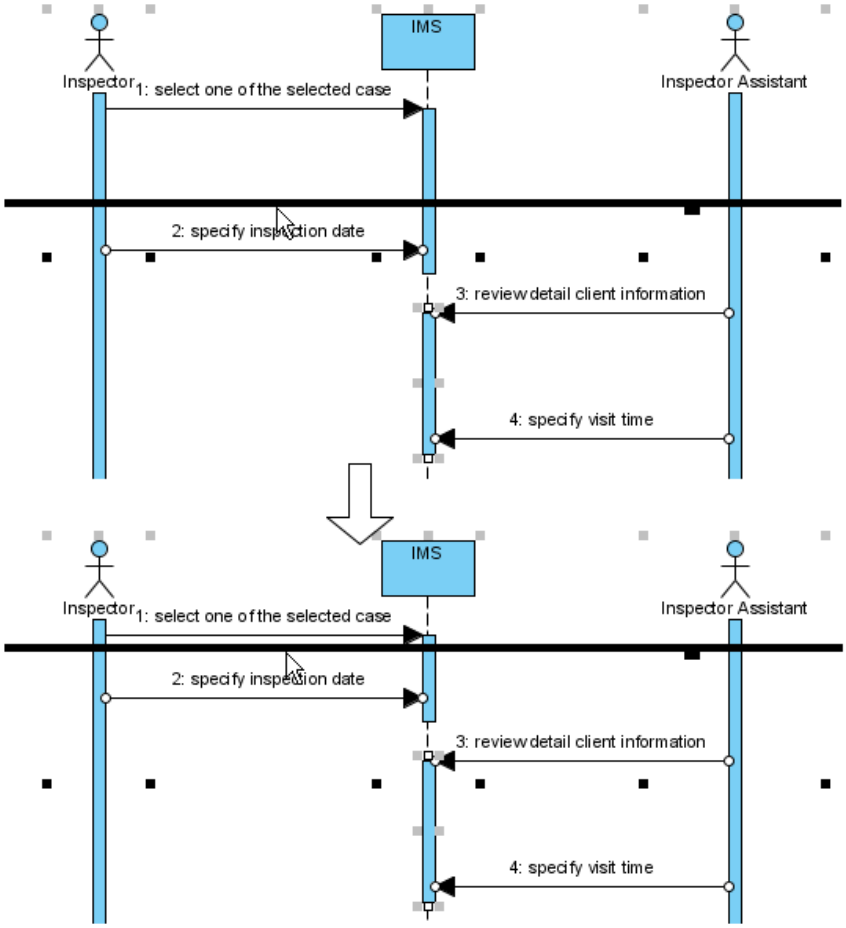
You can also use magnet to pull shapes together. To use magnet, click the **Selector** on the toolbar, then select **Magnet**.



Magnet

Click on empty space of the diagram and drag towards top, right, bottom or left. Shapes affected will be pulled to the direction you dragged.

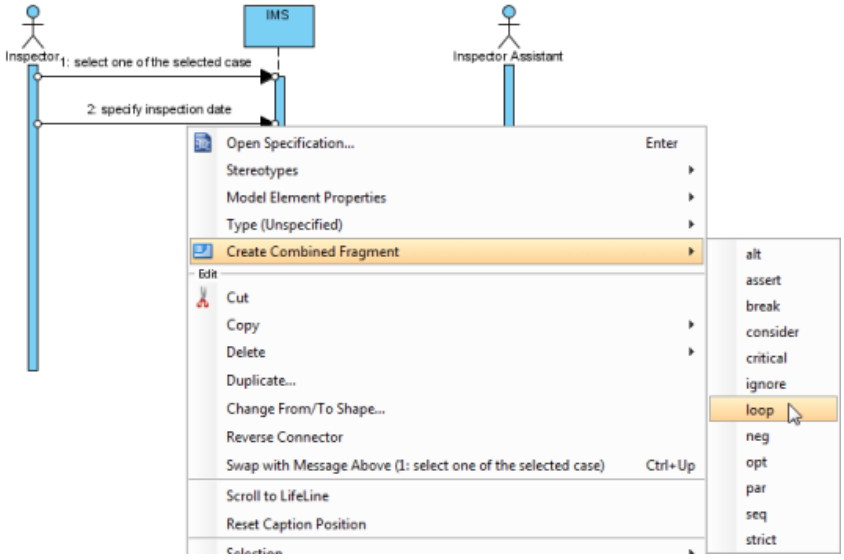
The picture below shows when drag the magnet upwards, shapes below dragged position are pulled upwards.



Pull shapes upwards using magnet

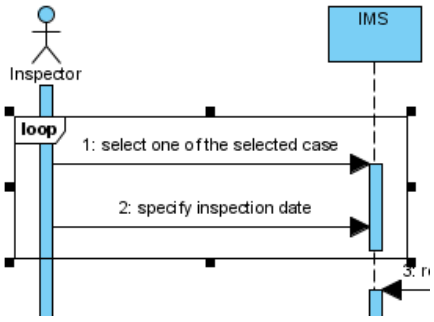
Creating combined fragment for messages

To create combined fragment to cover messages, select the messages, right-click on the selection and select **Create Combined Fragment** and then select a combined fragment type (e.g. loop) from the popup menu.



Create combined fragment for messages

A combined fragment of selected type will be created to cover the messages.

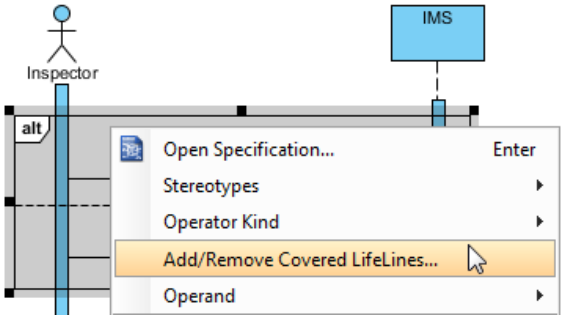


Combined fragment created

Adding/removing covered lifelines

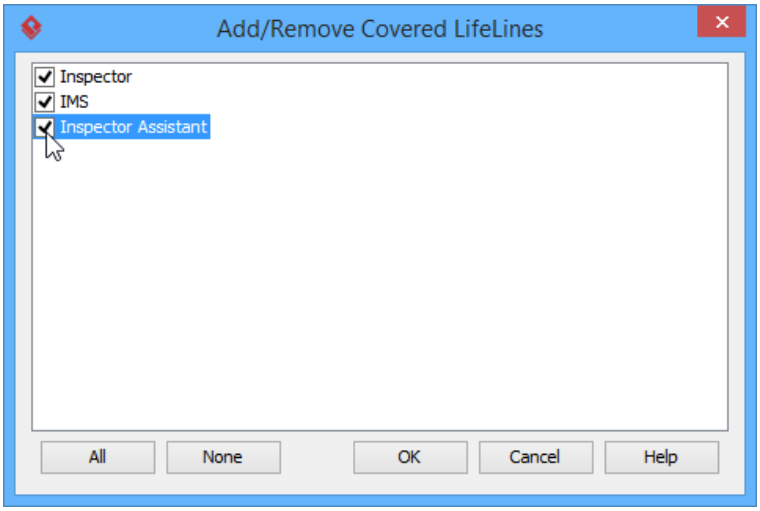
After you've created a combined fragment on the messages, you can add or remove the covered lifelines.

1. Move the mouse over the combined fragment and select **Add/Remove Covered Lifeline...** from the pop-up menu.



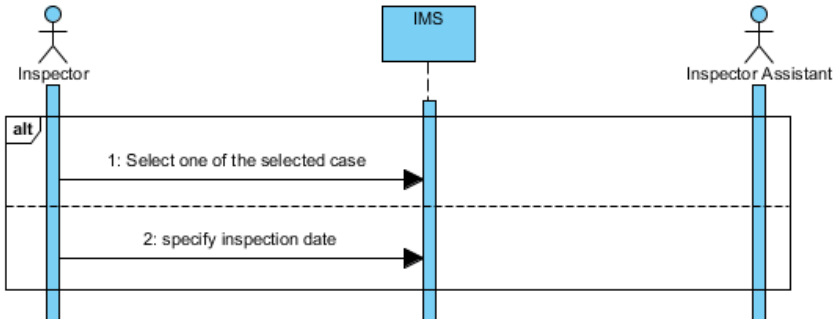
Add/Remove covered lifelines

2. In the **Add/Remove Covered Lifelines** dialog box, check the lifeline(s) you want to cover or uncheck the lifeline(s) you don't want to cover. Click **OK** button.



*Check **Inspector Assistant***

As a result, the area of covered lifelines is extended or narrowed down according to your selection.

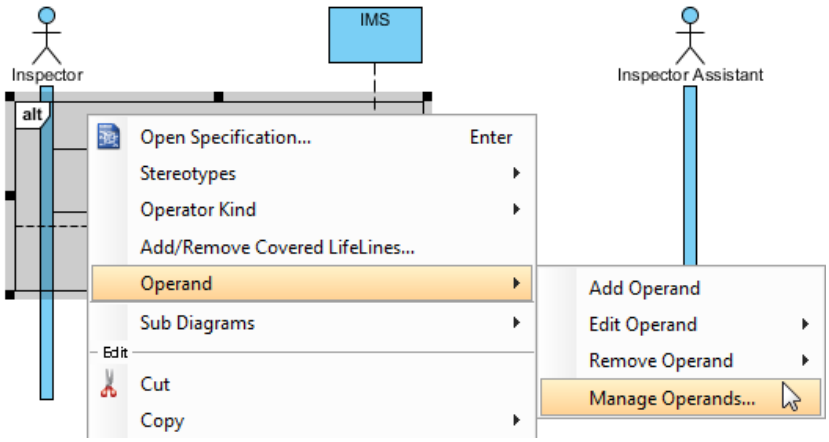


The area of covered lifelines is extended

Managing Operands

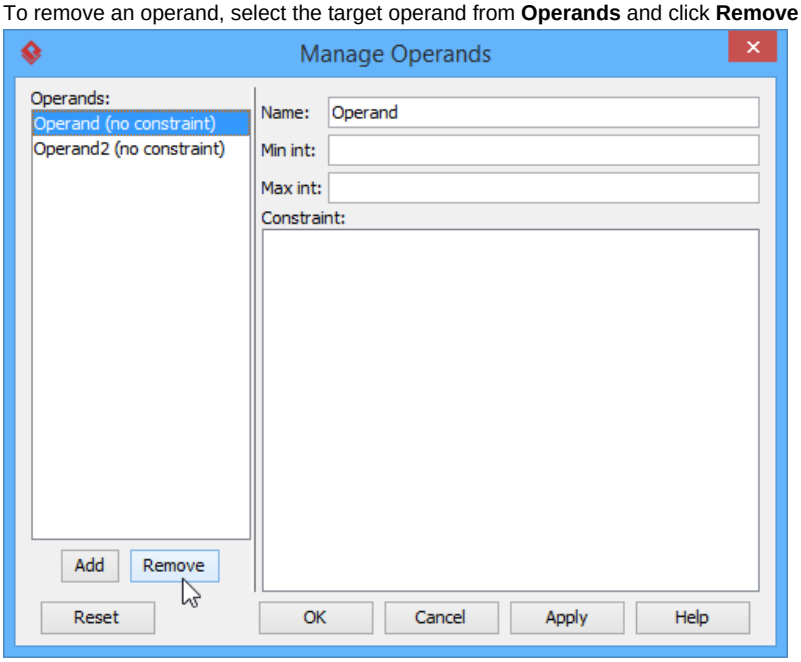
After you've created a combined fragment on the messages, you can also add or remove operand(s).

1. Move the mouse over the combined fragment and select **Operand > Manage Operands...** from the pop-up menu.



Manage operands

2. To remove an operand, select the target operand from **Operands** and click **Remove** button. Click **OK** button.

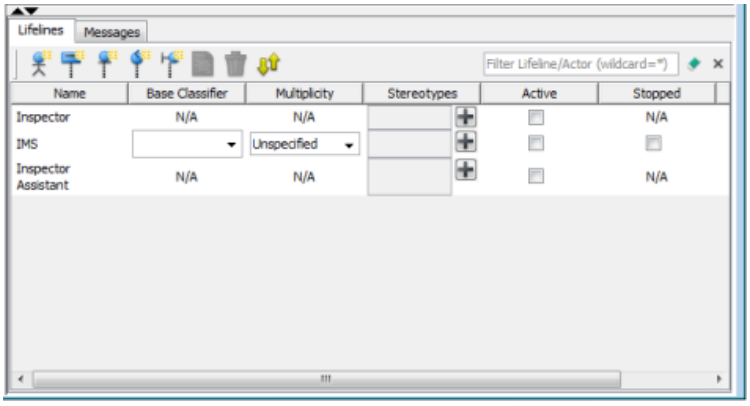


Remove Operand

Otherwise, click **Add** button to add a new operand and then name it. Click **OK** button.

Developing sequence diagram with quick editor or keyboard shortcuts

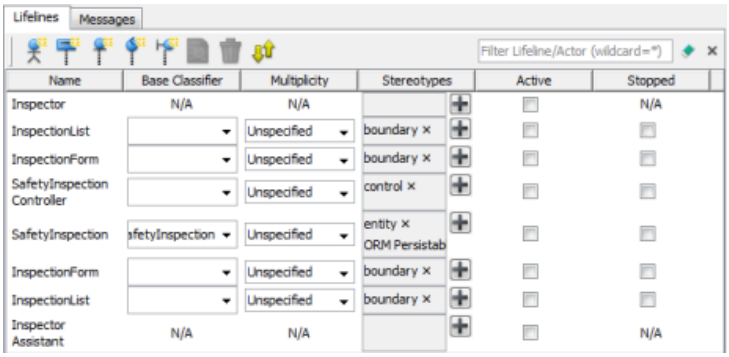
In sequence diagram, an editor appears at the bottom of diagram by default, which enables you to construct sequence diagram with the buttons there. The shortcut keys assigned to the buttons provide a way to construct diagram through keyboard. Besides constructing diagram, you can also access diagram elements listing in the editor.



The quick editor

Editing lifelines

There are two panes, **Lifelines** and **Messages**. The **Lifelines** pane enables you to create different kinds of actors and lifelines.



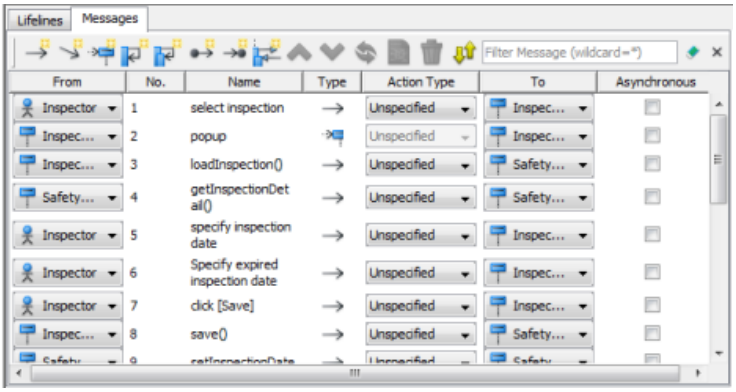
Lifelines pane in quick editor

Button	Shortcut	Description
	Alt-Shift-A	To create an actor
	Alt-Shift-L	To create a general lifeline
	Alt-Shift-E	To create an <<entity>> lifeline
	Alt-Shift-C	To create a <<control>> lifeline
	Alt-Shift-B	To create a <<boundary>> lifeline
	Alt-Shift-O	To open the specification of the element chosen in quick editor
	Ctrl-Del	To delete the element chosen in quick editor
	Ctrl-L	To link with the diagram, which cause the diagram element to be selected when selecting an element in editor, and vice versa

Buttons in Lifelines pane

Editing messages

The **Messages** pane enables you to connect lifelines with various kinds of messages.



Messages pane in quick editor

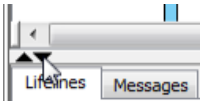
Button	Shortcut	Description
	Alt-Shift-M	To create a message that connects actors/lifelines in diagram
	Alt-Shift-D	To create a duration message that connects actors/lifelines in diagram
	Alt-Shift-C	To create a create message that connects actors/lifelines in diagram
	Alt-Shift-S	To create a self message on an actor/lifeline in diagram

	Alt-Shift-R	To create a recursive message on an actor/lifeline in diagram
	Alt-Shift-F	To create a found message that connects to an actor/lifeline
	Alt-Shift-L	To create a lost message from an actor/lifeline
	Alt-Shift-E	To create a reentrant message that connects actors/lifelines in diagram
	Ctrl-Shift-Up	To swap the chosen message with the one above
	Ctrl-Shift-Down	To swap the chosen message with the one below
	Ctrl-R	To revert the direction of chosen message
	Alt-Shift-O	To open the specification of the message chosen in quick editor
	Ctrl-Del	To delete the message chosen in quick editor
	Ctrl-L	To link with the diagram, which cause the message to be selected when selecting a message in editor, and vice versa

Buttons in Messages pane

Expanding and collapsing the editor

To hide the editor, click on the down arrow button that appears at the bar on top of the quick editor. To expand, click on the up arrow button.



Collapse the quick editor

Setting different ways of numbering sequence messages

You are able to set the way of numbering sequence messages either on diagram base or frame base.

Diagram-based sequence message

Right click on the diagram's background, select **Sequence Number** and then either **Single Level** or **Nested Level** from the pop-up menu.

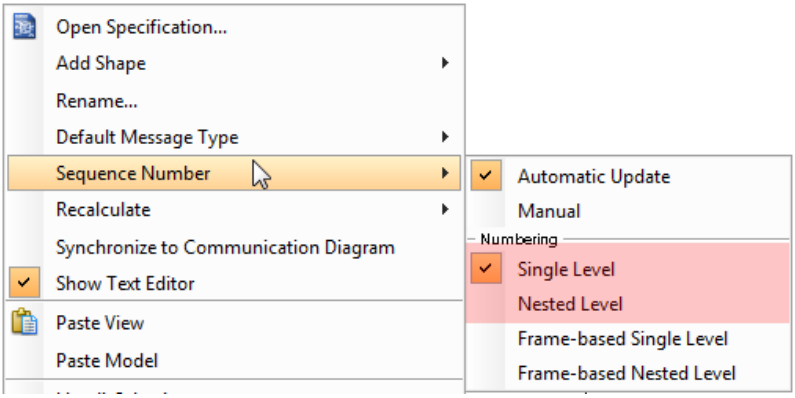
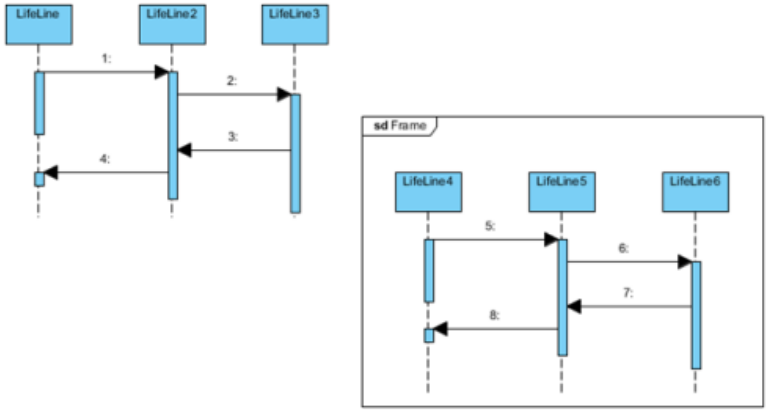


Diagram-based pop-up menu

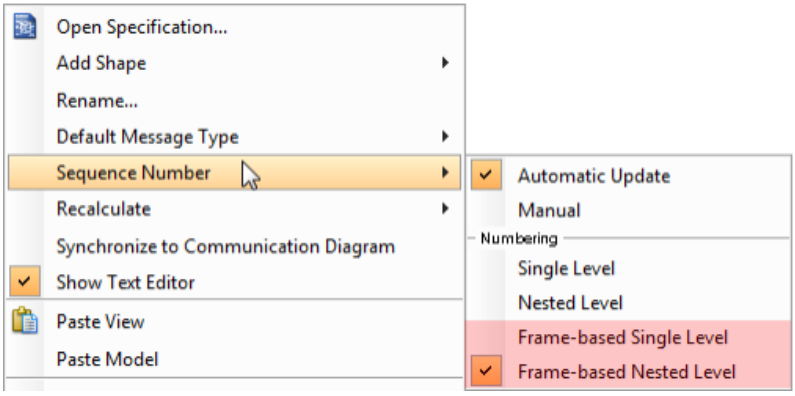
If you choose **Single Level**, all sequence messages will be ordered with integers on diagram base. On the other hand, if you choose **Nested Level**, all sequence messages will be ordered with decimal place on diagram base.



Single level

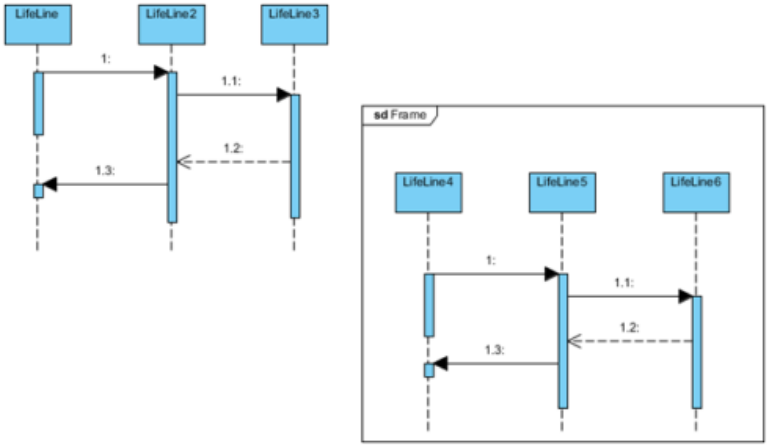
Frame-based sequence message

Right click on the diagram's background, select **Sequence Number** and then either **Frame-based Single Level** or **Frame-based Nested Level** from the pop-up menu.



Frame-based pop-up menu

When you set the way of numbering sequence messages on frame base, the sequence messages in frame will restart numbering sequence message since they are independent and ignore the way of numbering sequence message outside the frame.



Frame-based nested level

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Numbering sequence messages in Sequence Diagram
- Tutorial - Using duration constraint in sequence diagram
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Communication diagram

Communication diagram was designed for presenting the interactions between objects (i.e. lifelines). You will learn how to create communication diagram in this chapter.

Creating communication diagram

Learn how to create and draw communication diagram.

Drawing communication diagrams

Communication diagram is a kind of UML diagram that is designed for illustrating the dynamic view of the system. It emphasizes the structural organization of the objects' send and receive messages.

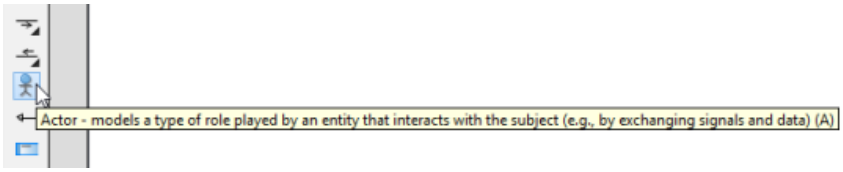
Creating communication diagram

Perform the steps below to create a UML communication diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Communication Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating actor

To create actor, click **Actor** on the diagram toolbar and then click on the diagram.



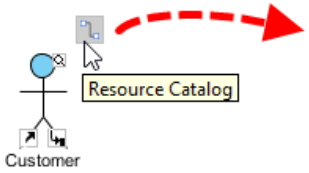
Create actor

Creating lifeline

To create lifeline, you can click **LifeLine** on the diagram toolbar and then click on the diagram.

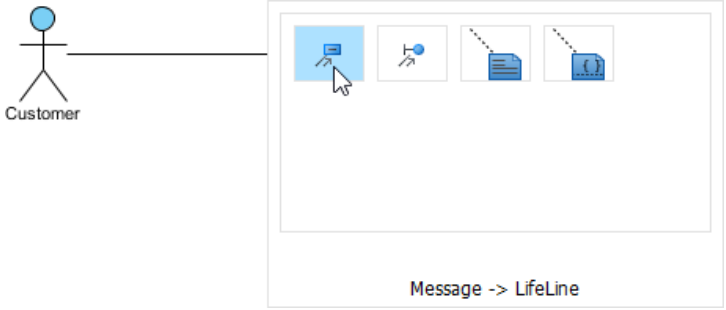
Alternatively, a much quicker and more efficient way is to use Resource Catalog:

1. Move your mouse pointer over the source lifeline.
2. Press on the **Resource Catalog** button and drag it out.



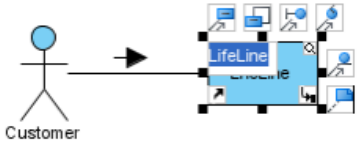
Using Resource Catalog

3. Release the mouse button at the place where you want the lifeline to be created.
4. Select **Message** -> **LifeLine** from Resource Catalog.



To create a lifeline

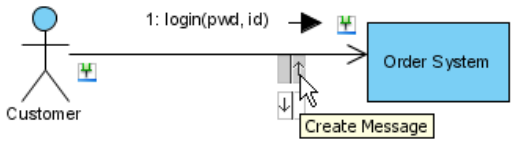
5. A new lifeline will be created and connected to the actor/lifeline with a message. Enter its name and press **Enter** to confirm editing.



Lifeline created

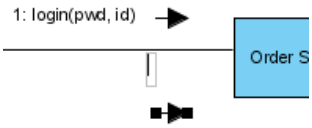
Creating message on link

To create message on link, click its **Create Message** resource.



Create message on link

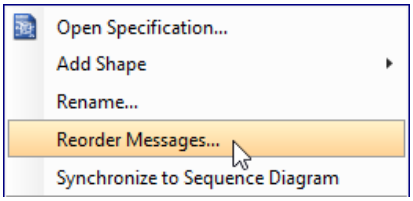
A message will be created on the link.



Message created on link

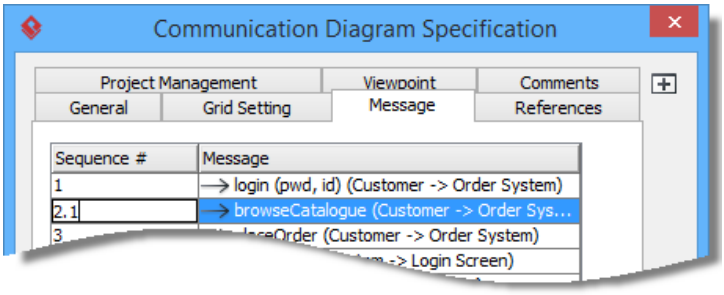
Editing sequence number of messages

To edit sequence number of messages, for example, to show certain messages are in nested level of interaction, right-click the diagram and select **Reorder Messages ...** from the pop-up menu.



Reorder messages

When the **Communication Diagram Specification** window appears, the **Message** tab is opened by default. Double click on the **Sequence #** cell of a message to edit it. Click **OK** button to apply the changes.



Edit sequence number of messages

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

State machine diagram

State machine diagram shows flow of control from state to state within single object. You will learn how to create a state machine diagram in this chapter.

Creating state machine diagram

Learn how to create state machine diagram and configure state properties.

Creating state machine diagrams

State machine diagram is a kind of UML diagram that shows flow of control from state to state within single object. It usually contains simple states, composite states, composite states, transitions, events and actions.

Creating state machine diagram

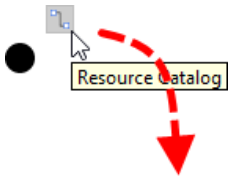
Perform the steps below to create a UML state machine diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **State Machine Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating states and transitions

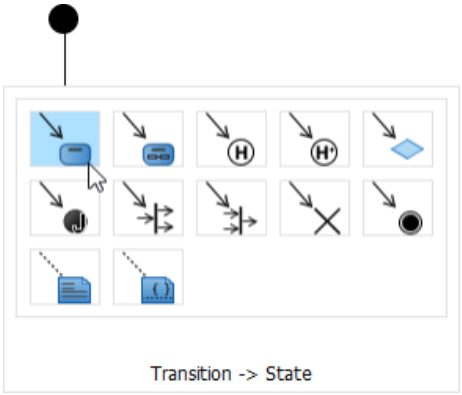
After creating a state machine diagram, an initial pseudo state appears by default. You can create other states by using Resource Catalog:

1. Move your mouse pointer over the source state.
2. Press on the **Resource Catalog** button and drag it out.



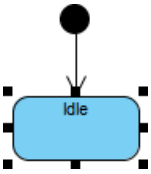
Using Resource Catalog

3. Release the mouse button at the place where you want the state to be created.
4. Select the state to be created from Resource Catalog.



To create a state

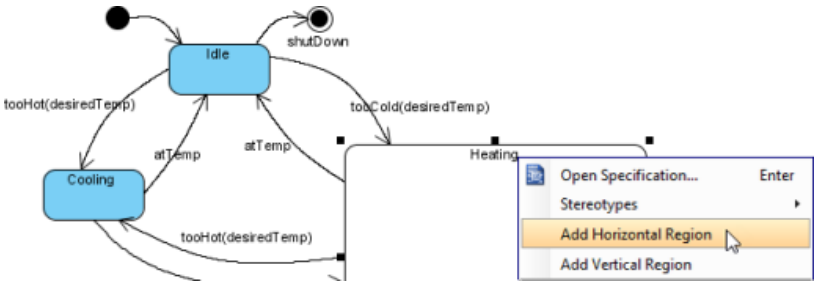
5. A new state will be created and is transited from the source state. Enter its name and press **Enter** to confirm editing.



State created

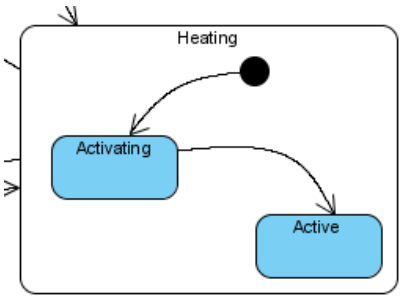
Adding region to state

To model substates of a composite state, you need to add one or more regions to it. To add a region, right-click the state and select **Add Horizontal Region** from the popup menu.



Add region to state

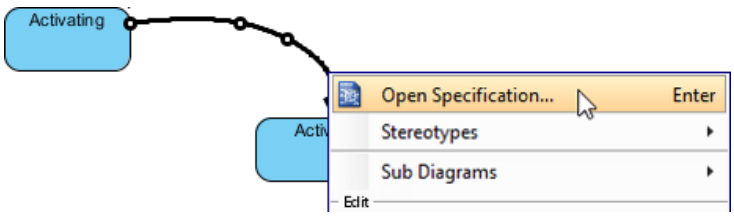
Next, you can draw the substates inside the region.



Substates in a composite state

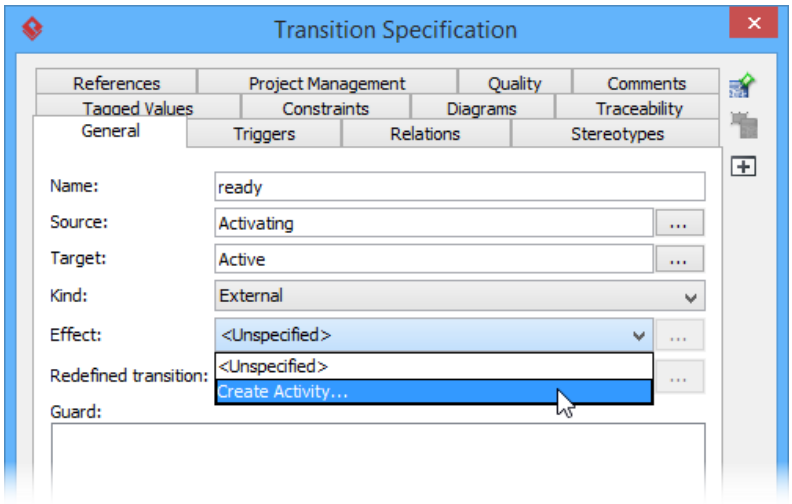
Modeling properties of transition

To model properties of transition such as effect and guard, right-click the transition and select **Open Specification...** from the pop-up menu.



Open specification of transition

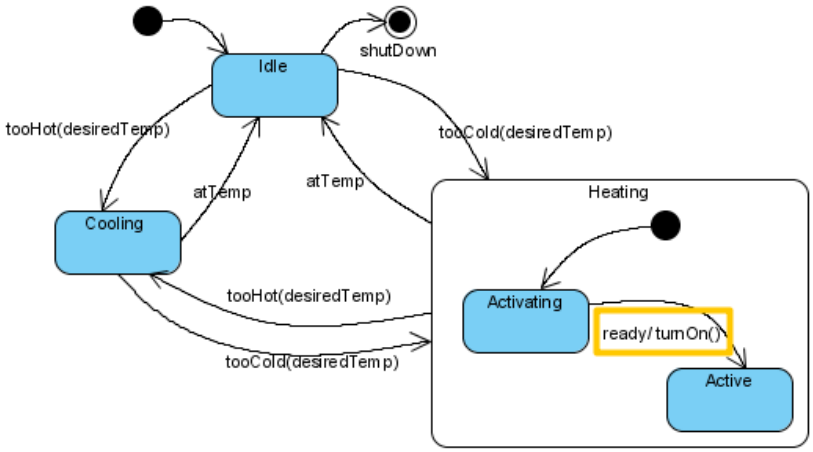
When the **Transition Specification** pops out, you can edit its name, effect and guard. Next, select **Create Activity...** from the **Effect** property.



Create Activity from transition

In **Activity Specification (Effect)** window, change its name and then click **OK** button to apply the change.

Click **OK** in the **Transition Specification** to close it. The name and effect are shown on the transition caption.



Name and effect shown in caption of transition

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Activity diagram

Activity diagram is a flowchart-based diagram showing flow of control from activity to activity. It shows concurrency, branch, control flow and object flow. You will learn how to create activity diagram in this chapter.

Creating activity diagram

Learn how to create activity diagram.

Drawing activity diagrams

Activity diagram is a kind of UML diagram that shows flow of control from activity to activity. It shows concurrency, branch, control flow and object flow. Furthermore, swimlane is used for partitioning actions based on the participants involved.

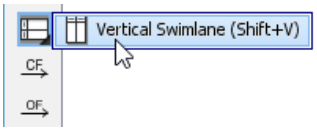
Creating activity diagram

Perform the steps below to create a UML activity diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Activity Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

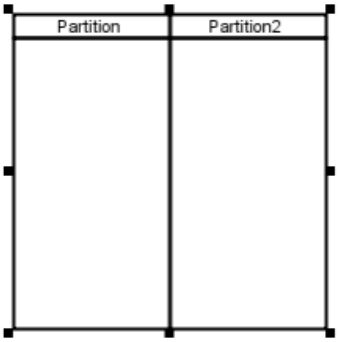
Creating swimlane

You can click either **Horizontal Swimlane** or **Vertical Swimlane** on the diagram toolbar.



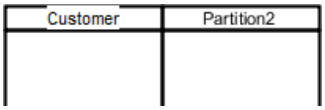
Create swimlane

Click on the diagram to create the swimlane.



Swimlane created

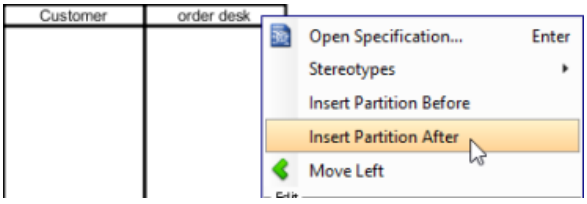
Double-click the partition name to rename it.



Rename partition

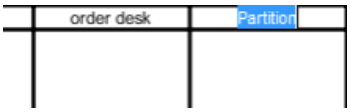
Inserting partition to swimlane

To insert partition to swimlane, right-click on a partition and select either **Insert Partition Before** or **Insert Partition After** from the pop-up menu.



Insert partition to swimlane

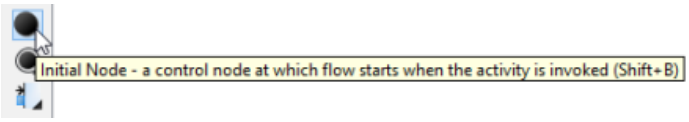
A partition is inserted.



Partition inserted

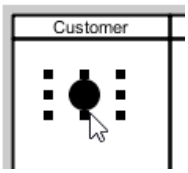
Creating initial node

Click **Initial Node** on the diagram toolbar.



Create initial node

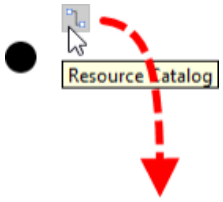
Click inside the partition to create the initial node there.



Initial node created

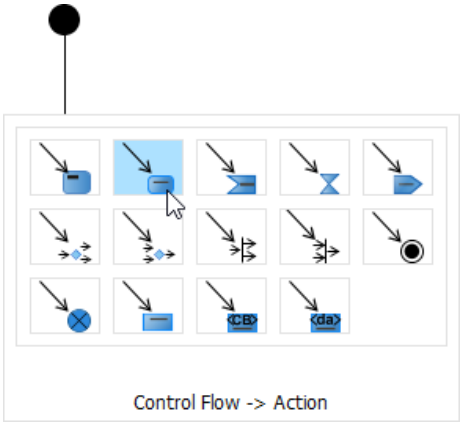
Creating action

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.



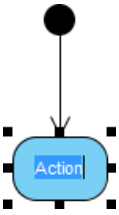
Using Resource Catalog

3. Release the mouse button at the place where you want the action to be created.
4. Select **Control Flow** -> **Action** from Resource Catalog.



To create a action

5. A new action will be created and is connected to the source shape with a control flow. Enter its name and press **Enter** to confirm editing.



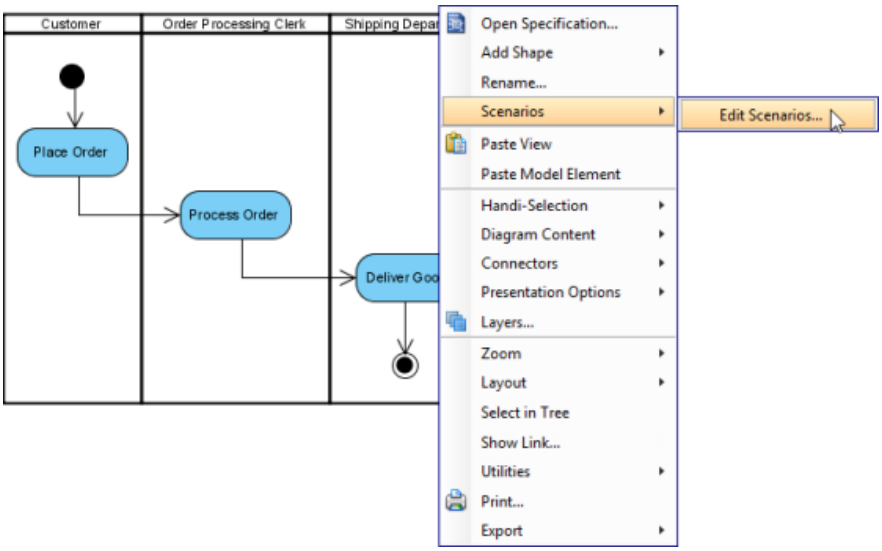
Action created

Working with scenario

A scenario is a diagram formed by the internal interaction of a sequence of action, modeled by their sub-diagrams. With scenario, you can produce a diagram which presents an overview of an execution path in activity diagram, so as to know how user and system communicate with each other in order to complete the flow.

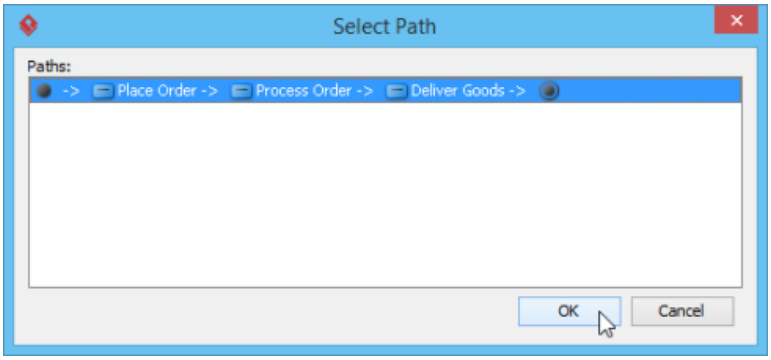
Producing scenario from activity diagram

1. Right click on the activity diagram that contains the flows that you want to produce a scenario and select **Scenarios > Edit Scenarios...** from the popup menu.



Edit scenarios

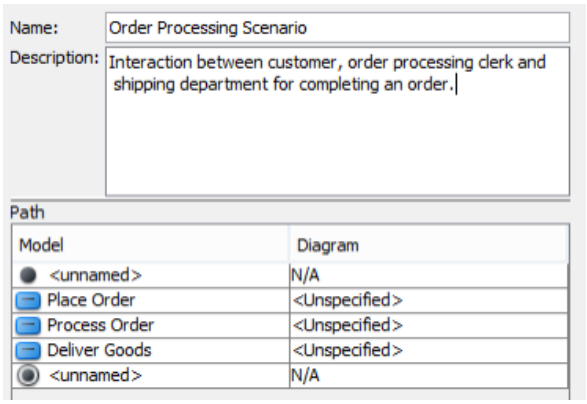
2. In the **Edit Scenarios** dialog box, click **Add...** button at the bottom left corner.
3. Select a path for generating scenario. Click **OK** to confirm.



Select a path for generating scenario

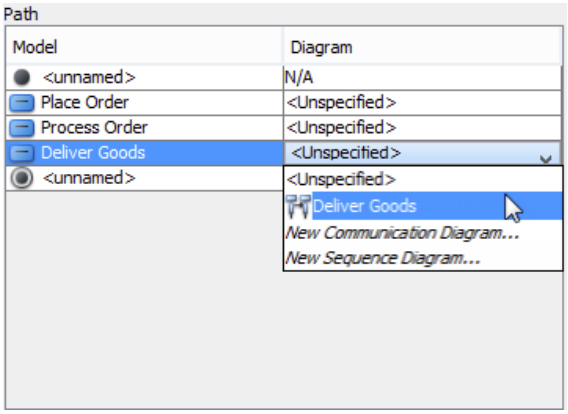
> **NOTE:** A path is a continuous flow of actions in the diagram, with an initial node placed at the beginning of the actions. Multiple paths are obtained by determining the existence of decision nodes within the flow.

4. Name the scenario. Add description if necessary.



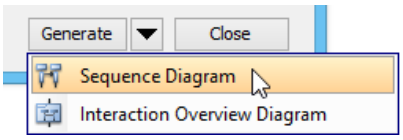
Name and describe scenario

5. The actions being involved in the flow are listed in the **Path** table. For actions that have sub-diagram(s), pick up the sub-diagram in **Diagram** column or just create a new one. You may, however, leave it unspecified which cause that action to be ignored when producing scenario.



Select diagram for action

6. Click on the arrow beside the **Generate** button and select the type of diagram of the scenario.



Generate scenario with specific diagram type

Updating scenario

Whenever the sub-diagram(s) of action(s) are updated, you can update the scenario to make it represents the latest information of interaction. To update scenario, right click on the activity diagram that have scenario produced before, select **Scenarios**, then the name of scenario from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

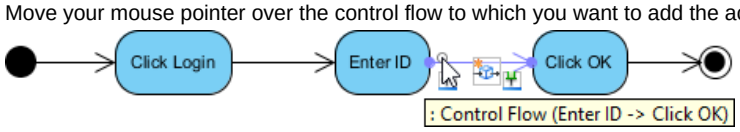
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Splitting control flow in activity diagram

When you found an action missing and you want to add it back into an activity diagram, you can make use of the split feature to easily insert the action shape back to a control flow. The insertion of action shape will result in the creation of new flow that connects the new action shape and the "to-shape" that is originally connected by the original flow. The original flow will be updated to connect to the new shape. In other words, the details specified to the original flow, if any, will remain intact.

To use the split resource:

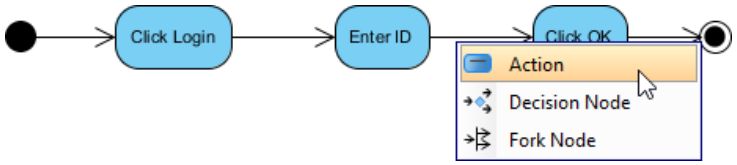
1. Move your mouse pointer over the control flow to which you want to add the action shape.



To split a control flow

2. Click on  .

3. Select **Action** in the popup menu. You may also add a decision node and fork node into the flow.

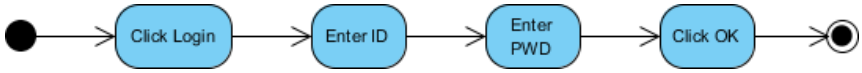


Adding an action into a control flow

4. Enter the name of the action and press **Enter** to confirm.

Entering the name of new action shape

5. Tidy up the flow.



A flow with action shape added

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Component diagram

Component diagram shows the physical aspect of an object-oriented software system. You will learn how to create component diagram in this chapter.

Creating component diagram

Learn how to create component diagram.

Drawing component diagrams

Component diagram is a kind of UML diagram . shows the physical aspect of an object-oriented software system. It illustrates the architectures of the software components and dependencies between them.

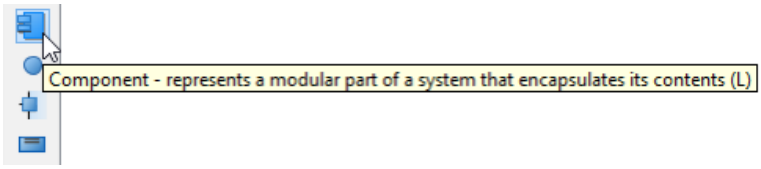
Creating component diagram

Perform the steps below to create a UML component diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Component Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating component

To create component in component diagram, click **Component** on the diagram toolbar and then click on the diagram.



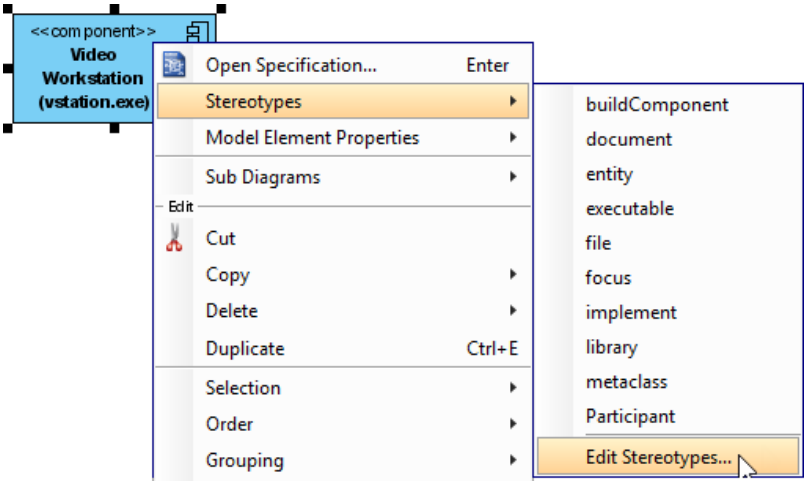
Create component

A component will be created.

Component created

Assigning stereotypes

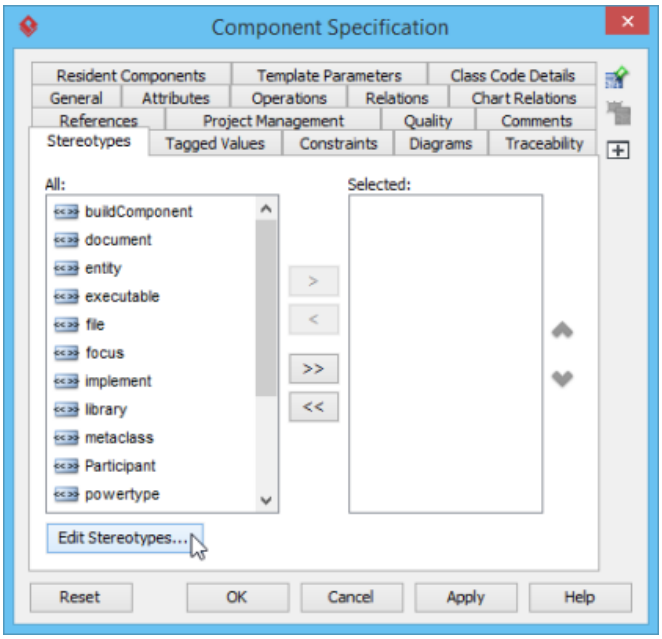
Right click on the package and select **Stereotypes** > **Edit Stereotypes**... from the pop-up menu.



Assign stereotypes

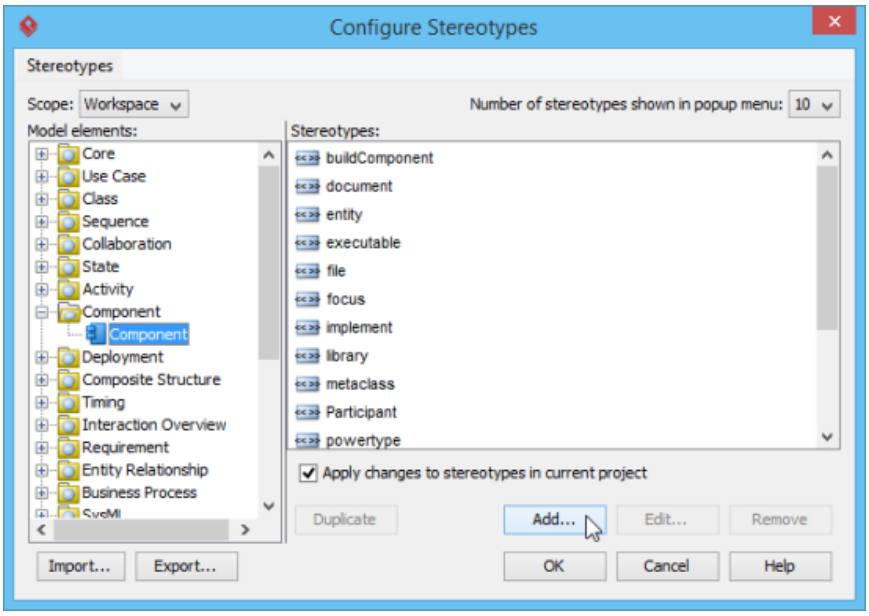
When the **Component Specification** dialog box pops out, the **Stereotypes** tab is opened by default. The list on the left shows the selectable stereotypes.

If the stereotype you want to use is not on the list, click **Edit Stereotypes...** button.



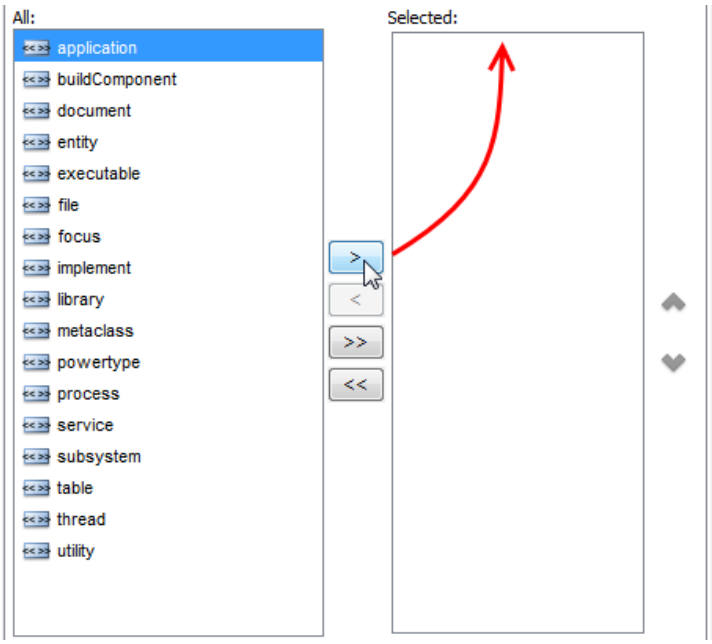
Edit stereotypes

Click **Add...** button in the **Configure Stereotypes** dialog box.



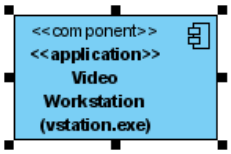
Add stereotype

Name the stereotype (e.g. *application*) in the **Stereotype Specification** dialog box and then click **OK** button to close it. Click **OK** button in the **Configure Stereotypes** dialog box. The added stereotype will then be shown on the list in the **Component Specification** dialog box. Select it and click **Add Selected** button. Finally, click **OK** button to confirm.



Add selected stereotypes

Close the specification dialog box. Stereotypes will be applied to the package.

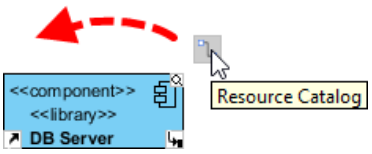


Stereotypes assigned

Creating provided interface

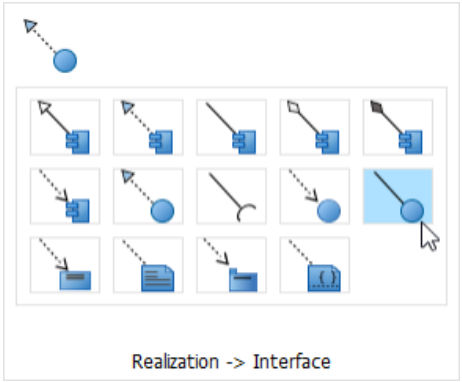
To create provided interface for a component:

1. Move your mouse pointer over the source component.
2. Press on the **Resource Catalog** button and drag it out.



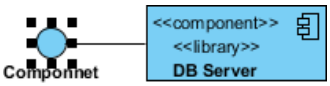
Using Resource Catalog

3. Release the mouse button at the place where you want the interface to be created.
4. Select **Realization** -> **Interface** from Resource Catalog.



To create a provided interface

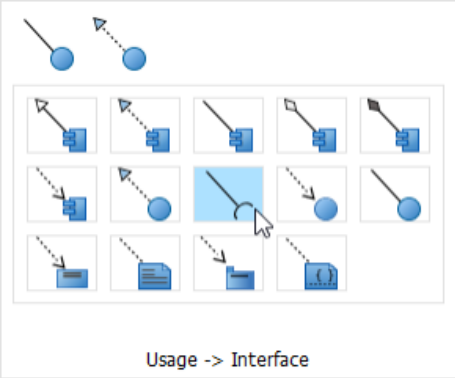
5. A new interface will be created and is connected to the source component. Enter its name and press **Enter** to confirm editing.



Interface created

Creating required interface

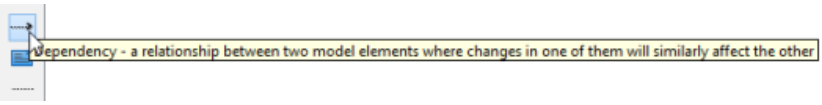
To create required interface for a component, just follow the steps described above for creating provided interface, but change to select **Usage**-> **Interface** in Resource Catalog.



Create required interface

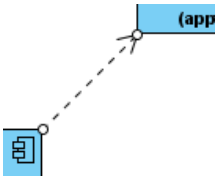
Creating dependency

To create dependency, click **Dependency** on the diagram toolbar.



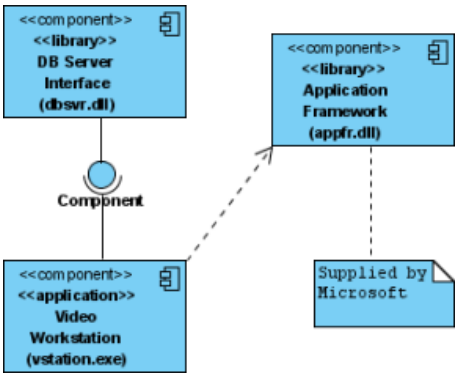
Create dependency

Drag from the source shape, move the mouse over the target shape and then release the mouse button to create the dependency.



Dependency created

Continue to complete the diagram



Completed diagram

Showing/hiding attributes in component

Per diagram

You can add attributes to a component. To show/hide the attributes for all components in a diagram:

1. Right click on the background of the component diagram.
2. Select **Presentation Options > Component Display Options** from the popup menu.
3. Select/De-select **Show Attributes** to cause attributes to be shown or hidden.

Per component

You can add attributes to a component. To show/hide the attributes for a specific component:

1. Right click on the desired component.
2. Select **Presentation Options > Show Attributes Mode** from the popup menu.

3. Select **Follow Diagram/Show All/Hide All/Customized...** from the popup menu. If you have selected the **Customized** option, you can select the specific attribute(s) to be shown or hidden.

Showing/hiding operations in component

Per diagram

You can add operations to a component. To show/hide the operations for all components in a diagram:

1. Right click on the background of the component diagram.
2. Select **Presentation Options > Component Display Options** from the popup menu.
3. Select/De-select **Show Operations** to cause attributes to be shown or hidden.

Per component

You can add operations to a component. To show/hide the operations for a specific component:

1. Right click on the desired component.
2. Select **Presentation Options > Show Operations Mode** from the popup menu.
3. Select **Follow Diagram/Show All/Hide All/Customized...** from the popup menu. If you have selected the **Customized** option, you can select the specific operation(s) to be shown or hidden.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Deployment diagram

Deployment diagram shows the physical aspects of an object-oriented system. You will learn how to create a deployment diagram in this chapter.

Creating deployment diagram

Learn how to create deployment diagram.

Drawing deployment diagrams

Deployment diagram is a kind of UML diagram that shows the physical aspects of an object-oriented system. It also shows the configuration of run time processing nodes and artifacts.

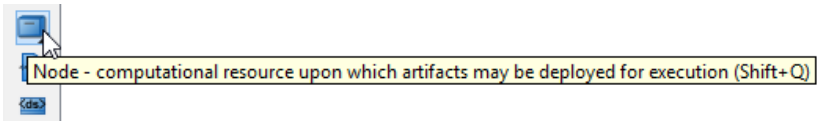
Creating deployment diagram

Perform the steps below to create a UML deployment diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Deployment Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating node

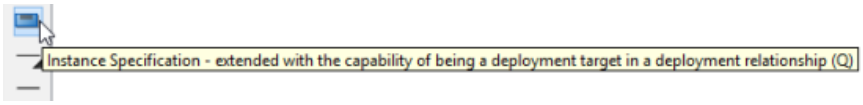
To create node in deployment diagram, click **Node** on the diagram toolbar and then click on the diagram.



Node selected

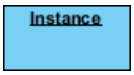
Creating instance of node

To create instance of node, click **Instance Specification** on the diagram toolbar and then click on the diagram.



Create instance specification

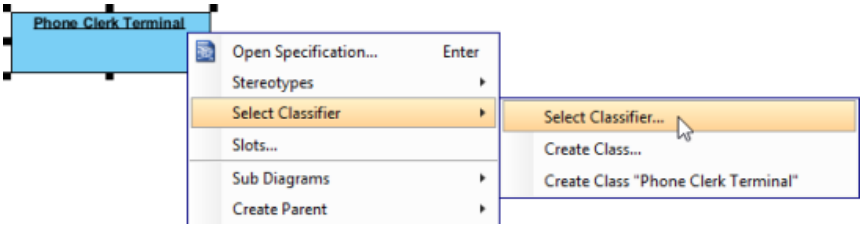
An instance specification will be created.



Instance specification created

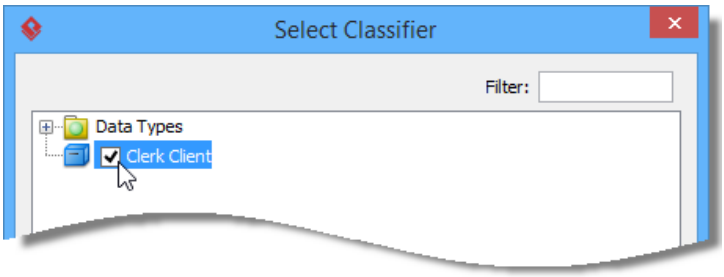
Selecting classifiers

To specify classifiers for an instance specification, right-click it and select **Select Classifier** > **Select Classifier...** from the pop-up menu.



Select classifier

When the **Instance Specification Specification** dialog box pops out, the **Classifiers** tab is opened by default. Click **Add...**. Then, select the classifier(s) in the popup window and click **OK**.



Select node

Click **OK** button to close the specification dialog box. The selected classifiers are assigned to the instance specification.

Classifiers assigned

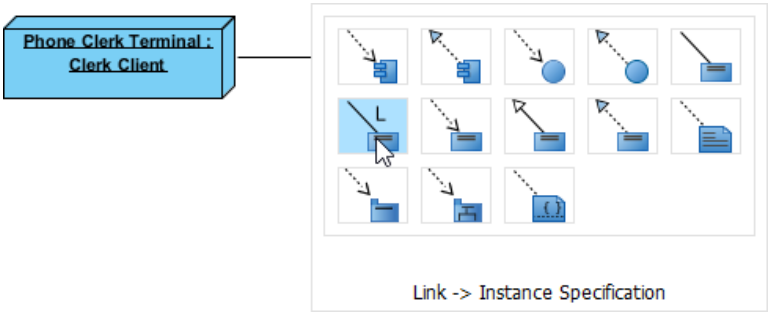
Creating link

To create link from instance specification:

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.

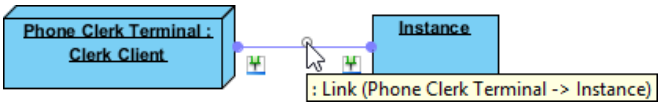
Using Resource Catalog

3. Release the mouse button at the place where you want the instance specification to be created.
4. Select **Link -> Instance Specification** from Resource Catalog.



To create an instance specification

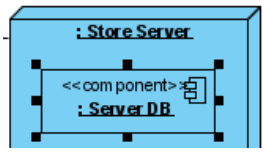
5. A new instance specification will be created and is connected to the source shape. Enter its name and press **Enter** to confirm editing.



Instance specification created

Creating instance of component

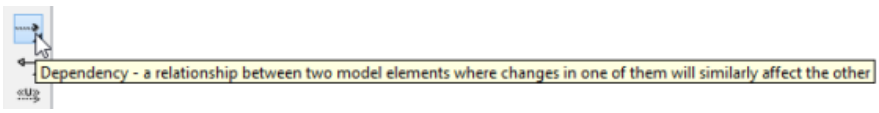
Similar to creating instance of node, you first create a component model element and then create an instance specification. However, this time assigns a component to the instance specification as classifier. After that the instance specification will be displayed as a component.



Instance of component

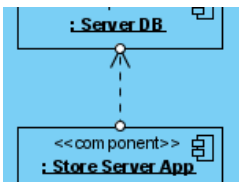
Creating dependency

To create dependency, click **Dependency** on the diagram toolbar.



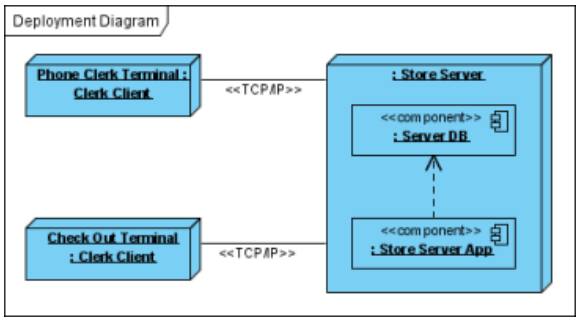
Create dependency

Drag from the source shape, move the mouse over the target shape and then release the mouse button to create the dependency.



Dependency created

Continue to complete the diagram.



Completed diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Package diagram

Package diagram shows the arrangement and organization of model elements in middle to large scale project. You will learn how to create package diagram in this chapter.

Creating package diagram

Learn how to create package diagram.

Drawing package diagrams

Package diagram is a kind of UML diagram that shows the arrangement and organization of model elements in middle to large scale project. It can show both structure and dependencies between sub-systems or modules.

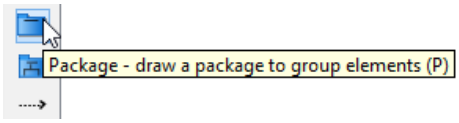
Creating package diagram

Perform the steps below to create a UML package diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Package Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

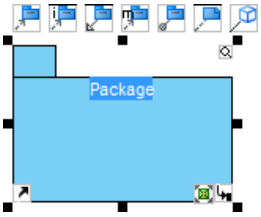
Creating package

To create package in package diagram, click **Package** on the diagram toolbar and then click on the diagram.



Create package

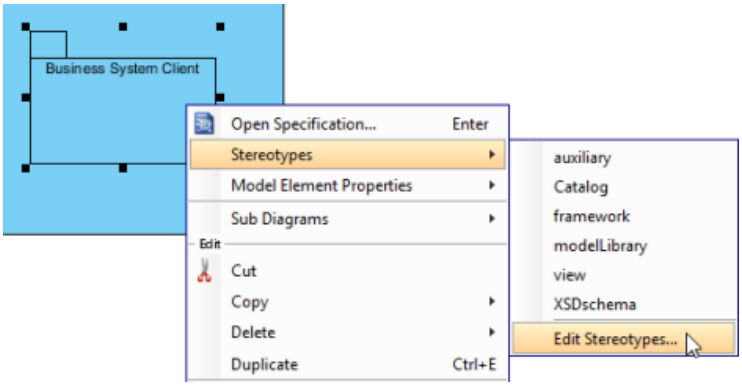
A package will be created.



Package created

Assigning stereotypes

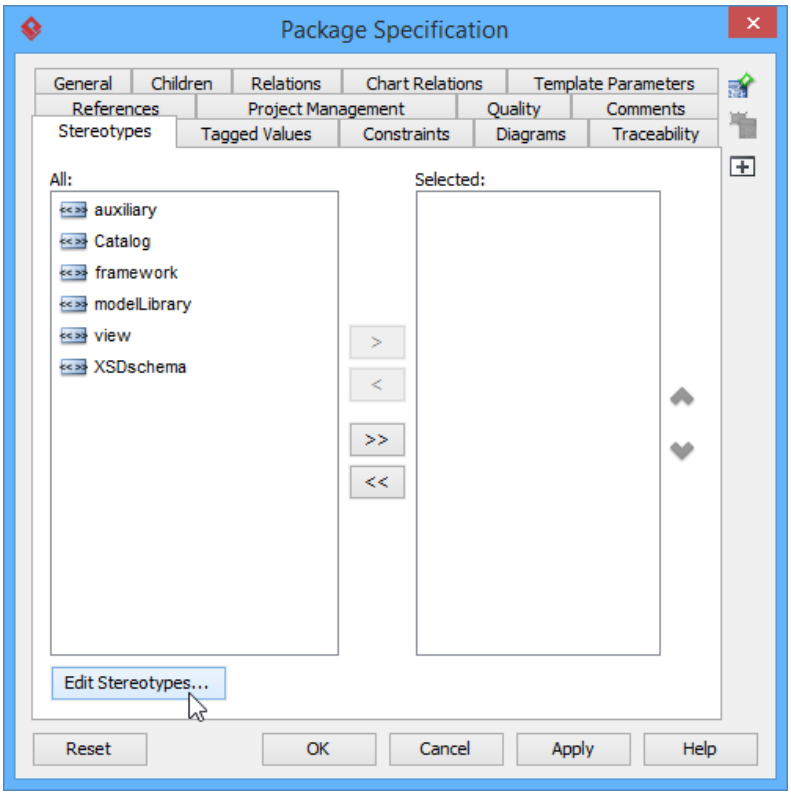
Right click on the package and select **Stereotypes** > **Edit Stereotypes**... from the pop-up menu.



Assign stereotypes

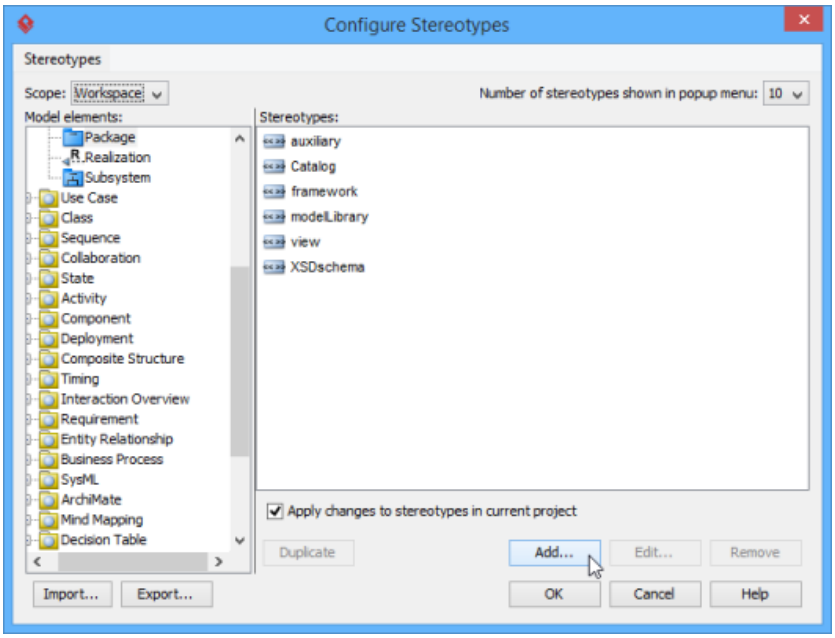
When the **Package Specification** dialog box pops out, the **Stereotypes** tab is opened by default. The list on the left shows the selectable stereotypes.

If the stereotype you want to use is not on the list, click **Edit Stereotypes...** button.



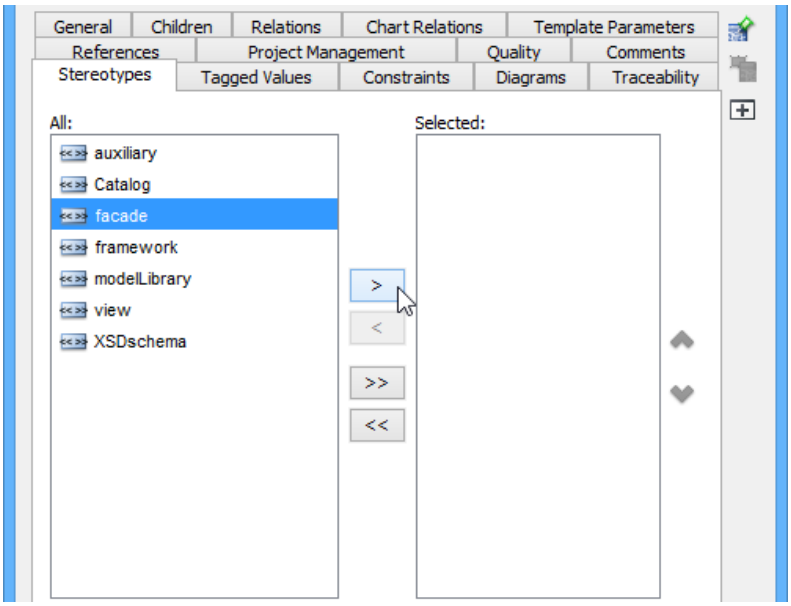
Edit stereotypes

Click **Add...** button in the **Configure Stereotypes** dialog box.



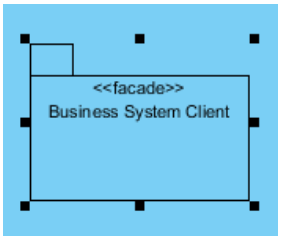
Add stereotype

Enter name for the new stereotype (e.g. *facade*). Click **OK** button in **Stereotype Specification** dialog box and the **Configure Stereotypes** dialog box. You will see the added stereotype appears on the list in **Package Specification** dialog box. Select it and click **Add Selected** button. Next, click **OK** button to proceed.



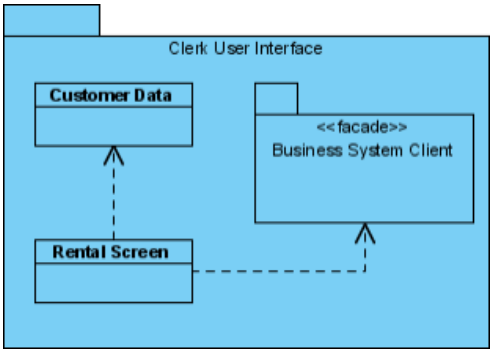
Add selected stereotypes

Close the specification dialog box. Stereotypes will be applied to the package.



Stereotypes assigned

Continue to complete the diagram.



Completed diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - UML Package Diagram
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Object diagram

Object diagram shows a snapshot of instances of things in class diagram. You will learn how to create an object diagram in this chapter.

Creating object diagram

Learn how to create an object diagram.

Drawing object diagrams

Object diagram is a kind of UML diagram that shows a snapshot of instances of things in class diagram. Similar to class diagram, it shows the static design of system from the real or prototypical perspective.

Creating object diagram

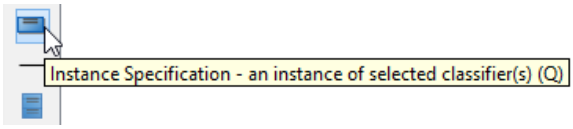
Perform the steps below to create a UML object diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Object Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating instance specification

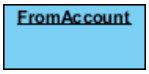
To create instance specification in object diagram:

1. Select **Instance Specification** from the diagram toolbar.



Create instance specification

2. Click on the diagram to create an instance specification shape. Name it.

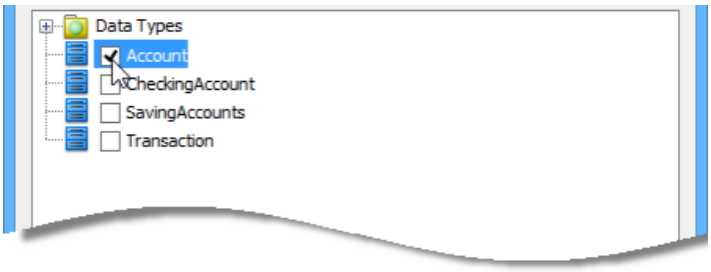


Instance specification created

Selecting classifiers

To specify classifiers for an instance specification:

1. Right-click on the desired instance specification shape and select **Select Classifier** > **Select Classifier...** from the pop-up menu.
2. This opens the **Classifiers** tab. Click **Add...** in it.
3. In the **Select Classifier** window, select the class(es) to be the classifier of the instance specification. If you are referencing another project, you can select its model element to be the classifier. Just change the **from project** selection at the top of the window.



Selecting classifier

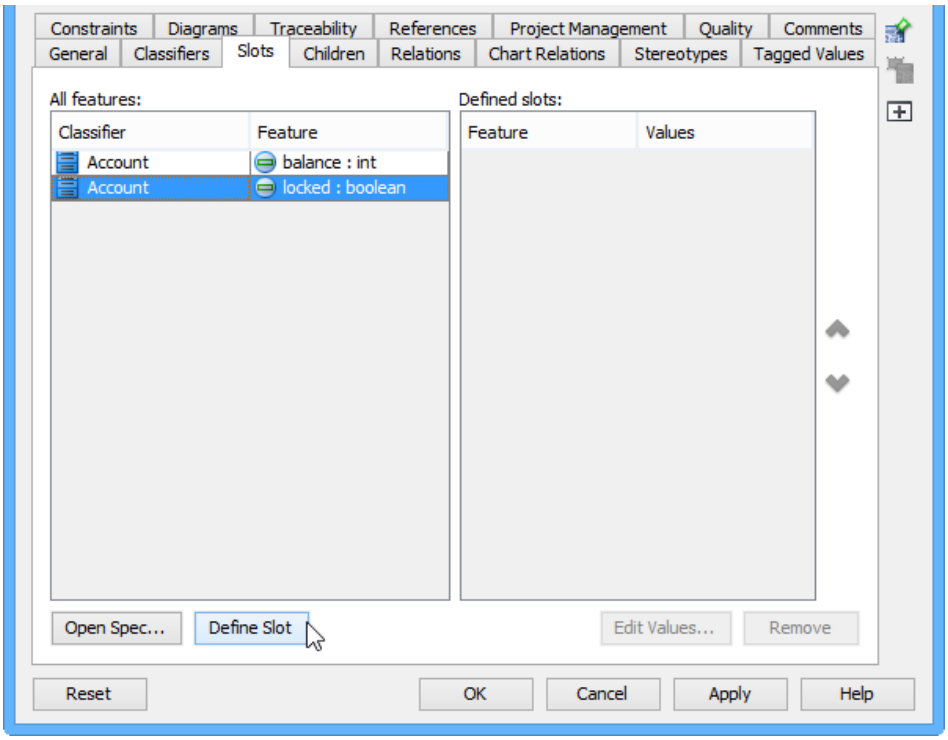
4. Click **OK** to return to the **Instance Specification Specification** window.
5. Click **OK** to return to the diagram.

Classifier selected

Defining slots

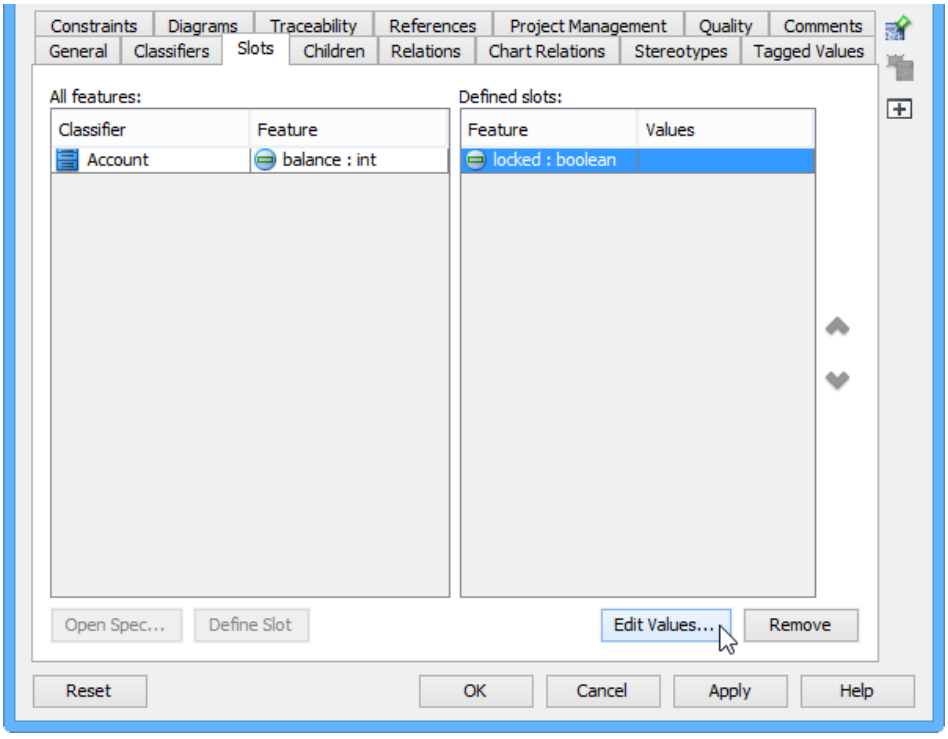
To define slots for an instance specification:

1. Right-click on the desired instance specification shape and select **Slots. ..** from the pop-up menu.
2. The **Instance Specification Specification** dialog box appears with the **Slots** tab selected. Select the features that you want to define slots on the left and click **Define Slot**.



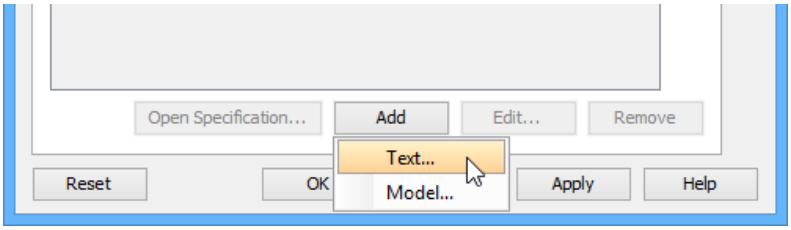
Defining slot

3. Select a defined slot and click **Edit Values...** at bottom right.



Edit values

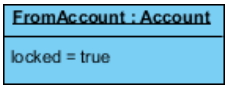
4. The **Slot Specification** window pops out, the **Values** tab is opened by default. Click **Add** button and select **Text** from the pop-up menu.



Add values to defined slot

5. Enter the slot value and click **OK** to confirm.

6. Click **OK** again in the **Instance Specification Specification** window to return to the diagram.

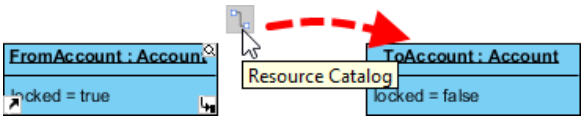


Instance specification with slot defined

Creating link

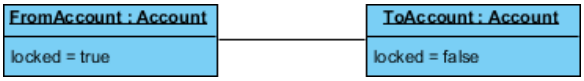
To create link between instance specifications:

1. Move the mouse pointer over the source instance specification.
2. Press on the Resource Catalog button and drag it out. Drag to the target instance specification and release the mouse button.



Create a link

3. Select Link from Resource Catalog. A link is created.



Link created

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Composite structure diagram

Composite structure diagram visualizes the internal structure of a class or collaboration. You will learn how to create composite structure diagram in this chapter.

Creating composite structure diagram

Learn how to create composite structure diagram.

Drawing composite structure diagrams

Composite structure diagram is a kind of UML diagram that visualizes the internal structure of a class or collaboration. It is a kind of component diagram mainly used in modeling a system at micro point-of-view.

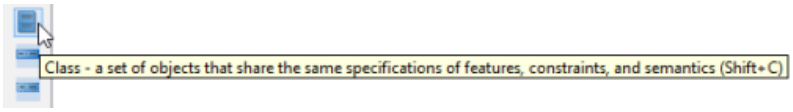
Creating composite structure diagram

Perform the following steps to create a UML composite structure diagram.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Composite Structure Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

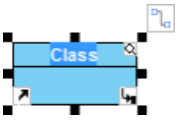
Creating class

To create a class in composite structure, click **Class** on the diagram toolbar and then click on the diagram.



Create class

A class will be created.

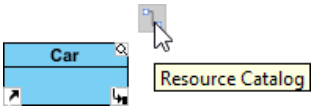


Class created

Creating part

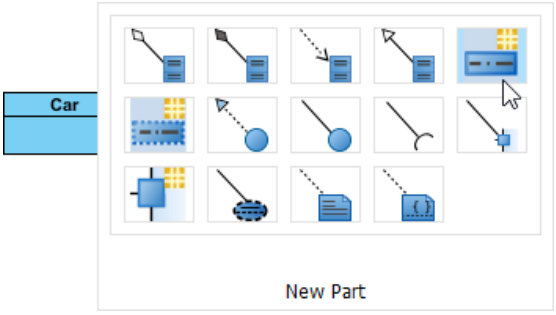
To create a part inside a class:

1. Move your mouse pointer over the class.
2. Click on the **Resource Catalog** button.



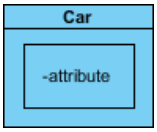
Clicking on Resource Catalog button

3. Select **New Part** from Resourcce Catalog.



To create part

A part is created.

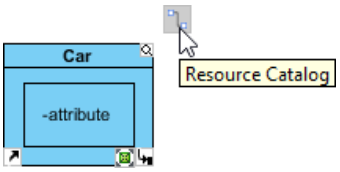


Part created

Creating port

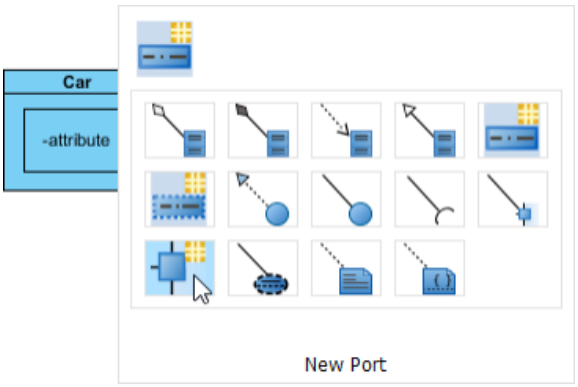
To create a port that attaches to a class:

1. Move your mouse pointer over the class.
2. Click on the **Resource Catalog** button.



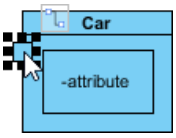
Clicking on Resource Catalog button

3. Select **New Port** from Resourcce Catalog.



To create port

A port is created.

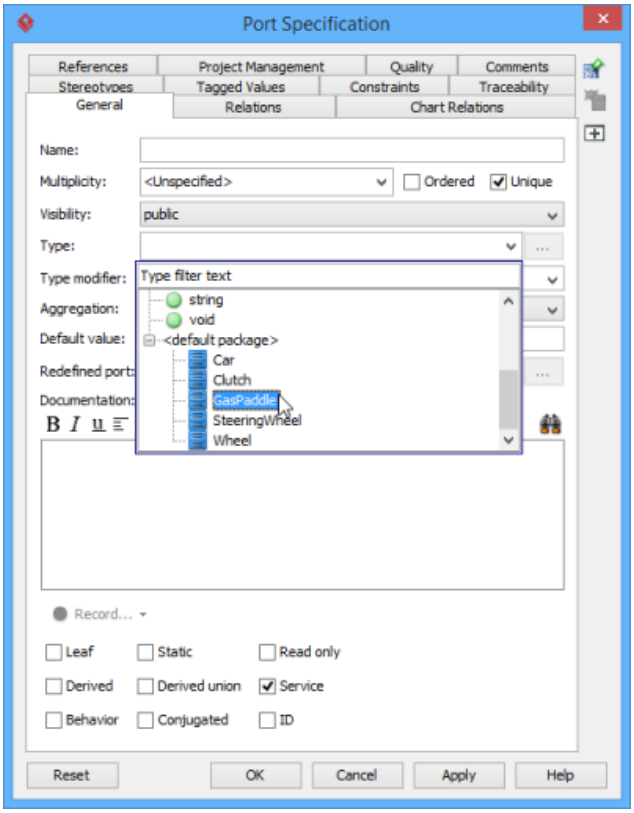


Port created

Specifying type of port

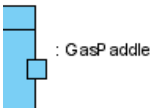
Right-click the port and select **Open Specification...** from the pop-up menu. The **Port Specification** dialog box appears.

Click the combo box of **Type** and select a class.



Select type

Click **OK** button to apply the changes. Type will be shown on the caption of the port.



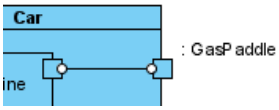
Type shown on port

Creating connector

To create connector, click **Connector** on the diagram toolbar.

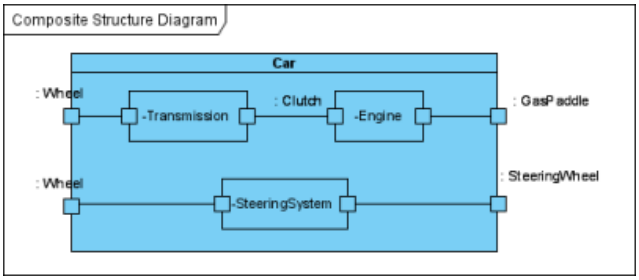
Create connector

Drag from the source shape, move the mouse over the target shape and then release the mouse button to create the connector.



Connector created

Continue to complete the diagram.



Completed diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Timing diagram

Timing diagram shows time, event, space and signal for real-time and distributed system. You will learn how to create timing diagram in this chapter.

Creating timing diagram

Learn how to create timing diagram.

Drawing timing diagrams

Timing diagram is a kind of UML diagram that shows time, event, space and signal for real-time and distributed system.

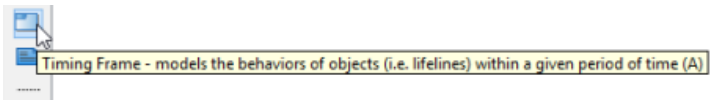
Creating timing diagram

Perform the steps below to create a UML timing diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Timing Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

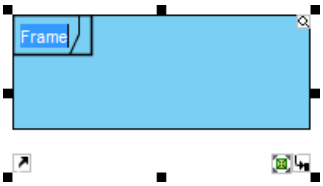
Creating timing frame

To create timing frame in a timing diagram, click **Timing Frame** on the diagram toolbar and then click on the diagram.



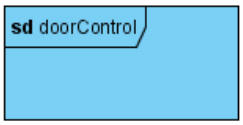
Create timing frame

Double click on the top left corner of the frame to rename it.



Rename frame

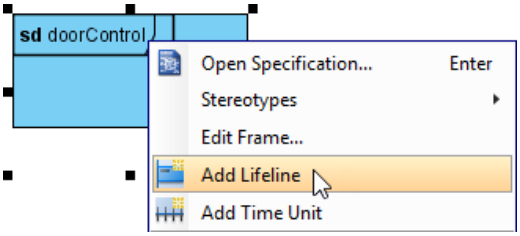
The name of a timing frame is usually preceded by the **sd** keyword.



Frame renamed

Adding lifeline to frame

To add lifeline to a timing frame, right-click the frame and select **Add Lifeline** from the pop-up menu.

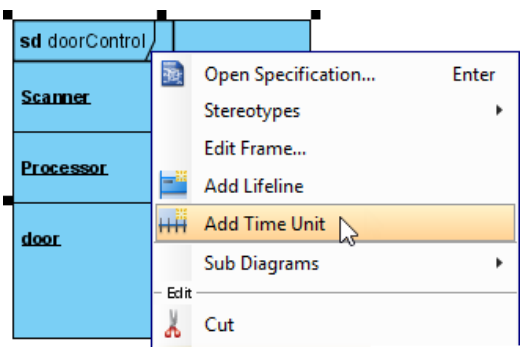


Add lifeline

Double-click on the name of the lifeline to rename it.

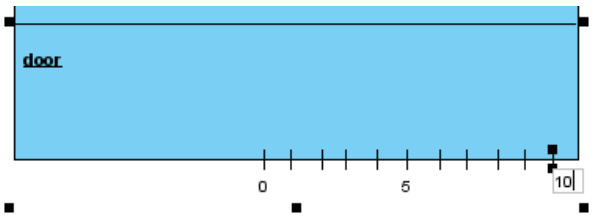
Adding time unit to frame

To add time unit to a timing frame, right-click the frame and select **Add Time Unit** from the pop-up menu.



Add time unit

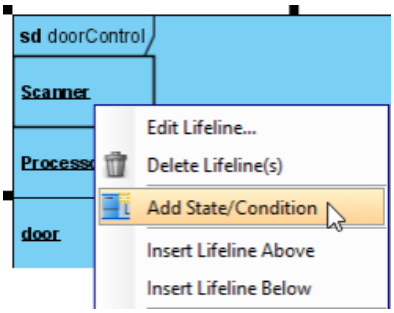
Repeat the step to add as many as time units you need. Double-click on a time unit to rename it.



Rename time unit

Adding state/condition to lifeline

To add state/condition to a lifeline, right-click the lifeline and select **Add State/Condition** from the pop-up menu.

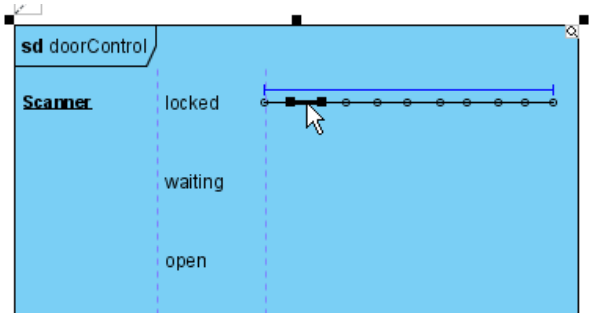


Add state/condition

Double click on the name of the state/condition to rename it.

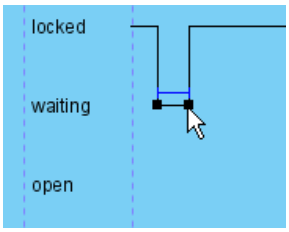
Dragging time instance

Move your mouse pointer over the line segment of a time instance, click and drag it.



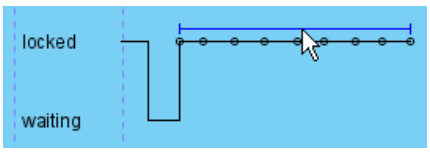
Drag time instance

Release the mouse button when reached the target state/condition.



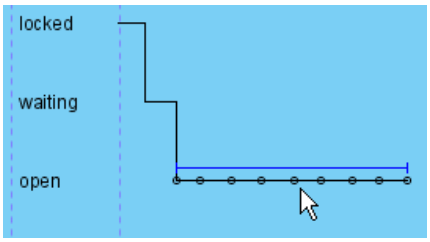
Dragged time instance

You can also move a group of time instances that are at the same state/condition. Mouse over the time instances and you will see a blue line above them, click and drag on the blue line.



Move a group of time instances

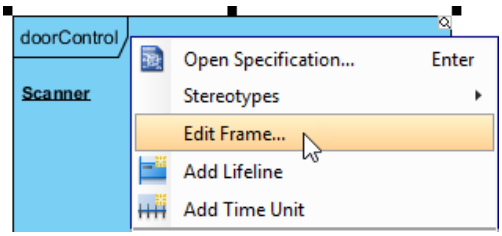
Release the mouse button when reached the target state/condition. The group of time instances is moved at once.



Moved group of time instances

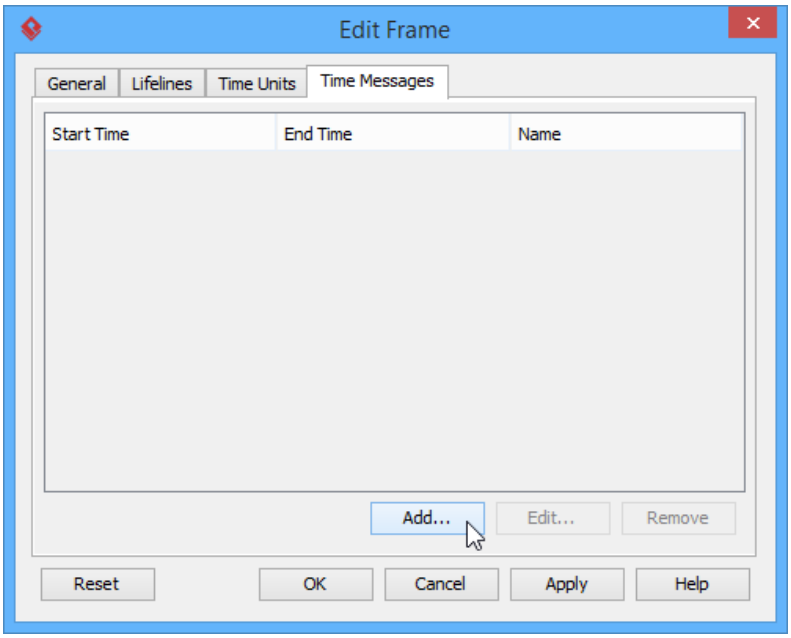
Adding time messages to frame

To add time messages to frame, right-click the timing frame and select **Edit Frame...** from the pop-up menu.



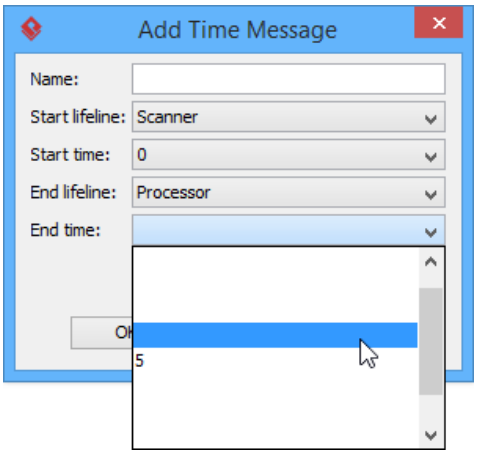
Edit frame

In the **Edit Frame** window, open the **Time Messages** tab and click **Add...** button.



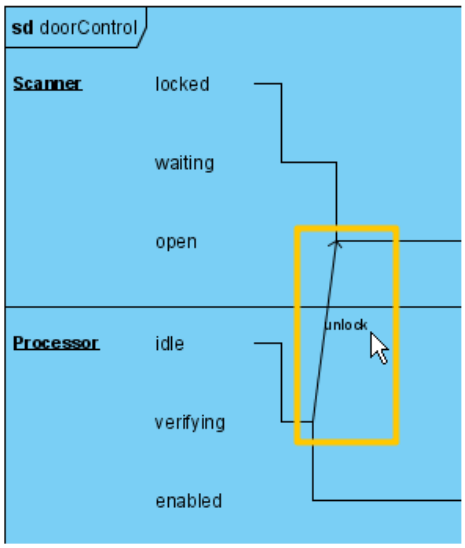
Add time message

When the **Add Time Message** dialog box pops out, enter name and select the start lifeline, start time, end lifeline and end time for this time message. Note that as time units may be unnamed, when selecting start/end time you should check the relative position of the time unit in the list.



Select end time of time message

The time message is shown on the frame.

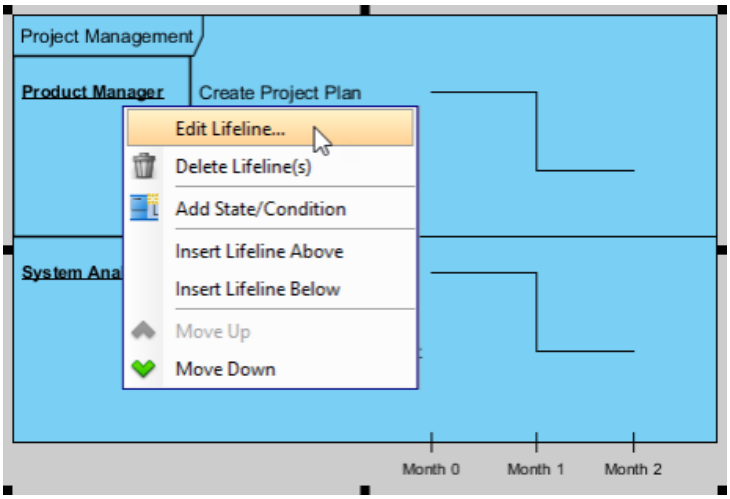


Time message

Adding duration constraint

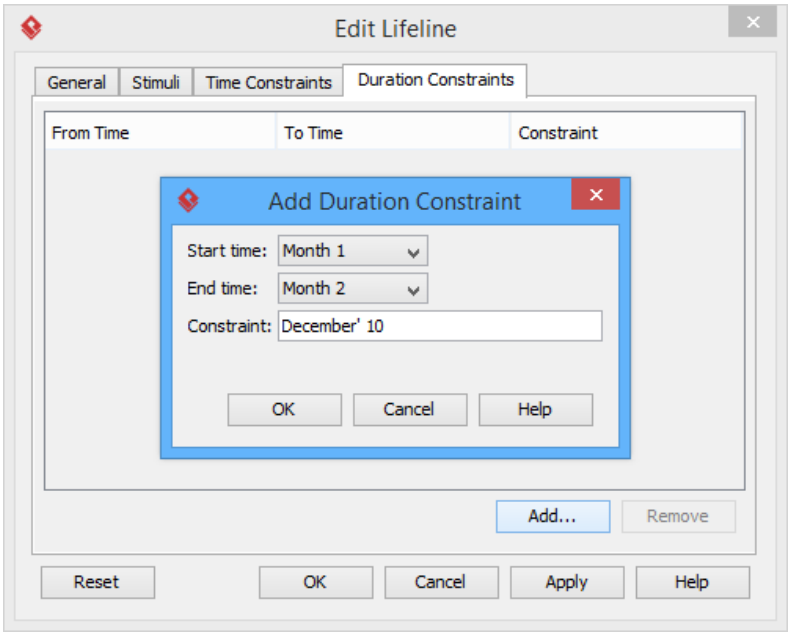
Duration constraint is used to show the duration limitation of a particular lifeline over a period of time.

1. To set the duration constraints of a lifeline, right-click on the lifeline and select **Edit Lifeline...** from the pop-up menu.



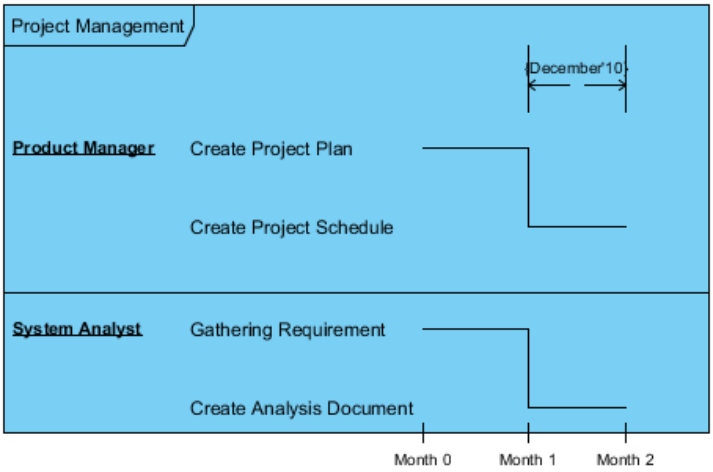
Edit lifeline

2. In the **Duration Constraints** tab, click on the **Add...** button. In the **Add Duration Constraint** dialog box, select the appropriate **Start time** and **End time** from the drop down menu. Fill in the duration constraint of the selected time on the **Constraint** field. Click on the **OK** button to close the dialog box.



Add duration constraint

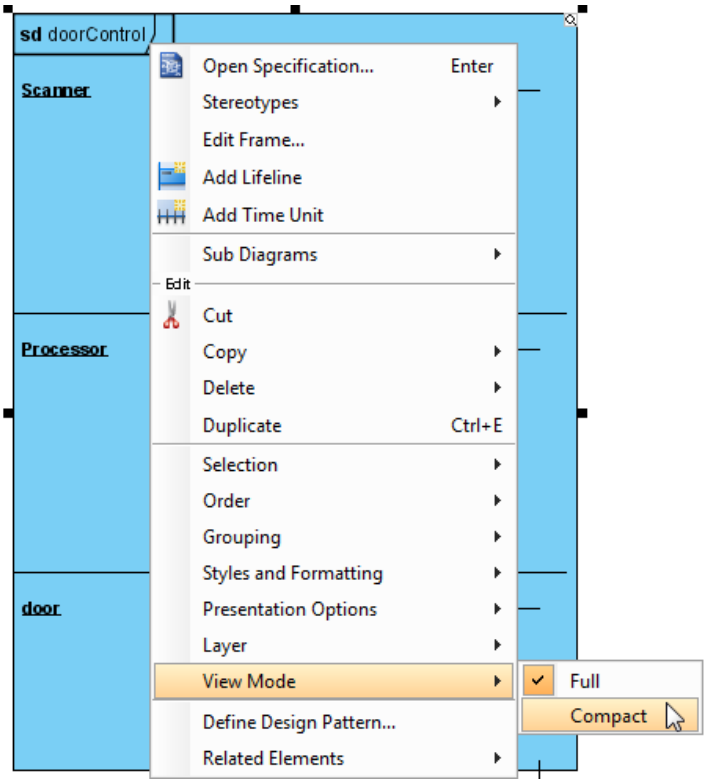
3. Click **OK** to return to diagram.



Duration constraint is added

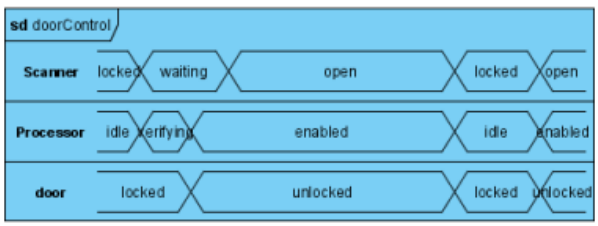
Switching to compact view mode

To switch to compact view mode, right-click the frame and select **View Mode** > **Compact** from the popup menu.



Switch to compact view mode

The frame will be shown in compact mode.



Frame shown in compact mode

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Interaction overview diagram

Interaction overview diagram is the variant of activity diagram in which a control flow with nodes represents interaction diagrams. You will learn how to create interaction overview diagram in this chapter.

Creating interaction overview diagram

Learn how to create interaction overview diagram.

Drawing interaction overview diagrams

Interaction overview diagram is a kind of UML diagram. It is the variant of UML activity diagram that shows specifically the flow of internation diagrams like sequence diagrams.

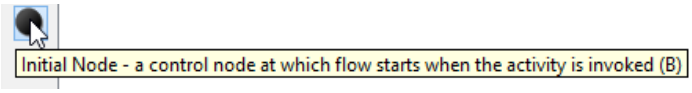
Creating interaction overview diagram

Perform the steps below to create an UML interaction overview diagram in Visual Paradigm.

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Interaction Overview Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

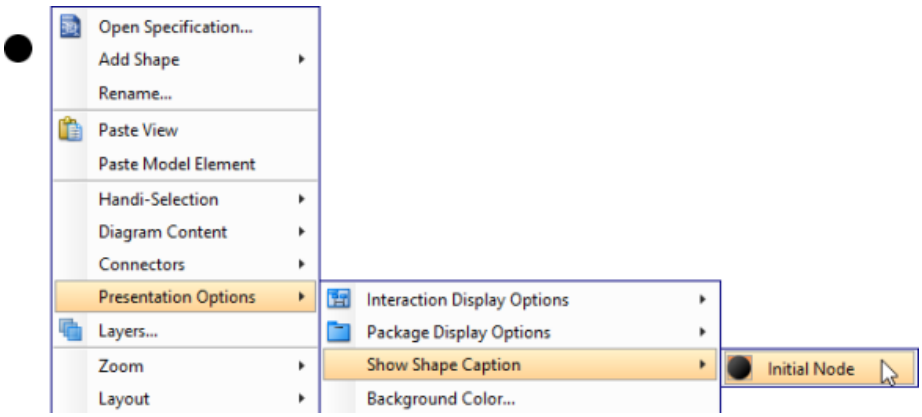
Creating initial node

Initial node is the beginning of a control flow. To create initial node in interaction overview diagram, click **Initial Node** on the diagram toolbar and then click on the diagram.



Create initial node

An initial node is created. The caption of initial node is hidden by default, to show it, right-click on the diagram and select **Presentation Options** > **Show Shape Caption** > **Initial Node** from the pop-up menu.

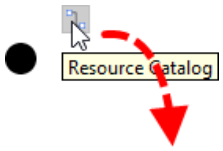


Show caption of initial node

Creating decision node

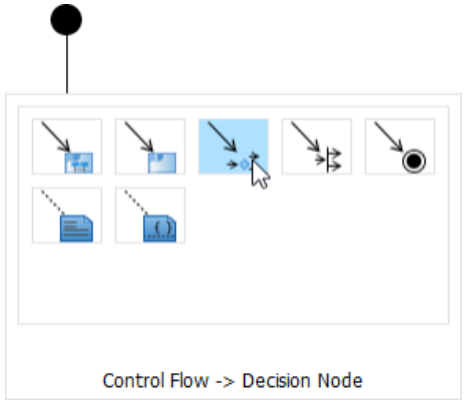
To create a decision node from an initial node:

1. Move your mouse pointer over the initial node.
2. Press on the **Resource Catalog** button and drag it out.



Using Resource Catalog

3. Release the mouse button at the place where you want the decision node to be created.
4. Select **Control Flow -> Decision Node** from Resource Catalog.



To create a decision node

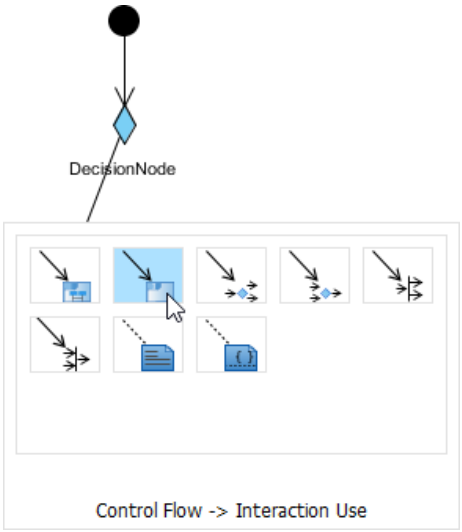
5. A new decision node will be created and is connected to the initial node. Enter its name and press **Enter** to confirm editing.

Decision node created

Creating interaction use

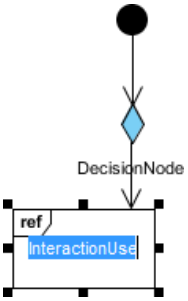
To create an interaction use:

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.
3. Release the mouse button at the place where you want the interaction use to be created.
4. Select **Control Flow -> Interaction Use** from Resource Catalog.



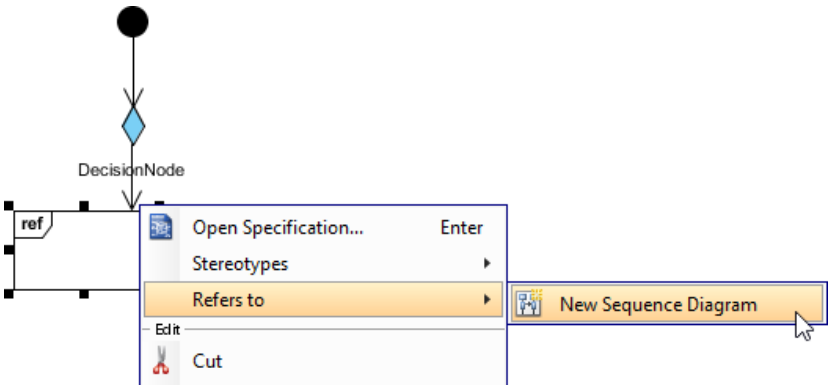
To create an interaction use

5. A new interaction use will be created and is connected to the source node. Enter its name and press **Enter** to confirm editing.



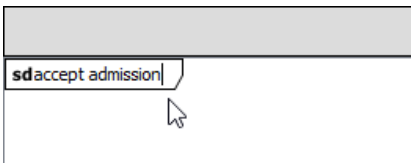
Interaction use created

You can make the interaction use refers to a diagram by right clicking on it and select **Refers to** > **New Sequence Diagram** from the pop-up menu.



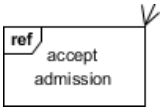
Make interaction use refers to diagram

When sequence diagram is created, rename the diagram.



Rename sequence diagram

When you return to the interaction overview diagram, you can see the interaction use caption shows the diagram it refers to.



Interaction use caption updated

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Requirement diagram

Requirement diagram lets you visualize system functions as well as the ways to test the functions. This chapter teaches you how to work with requirement diagram.

Creating requirement diagram

This page shows you how to create requirements in requirement diagram, specify requirements body, relate requirements and create test cases.

Customizing requirement types

You will see how to define your own requirement type in this page.

Modeling and documenting test cases

Make use of the test case element and its test plan editor to model the test case of requirements.

Drawing requirement diagram

The requirement diagram is designed specifically for the Systems Modeling Language (SysML). It is created in requirement containers and requirements to present the relationship between requirements and other model elements.

Creating requirement diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Requirement Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating requirement

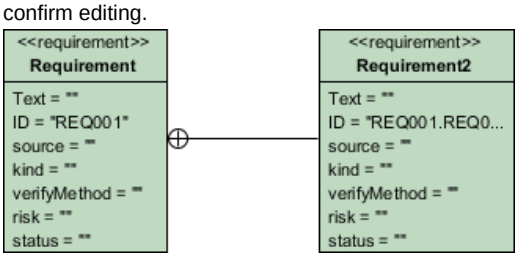
To create a Requirement, click the **Requirement** button on the diagram toolbar and then click on the diagram.

Create requirement

Decomposing requirement

To decompose a Requirement:

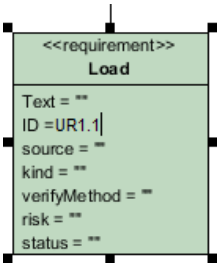
1. Move your mouse pointer over the requirement.
2. Press on the **Resource Catalog** button at top right and drag it out.
3. Release the mouse button at the place where you want the decomposed requirement to be created.
4. Select **Containment -> Requirement** from Resource Catalog.
5. A new requirement will be created and is connected to the source requirement with a containment connector. Enter its name and press **Enter** to confirm editing.



Requirement and Containment created

Inline editing requirement properties

To inline edit the property of a Requirement (e.g. ID), double-click on the property, enter new value and press Enter to confirm.

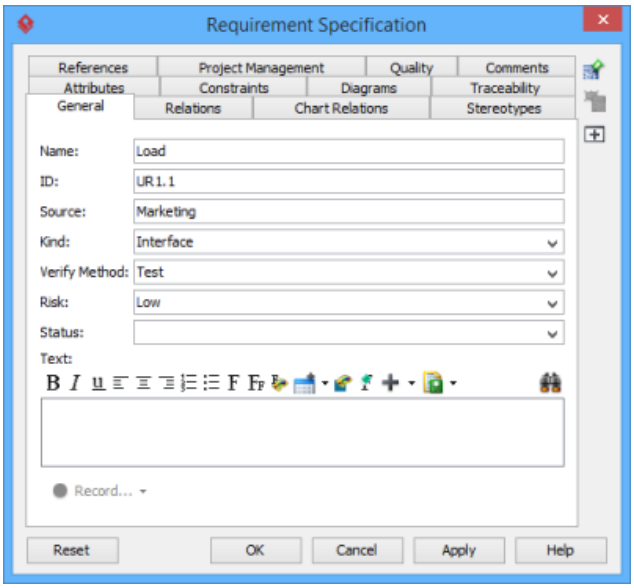


Inline editing Requirement properties

Editing requirement properties with specification dialog box

You can also open specification dialog box of a Requirement to edit its properties. Click on the tiny magnifier icon at the top right of a Requirement shape to open the specification window.

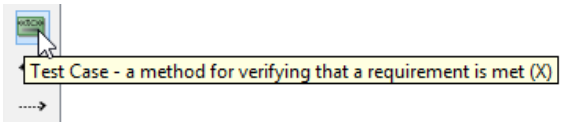
The **Requirement Specification** window shows. Edit the properties and click **OK** button to apply the changes.



Requirement Specification

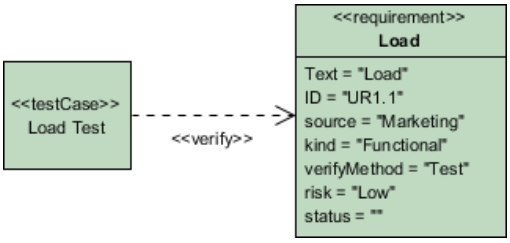
Creating test case and link to requirement

To create a Test Case, click the **Test Case** button on the diagram toolbar and then click on the diagram.



Create test case

Move your mouse pointer to the Test Case. Press on the **Resource Catalog** button at top right and drag it out. Move the mouse pointer over a Requirement and then release the mouse button, a Verify relationship will be created from the Test Case to the Requirement.



Verify relationship created

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

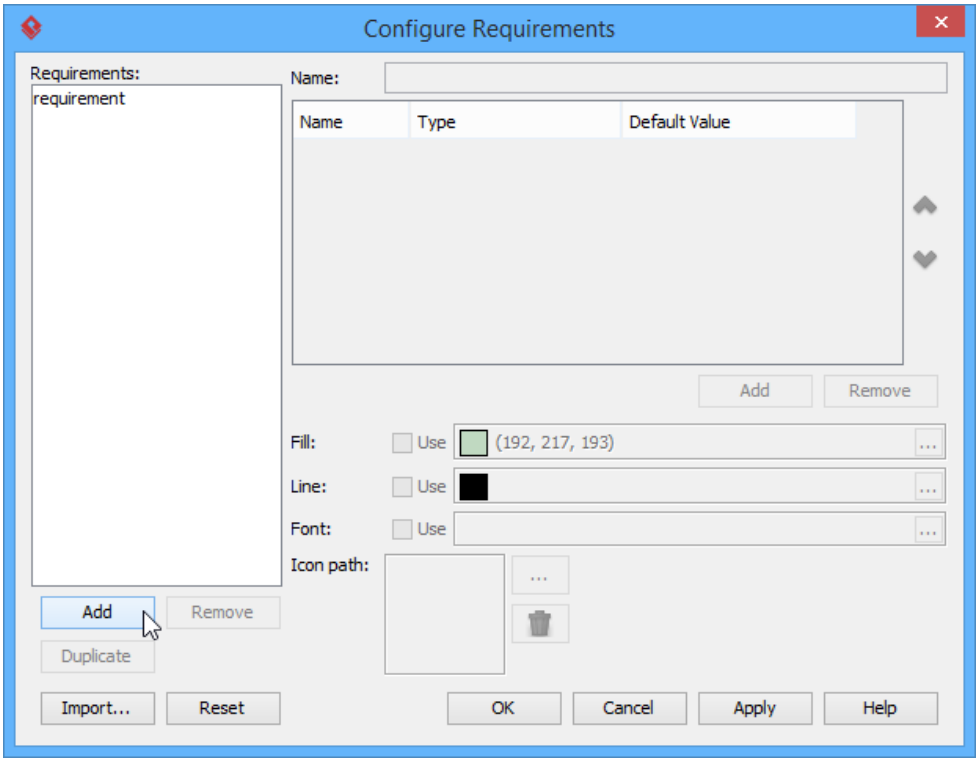
Customizing requirement types

Users can record and present requirements as boxes visually through requirement modeling. The name of requirements summarizes the requirement while a set of attributes defines the requirement. The default requirement box enables users to specify general attributes, such as ID, source, kind, verify method, risk and status. Moreover, you can customize your own requirement types that contain attributes related to your domain.

Creating new requirement type

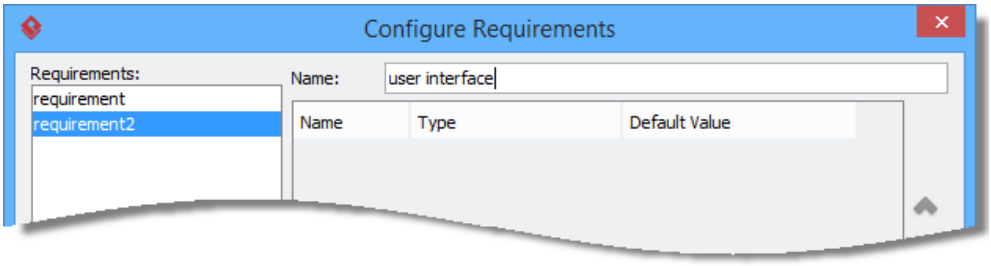
Before creating new Requirement type, create a new requirement diagram or open your target requirement diagram where you want to customize your our own requirement types. Select **Windows**, then click **Configuration > Configure Requirements...** from the toolbar.

The **Configure Requirements** dialog box appears. Click **Add** to add a new requirement type.



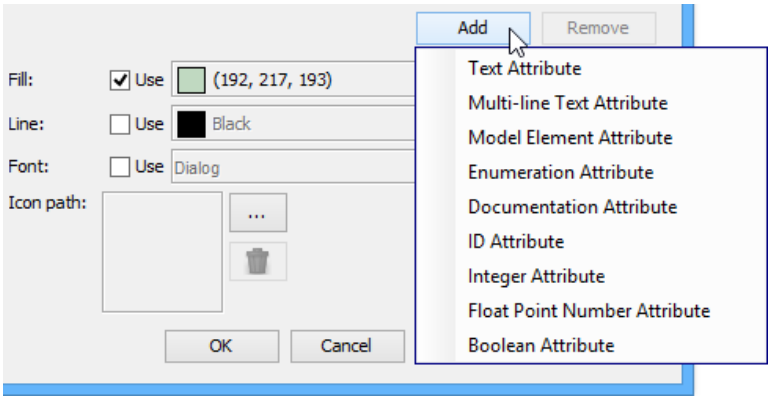
***Configure Requirements** dialog box*

Enter name of the Requirement type in **Name** field.



Enter name for Requirement type

Add attributes for the requirement type to make it meaningful. Click **Add** button below the attribute table and select an attribute.



Add documentation attribute

Name the newly created attribute. Create as much as attribute you need by following the previous step.

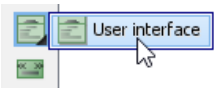
> **NOTE:** If you select **Enumeration Attribute** from the drop-down menu, **Edit Enumeration...** button will appear. Click **Edit Enumeration...** button to edit it.

Besides defining attributes, you can format the requirement type with fill, line and font. Click the **...** button of **Fill** if you want to customize a color for the requirement type.

> **NOTE:** Click the **...** button of **Line** if you want to customize its line property while click the **...** button of **Font** if you want to customize its font property.

Once you finish configuring requirement types, click **OK** button to return to your target requirement diagram.

Finally, you can see the customized requirement type is available on the diagram toolbar. You can select and click it on the diagram to create the shape.



The customized requirement type

Related Resources

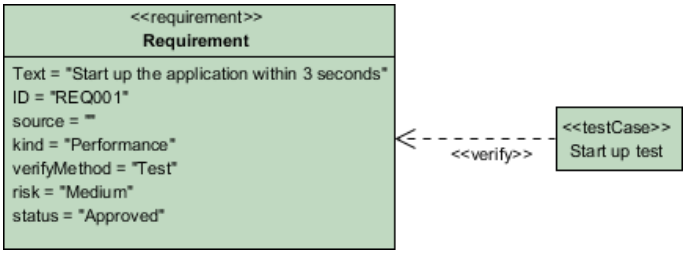
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Modeling and documenting test cases

Produce test case from requirement

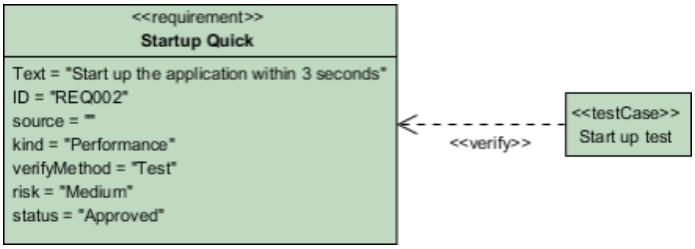
1. In a requirement diagram, move your mouse pointer over a requirement that you want to produce **Test Case**.
2. Press on the **Resource Catalog** button at top right and drag it out.
3. Release the mouse button at the place where you want the test case (shape) to be created.
4. Select **Verify** <- **Test Case** from Resource Catalog.
5. Release the mouse button to create a test case. Name it.



Test case is created

Creating test case and link to requirement

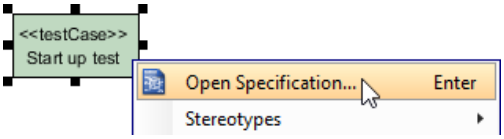
1. In a requirement diagram, click the **Test Case** button on the diagram toolbar and then click on the diagram to create a Test Case.
2. Move your mouse pointer to the Test Case.
3. Press on the **Resource Catalog** button at top right and drag it out.
4. Move the mouse pointer over a Requirement and then release the mouse button, a Verify relationship will be created from the Test Case to the Requirement.



Verify relationship created

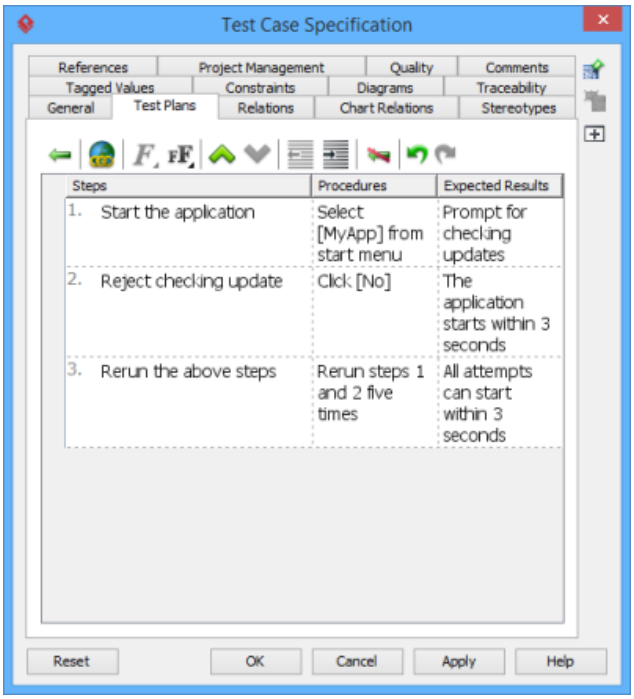
Documenting test case

1. Right click on a test case and select **Open Specification...** from the popup.



Open specification of test case

2. In the **Test Plans** tab, fill in the **Steps**, **Procedures** and **Expected Results**.



Test Plan filled

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Textual analysis

Textual analysis is a tool for recording customers' needs. Furthermore, it lets you extract key terms from a passage you recorded, and transform the terms to model elements or put them into glossary to build a project -based dictionary.

Recording requirements

Document customers' needs by performing textual analysis.

Identifying important terms

Shows you how to identify glossary term from a passage recorded by textual analysis.

Identifying candidate objects

Shows you how to identify candidate model element from a passage. You selectively convert candidate to actual model element and visualize it in diagram.

Forming diagram from candidate objects

Shows you how to visualize candidate elements in a diagram.

Candidate pane view

A view that shows candidate elements in visualized form - as boxes.

Documenting Requirements with Textual Analysis Tool

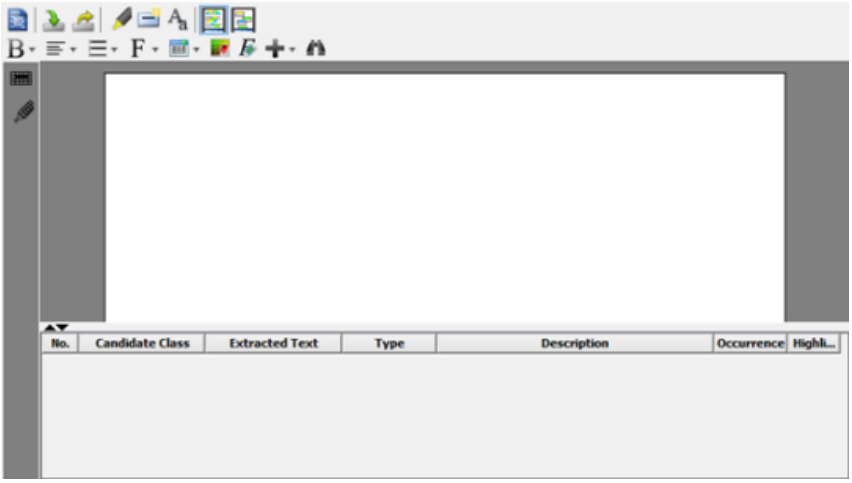
Before you start business process modeling, you usually have to discuss with your customers about their needs and to familiarize yourself with their company's operations as well as their problems. During the meeting you can collect useful information from customers, including the conversation log and documents. You can make use of textual analysis , a text-based editor to help recording those textual information. In addition to a plain text editor, you can identify important terms or objects (e.g. class, use case) from the problem description.

Creating textual analysis

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Textual Analysis**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

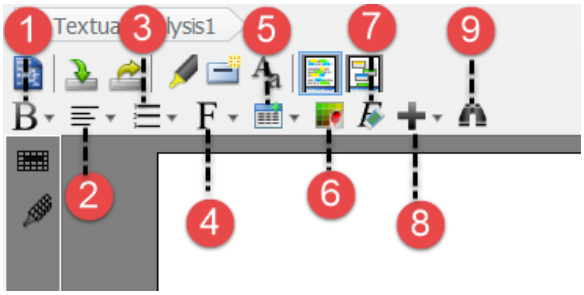
Problem statement editor

The problem statement editor is where you can record the textual information you obtain from your customers.



Problem statement editor

All buttons on editor's toolbar are depicted in the following table:



Editor's toolbar

No.	Name	Description
1	Bold	Set the highlighted text to bold.
	Italic	Set the highlighted text to italic.
	Underline	Underline the highlighted text.
2	Left Justify	Set the alignment of highlighted text to the left.
	Center Justify	Set the alignment of highlighted text to the center.
	Right Justify	Set the alignment of highlighted text to the right.
3	Ordered list	Add a numbered list.
	Un-ordered list	Add a list with bullet points.
4	Font	Select the font family of highlighted text.
	Font size	Select the size of highlighted text.
	Font color	Select the color of highlighted text.

5	Table	Add a table.
6	Background color	Select the background color of highlighted text.
7	Clear formats	Clear formats of the whole editor to convert the content to plain text.
8	Link	Add a hyperlink.
	Image	Add an image.
	Add Model Element...	Insert an existing model element or create a new one.
	Add Diagram	Add a diagram
9	Find	You can search your target word/phrase in problem statement in shortcut through here. Simply enter the word/ phrase in search field, as a result, the word/ phrase matching you typed will be highlighted.

The description of buttons on problem statement's toolbar

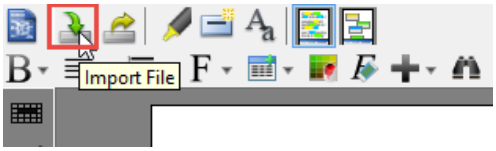
Entering problem statement

Three ways of entering problem statement are provided in Visual Paradigm .

- Typing on the editor
- Importing an external text file
- Copying and pasting from an external source

To type in the editor, type the problem statement directly on the editor.

To import a text file, click **Import File** on the toolbar.



Import file

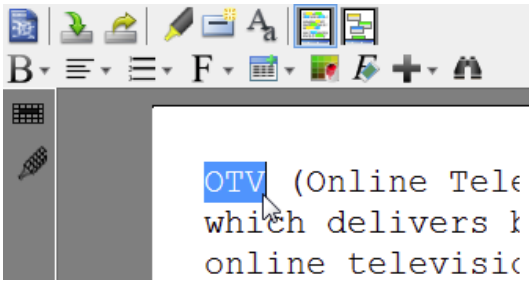
When the **Open** dialog box pops out, select a text file to import. As a result, the imported problem statement will be shown on the text area.

To copy and paste from an external source, press **Ctrl + C** on the selected text and press **Ctrl + V** for pasting it on the editor.

Formatting text

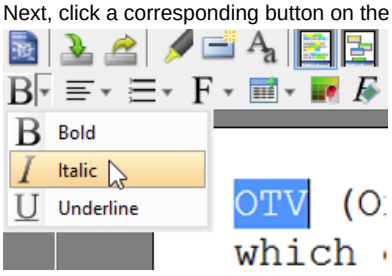
Since Visual Paradigm supports rich text format (RTF), you can format the problem statement on the editor, such as making it bold, italic, or inserting a table.

1. To format text, highlight the target word/ phrase in advance.



Highlight OTV

2. Next, click a corresponding button on the toolbar. i.e. Click **Italic** button to make the target word italic.



*Click **Italic** button*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

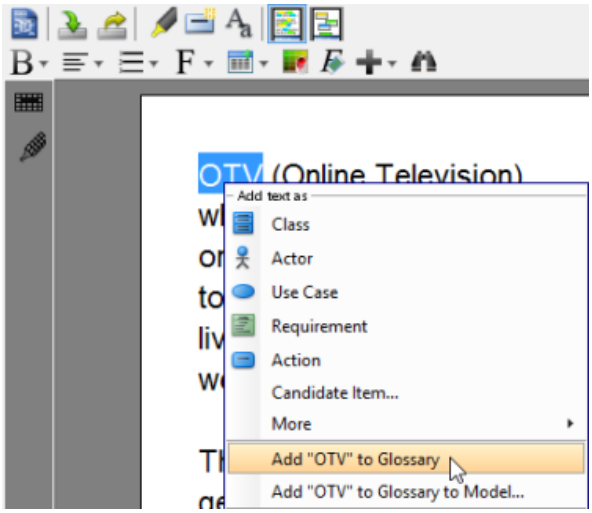
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm

- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Identifying important terms

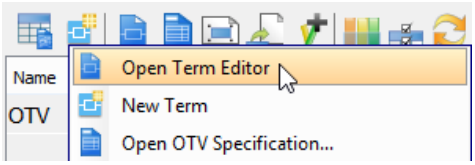
A word usually can have various meaning under different domains. To clarify and standardize the meaning of your specific word, you can extract it from textual documentation to define it as a glossary term. After adding the word as glossary term, you can define its aliases and enter its documentation to provide additional information. In textual analysis, you can define a specific word by highlighting it on problem statement editor and add it to glossary. After that, define aliases and enter documentation for the glossary term in term editor.

1. Highlight the specific term on problem statement editor, right click on it and select **Add [the highlighted term] to Glossary** from the pop-up menu.



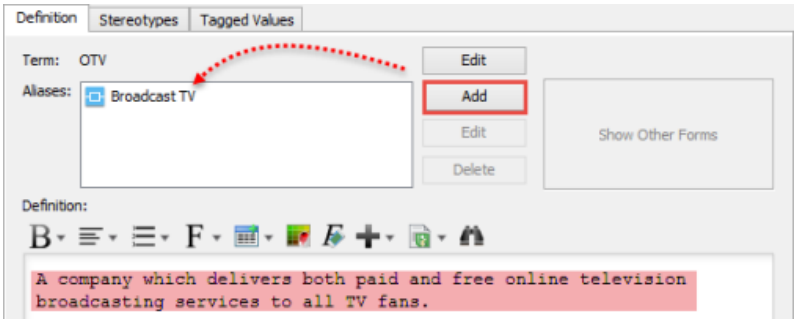
Add OTV to glossary

2. When the **Glossary Grid** page is opened, right click on the newly created term and select **Open Term Editor** from the pop-up menu.



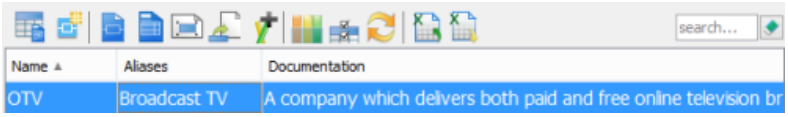
Open Term Editor

3. In the **Term Editor** page, open **Definition** tab.
4. You can define aliases for the term and enter documentation as definition for the term. To insert an alias, click **Add** button to type the alias in the pop-up **Input** dialog box. To enter the definition of the term, enter under **Definition** directly.



Define aliases and enter documentation

As a result, the columns of **Aliases** and **Documentation** are filled when you return to **Glossary Grid** page.



OTV is defined

NOTE: If the **Aliases** column is hidden, click **Configure Columns...**, open the **Properties** tab and select it under the **Details** folder.

NOTE: In Glossary Grid, you can jump to the source from which a term was defined by right clicking on the term and selecting **Transit From > %SOURCE_ELEMENT_NAME%** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

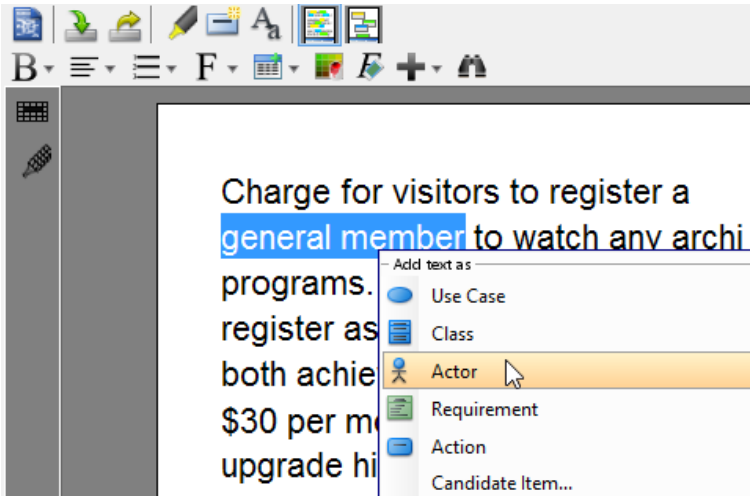
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Identifying candidate objects

By studying the problem statement, you can extract words or phrases that are relevant to the system and convert them into model elements, such as classes, use cases (system goals) and action, etc. Those objects are regarded as candidate objects. You can extract words or phrases from problem statement to become specific type of candidate objects and edit their properties when necessary.

Identifing candidate objects

Highlight the word/ phrase from the problem statement and select **Add text as [model element type]** from the pop-up menu.



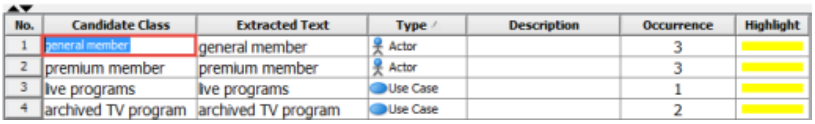
Select actor as its type

Editing candidate objects

You can rename candidate objects, change their type, write their description and change their color of highlight in the grid at the bottom of textual analysis .

To rename the candidate object:

Double click on the **Candidate Class** cell and rename the candidate object.

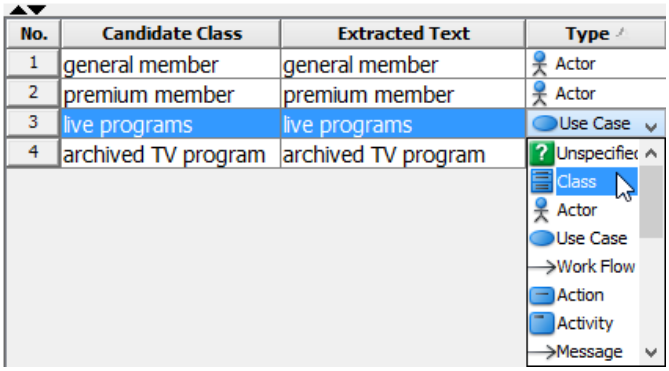


No.	Candidate Class	Extracted Text	Type	Description	Occurrence	Highlight
1	general member	general member	Actor		3	
2	premium member	premium member	Actor		3	
3	live programs	live programs	Use Case		1	
4	archived TV program	archived TV program	Use Case		2	

Rename candidate object

To change the candidate object's type:

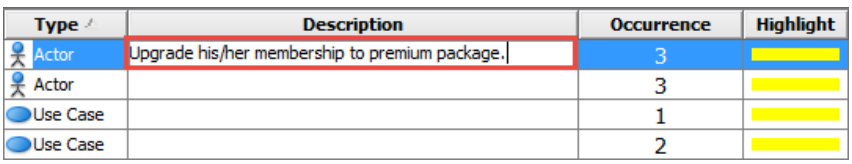
Double click on the **Type** cell and select a type from the combo box.



Select class as its type

To add description for the candidate object:

Double click on **Description** cell and type text inside the cell.



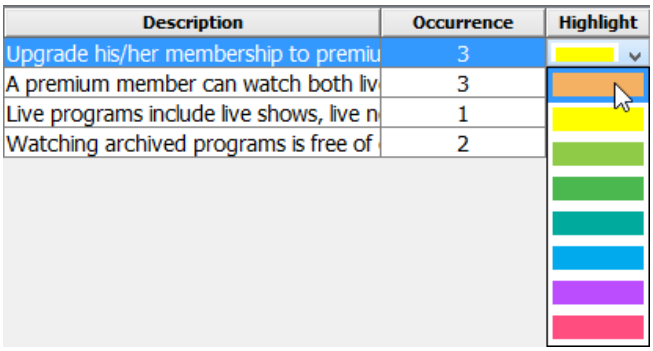
Type	Description	Occurrence	Highlight
Actor	Upgrade his/her membership to premium package.	3	
Actor		3	
Use Case		1	
Use Case		2	

Enter description

NOTE: The text you typed in **Description** cell will become the documentation of the corresponding model element.

To change the highlight color of candidate object in problem statement:

1. Click the **Highlight** cell and press the inverted triangle.
2. Select a color from the combo box.



Select the highlight color

Related Resources

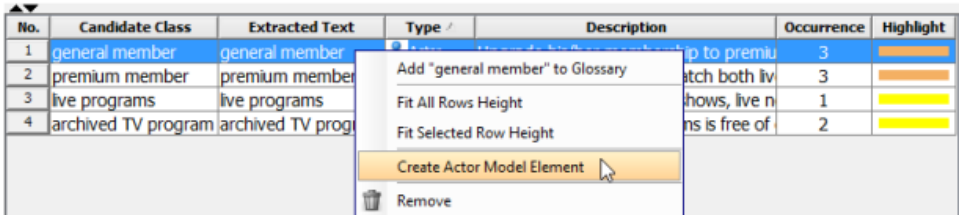
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Forming diagram from candidate objects

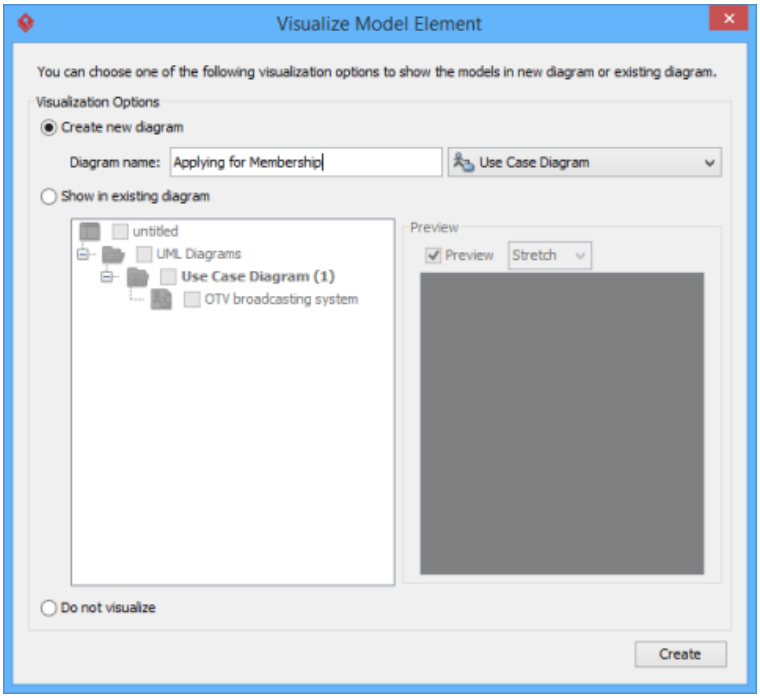
In problem statement editor, you can form a diagram from candidate objects or show it in an existing diagram by visualizing it.

1. Right click on the target candidate object and select **Create [candidate object's type] Model Element** from the pop-up menu.



Create a model element

2. In the **Visualize Model Element** dialog box, either check **Create new diagram** to show your model element on a new diagram or check **Show in existing diagram** to show on an existing diagram. Finally, click the **Show** button to proceed.



*Check an option in **Visualize Model Element** dialog box*

As a result, the model element will be shown on the selected diagram.

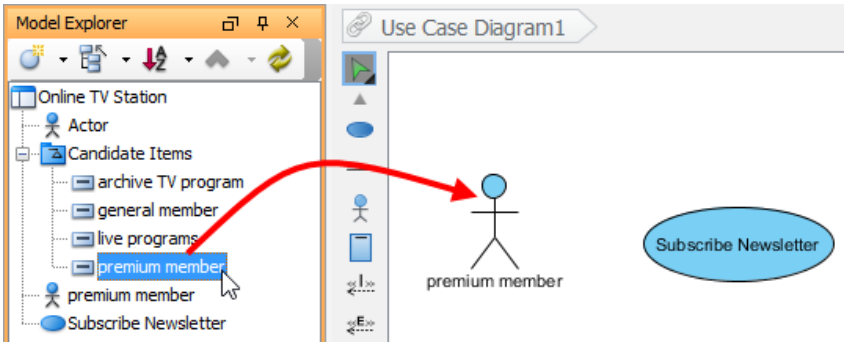
NOTE: If you have already made a model element for the candidate object, the **Create Model Element** option will be hidden even after you right click on it.

Dragging and dropping candidate objects

You can visualize existing candidate objects by dragging from **Model Explorer** and dropping on the diagram.

To open **Model Explorer**, click **View** in the toolbar, and then select **Panes > Model Explorer**

To visualize a candidate object or several candidate objects, select a candidate object (or a few candidate objects) from **Model Explorer**, drag and drop it(or them) on the target diagram. As a result, the view of the selected candidate object(s) will be shown on diagram.



Drag from Model Explorer and drop on the diagarm

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Candidate pane view

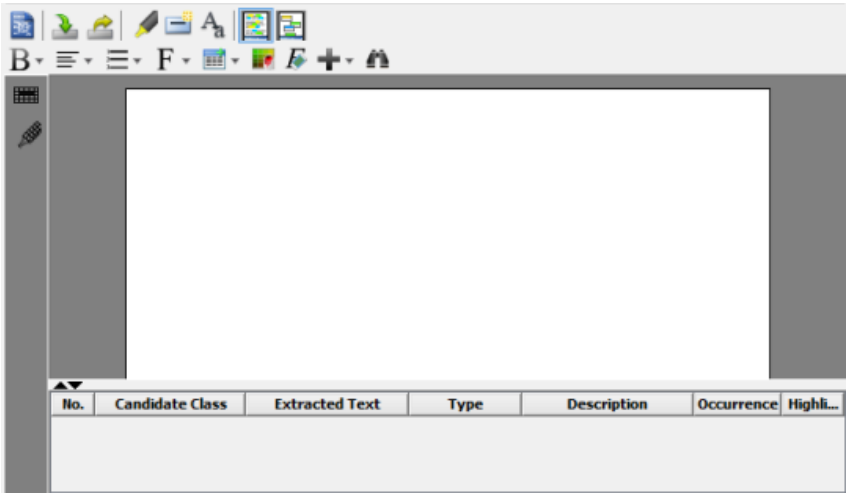
Textual analysis can be divided into two views: problem statement view and candidate pane view. While you can edit problem statement and format text in problem statement view, you can edit and organize candidate objects in candidate pane view. The main characteristic of candidate pane view is that you can visualize candidate objects as boxes on candidate pane for easy arrangement.

To switch to candidate pane view:

Click **Candidate Pane View** button.

*Click **Candidate Pane View** button*

The overview of candidate pane view

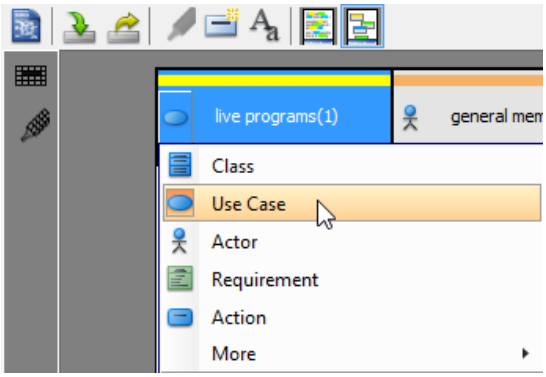


Candidate pane view

Editing candidate objects

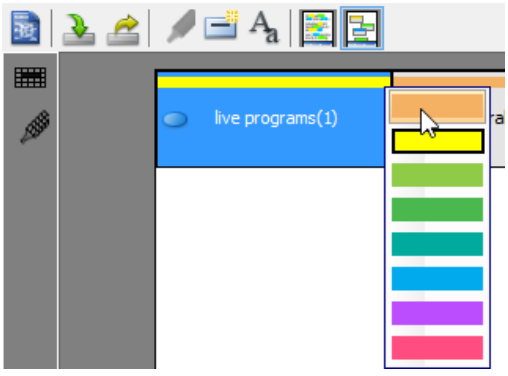
Candidate pane view is similar to problem statement view where you can rename candidate objects, enter their description, change their highlight color and type. Except editing in the grid at the bottom, you can also edit through candidate object's box on candidate pane.

To change the model element type of a candidate object, move the mouse over the target candidate object's box. Click the inverted triangle next to the model element's icon when it reveals. Select a model element type from the pop-up menu.



Change model element type

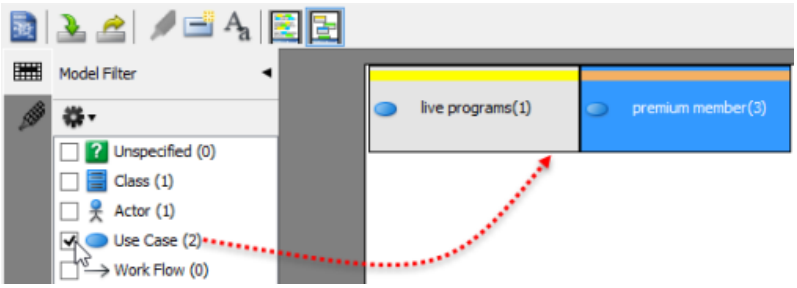
To change the highlight color of a candidate object, move the mouse over the target candidate object's box. Click the inverted triangle on its top-right corner when it reveals. Select a color from the pop-up menu.



Change highlight color

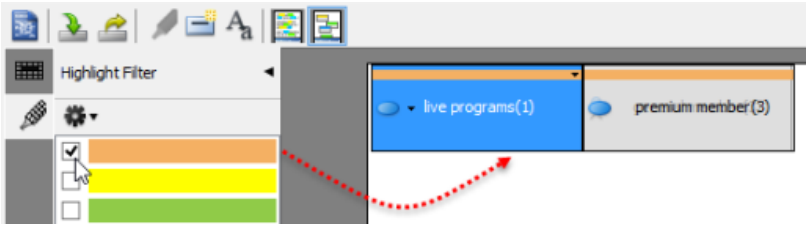
Filtering candidate objects

To filter specific model element of candidate objects, click **Model Filter** button. Check the target model element(s) from the pop-up menu that you want to view on candidate pane.



*Check **Use Case***

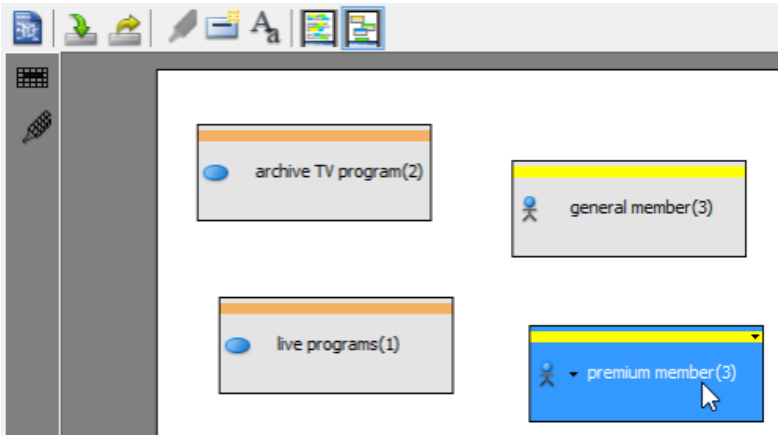
To filter specific highlight of candidate objects, click **Highlight Filter** button. Check the target highlight from the pop-up menu that you want to view on candidate pane.



Check yellow

Freely moving candidate objects

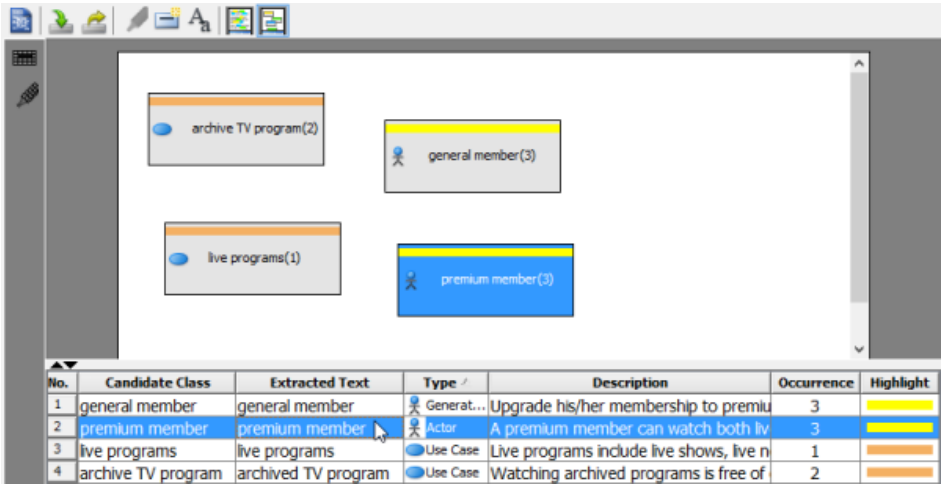
To move candidate object's box, just press the target box and drag it to your preferred location.



*Press and drag **Premium Member***

Selecting candidate objects

When you click a specific candidate object in the grid at the bottom of candidate pane, the corresponding candidate object's box will be selected on candidate pane, and vice versa.



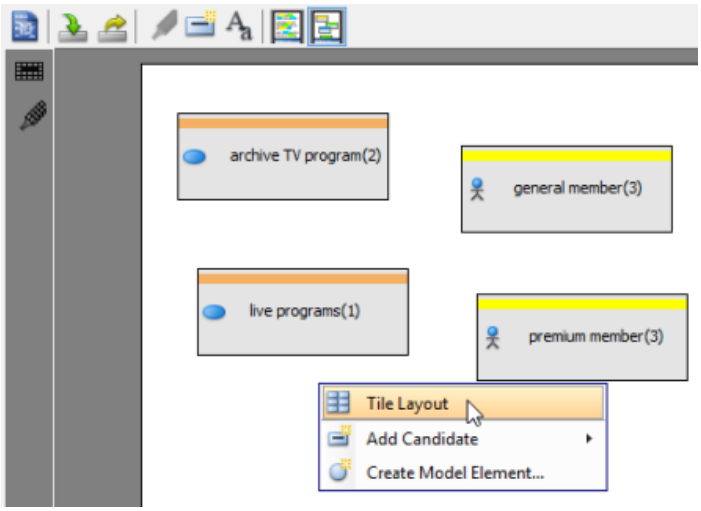
Click candidate object in grid

Setting tile layout

Tile layout refers to the selected objects are arranged in horizontal row.

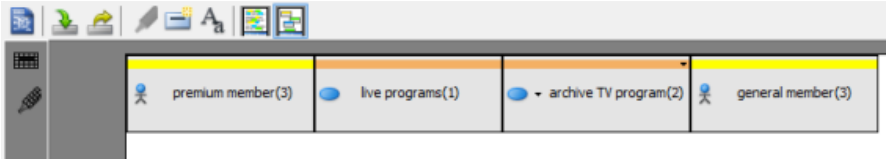
To set tile layout for all candidate object's boxes on candidate pane :

Right click on candidate pane's background and select **Tile Layout** from the pop-up menu.



Set tile layout

As a result, all candidate objects are arranged in tile layout.



Tile layout

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

CRC card diagram

Class-Responsibility Collaborator (CRC) card visualizes classes in card-like presentation. In this chapter, you will learn CRC card diagram, and see how to draw it.

Drawing CRC card diagram

Teaches you how to draw CRC card diagram.

Drawing CRC card diagram

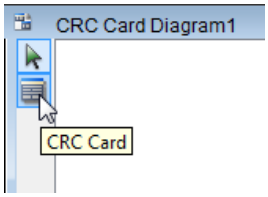
Class-Responsibility Collaborator (CRC) card visualize classes in card-like presentation. Each CRC card contains information like the description of class, its attributes and responsibility. A CRC card diagram is a holder of these cards.

Creating CRC card diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **CRC Card Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating CRC card

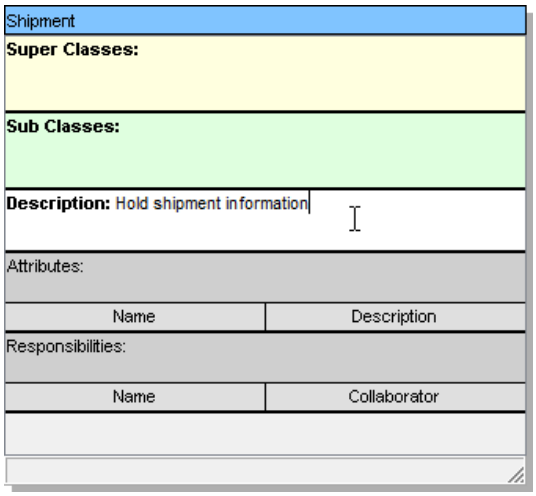
Click **CRC Card** on the diagram toolbar and then click on the diagram to create a CRC card. You can create as many as CRC card on a diagram by repeating this step.



Create CRC card

Editing CRC card properties

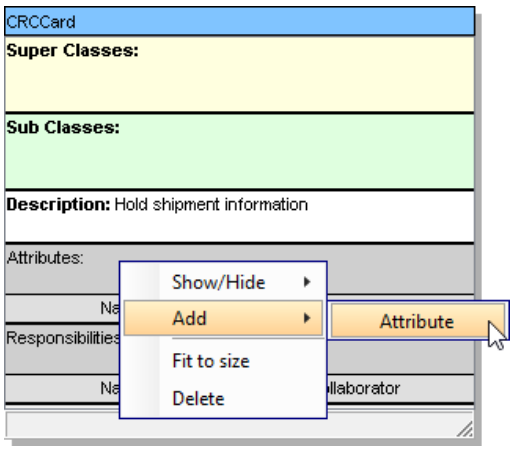
All properties in a CRC card must be edited inline. To edit, double click on the desired field, update its value, and click on the diagram background to confirm editing.



Edit description

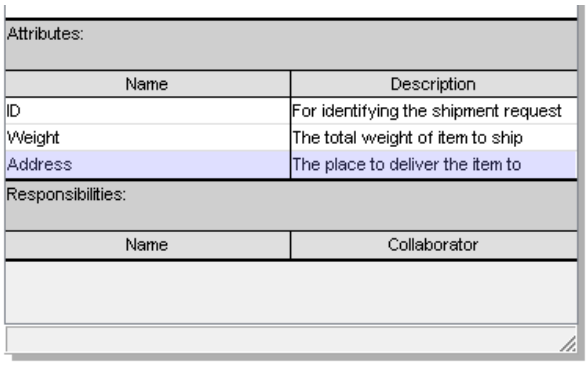
Adding attributes

Right-click on the **Attributes** heading and select **Add** > **Attribute** from the pop-up menu.



Add attribute

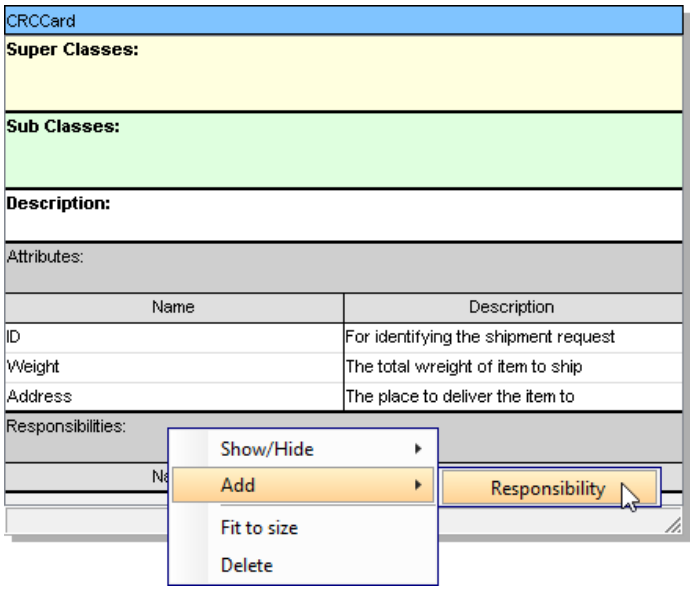
Enter the name and description. Repeat this step until all attributes are added.



Attribute added

Adding responsibilities

Right-click on the **Responsibilities** heading and select **Add** > **Responsibility** from the pop-up menu. Similar to creating an attribute, enter the name and collaborator of each responsibility to show the relationship with other parties.



Add responsibility

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Working with glossary

Glossary is a place where you can read and define project/domain -wide terminologies. This chapter shows you how to identify/define term, and how to add alias. The use of glossary grid will be covered, too.

Identify glossary term

Besides defining a term from scratch, you may extract it from documentation of model elements or from problem statements in textual analysis. You will see how to identify terms from various source.

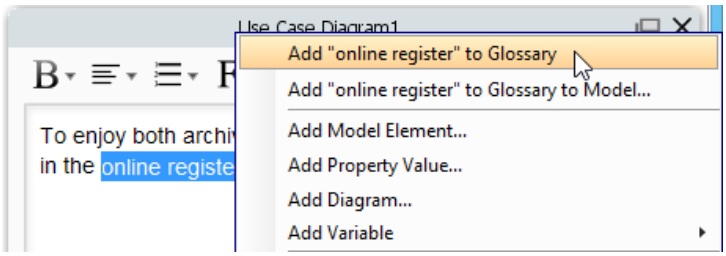
Glossary grid

The glossary grid is the primary area where you can read and define terms. You will learn how to create term in grid, and how to work with functions like label configuration and Excel exporting.

Identify glossary term

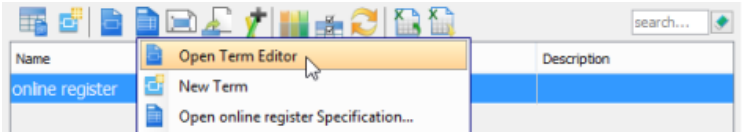
You identify specific terms by adding them to glossary and clarify them by defining aliases and entering description in any textual documents.

1. Highlight the specific term on **Description** editor, right click on it and select **Add "[the highlighted term]" to Glossary** from the pop-up menu.



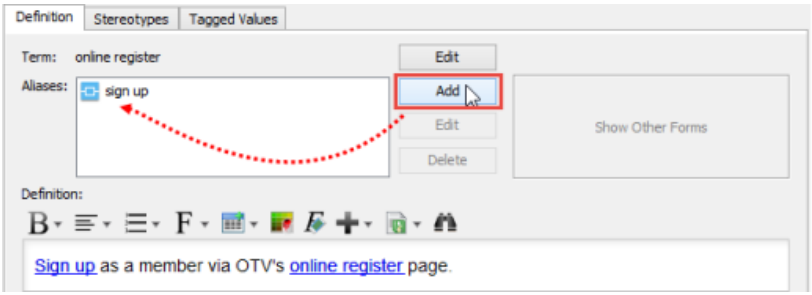
Add "online register" to glossary

2. When the **Glossary Grid** page is opened, right click on the newly created term and select **Open Term Editor** from the pop-up menu.



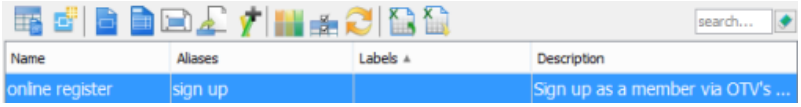
Right click to open term editor

3. In the **Term Editor** page, open **Definition** tab.
4. You can define aliases for the term and enter definition as description for the term. To insert an alias, click **Add** button and type the alias in the pop-up **Input** dialog box. To enter definition, enter under **Definition** directly.



Define aliases and enter definition

5. As a result, the columns of **Aliases** and **Description** are filled when you return **Glossary Grid** page.

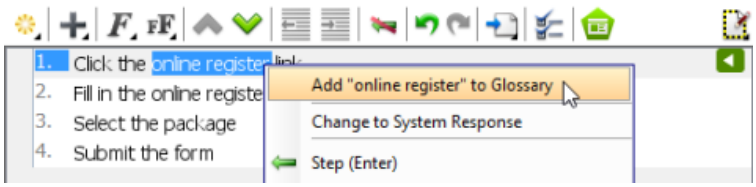


Completed glossary grid

NOTE: In Glossary Grid, you can jump to the source from which a term was defined by right clicking on the term and selecting **Transit From > %SOURCE_ELEMENT_NAME%** from the popup menu.

Identify term from flow of events

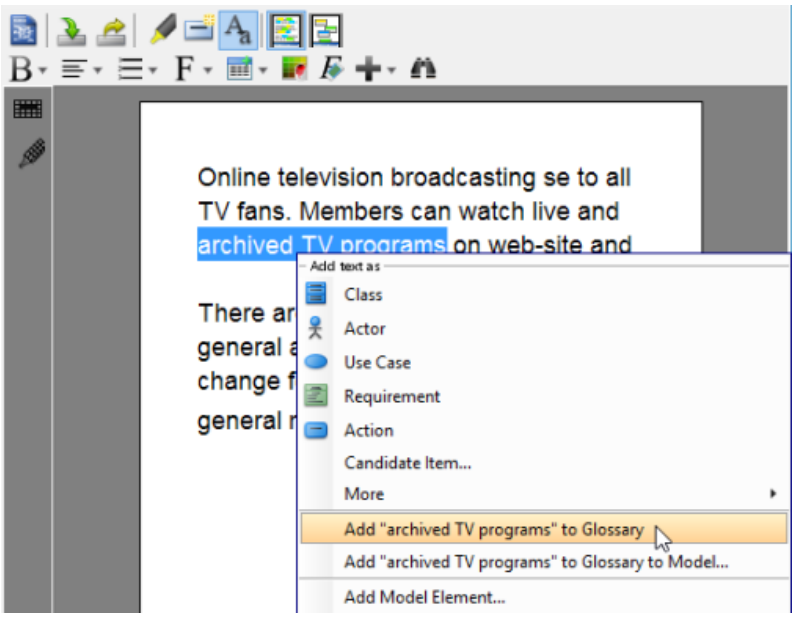
Highlight the specific term on flow of events editor, right click on it and select **Add "the highlighted term" to Glossary** from the pop-up menu.



Add "online register" to Glossary

Identify term from textual analysis

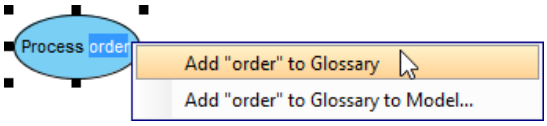
Highlight the specific term on textual analysis , right click on it and select **Add "[the highlighted term]" to Glossary** from the pop-up menu.



Add "archived TV programs" to Glossary

Identify term from shape name

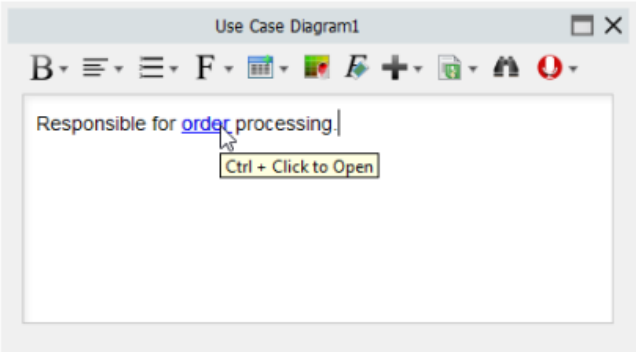
Highlight the specific term when editing a shape inline. Right click on it and select **Add "[the highlighted term]" to Glossary** from the pop-up menu.



Add term to glossary when renaming shape

Opening term

To read the definition of a term, press the **Ctrl** key and click on the term from description/flow of events content/shape name. By doing so, the glossary grid will be opened, with the selected term highlighted.



To open a term

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

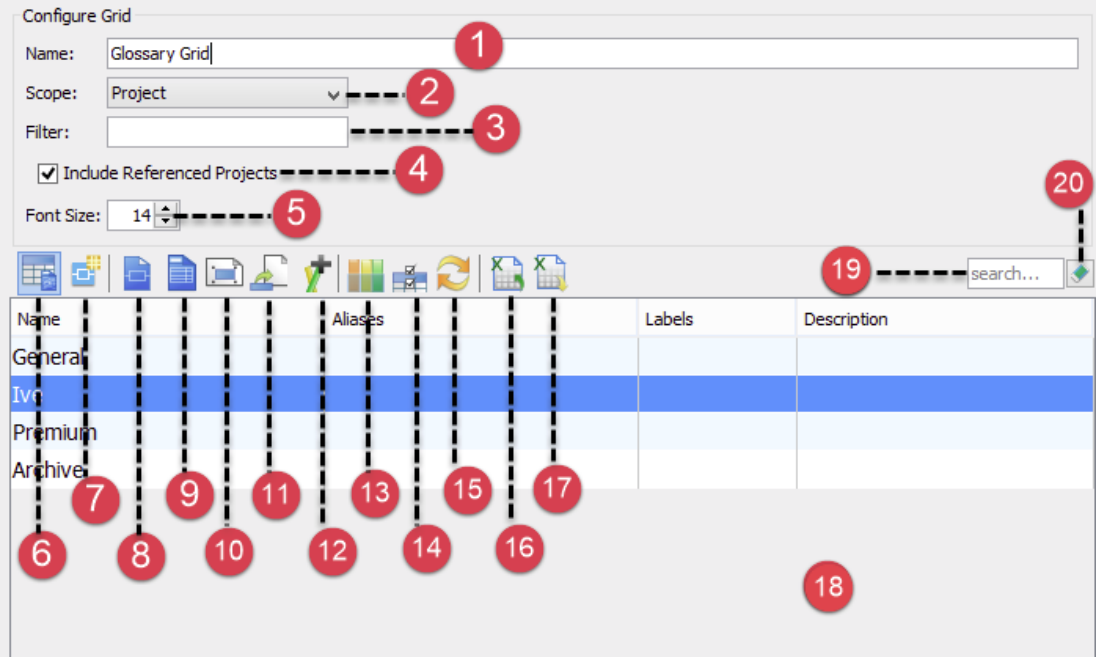
Using Glossary Grid

Glossary grid is a table where you can identify specific glossary term. In addition, you can define aliases and enterdescription for the glossary term. With Visual Paradigm , you can categorize the terms by defining and assigning label(s) to them.

Creating the Glossary Grid

To create a glossary, select **Modeling > Glossary** from the toolbar.

The overview of Glossary Grid



The Glossary Grid

No.	Name	Description
1	Name	The name of this **Glossary Grid**.
2	Scope	The location to look for the terms to list in grid. By default, terms are found from the whole project. You can change to find terms from specific model or package, or to find only terms right at the root level. You can also restrict the scope to all diagrams, to within a specific diagram or to all terms that has not been visualized in any diagram.
3	Filter	Apply filter to grid content. Text entered here is matched against the **Name** property of terms listed in grid. terms that do not contain the entered text in their name are hidden.
4	Include Referenced Projects	Check it to list also terms in referenced projects, in **Glossary Grid**.
5	Font Size	Click to adjust the font size of text in **Glossary Grid**.
6	Configure Grid	Click to show/hide the grid configuration panel, which allows you to enter the name of grid, the model element to be listed in grid, the scope and to apply filter to grid content.
7	New Term	Click to create a term.
8	Open Term Editor	Select a term in **Glossary Grid** and click this button to open the term editor for editing it.
9	Open Specification...	Select a term in **Glossary Grid** and click this button to open its specification.
10	Show View...	Select a term in **Glossary Grid** and click this button to list the diagrams that contains the view of the selected term.
11	Visualize...	Select a term in **Glossary Grid** and click this button to show it in a new or existing diagram.
12	Add Label to Selected Term(s)	Select a term in **Glossary Grid** and click this button to add labels to it.
13	Manage Label...	Click to add/edit/delete labels.
14	Configure Columns...	Click to select the property(ies) of terms to be listed in the grid, as columns.
15	Refresh	Click to refresh the grid content by showing the most updated information of terms listed.
16	Export to Excel	Click to export grid content to Excel file.

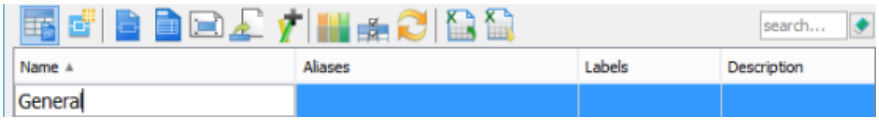
17	Import from Excel	Click to import grid content from exported Excel file.
18	List of terms	Terms are listed here.
19	Search	Find term(s) by entering search criteria.
20	Clear	Click to clear the text entered in **Search** box.

The fields in Glossary Grid

Creating term in Glossary Grid

To create a term in **Glossary Grid**:

1. Click on **New Term** above the **Glossary Grid**.
2. Enter the name of term.



Creating term in Glossary Grid

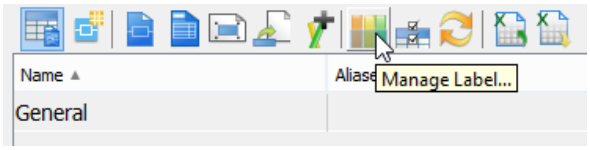
3. Press **Enter** to confirm editing.

Organizing terms with labels

You can categorize the terms by defining and assigning label(s) to them.

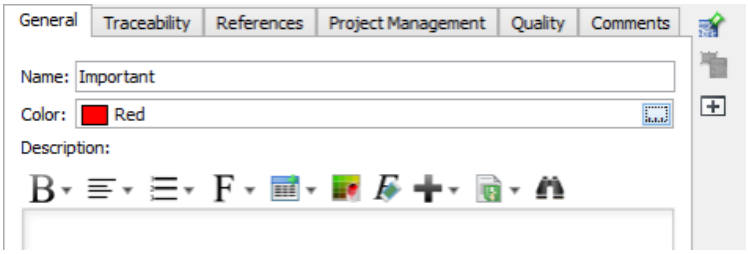
Creating label

1. Click on **Manage Label...** at the top of the **Glossary Grid**.



Manage Label

2. In the **Manage Label** window, click **Add...**.
3. In the **Glossary Label Specification** window, enter the name of label and give it a unique color.

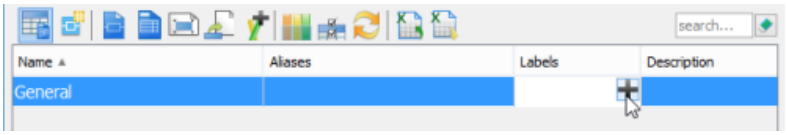


Entered label's details

4. Click **OK** to confirm editing.

Adding label to a term

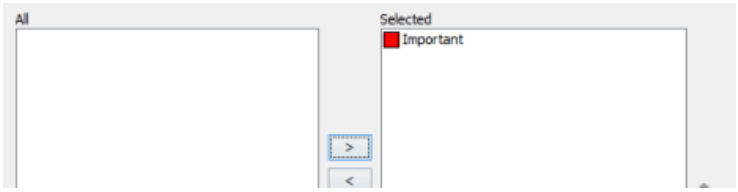
1. Select the desired term in the **Glossary Grid**.
2. Click the **+** button under the **Labels** column.



Add label

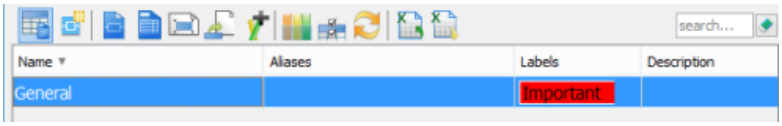
> **NOTE:** If the Labels column doesn't appear, click **Configure columns...**, open the **Properties** tab and select it under the **Others** folder..

3. Select the label(s) to add to the term and click **>**.



Add label

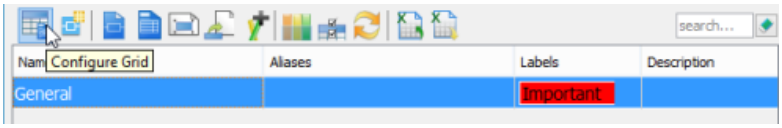
4. Click **OK** to apply and return to the **Glossary Grid**.



Label added to term

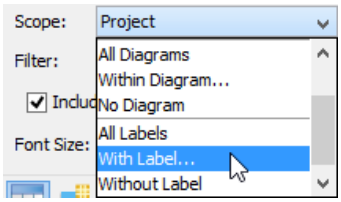
Listing terms by their label

1. Click on **Configure Grid...** at the top of the **Glossary Grid**.



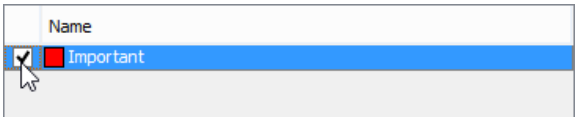
Configure Grid

2. Under the drop down menu **Scope**, select **With Label...**.



Adjust scope

3. In the **Scope** window, select the label to be included into the scope.



Selecting label

4. Click **OK** to confirm and return to the **Glossary Grid**. From now on, only terms that contain the selected labels will be listed in the grid.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Grid diagram

A grid is capable in listing model elements in tabular form, with them appear as row and properties as columns. This chapter will teaches you how to create and configure grid.

Creating grid diagram

The steps required to create a grid.

Creating element in grid

Shows you how to create model element in a grid.

Configuring property columns

As mentioned before, columns of model elements are presented as columns in grid. You will see how to add and remove property columns.

Setting the scope of grid content

Shows you how to set the scope (e.g. project, model, diagram) of grid content.

Filtering and searching in grid

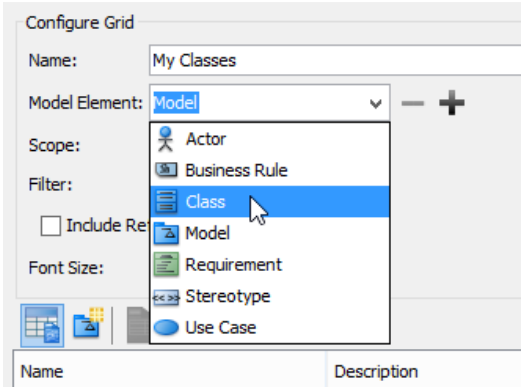
Filter model elements in grid by using filter and search.

Creating grid diagram

The Grid Diagarm provides a convenient way to view specific type(s) of model element within your project. You can create a grid to list whatever model element type(s) you like and select the properties of model element being shown inside the grid.

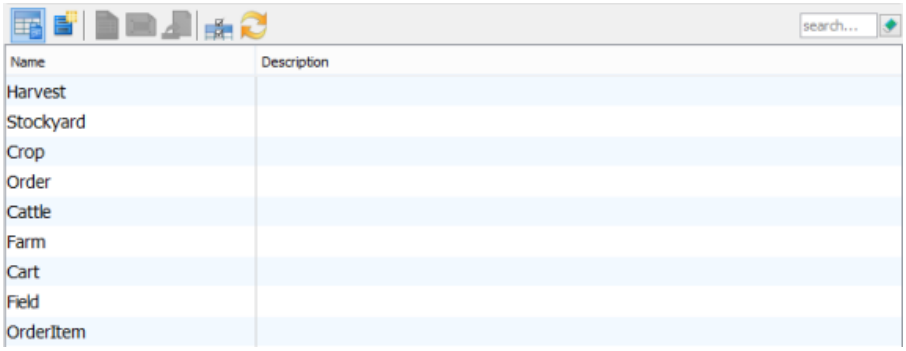
To create a grid:

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Grid**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.
6. The **Grid Diagram** is opened. Select a model element type from the drop down menu of **Model Element**. Click **More...** to list all the available choices. If you want to list multiple model element types say, class and use case, click "+" to add more type row(s).



Select Class

The model element of selected type will then be shown on the grid.



The model element of selected type is shown

Related Resources

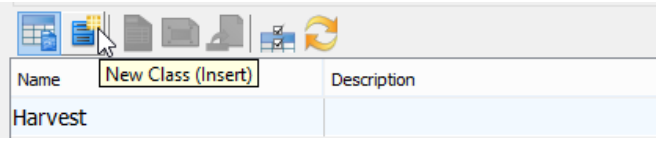
The following resources may help to you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating element in grid

Besides creating elements by drawing them on diagram, you can also create them in grid. Creating elements in grid gives you an overview on all elements of same type.

1. Click on the **New [Element]** button, where **Element** refers to the type of model element you have chosen in the **Model Element**'s drop down menu.



Create a model element in grid

2. Enter name for the newly created model element and then press **Enter** to confirm.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

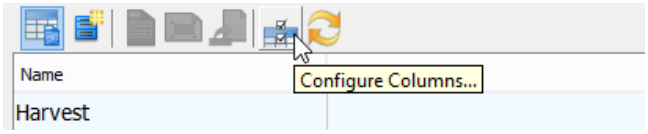
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Configuring property columns

In grid, rows represent model elements while the columns show their properties. Name and description columns are shown by default, in addition, you can optionally add or remove columns to display the data you are interested in.

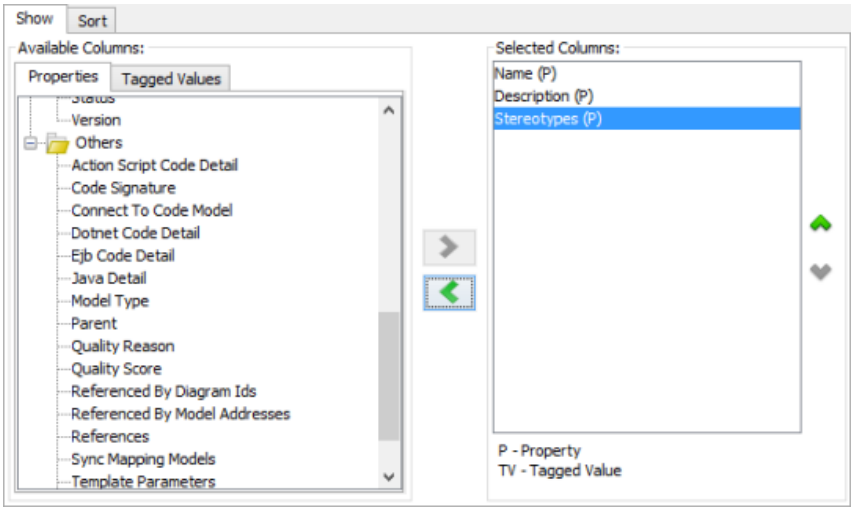
Adding extra property column

1. Click **Configure Columns...** on top of the grid.



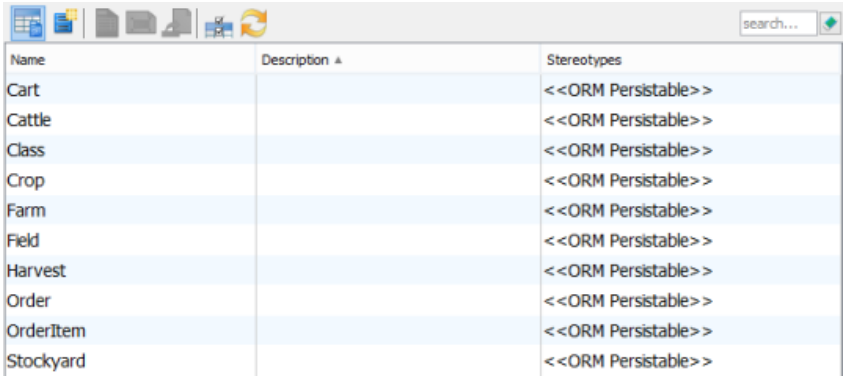
Configure Columns

2. In the **Configure Columns** dialog box, select your preferred property under **Properties** tab and then click **OK** button.



Choose properties

3. Click **OK** to confirm editing and return to the grid.



Grid updated

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

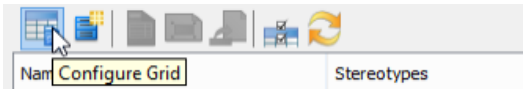
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Setting the scope of grid

In grid, you can customize the scope of grid content after you have created various types of model elements. After that, your desired model elements within a particular scope will be shown on the grid.

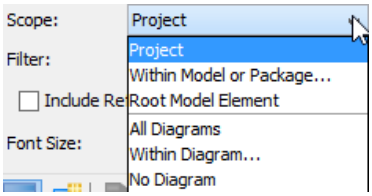
To set the scope of grid content:

1. Click **Configure Grid...** on top of the grid.



Configure Grid

2. Select your desired scope from the combo box of **Project**.



Select scope

> **NOTE:** **Within Model or Package**: Select this to show all selected model element types within a particular model/ package.
> **Root Model Element**: Select this to show all selected model element types under root node.
> **All Diagrams**: Select this to show all selected model element types within all diagrams.
> **Within Diagram**: Select this to show all selected model element types within a particular diagram.
> **No Diagram**: Select this to show all selected model elements without diagram.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

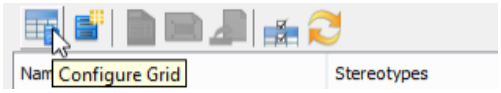
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Filtering and searching in grid

To locate a specific model element or model elements in grid, you can make use of the filtering or searching feature. Filtering excludes rows that do not match with a specified criteria, while searching highlights the rows that matches a specified criteria.

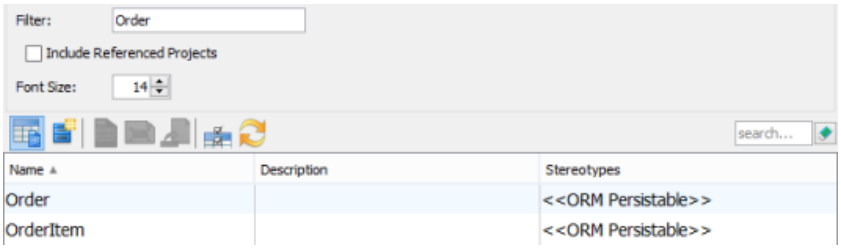
Filtering

1. Click **Configure Grid...** on top of the grid.



Configure Grid

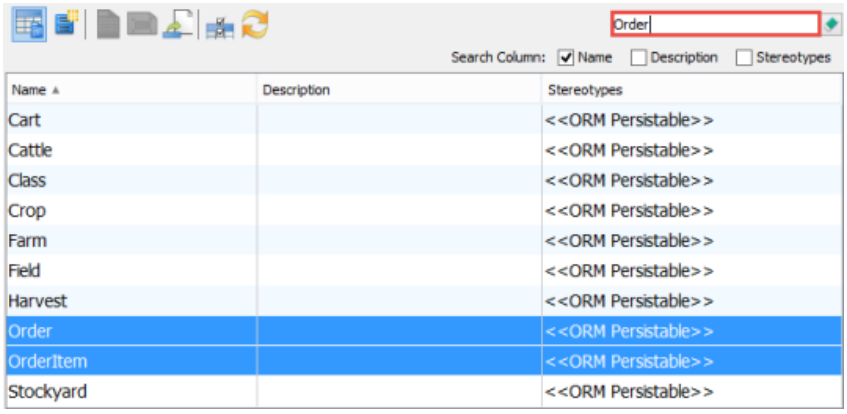
2. Enter in the **Filter** field the name of the model element(s) that you want to find or part of its name. You can use an asterisk (i.e. the * character) to indicate wildcard characters. Upon typing, the grid is updated to exclude the rows that do not match with the entered text.



Grid content filtered

Searching

Enter in the **Search** field the key words of the model element(s) that you want to find. Key words refer to words that are contained by the properties (columns) listed in grid. You can use an asterisk (i.e. the * character) to indicate wildcard characters. Upon typing, the grid is updated to highlight the rows that match with the entered text.



Searching in grid

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Adding new property to model elements through grid

Sometimes, you may find the standard modeling notations not enough in supporting your domain specific needs. For example, if you are working on a project that relies heavily on the use of third-party libraries, you might want to specify in UML component diagram the API source of those components modeled. And in business process modeling, you might want to record the locations where the tasks are performed. Usually, these kind of specific needs cannot be directly supported by the standard notations.

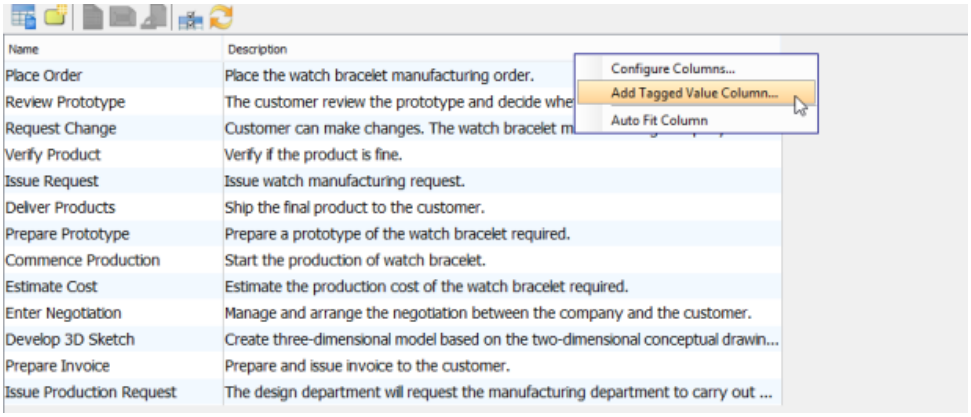
Fortunately, OMG Unified Modeling Language (UML) introduces stereotype and tagged values, which provides designers with ideal solutions to these situations. Stereotype and tagged values are kind of extensibility mechanisms that allow designers to extend the vocabulary of UML in order to create new model elements. With the use of stereotype and tagged values, designers can introduce model elements with domain/problem specific properties.

Starting from UML 2.0, tagged values are considered to be attributes of stereotype. Yet, Visual Paradigm allows to use tagged values independently, that is, to add tagged values directly to model elements as custom properties instead of prior attachment to any stereotype. Besides, Visual Paradigm allows you to take the advantage of stereotypes and tagged values in not just UML but all the modeling notations like BPMN and even in ERD, DFD, etc. Simply put with Visual Paradigm, you can add custom properties to any model elements by creating tagged values.

There are several ways you can take to create tagged values to model elements. One is to create in a Grid. This method is extremely efficient when you want to add into and fill in the tags for multiple model elements in the same type. You will see how it works in this page.

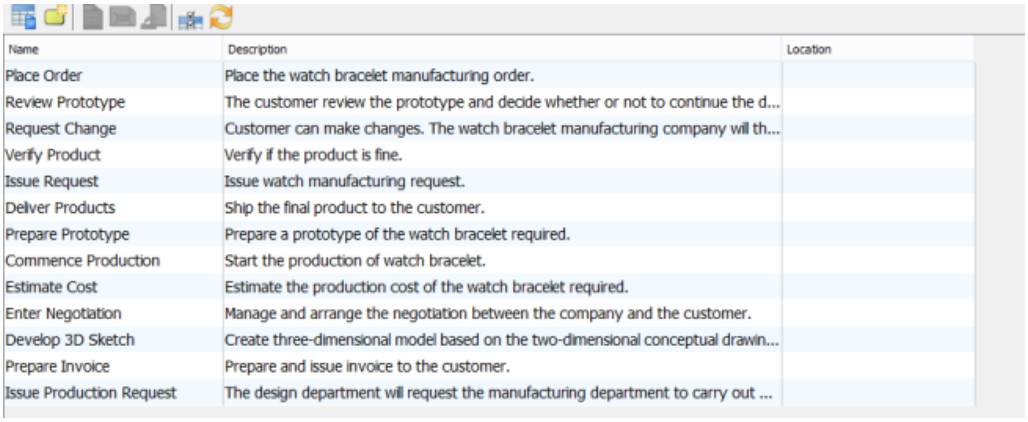
Adding a new tag to model elements in grid

1. Open the grid where the target model elements are listed or create a grid if necessary.
2. Right click on the column header at the top of a grid and select **Add Tagged Value Column...** from the popup menu.



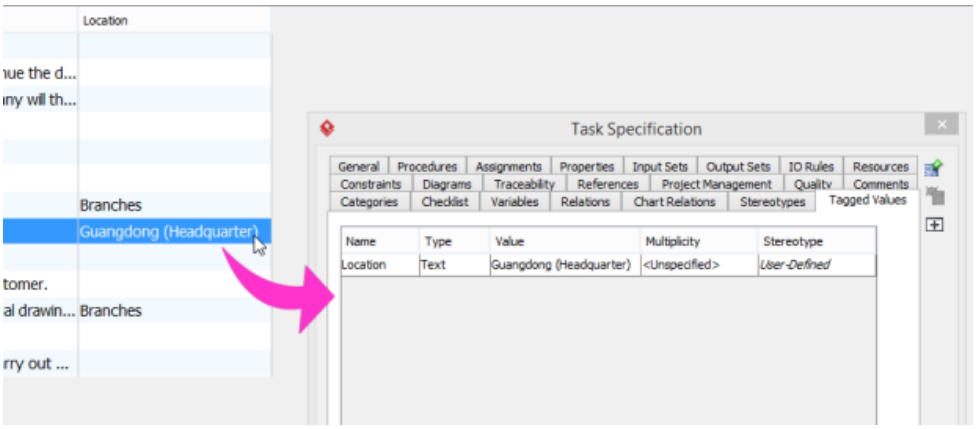
To add a tagged value column

3. Enter the tag name in the **Input** dialog box and click **OK** to confirm.
4. You will see a new column appear in the last column of the grid.



New "Location" column added to a grid

5. Now, you can double click on a cell in that column to enter the value of the property (tag). When and only when you have entered a value, a tag will be added into that specific model element. Note that once a tag is added, you cannot remove the tag by clearing the cell content. If you want to remove the tag, you have to do it in the model element specification window.



Tagged value added and specified

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business process diagram

This chapter talks about how to create a business process diagram, with description of most of the common notations about their usage and tips when creating or editing them.

Creating business process diagram

Shows you how to create a business process diagram as well as to update the format of ID.

Pool and lane

Pool and lane are used to model roles in a process. This part will cover some of the common operations with pool and lane, such as to change their orientation, to define a black box and to move lanes up and down, left and right (depending on the orientation of pool).

Task and sub-process

Task and sub-process are used to model the activity needed to do in a process. This part will cover the use of marker, a description of various types of task, how to define working procedure for task and how to create a sub-process.

Event

An event in a business process refers to something that happens and affects the flow of process. This part will describe the various types of start, intermediate and end event.

Gateway

Gateway is a kind of flow objects which is used to direct sequence flows within a process, based on certain condition(s). There are several kinds of gateway for different kinds of control behavior. We will go through each of them in detail.

Sequence and message flow

There are two types of connectors for modeling flows in a process - Sequence flow and Message flow. You will see how to correct invalid flow and how to visualize message that pass between pools.

Choreography task and sub-process

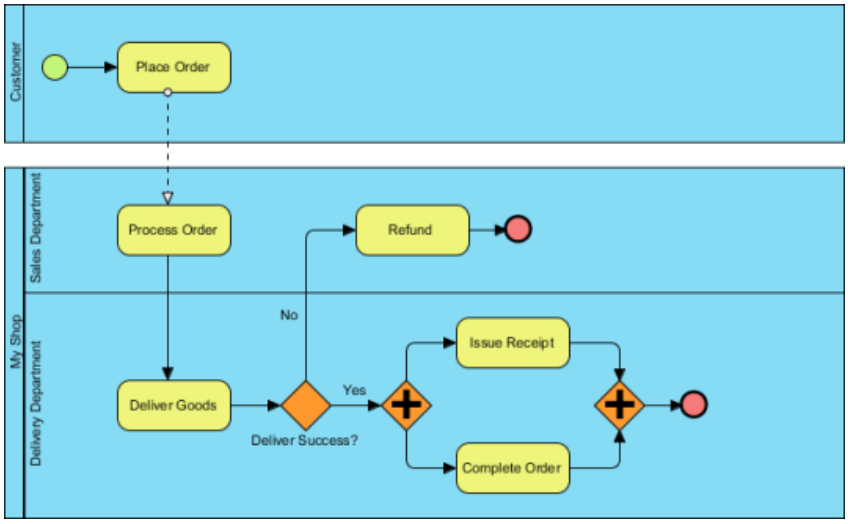
A choreography is a type of process which defines the sequence of interaction between participant. You will learn how to set participants to choreography task and sub-process.

Data object

You can use data objects to model data within process flow. You will learn how to define states for data object.

Drawing business process diagram

Business Process Modeling Notation (BPMN) is a graphical representation for designing and modeling business processes visually. It is a standard for business process modeling and provides a graphical notation for specifying business processes in a business process diagram (BPD).



A sample business process diagram

Creating business process diagram

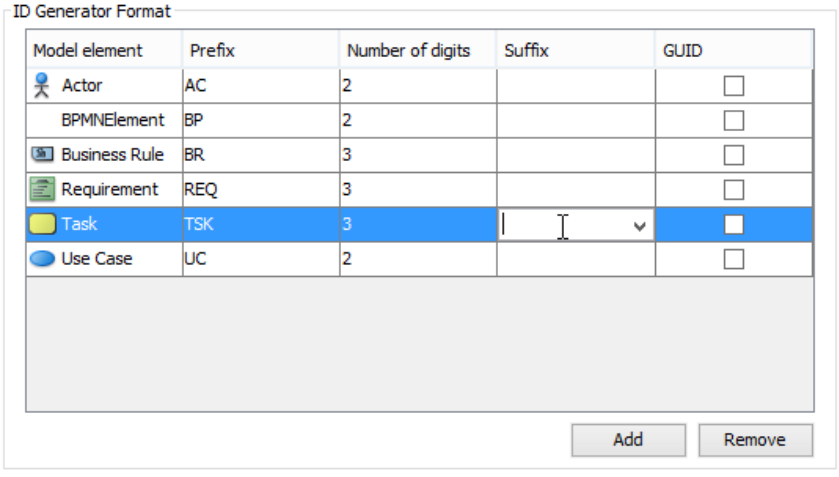
1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Business Process Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Assigning IDs to model elements

It is possible to assign IDs to objects in business process diagram. By default, IDs are assigned by following the order of object creation, starting from number 1. However, you can define the format or to enter an ID manually.

Defining the format of ID

To define the format of ID, open the **Project Options** window by selecting **Windows> Project Options** from the toolbar. Select **Diagramming** from the list on the left hand side and open the **Model Generation** tab. Click **Add** and then select the type of model element that you want to change its ID format (E.g. Task). Then, you can adjust the format, say to specify the prefix, the number of digits and suffix.



Defining format of ID

Below is a description of the options.

Option	Description
Prefix	Text to be added before the number
Num of digits	The number of digits of the number. For example, when digit is 3, ID "1" will become "001"
Suffix	Text to be appended to the number
GUID	A randomly generated string. Note that the string will be very long. And by selecting this option, the prefix, num of degits and suffix options will be ignored.

Controlling how ID is shown on new diagram

By default, ID is just a text property that won't appear in diagram. However, you can make it appeared either near or within a shape.

Different looks of a task when ID is not shown, ID is shown as label and ID is shown below caption

To define the way ID is shown in BPD, open the **Project Options** window by selecting **Windows > Project Options** from the toolbar. Select **Diagramming** from the list on the left hand side and open the **Business Process > Behavior** tab.

In the **ID Generator Format** section, select whether or not to show ID on diagram and whether to show it as a label that is attached to a shape or as text below caption. If you select **Show as Customized**, you can edit in the table of model element the way ID is shown.

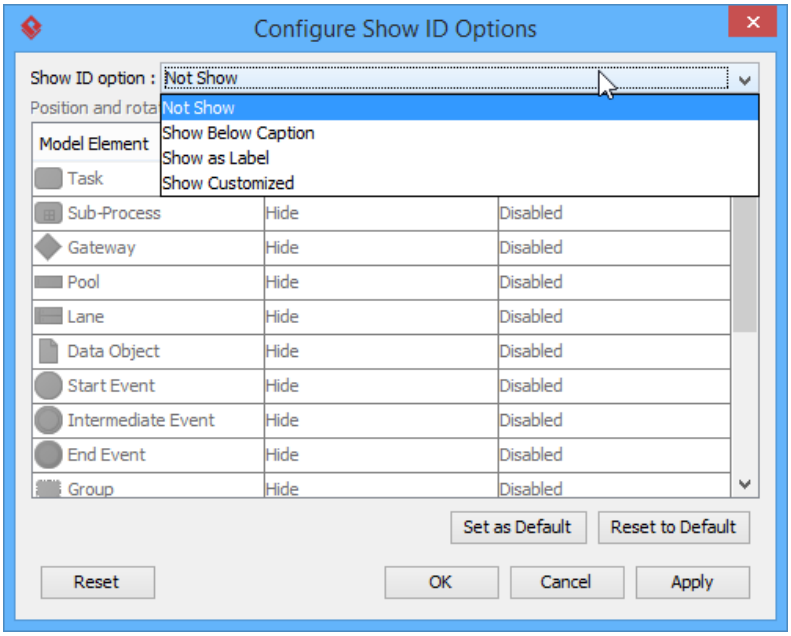
Note that the setting is effective only in newly created diagrams.

Controlling how ID is shown on an existing diagram

The previous section described how to set the appearance of ID in new diagrams. Here, you can see how to change the ID appearance of an existing diagram.

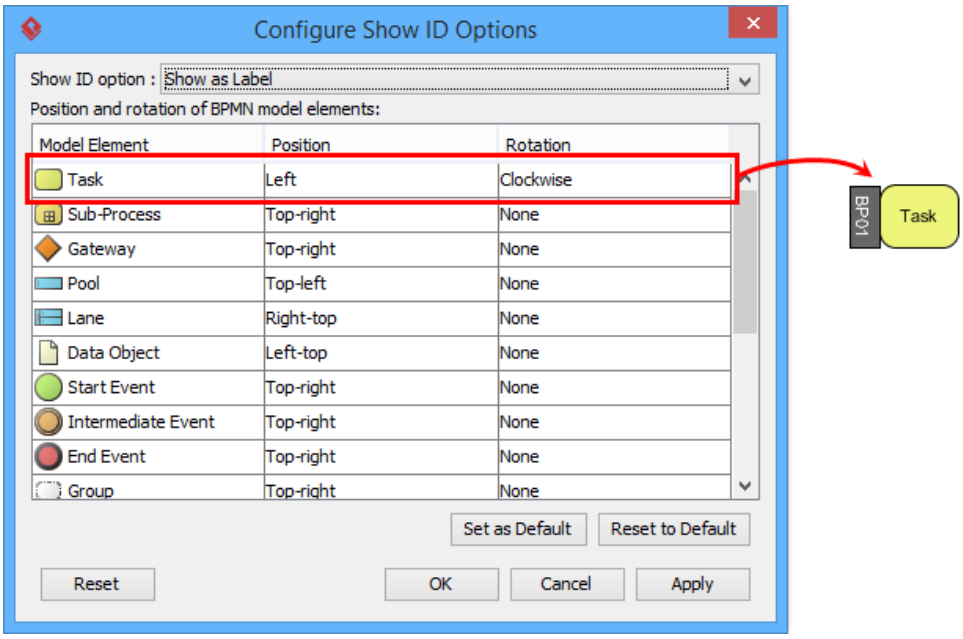
To do this, right click on the BPD you want to edit and select **Presentation Options >Configure Show ID Options...** from the pop-up menu.

In the **Configure Show ID Options** window, click on the drop down menu **Show ID Option** and select if you want to show IDs. If yes, select where to show, such as below caption or as a label.



To configure the whether or not to show ID

If you have selected to show as label, you can adjust in further the position of ID, relative to the shape (e.g. Top right of task) and the rotation.



To make ID of task show as label, position at the left of shape

ID assignment

There are several ways that you can assign an ID to an element, including:

- Through the specification dialog box (Right click on it and select **Open Specification...** from the popup menu)
- Through the ID label (available only when ID is shown as label on diagram)
- Through the **Property Pane**

Nested ID

When you draw something in a lane/pool or to drill down a sub-process into another level and draw something, this forms a nested element hierarchy like a task is contained by its parent pool. ID is formed nested according to the hierarchy. For instance, while a pool has '3' as ID, its children have 3.1, 3.2... as IDs. You can turn this function on and off.

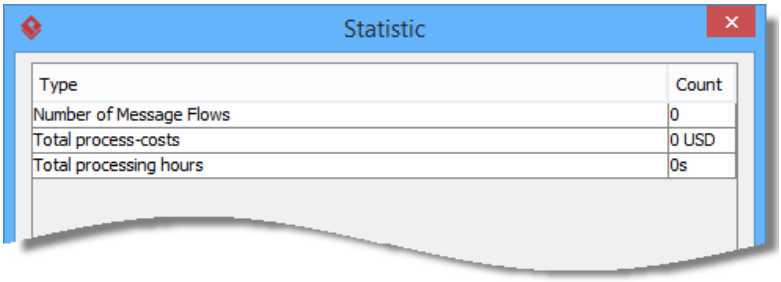
To turn on or off, right click on a BPD and select **Diagram Content > Edit IDs...** from the popup menu. At the bottom of the dialog box, check or uncheck the option **Sub-Level ID**.

Showing process statistic

Process statistic refers to the results of the statistical analysis that can be conducted upon your process. There are three types of figures: number of message flows, total process-costs and the total processing hours.

To show process statistic:

1. Right click on the background of the business process diagram.
2. Select **Utilities > Show Statistic...** from the popup menu. This opens the **Statistic** window like this:



Process statistic

Below is a description of figures.

Figure	Description
Number of Message Flows	The number of message flows that exist in the current diagram.
Total process-costs	A summation of costs specified for tasks and sub-processes in the current diagram.
Total processing hours	A summation of duration specified for tasks and sub-processes in the current diagram.

Description of figures in process statistic

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

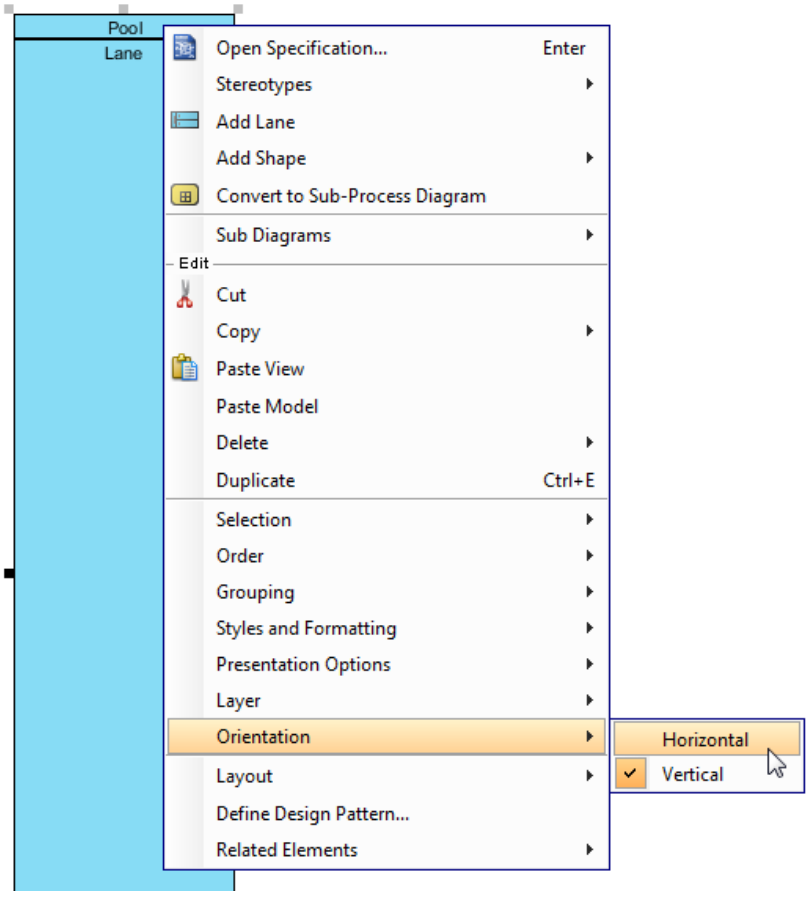
Pool and lane

A pool represents a participant who takes part in a process. It is visually a rectangular box that can contain flow objects like task and activity. A lane is a sub-partition in a pool. It is often used to represent internal roles or a department under the role represented by pool.

Horizontal pool that contains two lanes

Orientation of pool

In a business process diagram, pool and lane can be shown either vertically or horizontally. To change the orientation of pool and contained lane, right click on the pool header and select **Orientation > Vertical/Horizontal** from the popup menu. Note that this function is available when the pool is empty.

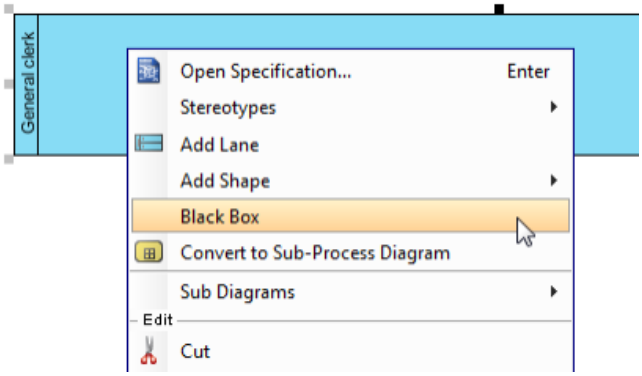


Changing the orientation of pool from vertical to horizontal

NOTE: You can only change the orientation of pool/lane when it contains no flow object

Defining black box pool

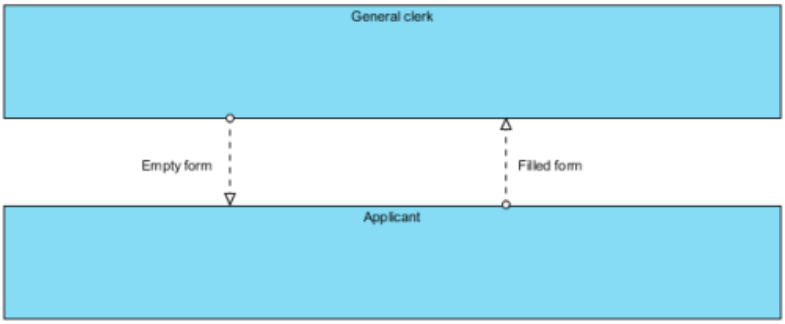
A pool can be shown as an empty box, called a black box. A black box represents a role solely, with all details hidden - You cannot create flow objects in it. To define a black box pool, right click on an empty pool and select **Black Box** from the popup menu.



Defining a black box

NOTE: You can only create a black box for an empty pool that has neither flow objects nor lanes in it.

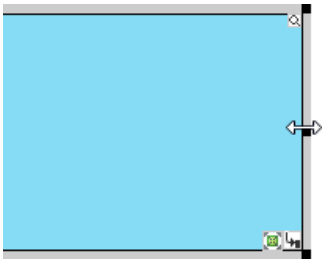
You can create message flows between black boxes to represent the collaboration between participants or create message flows between flow objects in another pool/lane with a black box.



Black boxes with message flows in between

Stretching of pool

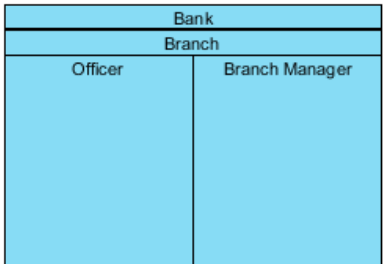
When you create a pool, it is automatically expanded to fit the width or length of diagram. We call this behavior *stretched*. When a pool is stretched, the pool and the contained lane(s) will expand or collapse by following the size of diagram and you cannot resize it manually. If you want to make a pool freely resize-able, you need to turn the stretch behavior off. To change the stretch option, right click on the pool involved and select **Presentation Options,** then select **Auto Stretch > Off** from the popup menu.



Pool can be resized freely when auto-stretch is turned off

Creating nested lanes

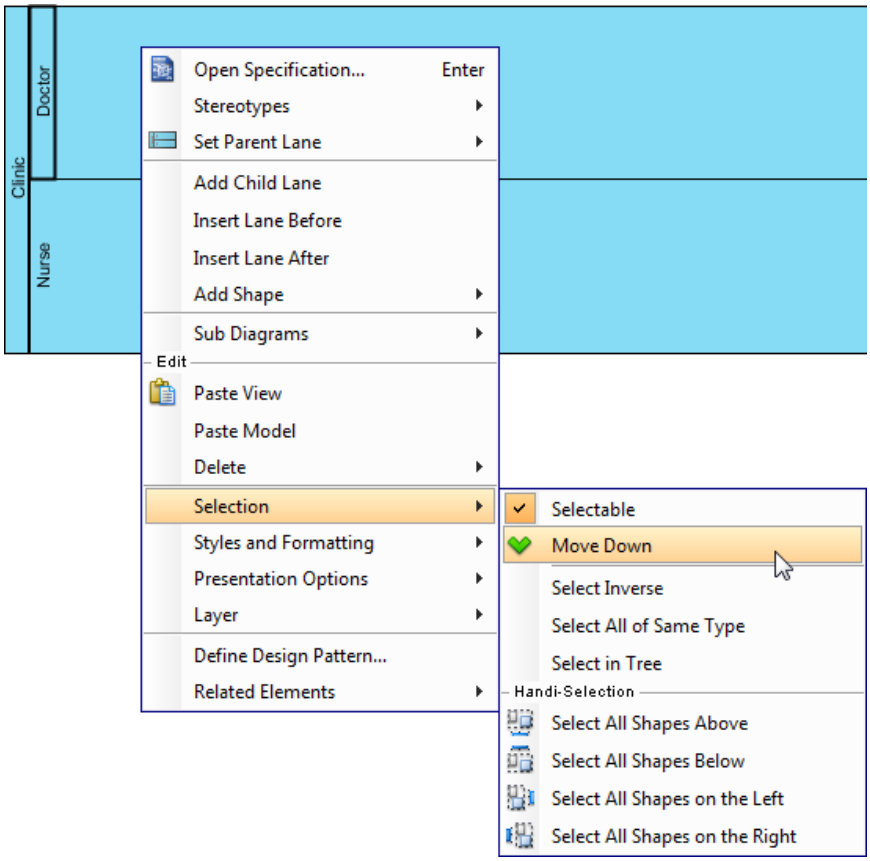
While a lane represents a sub-partition of a pool, a lane itself can contain lanes to form a nested structure. To create a nested lane, right click on a lane and select **Add Child Lane** from the popup menu



Nested lanes

Reordering lane

You can change the position of lanes within a pool by moving them up and down. To reorder a lane, right click on the lane you want to reorder and select **Selection > Move Down** from the popup menu.



Moving a lane downward

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

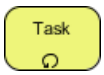
Task and sub-process

A business process is mainly formed by activities that need to be performed to complete the process. There are two kinds of activities - task and sub-process. A task is an atomic activity which represents work that cannot be broken down. On the contrary, sub-process represents work that can be broken down to a finer level of detail.

Task and sub-process

Task markers

You can assign markers to task. There are three markers: Loop, Multi-Instance, Compensation. A task can have one or two of these markers. Assignment of markers is done through the specification dialog box of task.

Name	Representation	Description
Loop		This marker indicates that the task will loop as long as the condition that defined in the loop is true. The condition is evaluated in each iteration, at either the beginning or the end of iteration. This marker can be used in combination with the compensation marker.
Multi-instance (parallel instances)		This marker indicates the execution of task in a desired number of instances or in a data driven approach. The instances will be started at the same time.
Multi-instance (sequential instances)		This marker indicates the execution of task in a desired number of instances or in a data driven approach. The instances will be executed one after the other.
Compensation		To undo (cancel) the result of another activity that have already been successfully completed. The execution of compensation task is due to the undesired results and possibily side effects produced by antoher activity that need to be reversed. A compensation task is performed by a compensation handler, which performs the steps necessary to reverse the effects of an activity.

Different markers of task

Adding a Loop marker

1. Right click on a task and select **Open Specification...** from the popup menu.
2. Select **Standard Loop** for **Loop type**. Click **OK** to confirm the changes.

> **NOTE:** You can click on the **...** button next to **Loop type** to set the loop condition, counter and the maximum number of iteration.

Adding a Multi-instance marker

1. Right click on a task and select **Open Specification...** from the popup menu.
2. Select **Multi-Instance Loop** for **Loop type**. Click **OK** to confirm the changes.

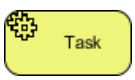
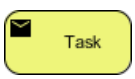
> **NOTE:** You can click on the **...** button next to **Loop type** to set the ordering of loop, either parallel or sequential.

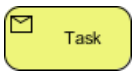
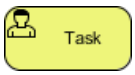
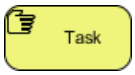
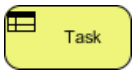
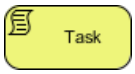
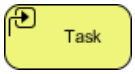
Adding a Compensation marker

1. Right click on a task and select **Open Specification...** from the popup menu.
2. Check **Compensation** at the bottom of specification and click **OK** to confirm the changes.

Task types

There are several types of task for you to separate the behavior of different tasks. You can set a type by right clicking on a task and selecting **Type**, then the type from the popup menu.

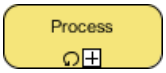
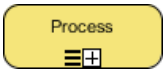
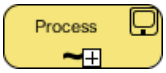
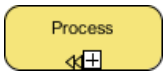
Name	Representation	Description
Service		A service task is a task that uses some sorts of service, e.g. a Web service.
Send		A send task is a task that sends a message to an external participate. The task is said to be completed once the message has been sent.

Receive		A receive task is a task that waits for a message to arrive from an external participant. The task is said to be completed once the message has been received.
User		A user task is a task performed by a human with the assistance of a software application.
Manual		A manual task is a task that is performed without the aid of any business process execution engine.
Business Rule		A business rule task allows the process to provide input to a business rules engine and to get the output from engine.
Script		A script task involves a script defined by modeler or implementer in a language that a business process engine can understand and is executed by a business process engine.
Reference		A reference task refers to another task for its content.

Types of tasks

Sub-process markers

You can assign markers to sub-process. There are four markers: Loop, Multi-Instance, Ad-hoc and Compensation. A sub-process can have up to three markers, excluding the marker for collapsed: A loop/multi-instance marker, an Ad-hoc marker, and a Compensation marker. Assignment of markers is done through the specification dialog box of sub-process.

Name	Representation	Description
Loop		This marker indicates that the sub-process will loop as long as the condition that defined in the loop is true. The condition is evaluated in each iteration, at either the beginning or the end of iteration. This marker can be used in combination with the ad-hoc and/or compensation marker.
Multi-instance (parallel instances)		This marker indicates the execution of sub-process in a desired number of instances or in a data driven approach. The instances will be started at the same time.
Multi-instance (sequential instances)		This marker indicates the execution of sub-process in a desired number of instances or in a data driven approach. The instances will be executed one after the other.
Ad-hoc		This marker indicates that a sub-process is a group of activities that have no required sequence relationship. The sequence and number of performances for activities are determined by the performers of the activities.
Compensation		To undo (cancel) the result of another activity that have already successfully completed. The execution of compensation sub-process is due to the undesired results and possibly side effects produced by another activity that need to be reversed. A compensation sub-process is performed by a compensation handler, which performs the steps necessary to reverse the effects of an activity.

Different markers of sub-process

Adding a Loop marker

1. Right click on a sub-process and select **Open Specification...** from the popup menu.
2. Select **Standard Loop** for **Loop type**. Click **OK** to confirm the changes.

> **NOTE:** You can click on the **...** button next to **Loop type** to set the loop condition, counter and the maximum number of iteration.

Adding a Multi-instance marker

1. Right click on a sub-process and select **Open Specification...** from the popup menu.
2. Select **Multi-Instance Loop** for **Loop type**. Click **OK** to confirm the changes.

> **NOTE:** You can click on the **...** button next to **Loop type** to set the ordering of loop, either parallel or sequential.

Adding an Ad-hoc marker

1. Right click on a sub-process and select **Open Specification...** from the popup menu.
2. Make sure the type of sub-process is set to be **Embedded Sub-Process**. Check **Ad-hoc** in the **Details** section and click **OK** to confirm the changes.

Adding a Compensation marker

1. Right click on a sub-process and select **Open Specification...** from the popup menu.
2. Check **Compensation** at the bottom of specification and click **OK** to confirm the changes.

Sub-process types

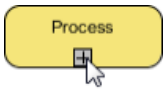
There are several types of sub-process for you to separate the behavior of different sub-processes. You can set a type by right clicking on a sub-process and selecting **Type**, then the type from the popup menu.

Name	Representation	Description
Embedded		An embedded sub-process is a sub-process that models its internal details in another process.
Reusable		A reusable sub-process calls a pre-defined process.
Reference		A reference sub-process refers to another sub-process.

Types of tasks

Breaking down a sub-process

A sub-process can be opened up to model the detail in a lower level. To open up a sub-process, click on the plus marker in sub-process and select **New Business Process Diagram**. This will create a new business process diagram that belongs to the sub-process. In the sub-process diagram you will see the in/out flows that allow you to connect the flow from parent to sub-diagram. Click here you if you want to learn more about in/out flows.

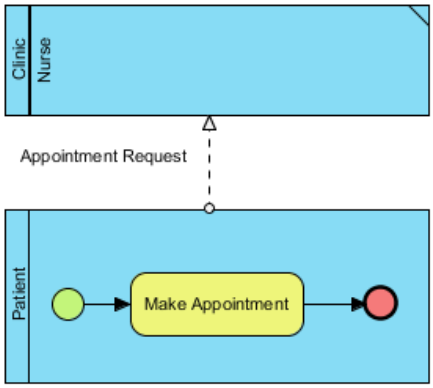


To break down a sub-process

NOTE: Once a sub-process diagram is created, its detail will be shown as the sub-process shape as a thumbnail of the diagram. To hide the thumbnail, click on the minus marker at the bottom of sub-process to turn it off.

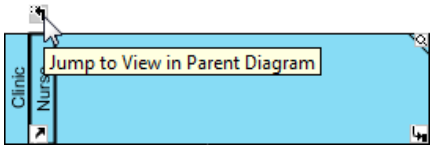
Re-using elements from parent diagram

In the sub-process diagram, you can re-use pools, lanes and flow objects that appear in the parent diagram. To do this, right click on the sub-process diagram and select Add Pools/Lanes/Sub-Processes/Gateways from Parent Diagram... from the popup menu, and choose the element to reuse. Elements being re-used will have dog ear appeared at their corners.



A sub-process diagram with a lane reused from parent diagram

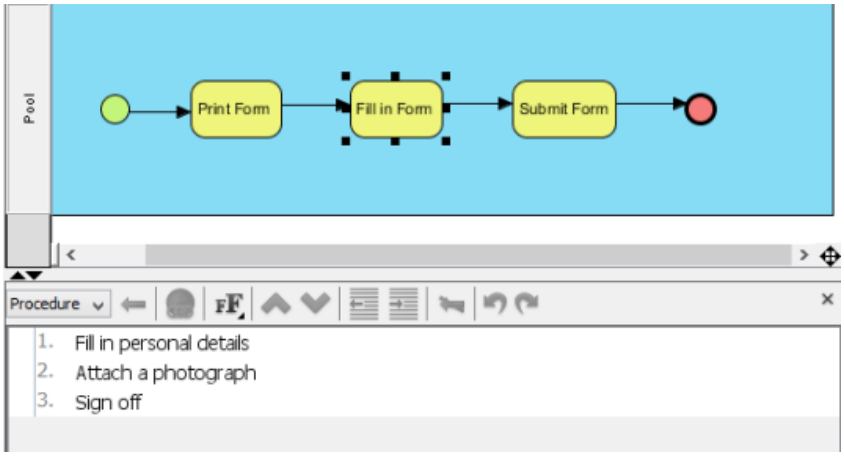
You can jump from a re-used element back to the parent diagram through the resource-centric interface.



Jump to parent diagram

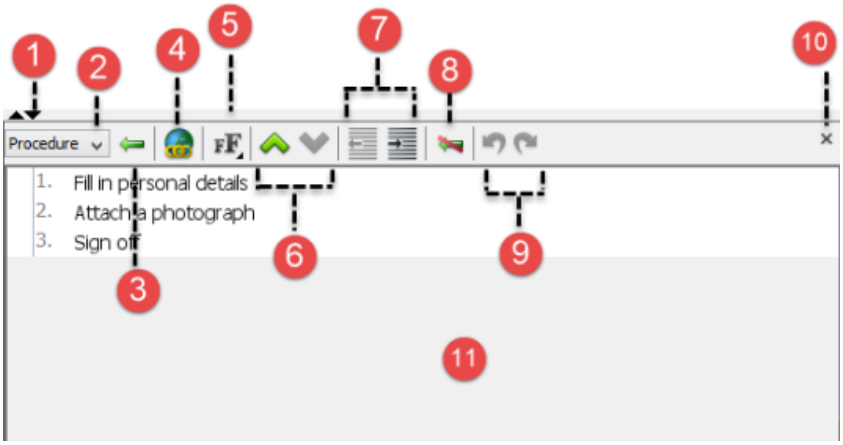
Defining procedure of activity

An activity within a process represents work need to be done. Each activity can be formed by a number of steps. For example, a task Process Application involves 2 steps - validate application, confirm application. To document the steps of an activity, you can make use of the procedure editor.



Procedure of a task

An overview of procedure editor



An overview of procedure editor

No.	Name	Description
1	Collapse/Expand	Click on the triangle on the left hand side to maximize the editor. On the contrary, click on the inverted triangle to minimize the editor.
2	Procedure selector	You can define multiple set of procedure per activity. Click on this drop down menu to select the one you want to read/edit.
3	Step	Click on this button to create a step under the step selected in editor.
4	Hyperlink...	Add a link in the selected step for reference.
5	Font formats selector	There are three buttons. While the first one increases the font size for one level, the second one decreases the font size for one level and the third button resets the font size setting to default.
6	Font size setting selector	Click on this drop-down menu to select the sizes of highlighted text. Press **Grow Font** button to increase the font size for one level, press **Shrink Font** button to decrease the font size for one level and press **Default Font** button reset the font size setting to default. Moreover, you can adjust the font size for the highlighted text manually by slider.
7	Reorder step	Click on **Move Up** button to move the selected step upwards or **Move Down** button to move the selected step downwards.
8	Decrease Indent/ Increase Indent	Click on **Decrease Indent** button to reduce the indentation of the selected step or click on **Increase Indent** button to indent the selected step.
9	Undo/ Redo	Click on **Undo** button to revert change or click on **Redo** button to redo reverted change.
10	Close editor	Click on this button to close the editor.

11	Steps editor	The place where you can read and edit steps.

Description of procedure editor

Showing/Hiding procedure editor

The procedure editor is opened in business process diagram by default. To hide it, right click on the background of business process diagram and deselect **Show Procedure Editor** from the popup menu. You can select the same menu to show it when hidden.

> **NOTE:** Alternatively, you can close the editor by clicking on the cross button at the top right corner of editor panel.

Documenting the procedure

1. Select the task or sub-process that you want to document its procedure.
2. Click on the first row labelled 1 and enter the first step.
3. Press **Enter** to go to the next step. You can create a sub-step by pressing **Tab** on a step. Pressing **Shift-Tab** decreases the indentation of a sub-step. Repeat step 2 and 3 to enter the remaining steps the activities involve.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Event

An event in a business process refers to something that happens and affects the flow of process. There are three types of events: Start, intermediate and end.

Start, intermediate and end events with different kinds of triggers and results

Start event

A start event indicates the place where and possibly why a process start. Since start event is used for initiating a process, it does not have any incoming sequence flow.

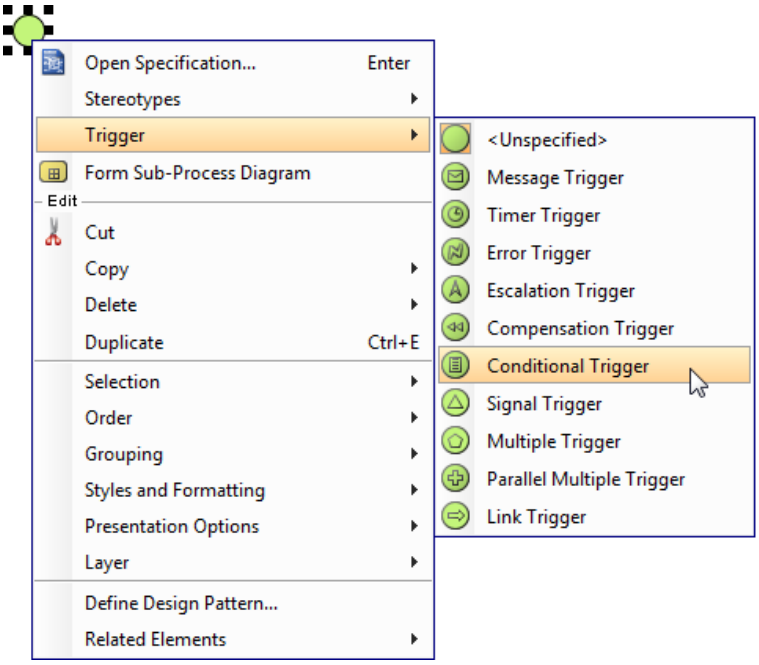
You can define a trigger for start event, to show the condition(s) that will cause a process to initiate.

Trigger name	Representation	Description
None		The none start event does not have a defined trigger.
Message		This trigger starts the process by receiving a message from a participant.
Timer		This trigger starts the process in a specific time-date or a specific cycle (e.g. every Friday).
Error		This trigger starts an in-line event sub-process when an error occurs. Note that this trigger can only be used with an event-sub-process.
Escalation		This trigger starts or not to start an in-line event sub-process when the constraint specified is not satisfied. Note that this trigger can only be used with an event-sub-process.
Compensation		This trigger starts an in-line event sub-process when an compensation occurs, which requires undoing some steps. Note that this trigger can only be used with an event-sub-process.
Conditional		This trigger starts the process when a specific condition becomes true.
Signal		This trigger starts the process when a signal broadcasted from another process has arrived. Note that signal is different from message in the sense that it has a specific target for message.
Multiple		This means that there are multiple triggers of the process. Any one of them can cause the process to start.
Parallel Multiple		This means that there are multiple triggers of the process. All of the triggers must be triggered in order to start the process.
Link		This trigger provides a means to connect the end result of one process to the start of another.

Different types of start event trigger

Defining a trigger

To define a trigger on an event, right click on the event and select **Trigger**, then click on the type of trigger from the popup menu.



To define a start event trigger

If you want to edit the properties of the trigger, such as the condition of a conditional trigger, right click on the event and select **Open Specification...** from the popup menu. Then, click on the **...** button next to the drop down menu of **Trigger** to edit its properties in the popup dialog box.

Interrupting or Non-interrupting event sub-process

Start event can be attached to the border of an event sub-process to initiate the sub-process inline. You can define this kind of trigger as either interrupting or non-interrupting, which means to interrupt its containing process or not to interrupt its containing process respectively. To set a trigger to be Interrupting or Non-Interrupting, right click on the event and select/de-select **Triggers > Interrupting** from the popup menu.

Interrupting (left) and Non-Interrupting (right) events

NOTE: Only triggers that can be attached to event sub-process can set as interrupting/non-interrupting. The supported trigger types include: Message, Timer, Escalation, Error, Cancel, Compensation, Conditional, Signal, Multiple, and Parallel Multiple.

Intermediate event

An intermediate event indicates where something happens in between the start and end event of a process. You can use an intermediate event to show where messages are received or sent, show the necessary delay, perform exception handling and show the need of compensation. You can place an intermediate even in two places: Attaching the boundary of task/sub-process, Normal flow (i.e. connected from a flow without attaching to an activity).

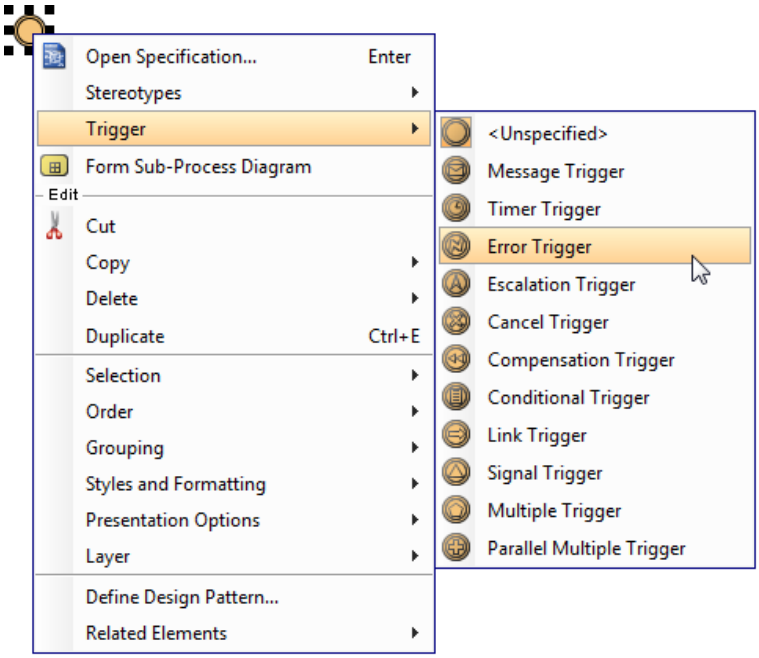
Trigger name	Representation	Description
None		The none intermediate event does not have a defined trigger. It is used to indicate change of state in the process. You can only use a none intermediate event in a normal flow.
Message		This trigger represents either a send or receive of message
Timer		This trigger acts as a delay mechanism on a specific date-time or cycle (e.g. every Friday). You can only use a timer intermediate event in a normal flow.
Error		This trigger reacts to a named error or to any error if no name is specified.
Escalation		The trigger indicates where an escalation is raised. You can only use an escalation intermediate event in a normal flow.
Cancel		This trigger will be fired when a cancel end event is reached within the transaction sub-process. It also shall be triggered if a Transaction Protocol "Cancel" message has been received while the Transaction is being performed.
Compensation		The trigger indicates the need of compensation.
Conditional		The event will be triggered when the condition specified become true.

Link		This trigger is used for linking two sections of a process. You can use it to mode a looping of flow or to avoid having long sequence flow connectors appear on diagram. You can only use a link intermediate event in a normal flow.
Signal		This trigger indicates the sending or receiving of signals, which is for general communication within and across process levels, across pools, and between business process diagrams.
Multiple		This means that there are multiple triggers defined. Any one of them can cause the event to be triggered.
Parallel Multiple		This means that there are multiple triggers defined. All of the triggers must be triggered in order to trigger the multiple event.

Different types of start event trigger

Defining a trigger

To define a trigger on an event, right click on the event and select **Trigger**, then the type of trigger from the popup menu.



To define an intermediate event trigger

If you want to edit the properties of the trigger, such as the condition of a conditional trigger, right click on the event and select **Open Specification...** from the popup menu. Then, click on the **...** button next to the drop down menu of **Trigger** to edit its properties in the popup dialog box.

Throw and catch

You can set an event to be catch or throw. Catch means to react to a trigger, while throw means to create a trigger. To set, right click on an event and select **Trigger**, then either **Catching** or **Throwing** from the popup menu.

A catch event (left) and a throw event (right)

NOTE: The trigger types that can set as throw/catch include: Message, Escalation, Compensation, Link, Signal, and Multiple.

Interrupting or Non-interrupting event

Intermediate event can be attached to the border of an activity. You can set an event to interrupt or not to interrupt the activity to which it is attached. To set a trigger to be Interrupting or Non-Interrupting, right click on the event and select/de-select **Triggers > Interrupting** from the popup menu.

Interrupting (left) and Non-Interrupting (right) events

NOTE: Only triggers that can be attached to event sub-process can set as interrupting/non-interrupting. The supported trigger types include: Message, Timer, Escalation, Conditional, Signal, Multiple, and Parallel Multiple.

End event

As an opposite of start event, end event indicates where a process will end. Since end event is used for terminating a process, it does not have any outgoing sequence flow.

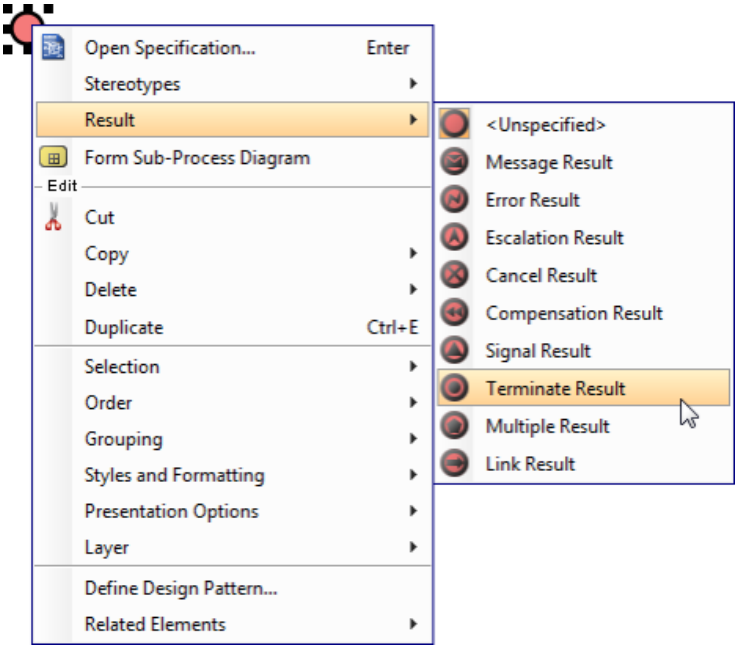
You can define a result for end event, to show what will happen when reaching the end.

Trigger name	Representation	Description
None		The none end event does not have a defined result.
Message		This result ends the process by sending a message to a participant.
Error		This result indicates the generation of a named error when the process ends.
Escalation		This result indicates the trigger of escalation when the process ends.
Cancel		This result indicates that the transaction should be cancelled.
Compensation		This result indicates the need of compensation, which require undoing some steps.
Signal		This result indicates that a signal will be broadcasted when the process ends. Note that signal is different from message, which has a specific target for message.
Terminal		This result indicates that all activities in the process should be immediately ended.
Multiple		This result indicates that there are multiple consequences of ending the process.
Link		This result provide a means to connect the end result of one process to the start of another.

Different types of end event result

Defining a result

To define a result on an event, right click on the event and select **Result**, then the type of result from the popup menu.



To define an end event result

If you want to edit the properties of the result, such as the message produced by a message result, right click on the event and select **Open Specification...** from the popup menu. Then, click on the **...** button next to the drop down menu of **Result** to edit its properties in the popup dialog box.

Related Resources

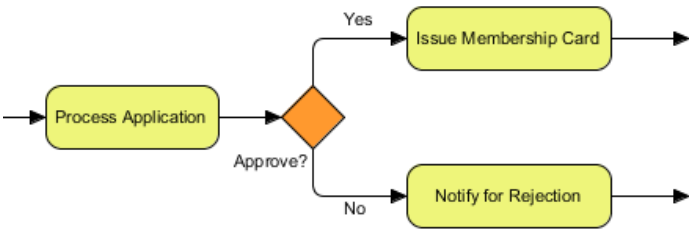
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

- Contact us if you need any help or have any suggestion

Gateway

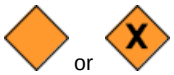
Gateway is a kind of flow objects which is used to direct sequence flows within a process, based on certain condition(s). It acts like a gate that either allows or disallows passage, and possibly control the selection of outgoing flow that pass through the gateway.



A typical use of gateway - for modeling a situation of decision making

Gateway types

There are several kinds of gateway for different kinds of control behavior, such as making decision, branching, merging, forking and joining.

Gateway type	Representation	Description
Exclusive		An exclusive gateway can be used to model alternative paths within a flow. It is where the diversion takes place.
Event-based		An event-based gateway can be used to model the alternative paths that follow the gateway which are based on events that occur instead of the expression of flow.
Inclusive		An inclusive gateway can be used to model alternative and parallel paths within a process. Unlike exclusive gateway, all condition expressions are evaluated. All the outgoing paths that give a positive result of evaluation will be taken.
Complex		A complex gateway can be used to model complex synchronization behavior.
Parallel		A parallel gateway can be used to create and join parallel flows. It creates the paths without checking any conditions.

Different types of gateway

Showing internal indicator "X" for exclusive gateway

You can optionally show the letter "X" inside the shape of exclusive gateway to distinguish it from other gateways (see the table above). To do this, right click on the background of business process diagram and select **Presentation Options**, then select **Gateway Display Options > Data-Based gateway Markers Visible** from the popup menu.

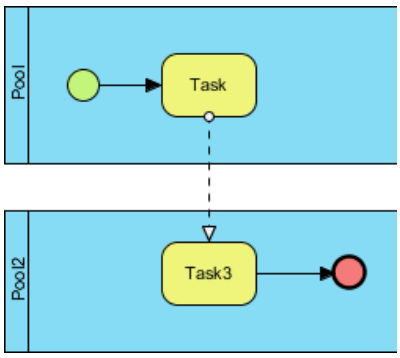
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Sequence and message flow

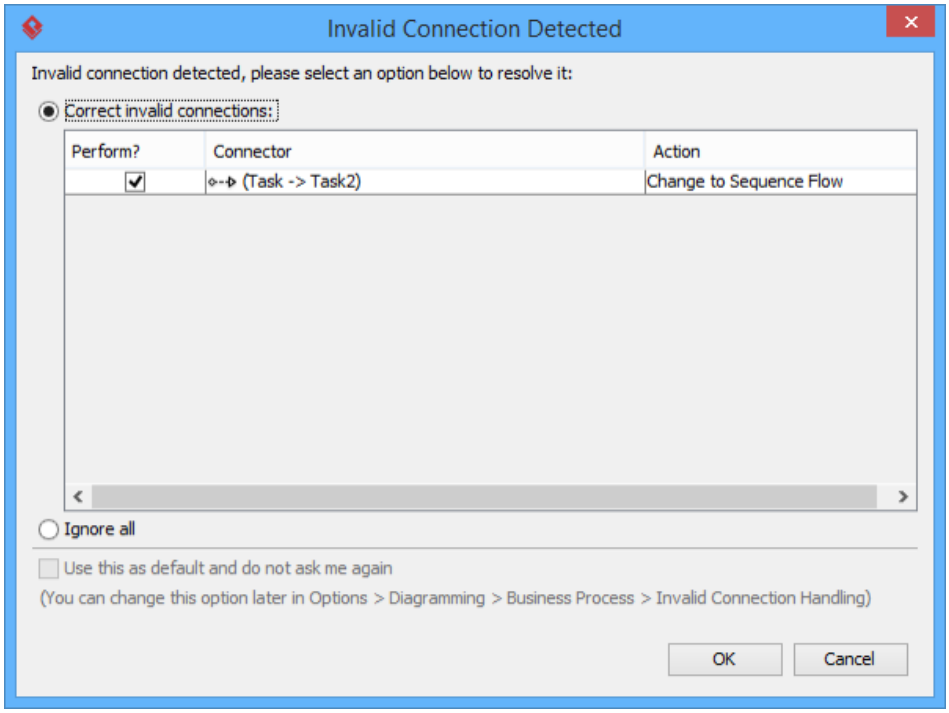
There are two types of connectors for modeling flows in a process - Sequence flow and Message flow. A sequence flow is used to connect flow objects in a process or a choreography to show the flow. Message flow is used to show the flow of messages between separate pools/lanes. You cannot use message flow to connect flow objects within the same participant.



Sequence and message flows can be used to connect flow objects

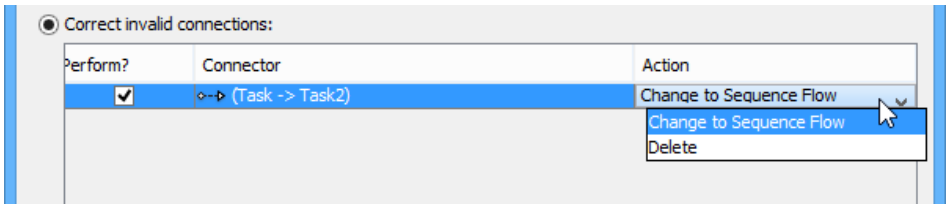
Correcting invalid flow

As mentioned before, you can use sequence flow to connect flow objects within a participant and use message flow to connect flow objects in separate participants. If you attempt to use a type of flow incorrectly, like to connect flow objects within participant with message flow, you will be prompt to correct your flow.



Invalid connector is detected

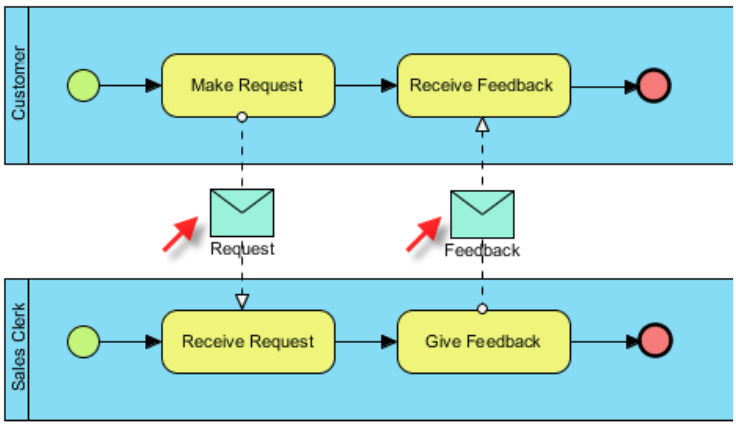
There are several actions you can perform. First, you can correct your invalid flow by changing its type, like to change from message to sequence flow. If the connector should not be there, you may select to delete it. If you really want to keep the invalid connector, choose **Ignore all** at the bottom of dialog box. Click **OK** to confirm.



Possible actions of handling invalid connector - Correct it or delete it

Modeling and visualizing message pass by message flow

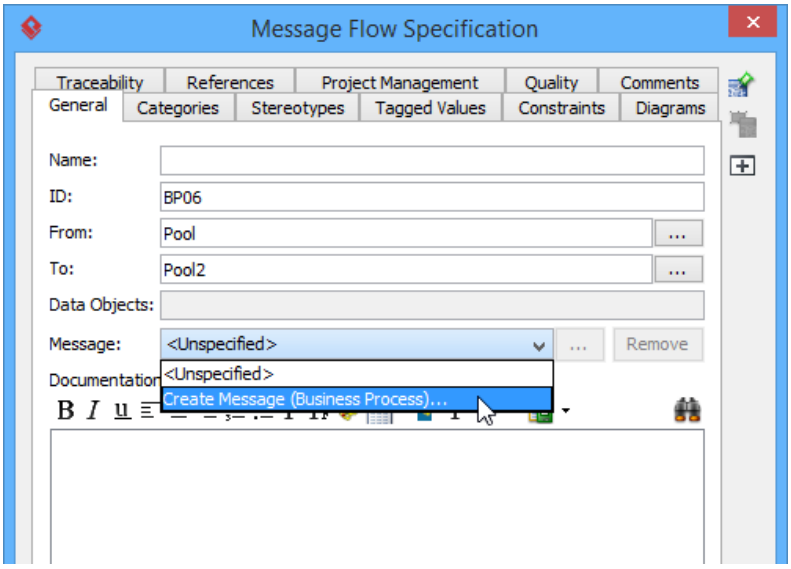
You can define a message that passes by message flow and visualize it.



A sample BPD with messages

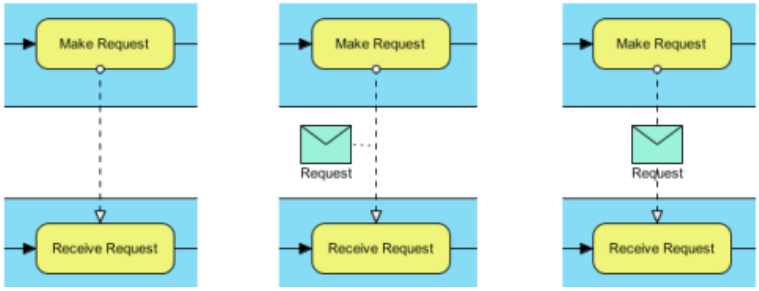
To do this:

1. Right click on the message flow that you want to model its message and select **Open Specification...** from the pop-up menu.
2. Click on the drop down menu of Message and select **Create Message (Business Process)** from the pop-up menu.



Create a message

3. Name the message in the Message specification and click **OK** to confirm.
4. You should see the message added appear in the drop-down menu of **Mesasge**. Click **OK** to confirm the change and go back to diagram.
5. Although the message is defined, you still need to visualize it. To visualize the message, right click on the diagram's background, click **Presentation Options**, select **Message Flow Display Option > Show Message of Message Flow** from the pop-up menu, then select the way of visualizing the message.



Ways of showing message (from left to right) - Do Not Show, Associated with Message Flow, Overlapping Message Flow

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Choreography task and sub-process

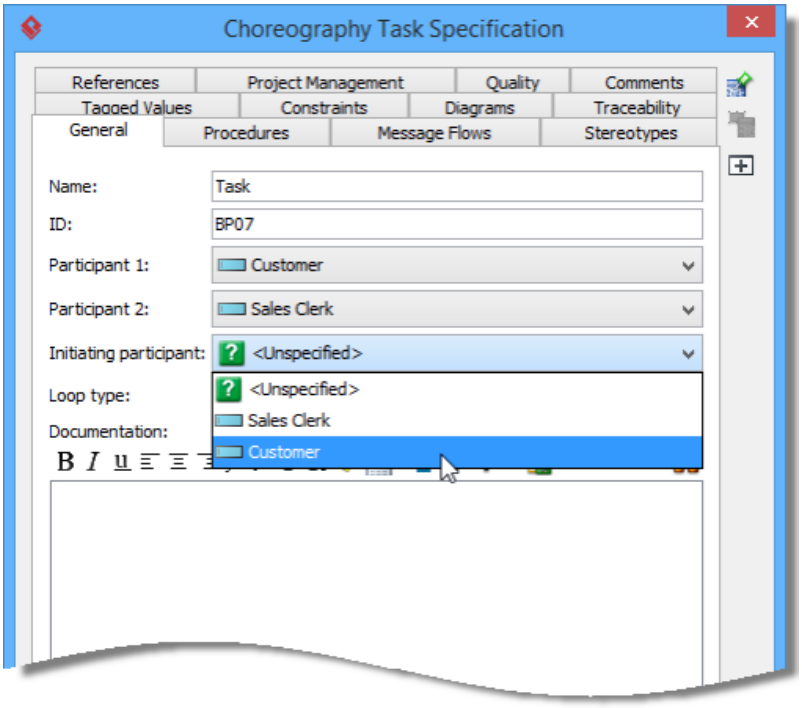
Unlike a standard BPMN process which defines the flow of activities in a process, a choreography is a type of process which defines the sequence of interaction between participant. Choreography does not belong to any pool. It exist outside or in between pools and shows the messages that pass between pools.

Choreography task

A choreography task is an atomic activity which represents an interaction among participants (pools) and consists of one or more messages that exchange between the pools. A choreography shape is formed by multiple parts. We call them bands. The name of choreography task and each of the participants are all displayed in different bands.

Setting participants and initiating participant

1. Right click on the choreography task and select **Open Specification...** from the popup menu.
2. In the specification dialog box, choose the pools for participant 1 and 2.
3. Select the pool which starts the interaction from the drop down menu of **Initiating participant**.



Selecting initiating pool

4. Click **OK** to confirm editing and go back to diagram.

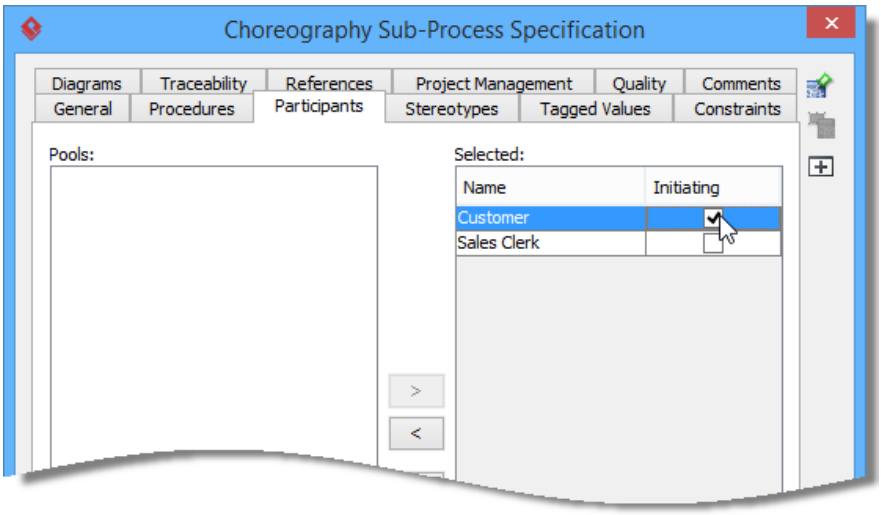
Choreography task

Choreography sub-process

A choreography sub-process is a compound activity in which it has detail that is defined as a flow of other activities.

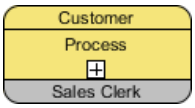
Setting participants and initiating participant

1. Right click on the choreography sub-process and select **Open Specification...** from the popup menu.
2. In the specification dialog box, open the **Participants** tab.
3. Select the pools the choreography sub-process involve and click **>** to assign them.
4. Check the initiating pool.



Select initiating participant

5. Click **OK** to confirm editing and go back to diagram.



Choreography sub-process

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Data object, data input, data output and data store

Data object
You can use data objects to model data within process flow. Typical examples of data object include purchase order, receipt, e-mail, delivery notice, etc.

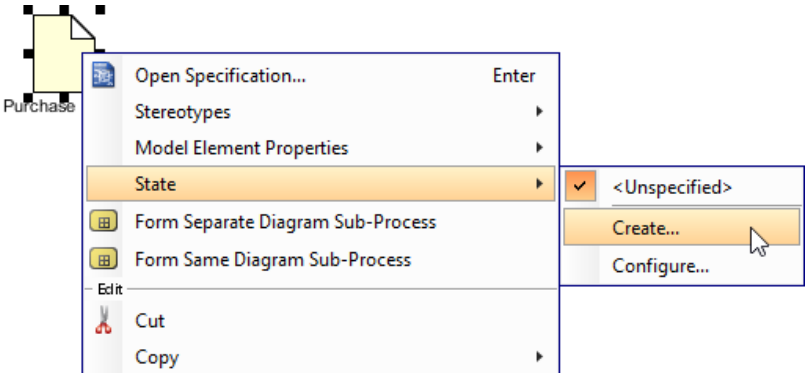
Data input
Data input is a special kind of data used as input of a process. You draw data input in a business process diagram to show the input of data to the top-level process or to show the input of a called process.

Data output
Data output is a special kind of data produced as output of a process. You draw data output in a top-level business process diagram to show the output of the process.

Data store
Data store enables activities to retrieve or updated stored information that will persist.

Defining state
You can optionally record the state of data object. For example, the data object *Order* has states created, submitted and processed. To define state:

1. Right click on data object and select **State > Create...** from the popup menu.



To create a state

2. In the **Create State** dialog box, enter the name of state and click **OK** to confirm.

NOTE: State is a view based option. You may copy a data object, paste as a new view and set the state to the new view. This enables you to show the change of state of a data throughout a process flow.

Related Resources
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Conversation diagram

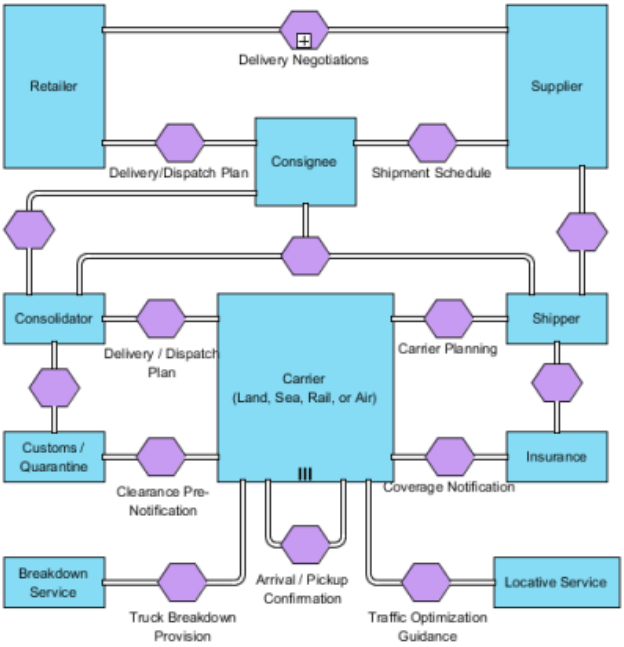
Conversation diagram gives you a high level of understanding to the relationships between pools under the domain being modeled. This chapter teaches you how to create a conversation diagram.

Creating conversation diagram

This page teaches you how to create a conversation diagram through the diagram navigator.

Drawing conversation diagram

Conversation diagram gives you a high level understanding to the relationships between pools under the domain being modeled.



A sample conversation diagram

Creating conversation diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Conversation Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Editing diagrams

In this chapter, you will learn not only how to create and edit diagrams, but also some of the frequently used functions that aid in diagramming, including resource-centric interface, using tagged values to add custom properties and spell checking.

Creating diagrams

You will see how to create a diagram, how to create shapes and draw freehand shapes.

Model element and view

When model element is seen as data, view is a representation of data. This page tells you more about the relationship between model element and view.

Master view and auxiliary view

Multiple views can be added for model elements. Among the views, one master view can be set. The rest are called auxiliary views. This page provides you with more information about views.

Resource centric interface

Resource-centric interface provides you with a set of function icons around shapes, for triggering functions in quick. You will learn how to make use of resource-centric interface to create and connect shapes, and know the differences between types of resource icons.

Tagged values

Tagged values can be used to add domain specific properties to shapes. There are several types of tag, such as text, integer, floating point number, etc. You will see how to create tagged values.

Spell checking

Spell checking is an automatically enabled feature for ensuring the accuracy of written content such as shape name and documentation. You will see how to configure spell checking.

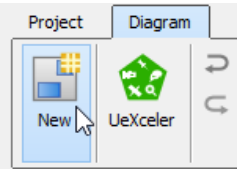
Creating diagrams

You can create diagrams to help visualize what you did, what you are doing and what you need to do on your target system or application. There are different types of diagrams to fulfill different needs, such as UML diagrams, requirements capturing, database modeling, business process modeling and others.

Creating a diagram

Through the toolbar

1. Select **Diagram > New** from the toolbar.



Create diagram through toolbar

2. In the **New Diagram** Window, select the type of diagram to create.
3. Click **Next**.
4. Enter the diagram name and description.
5. You can set the **Location**, which is the model for storing the diagram. Note that if a diagram is opened in background and if that diagram is being stored in a model, we will automatically select that model as **Location**.
6. Click **OK**.

Through shortcut key

You can press **Ctrl-N** to create a diagram.

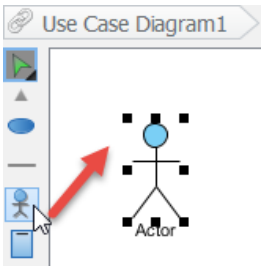
Drawing shapes

After creating a new diagram, diagram elements can be created as well through the diagram toolbar. In this section, these techniques of drawing shapes will be explicated:

- Creating Shapes
- Creating Connectors
- Creating Self-Connection

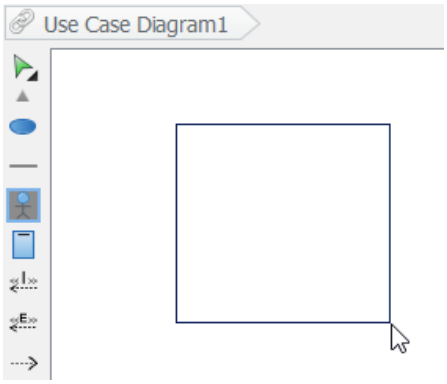
Creating shapes

To create a shape, click a diagram element from the diagram toolbar and click it on the diagram pane for creating. The generated element will appear with the default size.



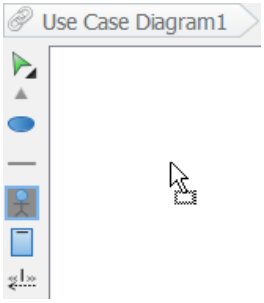
Click an actor on the diagram pane

For defining a specific shape size, drag a specific boundary with the mouse after clicking a diagram element from the diagram toolbar.



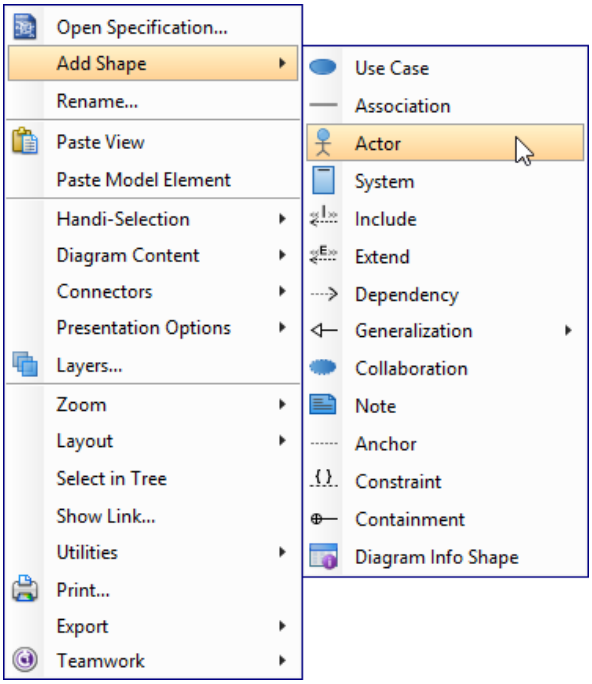
Drag a specific boundary with the mouse

Alternatively, a diagram element can be created by dragging the diagram element and then dropping it on the diagram pane.



Drag and drop to create a shape

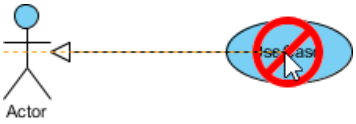
In addition, you can add a shape through the pop-up menu of diagram. Right click on the diagram background, select **Add Shape** and then a specific shape from the pop-up menu.



Create a shape through the pop-up menu of diagram

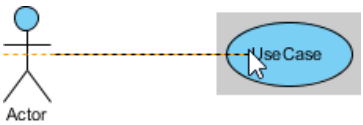
Creating connectors

To create a connector, select the desired connector from the diagram toolbar, drag the connector from the source shape to the destination shape. Since Visual Paradigm provides a continuous UML syntax checking, if you create an invalid connection, a stop sign will be pop-out. For instance, you are not allowed to connect an actor and a use case with a generalization relationship.



An invalid connection is created

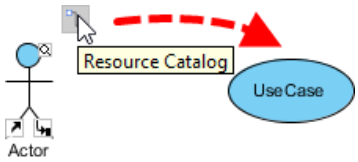
If the connection is valid, a gray rectangle surrounding the destination shape can be seen.



A valid connection is created

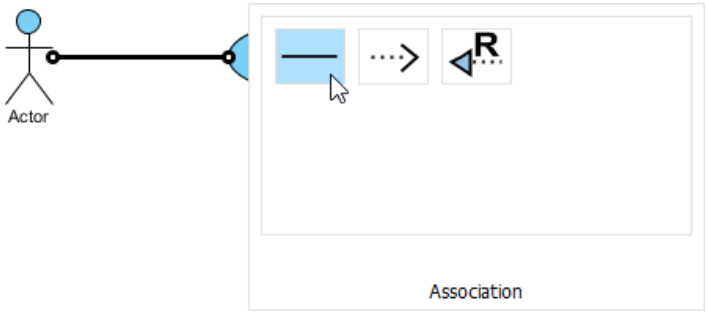
Moreover, connectors can be created through resource icons:

1. Move the mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.



Resource Catalog

3. Release the mouse button on the target shape..
4. Select the desired the connector type from Resource Catalog.

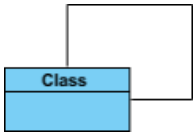


Select Association to be created

If you release the mouse on an empty space, a new shape can be created with a connector.

Creating self-connection

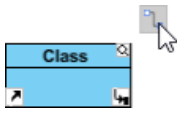
Some shapes can make a connection for itself, for example, **Self-Association** of a **Class** in class diagram and **Self-Link** of an **Object** in communication Diagram.



Caption Here

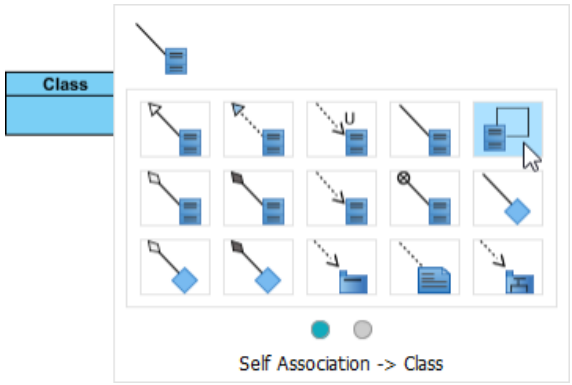
To create a self-connection:

1. Move the mouse pointer to the shape.
2. Click on the **Resource Catalog** button.



Caption Here

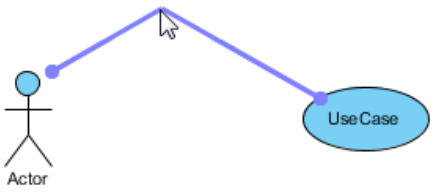
3. Select the self connector to be created.



Caption Here

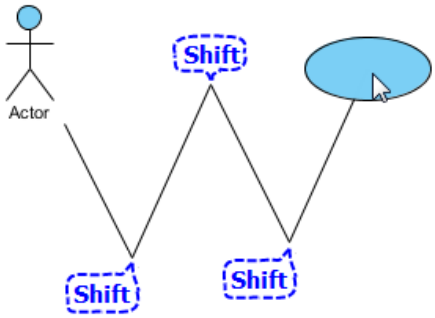
Creating turning point on connector

A turning point is a point on a connector where a bending takes place. To create a turning point on an existing connector, press on the connector and drag to bend the connector.



Create turning point on existing connector

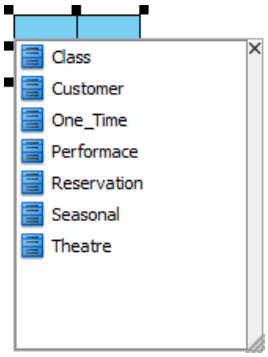
You can also create a turning point when creating a shape through Resource Catalog. When dragging out **Resource Catalog** button, press the **Shift** button at the place you want to create the turning point.



Create turning points when creating shape by dragging Resource Catalog

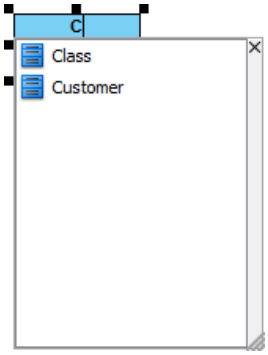
Name completion

The model name completion feature enables quick creation of multiple views for the same model element. When you create a model element, press **Ctrl-Space** to reveal the name completion list.



Model name completion

Type text to filter the elements in the list.



List filtered by typed text

Press up or down key to select the model element to use. When selected, press **Enter** to confirm.

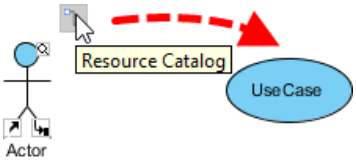
Resource-centric interface

Visual Paradigm is the first vendor to introduce the resource centric diagramming interface. The resource centric interface greatly improves the efficiency of modeling. You don't have to traverse between the toolbar and the diagram to create diagram elements, make connections and modify the diagrams. The resource centric interface can make sure the modeler is able to create a diagram with correct syntax more quickly.

There are tree types of resource icons:

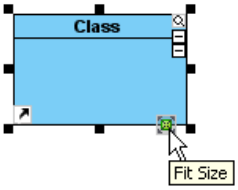
- Connection Resource
- Manipulation Resource

Connection resource (Resource Catalog)
It is designed for creating elements and making connections.



To associate an actor with use case with Resource Catalog

Manipulation resource
You can use Manipulation Resource to modify properties or appearance of elements. For example, you can show or hide compartments, add references, add sub-diagram and fit size.

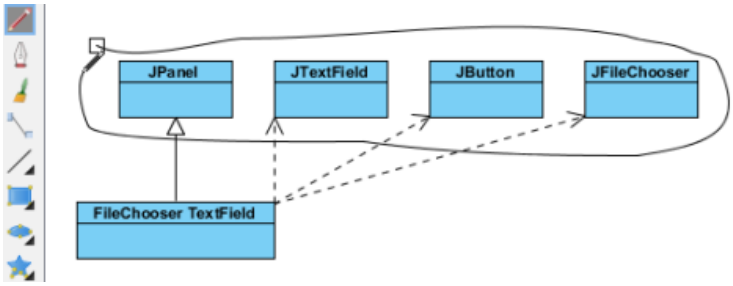


Fit size through manipulation resource

Drawing freehand shapes

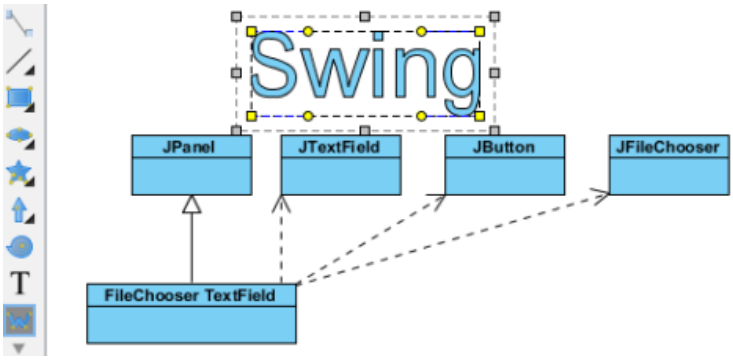
Freehand shape is a kind of general graphic shapes. Pen shapes, pencil shapes, and some regular shapes like circle, rectangle and arrow are available. Freehand shape can be used for annotating diagram. For example, you can use freehand shapes to emphasize some shapes.

A specific shape can be highlighted with a pencil shape.



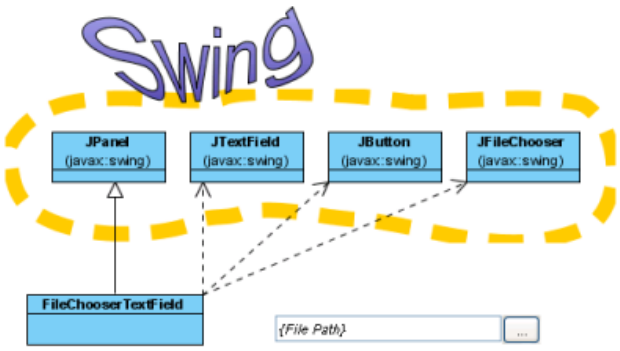
Pencil

An outstanding text can be shown with word art.



Word Art

A freehand shape style can be formatted in order to address your information.



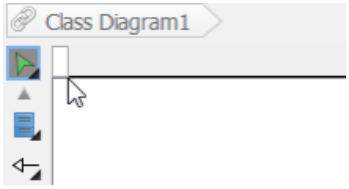
Styled freehand shapes

Changing package header

You can specify the parent package of any diagram through Package Header.

Package header when diagram created

When a diagram is created, the package header will be unfolded as it allows you to specify the parent package of the diagram. Specify the package by entering the fully qualifier name of the package.



Specify parent package in package header

After entering the name of parent package, you will find that the diagram name is the same as the name of parent package.

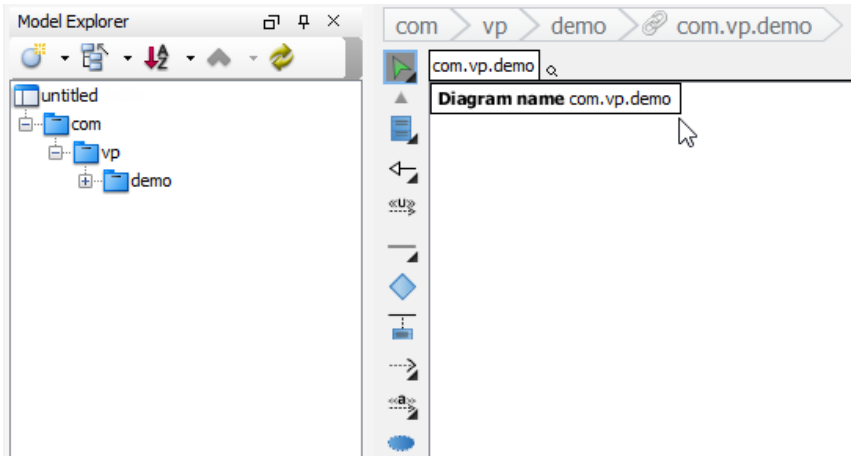
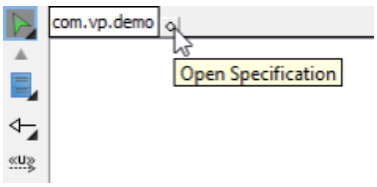


Diagram name will be same as fully qualify of paret package

The diagram name can be renamed. However, the name of parent package won't be changed to follow the diagram name.

You can open specification of parent package by pressing the **Open Specification** button on the right-hand side of the parent package name.



Open Specification

You can rename the parent package of the diagram by double clicking on it.

Double click on the parent package

A new package will be created if you enter a completely new package name. If the previous parent package does not contain elements, it will be deleted. That means the description (or other properties) of previous parent package will be lost.

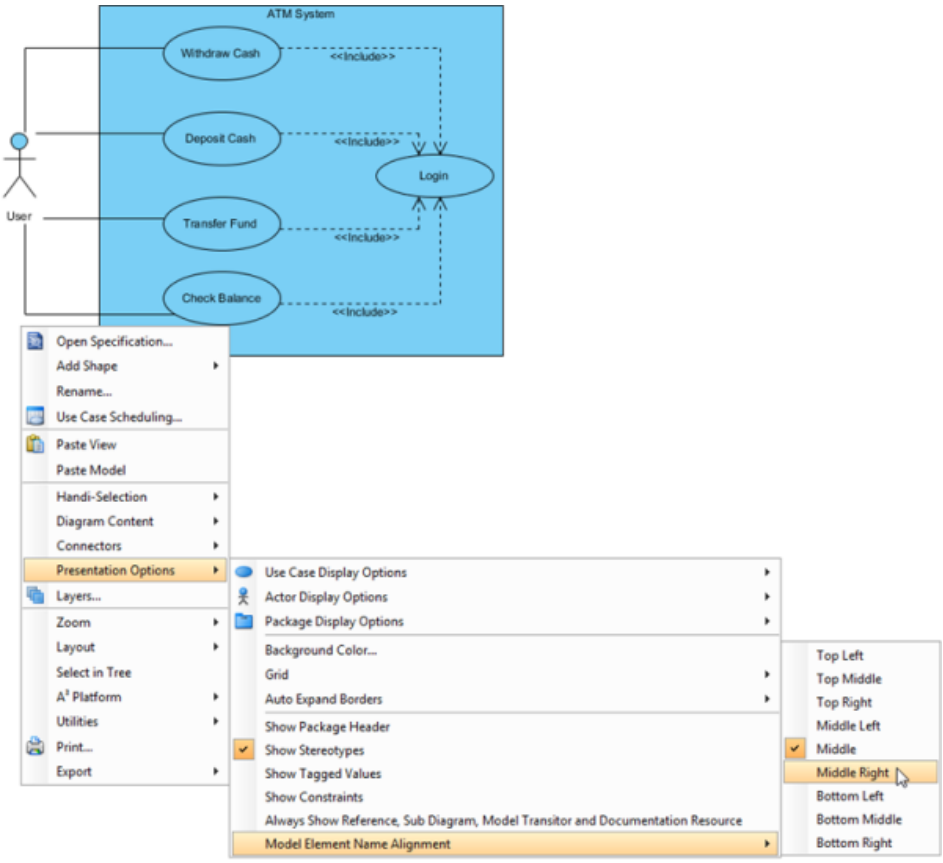
A package header can be either shown or hidden through the pop-up menu of diagram. Right click on the diagram background and select **Presentation Options > Show Package Header**from the pop-up menu.

Justifying shape's name

In Visual Paradigm, a shape's name is aligned with center horizontally or top and middle vertically, depending on the characteristic of shapes. However, the shape's name can be realigned. Even if a language, such as Modern Hebrew, that is written from the right to the left can be displayed on a shape clearly.

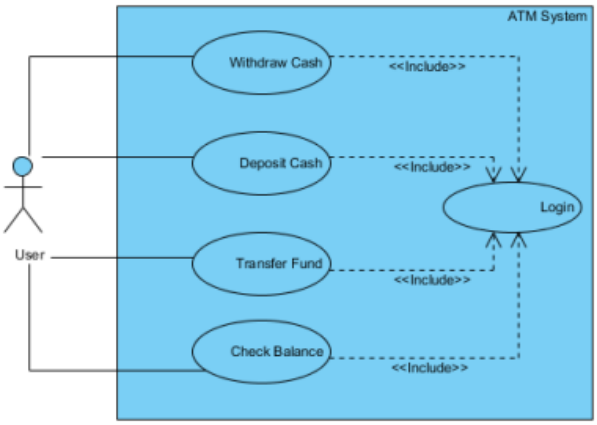
Adjusting shape name's position

1. Right click on the diagram background, select **Presentation Options > Model Element Name Alignment** and then select a specific alignment option from the pop-up menu.



Select an alignment option from the pop-up menu

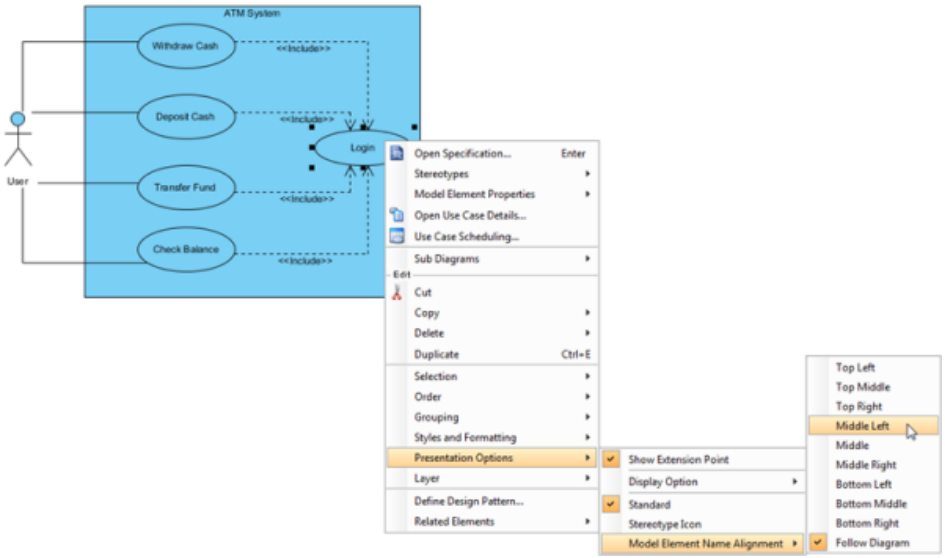
2. As a result, all shapes' name in the whole diagram will turn into the alignment option you set previously.



All shapes' names turn into middle right

Apart from the whole diagram setting, a specific shape can also be set:

1. Right click on a shape, select **Presentation Options > Model Element Name Alignment** and then select a specific alignment option from the pop-up menu.

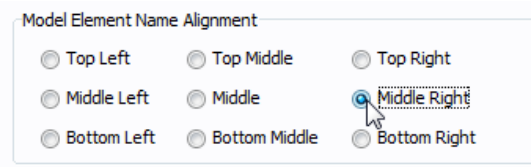


Select an alignment option from the pop-up menu

2. As a result, the specific shape will turn into the alignment option you set previously.

In addition to the current diagram, future diagrams can also be set:

1. Select **Windows > Project Options...** from the toolbar.
2. In the **Options** dialog box, select the **Diagramming** category, check an option out of **Model Element Name Alignment** section under the **Appearance** tab.



*Check an alignment option in the **Options** dialog box*

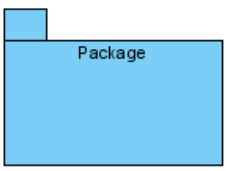
Exceptions

Although most shapes' name can be justified, some are exceptional. Two main kinds of shapes that their names cannot be justified are introduced as follows:

On one hand, shapes neither with floating name label (freely movable) nor with a label outside the shapes can be justified. Actor, Initial Pseudo Node and BP Start Events are examples of this kind of shape.

Floating name label

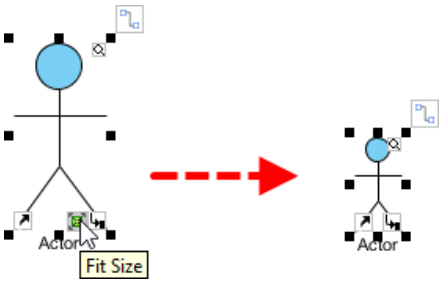
On the other hand, the names of container shapes are not available for positioning. Since their “bodies” are used for containing other shapes, thereby, they have a limited scope of displaying names. Package, State and System are example of this kind of shape.



Container shape

Enabling and disabling minimum shape size

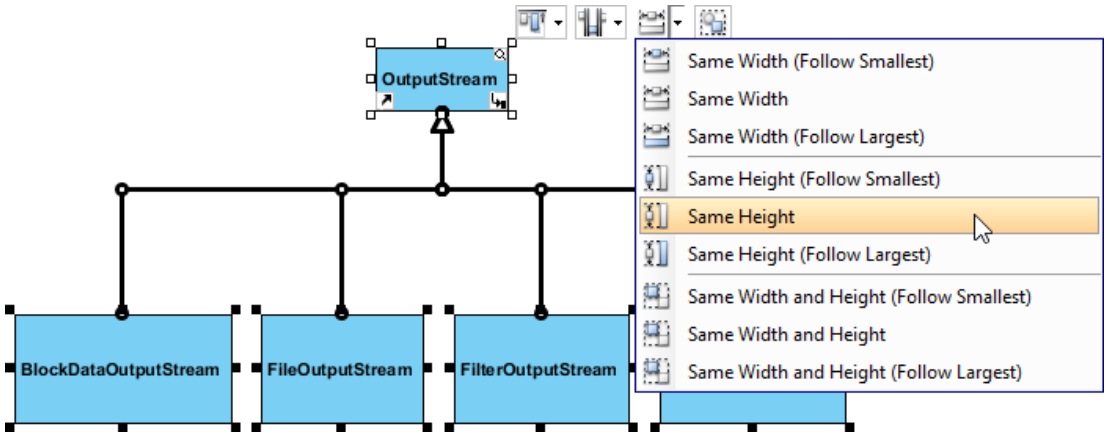
Since all shapes have their own default minimum size, users are not allowed to resize them to smaller than the minimum size. The default setting helps to ensure those compact shapes are clear enough to be seen on a diagram under normal circumstance. The minimum size of a shape can be determined by pressing its fit size button.



The minimum size of a shape can be determined by pressing its fit size button

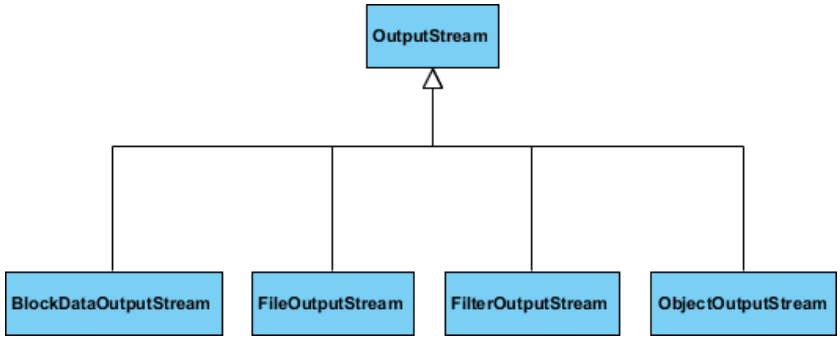
Now, it is possible to disable such setting, so that shapes can be resized to extremely small in size, despite their minimum size:

1. To disable the function of the minimum size checking, select **Windows > Project Options...** from the toolbar.
2. In the **Project Options** dialog box, select the **Diagramming** category and uncheck **Enable minimum shape size when resizing shape** under **Appearance** tab. Click **OK** to confirm.
3. After that, you can resize a shape to the size as small as you want.
4. Furthermore, select other shapes and select an option from the drop-down menu of resource icon **Same Width**.



Make all shapes in same height

5. As a result, other shapes will turn into the same size as the shape you have done previously.



All shapes are turned into the same small height

Undo and redo

During the process of editing a diagram, you may make some careless mistakes, such as accidentally deleting a shape. You can use the **Undo** function to cancel the previous action. On the contrary, you may re-perform the action through the **Redo** function. Note that the undo/redo feature in Visual Paradigm is diagram based.

Undo

You can roll back undesirable changes by performing **Undo**. Undo function can be executed in the following three ways:

- Select **Diagram > Undo** from the toolbar.
- Press **Ctrl-Z**.

Redo

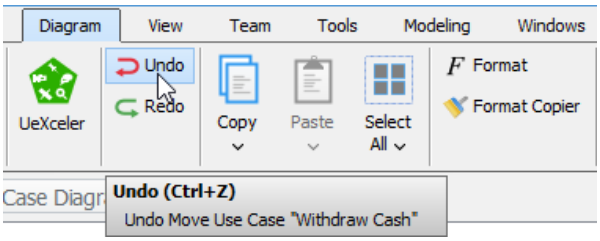
This feature is to re-perform actions that have just been undone. Redo function can be executed in the following three ways:

- Select **Diagram > Redo** from the toolbar.With
- Press **Ctrl-Y**.

Showing name for undo and redo action

It's hard for us to remember all actions we have done previously. By Visual Paradigm, we can recall the actions we have done before.

You can find an action name of undo/ redo on toolbar button's tooltip. Move the mouse over the **Undo** or **Redo** button and then a tooltip with **Undo**/ **Redo** action name will appear.



Toolbar button's tooltip shows undo/redo action

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Closing a diagram in Visual Paradigm

You can close an active diagram by pressing the hotfix as listed below.

Operating System	Hotkey
Windows	Ctrl-F4
Linux	Ctrl-W
Mac OS X	Command +W

Hotkeys for closing a diagram

Once a diagram is closed, the previously opened and non-closed diagram will be shown. If such a diagram does not exist, the **Project Browser** will be opened.

Related Resources

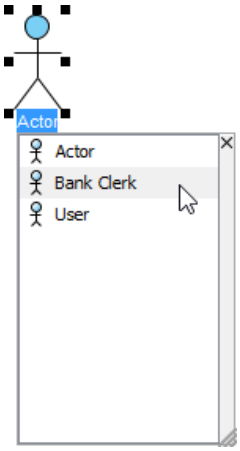
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model element and view

A model element is an elementary component of a model. It will be created when you create a shape on a diagram . An example of model element is actor.

You can visualize an existing model element or we sometimes call it to 'reuse' a model element with the model completion feature. When you create a shape, press **Ctrl-Space** to popup a list of existing elements. Click on the desired element to reuse it. We call the visualized form of model element a view or a shape, depending on whether we want to emphasis the differences against model element or we want to focus on diagramming operations.



Creating a view from an existing actor

When developing context-based diagrams, you will reuse a model element in different diagrams, resulted in the creation of multiple views. Each model element can associate with zero to multiple views. When you make specification-level change, such as changing of name on any view, the change will be applied to all views.

Showing the other views of a view

If you want to open another view of an existing view (i.e. shape), right click on the view in the diagram and select **Related Elements > Show Other Views...** from the popup menu. Then, select the view to be opened in the **Show View** window and click **Go to View** to open the diagram.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

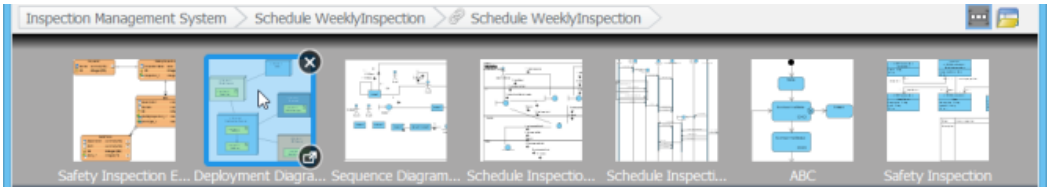
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening diagram

This article shows you how to open a diagram in Visual Paradigm. You can switch to a recently opened diagram, or open one from the **Project Browser**.

Switching to another diagram

The diagram switch provides you quick access to diagrams opened within the active session. To switch to another diagram, click the **Switch Diagram** button on the right hand side of the navigation bar. Then, double click on the thumbnail of the desired diagram to open it.



Switch to another diagram

You can also press **Ctrl-Tab** in an active diagram to toggle this pane.

Opening Project Browser

The **Project Browser** incorporates several 'views' of the project, such as the **Diagrams** view, **Model Structure** view, **My Recent** and **Team Recent**. These views provide different ways for realizing a project's model hierarchy as well as to find out the desired diagrams. The **Project Browser** supports view-based finding and project-level searching, which makes searching of project data much easier.

You can open the **Project Browser** by clicking on **Open Project Browser** on the right hand side of the navigation bar. Click here if you want to know more about the Project Browser.

Open Project Browser

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Master view and auxiliary view

When your project is simple, you are able to express all of the design ideas with just a few diagrams. The diagrams are simple and self-explanatory. Each of them represents a distinct design idea and there is no overlapping between diagrams.
When you are dealing with a complex project, you may need to draw multiple diagrams to represent different contexts. You need to borrow shapes from a diagram to make them appear in other diagrams (i.e. contexts). In fact, this is extremely common when modeling with class diagram and business process diagram. Take UML class diagram as an example, there may be a domain diagram that presents all the entity classes and another diagram that presents the associations and dependencies between a specific controller class and its related entity classes. So in this case, both diagrams contain the same set of entity classes.

Instead of re-creating those classes again and again in different diagrams, Visual Paradigm allows you to "re-use" them. Through simple copy and paste (**Ctrl-C** and **Ctrl-V**), you can easily copy a shape from one diagram to another. Each shape is formally known as a "view". So with this, you can create multiple views for a model element in representing different contexts. Changes made on a shape are all synchronized to other instances that appear in other diagrams without extra effort. This is great but there is a drawback though.

A master view is simply the specific view of model element that decides the placement of that model element within a model hierarchy. When you draw a shape on a diagram without reusing an existing one, the created view will be treated as the master. Subsequent views are all known as auxiliary views. When you attempt to move a master view to another parent shape, you are updating the real model structure as well (as reflected in **Model Structure View**). However, when you move any auxiliary view to a different parent shape, there will be no change at all on the model structure.

Selecting a master view

A model element can have multiple views. Yet, it can only have one master view. You can change the master view of a model element. By doing so, the original master view will become auxiliary and the assignment of parent element will be updated immediately based on the new master view.

1. Open the **Show Views** window first. In **Model Structure View**, you can open it by right clicking on the target model element and selecting **Show View...** from the popup menu. In diagram, you can open it by right clicking on the target view and selecting **Related Elements > Show Other Views...** from the popup menu.
2. In the window you can see a list of views of the selected model element/view. Click **Set Master View...**at bottom left corner.
3. This shows the **Set Master View** window. You can select on the left hand side the **Model Explorer** or a specific diagram to be the master view of selected element. To select **Model Explorer** means that any re-positioning made in views in any diagram will not affect the model hierarchy. Click **OK** to confirm your selection.

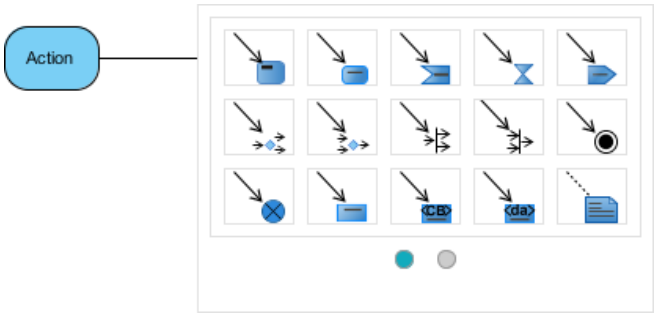
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Resource centric interface

Visual Paradigm is the first vendor to introduce the resource centric diagramming interface. The resource centric interface greatly improves the efficiency of modeling. It can make sure the modeler is able to create a diagram with correct syntax more quickly. The utmost importance thing is that you don't have to go back and forth between the toolbar and the diagrams for creating diagram elements, making connections and modifying the diagrams.



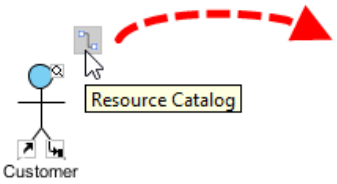
Resource Catalog

Creating shapes through Resource Catalog

Resource Catalog is an important part of the entire Resource Centric Interface. Resource Catalog presents a single icon that appear top right of each shape. You can create shape by dragging out that icon and choosing the shape to create from the popup catalog panel. Resource Catalog also records the frequency at which the icons are being used. You can always access the frequently used icons quickly, which makes editing much more efficient.

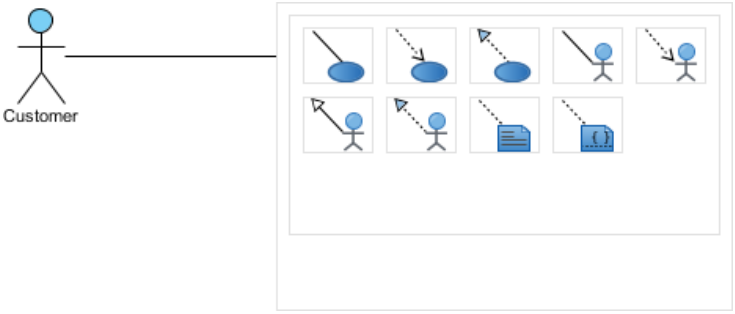
To create a 'related shape' from an existing shape:

1. Move the mouse pointer over the source shape.
2. Press on the **Resource Catalog** button at top right and drag it out.



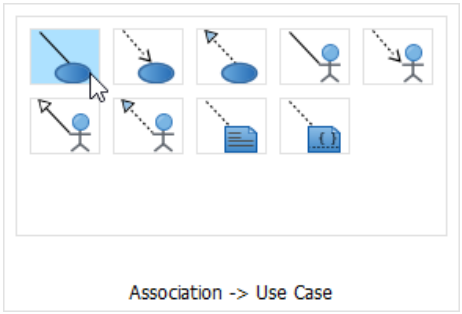
Dragging Resource Catalog button

3. Release the mouse button at the position where you want the target shape to be created. This popup the Resource Catalog.



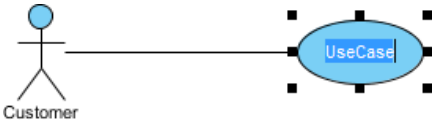
Resource Catalog

4. Shape types that can be created from the chosen source shape are listed in Resource Catalog. By moving your mouse pointer over each resource, you can view the name of shape and connector type at the bottom of the catalog panel.



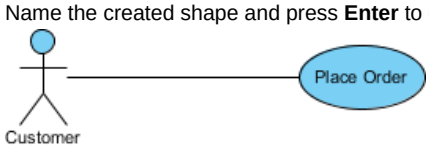
To create a use case

5. Click on the appropriate shape to be created. As a result, another shape is created and linked with the source shape.



Use case created from actor

6. Name the created shape and press **Enter** to confirm editing.

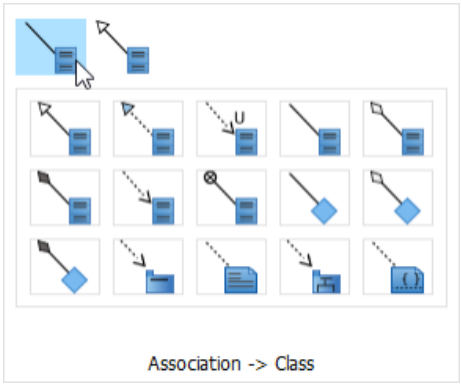


Use case named

Frequently created shape

Resource Catalog also records the frequency at which the resources are being used. You can always access the frequently used icons quickly, which makes editing much more efficient.

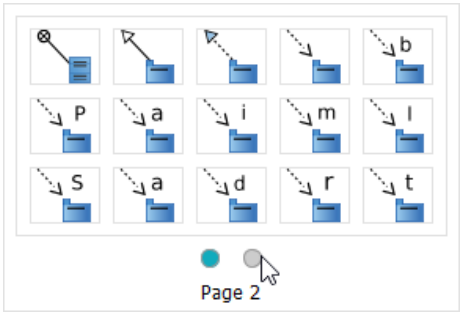
Take the following image as an example. You can see an additional row of resources are displayed at the top of Resource Catalog, with two resources in it (in this example). Those resources are actually resources that you have used before. They are ordered based on the number of time being used. In this example, **Association -> Class** is the mostly used resource, so come first. **Generalization -> Class** resource is less popular than the **Association** resource, so come second. And same as how you can use other resources, you can click on a frequently used resource to use it now.



To access frequently used resource

Using slider

When there are too many resources available to a chosen shape, you may have to find the resource that you want to access on another 'page'. You can do this by clicking on the buttons on the slider, at the bottom of Resource Catalog.

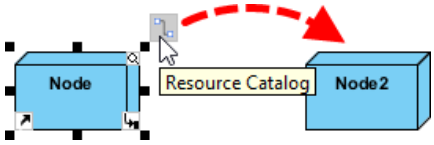


To open the second page of Resource Catalog

Connecting shapes through Resource Catalog

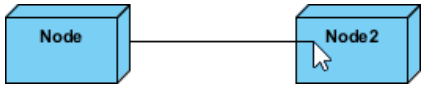
For relating two shapes together, you have to link one shape to another. With Resource Catalog, you can not only link them up, but also select an appropriate relationship for them.

1. Move the mouse pointer over the source shape.
2. Press on the **Resource Catalog** button at top right and drag it out.



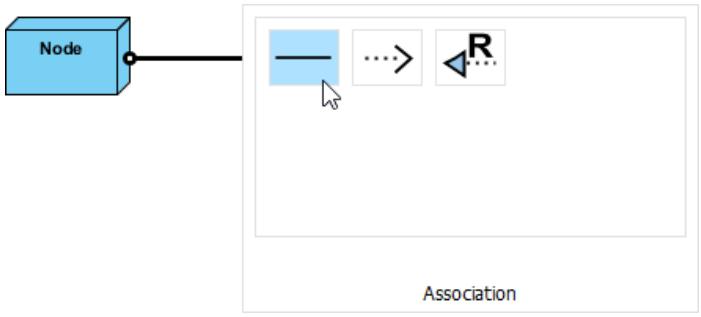
Dragging Resource Catalog button

3. Release the mouse button on the target shape.



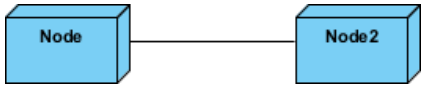
Dragging to another node

4. The connector types supported by the source and target shapes are listed in Resource Catalog. By moving your mouse pointer over each resource, you can view the name of shape and connector type at the bottom of the catalog panel.



To create an association between two nodes

5. Click on the type of connector to be created. As a result, two shapes are linked together.

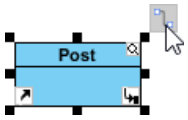


Association created between nodes

Using quick connect

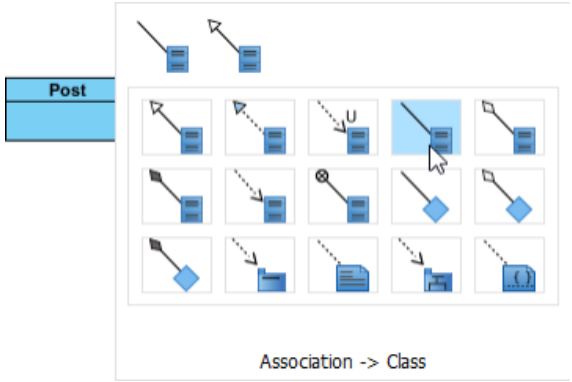
With quick connect, you are able to search the preferred shape and connect to it on the diagram accurately and quickly.

1. Move the mouse pointer over the source shape.
2. Click on the **Resource Catalog** button at top right.



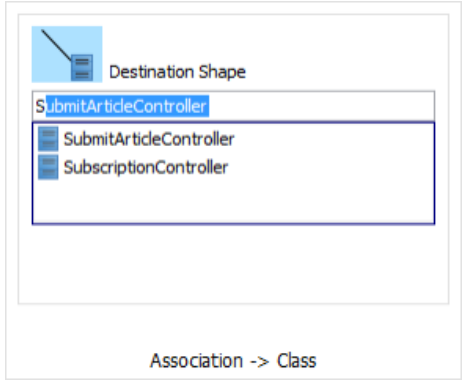
Clicking on the Resource Catalog button

3. From Resource Catalog, click on the appropriate resource. For example, click on **Association -> Class** if you want to connect to an existing class on the same diagram with an association.



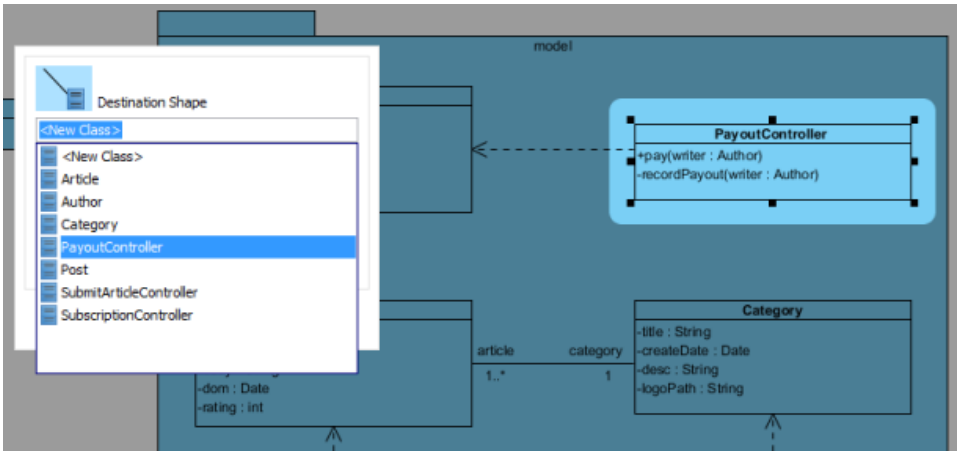
To create an association with another class

4. You are prompted to enter the name of the shape to connect to. You can either create a new shape or select an existing shape by typing its name in the text field of **Quick Connect** dialog box. To select an existing shape, enter its full name directly or just type the first letter of its name. As a result, a list of shapes which match with the word(s) you typed will be displayed. You can also press the **Down** key to select it from the popup list.



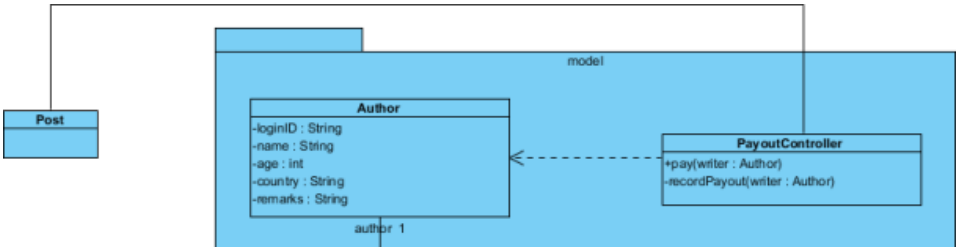
Classes are filtered according to the entered text

5. The shape will be spotlighted on the diagram following the active selection in the drop-down list.



Class is spotlighted

6. Press **Enter** to confirm your selection. The two shapes will be linked automatically.



Association created between classes

> **NOTE:** You should click the resource icon in accordance with the existing shapes on your diagram. If the shape does not exist on the diagram, you won't be able to find it in the drop-down list.

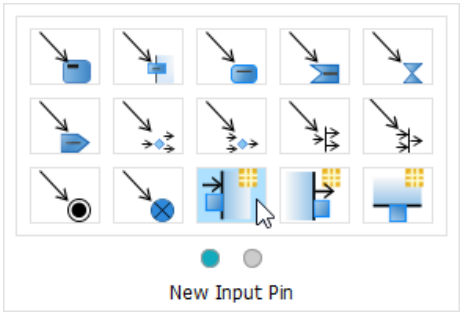
Creating part shape

Resource Catalog also supports the creation of part shapes, such as a pin attached to UML action, a class put inside a composite class, an activity parameter node attached to activity, a port attached to a component etc. To create a part shape:

1. Move the mouse pointer over the source shape.
2. Click on the **Resource Catalog** button at top right.

Clicking on Resource Catalog button

3. From Resource Catalog, click on the part shape to creat.



Create input pin

As a result, two shapes are linked together.

Input pin created

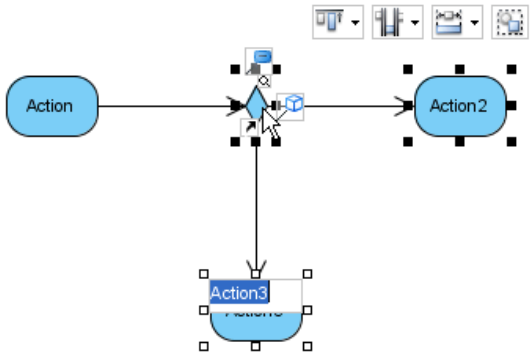
Creating structure

Resource Catalog supports some model elements (e.g. Action in Activity Diagram) with creating a structure of elements. Instead of creating shapes one by one, branch resource icon helps tro speed up the shape creation process; you thereby can create several shapes simultaneously.

1. Move the mouse on a shape and select a branch resource icon out of resource centric interface.

*Select **Create Branch with Decision Node** on resource centric interface*

2. Drag the resource icon and release it until reaching to your preferred place.
3. As a result, two shapes and a decision node are created and connected.



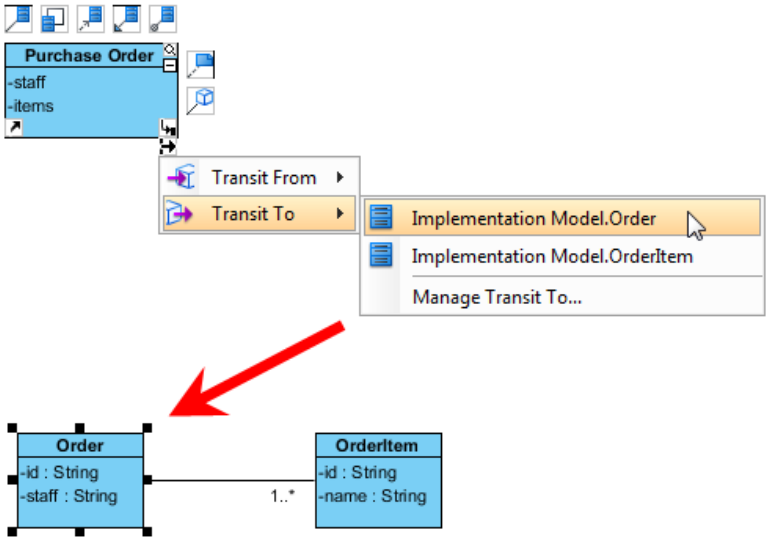
Create Decision Node and Actions

> **NOTE:** There is an orange asterisk on the branch resource icon.

Managing transition of shapes

Model transitor enables you to trace between model elements across different phases of development. Once a transition is created between two shapes, you can navigate between them through the resource centric interface.

Move the mouse on a shape and select **Model Transitor> Transit To** and then select the shape's name you would like to transit to/from from the pop-up menu. As a result, the vision will be diverted to the selected shape after transition is selected. Thereafter, you can transit to/ from between these two shapes.



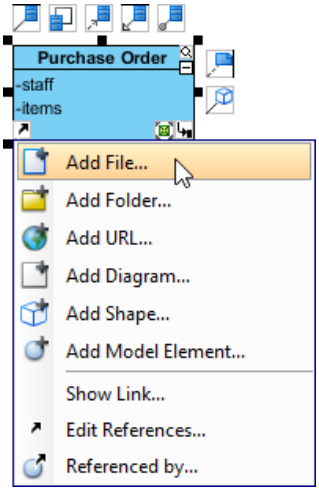
Transit from one shape to another

> **NOTE:** Transition does not only apply on two shapes on the same diagram, but also two shapes in different diagrams.

Adding and opening reference

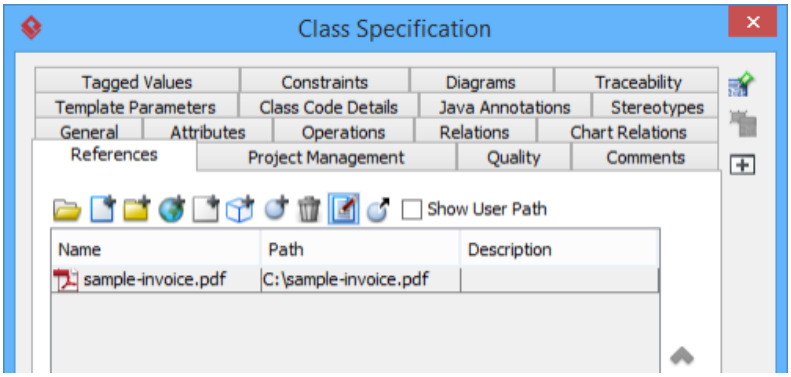
For providing extra information to shapes, you can insert both internal and external resources, such as a shape, a diagram, a file, a URL through the resource centric interface. After editing references, they can be opened throughout the resource centric interface.

1. Move the mouse on a shape that you would like to insert references for, press resource icon **Resources** at the bottom left corner and select a reference option from the pop-up menu.



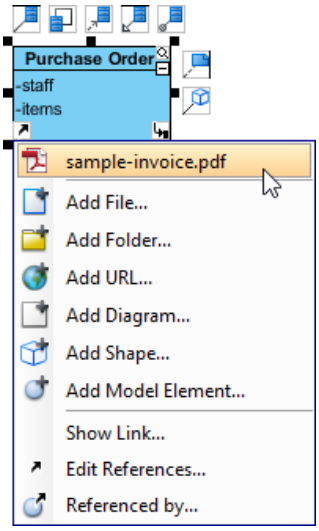
*Select **Add File...** from the pop-up menu*

2. Insert the corresponding resource in **References** tab of your shape's specification window.



*Insert a file in **Class Specification***

3. After adding reference, you can click resource icon **Resources** again and the reference you have just created will be revealed on the pop-up menu.

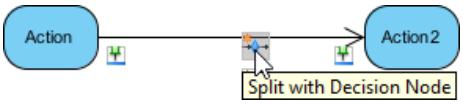


The newly created reference

Splitting connection by shape

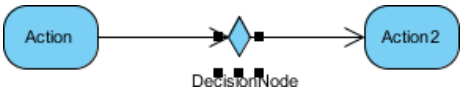
Resource centric supports some connectors (e.g. Control Flow in Activity Diagram) with splitting the connector by adding another shape.

1. Move the mouse on the connector between two shapes and select a split resource icon out of resource centric interface.



Select a split resource icon

2. As a result, an extra shape is inserted between two existing shapes.



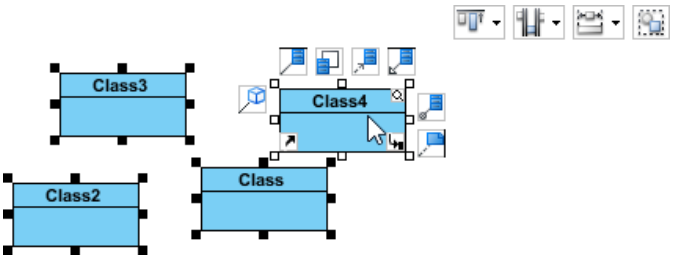
*Control Flow is split by **Decision Node***

> **NOTE:** There is an orange asterisk on the split resource icon.

Group selection resource

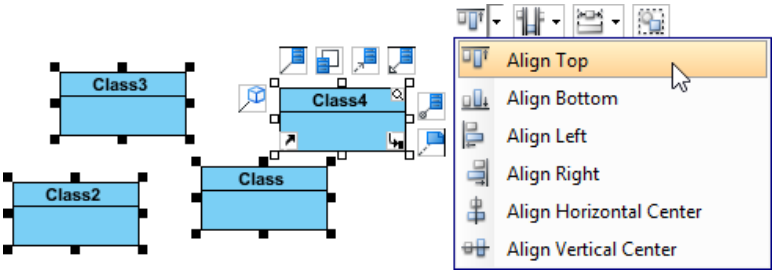
When a great amount of shapes are displayed on the diagram disorderly, resource centric interface can support alignment and grouping after selecting several shapes.

1. Select several shapes on diagram.
2. Move the mouse on the last selected shapes and group selection resources will be shown instantly.



Group Selection Resources are shown

3. Press the reversed triangle of **Align Top** and select an alignment option from the drop-down menu.



*Select **Align Top** from the drop-down menu*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Tagged values

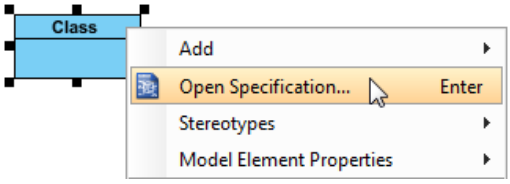
Sometimes, you may find the standard modeling notations are not enough in supporting your domain specific needs. For example, if you are working on a project that relies heavily on the use of third-party libraries, you might want to specify the API source of those components modeled in UML component diagram. And in business process modeling, you might want to record the locations where the tasks are performed. Usually, these kind of specific needs cannot be directly supported by the standard notations.

Fortunately, OMG Unified Modeling Language (UML) introduces stereotype and tagged values, which provide designers with ideal solutions to these situations. Stereotype and tagged values are kind of extensibility mechanisms that allow designers to extend the vocabulary of UML in order to create new model elements. With the use of stereotype and tagged values, designers can introduce model elements with domain/problem specific properties.

Starting from UML 2.0, tagged values are considered to be attributes of stereotype. Yet, Visual Paradigm supports to use tagged values independently, that is, to add tagged values directly to model elements as custom properties instead of prior attachment to any stereotype. Besides, Visual Paradigm allows you to take the advantage of stereotypes and tagged values in not just UML but all the modeling notations like BPMN and even in ERD, DFD, etc. Simply put with Visual Paradigm, you can add custom properties to any model elements by creating tagged values.

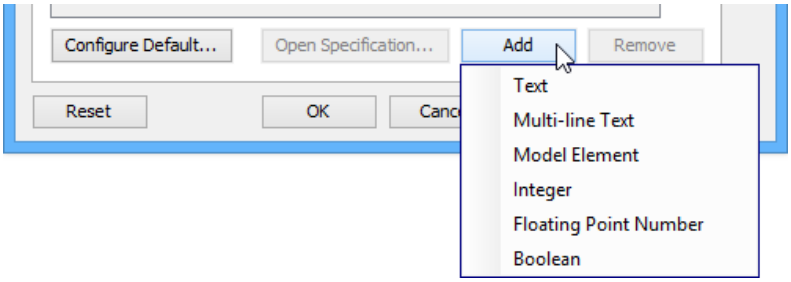
Adding user-defined tags

1. Right click on the selected shape and select **Open Specification...** from the pop-up menu.



*Right click to select **Open Specification...***

2. In **Specification** dialog box, select **Tagged Values** tab and click **Add** button to select the tag type.



Add an option of value type

Different types of tagged values

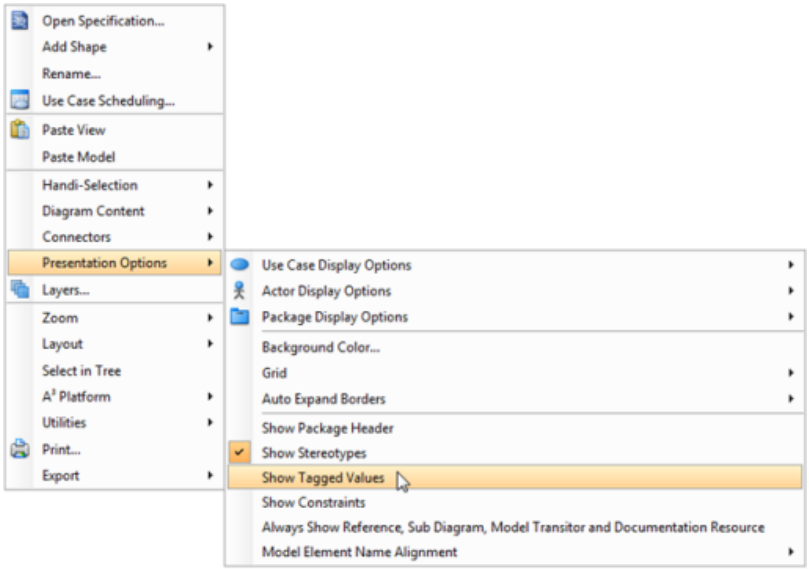
Type	Description
Text	String-based value. To edit, double click on the **Value** cell and enter the value.
Multi-line Text	String-based value, in multiple lines. To edit, click on **...** in the **Value** cell, and then enter the value in the popup window.
Model Element	Reference(s) of model element. To edit, click the reverse triangle button of **Value**, and then select the model element in the popup window.
Integer	Numerical value. To edit, double click on the **Value** cell and enter the value.
Floating Point Number	Value with decimal places. To edit, double click on the **Value** cell and enter the value.
HTML	Value in rich-text. **HTML** is a hidden tagged value. To reveal this option, select **Windows > Project Options** from the toolbar. When the **Project Options** window, select **Diagramming > Environment** tab and check **Support HTML tagged value**. To edit an HTML value, click on **...** in the **Value** cell, and then enter the value in the popup window.
Boolean	Either true or false. To edit, click on the **Value** cell, and then select **True**/ **False**.

Different types of tagged value

NOTE: The value of stereotype can be edited, however, its name and its type cannot be edited as they are defined in stereotype.

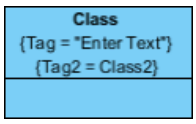
Visualizing tagged values on diagram

Right click on the diagram background and select **Presentation Options > Show Tagged Values** from the pop-up menu.



Show tagged values

If it is defined, the tagged values will be seen within the shape(s) on the diagram.



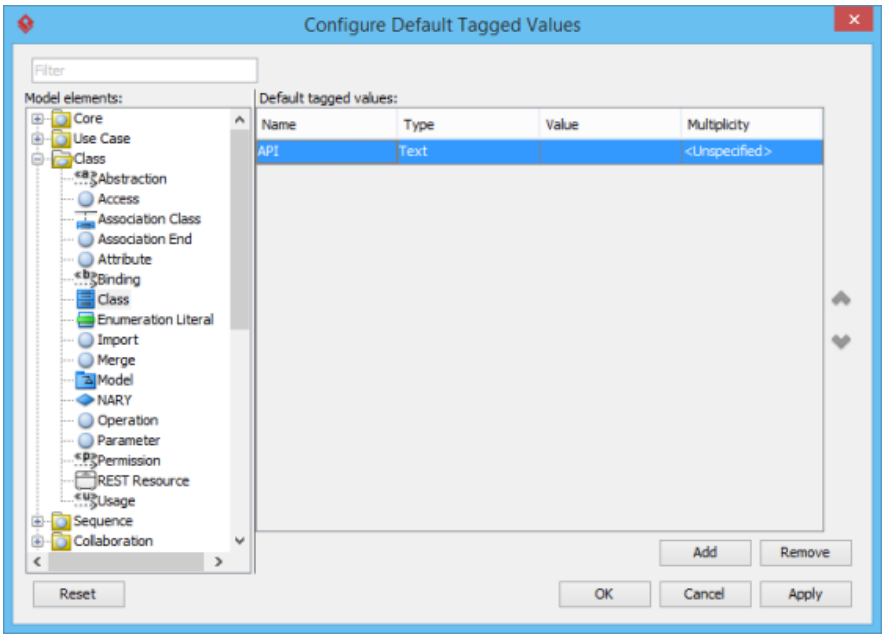
Tagged values are shown

Creating default tagged values

Besides adding tagged values to model elements, you can make a tag as the default of certain model element type. This is an ideal solution for creating custom properties for model elements. For instance, you can create a default tag "API" to UML class. Then, when you create a class, the tag "API" will be added to the class automatically. You can then specify the tagged values of "API" of that class.

To create default tagged values:

1. Select **Configuration > Configure Default Tagged Values** under the **Windows** tab of the toolbar.
2. Select the type of model element to be configureon on the left hand side.
3. Click **Add** on the right hand side to select the tag type.
4. Input the tag name and default value, if any.
5. Click **OK** to confirm the changes.



Added a tag "API" to class

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Spell checking

You will never find it hard to avoid making mistakes in your diagram after using spell checking. It can help you to correct both typing mistakes and spelling mistakes. However, it is slightly different from other spelling and grammar checking tools you have used before as it doesn't check your whole diagram automatically but underlines wrong words with a curve line.

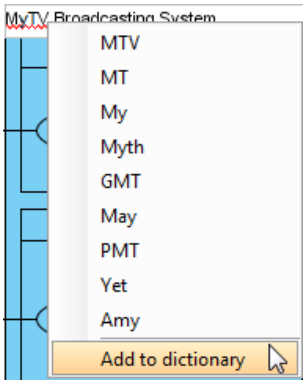
Correcting a word

If you type an incorrect word mistakenly, you can:

- Either re-type the word correctly or
- Right click on the wrong word and then select one out of the suggest words.

Adding a new word to dictionary

Sometimes, the dictionary cannot recognize the word you type and it doesn't always mean you type an incorrect word . It may be a rare word or a new word that you create, for example, your company's name. You can simply right click on the wrong word and select **Add to dictionary** to add a new word to dictionary. You type this word again next time, it won't be marked as a wrong word.

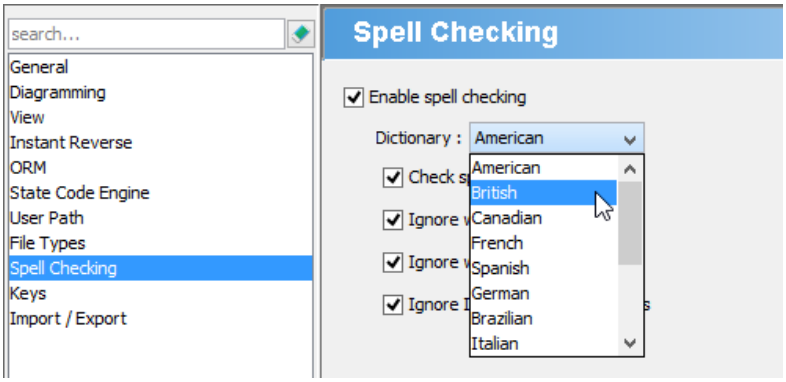


Adding a new word to dictionary

Changing the language of dictionary

The dictionary usually defaults as American English for spell checking. If you want to change the language of dictionary, you just need to:

1. Select **Windows > Application Options** from the toolbar. In the **Application Options** window, click **Spell Checking**.
2. For example, you can change from American to British.



Changing the language of dictionary

NOTE: There are a few more languages in dictionary that can be used, such as French and German.

More options of spell checking

There are a few more options of spell checking, for example **Ignore words with numbers** and **Ignore Internet and file address**. Check the item you want to be included in spell checking while uncheck the item you don't want to be included.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram specification window

You can customize the settings of each diagram you created in diagram specification window. Those settings include: general diagram information, grid setting, references, project management and comments. Diagram specification window is similar to model element specification window in which you can also enter general information, add/ remove references, enter project management and add/ remove comments.

Opening diagram specification window

To open diagram specification window of a diagram, right click on the background of the diagram and select **Open Specification...** from the pop-up menu. Alternatively, right click on the thumbnail of diagram in the **Diagrams** view of the **Project Browser** and select **Open ${diagram-name} Specification...** from the popup menu.

The overview of diagram specification window

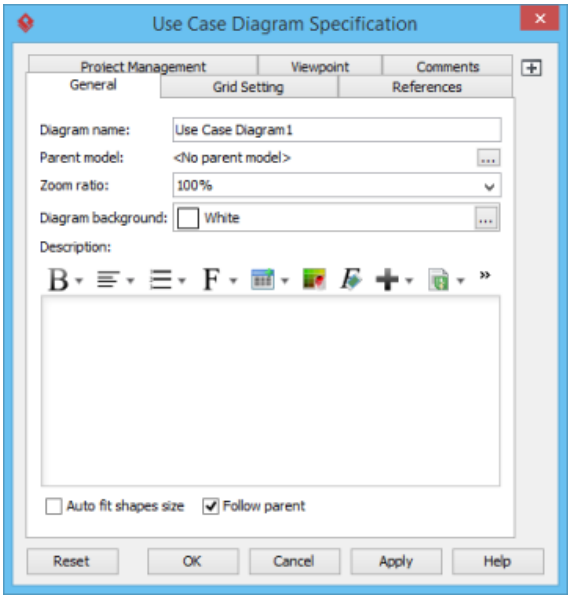


Diagram specification

A diagram specification window consists of 5 tabs: **General**, **Grid Setting**, **References**, **Project Management** and **Comments** respectively. You may notice that an icon named **Maximum** located at the right hand side of window. We'll introduce it after giving you a brief on 5 tabs.

General tab

You can specify diagram name, select zoom ratio and diagram background color and enter description for the diagram. Furthermore, you can record voice description for the diagram. You can set all shapes including existing and future shapes to fit the size automatically.

Grid Setting tab

You can visualize the grid of diagram background by checking Grid visible. Moreover, you can set the size and select the color for the grid of diagram background.

References tab

You can add/ remove internal and external references for the diagram. Those references refer to file(s), folder(s), URL, diagram(s), shape(s), model element(s) and A³ resource(s).

Project Management tab

You can specify diagram process, priority, status, etc for project details.

Comments tab

You can add/ remove comment(s) for the diagram.

Maximize

Click ⊞ to enlarge the specification window to the maximum screen size. Click it again to reduce it to the default size.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

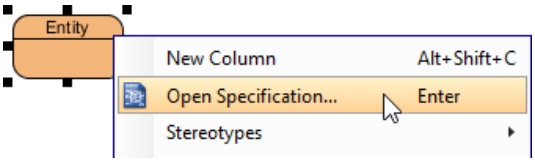
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model element specification window

In Visual Paradigm , you can open a model element specification window to view and edit the element's details. The options, such as references, project management and comments are categorized into tabs in the specification window. Furthermore, three buttons attached on the right-hand side of the window are **Pin**, **Auto open specification when select** and **Maximum**. Model element specification window is similar to diagram specification window in which you can also enter general information, add/ remove references, enter project management and add/ remove comments.

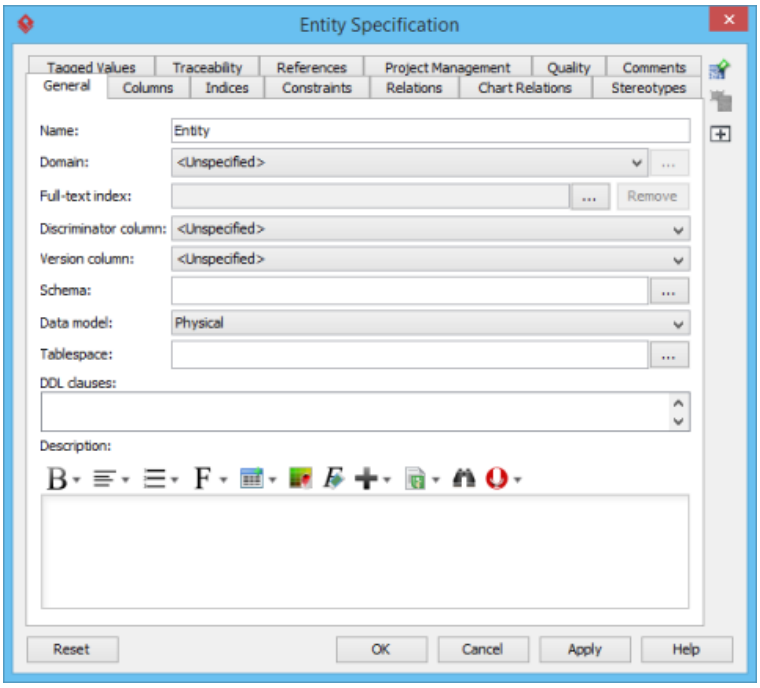
Opening a model element's specification window

Right click on the target model element which you want to view or edit its details and select **Open Specification...** from the pop-up menu.



Open specification window

As a result, the model element specification window pops out.

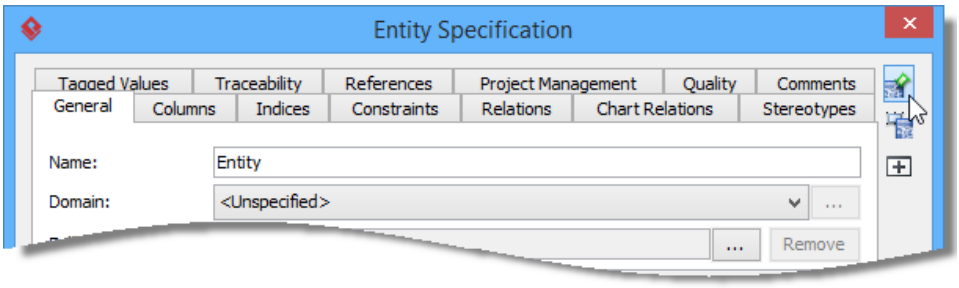


The Entity Specification window

The overview of three buttons on model element specification window

Pin

To pin the specification window, press **Pin** button.



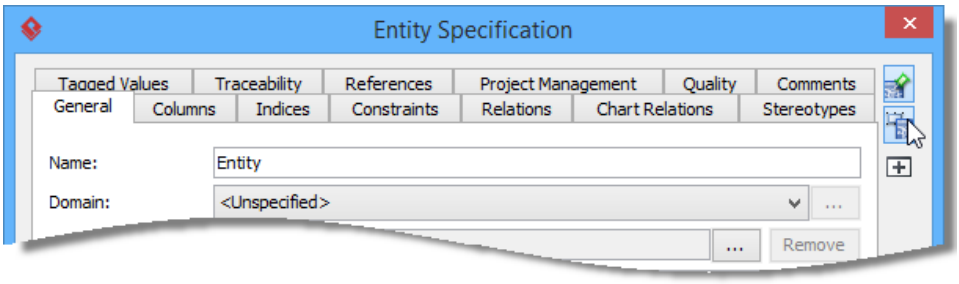
*Press **Pin** button*

NOTE: This button works in combination with **Auto open specification when select**.

Auto open specification when select

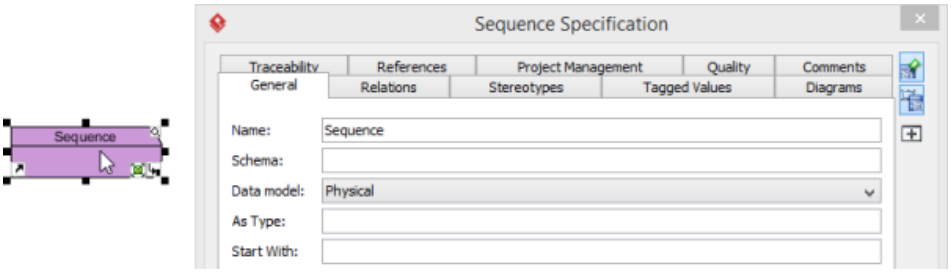
After you've pinned the specification window, press this button. After all, you don't have to close the current window in order to open another model element specification box.

1. Press **Auto open specification when select** button.



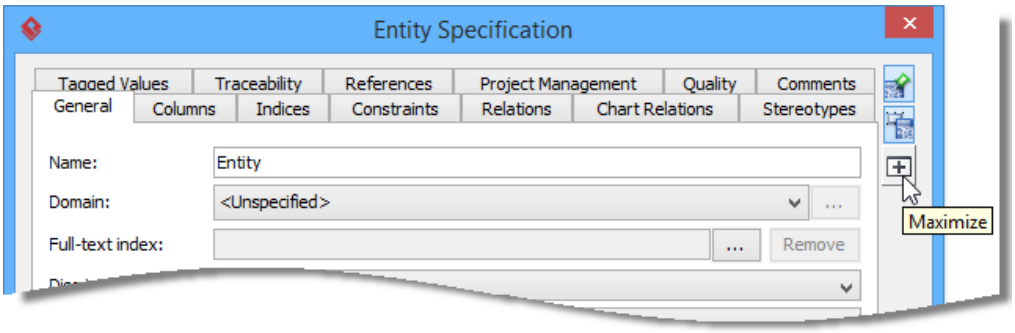
Pressing the Auto open specification when select button

2. Select another model to view its specification window by clicking on the target model.



Selecting Sequence

Maximum
Press this button to enlarge the specification window to the maximum screen size.



*Click **Maximum** button*

After that, click it again to reduce it to the default size.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

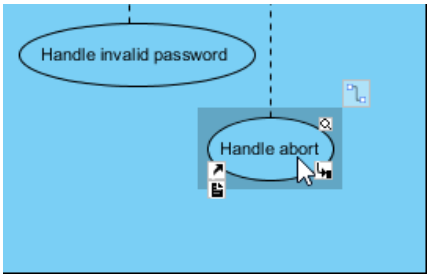
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Showing visible description to diagram

You can always view the description of model element in Description Pane or inside its specification window. But sometimes, you may want to have the description visible in diagram. You can do this by making use of the Note shape.

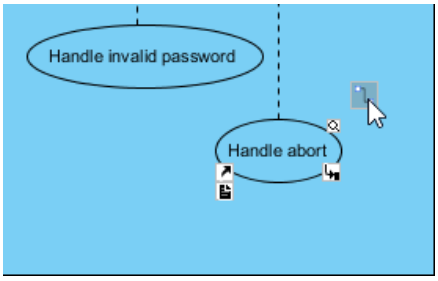
To show visible description to diagram:

1. Move your mouse pointer over the shape from which you want to show its description.



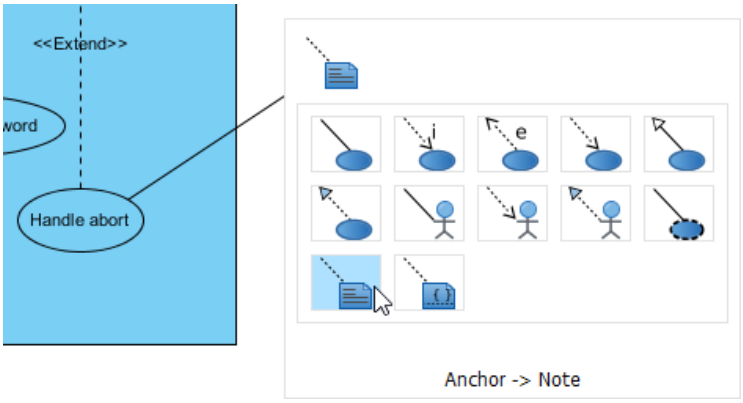
Moving mouse pointer over a shape

2. Press on the **Resource Catalog** button and drag it out to the position to show the description.



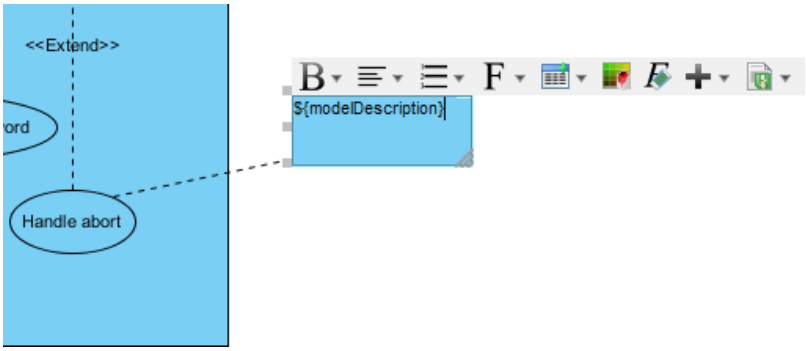
Creating a shape from resource catalog

3. Release the mouse button and select **Anchor -> Note**.



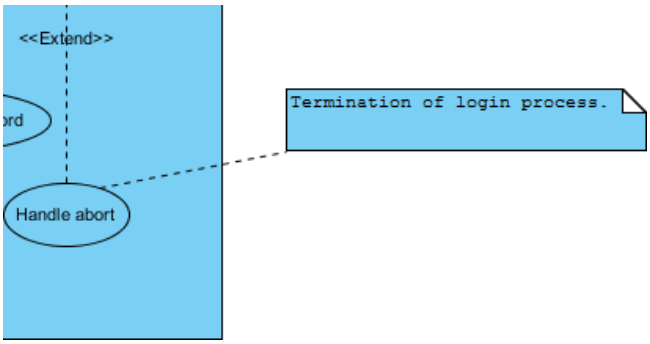
Creating note

4. Enter the **${modelDescription}** as note content. Do not enter anything else otherwise it won't work.



Entering ${modelDescription}

5. Click on the diagram to confirm editing.



Description is visible

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Project management properties

Project management properties are a set pre-defined properties, made for recording additional management-level information for all kinds of project data (e.g. diagrams, model elements, diagram elements).

Using project management properties

You will see a list of project management properties with their description.

Configuring project management properties look-ups

Add available value selection for project management properties, and set defaults.

Using project management properties

Project management properties are a set pre-defined properties, made for recording additional management-level information for all kinds of project data (e.g. diagrams, model elements, diagram elements). Here are all the project management properties you can find:

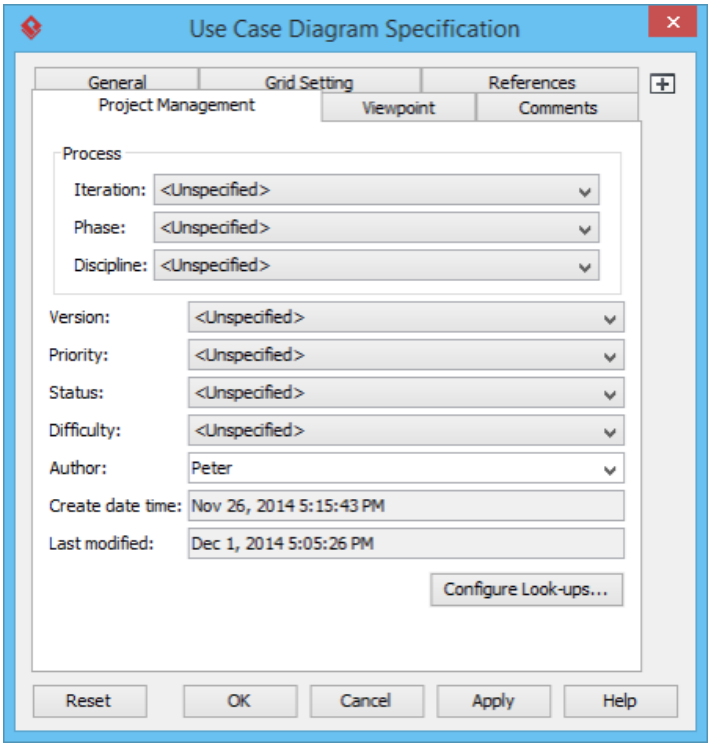
Property name	Description
Process	Specify the part of the process where the editing element is involved. Three sub-properties for further specification. They include: Iteraction, Phase and Discipline.
Version	The stage of the editing element. For example, you may have two class diagrams for modeling a system from design and implementation angles respectively. The two diagrams will have version 1.0 and 2.0 to show the progress of work.
Priority	The importance of editing element.
Status	The status of editing element. It is particularly useful for use case and BPMN activity shapes for setting their status such as Proposed, Planned, Tested, etc.
Difficulty	How difficult it is to complete the goal as modeled by the edting element.
Author	The person who created the editing element. This is particularly useful in a team working environment. Once the author is known you can contact that person in case you have questions about a part of a model.
Create date time	The date and time when the editing element was created. This property is a read-only property that is filled automatically to all elements when creation.
Last modified	The date and time when the editing element was modified. This property is a read-only property that is filled automatically to all elements when the project file is being saved .

A list of project management properties

Further to recording project management properties, you can print those properties in document, too.

Editing project management properties

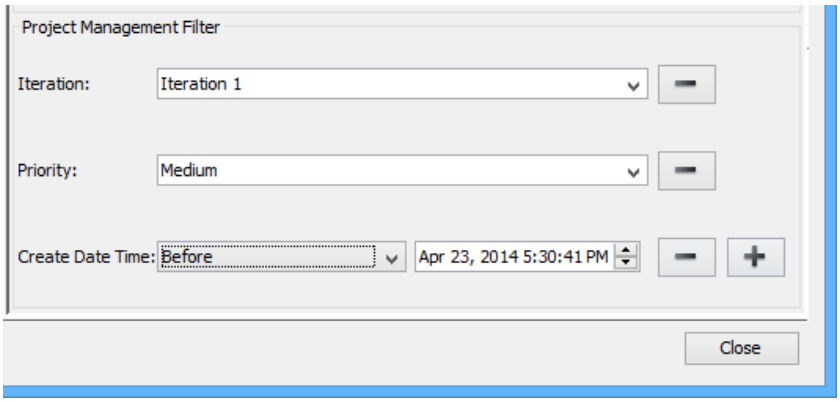
Like all the other specification level properties, project management properties can be edited through the specification dialog box of all diagrams, model and diagram elements. Select the desired diagram/model element/diagram element. Right click and select **Open Specification...** from the popup menu. Under the tab **Project Management** you can find the properties of the chosen element.



Editing project management properties

Project management properties in document generation

Like most other properties, project management properties will be presented in PDF, HTML and Word docuemnt, too. Furthermore, you can filter the elements to present in document by project management properties when editing template.



Editing project management filter in editing document template

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

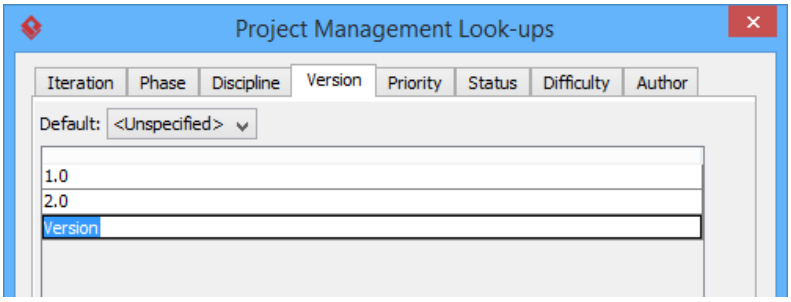
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Configuring project management properties look-ups

For each project management property, there are several default values available for selection. For example, values "1.0" and "2.0" can be selected for property **Version**. You can edit an existing value, or add additional values to a property by editing the look-ups.

To configure a property:

1. Right click on a diagram and select **Open Specification...** from the popup menu
2. Under the tab **Project Management**, select **Configure Look-ups...**
3. In the **Project Management Look-ups** dialog box, open the tab of the property that you want to edit its look-ups.
4. If you want to rename a value, double click and re-enter its value. If you want to add a lookup value, click **Add** at the bottom right corner.



To add a version

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Style and formatting

You can change shapes' appearances freely by editing their fill color, font and line style. In this chapter, we will walk through in detail.

Applying fill, line and font styles on diagrams

Shows how to edit the fill, line and font of shapes.

Managing and applying styles

You can save formatting properties as a style to aid in reusing in other shapes. This page tells you how to save a style, and reuse it on another shape.

Setting line style

Controls how connector routes. There are five options: rectilinear, oblique, curve, round oblique and round rectilinear.

Setting line jumps options

When two connectors intersect with each other, the line jump option determines how the intersection will be rendered.

Setting connector caption orientation

Controls how a caption of connector appear against different connector orientations.

Format copier

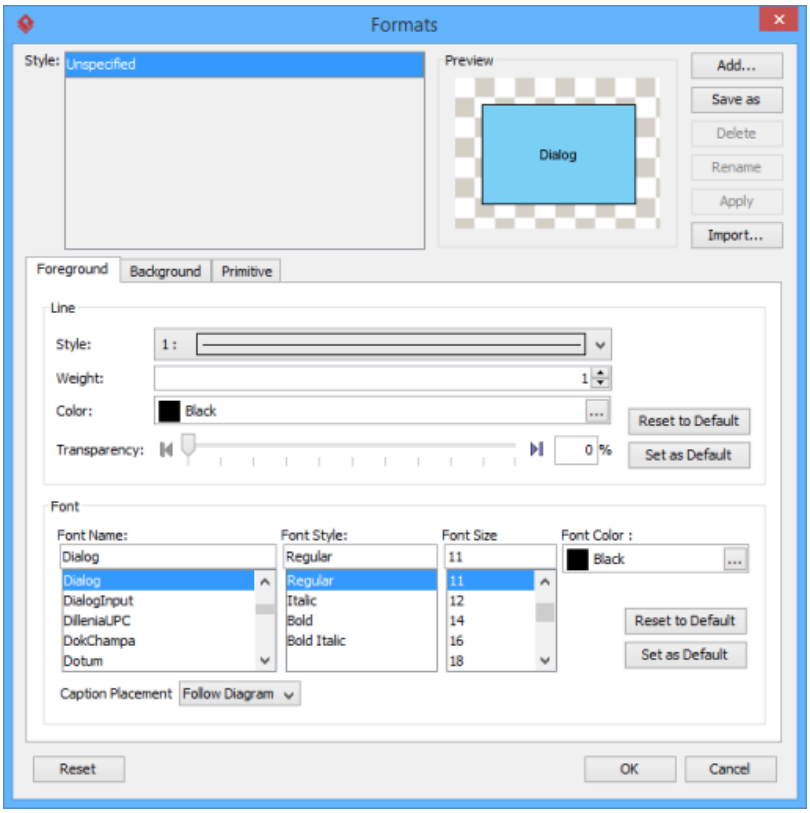
Shows you how to copy the formatting properties of a shape and copy to another.

Set connection point

Determines how to position the end point of connector, whether to point to the shape's border or point into the center.

Applying fill, line and font styles on diagram elements

You can change the diagram element's style in the **Formats** window. To open the **Formats** window, right click on the shape and select **Styles and Formatting** > **Formats...** from the pop-up menu.



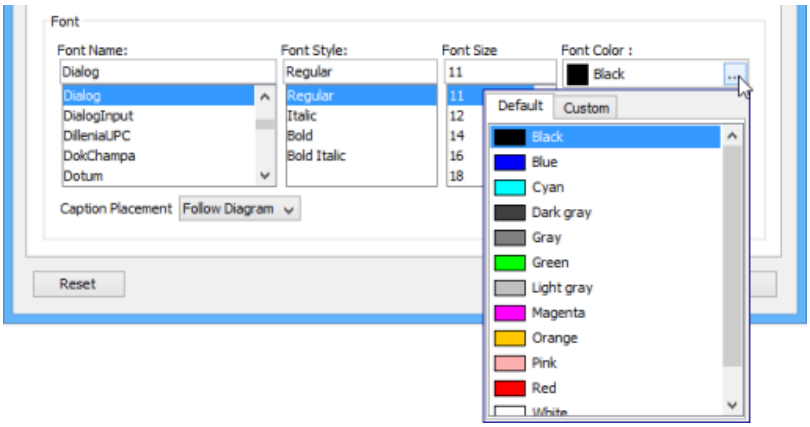
The Formats window

You can change the following settings from the **Formats** window

- Changing shapes foreground style
- Changing shapes font style
- Changing shapes background style

Changing shapes foreground style

In the **Format** window, you can change the foreground style in the **Font** section. Just click on the **...** button beside the **Color** field to select a color either from the **Default** page (which shows predefined colors) or from the **Custom** page (which shows a larger variety of colors, and allows you to define any custom colors). If you want to specify a custom color, you can switch to the **Custom** pane.



Font section

After change the font color, the preview will update automatically.

Changing shapes font style

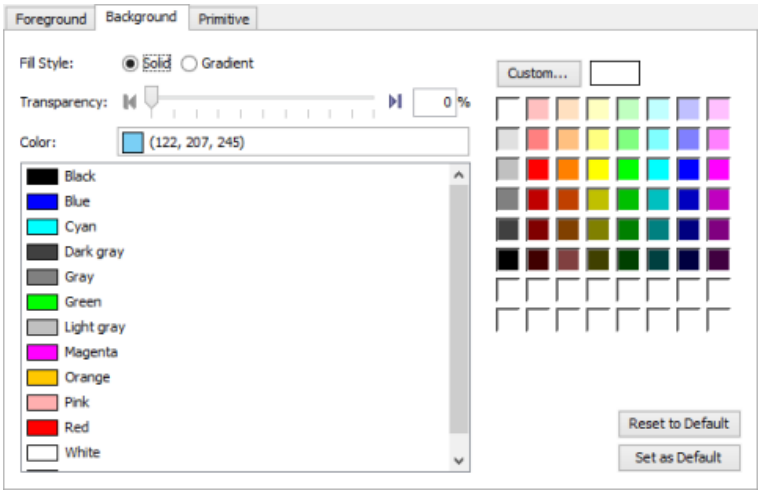
In the **Font** section, you can also change the font name, style and size.

Field	Description
Font Name	Select different types of font. The number of fonts depends on the fonts available in your computer.

Font Style	Select the style of font. You can select one of the 4 styles, a preview will be shown for each of the style items.
Font Size	Select the size of font. You may either click on the default sizes or enter the font size in the text fields.

Changing shapes background style

You can click on the **Background** tab to custom the background style.

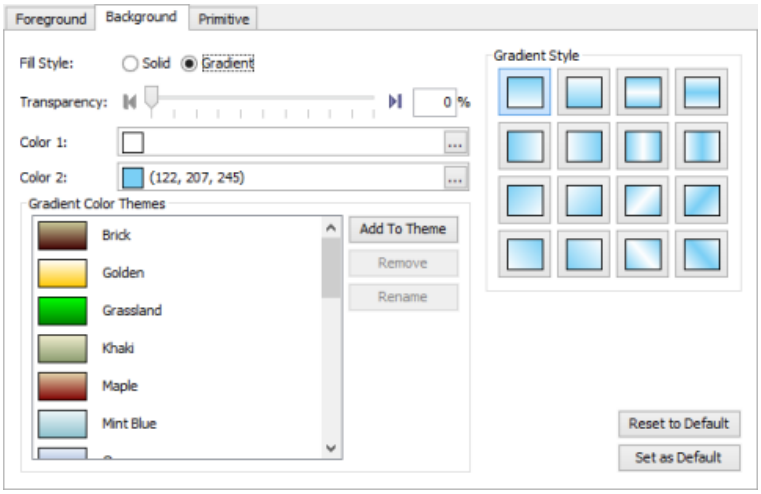


Change background style

In the **Background** tab, it allows you to select a solid fill color or a gradient fill color as well as define its transparency.

Field	Description
Fill Style	Select the fill style of the fill color. It can either be **Solid** (a single color) or **Gradient** (a fill color that is mixed by two colors).
Transparency	Specify the transparency of the fill color. The greater the value, the more transparent is the shape. 0 (zero) transparency makes the fill color completely opaque, while 100 (one hundred) transparency makes the fill color completely transparent. You can adjust the transparency by dragging the slider or by typing the value in the text field. Alternatively, you can click the Opaque button to set the fill color to opaque or click the Transparent button to set the fill color to transparent.
Preview	The Preview pane displays a rectangle that is filled with the editing fill color. The background is checked so that you can also preview the transparency of the fill color as well.
Save as Default	Save the current fill color as the default fill color for new shapes, click the **Set as Default** button.
Reset to Default	Reset the current fill color to the default fill color, click the **Reset to Default** button.

Upon selecting **Gradient** from the **Fill style** field, you will see the detail pane for formatting a gradient fill color.



Check gradient fill style

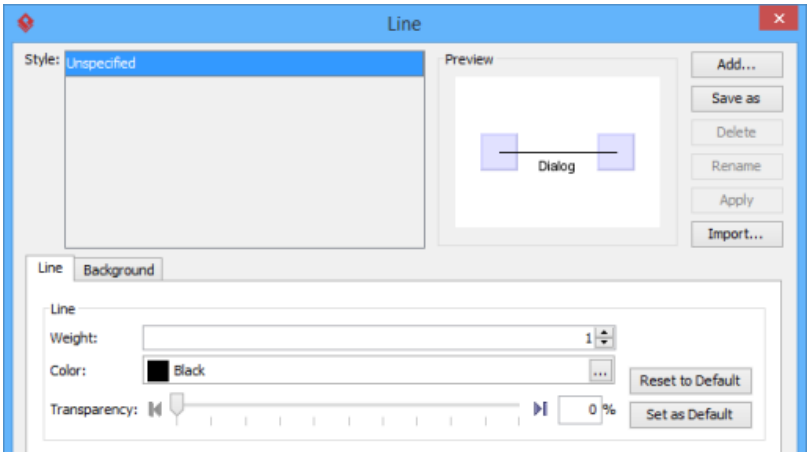
Field	Description
Color 1	You can select the first color of the gradient from the **Color 1** field. To select a color click the **...** button or double-click on the color editor. A color chooser will appear for you a select a color.
Color 2	You can select the second color of the gradient from the **Color 2** field. To select a color click the **...** button or double-click on the color editor. A color chooser will appear for you to select a color.

Gradient Color Themes	The Gradient Color Themes pane displays a list of pre-defined gradient color themes. To add a new color theme select **Color 1** and **Color 2** then click the **Add to Themes...** button. Please note that you must select a combination of colors that does not already exist in the color themes. To rename a theme click on the **Rename...** button or double-click on the desired theme. An input dialog will appear for you to enter a new name. To remove a theme select the theme and click on the **Remove** button, or use the Delete key instead.
Gradient Style	The Gradient Style pane allows you to select the gradient style of the gradient fill color (the angle of how the gradient color is drawn). There are sixteen pre-defined gradient styles, which are shown as toggle button in the Gradient Style pane. To select a gradient style to use click on one of the styles.

Click **OK** button to confirm editing. As a result, the shape is changed into formatted style.

Changing connector line style

To change a connector line's style, open the **Formats** window first. Right click on the connector and select **Styles and Formatting** > **Formats...** from the pop-up menu. Then, you can format the line style in the **Line** tab. It allows you to adjust weight (thickness), color and transparency.



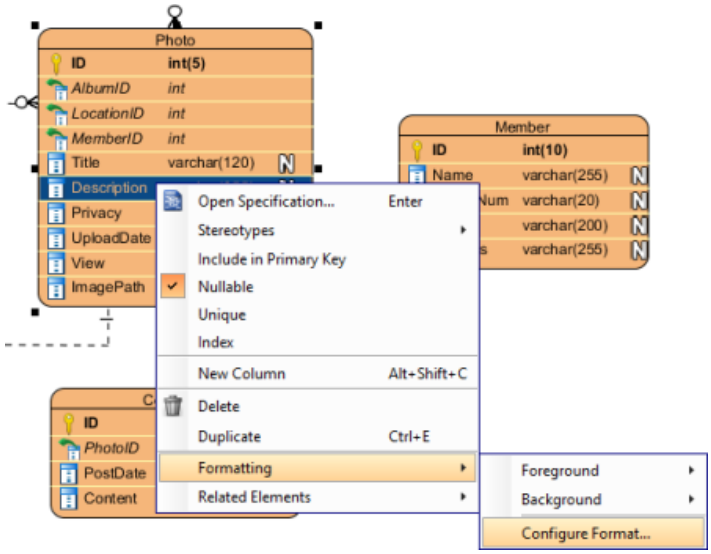
Line section

Field	Description
Weight	Adjust the weight (thickness) of a line. The greater the value, the thicker the line. You can use the up/down button to increase/ decrease the line weight, or you can type directly into the text field. The line weight ranges from 1 to 20.
Color	Specify the line color. Click on the **...** button beside the **Color** field to select a color, either from the **Default** page (which shows predefined colors) or from the **Custom** page (which shows a larger variety of colors, and allows you to define any custom colors). **NOTE:** Only integer values can be used for line weight. If you type 2.8 in the text field, 2 will be applied instead.
Transparency	Specify the transparency of the line. The greater the value, the more transparent the line. 0 (zero) transparency makes the line completely opaque, while 100 transparency makes the line completely transparent. You can adjust the transparency either by dragging the slider, or by typing the value in the text field. Alternatively you can click on the Opaque button to set the fill color to opaque, or click on the Transparent button to set the fill color to transparent.
Preview	The Preview pane displays a rectangle surrounded by the line with the selected line format applied.

Changing class/entity members' styles

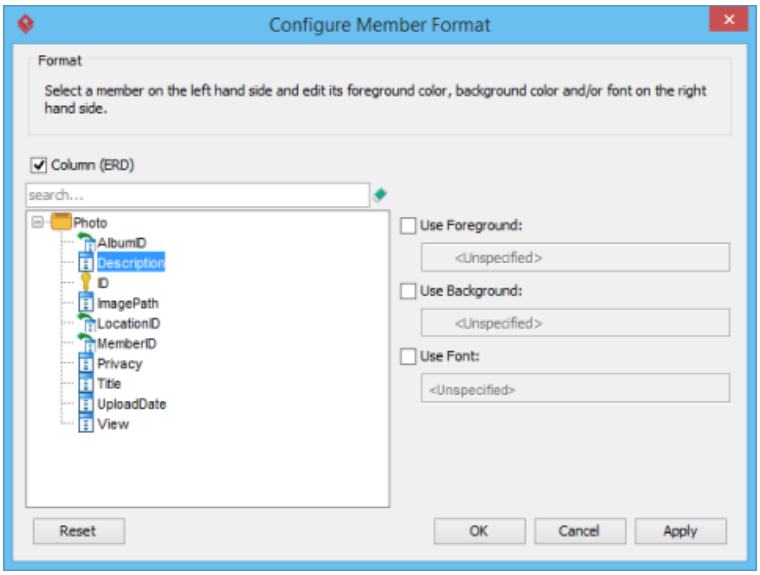
You can also configure the formatting of class and entity members like attributes, operations and entity columns. To configure the formatting of member:

1. Right click on that member in the parent shape.
2. Select **Formatting > Configure Format...** from the popup menu.



Configure the format of an entity column

3. In the **Configure Member Format** window, select the property or properties that you want to edit - foreground, background and/or font, and then make the changes you want. If you want to apply changes to multiple members, you can perform a multiple selection in the list on the left hand side. Press **Shift/ Ctrl** for a multiple selection.



Configure Member Format

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Managing and applying styles

You can format shapes and connectors in Visual Paradigm by changing their attributes, such as line styles, weight, color and transparency. Moreover, you can apply your preferred styles or remove them after creating. Since adding and applying styles of shapes and connectors is as simple as clicking few clicks, the newly created format settings will be applied on the selected shapes/ connectors easily and instantly.

Adding styles

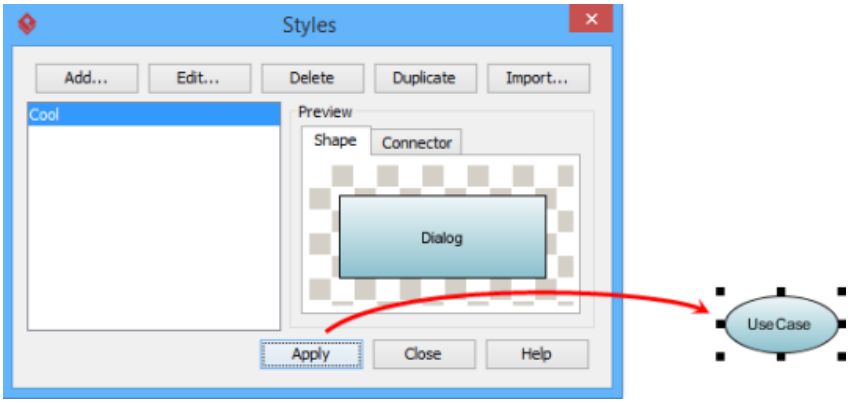
To add styles:

1. Open the styles window by selecting **View > Styles** from the toolbar.
2. In the **Styles** window, click **Add...** to create and edit a new style.
3. In the **Edit Style** window, set the name, foreground line style, font style, background style and arrow style.
4. Click **OK** button after you finish editing.

Applying styles

Upon keeping the **Styles** dialog box open, select a target shape on the diagram and click **Apply** in the **Styles** dialog box.

As a result, the shape is changed into the newly created style.



Apply styles to selected shape

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

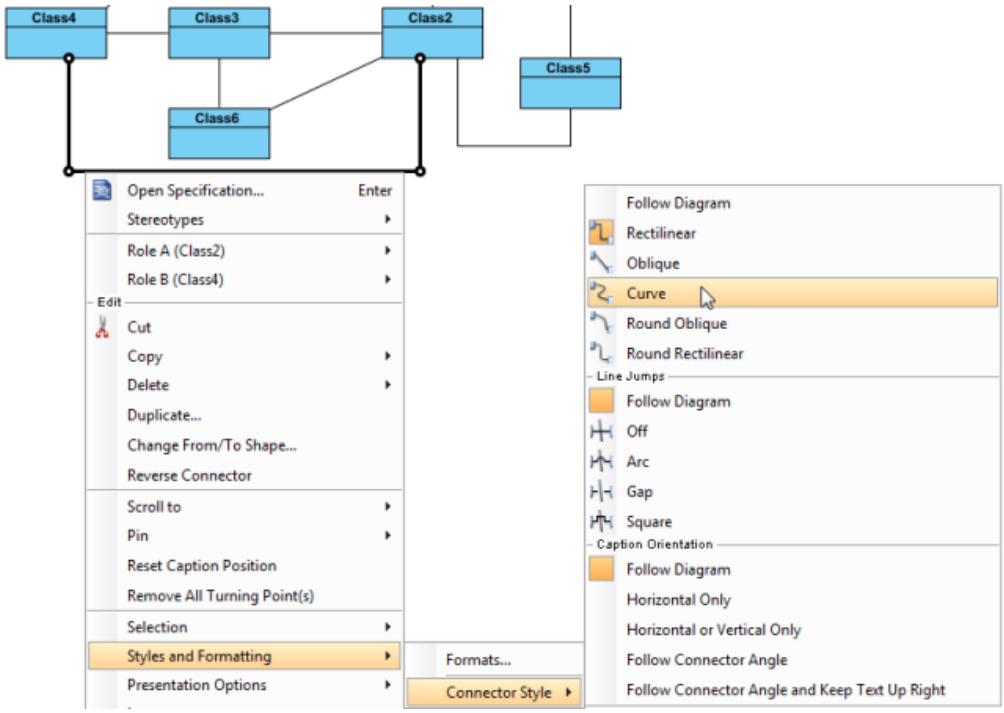
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Setting line style

Connectors are the lines that connect two shapes. When more shapes are created and more connectors appear, you may find that it is difficult to handle the straight spaghetti-like connectors. To overcome this problem, Visual Paradigm provides five connector styles to help you handle the connectors, namely **Rectilinear**, **Oblique**, **Curve**, **Round Oblique** and **Round Rectilinear**.

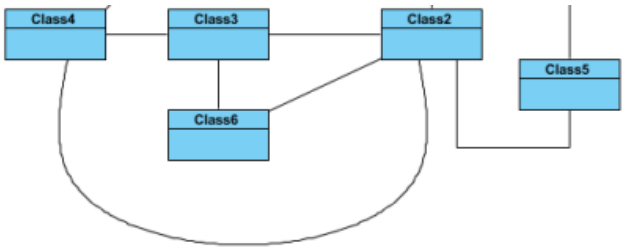
Setting connector line style

To change the line style, right click on the target connector and select **Style and Formatting > Connector Style** and one of five line style options from the pop-up menu.



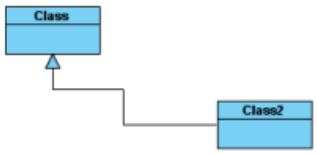
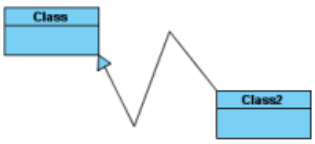
Change line style

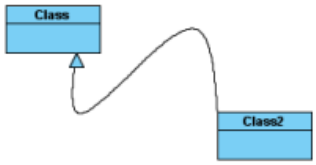
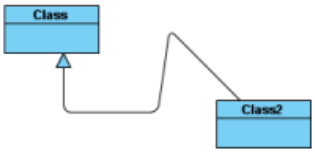
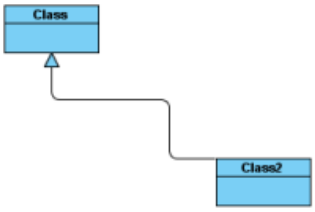
As a result, the connector is changed into the selected line style.



Change line style

Line style options

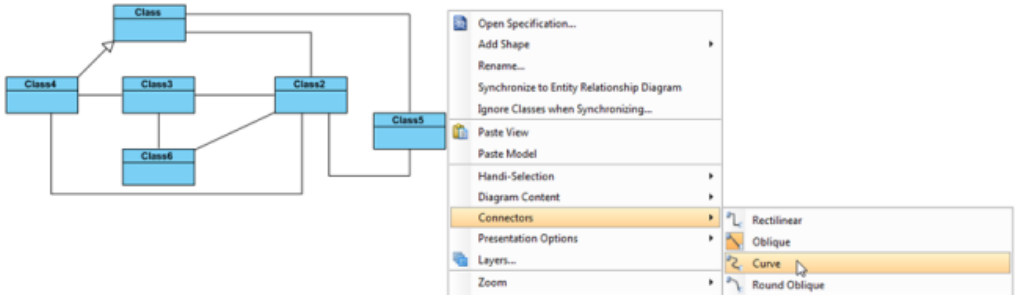
Name	Sample
Rectilinear	
Oblique	

Curve	
Round Oblique	
Round Rectilinear	

Setting diagram base line style

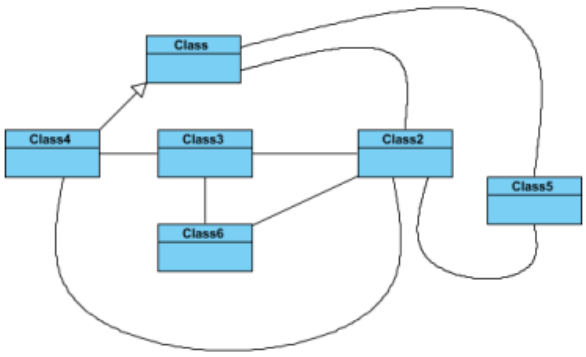
Beside the five styles mentioned above, there also have **Follow Diagram** feature, you don't need to set the connector one by one if you want to change all connectors in the diagrams which defined as **Follow Diagram**.

To change the style of all lines on diagram, right click on the diagram background, select **Connectors** and one of five line style options from the pop-up menu.



Change diagram line style

As a result, all lines on the diagram are changed into the selected line style.



All lines are changed into curve

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

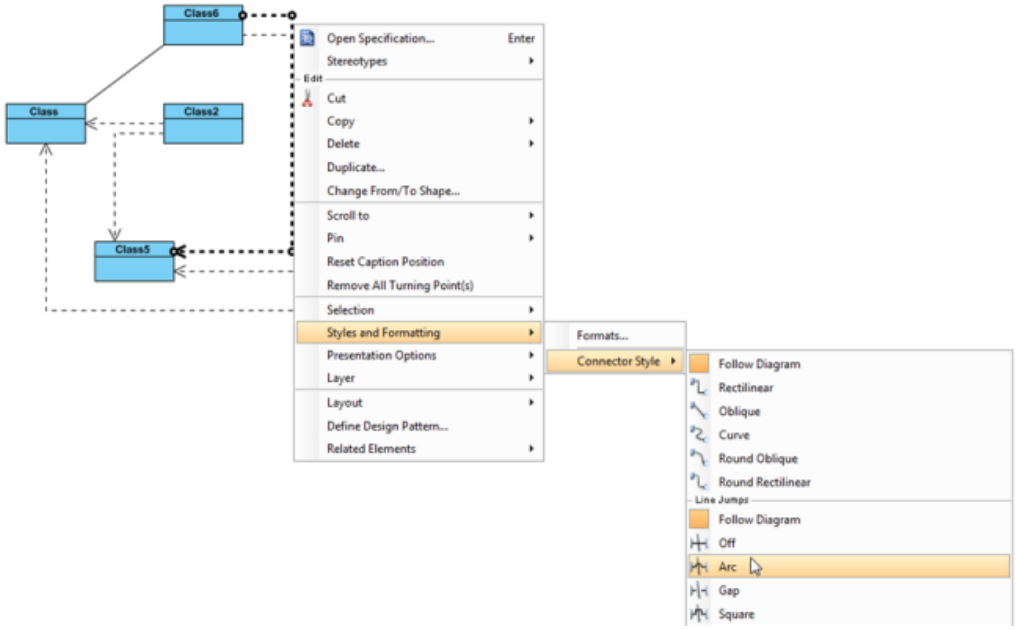
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Setting line jumps options

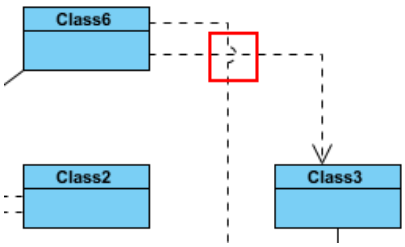
As more diagram elements on your diagram, more miscellaneous connectors are overlapped with each other. It is impossible to explicate on those intersections which connector you indicate. The advantage of Line Jumps is making one of the two connectors different to another to indicate that which connector links with which diagram element clearly. Visual Paradigm provides four line jumps options to help you to distinguish connectors. Furthermore, the enhanced feature of line jumps in Visual Paradigm enables you to set different size of line jumps.

Setting connector line jumps options

To change the jumps option of a connector, right click on the connector, select **Style and Formatting > Connector Style** and then select an option under **Line Jumps**.



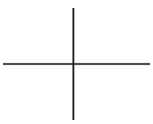
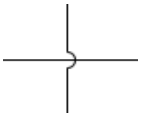
Set Arc

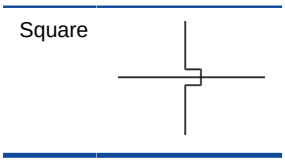


Arc sample

NOTE: The line jumps options are available only when two connectors are overlapped.

Line jump options

Name	Sample
Off	
Arc	
Gap	

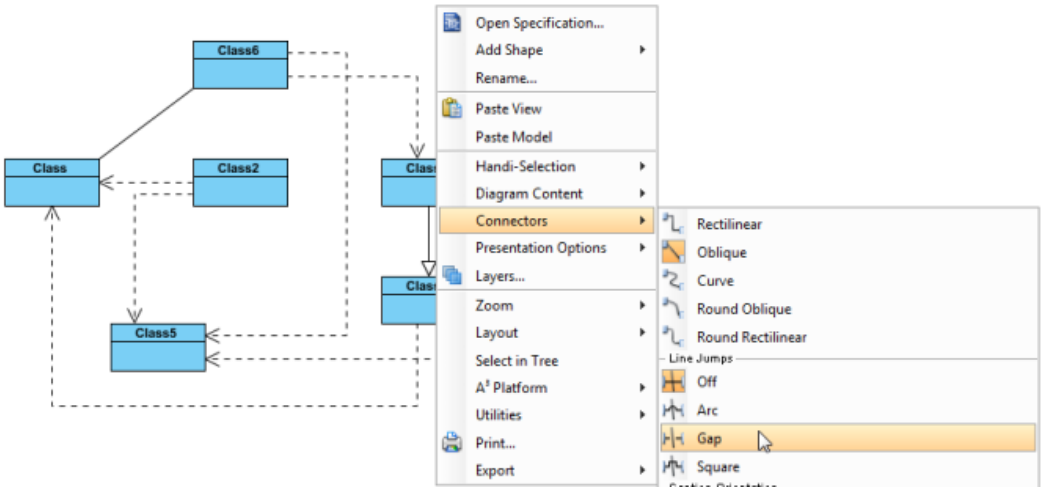


The description of 4 line jump options

Setting diagram base line jumps options

In addition to the 4 options mentioned above, **Follow Diagram** is another choice for altering the connectors. The main feature of **Follow Diagram** is, all connectors in the diagrams can be changed simultaneously instead of setting one by one.

Right click on the diagram's background, select **Connectors** and select an option under **Line Jumps** from the pop-up menu.



Select Gap from the pop-up menu

Setting line jump for new project

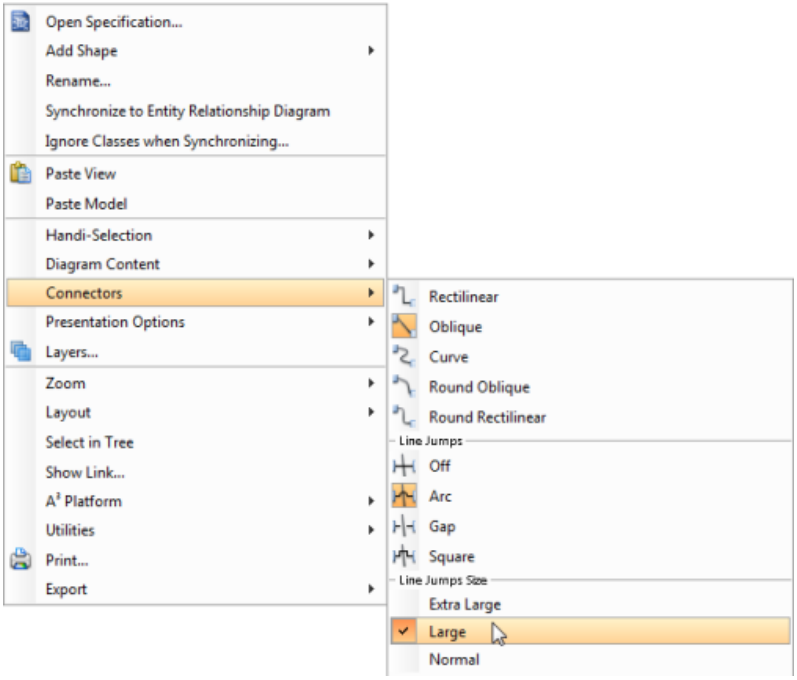
1. Select **Windows > Project Options** from the toolbar to open the **Project Options** window.
2. In the **Project Options** window, click the **Diagramming** page, open the **Connector** tab and select the **Line Jumps** style or select **Off** to disable it. At last, click **OK** to confirm the changes.

Setting different line jump size

You can enlarge the line jump size to make the selected line jump (or all line jumps) on the connector(s) more obvious. Furthermore, the size of line jump can be customized in either the current diagram or the future diagram. If you only want to set to the connectors of current diagram, right click on the diagram's background, select **Connectors** and then a size option from the pop-up menu. Otherwise, if you want to set to the connectors of future diagram, set it through **Project Options** window. Three size are availabe for chooising. Normal is the standard size by default while extra large is the maximum size.

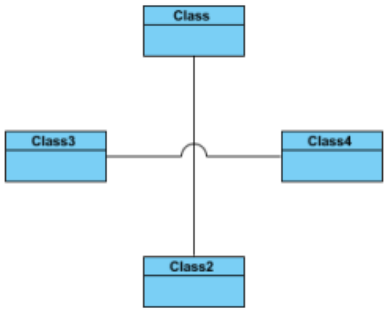
Setting line jump size for current diagram

1. Right click on the diagram's background, select **Connectors** and then a size option from the pop-up menu.



Set Large

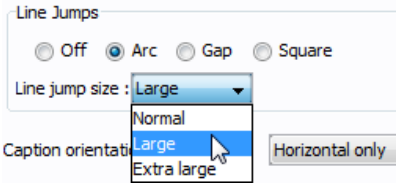
2. The line jump(s) on the current diagram will turn into the size you selected.



Arc in large size

Setting line jump size for future diagram

1. Select **Windows > Project Options** from the toolbar to open the **Project Options** window.
2. In the **Project Options** window, click **Diagramming** page and open **Connector** tab.
3. Under the **Line Jumps** section, check an line jump option and then select a line jump size option from **Line jump size**'s drop-down menu.



select Large from the drop-down menu

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

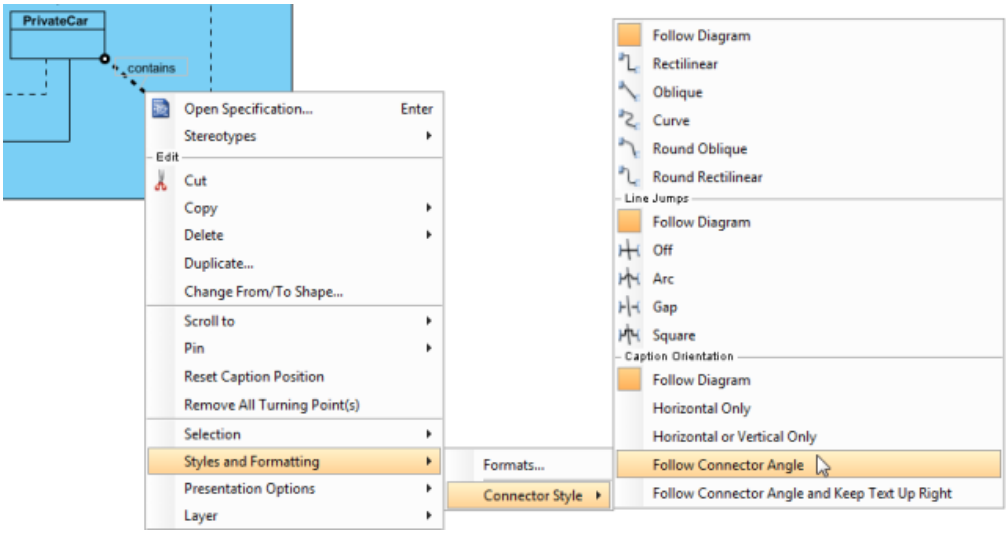
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Setting connector caption orientation

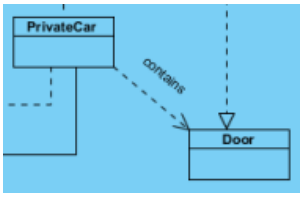
Visual Paradigm supports a number of ways of aligning connector caption, which apply on different modeling preferences. By default, the caption of a connector is aligned horizontal only, but you also can customize it to **Follow Diagram**, **Horizontal Only**, **Horizontal or Vertical Only**, **Follow Connector Angle**, and **Follow Connector Angle and Keep Text Up Right**. You can either customize it one by one or change all connectors in the diagram which defined **Follow Diagram**.

Setting connector caption orientation

To customize the caption orientation option of a connector, select the connector, right click and select **Style and Formatting > Connector Style**, and then select one out of four options under **Caption Orientation**.

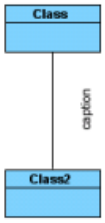
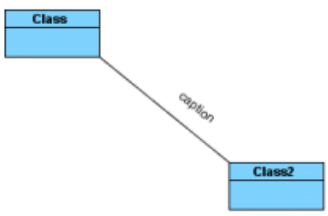


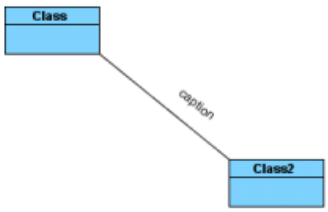
Change caption orientation to Follow Connector Angle



Follow Connector Angle sample

Caption orientation options

Name	Sample	Description
Horizontal Only		The caption of connector is aligned horizontally.
Horizontal or Vertical Only		The caption of connector is aligned either horizontally or vertically, according to the connector.
Follow Connector Angle		The caption of connector is aligned the diagonal angles of both shapes.

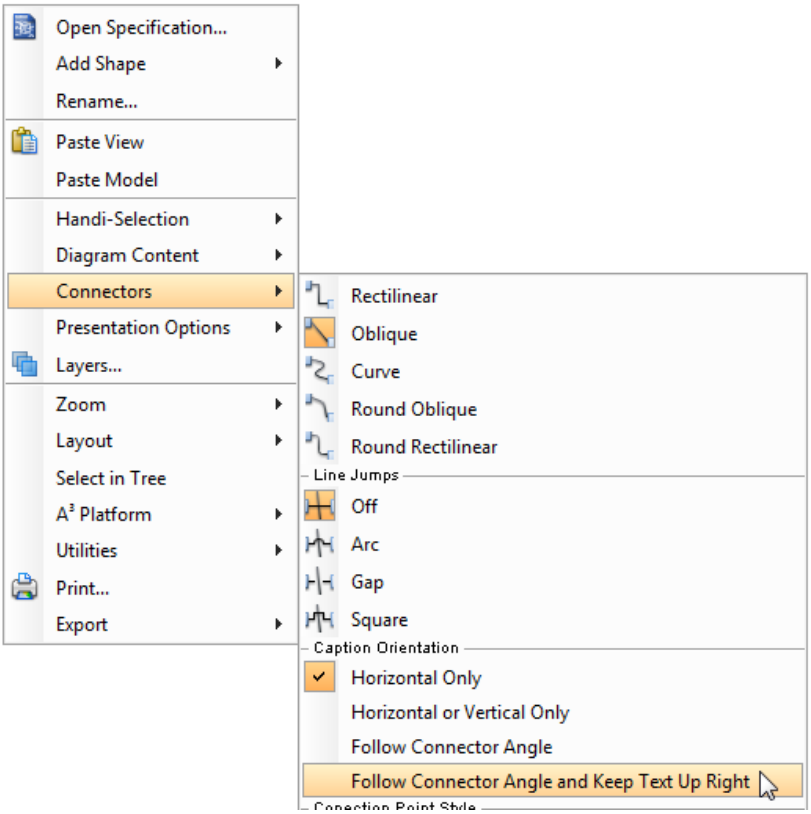
Follow Connector Angle and Keep Text Up Right		The caption of connector is aligned the diagonal angles of both shapes, but keeps top right.

The description of 4 caption orientation options

Setting diagram base connector caption direction

In addition to the 4 options mentioned above, **Follow Diagram** is another choice for altering the connector. The main feature of **Follow Diagram** is, all connectors in the diagrams can be changed simultaneously instead of setting one by one.

Right click on the diagram background, select **Connectors** and select one out of four options under **Caption Orientation** from the popup menu.



Change caption orientation by diagram popup menu

Workspace wide

Workspace wide setting affects the new connectors being created in projects created or will be created under the current workspace, including the opening project. To set, select **Windows > Project Options** from the toolbar.

In the **Project Options** window, open the **Diagramming** page, switch to the **Connector** tab and select the desired way of aligning caption under the **Caption Orientation** drop down menu. At last, click **OK** to confirm the changes.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

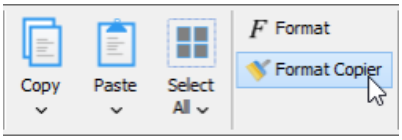
Format Copier

Format is defined as the properties for a shape in terms of fill, line and font. Shapes are formatted for two major reasons: making your project more attractive and giving emphasis on the meaning of shapes. However, it would be troublesome and time-consuming to repeat the same action when you need other shapes to have exactly the same format as the previous one you have already done. Format copier can deal with this problem for you. It's so handy that you can clone the formatting properties from one shape to another or even more.

Copying format to another shape

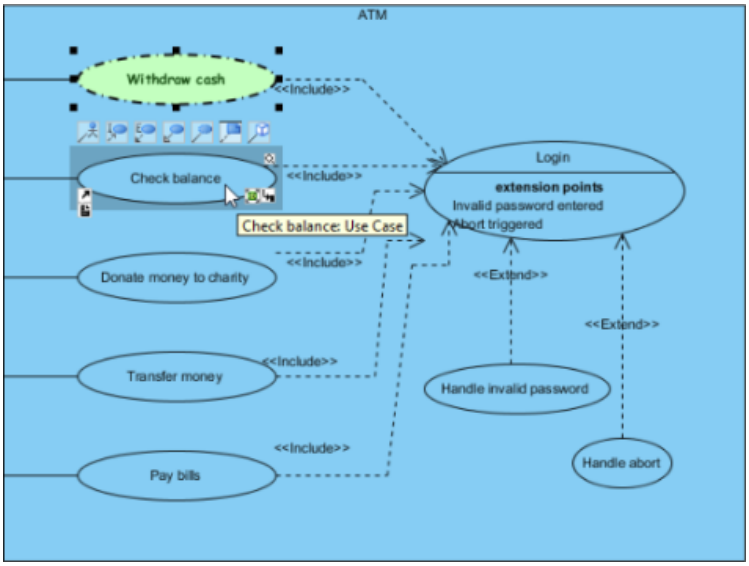
If you want another shape to have exactly the same formatting properties as the previous one you have done, you can simply:

1. Click on the shape that you want its format to be cloned.
2. Select **Diagram > Format Copier** from the toolbar.



*Click **Copier***

3. Click the shape you want to format.



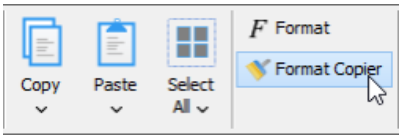
Clone the format property from one shape to another

NOTE: You can copy formatting to another type(s) of shape.

Copying format to multiple shapes

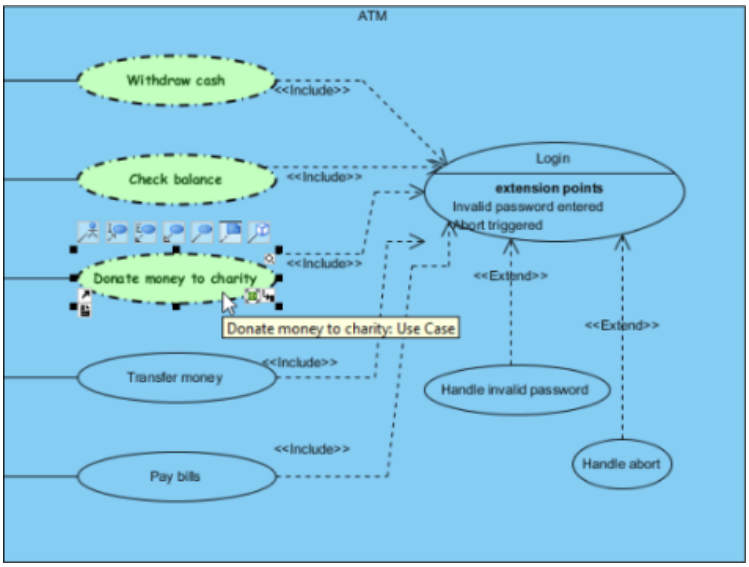
If you want the format properties of your previous shape to be cloned to more than one shape, you should:

1. Click on the shape that you want its format to be cloned.
2. Select **Diagram** and double click on the **Format Copier** button on toolbar.



*Double click **Copier***

3. Click the shape you want to format.



Clone the format property from one to multiple shapes

NOTE: If you don't want the format properties to be cloned to other shapes any more, you should cancel cloning by clicking **Copier** once again.

NOTE: You can only copy format to shapes within the same diagram.

Related Resources

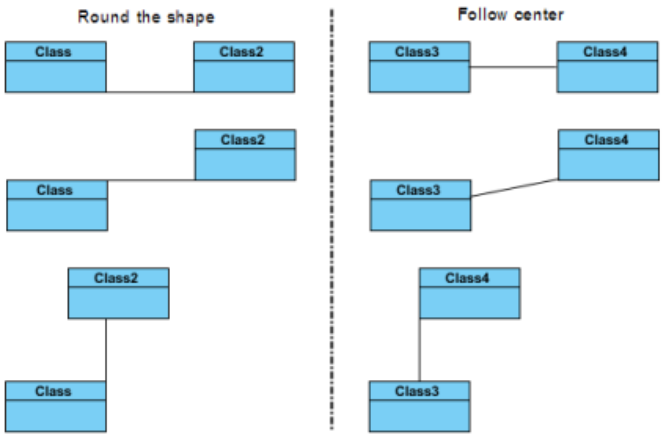
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Set connection point style

The connection point of a connector is used to connect from the original shape to the target shape using a connector. In Visual Paradigm, you can choose one of two kinds of connection point style: either Round the shape or Follow center for each shape. Round the shape allows you to set the last connection point of the connector moving along the boundaries of the original shape while Follow center refers the last connection point of the connector depends on the center of the original shape. The most attractive point of connection point style feature is that the animation of a connection point style will be playing repeatedly once you select the corresponding connection point style.

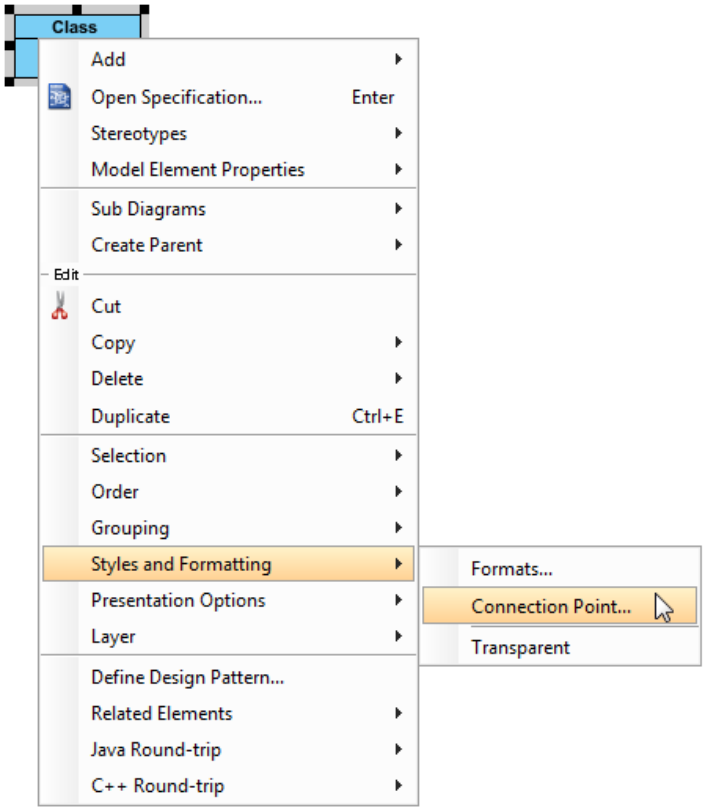
You can compare the differences of two kinds of connection point style shown as follows:



Compare two kinds of connection point style

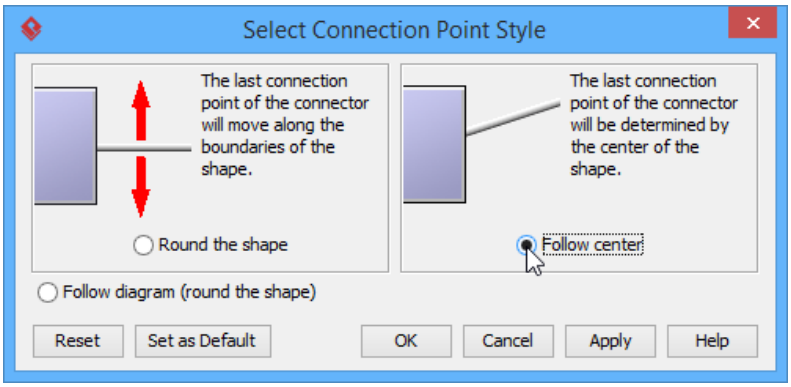
To set a connection point style:

1. Right click on shape and select **Styles and Formatting > Connection Point...** from the pop-up menu.



*Open **Select Connection Point Style** dialog box*

2. In the **Select Connection Point Style** dialog box, choose a connection point style option. Click **OK** button.



*Choose **Follow center** option*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Present shape as primitive shape

In addition to the standard appearance of notations (e.g. stickman for actor, box for class, oval for use case, etc), here you have one more option - the primitive option. The primitive option enables you to present any kind of shape as simple rectangle/oval/rounded rectangle shape. You can specify custom text to such "primitive shape", which is particular useful for modeling for general-purposes. A sample use would be to present an actor that represents a computer as a rectangle, and then describe its role with custom text and have it presented on diagram.

To show a shape as primitive shape:

1. Right click on the shape and select **Presentation Options > Primitive Shape...** from the popup menu.
2. Adjust the appearance of shape by setting **Shape type**.
3. Adjust the content of shape by setting **Text**. The **Name** option makes the name of shape presented. The **Tagged value** option enables you to display the value of a specific tag. You need to enter the tag name in the drop down menu, or select from the list of existing tags. The **Custom** option enables you to display whatever text you want.
4. Click **OK** when ready.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

General modeling techniques

This chapter covers all the general model techniques that increase the productivity of your work.

Automatic diagram layout

Helps you reorganize shapes and connectors on diagram to make diagram tidy. Several options are provided to produce different results of layout.

Fit shape size

To fit the size of shape against the update of shape content, or to fit manually.

Diagram element selection

Description of various ways of shape selection including traditional range selection and handi-selection.

Copy and paste

Common copy-and-paste editing features. Instead of pasting inside Visual Paradigm you can paste to external applications like MS Word, or to paste as XML to aid in interoperability.

Alignment guide

Helps to make sure shapes are aligned well through the visible guide line.

Reverse connector direction

To immediately reverse the direction of connector without deleting and recreating it.

Visualize model elements on diagram

Shows you how to show a model element on a diagram.

Visualize related model elements

Shows you how to show related elements of a shape on a diagram.

Adding comments

Shows how to add comments to shapes or diagrams.

Pinning connector ends

By pinning a connector's end(s) you freeze their position and make them point to the desired position.

Align and distribute diagram elements

Describes the steps about aligning and distributing shapes.

Adjusting caption's position and angle

For BPMN event and gateway, their captions can be configured to show in specific orientation. You can also make it draggable (default not).

Zooming diagram

Magnify or diminish diagram content by zooming in and out.

Diagram grids

Learn how to improve shapes' alignment and positioning with the help of diagram grids.

Automatic diagram layout

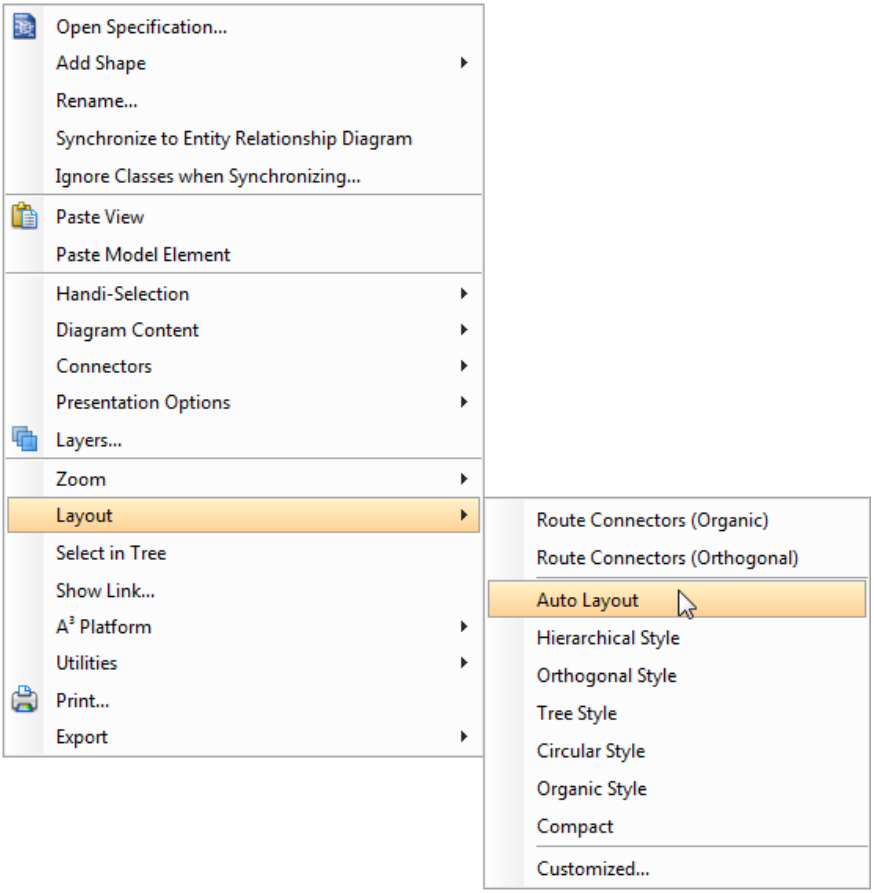
Visual Paradigm provides a layout facility for arranging diagram elements in diagrams. Diagram elements do not overlap and the relationship links do not cross over one another. Different layout styles and configurable options are provided, which allows extremely flexible and sophisticated layouts to be applied to diagrams.

Automatic layout diagram

There are a few different kinds of layouts: **Auto Layout**, **Orthogonal Layout**, **Hierarchic Layout**, **Directed Tree Layout**, **Balloon Tree Layout**, **Compact Tree Layout**, **Horizontal-Vertical Tree Layout**, **BBC Compact Circular Layout**, **BBC Isolated Circular Layout**, **Single Cycle Circular Layout**, **Organic Layout** and **Smart Organic Layout**.

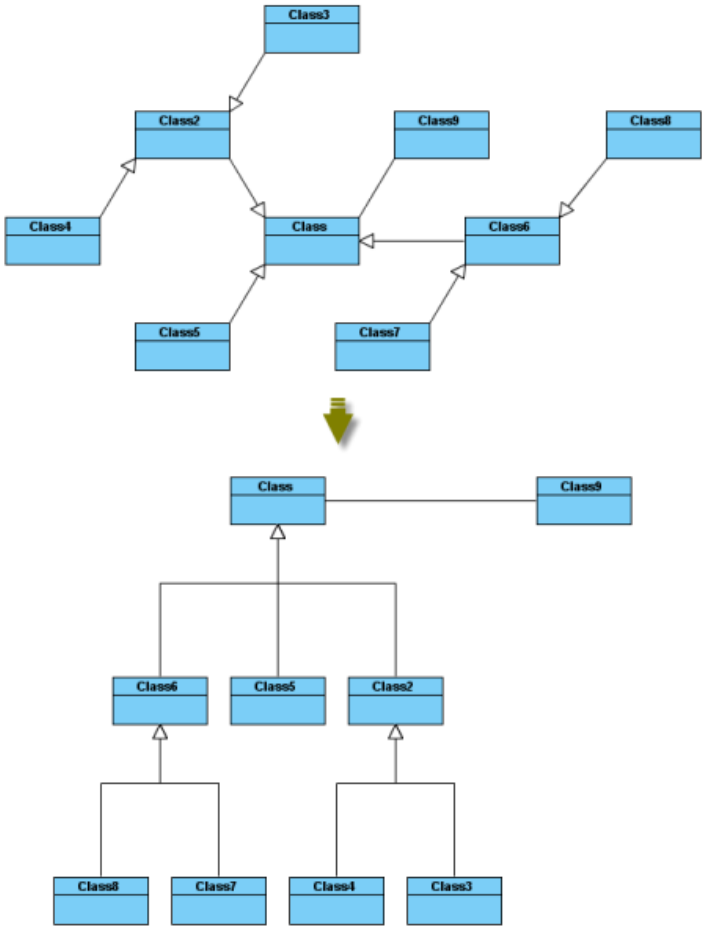
Auto layout

Selecting auto layout signifies that the most suitable layout is arranged for shapes automatically. It is the best choice for users when they have no preference in selecting a specific layout. To apply **Auto Layout** to the diagram, right click on the diagram and select **Layout > Auto Layout** from the pop-up menu.



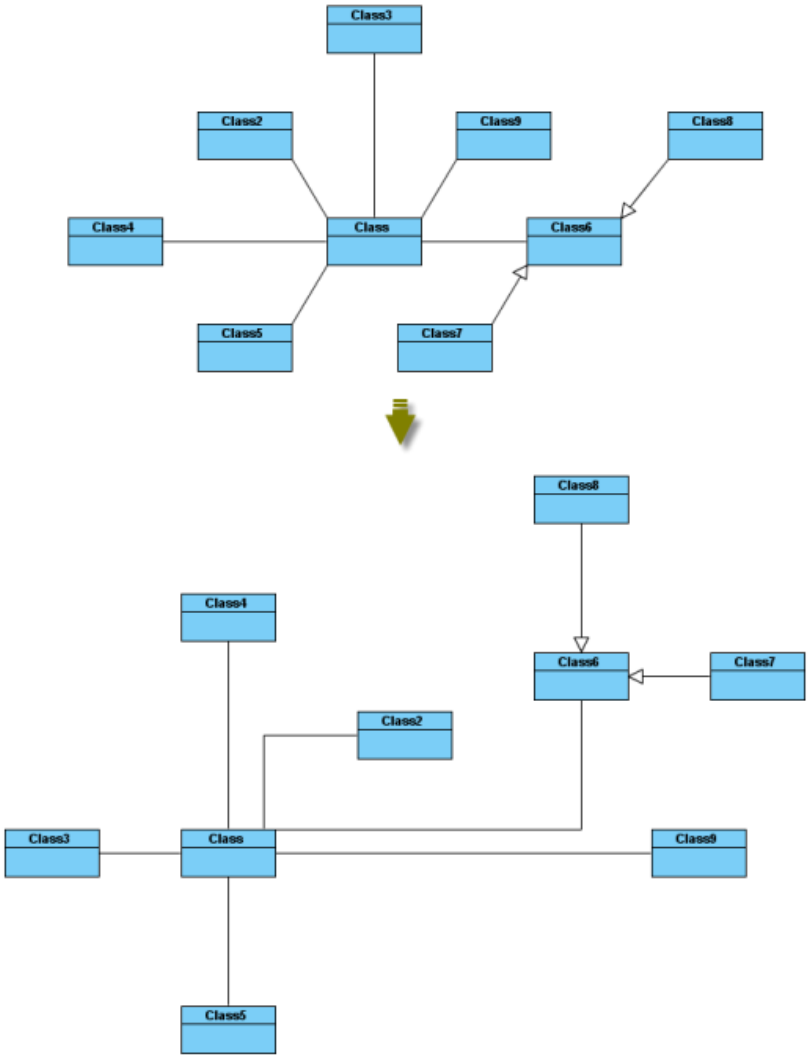
Select Auto Layout

Class diagram (Hierarchy base / factory class diagram)



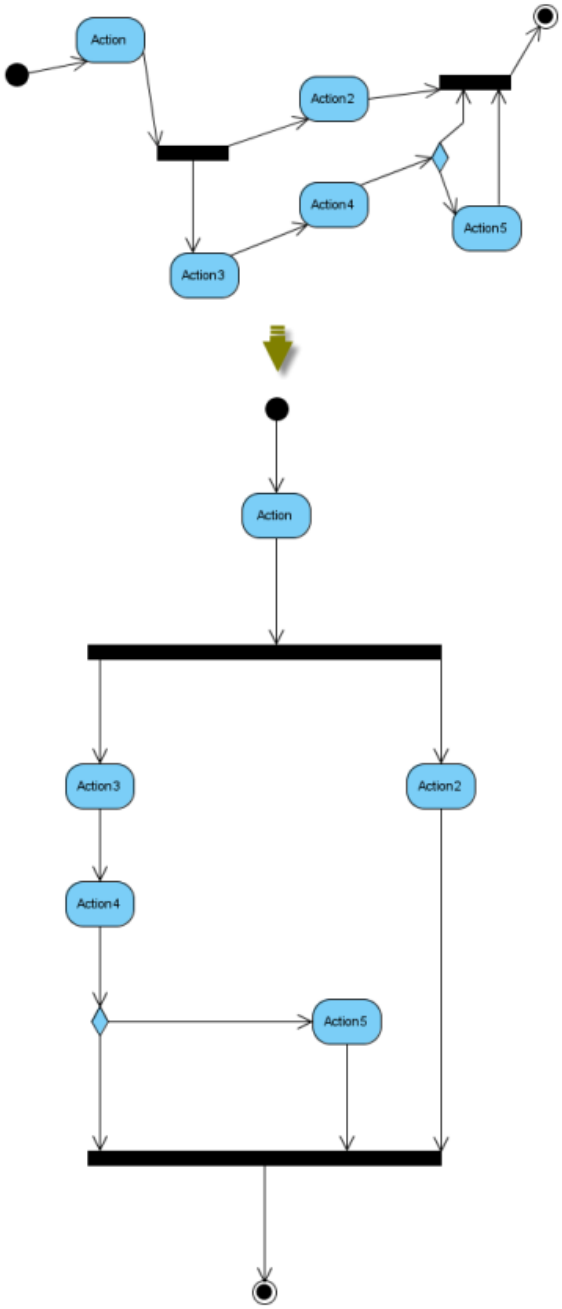
Hierarchy base (Factory class diagram)

Class diagram (Navigation base / mediator class diagram)



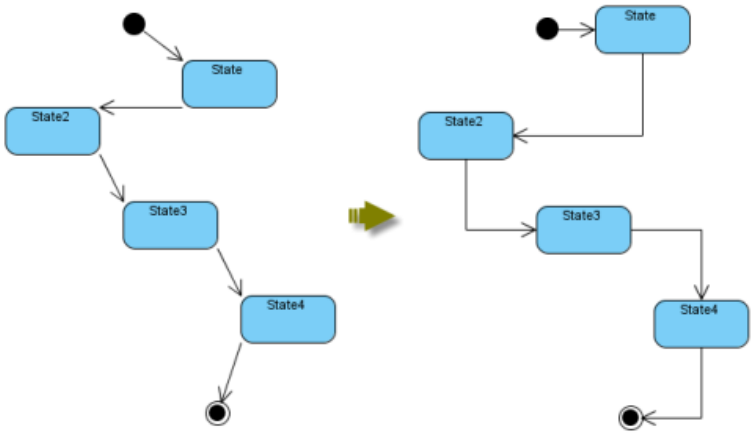
Navigation base (Mediator class diagram)

Activity diagram



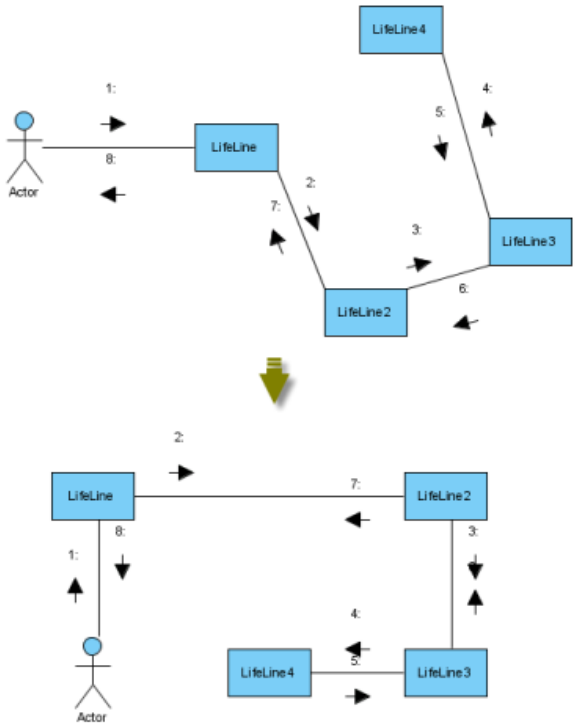
Auto Layout of activity diagram

State machine diagram



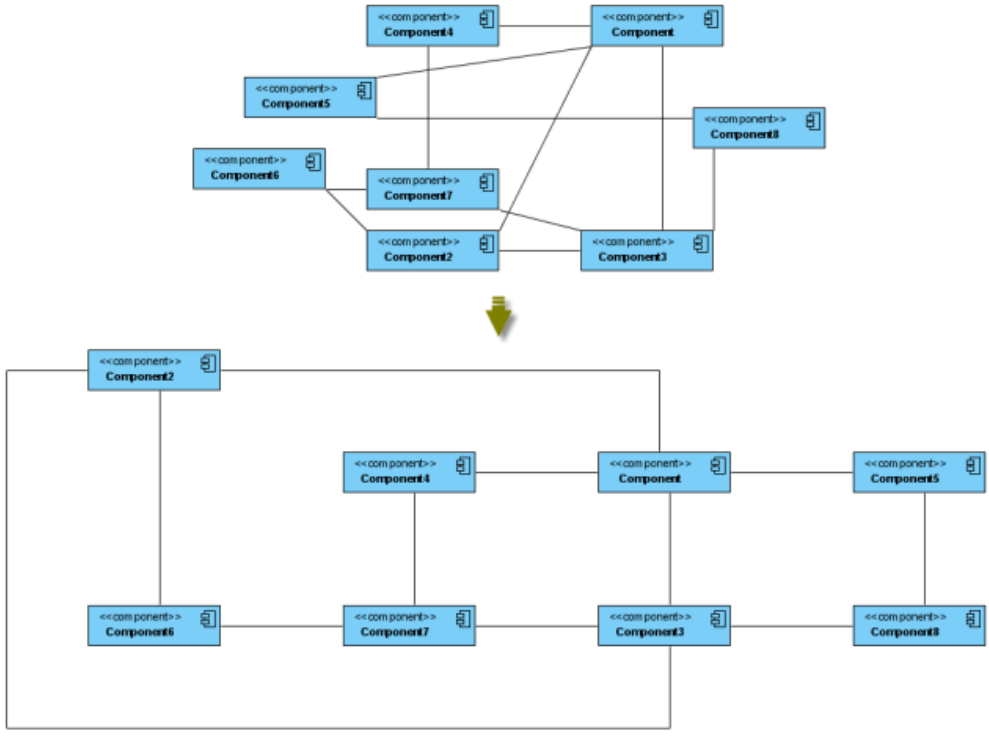
Auto layout of state machine diagram

Communication diagram



Auto layout of communication diagram

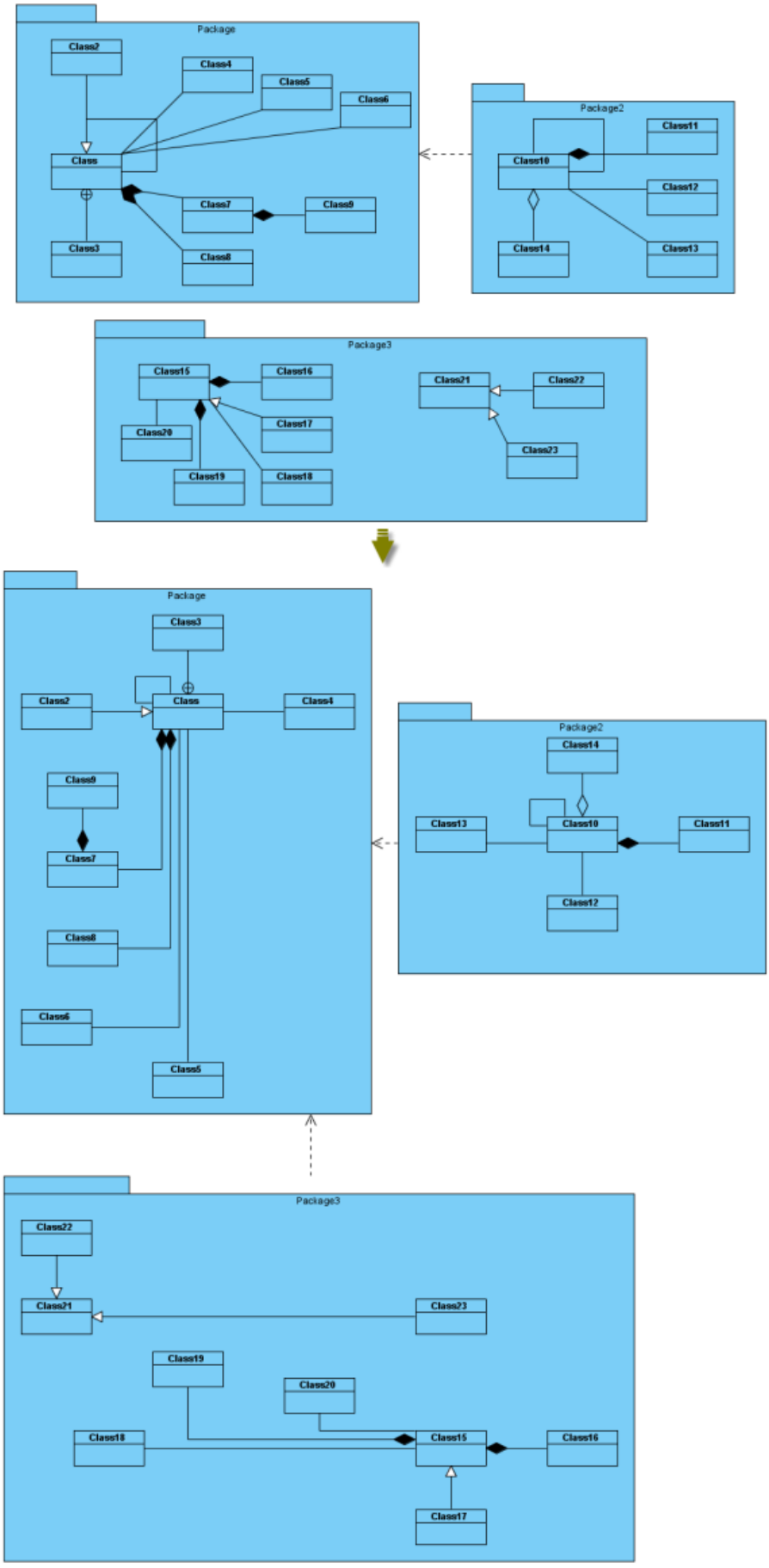
Other diagrams



Auto layout of other diagrams

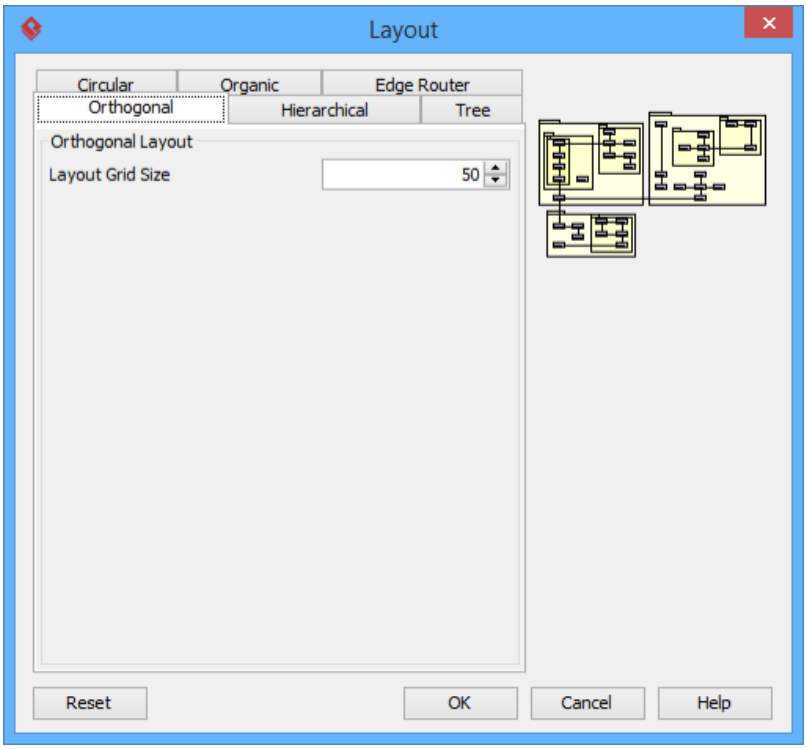
Orthogonal layout

Shapes are arranged based on the topology-shape-metrics approach in orthogonal layout. It is the best way for users to arrange shapes and connectors in Class Diagrams. As it is default layout in Visual Paradigm, every time you drag the models from the Model Tree to a diagram, the orthogonal layout will be applied to arrange the newly created shapes in the Class Diagram.



Orthogonal Layout

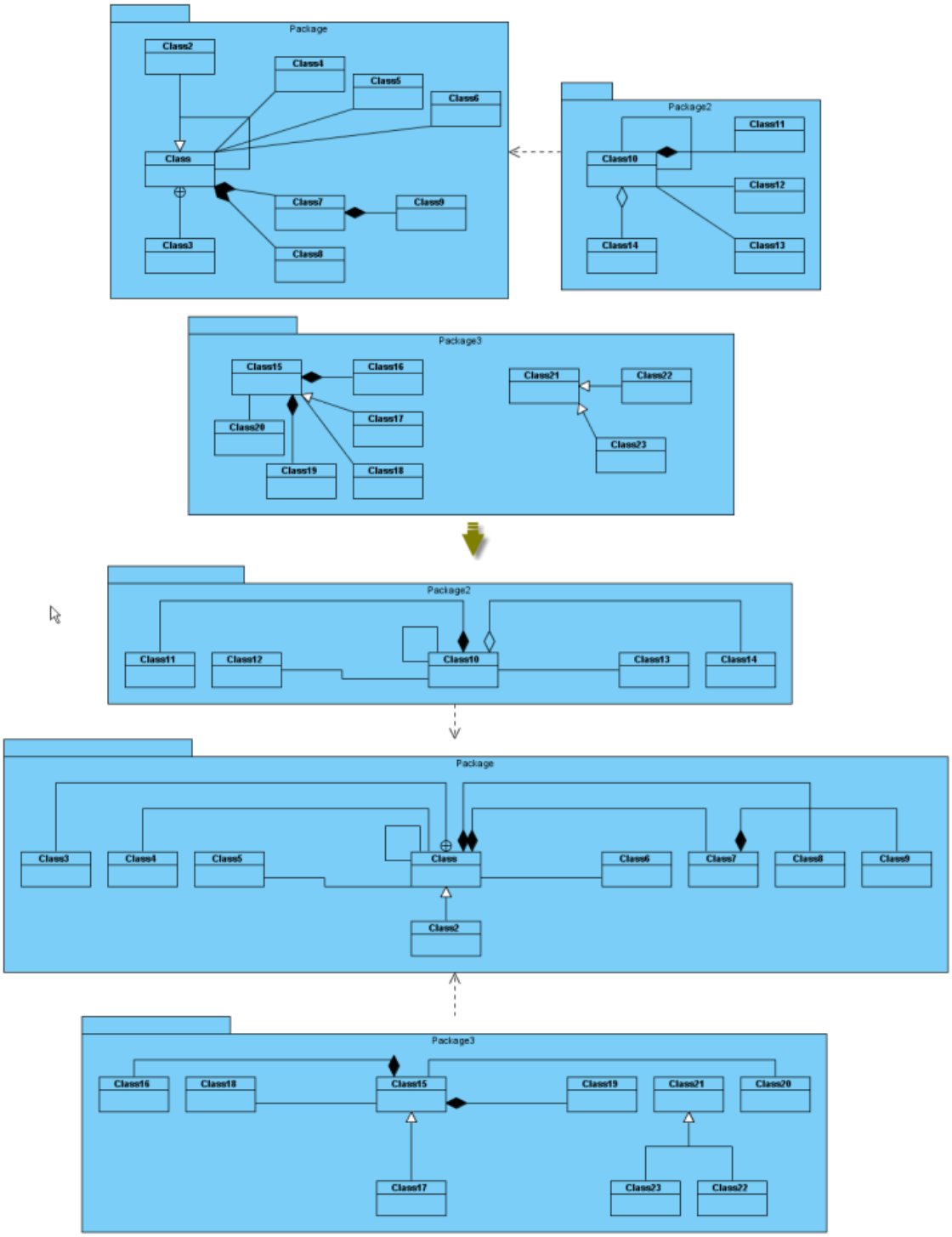
Layout Grid Size: the virtual grid size for layout. Each shape will be placed in accordance with its center point lays on a virtual grid point.



Orthogonal Layout setting

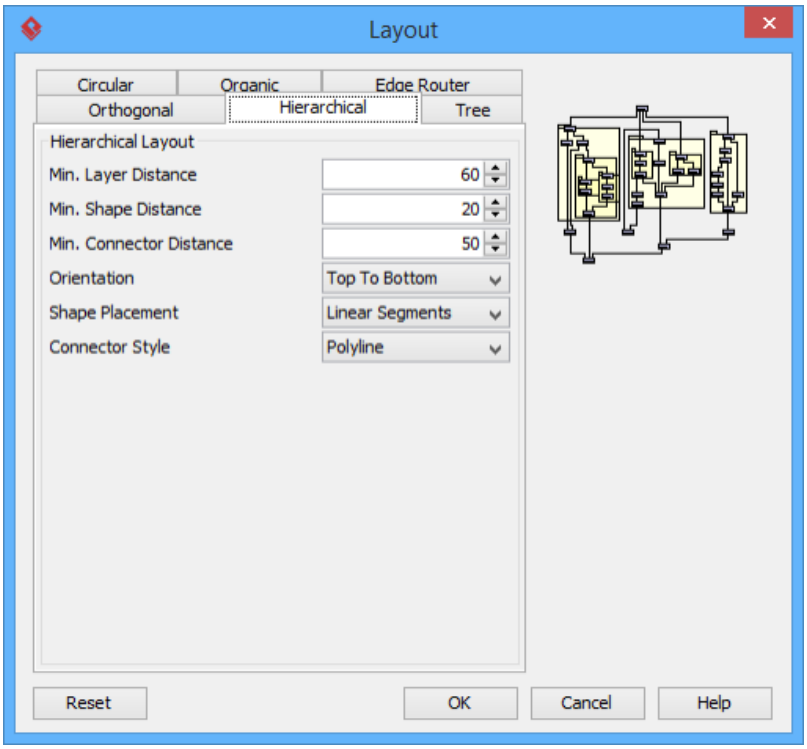
Hierarchic layout

Hierarchic Layout arranges shapes in a flow. It is the best way for users to arrange shapes that have hierarchical relationships, such as generalization relationships and realization relationships.



Hierarchic Layout

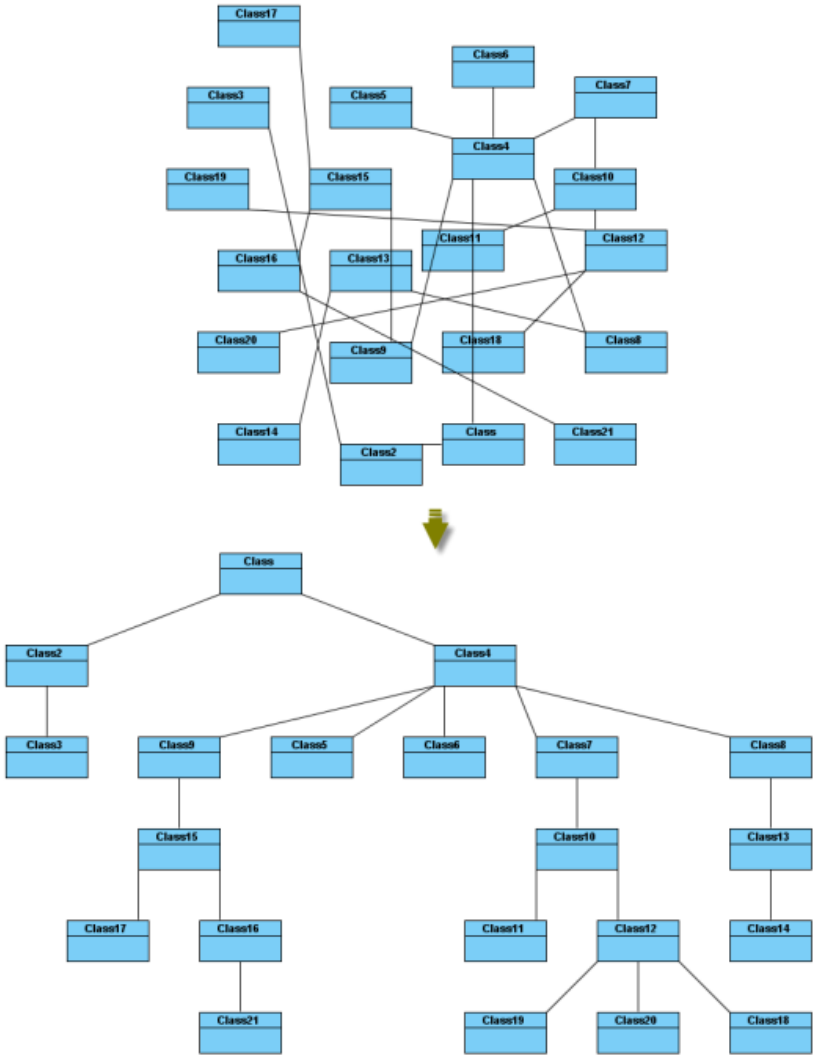
Min. Layer Distance: the minimal horizontal distance between the shapes.
Min. Shape Distance: the minimal vertical distance between the shapes.
Min. Connector Distance: the minimal vertical distance of the connector segments.
Orientation: the layout direction for arranging nodes and connectors -top to bottom, left to right, bottom to top, and right to left.
Shape Placement: affects the horizontal spacing between shapes, and the number of bends of the connectors -pendulum, linear segments, polyline, tree and simplex.



Hierarchical Layout setting

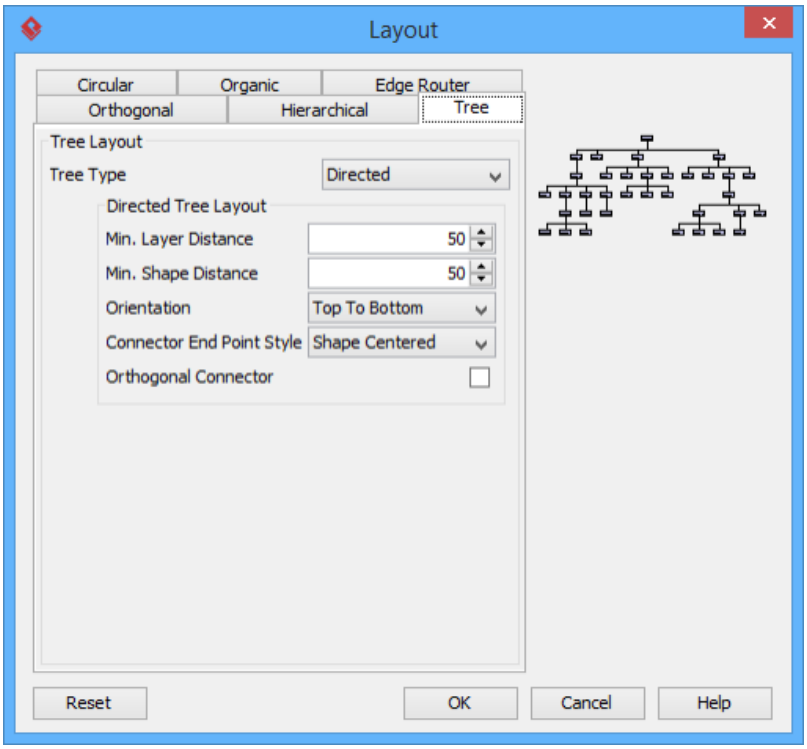
Directed tree layout

Directed Tree Layout is one of the tree layouts in Visual Paradigm which arranges shapes in a tree structure. It is the best way for users to arrange shapes except those which have hierarchical relationships, such as generalization relationships and realization relationships.



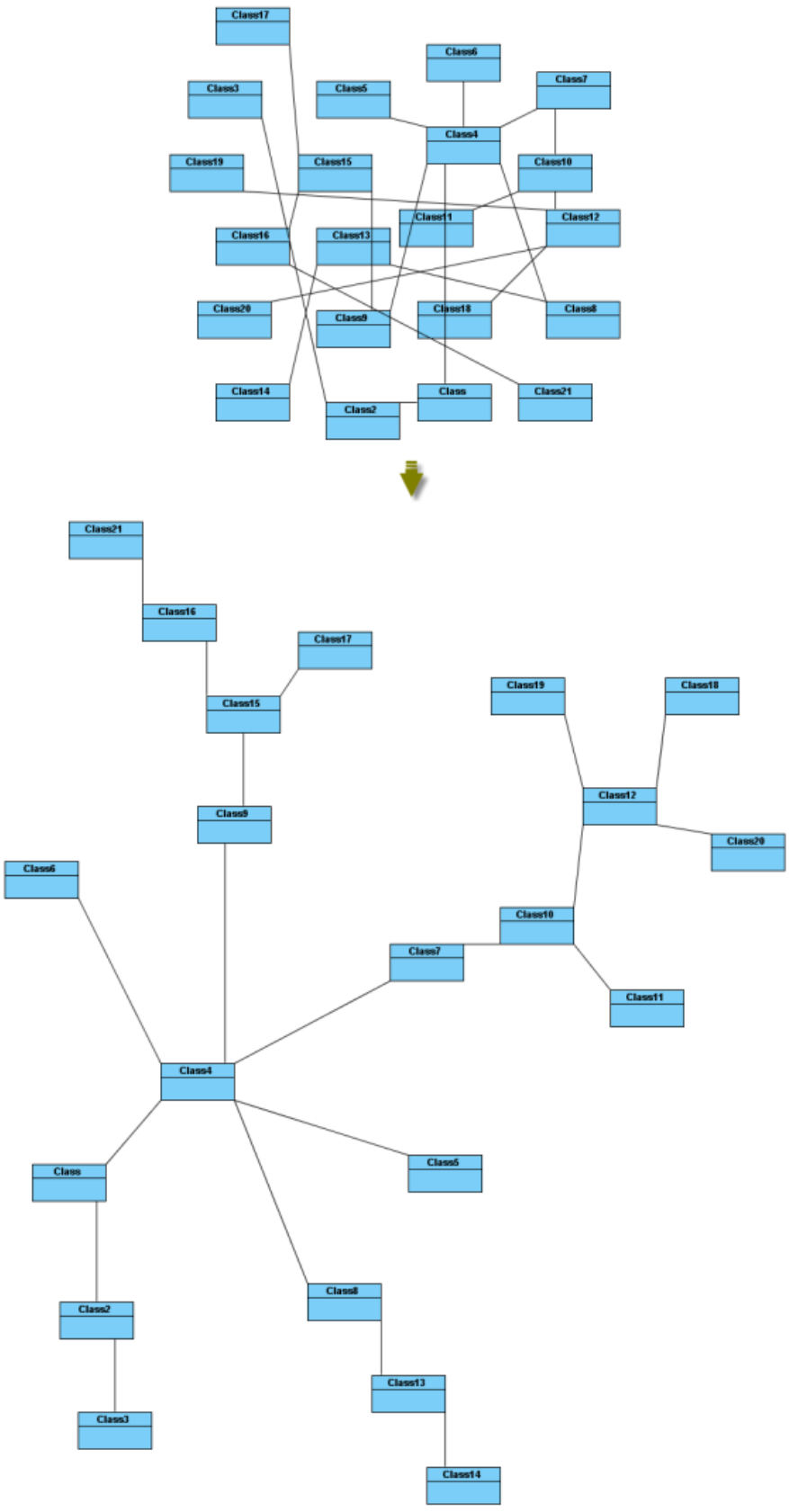
Directed Tree Layout

Min. Layer Distance: the minimal horizontal distance between the shapes.
Min. Shape Distance: the minimal vertical distance between the shapes.
Orientation: the layout direction for arranging nodes and connectors – top to bottom, left to right, bottom to top, and right to left.
Connector End Point Style: how the connector end points will be placed – shape centered, border centered, border distributed.
Orthogonal Connector: whether the connectors will be arranged in orthogonal.



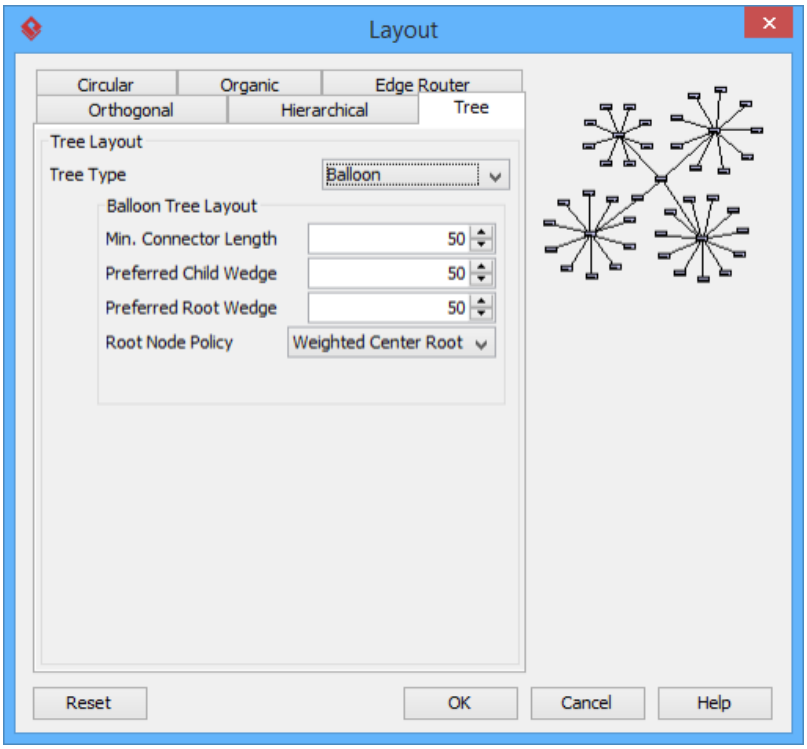
Directed Tree Layout setting

Balloon tree layout
Balloon Tree Layout, which is one of the tree layouts in Visual Paradigm, arranges shapes in a tree structure in a radial fashion. It is the best way for users to arrange large trees.



Balloon Tree Layout

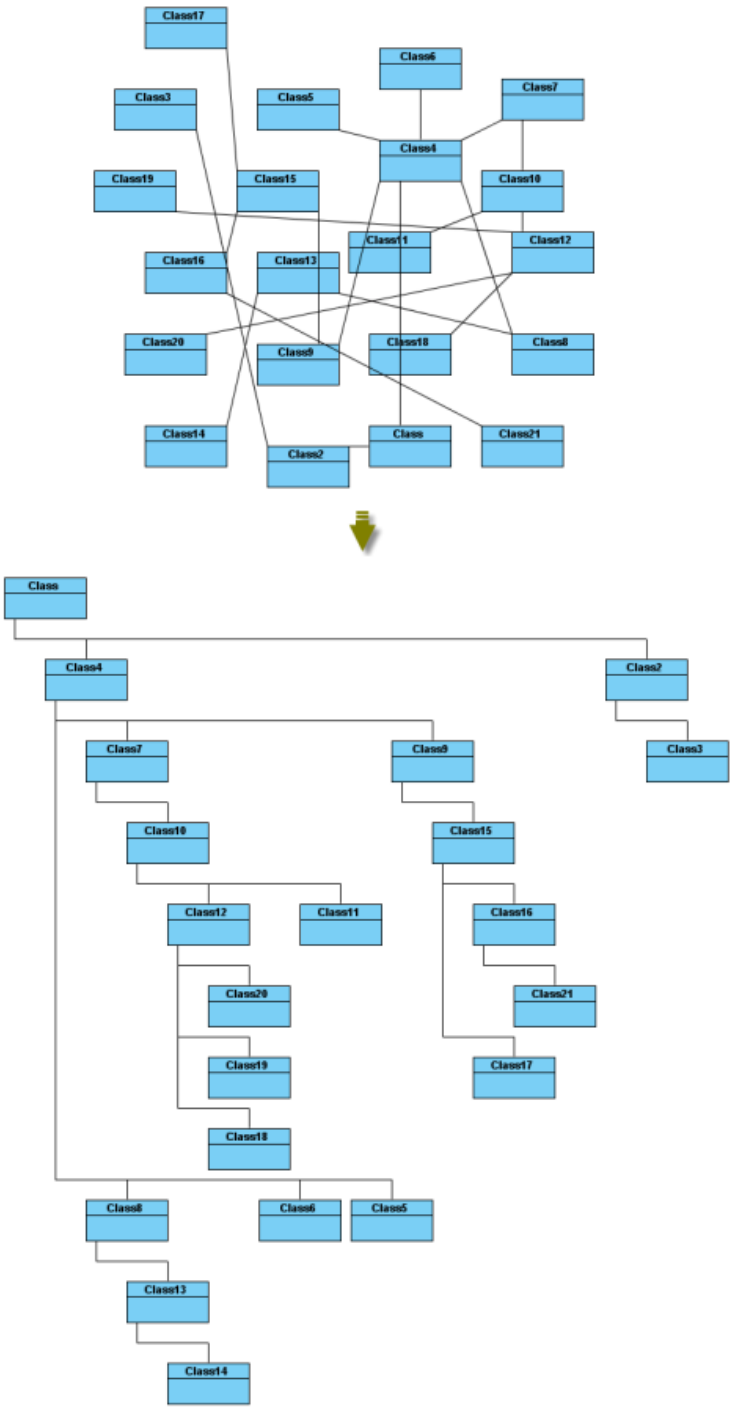
Min. Connector Length: the minimal distance between the connectors and shapes.
Preferred Child Wedge: the angle at which the child node will be placed around its parent node.
Preferred Root Wedge: the angle at which a node will be placed around the root node.
Root Node Policy: determines which node is chosen as the tree root node for layout – directed root, center root, and weighted center root.



Balloon Tree Layout setting

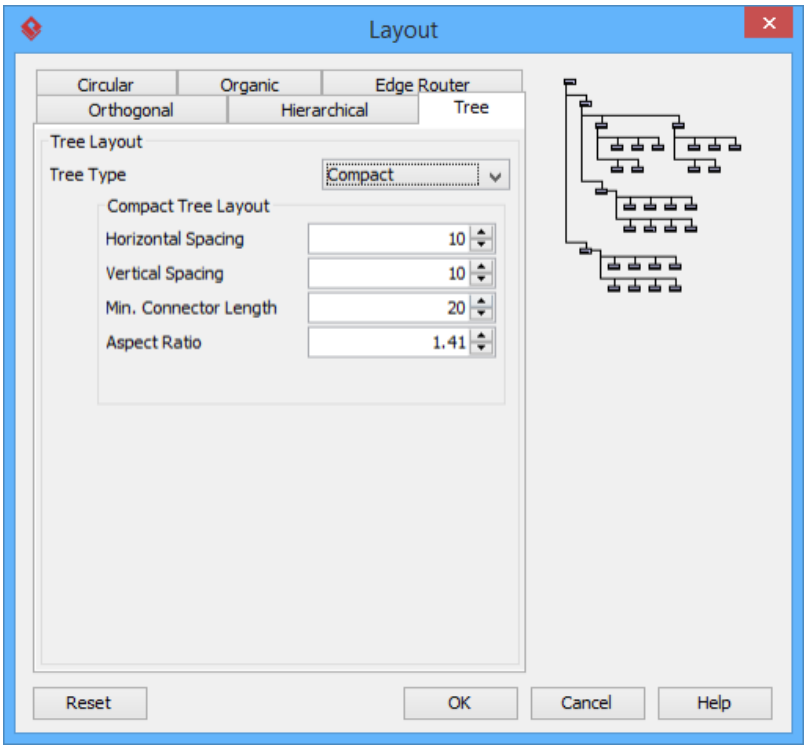
Compact tree layout

Compact Tree Layout is one of the tree layouts in Visual Paradigm which arranges shapes in a tree structure. The aspect ratio (relation of tree width to tree height) of the resultant tree can be set.



Compact Tree Layout

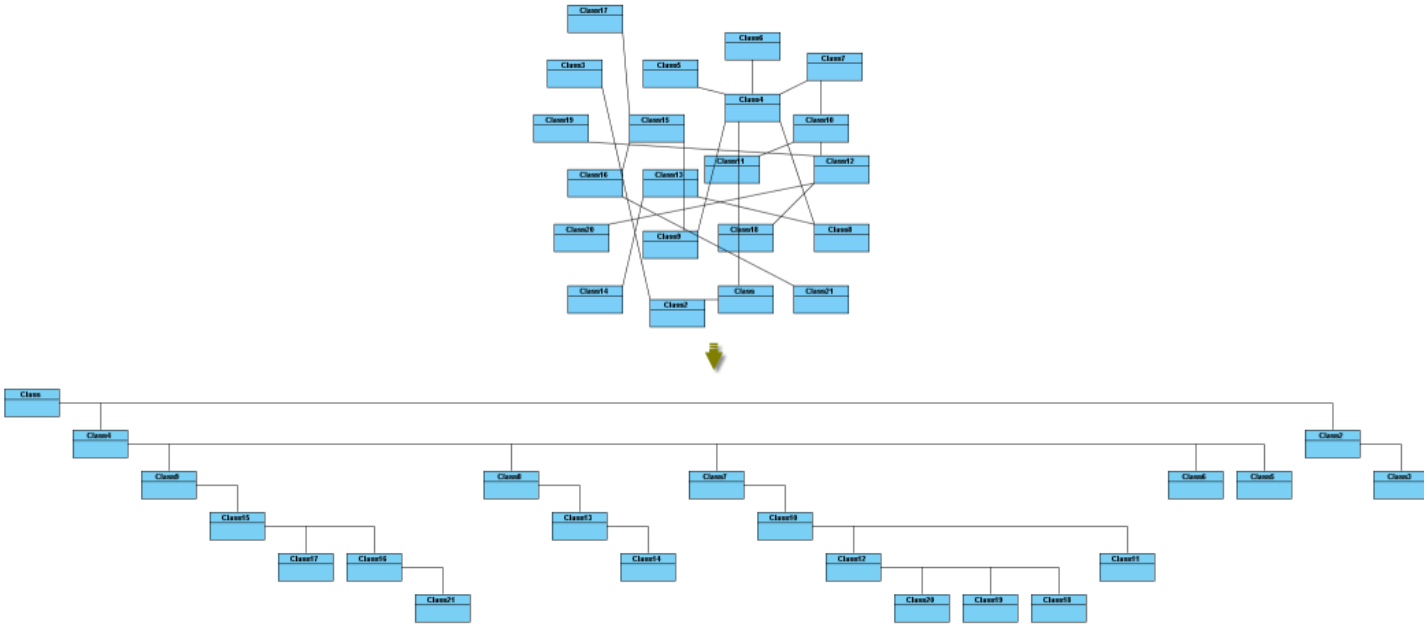
Horizontal Spacing: the horizontal spacing between the shapes.
Vertical Spacing: the vertical spacing between the shapes.
Min. Connector Length: the vertical distance of the connector segments.
Aspect Ratio: the relation of the tree width to the tree height.



Compact Tree Layout setting

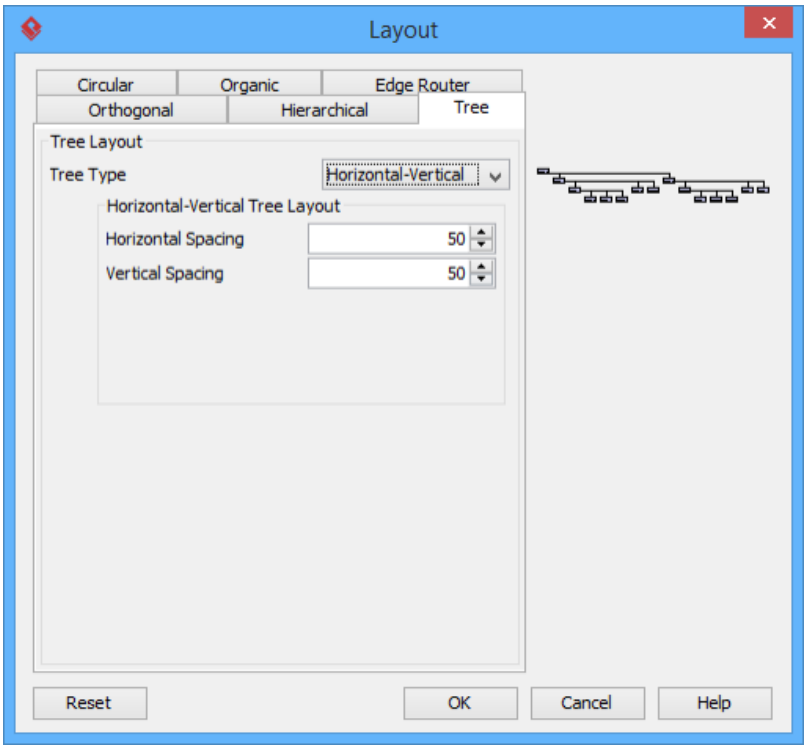
Horizontal-Vertical tree layout

Horizontal-Vertical Tree Layout is one of the tree layouts in Visual Paradigm which arranges shapes in a tree structure horizontally and vertically.



Horizontal-Vertical Tree Layout

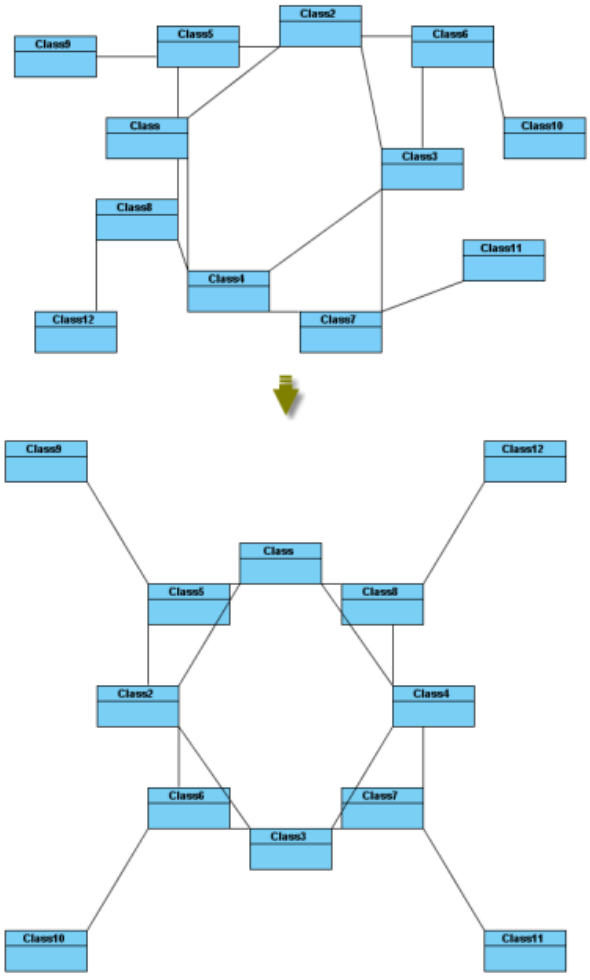
Horizontal Spacing: the horizontal spacing between the shapes.
Vertical Spacing: the vertical spacing between the shapes.



Horizontal-Vertical Tree Layout setting

BBC compact circular layout

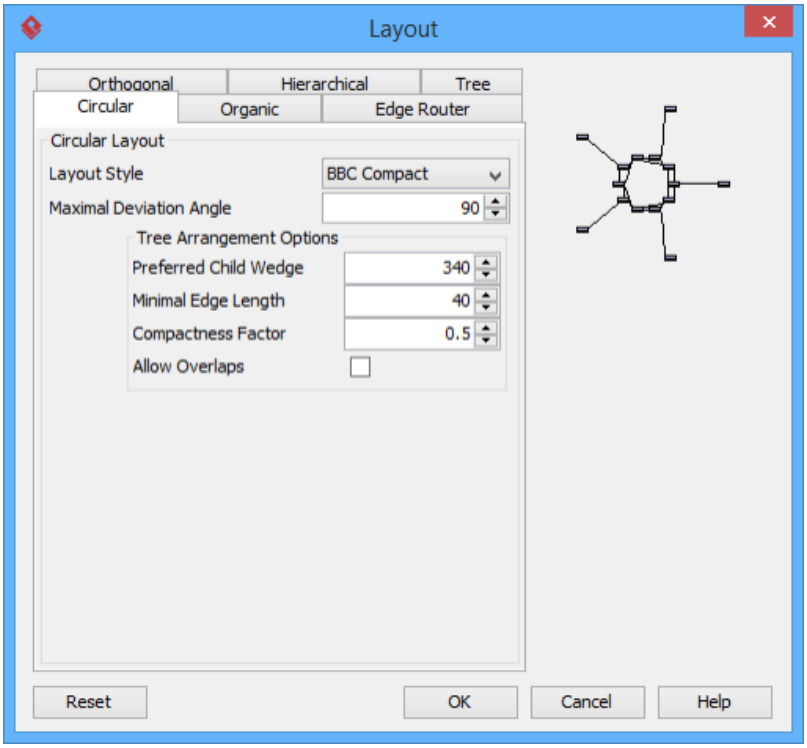
BBC Compact Circular Layout is one of the circular layouts in Visual Paradigm which arranges shapes in a radial tree structure. The detected group is laid out on the separate circles. It is the best way for user to arrange shapes that belong to more than one group with a ring structure.



BBC Compact Circular Layout

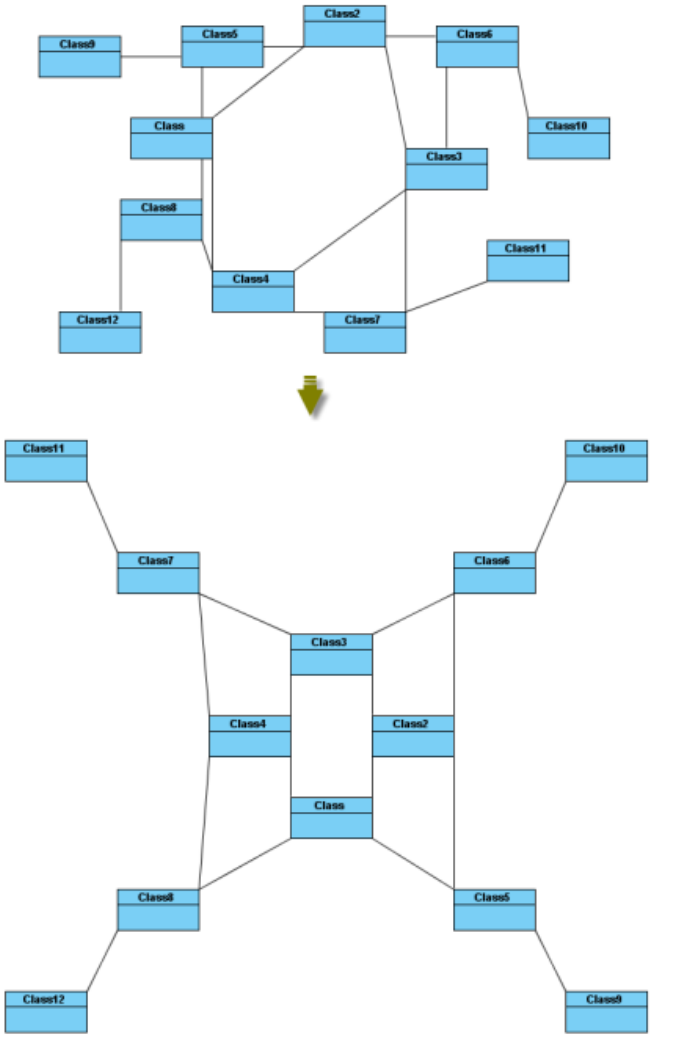
Maximal Deviation Angle: the maximal angle of deviation.
Preferred Child Wedge: the angle at which the child node will be placed around its parent node.

Minimal Edge Length: the minimal distance between the shapes.
Compactness Factor: the parameter that affects the length of connector. The smaller the compactness factor, the length of connectors will be shorter and the layout will be more compact.
Allow Overlaps: whether the shape can be overlapped.



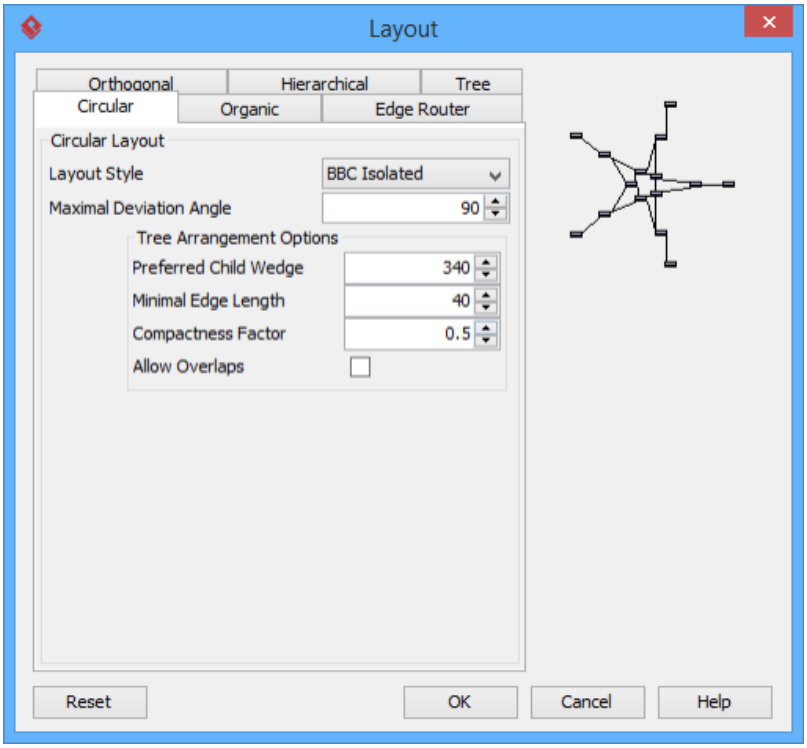
BBC Compact Circular Layout setting

BBC isolated circular layout
BBC Isolated Circular Layout is one of the circular layouts in Visual Paradigm which arranges shapes into many isolated ring structures. It is the best way for users to arrange shapes that belong to one group with ring structure.



BBC Isolated Circular Layout

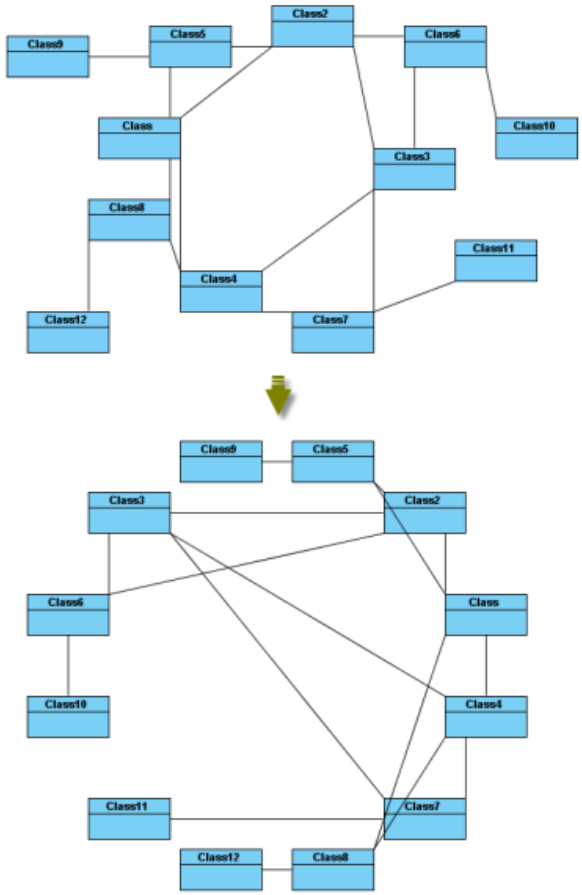
Maximal Deviation Angle: the maximal angle of deviation.
Preferred Child Wedge: the angle at which the child node will be placed around its parent node.
Minimal Edge Length: the minimal distance between the shapes.
Compactness Factor: the parameter that affects the length of connector. The smaller the compactness factor, the length of connectors will be shorter and the layout will be more compact.
Allow Overlaps: whether the shape can be overlapped.



BBC Isolated Circular Layout setting

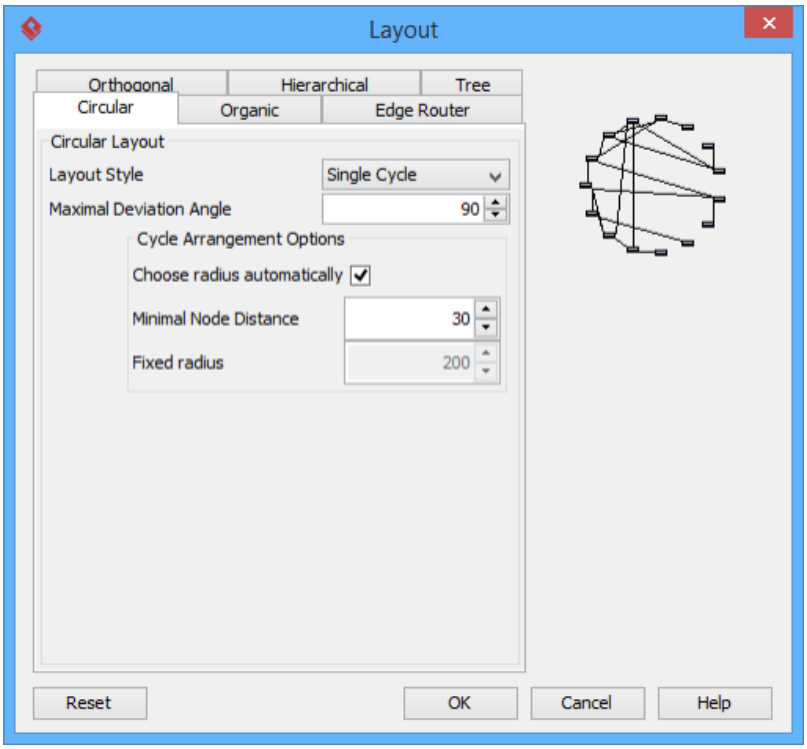
Single cycle circular layout

Single Cycle Layout is one of the circular layouts in Visual Paradigm which arranges shapes in circular structure in single circle.



Single Cycle Circular Layout

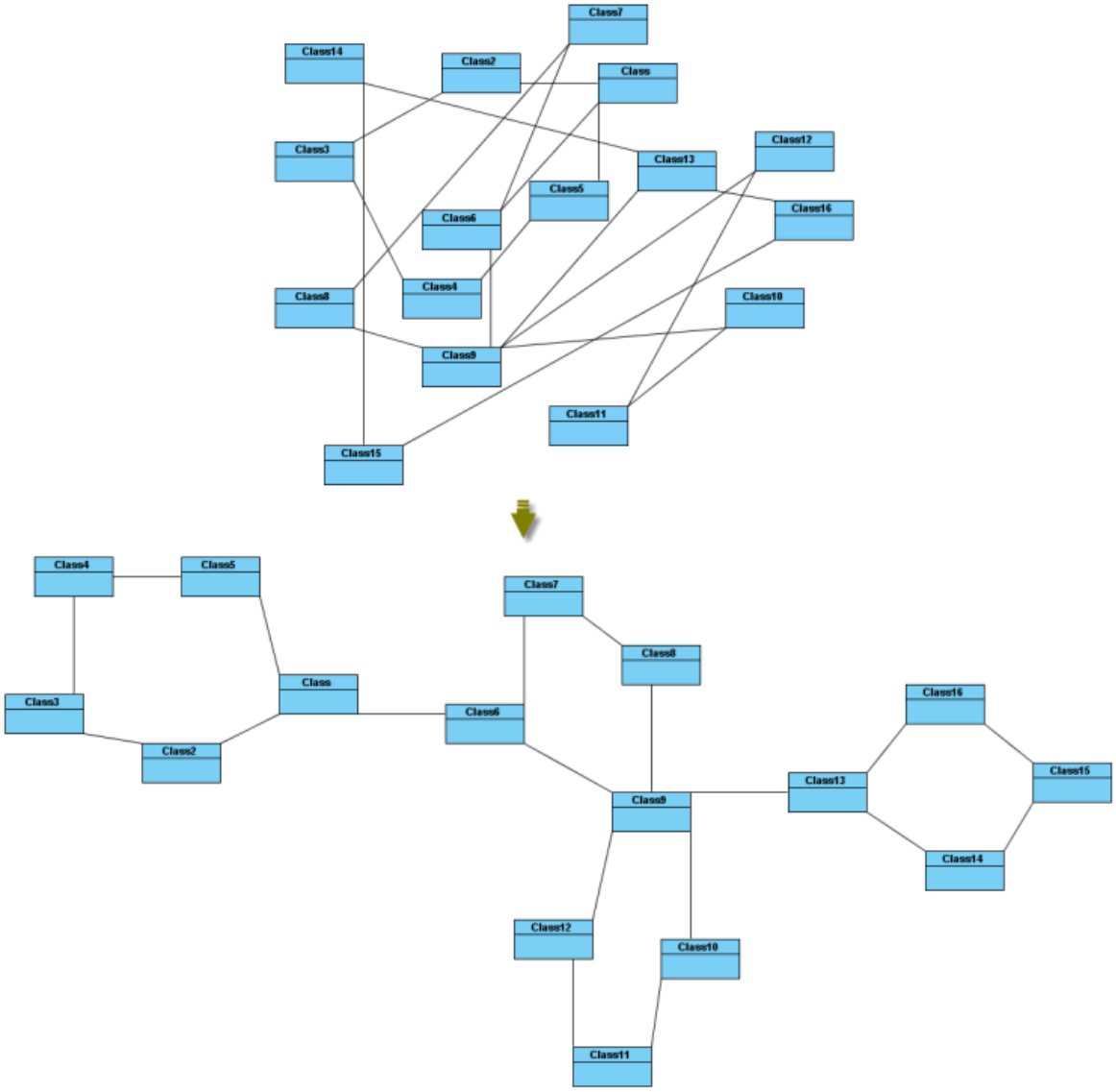
Choose radius automatically: determine the radius of circular structure automatically or manually.
Minimal Node Distance: the minimal distance between the nodes.
Fixed radius: the radius of circular structure.



Single Cycle Circular Layout setting

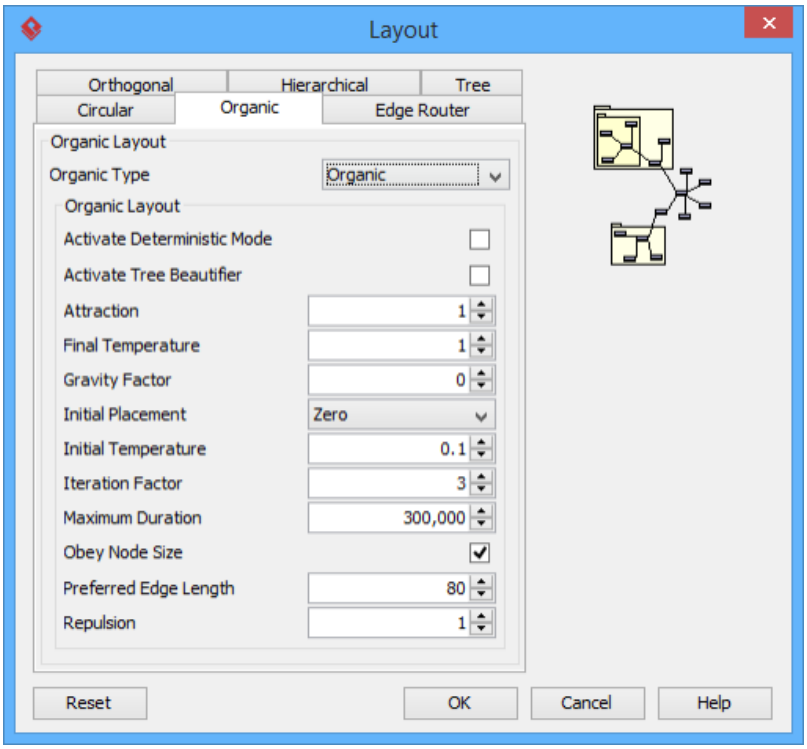
Organic layout

Organic Layout is one of the organic layouts in Visual Paradigm which arranges shapes in a star or ring structure. It is the best way for users to arrange the shapes that have highly connectivity relationship.



Organic Layout

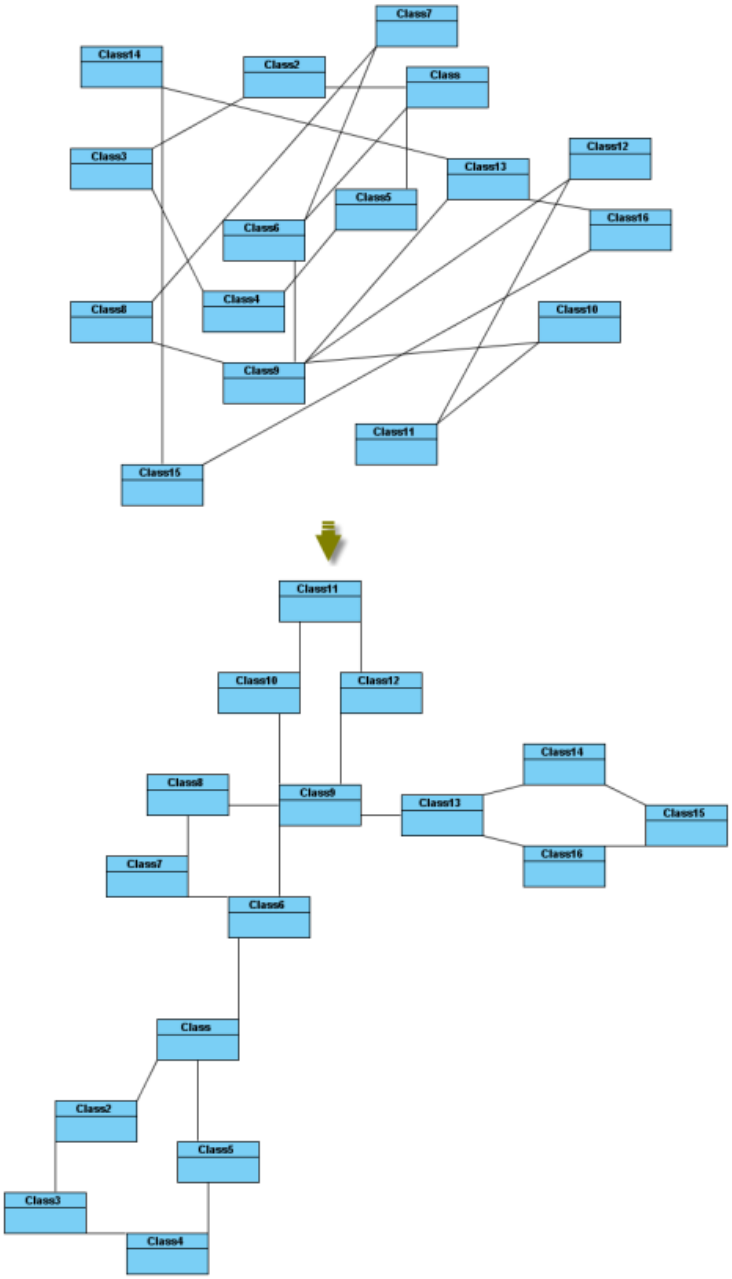
Activate Deterministic Mode: whether the layouter is in deterministic mode.
Activate Tree Beautifier: whether or not to activate the subtree beautifier.
Attraction: the degree of the attraction between shapes.
Final Temperature: the factor that affects the distance between shapes.
Gravity Factor: the factor that affects the distance between shapes and the center.
Initial Placement: the initial value of placement.
Initial Temperature: the initial value of temperature.
Iteration Factor: the degree of iteration.
Maximum Duration: the maximum degree of duration.
Obey Node Size: the size of obey shapes.
Preferred Edge Length: the preferred length between the nodes.
Repulsion: the factor that affects the distance between shapes which belong to the same ring or star structure.



Organic Layout setting

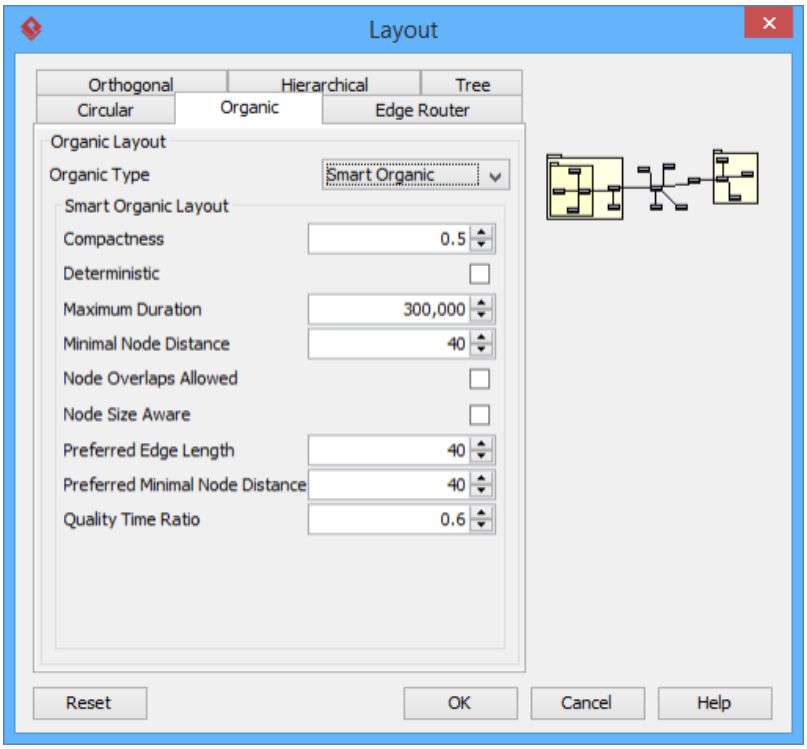
Smart organic layout

Smart Organic Layout is one of the organic layouts in Visual Paradigm which is a variant of the Organic Layout. It can set the ratio of the quality: producing time of layout and controls the compactness of layout.



Smart Organic Layout

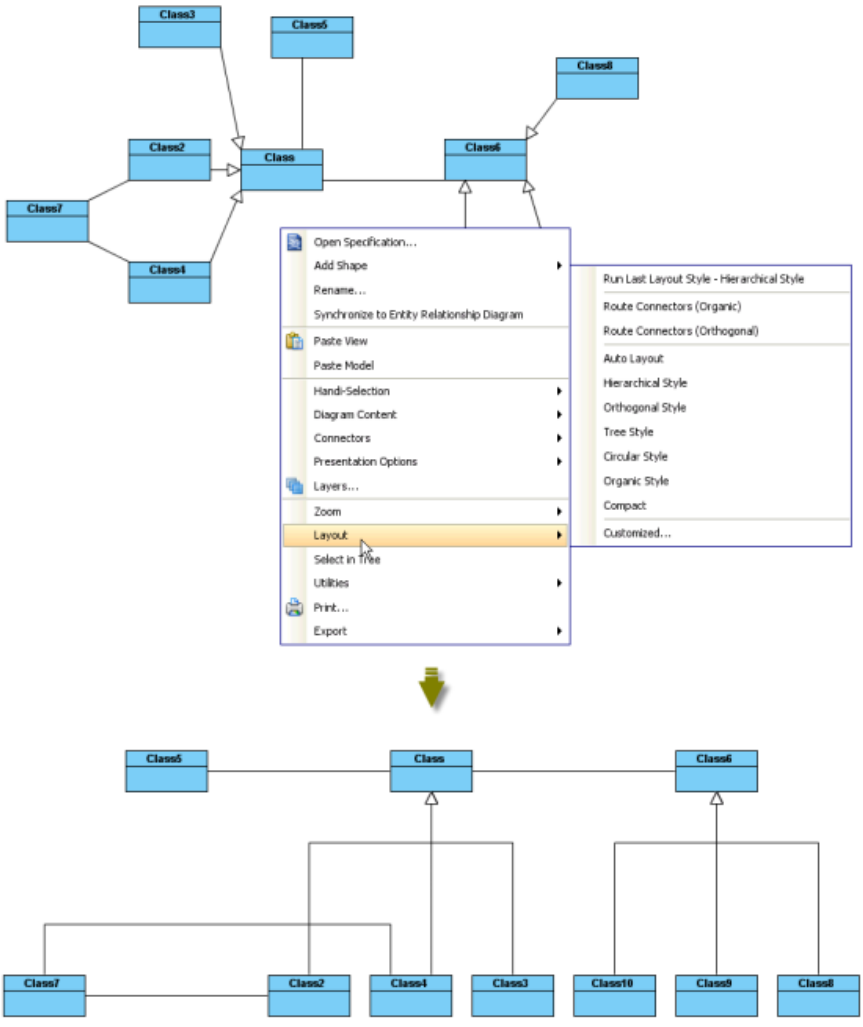
Compactness: the factor that sets less/more compact layout.
Deterministic: whether the layouter is in deterministic mode.
Minimal Node Distance: the minimal distance between nodes.
Node Overlaps Allowed: whether the node can be overlapped.
Node Size Aware: whether the node size can be aware.
Preferred Minimal Node Distance: the preferred minimal distance between the nodes.
Quality Time Ratio: the ratio of the quality of layout to the producing time of layout.



Organic Layout setting

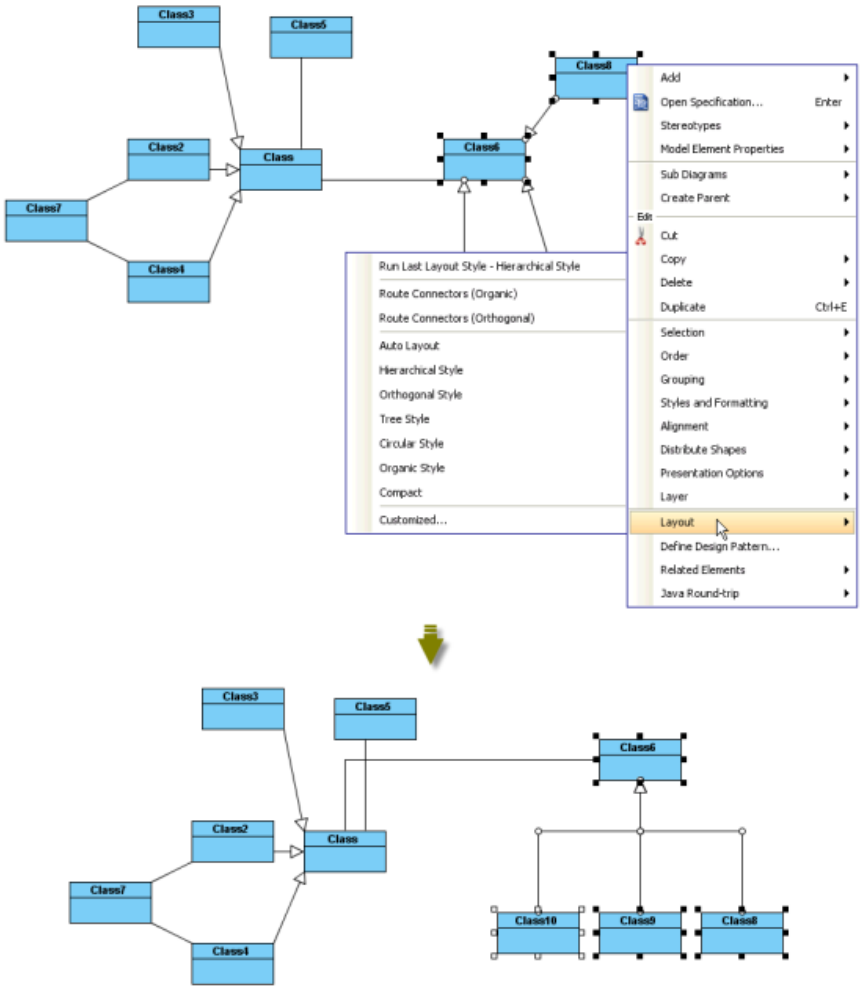
Automatic layout selected shapes

To layout all the shapes in the diagram, right-click on the diagram and select Layout from the pop-up menu.



Perform layout with all shapes of diagram

To layout the selected shapes, right-click on the selection and select **Layout** from the pop-up menu (make sure there are more than one diagram elements selected).



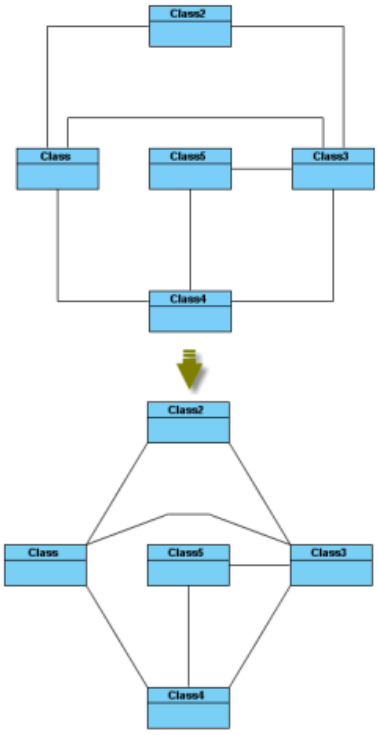
Perform layout with selected shapes

Automatic route connectors

There are 2 kinds of layouts which do not change the location of shapes but only change the connectors: **Organic Edge Route Layout** and **Orthogonal Edge Route Layout**

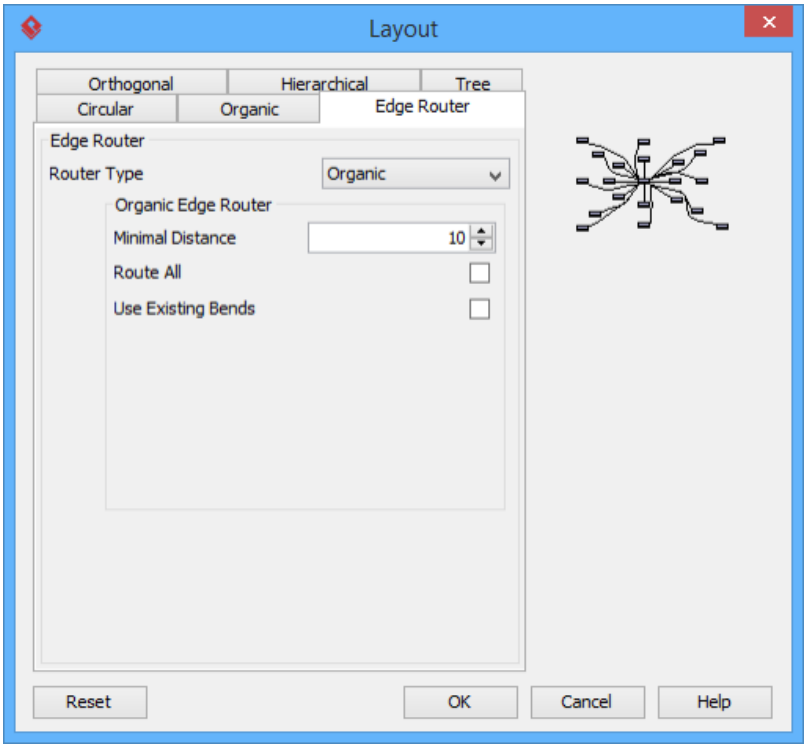
Organic edge route layout

Organic Edge Route Layout is one of the edge route layouts in Visual Paradigm which arranges the connectors without affecting the location of shapes. It can ensure that the shapes will not overlap and keep a specific minimal distance.



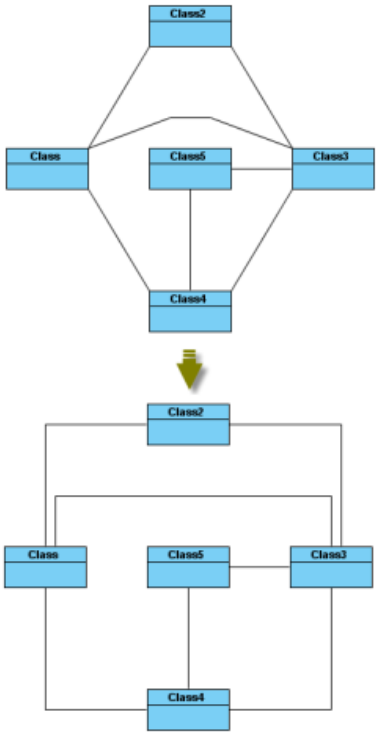
Organic Edge Route Layout

Minimal Distance: the minimal distance of the connectors.
Route All: whether all the connectors will be routed.
Use Existing Beans: whether using existing bends.



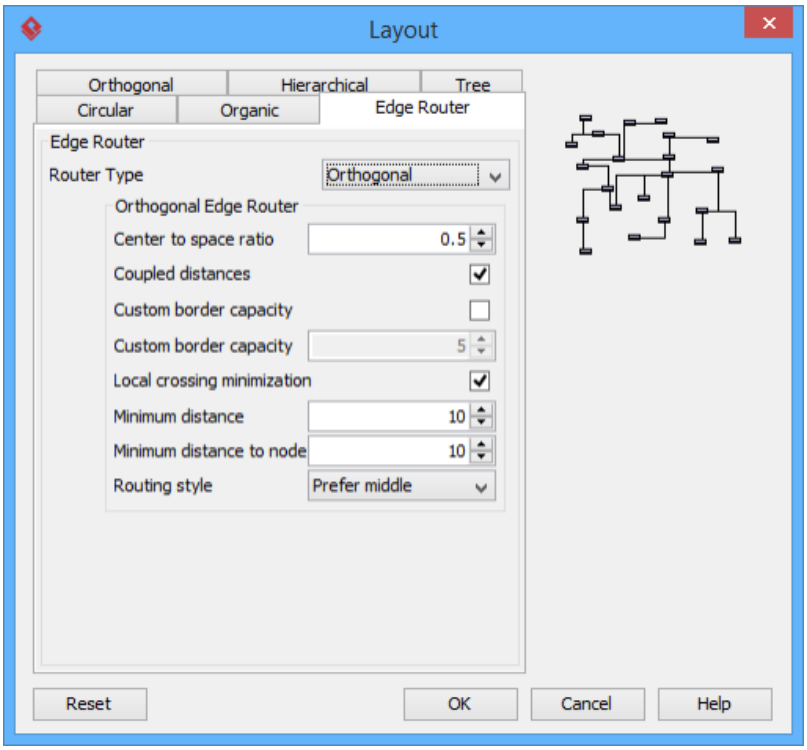
Organic Edge Route Layout setting

Orthogonal edge route layout
Route Connectors can arrange the connectors using vertical and horizontal line segments only. It is the best way for users to arrange the connectors that have complicated route.



Orthogonal Edge Route Layout

Center to space ratio: the ratio of center to the distance between center and nodes.
Coupled distances: the distance between coupled nodes.
Crossing cost: the cost of crossing connectors.
Custom border capacity: the capacity of the border.
Local crossing minimization: whether the local crossing of connectors will be minimized.
Minimum distance: the minimum distance of connectors.
Minimum distance to node: the minimum distance between the shapes.
Rerouting: whether the connector that has many crossings will be rerouted.
Routing style: the style of routing.



Orthogonal Edge Route Layout setting

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm

- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

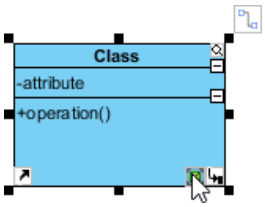
Fit shape size

In some cases, shapes are found oversized. For better presentation, you may need to resize them smaller. Fit size can help you to adjust shapes into the smallest size based on their content, such as the name of shape. The size of shapes can be fixed either manually or automatically.

NOTE: The size of shapes can be fixed automatically by right clicking on the diagram's background and checking **Diagram Content> Auto Fit Shapes Size**.

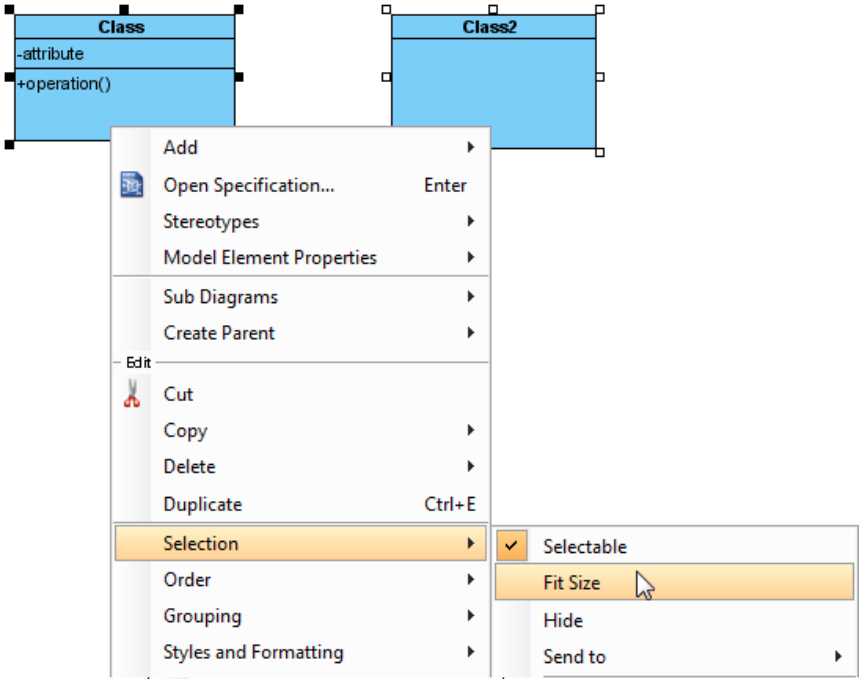
Fit selected shapes size

To adjust a shape size, move the mouse on a shape and fit size resource centric interface will be shown. Click **Fit Size**resource icon at the bottom of the shape.



Click Fit Size

To fit several shapes' size, select those shapes, right click on a selected shape and then select **Selection > Fit Size** from the pop-up menu.



Fit size for several shapes from the pop-up menu

Each shape will be adjusted to its fit size in accordance with its content, instead of fixing all selected shapes into the exactly same size.

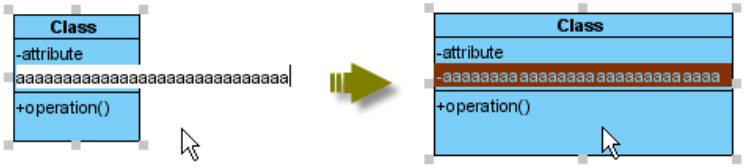
Shapes are fitted size

Check/uncheck automatic fit shape size mode

You can check/ uncheck the **Auto Fit Shapes Size** on diagram to make all the shapes on the diagram to be fitted size automatically. To do so, right click on the diagram's background, select **Diagram Content > Auto Fit Shapes Size** from the pop-up menu.

All the shapes are subsequently fitted size and they will become non-sizable.

If the content of shape is changed, the shape itself will be resized automatically.



Class shape is resized automatically after new attribute added

You can also check **Auto Fit Size** for future usage.

1. Select **Windows > Project Options...** from the toolbar to open the **Project Options** window.
2. In the **Project Options** window, choose the **Diagramming** category, select the **Shape** tab and check **Auto fit size (diagram-based)**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram element selection

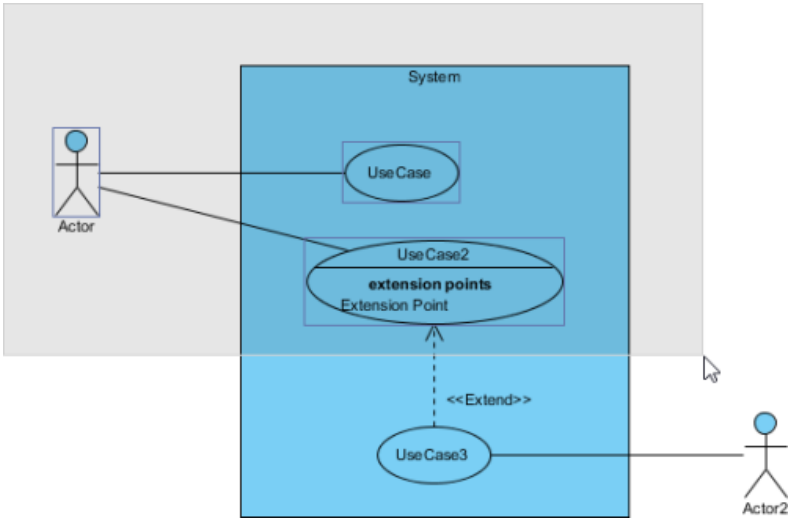
When a number of diagram elements need to be selected simultaneously, diagram element selection supports this purpose. You can either click directly with hot keys or select a range of selection with the mouse. A specific type of diagram element(s) on a diagram can be selected as well.

Selecting multiple shapes

Multiple shapes can be selected by either selecting a range of shapes with the mouse on diagram or clicking shapes with pressing hot keys.

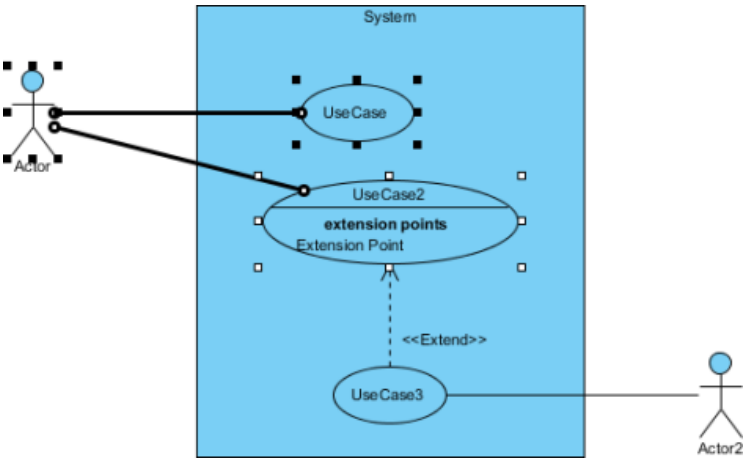
Selecting a range of shapes with the mouse

1. For selecting multiple shapes, drag them from corner to corner diagonally with the mouse.



Select multiple shapes with the mouse

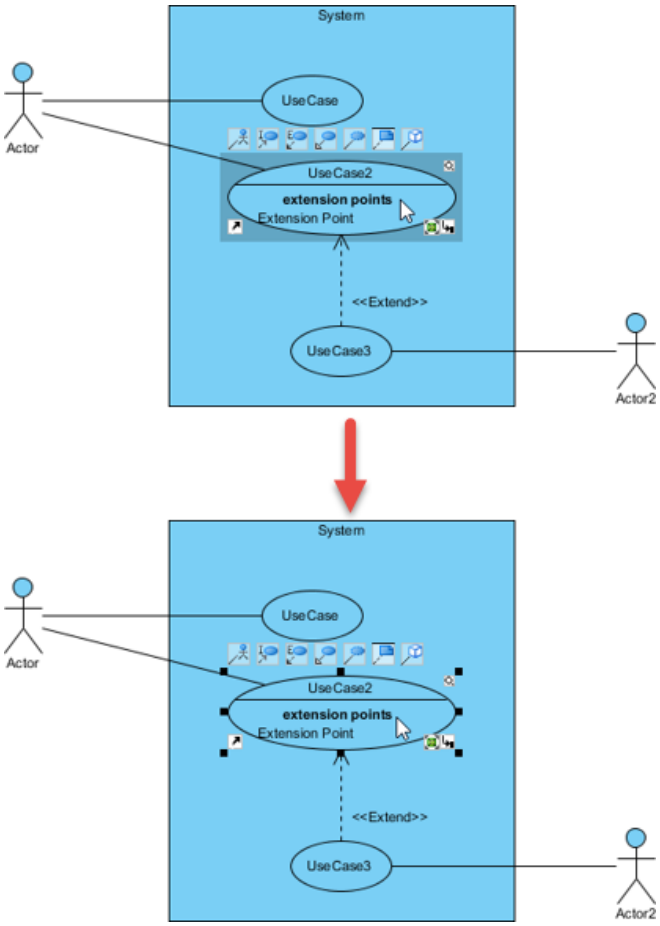
2. After releasing the mouse, those shapes will be selected.



Shapes are selected

Clicking with pressing ctrl/Shift key

Click a shape in advance and then click other shapes with pressing **Ctrl** or **Shift** key. As a result, those shapes will be selected.

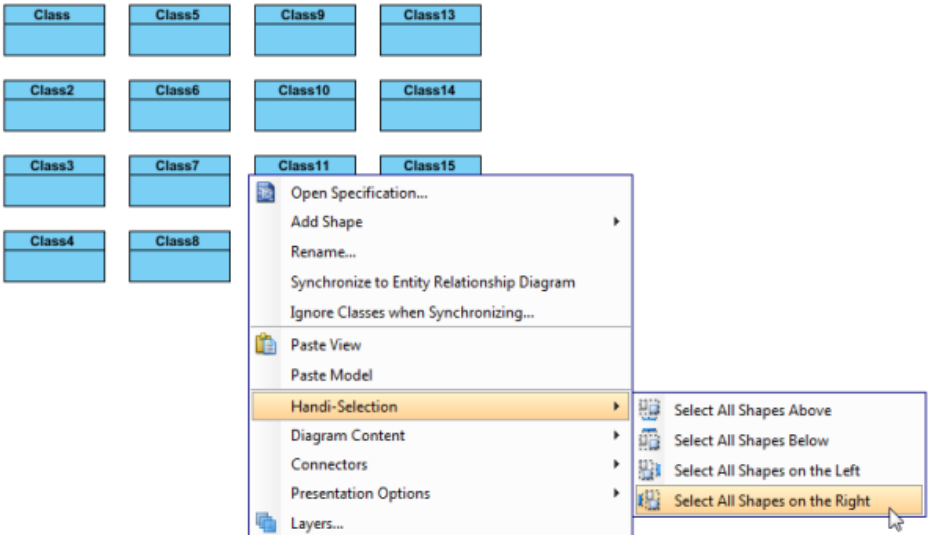


*Click shapes with pressing **Ctrl** or **Shift** key*

Handi-Selection

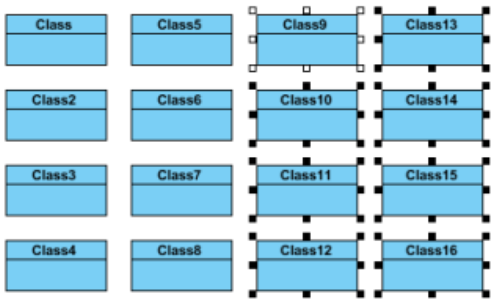
In some cases, the shapes are too complicated and in more serious occasion, the whole diagram is extremely enormous that neither selecting a range of shapes with the mouse nor clicking shapes with pressing **Ctrl** or **Shift** key are the most suitable application. It is hard to drag the mouse on the large diagram, or is troublesome to click on many shapes. Using **Handi-Selection** is probably the best choice for you in this situation.

1. Right click on the diagram's background where is in the vicinity of those shapes you are going to select, select **Handi-Selection** and then select a scope for selecting shapes (i.e. above/ below/ left/ right) from the pop-up menu.



Select all shapes on the right from the pop-up menu

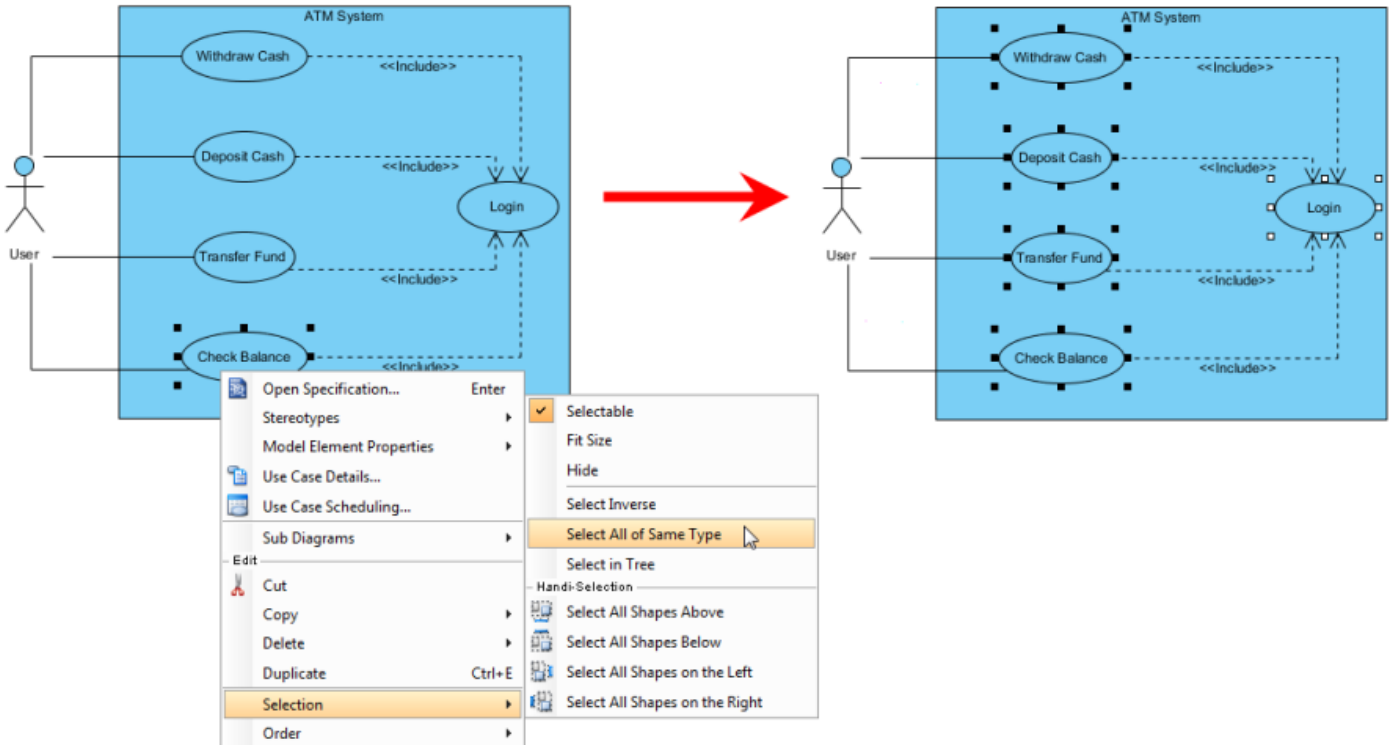
2. As a result, all the shapes of the particular scope will be selected.



All shapes on the right are selected

Selecting same type of shapes

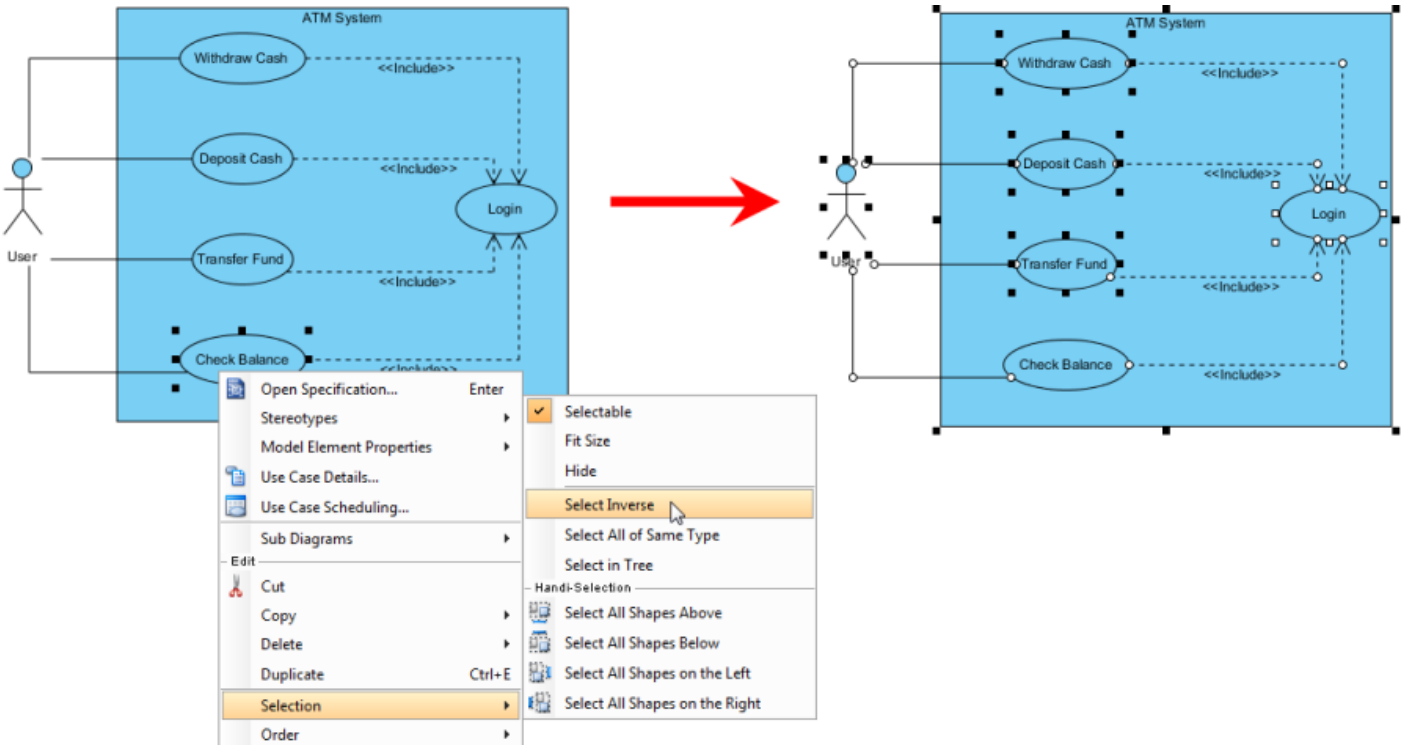
When you want to select a few shapes of the same types on the diagram, right click on a shape and select **Selection > Select All of Same Type** from the pop-up menu. As a result, other shapes of same type as the shape you selected previously will be selected.



All shapes of same type are selected

Inverse selection

Shapes can be selected inversely. Right click on a shape that you don't want to be selected and select **Selection > Select Inverse** from the pop-up menu. As a result, all shapes will be selected except the shape you right clicked on previously.



Selection is inversed

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

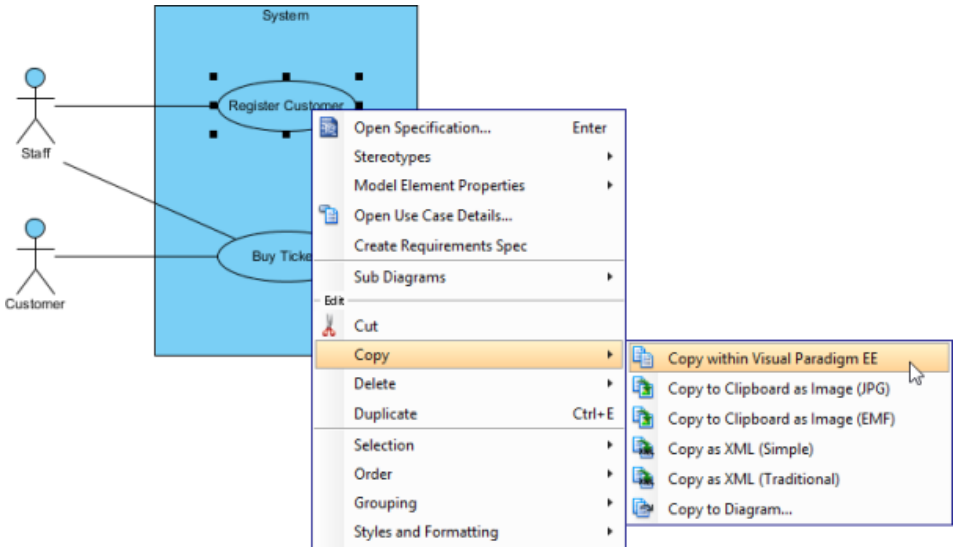
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Copy and paste

You can create a view of a model element by copying a view and pasting as view, while pasting as model creates a new model from the copied one.

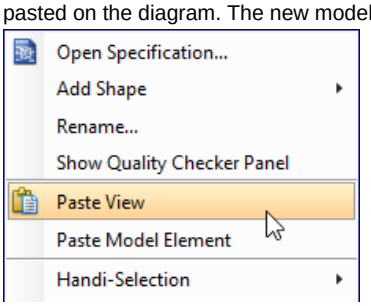
Copying within Visual Paradigm

1. Right click on the selected shape(s), select **Copy > Copy within Visual Paradigm** from the pop-up menu.



Copy selected shapes with Visual Paradigm

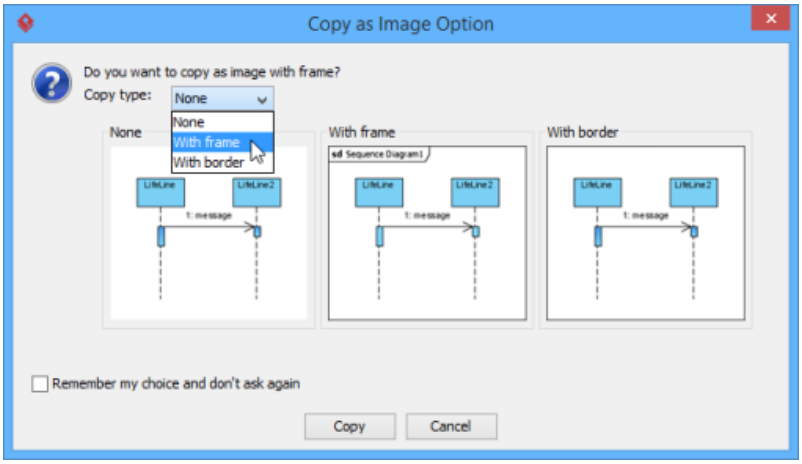
2. After you switch to the destination diagram, right click on the diagram background and select either **Paste View** or **Paste Model** from the pop-up menu. The feature of **Paste view** refers to present the same model element in another view under a new context. The shape is pasted without creating model; while the feature of **Paste Model** refers to duplicate the shape and present it in a new view. The model will be copied and pasted on the diagram. The new model will be named with appending a sequential number.



Paste the selected shape

Copying to clipboard as image (JPG)

1. Right click on the selected shape(s) and select **Copy > Copy to Clipboard as Image (JPG)** from the pop-up menu.
2. When the **Copy as Image Option** dialog box pops out, select an option from the drop-down menu of **Copy type**. Click **Copy** button to proceed.



*Select an option from the drop-down menu of **Copy type***

3. You should select a destination document (e.g. MS Word) for pasting the copied shape(s) to do further description. After you switch to the destination document, right click on the desired place and select **Paste** from the pop-up menu.
4. As a result, your selected shape(s) will be pasted on the destination document.

Copying to clipboard as image (EMF)

1. Right click on the selected shape(s) and select **Copy > Copy to Clipboard as Image (EMF)** from the pop-up menu.
2. When the **Copy as Image Option** dialog box pops out, select an option from the drop-down menu of **Copy type**. Click **Copy** button to proceed.
3. EMF is a kind of scalable image which can be pasted on a document for further description. After you switch to the destination document, right click on a desired place and select **Paste** from the pop-up menu.
4. As a result, your selected shape(s) will be pasted on the destination document.

Copying as XML

You can convert selected shapes into XML which contain the data of selected shapes in XML format. The XML data can then be imported into another project.

1. Right click on the selected shapes and select **Copy > Copy as XML** from the pop-up menu.
2. Open a text editor and create a new text file. Paste the XML there and save it as an XML file.
3. After that, the file can be imported to another project by selecting **File > Import > XML**.
4. When **Import XML** dialog box pops out, select the xml file path. Finally, click **OK** button to proceed.

Copying to Diagram

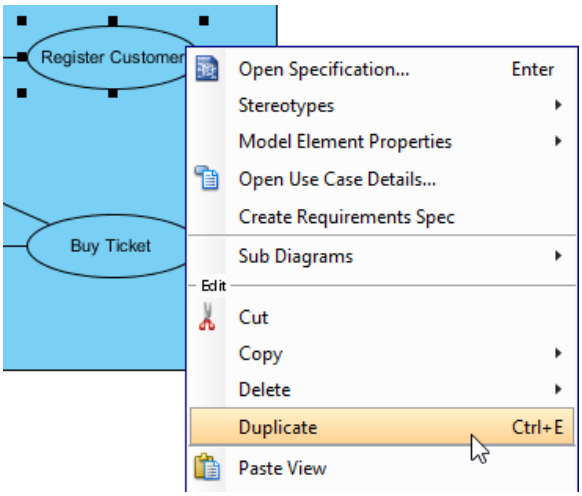
You can also copy the selected shape(s) to either a new diagram or an existing diagram. To copy to a diagram:

1. Right-click on the selected shapes, select **Copy > Copy to Diagram...** from the pop-up menu.
2. When the **Copy to Diagram** dialog box pops out, check **Create new diagram** if you want to paste to a new diagram or check **Select an existing diagram** if you want to paste to an existing diagram.
3. Click **Copy to** button. As a result, the selected shape(s) will be duplicated on the selected diagram.

Duplicating

With the feature of duplicate, shapes can be duplicated on the same diagram instantly.

1. Right click on the selected shape(s) and select **Duplicate** from the pop-up menu.



Duplicate a selected shape

2. As a result, the selected shape(s) will be duplicated.

The shape is duplicated

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Alignment guide

Alignment guide acts as a guide to help aligning shapes perfectly in the shortest period of time.

It is a dotted line that appears when you are moving the shape or resizing the shape within more than one shapes.

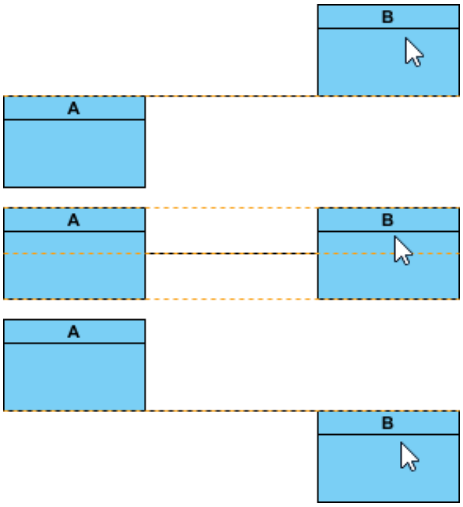
With the use of alignment guide, you can organize the diagram neatly with only little effort.

Arrange shapes in orderly arrangement

After dragging one shape from another shape, you will see a dotted line (the alignment guide) between the two shapes when you move either shape.

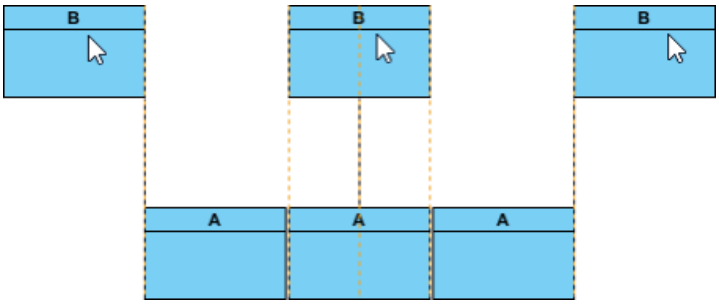
The Alignment Guide is here to help you to align the two shapes.

Take the following diagram as an example. If you move shape B vertically, you can see horizontal alignment guides appears on top, in the middle or at the bottom. You can choose to follow one of them depends on where you want to position shape B.



Aligning classes horizontally

Similarly, when you move shape B horizontally, you can see vertical alignment guides appears on the left, in the middle and at the right.



Aligning classes vertically

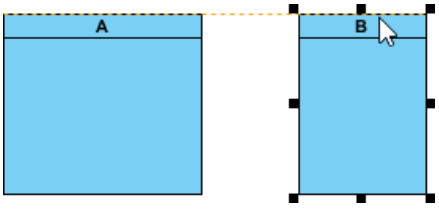
With the help of alignment guide, you can produce a neat diagram with ease.

Other than moving the shape, alignment guide also appears when you resize the shape.

Let's take a look at the diagram below. If you want to resize shape B to produce equal height as shape A. The alignment guide can guide you to achieve this.

Evenly distribute the spacing among shapes

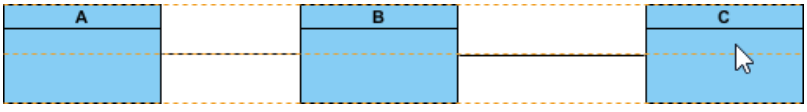
If you want the spacing among shape B and C equals to shape A and B, simply move shape C parallel to shape B, an additional dotted line at the bottom of the diagram (as pointed by red arrows) represents equal spacing among shape A and B as well as shape B and C.



Aligning classes during resize

Therefore, this additional dotted line acts as a hint that help you to create equal spacing efficiently and precisely.

On the contrary, the additional dotted line will not appear if the spacing are not equal.



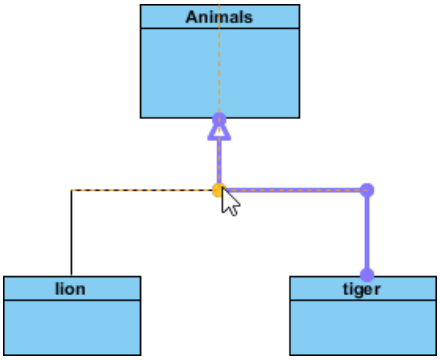
Aligning classes with equal spacing

Layout generalization hierarchy with ease

Alignment guide is very useful in producing a neat layout of generalization hierarchy as well.

While it's nice to present super-class and its sub-classes in hierarchical form, it's uneasy to route the generalization connectors perfectly to make the hierarchy looks neat and tidy. Sometimes, the connectors are either partially overlapped or entirely detached from each other. All these affect the harmony of the design and may even reduce readability.

Alignment guide is here to help. By dragging on a generalization connector, you can make its turning point snap to a position that aligns with another generalization horizontally. Therefore, you can produce a neat hierarchy effortlessly with the aid of alignment guide.

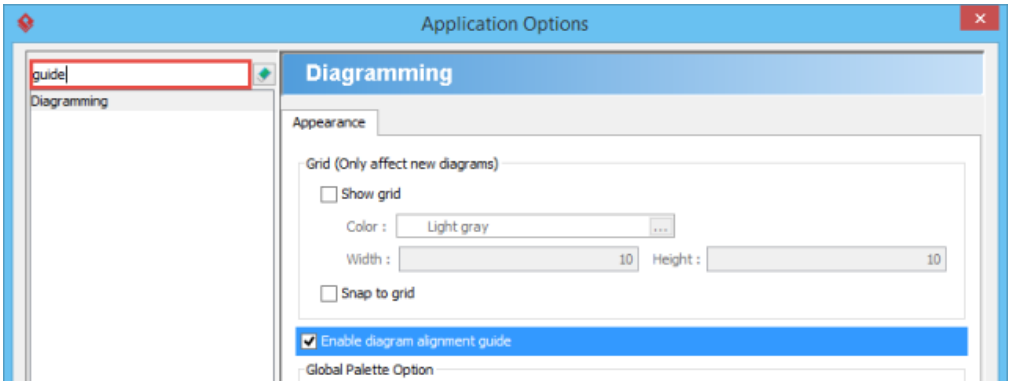


Aligning generalization hierarchy

Turn off the Alignment guide

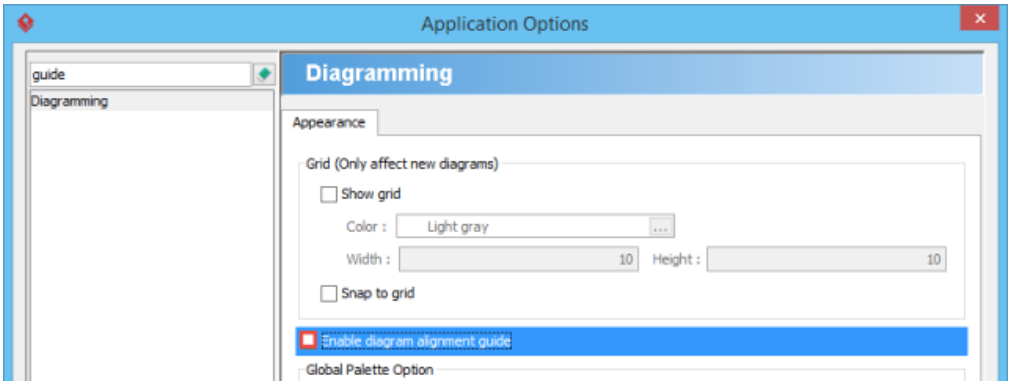
In case users want to disable the Alignment guide function, here are some simple steps:

1. Select **Windows > Application Options...** from the toolbar.
2. Enter *guide* in the search field of the **Application Options** window to locate the option.



Entered 'guide' in search field

3. In the **Appearance** tab of the **Diagramming** page, uncheck **Enable diagram alignment guide**.



Uncheck "Enable diagram alignment guide."

4. Click **OK** to confirm the setting.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

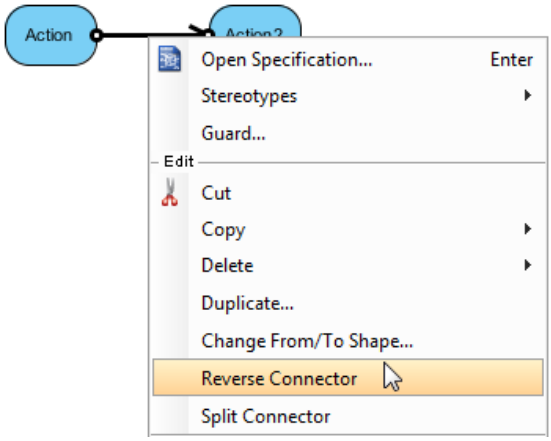
Reverse connector direction

The flow between shapes is represented by connectors, for example, a sequence message between two lifelines of *Student* and *StudentController* which represents the call from *Student* to *StudentController*. If the flow is created mistakenly or the flows need reverting due to an updated data, the flows can be fixed by reverting connectors.

The function of reverse connector is not only for reverting the connector's direction but also for repositioning the information contained by the end of connection. For connectors like association, each end contains specific information like multiplicity, role name, visibility, etc. Reverting connector will also swap the information.

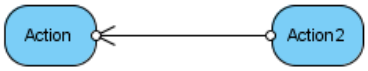
Reverse connector direction

1. Right click on the connector between two shapes and select **Reverse Connector** from the pop-up menu.



*Select **Reverse Connector** from the pop-up menu*

2. As a result, the flow of connector is reversed.



Connector is reversed

> **NOTE:** Connectors, such as create message in sequence diagram, are not reversible.

Related Resources

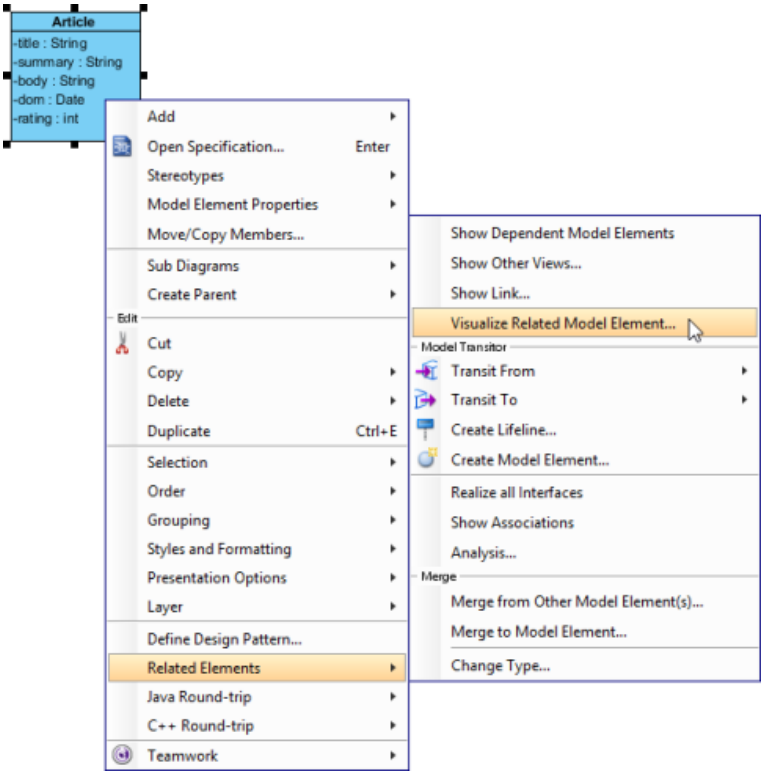
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Visualize related model elements

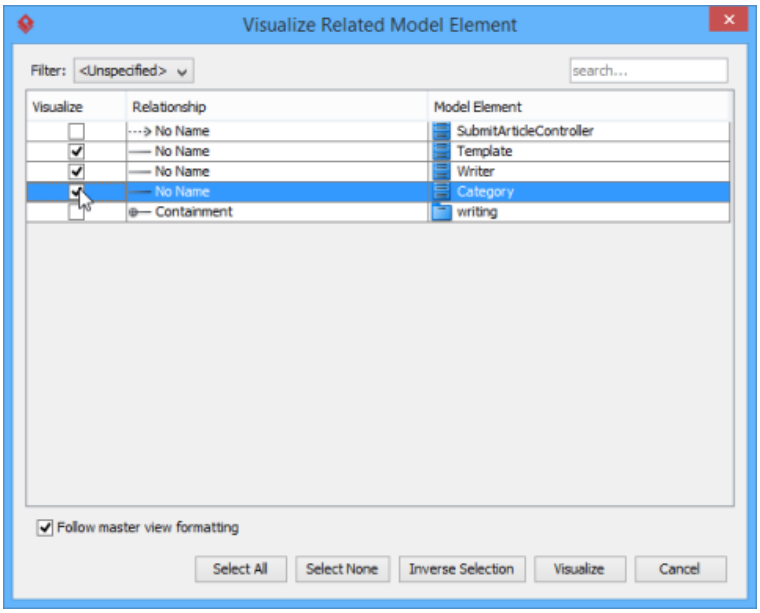
Sometimes, several model elements that related to the current model element(s) are hidden for various reasons. These related model elements, in fact, can be revealed through the feature of visualize related model element. With this feature, the relationship between model elements can be viewed thoroughly.

1. Right click on a model element and select **Related Elements > Visualize Related Model Element...** from the pop-up menu.



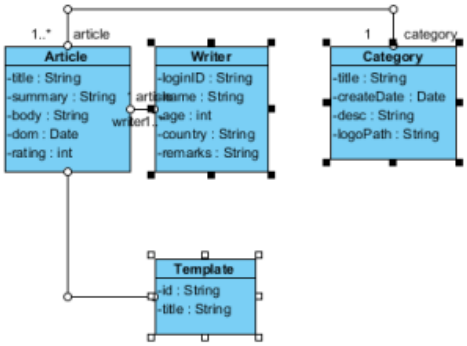
*Select **Visualize Related Model Element...***

2. In the **Visualize Related Model Element** window, check the related element(s) you want to be shown with the corresponding relationship on the diagram in **Visualize**. Click **Visualize** to proceed with.



To visualize three relationships

As a result, the related models with connectors are shown on the diagram.



Related model elements are visualized

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

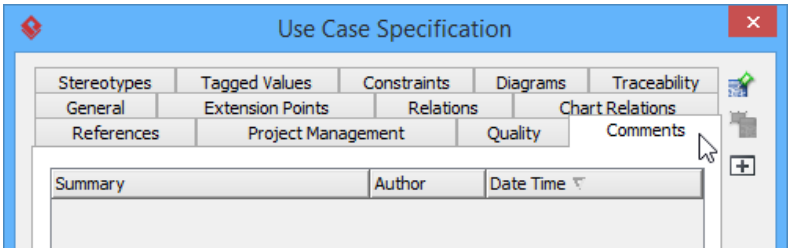
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Adding comments

Visual Paradigm supports comments on model elements. Since comments are usually used to record the progress and status of model elements, they are regarded as a textual annotation for model elements.

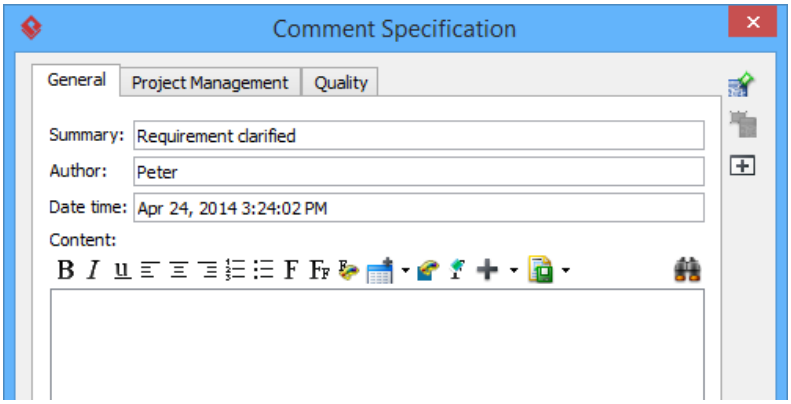
Adding comment to model element

1. For adding comment in a particular shape, right click on the shape and select **Open Specification...** from the pop-up menu.
2. In **Specification** window, select the **Comments** tab.



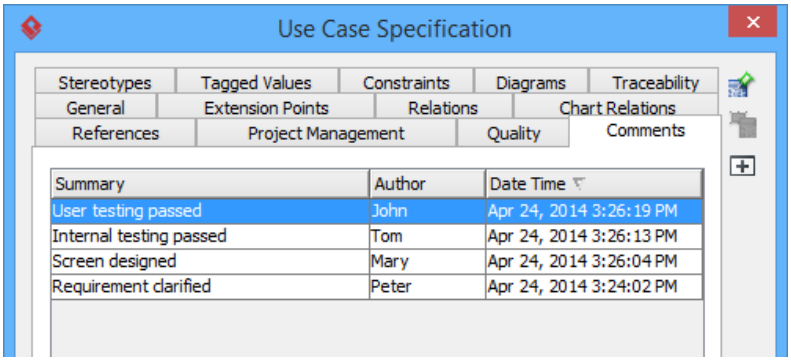
The Comments tab

3. Click **Add...** to create a comment.
4. In **Comment Specification** window, enter summary, author, date time and content. Click **OK** to confirm editing.



Entering a comment

As a result, the comment you entered previously is shown on **Specification** window.



Comments added

Managing comment of model element

1. To modify comment in a particular shape, right click on the shape and select **Open Specification...** from the pop-up menu.
2. In **Specification** window, select the **Comments** tab
3. Select a summary comment and click **Open Specification...**.
4. In the **Comment Specification** window, edit the comment and click **OK** to confirm changing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

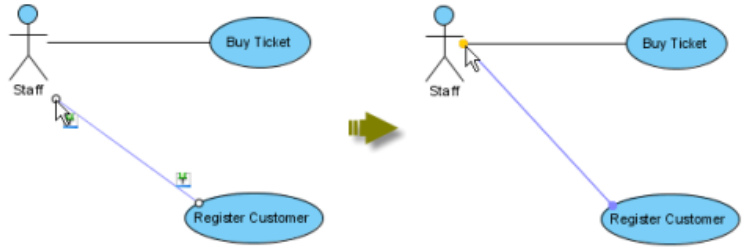
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Pinning connector ends

Connectors can be either pinned temporarily or pinned permanently. Connector ends help you to point out a specific position of shapes with pin.

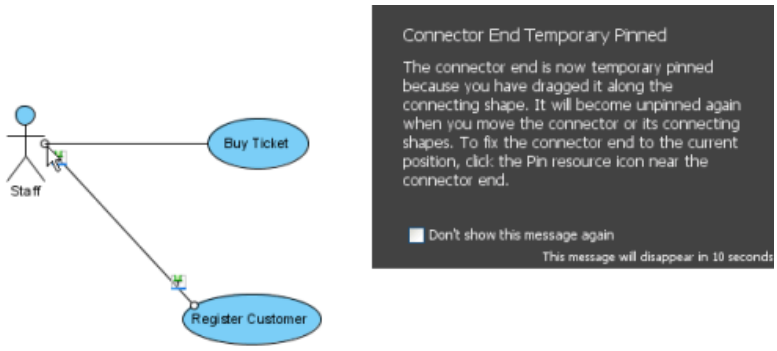
Adjusting connector ends temporarily

1. Connectors can be joined at the same point of a shape on the diagram. To do so, drag one end of a connector to the shape.



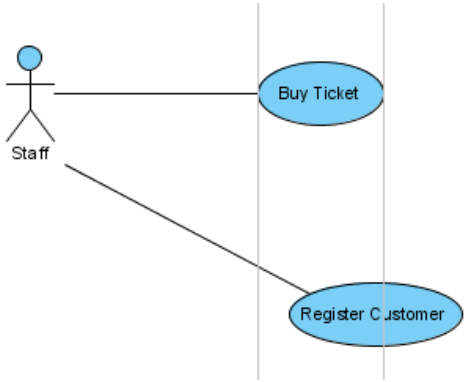
Drag one end of a connector to a shape

2. The connector is temporarily pinned. A dialog box will be shown on the top right corner of diagram to instruct you how to pin the connector.



Connector is temporarily pinned

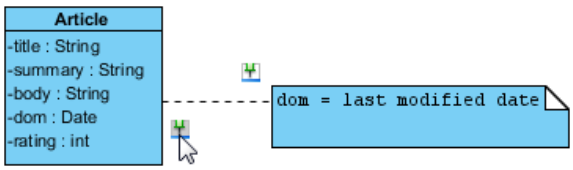
3. Since the connector that links from shape and to shape together is temporarily pinned, either from shape or to shape is moved, the connector between them will be unpinned.



Connector is unpinned

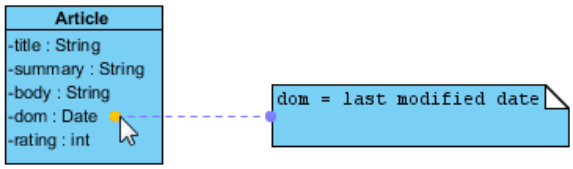
Pinning connector ends

1. Move the mouse over a connector and press its resource icon **Pin**.



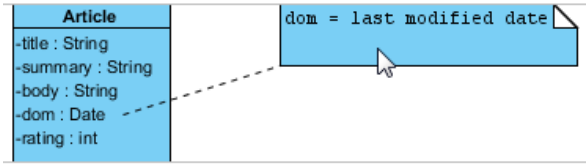
*Press resource icon **Pin***

2. Drag one end of connector to point out a specific position. Note that no dialog box of temporary pin will be shown this time.



Drag one end of connector to point out a specific position

3. Moving either from shape or to shape will not unpin the connector.



Connector is still pinned

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

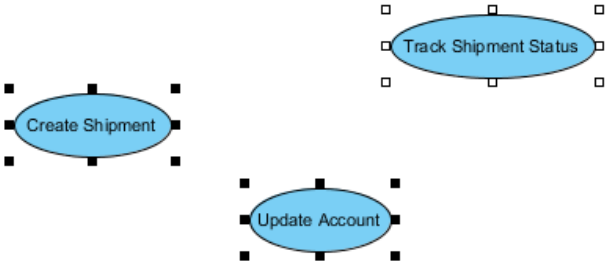
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Align and distribute diagram elements

Visual Paradigm supports two types of positioning features: alignment and distribution which allow the positioning of selected shapes in accordance with the alignment/ distribution option through the toolbar or grouping resource icons. Alignment refers to the edges and the centers of selected shapes are aligned to each other while distribution refers to selected shapes are distributed in the same direction based on their centers or edges.

Aligning diagram elements

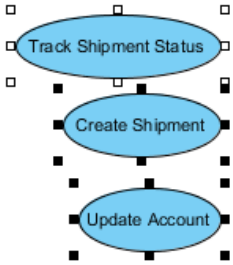
Select a few model elements with the mouse on the diagram pane before executing alignment.



Selected shapes

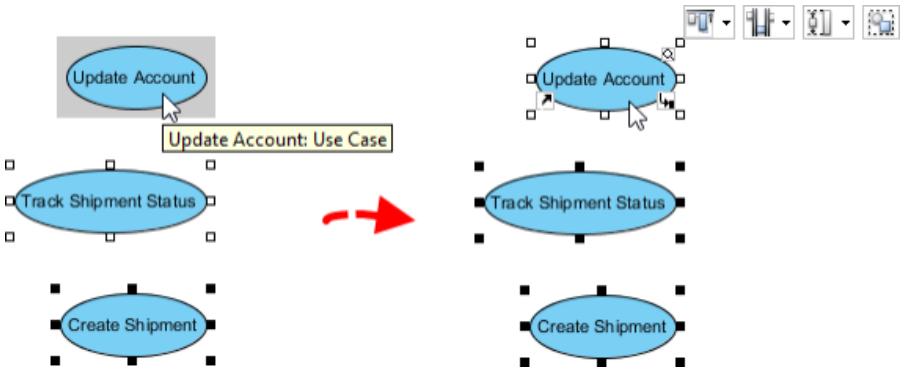
Through Toolbar

1. Select **Diagram > Alignment** and then an alignment option from the toolbar.
2. As a result, the alignment of all selected shapes is based on the last selected shape. Note that the last selected shape refers to the shape with no-filled selector.



Shapes are aligned

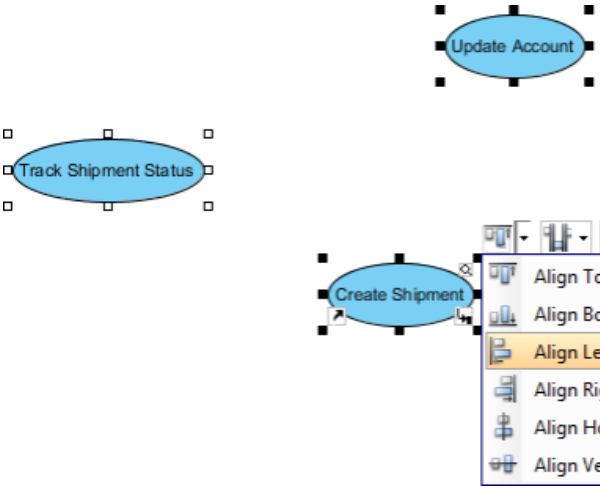
3. For turning the non-selected shape into the last selected shape, click a shape with pressing **Ctrl** key.



Change the last selected shape

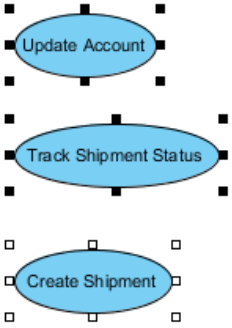
Through grouping resource icons

1. When move the mouse over one of the selected shapes, grouping resource icons will be shown.
2. Select an alignment option from the drop-down menu of **Align Top** on the grouping resource icons.



*Select **Align Left** on resources*

3. As a result, all selected shapes are aligned in accordance with the last selected shape.



Shapes are aligned

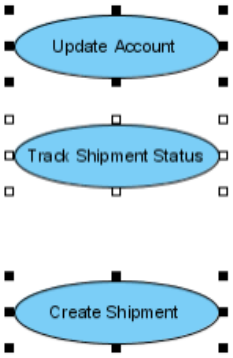
Setting model elements same width and height

Besides aligning the shapes, shapes can also be resized through the toolbar, grouping resource icons or **Align Shapes Dialog**.

Through toolbar

Select **Diagram > Alignment** and an alignment option from the toolbar.

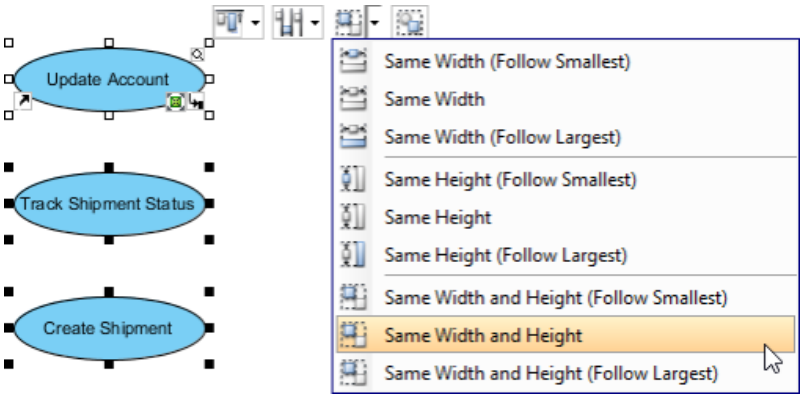
As a result, the shapes are resized.



Shapes are resized

Through grouping resources

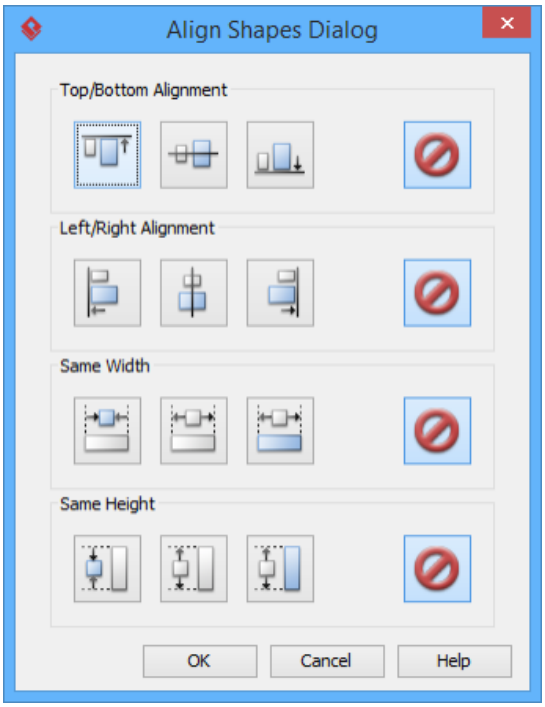
Select an option from the drop-down menu of the **Same Width** on grouping resource icons after select a few shapes on the diagram pane.



Resize through grouping resources

Through align shapes dialog

After select a few shapes on the diagram pane, select **Diagram > Alignment > Align Shapes...** from the toolbar to unfold **Align Shapes Dialog**. You can select an option by clicking the option button directly.



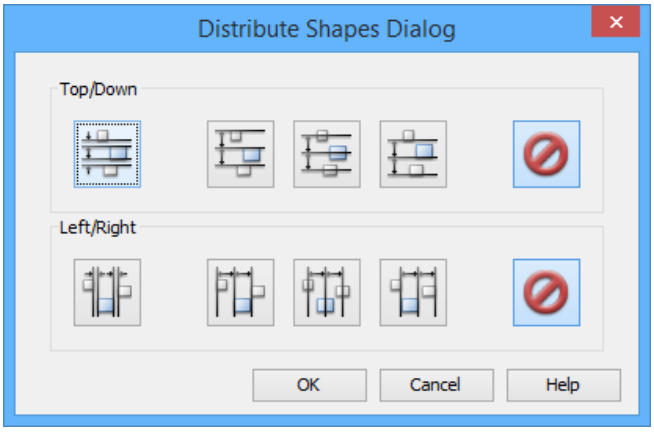
Align Shapes Dialog

Distributing diagram elements

In addition, model elements can be distributed in various directions through **Distribute Shapes Dialog**, the toolbar or grouping resource icons.

Through Distribute Shapes Dialog

After select a few shapes on the diagram pane, select **Diagram > Distribution > Distribute Shapes...** from the toolbar to unfold **Distribute Shapes Dialog**. You can select an option by clicking the option button directly.



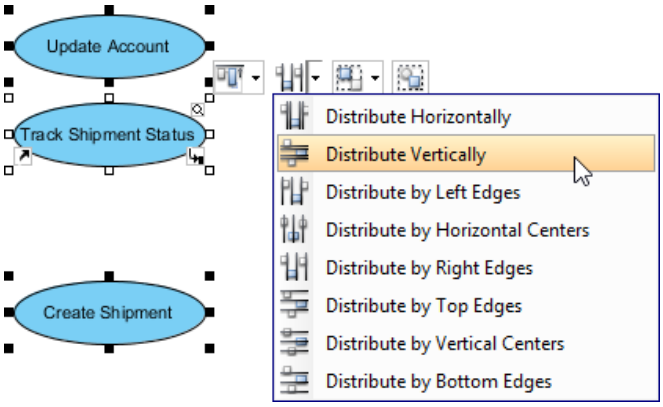
Distribute Shapes Dialog

Through toolbar

Select **Diagram > Distribution** and then select an distribution option from the toolbar after select a few shapes on the diagram pane.

Through grouping resource icons

Select a distribution option from the drop-down menu of **Distribute Horizontally** on grouping resource icons after select a few shapes on the diagram pane.



Distribute shapes on grouping resource icons

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

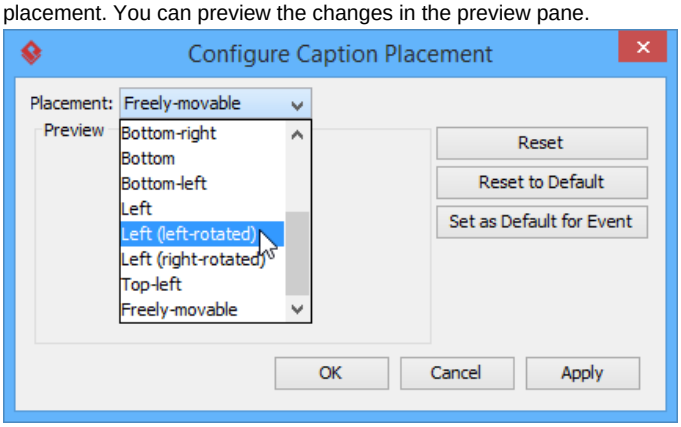
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Adjusting caption's position and angle in BPD

In BPD, for shapes like event and gateway, their names are put outside and below the shape, which may overlap with the outgoing sequence or message flow, making the name hard to read. To solve this problem, you can choose to place the caption elsewhere. Furthermore, you can rotate the caption to make it easier to read in print out.

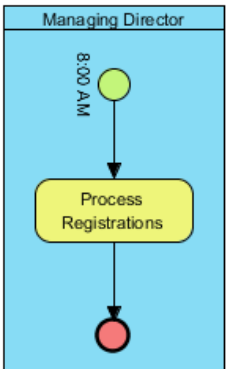
To set the position and angle of start, intermediate or end event, or gateway:

1. Right click on the start, intermediate or end event, or gateway shape and select **Presentation Options > Caption Placement...** from the popup menu.
2. Choose the placement, which is the position of caption. For some of the placement options, you can choose additionally the rotation of placement. You can preview the changes in the preview pane.



To choose a placement option

3. Click **OK** to confirm.



Caption position updated

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Zooming Diagram

If the diagram on diagram pane isn't clear enough, you can zoom in the diagram with your desired size through the **Zoom in** and **Zoom out** buttons.

To zoom in or zoom out a diagram, select **View> Zoom in** or **View > Zoom out** in the toolbar.

The zoom buttons in View toolbar

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram grids

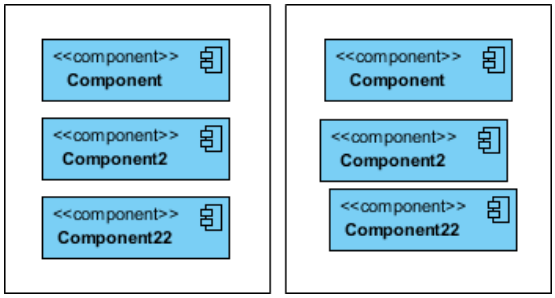
Diagram grids provide you with a precise visual cue to identify the position and boundary of shapes in diagram editors. You can make use of diagram grid to improve the positioning and alignment of shapes, which makes the diagram looks more neat, tidy and impressive. By default, the grid lines are not visible. In this article, we will see how to show them and how to configure the various grid settings. We will use UML component diagram to explain the ideas but in practice, you can apply diagram grids on any diagram types - UML diagrams, BPMN diagrams, DFD, ERD etc.

Visibility of grid lines

Grid lines are not visible on diagrams by default. You can optionally show them by updating the grid settings. Here are the steps:

1. Right click on the diagram where you want to show/hide grid lines.
2. Select **Open Specification...** from the popup menu.
3. Open the **Grid Setting** tab.
4. Check/uncheck **Grid visible** and click **OK**.

Snap to grid

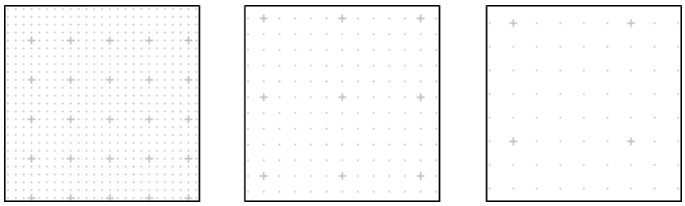


Effect of snap-to-grid on and off (left and right hand side)

The snap to grid function enables shapes to be positioned in an organized and well-aligned manner. When you draw, resize or move a shape on a diagram, it will align to the nearest grid line (even when grid lines are not visible). This means you can make multiple shapes apply to the same horizontal and/or vertical position, making the diagram looks more tidy. Note that the snap-to-grid option is turned on by default. If you want to change the setting:

1. Right click on the diagram where you want to enable/disable snap-to-grid.
2. Select **Open Specification...** from the popup menu.
3. Open the **Grid Setting** tab.
4. Check/uncheck **Snap to grid** and click **OK**.

Grid size

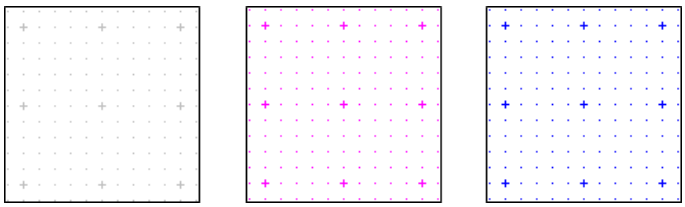


Different grid sizes - 5x5, 10x10 (default setting), 15x15

While the default spacing between grid line is set to 10 units, you can adjust the value to fulfill your diagramming needs. Please be reminded that when snap to grid is on, the shape you draw, resize or move will automatically align to the nearest grid line. This means that the value you input may influence the positioning of shapes. To update the grid size:

1. Right click on the diagram where you want to change the grid size.
2. Select **Open Specification...** from the popup menu.
3. Open the **Grid Setting** tab.
4. Edit the **Width** and **Height** of **Grid size** and click **OK**.

Grid color



Different grid colors - light gray (default setting), magenta, blue

Grid lines are not visible on diagrams by default. If you set them visible, you can see the light gray lines on diagram. If you want to change the color of grid lines, take the follow steps:

1. Right click on the diagram where you want to change the grid size.
2. Select **Open Specification...** from the popup menu.
3. Open the **Grid Setting** tab.
4. Pick up a **Grid color** and click **OK**.

Related Articles

- What to do If You Cannot Resize Shapes in a Diagram

Related Links

- Diagramming features of Visual Paradigm

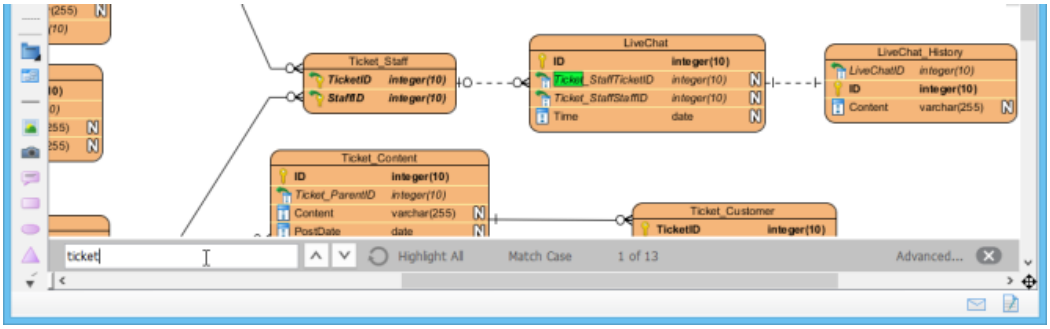
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Search bar

The search bar is a convenient way of locating the shape(s) you want without having to scroll through the entire diagram. To search for a shape, press **Ctrl-F** or select **Diagram > Search** from the toolbar to toggle the search bar. As you begin typing your search string in the search bar, the first result will be highlighted in the active diagram.



Searching with search bar

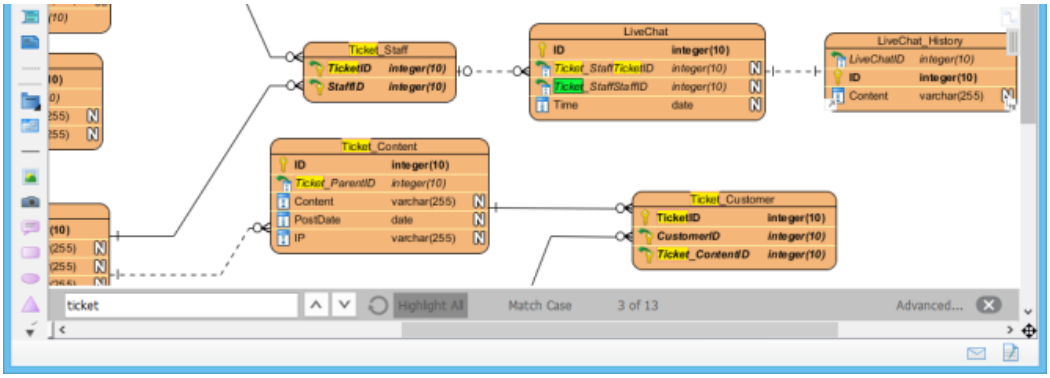
NOTE: The search process covers not just the shape name, but also it's content (for shapes like note, text annotation, BP annotation, etc). It also searches members like attributes and operations in class, and columns in entity.

Moving to next and previous result

There may be multiple shapes that match your search string. If you would like to see all the results for your search string, press **Enter** to proceed to the next occurrence after you've typed the search string. Alternatively, press the and buttons in the search bar to move between search results.

Highlighting all results

You can also highlight all the search results by clicking the **Highlight All** button in the search bar.



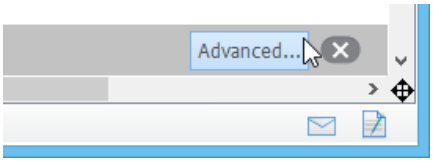
All results highlighted

Match case

Searching is a case in-sensitive process by default. If you want it to be case sensitive, click **Match Case** in the search bar.

Advanced 'Find' feature

The search bar provides a relatively simple and light-weight search support. If you want a more advanced search support, click **Advanced**... on the right of the search bar to access the advanced **Find** feature.



Opening advanced Find feature

Closing the search bar

To close the search bar, click on the  button on the right of the search bar.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Advanced modeling techniques

This chapter covers less frequently used or comparatively complex modeling techniques.

Sweeper and magnet

Sweeper and magnet are handy tools for moving a group of shapes back and forth.

Mouse gestures

You can create shapes, connect shapes, or perform certain operations through pressing and dragging your right mouse button. You can gain more information in the *Mouse gestures* page.

Jumping to shape

When you are looking for a shape, a diagram or a model element, you can make use of the jump to feature to enter its name and jump to it immediately. It's like a commonly-known search function, but a faster approach.

Grouping diagram elements

You can more and format shapes by grouping them together. You will see how to group diagram elements on a diagram.

Show/hide diagram elements

You can optionally hide away some of the diagram elements on a diagram, or hide specific type of elements.

Layer

Layer provides a logical shape division in diagram. For example, a comment layer for annotation shapes. You can hide, lock and set active to a layer.

Making shape non-selectable

You can make shape non-selectable to avoid accidental movements for particular shapes. This is particular helpful when trying to move shapes in a container like package, without moving the package by mistake.

Showing model element in multiple diagrams

A model element can have multiple views. In this page you can see how to make use of drag and drop to create multiple views for a model element.

Using overview diagram

Overview diagram is best used to illustrate the relationship between diagrams, hence the content (e.g. interaction) they represent.

Changing Model Element Type

Shows you how to convert the type of a model element

Sweeper and Magnet

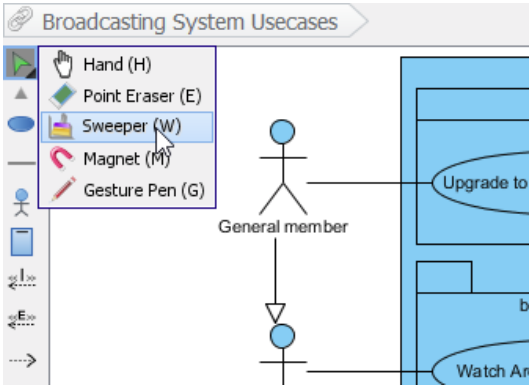
Modifying your diagrams from time to time is no longer a drag-- let's take our utilities: sweeper and magnet can help you to modify your diagrams easily. With these two features, you can move diagram elements easily without worrying too much about the layout. Sweeper can help you to increase more space between the diagram elements while magnet can help you to diminish the space.

Sweeper

The sweeper is one of the useful features for editing your diagrams. If you have ever experienced of moving the diagram elements without any tools, you probably understand how hard it is to manage the space between the diagram elements. Using sweeper to extend the space between the diagram elements allows you to move your diagram elements conveniently.

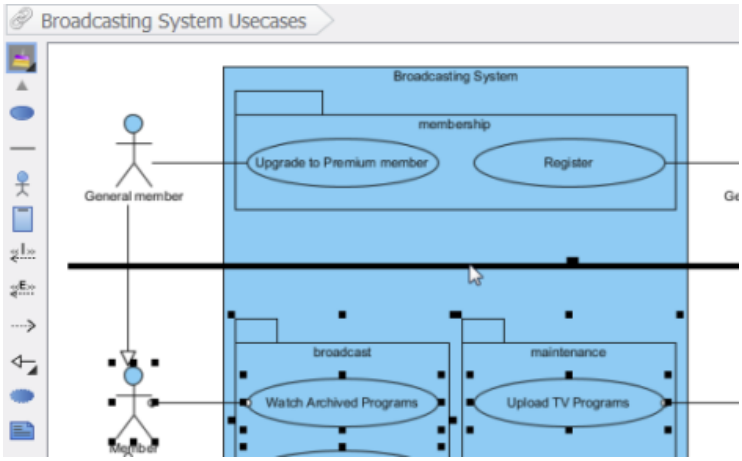
You can move the diagram elements by following the simple steps below:

1. Click **Sweeper** button from the diagram toolbar.



*Click **Sweeper***

2. Move the mouse on the diagram pane where you would like to move diagram elements.
3. Hold onto your mouse and move the line horizontally or vertically.

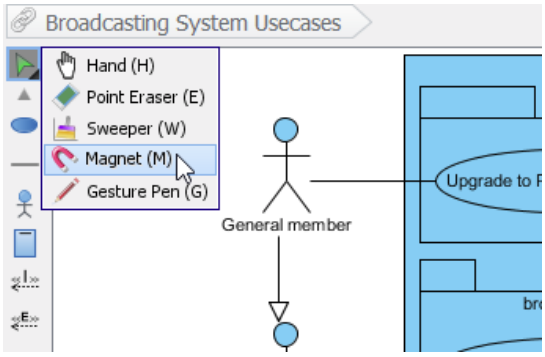


Moving down the diagram element horizontally

Magnet

The magnet is another convenient feature for you to move your diagram elements. If you want to move a few diagram elements, you should try magnet. Its function is to diminish the space between the diagram elements and make your diagrams much tidier for printing. The steps of applying magnet on your diagram are shown as follows:

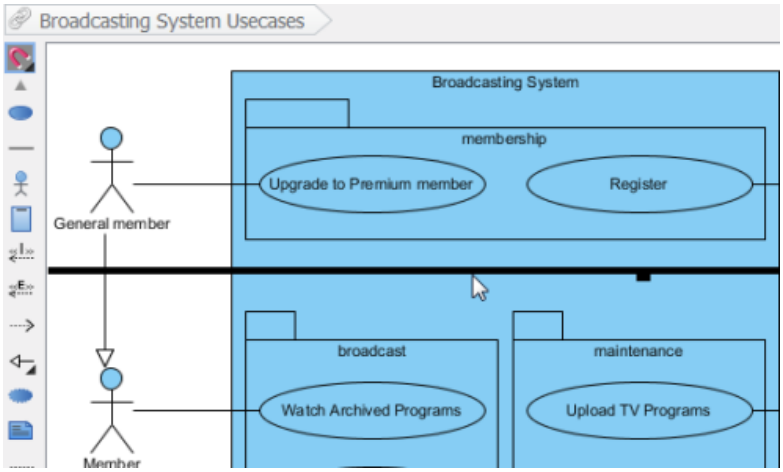
1. Click **Magnet** button from the diagram toolbar.



*Click **Magnet***

2. Move the mouse on the diagram pane where you would like to move diagram elements.

3. Hold onto your mouse and move the line horizontally or vertically.



Moving up the diagram element horizontally

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Mouse gestures

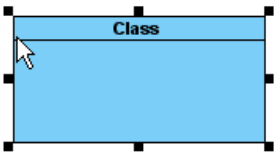
A variety of shapes and model elements can be created by sketching a path directly on the diagram pane with dragging the right mouse button to form a gesture. For your convenience and quick creation, mouse gestures allow you to execute common commands and create UML models within all diagrams.

Drawing shapes

1. To start using a mouse gesture, press the right mouse button and drag it until finished drawing a shape.

Drawing clockwise rectangle

2. When the shape is done, release the mouse. After the shape is created, the action description will be shown on the top right corner of the diagram.



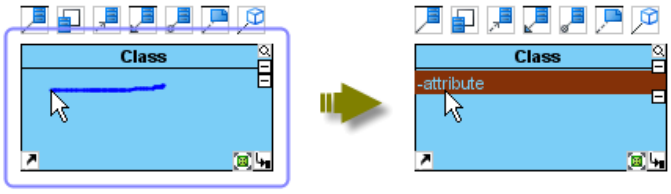
Class created

Creating Class member

You can learn how to create attribute and operation within the class in the following sub-sections.

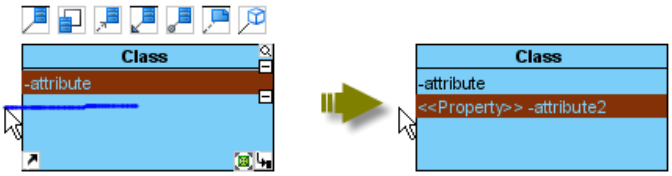
Creating an attribute

1. To create attribute, draw a line from the right to the left within the class. As a result, an attribute is created.



Attribute is created

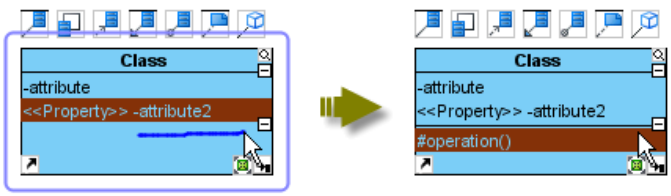
2. If you draw the line until outside the class, an attribute with <<Property>> stereotype will be created.



<<Property>> is created

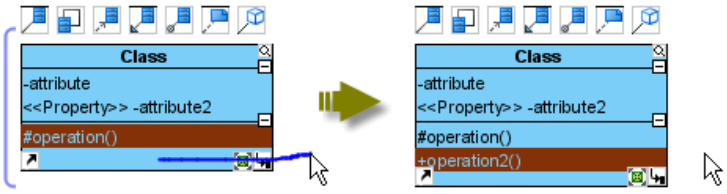
Creating an operation

1. To create operation, draw a line from the left to the right within the class, an operation with protected visibility is created.



Operation is created

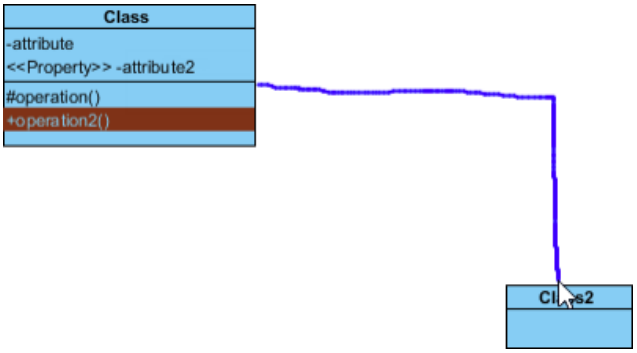
2. If draw the line until outside the class, a public operation will be created.



Public operation is created

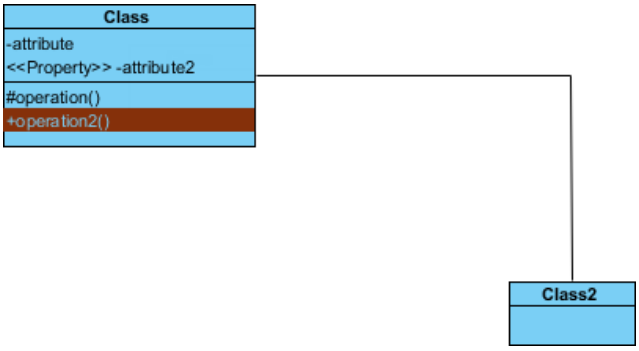
Connecting shapes

1. Draw a line from one shape to another.



Drawing from a shape to another

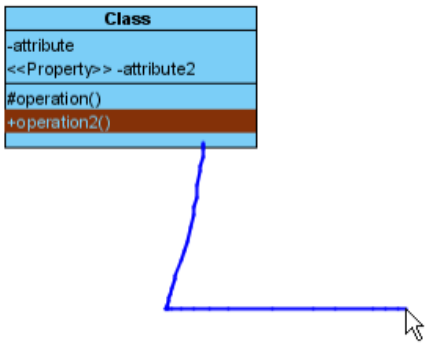
2. After the mouse is released, a connector is created between two shapes.



Association created

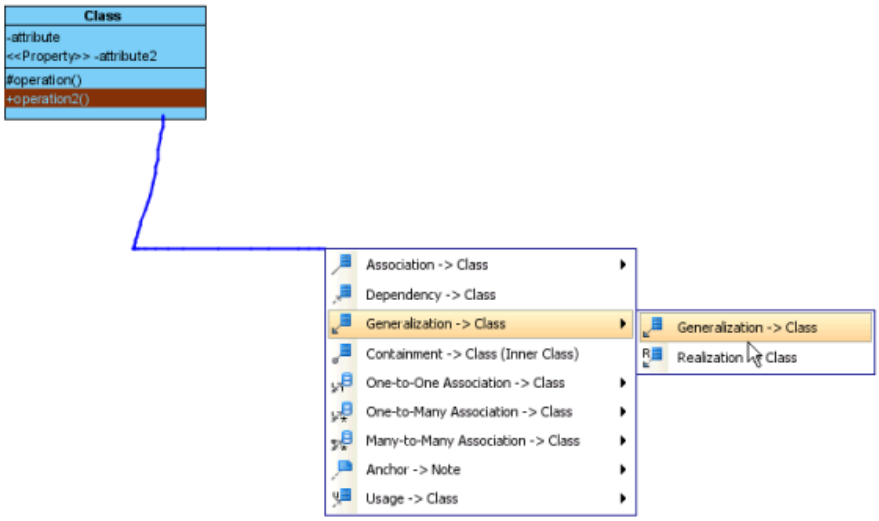
Creating a new shape

1. A new shape can also be created. To do so, draw a line from an existing shape to your preferred place.



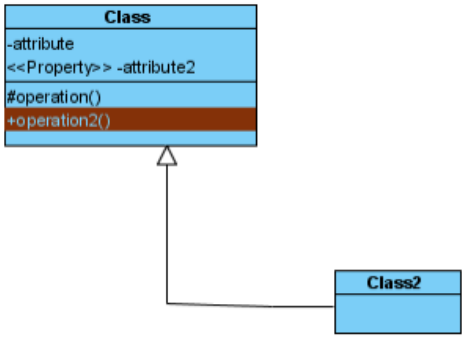
Drawing to empty area

2. After the mouse is released, a pop-up menu will be shown. You can select your preferred type of connector and shape from the pop-up menu.



Create generalization with class

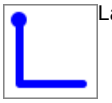
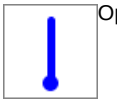
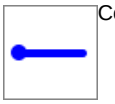
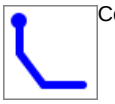
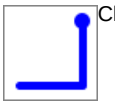
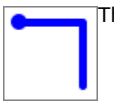
3. The two shapes with connector are created.



Class with generalization created

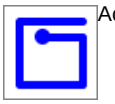
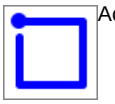
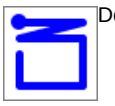
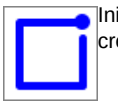
List of supported mouse gestures

General

Icon	Description
	Layout diagram
	Open diagram specification
	Connect new shape
	Connect existing shape
	Close Diagram
	Thumbnail view

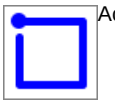
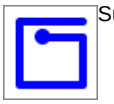
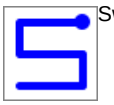
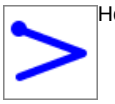
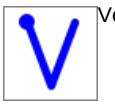
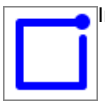
The description of general mouse gestures

Activity diagram

Icon	Description
	Action
	Activity
	Decision Node
	Initial Node/Finial Node (If there is no Initial Node, an Initial Node will be created. Else if there is no Final Node, a Final Node will be created.)

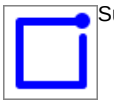
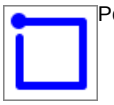
The description of mouse gestures for activity diagram

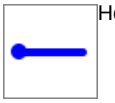
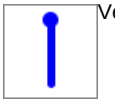
Activity diagram (UML 1.x)

Icon	Description
	Action State
	Sub-Activity
	Swimlane
	Horizontal Synchronization Bar
	Vertical Synchronization Bar
	Initial State/Final State (If there is no Initial State, an Initial State will be created. If there is no Final State, a Final State will be created.)

The description of mouse gestures for activity diagra

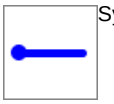
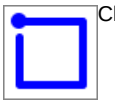
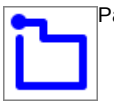
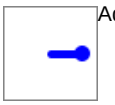
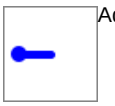
Business process diagram

Icon	Description
	Sub-Process
	Pool/Task

	Horizontal Lane
	Vertical Lane

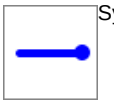
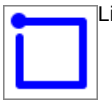
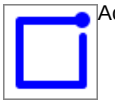
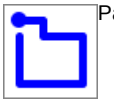
The description of mouse gestures for business process diagram

Class diagram

Icon	Description
	Sync. to ERD
	Class
	Package
	Add attribute (Add an attribute to class. If mouse released outside the class, getter and setter property will be set to true.)
	Add operation (Add an operation to class. If mouse released inside the class, visibility will be protected, otherwise it will be public.)

The description of mouse gestures for class diagram

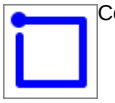
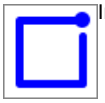
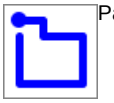
Communication diagram

Icon	Description
	Sync. To Sequence Diagram
	Lifeline
	Actor
	Package

The description of mouse gestures for communication diagram

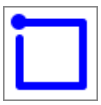
Component diagram

Icon	Description

	Component
	Instance Specification
	Package

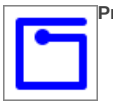
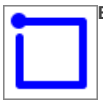
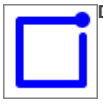
The description of mouse gestures for component diagram

Composite structure diagram

Icon	Description
	Class
	Interface
	Collaboration
	Collaboration Use

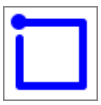
The description of mouse gestures for composite structure diagram

Data flow diagram

Icon	Description
	**Process**
	**External Entity**
	**Data Store**

The description of mouse gestures for data flow diagram

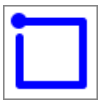
Deployment diagram

Icon	Description
	Node
	Component

	Instance Specification
	Package

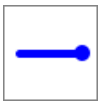
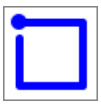
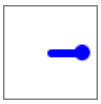
The description of mouse gesture for deployment diagram

EJB diagram

Icon	Description
	Entity Bean
	Message-Driven Bean
	Session Bean
	Package

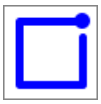
The description of mouse gestures for EJB diagram

Entity relationship diagram

Icon	Description
	Sync. to Class Diagram
	Entity
	Add column

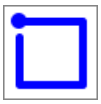
The description of mouse gestures for ERD

Interaction overview diagram

Icon	Description
	Interaction
	Decision Node
	Initial Node/Final Node (If there is no Initial Node, an Initial Node will be created. Else if there is no Final Node, a Final Node will be created.)

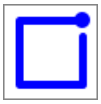
The description of mouse gestures for interaction overview diagram

Mind mapping diagram

Icon	Description
	Node

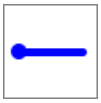
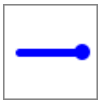
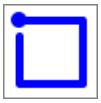
The description of mouse gesture for mind mapping diagram

Object diagram

Icon	Description
	Instance Specification
	Class
	Package

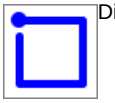
The description of mouse gestures for object diagram

ORM diagram

Icon	Description
	**Sync. Classes -> Entities**
	**Sync. Entities -> Classes**
	**Class**
	**Entity**
	**Package**

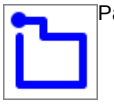
The description of mouse gestures for ORM diagram

Overview diagram

Icon	Description
	Diagram Overview

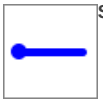
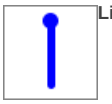
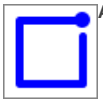
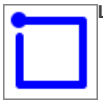
The description of mouse gesture for overview diagram

Package diagram

Icon	Description
	Package

The description of mouse gesture for package diagram

Sequence diagram

Icon	Description
	**Sync. to Communication Diagram**
	**Lifeline**
	**Actor**
	**Alt**
	**Loop**

The description of mouse gestures for sequence diagram

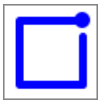
State machine diagram

Icon	Description
	**State**
	**Submachine State**
	**Initial Node/Final Node (If there is no Initial State, an Initial State will be created. Else if there is no Final State, a Final State will be created.)**

The description of mouse gestures for state machine diagram

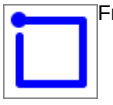
State machine diagram (UML 1.x)

Icon	Description
	State
	Concurrent State

	Submachine State
	Horizontal Synchronization Bar
	Vertical Synchronization Bar
	Initial State/Final State (If there is no Initial State, an Initial State will be created. Else if there is no Final State, a Final State will be created.)

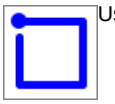
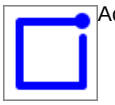
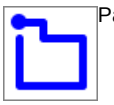
The description of mouse gestures for state machine diagram (UML 1.x)

Timing diagram

Icon	Description
	Frame

The description of mouse gesture for timing diagram

Use case diagram

Icon	Description
	Use Case
	Actor
	Package

The description of mouse gestures for use case diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

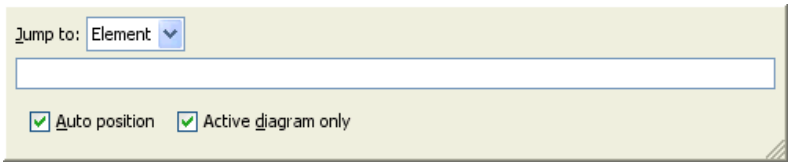
Jumping to shape

For searching a shape/shapes faster, the application of the jump to shape/shape facility is introduced. You can select either jump to a model element in an active diagram, or jump to any model elements in the current project, or even jump to a diagram in current project.

Jumping to a diagram/model element in project

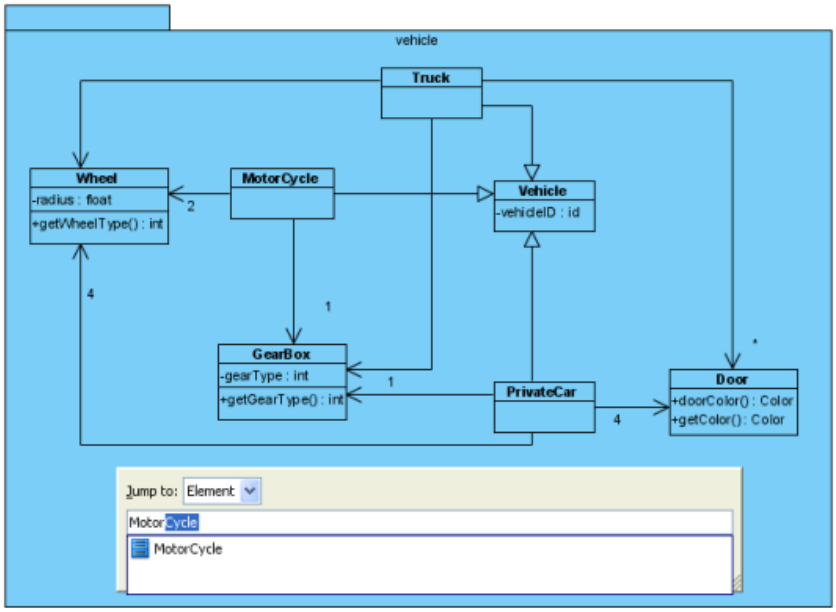
If there is an active diagram opened, you can jump to the model element in the active diagram.

1. You can select **Diagram > Jump > to Element in Active Diagram...** from the toolbar or press **Ctrl+J** to unfold **Jump to** dialog box.
2. Apart from jumping to element in active diagram, you can also jump to any element within the project. You may select **Diagram > Jump > to Element...** from the toolbar or press **Ctrl+Shift+J** to unfold **Jump to** dialog.
3. In **Jump to** dialog box, if you want all elements within project to be searched, uncheck the **Active diagram only** checkbox. In some cases, the checkbox is disabled because no diagram is opened.



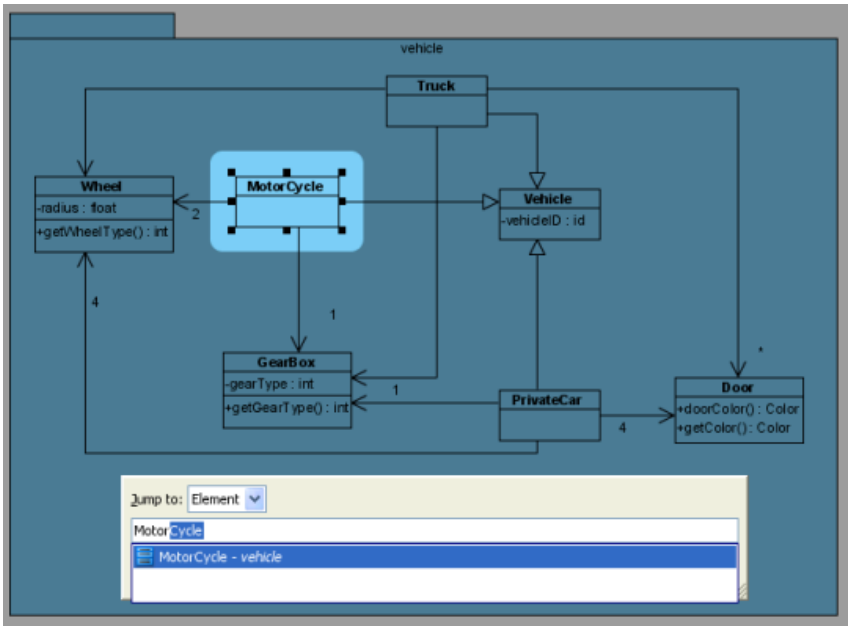
Jump to dialog box is shown

4. Enter a word in the text field, a list of model element's name that starts with the word you typed will be shown .



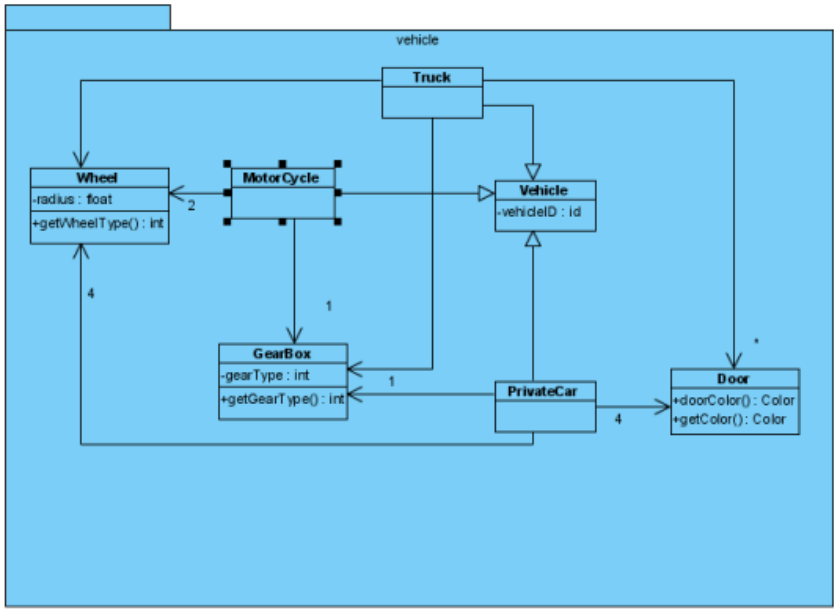
A list of possible shapes is shown

5. Press **Down** key to search for the model element's name if the list is too long. Click your preferred model element's name and it will be spot-lighted on the active diagram.



Shape is spot-lighted

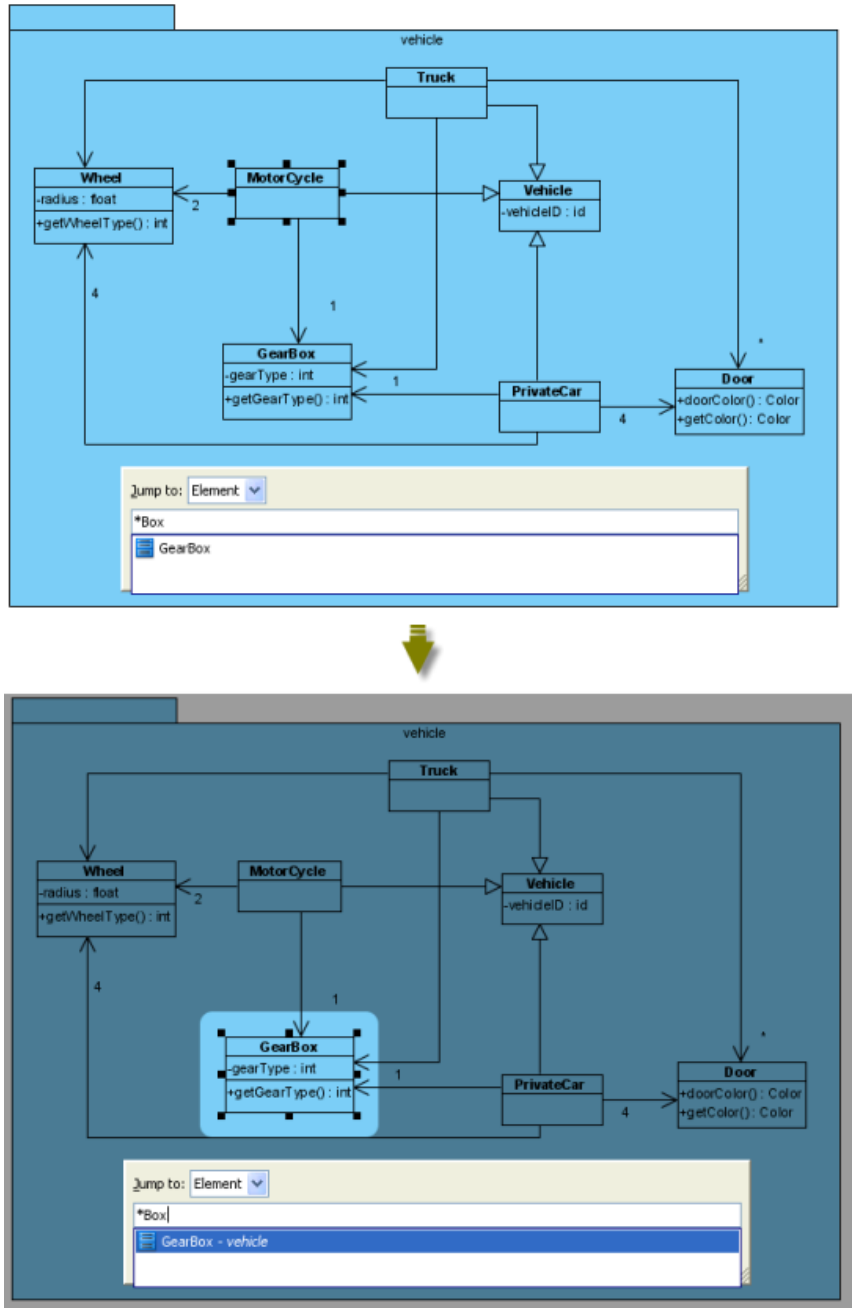
6. Press **Enter** to confirm jump to the model element. Finally, the **Jump to** dialog box will then be hidden and the model element will be selected on diagram.



Shape is selected after confirm jump to

Filtering with wild card character

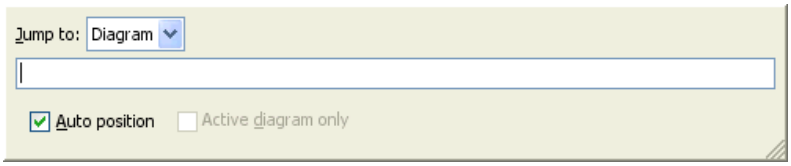
For quick search, you can type a word with * in the text field. The asterisk can substitute a character or a word when you don't remember the exact spelling. As a result, all names of shape that are similar to the word you typed will be shown.



*Entering name with **

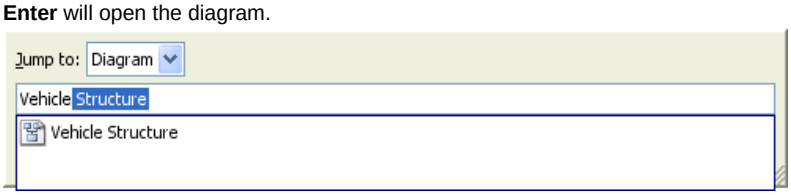
Jumping to diagram

1. To do so, select **Diagram > Jump > to Diagram...** from the toolbar or press **Ctrl+Shift+D** to show **Jump to** dialog box.
2. **Jump to** dialog box is shown with selected **Diagram** in combo box. It means **Jump to** dialog box will search all diagrams within the project.



Jump to dialog will search all diagrams within the project

3. Enter a word out of the whole diagram's name will show a list of diagrams' names similar with the word you typed . Select the diagram and press **Enter** will open the diagram.



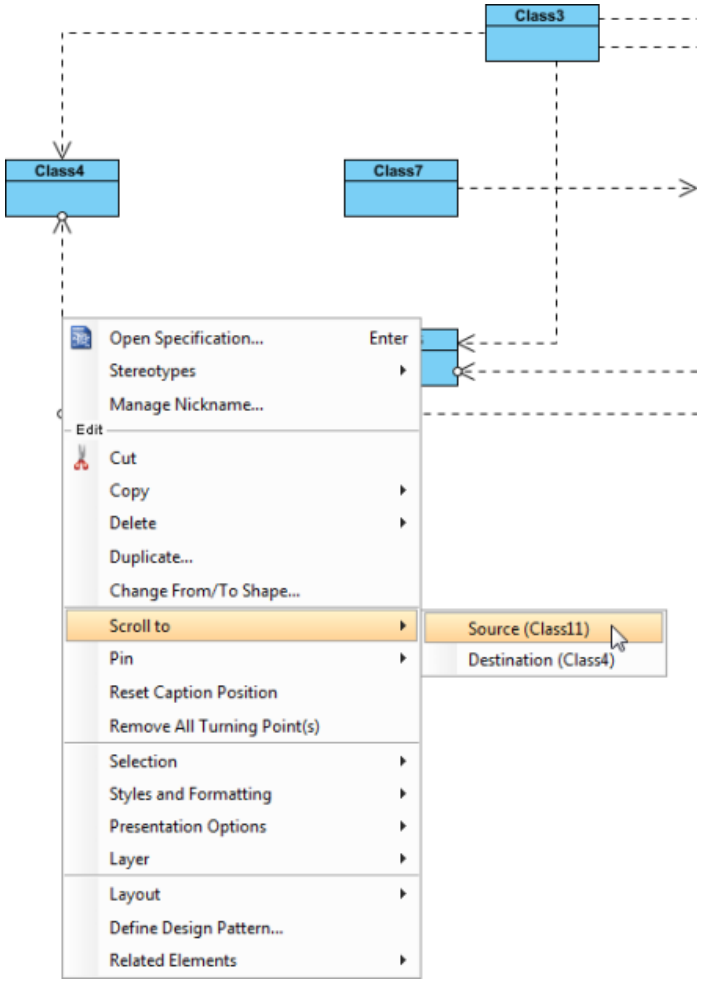
Select the diagram on list

Easy navigation to connected elements

If you have an oversized diagram that the from/to model elements of connector can't be shown on the screen, it is hard for you to know the from/to model elements. Visual Paradigm supports **Scroll to** function to scroll to from/to model element of a connector. If you want to know which model element is the from model element that is connected with a model element .

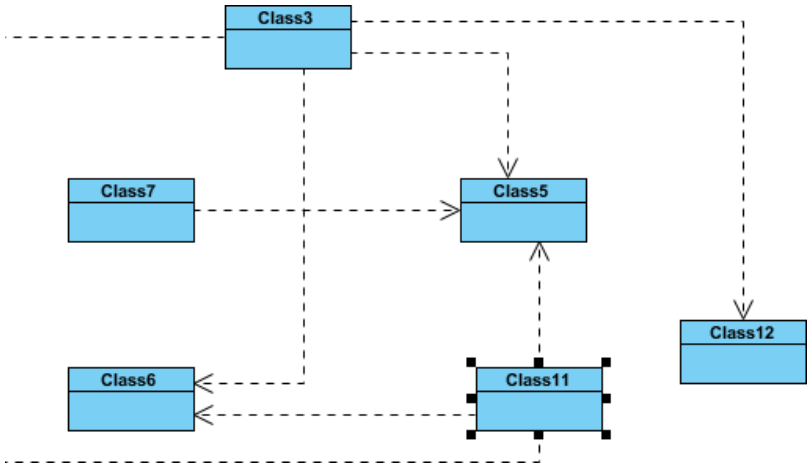
To achieve this:

1. Right click on the connector and select **Scroll to** and then select **Source** with parentheses from the pop-up menu. (The word in parentheses is the name of the from model element.)



Scroll to from model element

2. The from model element is subsequently selected on diagram.



From model element is selected

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

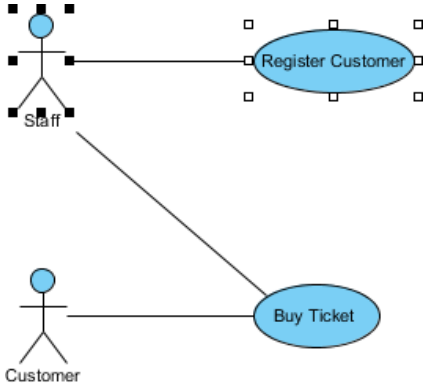
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Grouping diagram elements

After you have aligned shapes to a group of shapes, in some cases, you want to move a number of shapes simultaneously or make them share the same formatting properties, like background color and line formatting. Grouping can help you for this purpose exactly. By grouping shapes, shapes within the group will move together when moving any shape inside the group. If you edit formatting of one shape, all shapes will also be shared.

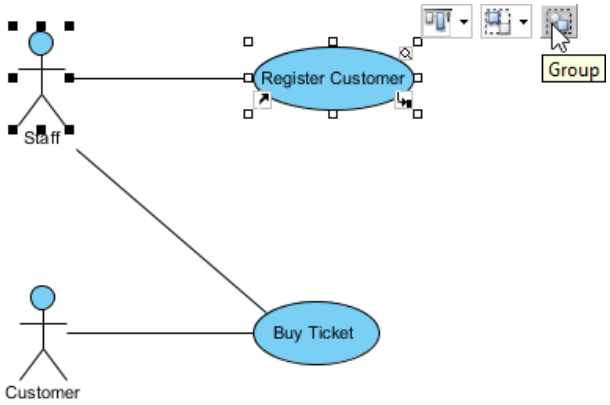
Grouping diagram elements

1. Select the shapes you want them to be grouped together.



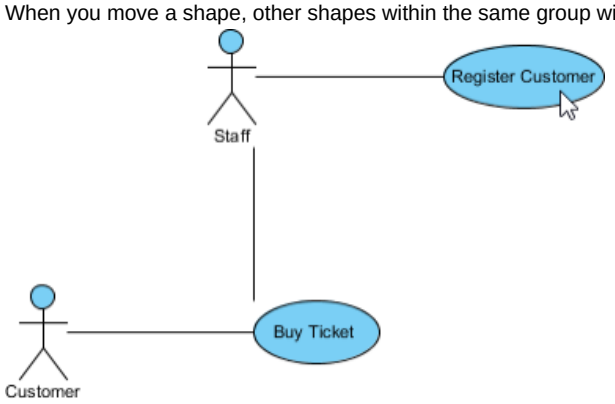
Select several shapes

2. Click resource icon **Group** to group the selected shapes.



*Group through resource icon **Group***

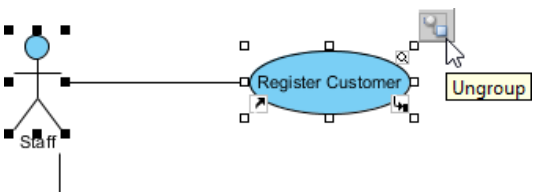
3. When you move a shape, other shapes within the same group will also be moved.



All shapes in group are moved together

Ungrouping diagram elements

To ungroup the shapes, click resource icon **Ungroup**.



*Click **Ungroup***

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

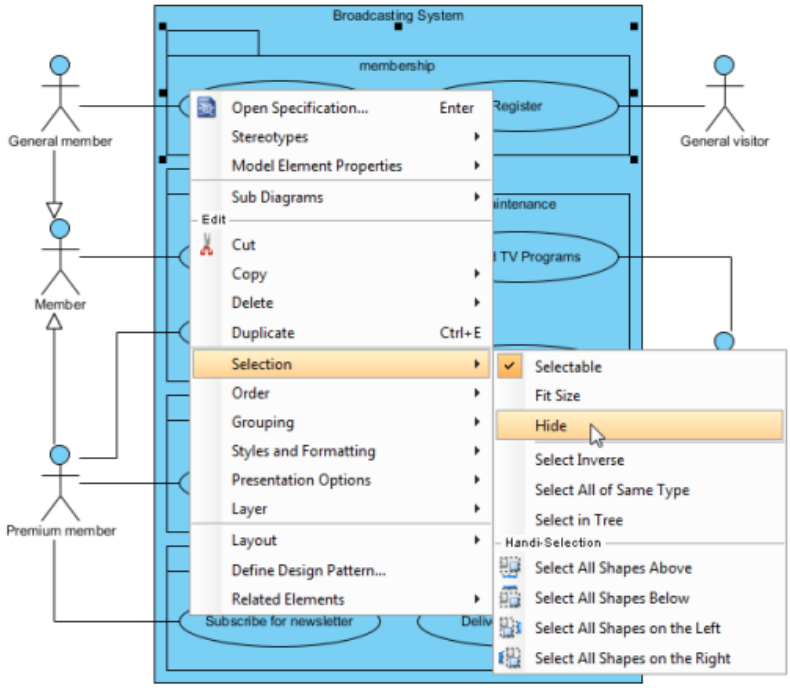
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Show/hide diagram elements

On a compact diagram, to hide or show some of the shapes is necessary. You can choose to hide away diagram elements on a diagram temporarily. For example, hiding away annotation shapes which record internal communications before printing out a diagram for customers' review. In Visual Paradigm, hiding of diagram elements can be done per shape, per shape type or by stereotype.

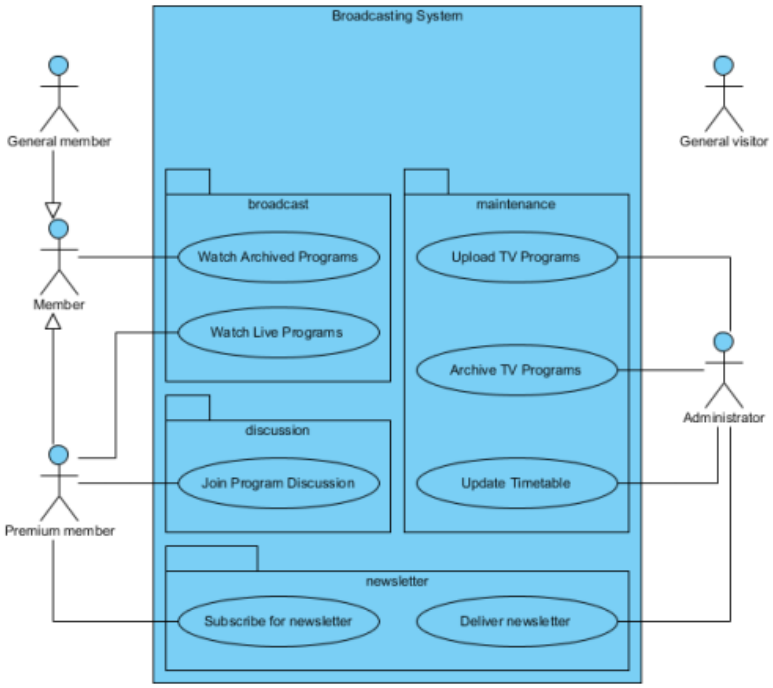
Hiding selected diagram element(s)

Right click on the selected shape(s) and select **Selection > Hide** from the pop-up menu.



*Select **Hide** from the pop-up menu*

Apart from the selected shapes, its related shape(s) (e.g. children and relationship) will also be hidden.



The selected shape and its related shapes are hidden

NOTE: To hide a shape will also make the connectors that attached to it hidden.

NOTE: To hide a container shape (e.g. package) will also make the contained shapes hidden.

Hiding diagram elements by shape type

All the shapes with same type from the diagram can also be hidden. Right click on the diagram's background and select **Diagram Content > Show/ Hide > Hide by Shape Type,** and then select the shape you want to be hidden from the pop-up menu. As a result, all shapes with the same type you selected will be hidden.

Hiding diagram elements by stereotype

The stereotype of all diagram elements from the diagram can be hidden. Right click on the diagram's background and select **Diagram Content > Show/Hide > Hide by Stereotype**, and then select one stereotype out of the list from the pop-up menu. As a result, all shapes with same stereotype you selected will be hidden.

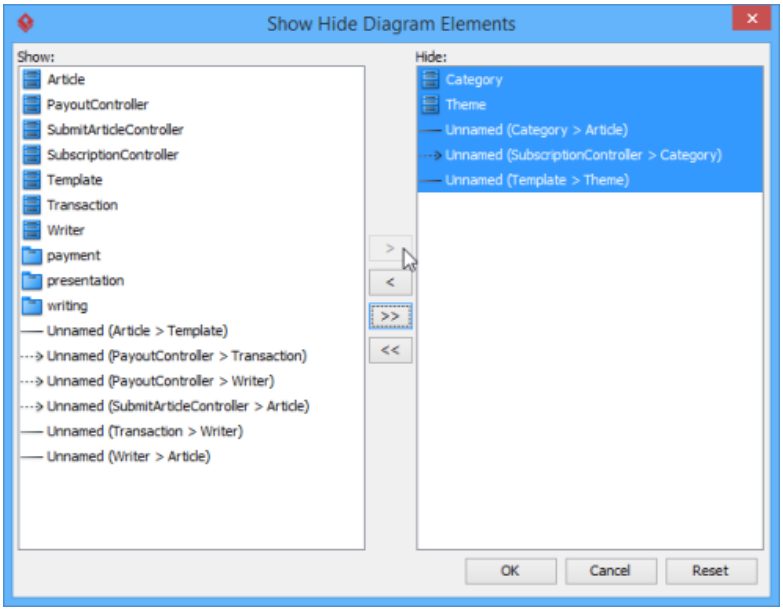
Showing all hidden diagram elements

All hidden shapes from the diagram can be shown again. Right click on the diagram's background and select **Diagram Content > Show/Hide > Show all Diagram Elements**. As a result, the hidden shapes will be shown on the diagram.

Managing show/Hide of specific diagram element(s)

A better management of showing and hiding the shapes can be done in **Show Hide Diagram Elements** dialog box..

1. Right click on the diagram's background, select **Diagram Content > Show/Hide > Show/Hide Diagram Elements...** from the pop-up menu.
2. When **Show Hide Diagram Elements** dialog box is unfolded, you may select the shape(s) you would like to be hidden or to be shown. Click the diagram element you would like to move to **Hide** list and press **Hide Selected** button; on the other hand, click the diagram element you would like to remove it back to **Show** list and press **Show Selected** button. For moving all diagram elements to **Hide** list, press **Hide All** button; press **Show All** button, vice verse. Finally, click **OK** to confirm.



*Select a diagram element to be moved to **Hide** list*

> **NOTE:** If a shape is selected from **Show** list, the related shape(s) of the selected shape will be also removed to **Hide** list.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

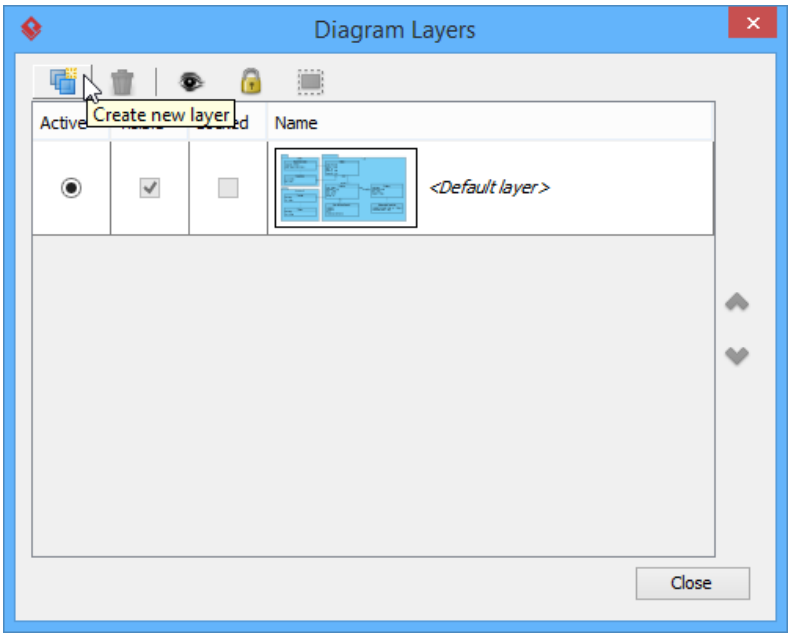
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Layer

If you have ever experienced how hard it is to deal with a number of shapes, try the application of multi-layers. Visual Paradigm supports multi-layers to help you manage different shapes efficiently. The functions of layer assist you in assigning different shapes into different layers, hiding unnecessary shapes, locking shapes and selecting shapes in shortcut.

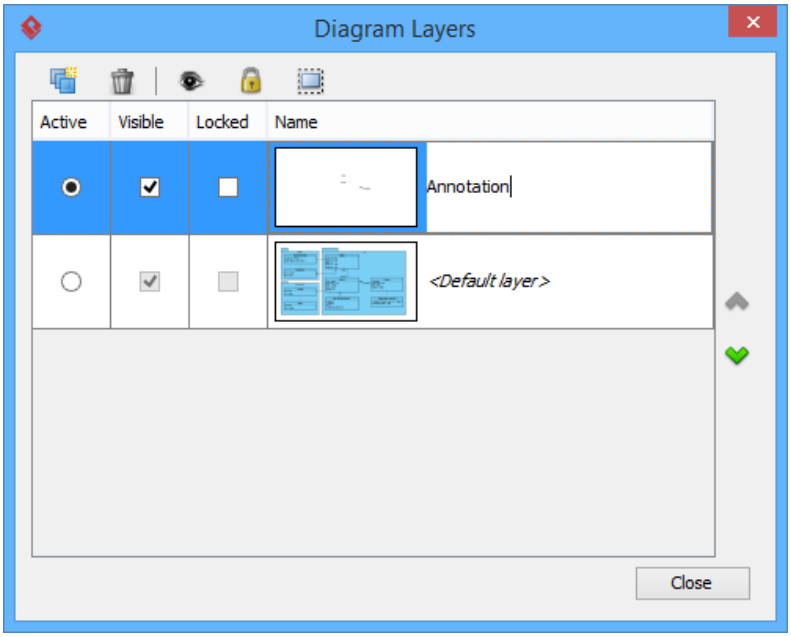
Creating a layer

1. Click on **View > Layers** from the toolbar.
2. In **the Diagram Layers** window, click **Create new layer** button to create a new layer.



*Create a new layer in **Diagram Layers** window*

3. Finally, define the name for the newly created layer.

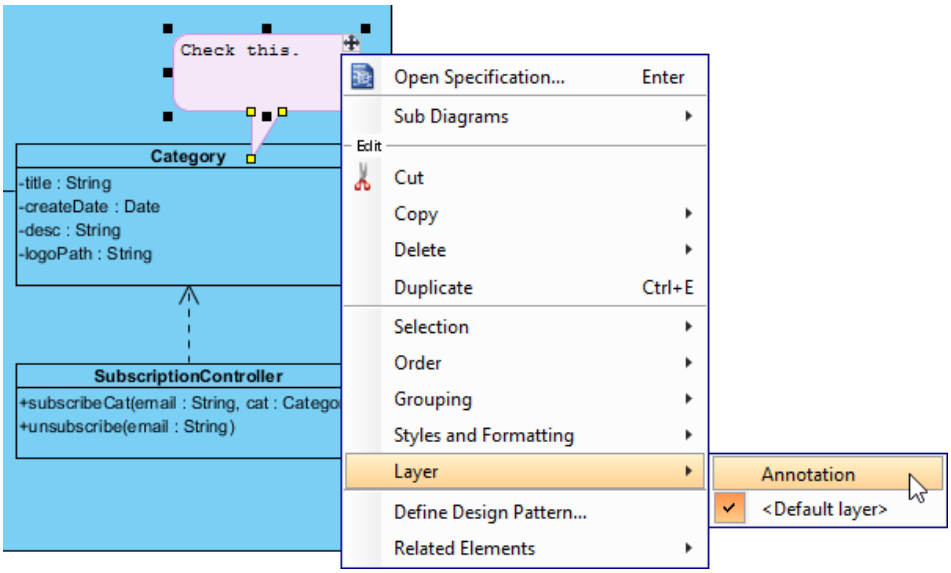


New layer will be an active layer

Sending shapes to a layer

The existing shapes are kept in default layer. However, both existing shapes and the newly created shapes on diagram can be sent to a new layer.

Create a new layer is just like the steps of previous section. Right click on a shape and select **Layers**, and then select the new layer you have created.

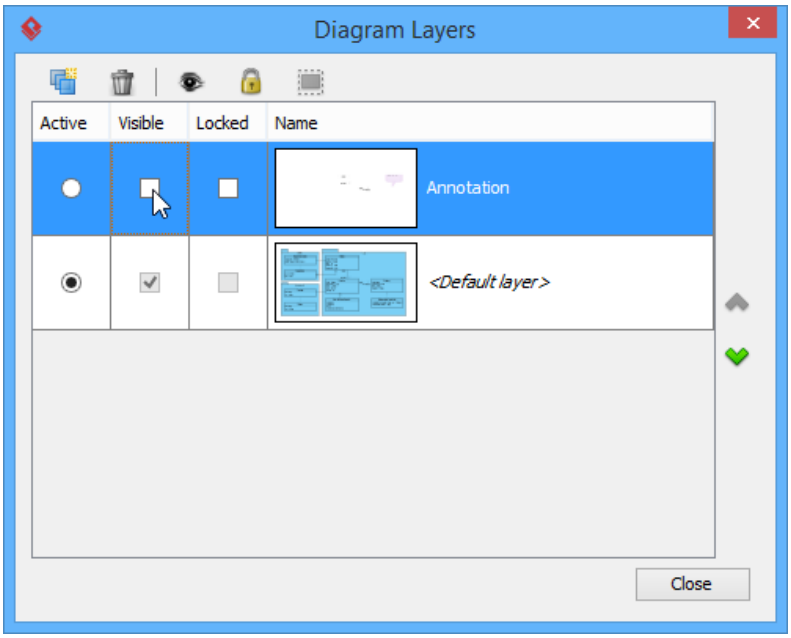


Send the shape to a new layer

As a result, the shape you selected will be sent to the new layer.

Hiding shapes on a layer

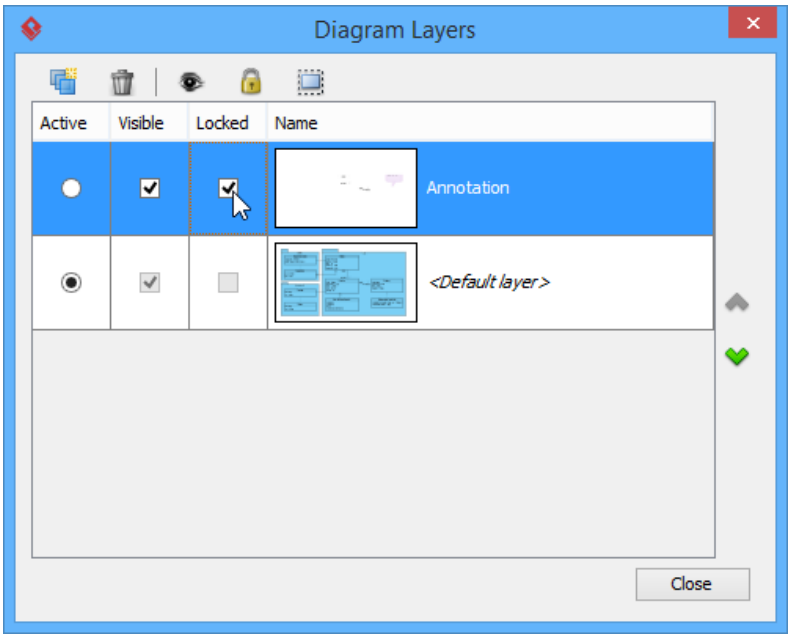
In the **Diagram Layers** window, you can make a layer invisible on diagram. To do so, uncheck **Visible** of the layer.



Shapes on the layers are invisible on diagram

Locking shapes on a layer

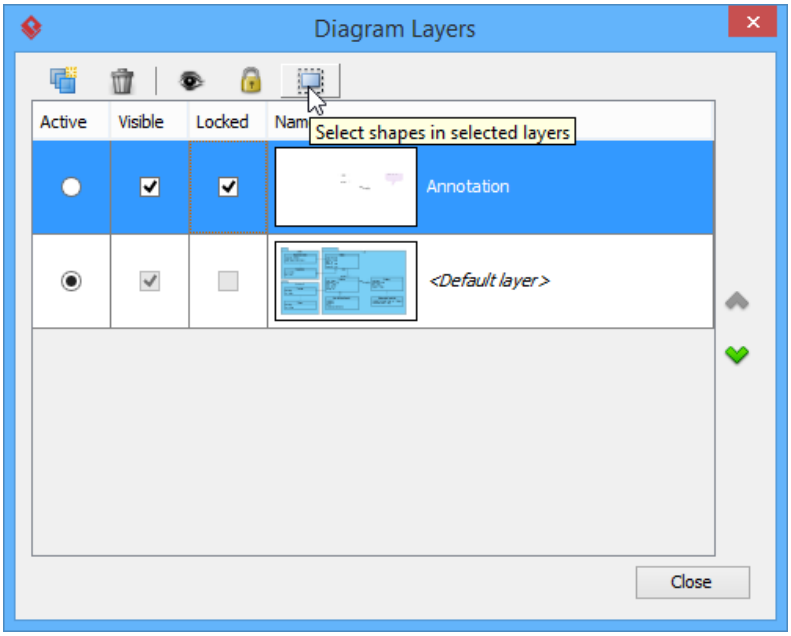
In the **Diagram Layers** window, check **Locked** to make all shapes in that layer immovable and non-editable while uncheck **Locked** to make them movable and editable.



Shapes on the selected layer are locked

Selecting shapes on a layer

You can also select all shapes of the selected layer. To do so, click **Select shapes in selected layers** button in **Diagram Layers** window.



Shapes of the layer are selected on diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

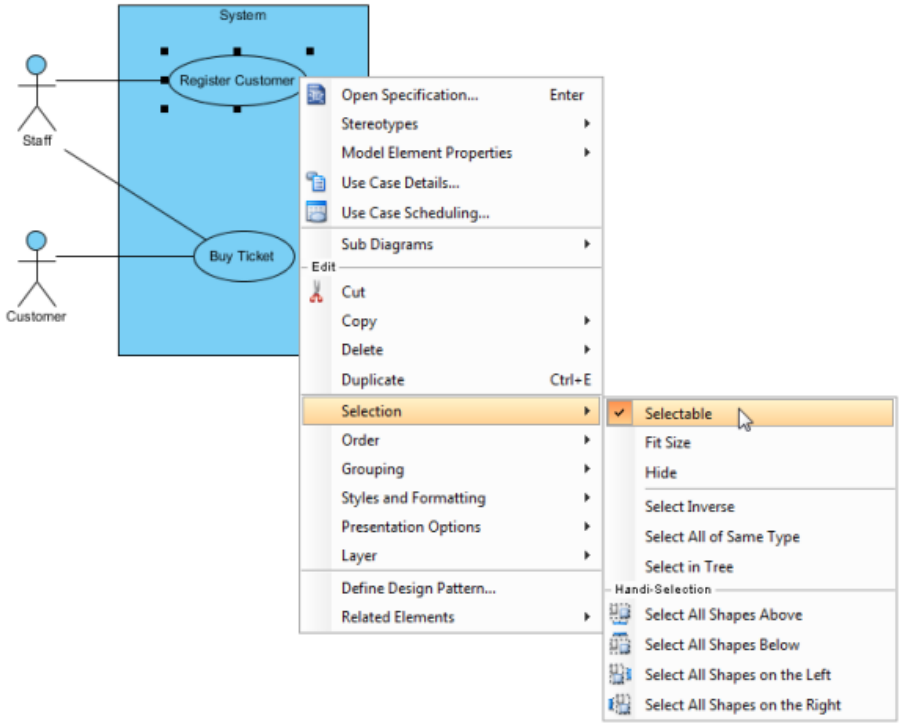
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Making shape non-selectable

If you find it annoying to move some shapes carelessly or select some shapes accidentally, the function of selectable/non-selectable would be your ideal trouble-shooter. This page is going to teach you how to make shapes non-selectable or even change them back to selectable with only few clicks.

Changing the shape to non-selectable

1. Right click on the selected shapes, uncheck **Selection > Selectable** from the pop-up menu.

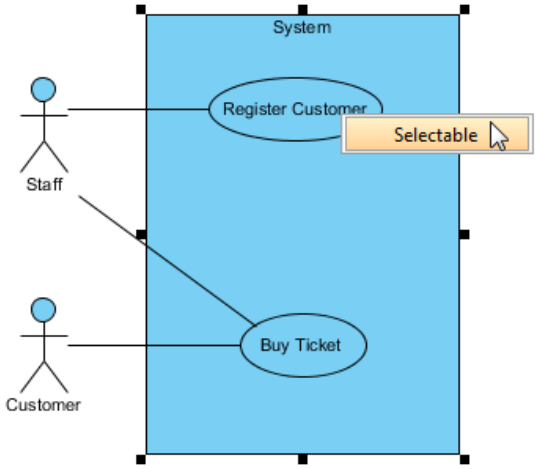


Selected shape to be non-selectable

2. After the shape is non-selectable, click the shape cannot make it to be selected. Neither will the shape to be selected by mouse dragging on diagram.
3. Therefore, the non-selectable shape(s) will not be moved when other shapes are moved.

Changing the shape to selectable again

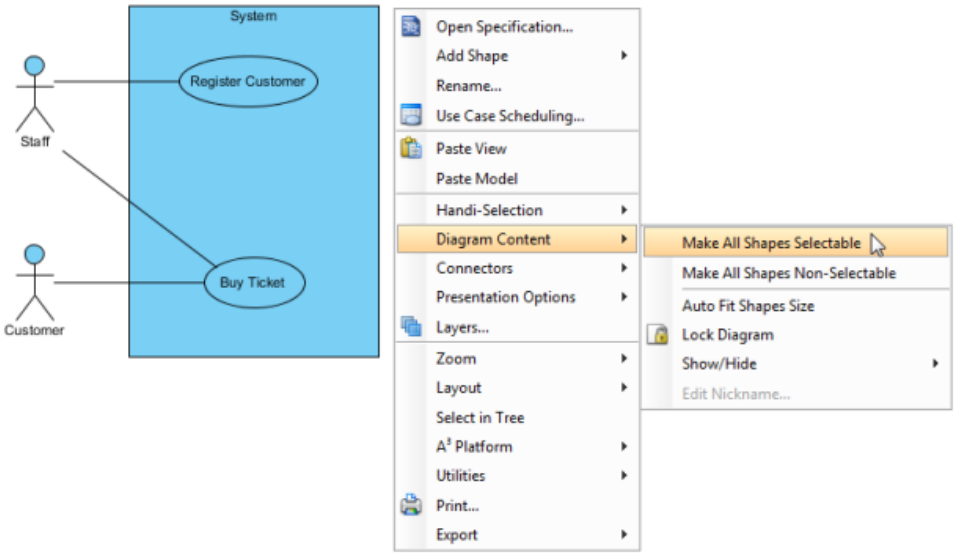
To make the shape selectable again, right click on the non-selectable shape and select **Selectable** from the pop-up menu.



Select Selectable

Setting all shapes to selectable or non-selectable

To makes all shapes on the diagram to be selectable or non-selectable, right click on the diagram background, select **Diagram Content** and then select either **Make All Shapes Selectable** or **Make All Shapes Non-Selectable** from the pop-up menu.



*Select **Make All Shapes Selectable***

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

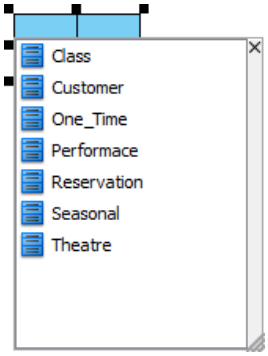
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Showing model element in multiple diagrams (Context based modeling)

You can show a model element in multiple diagrams to fit into different contexts and we call this to "reuse a model element". There are mainly two ways you can take to reuse a model element - through 'Name Completion' when creating shape or through copy and paste.

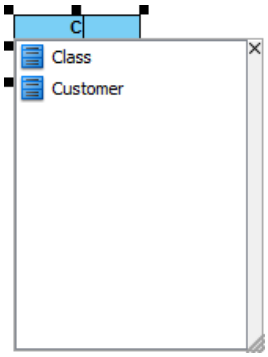
Reusing model element when creating shape

The model name completion feature enables quick creation of multiple views for the same model element. When you create a model element, press **Ctrl-Space** to reveal the name completion list.



Model name completion

Type text to filter the elements in the list.

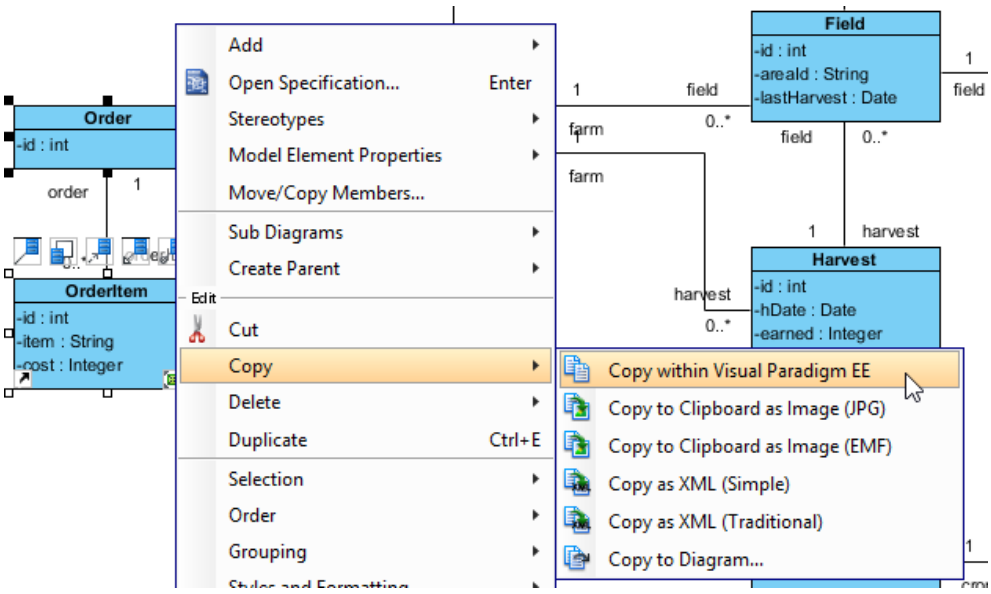


List filtered by typed text

Press up or down key to select the model element to use. When selected, press **Enter** to confirm.

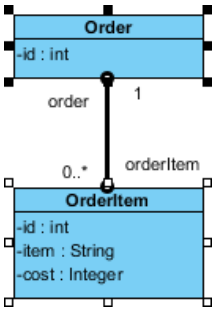
Reusing model element through copy and paste

You can copy view(s) and paste it at the destination diagram. Right click on the selected model element(s) in the source diagram and select **Copy > Copy with Visual Paradigm** from the pop-up menu.



Copy a model element

When you switch to the destination diagram, right click on the preferred place that you want the copied model element(s) to be pasted on and select **Paste View** from the pop-up menu.

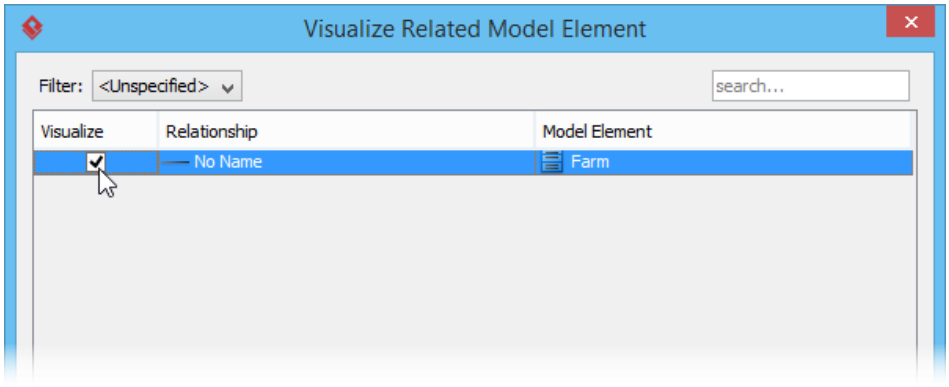


Class PrivateCar is pasted as a new view

NOTE: Choosing **Paste Model** means a new model element will be created and shown on the destination diagram.

The relationship between the model elements may not be shown on diagram. If you want their relationship to be revealed, right click on a view and select **Related Elements > Visualize Related Model Element...** from the pop-up menu.

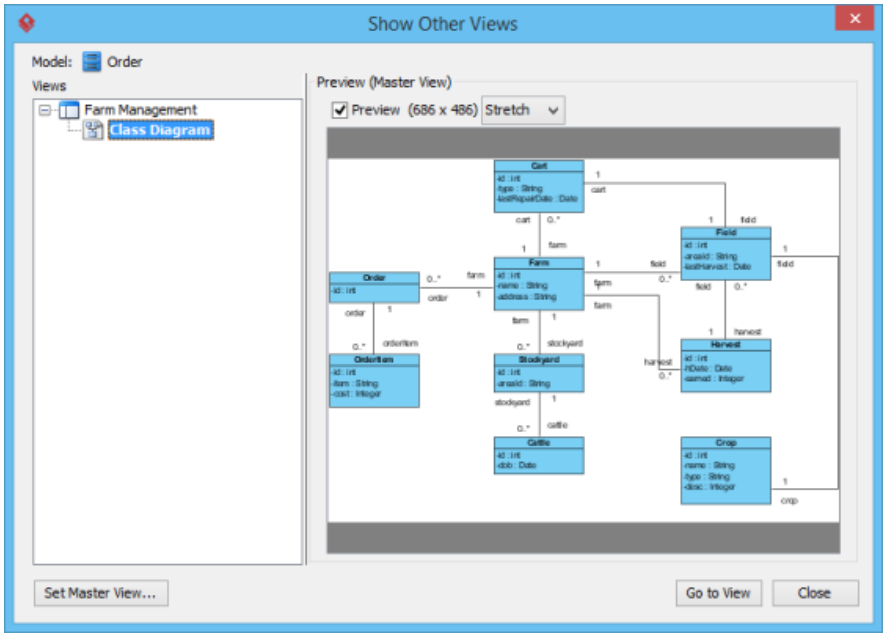
When the **Visualize Related Model Element** dialog box pops out, check a relationship for the model element(s) and then click the **Visualize** button to confirm.



To visualize an association between two classes

Opening other views

You can find out the source diagram of a model element by right clicking on it and select **Related Elements > Show Other Views...** from the pop-up menu.



The Show Other Views window

All diagrams which contain the same model elements you have selected are listed and shown in the **Show Other Views** window. You can check a diagram under **Views** and preview it under **Preview**; furthermore, click **Go to View** button to jump to and view the actual diagram.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

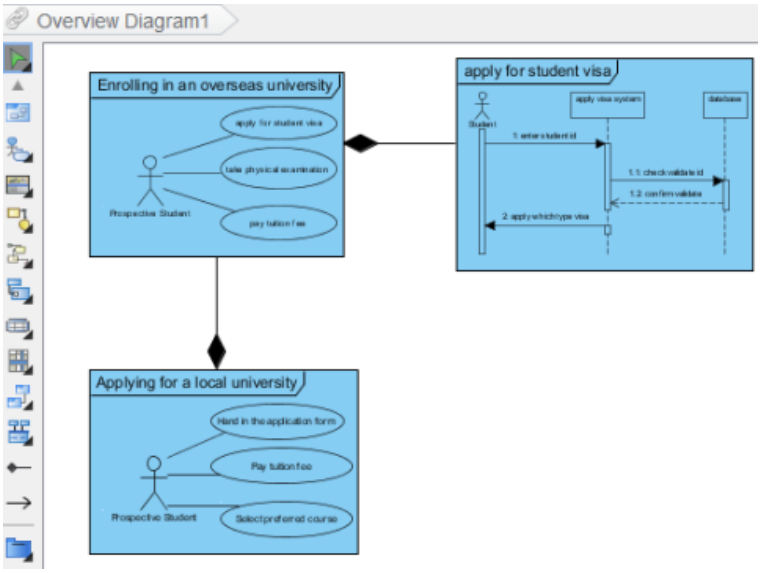
Using overview diagram

When various diagrams with different contexts are modeled in your system, an overview diagram can be created to illustrate their relationships related each other. With the overview diagram, you can have an overview of your system, after all, you can study their relationships. Since the overview diagram is consist of thumbnail of diagrams that you have created previously, the overview of each diagram if find to be related to another diagram can be linked up with connectors.

Creating an overview diagram

To create an overview diagram, Select **Diagram > New** from the toolbar. In the **New Diagram** window, select **Others > Overview Diagram**. Click **OK** to confirm.

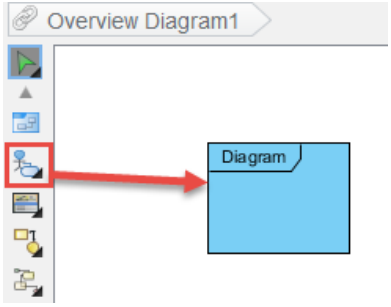
As a result, an empty overview diagram is created. Create overviews of diagrams and relate them to illustrate their relationship with each other.



The overview diagram

Creating an overview of new diagram

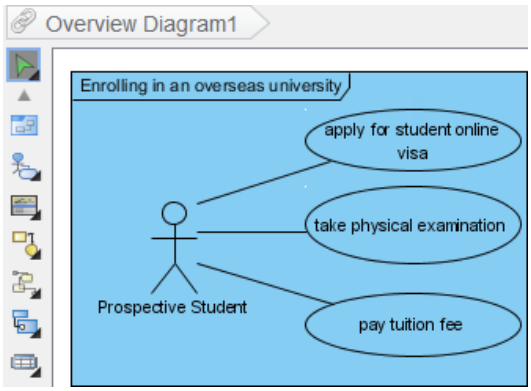
You can create a diagram from overview diagram. To create a new diagram from overview diagram, select your preferred diagram type from the diagram toolbar and drag it on the diagram pane.



Create a use case diagram overview

The new diagram will be opened subsequently. Enter a name for new diagram and start working on it.

If you go back to the overview diagram, the preview of newly created diagram can be seen. If you realize that the diagram in diagram overview is too small and vague for previewing, you may resize the diagram overview by dragging its bottom right corner.

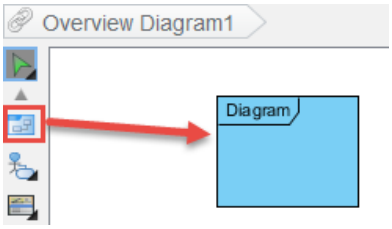


The newly created diagram is shown on overview diagram

Creating an overview of existing diagram

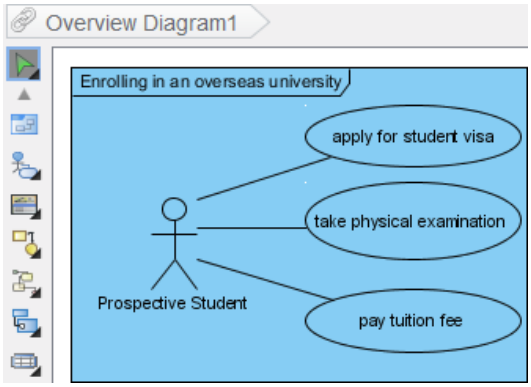
Besides creating an overview diagram of new diagram, you can also create an overview of existing diagram.

1. Create a diagram overview from the toolbar of overview diagram.



Create a diagram overview

2. Right click on the diagram overview and select **Associate to Diagram...** from the pop-up menu.
3. Select a diagram that you want to be shown on diagram overview and select **OK** button.
4. As a result, the selected diagram will be shown on diagram overview. If you realize that the diagram in diagram overview is too small and vague for previewing, you may resize the diagram overview by dragging its bottom right corner.

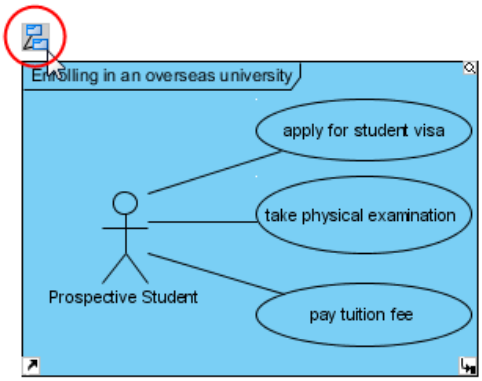


The selected diagram is shown

Visualizing sub-diagram

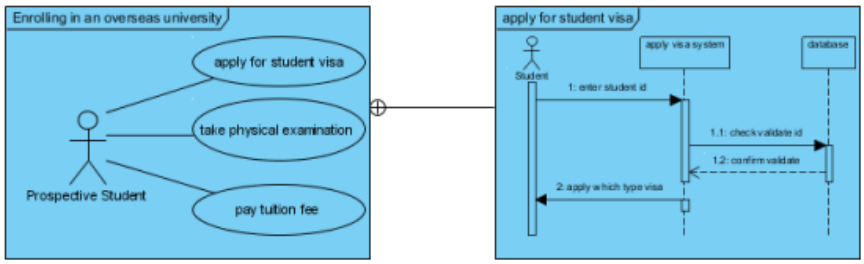
When a diagram overview you created has a sub-diagram, you can connect them together and make them shown on the current overview diagram.

Move the mouse over a diagram overview and click its resource icon **Visualize Related Diagrams**.



*Press **Visualize Related Diagrams***

As a result, the sub-diagram will be shown and both diagram overviews are connected.

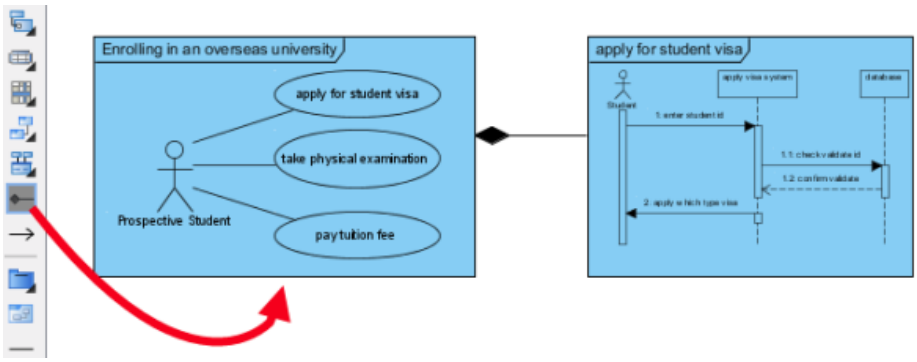


Two diagram overviews are connected

Selecting a connector for diagram overviews

In order to present a relationship between two diagrams, different connectors should be used. Diagram containment is used when a diagram contained is a part of the containing diagram overview while directional generic connector is used to model the sequence of diagrams, from advance to another.

Select a connector from the diagram toolbar and drag it from the source diagram overview to the destination diagram overview.



*Select **Diagram Containment***

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Changing model element type

As your project develops, requirements will be set, issues will be found, old requirements may no longer be valid, etc. Inevitably, all these lead to changes in your existing design. Since re-drawing means a big time investment, so, you would most likely refine your existing diagrams instead. In such case, the ability to convert a model element in the diagram from one type to another will be very useful.

When do You Convert a Model Element Type?

We support a rich set of modeling languages and notation. You can always find the best notation for thorough expression. When the type of an element is no longer appropriate or is not the best type to express an idea, you will want to replace it with a better notation. Here are some of the situations when type conversion is needed.

Design Evolvement

Take BPMN as an example. Task and sub-process are two notations introduced by BPMN. They both model activities to do within a business process. If you need to model an order processing system with BPMN, you may initially use a BPMN task to model the *Ship Order* activity. As the design evolves, let's just say that shipping order turns out to be much more complex than what vyou thought and involves several sub-activities like arranging shipment, delivering goods, collecting payment, etc. To depict that, you need to convert the atomic task to a sub-process, not only for representing its complex nature but also for making it possible to drill down the process and model its details.

Correcting Mistakes

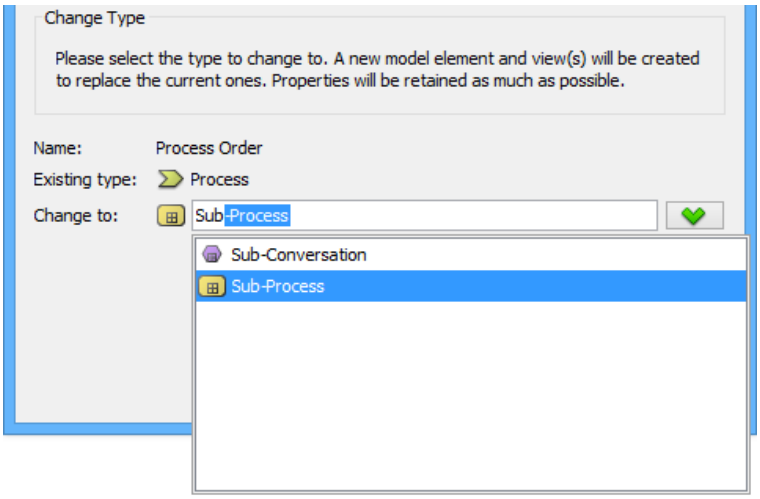
Things are never perfect and so are our work. Sometimes your knowledge in modeling language falls short. You may mistakenly apply a wrong type of model element in your design. For example, you could have incorrectly treated UML activity as UML action. When you realize it is a mistake, you want to correct it.

Describe More Precisely What You Want to Express

Some notations are sub-types of a more general one. For example, send-signal-action is a kind of UML action that creates a signal from its input and transmits it to the next object. While using UML action in a broad manner is not exactly wrong, using send-signal-action would reflect the truth more precisely.

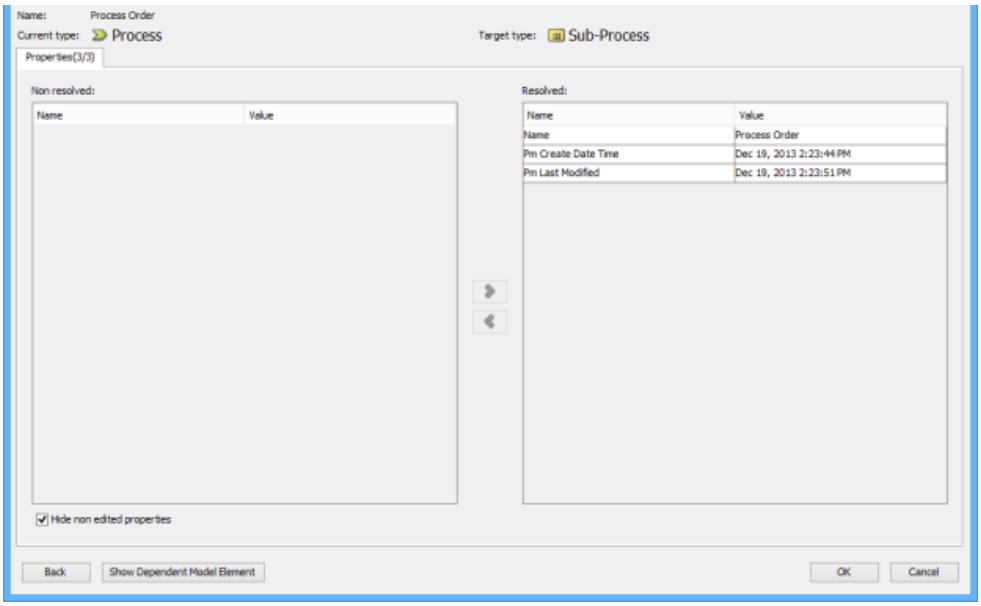
Changing Model Element Type

1. Right click on the desired shape and select **Related Elements > Change Type...** from the popup menu.
2. In the **Change Model Element Type** window, enter the type to convert to.



Entering the type to change to

3. Click **Next** to confirm the conversion.
4. Due to the difference in the element type, some properties, relationships and tagged values may no longer be compatible after the conversion and are forced to be discarded. In the Change Model Element Type window, you will find the properties, relationships and tagged values that will be kept and discarded after the conversion. The data that will be discarded is put on the left hand side while the resolvable data is on the right. Click **OK** at bottom right.



Change Model Element Type

5. Note that the undo history will be cleared after the conversion, meaning that you cannot perform any undo action after it. Click **Yes** to confirm the change and to return to the diagram.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting contained shapes with InstantFreeze

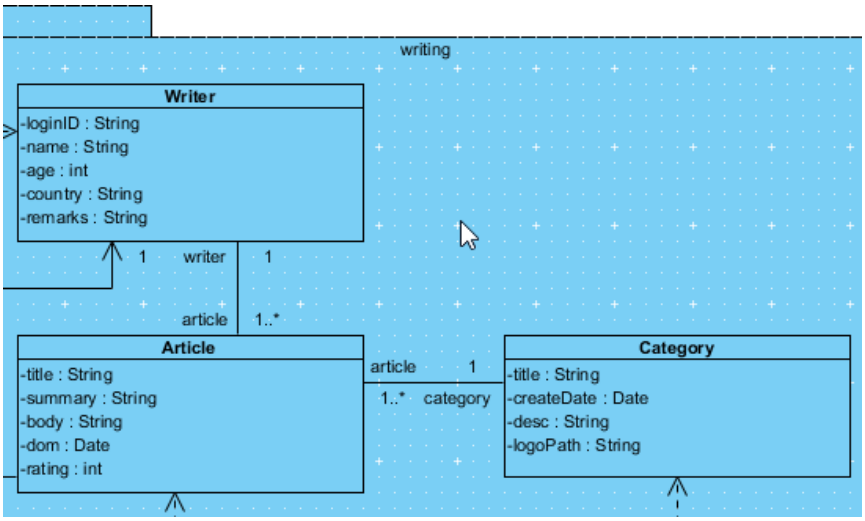
Imagine if there is a big, big container shape, such as a package shape, on a diagram. The shape is so big that spans over the viewing range. Inside the container shape, there are few child shapes and now, you want to move these shapes to somewhere else. You attempted to perform a ranged shape selection first but when you press on the container shape and start dragging, the package moves, following your mouse pointer movement. Here, in order to "tell" the application that you don't wantto move the shape in background, but to select shapes in foreground, use InstantFreeze.

InstantFreeze is a diagramming technique that allows you to temporarily freeze a container shape. By freezing a container shape, you cannot move and edit it until unfreeze. This allows you to select shapes inside the container shape as if the container shape does not exist.

Using InstantFreeze

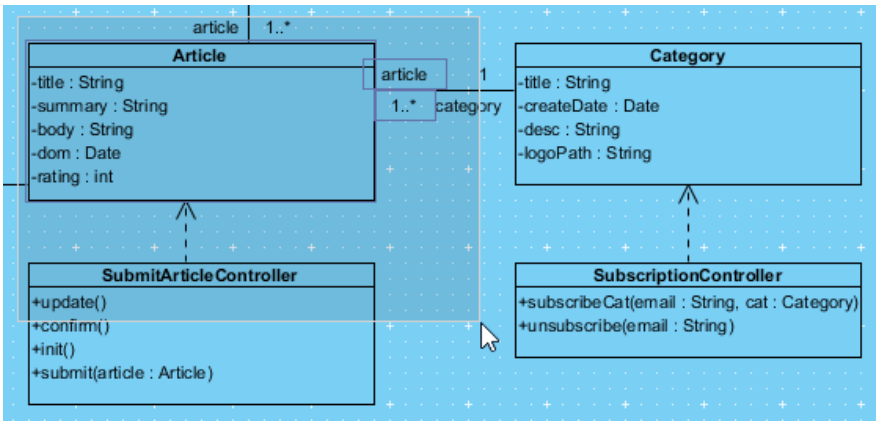
To temporarily freeze a container shape with InstantFreeze:

1. Move your mouse pointer over the container shape that you want to freeze.
2. Press twice. Note that it is NOT a double click. Instead, it is two separate presses, with a delay in between. When success, you can see the container shape painted with a grid background.



Using InstantFreeze

3. Now, you can select shapes freely inside the container shape without worrying about moving the container shape in background.



Selecting shapes

4. When you finish, unfreeze the container shape either by pressing the **Esc** key or by clicking once outside the container shape.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Annotations and freehand shapes

There are some types of shape which let you annotate shapes on diagrams. They include text annotation, callout and freehand shape.

UML note

A standard UML notation for commenting purpose.

Callout shape

A speech-bubble like shape made for annotation purpose. It's pointer can be adjusted to point to specific position.

Freehand shape

A set of shapes that can be bended to different styles to fit in different ways of annotation.

UML note

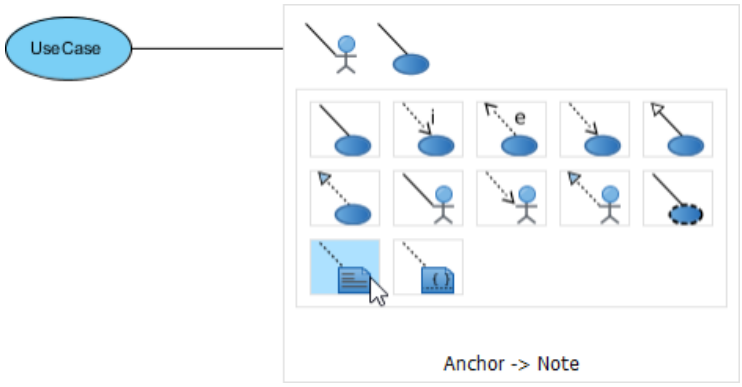
In Visual Paradigm, UML notes can be created and attached to a shape via an anchor connector. The premier advantage of using UML notes is to annotate a specific model element on the diagram with normal text or HTML text or to define OCL.

Creating a UML note with resource centric

1. Move the mouse over a shape.
2. Press on the **Resource Catalog** icon and drag it out.

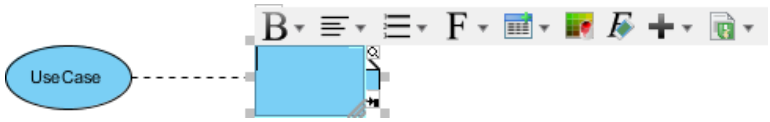
Mouse the mouse over a shape

3. Release the mouse button at the position where you want to create the note shape.
4. Select **Anchor -> Note** from Resource Catalog.



Drag the resource icon

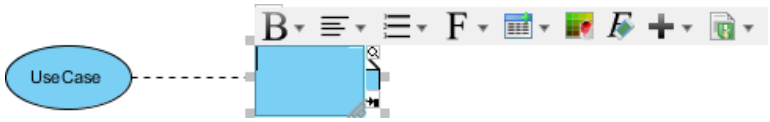
5. The note is created and is is connected with the shape.



Note created

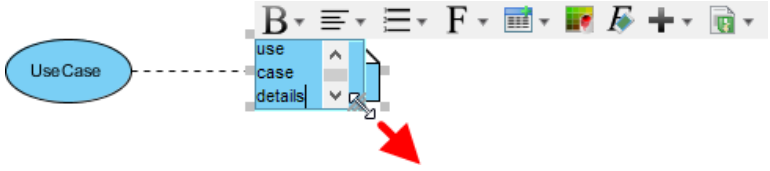
Editing UML note content

1. Double click the note to start editing its content. Note that the toolbar above the note can be used to format its content.



Start editing

2. Drag the bottom right corner of note to resize it.



Resizing

3. Click on the diagram background to confirm editing the note.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

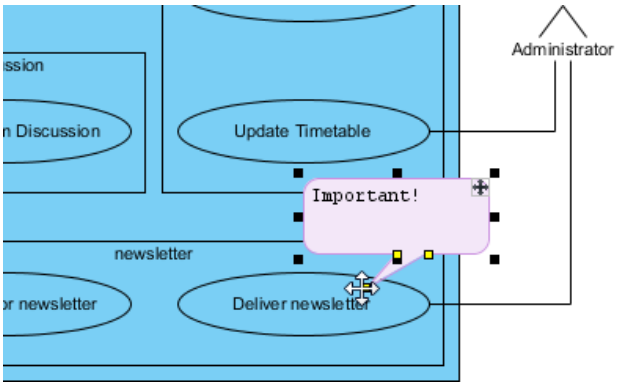
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Callout shape

Callout shape is a label that is used for explanation on your model elements. Inserting callout shape aims to draw others' attention and give them additional remarks. Basically, its function is similar to a photo caption or a comment. However, it does more than merely either a photo caption or a comment. In fact, it looks more likely to a dialog box that you can place it next to your model elements.

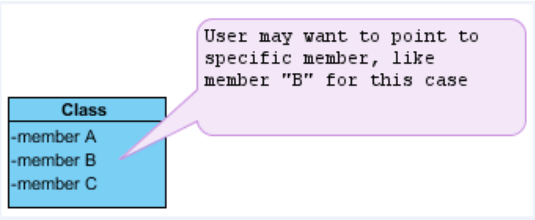
Adjusting the direction of callout shape pointer

1. A remark can be inserted by selecting **Callout** from the diagram toolbar and dragging it to the model elements directly on the diagram pane.
2. The direction of callout shape pointer can be adjusted by simply dragging the pointer's end.



Adjust the pointer

3. The pointer can be adjusted to point out a more specific position, such as pointing to the name of diagram element or a specific class member (attribute/operation) out of a class.

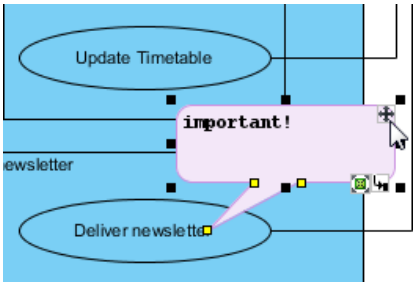


Adjust the pointer to a specific class member out of a class

> **NOTE:** You can adjust not only the pointer's direction, but also its length.

Repositioning the shape

The position of callout shape can be moved by dragging the **+** icon that is located on the top right corner of the callout. This icon is used to reposition the shape in a straight and simple way.



Reposition the shape

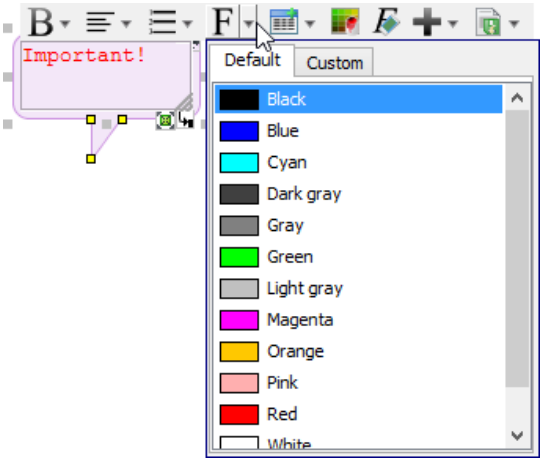
> **NOTE:** The callout shape will be moved but the position of its pointer won't be changed if you only drag the **+** icon. This helps you to remain the pointing and move the callout shape simultaneously.

Editing callout shape content

The content of callout shape can be edited by double clicking on the callout shape.

Emphasizing the content of callout shape

Formatting buttons can be used to insert formatting for the text in order to emphasize your key points while you are editing the content, such as changing font color. This helps to enhance the visual effect of the text as well.



Format the text

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

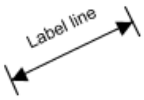
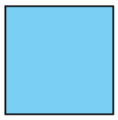
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Freehand shapes

Freehand shapes are multi-functional features in Visual Paradigm. With freehand shapes, you can not only draw various kinds of shapes, but also insert an annotation. If you want to stress an important message with visual effect, you may use word art rather than using a note or a callout because you can reshape the text in word art but not in a note or a callout. Moreover, the shapes in freehand are flexible that you can twist them freely, but you can only resize a note or a callout.

Freehand toolbar interface

Summary of freehand shapes

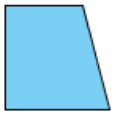
Shape Type	Sample
Pencil	
Pen	
Calligraphic Tool	
Connector	
Line	
Label Line	
Path Rectangle	
Rectangle	
Round Rectangle	

Round Rectangle 2

Diamond

Parallelogram

Trapezoid



Isosceles Trapezoid

Path Ellipse

Ellipse

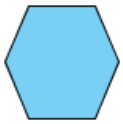
Arc

Chord

Pie

Regular Polygon

Isosceles Triangle

Hexagon	
Arrow	
Two Head Arrow	
Spiral	
Text	
Word Art	

Summary of freehand shape

Drawing free style path with pencil

1. Press on the empty space on diagram pane and drag to form the outline.

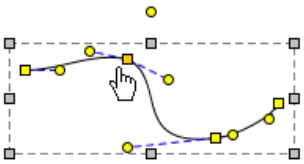
Drawing with pencil

2. Release the mouse and new freehand shape will then be created.

Activating the fine editing selector

Fine editing selector shows a second later after the freehand shape is being selected. To show it immediately, press keyboard ' **N**' key.

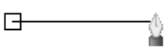
Press on a yellow selector for selecting and the selected selector will turn into orange. More fine editing selectors will appear for curve adjustment.



Selected fine editing selector

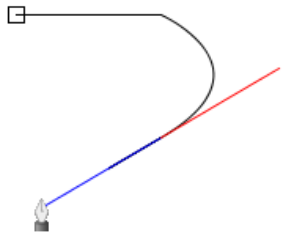
Drawing curve with pen

1. Click on empty space on diagram pane and drag it to create the first stroke.



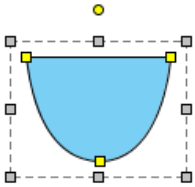
Straight line created

2. To create a curve, press and move the mouse. The indication line will appear. Release the mouse button when finishing editing. On the other hand, the last stroke can be cancelled by right clicking on the diagram.



Creating curve

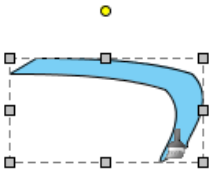
3. To confirm editing of the freehand shape, double click on diagram and a new freehand shape will be created. If the point returns to the starting point, it will form a closed path.



A close path

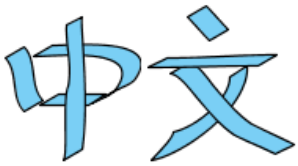
Drawing calligraphic path with calligraphic tool

1. Press on the diagram and drag to form the outline of shape. Release the mouse to create the shape.



Freehand shape created

2. By combining several other calligraphic shapes, you can create a complete diagram.



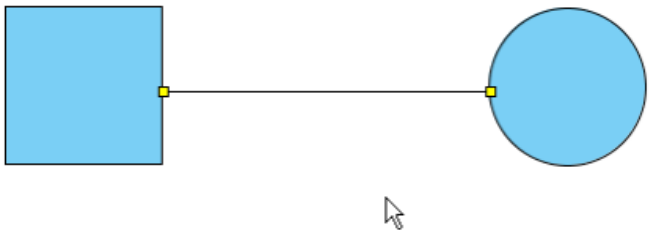
Calligraphy example

Draw straight and curve line with connector

1. Press on a source shape and drag it to the destination shape.

Connecting shapes

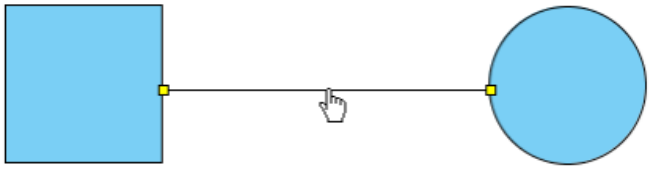
2. Release the mouse and a new connector will be created between them.



A line is created

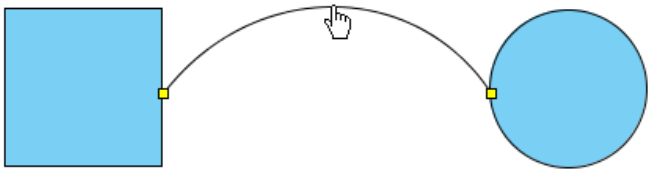
Bend a straight connector into a curve

1. Press on a straight connector.



Clicking on straight line

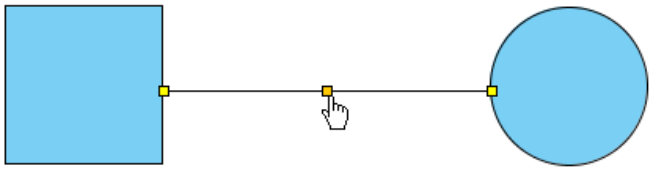
2. Bend it to your preferred direction and it will become a curve connector.



A curve connector

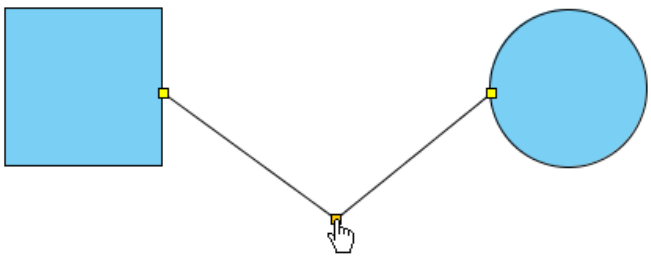
Split a straight connector

1. Press the **Ctrl** key.
2. Click on the specified location to split. A new point at where you have clicked will turn into orange.



Splitting line

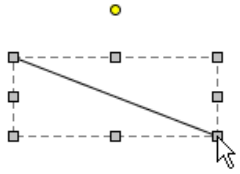
3. Drag on that point to split the line.



Moving mid point

Drawing straight and curved line

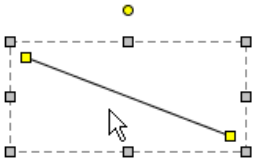
1. Press on the diagram pane and drag to form the outline.
2. Release the mouse button and a straight line will be created.



Line created

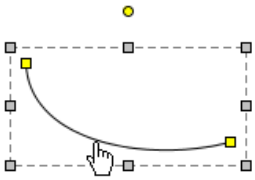
Bend a straight line into a curve

1. Select a straight line for a second to wait for the fine editing selectors popping out.



Showing fine editing selector

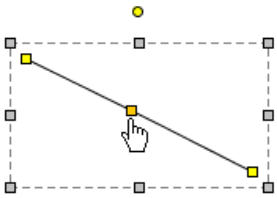
2. Press on the straight line. Drag it to bend into your preferred direction.



Dragging line as curve

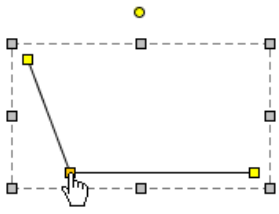
Split straight line

1. Press the **Ctrl** key.
2. Click on the specified location to split. A new point at where you have clicked will turn into orange.



Splitting line

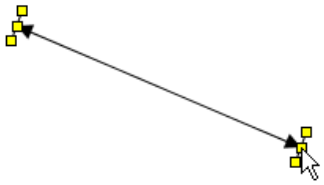
3. Drag on that point to split the line.



Moving mid point

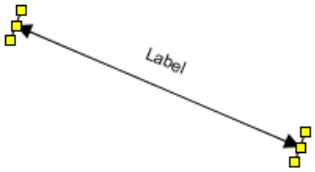
Drawing labelled line

1. Press on the diagram and starting dragging to form its outline.
2. Release the mouse button to create the labelled line.



Freehand shape created

3. Double click on the line. Enter the name for the line.
4. Press **Enter** to confirm editing.

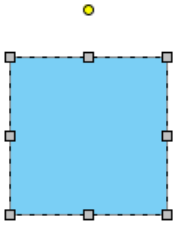


Label Line

5. You may drag the yellow selector to modify the line's outline.

Drawing rectangle

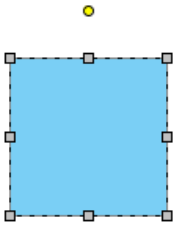
Click on the diagram to create a rectangle.



Freehand shape created

Drawing path rectangle

Click on the diagram to create a path rectangle.



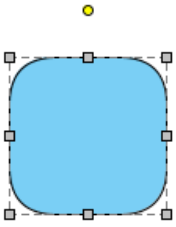
Freehand shape created

What's the difference between rectangle and path rectangle?

Path rectangle is formed by path, which enables you to freely reshape it, while rectangle always keeps shape as a rectangle.

Drawing rounded rectangle

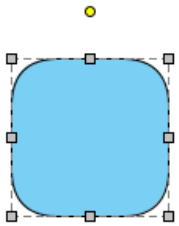
Click on the diagram to create a rounded rectangle.



Freehand shape created

Drawing rounded rectangle 2

Click on the diagram to create a rounded rectangle 2.



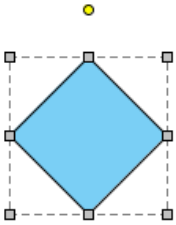
Freehand shape created

What's the difference between rounded rectangle and rounded rectangle 2?

Rounded rectangle uses a single control point to control the deepness of corner, which ensures that the four corners remain consistent while rounded rectangle 2 uses two points to control the deepness of corner, which can produce irregular corners.

Drawing diamond

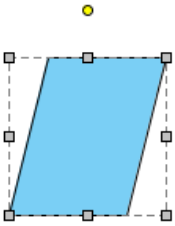
Click on the diagram to create a diamond shape.



Freehand shape created

Drawing parallelogram

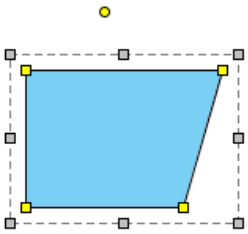
Click on the diagram to create a parallelogram shape.



Freehand shape created

Drawing trapezoid

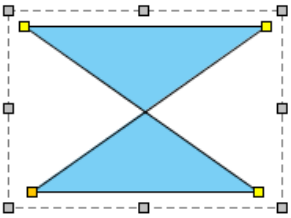
1. Click on the diagram to create a trapezoid shape.
2. You can adjust the slope by dragging the fine editing selectors in yellow.



Other trapezoid outline

Drawing isosceles trapezoid

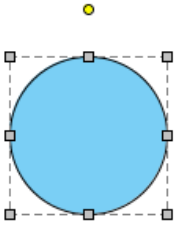
1. Click on the diagram to create an isosceles trapezoid shape.
2. You can reshape the Isosceles Trapezoid by dragging the fine editing selectors in yellow.



Other isosceles trapezoid outline

Drawing ellipse

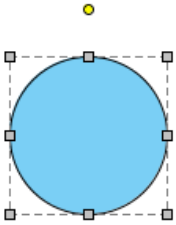
Click on the diagram to create an ellipse shape.



Freehand shape created

Drawing path ellipse

Click on the diagram to create a path ellipse shape.



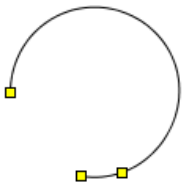
Freehand shape created

What's the difference between ellipse and path ellipse?

Path ellipse is formed by path, which enables you to freely reshape it while ellipse always keeps shape as an oval.

Drawing arc

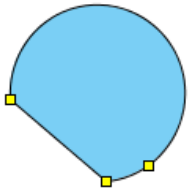
1. Click on the diagram to create an arc shape.
2. You can extend the line by dragging on the fine editing selectors in yellow.



Other arc outline

Drawing chord

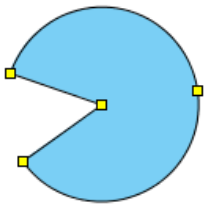
1. Click on the diagram to create a chord shape.
2. You can extend the arc of chord by dragging on the fine editing selectors in yellow.



Other Chord outline

Drawing pie

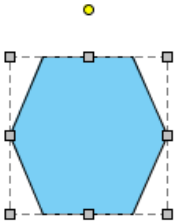
1. Click on the diagram to create a pie shape.
2. You can extend the arc of pie by dragging on the fine editing selectors in yellow.



Other pie outline

Drawing hexagon

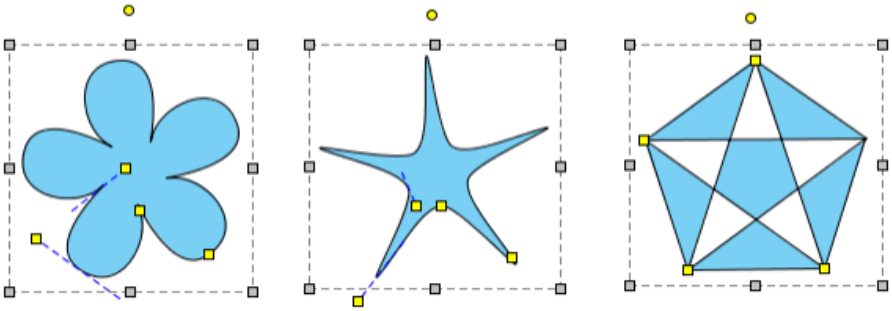
Click on the diagram to create a hexagon shape.



Freehand shape created

Drawing regular polygon

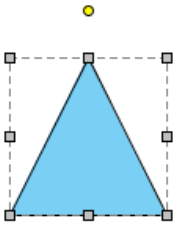
1. Click on the diagram to create a regular polygon shape.
2. You can modify the outline of shape by dragging the fine editor selectors in yellow.



Other regular polygon outline

Drawing isosceles triangle

Click on the diagram to create an isosceles triangle shape.

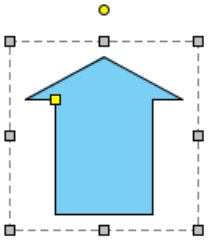


Freehand shape created

Drawing single head arrow

1. Click on the diagram to create an arrow shape.

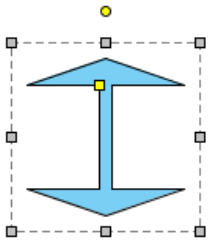
2. You can reshape it by dragging the fine editing selectors in yellow.



Other Arrow Outline

Drawing two head arrow

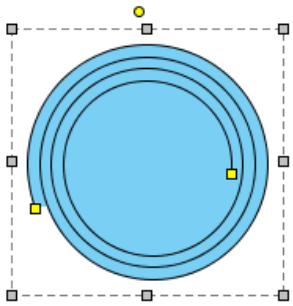
1. Click on the diagram to create a two head arrow shape.
2. You can reshape it by dragging the fine editing selectors in yellow.



Other two head arrow outline

Drawing spiral

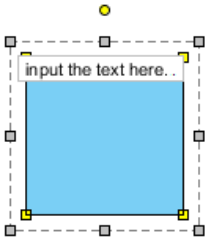
1. Click on the diagram to create a spiral shape.
2. You can reshape it by dragging the fine editing selectors in yellow.



Other Spiral outline

Inserting text

1. Click on the diagram to create a text shape, and input the text. You can press **Enter** to insert line break.



Input the text in text shape

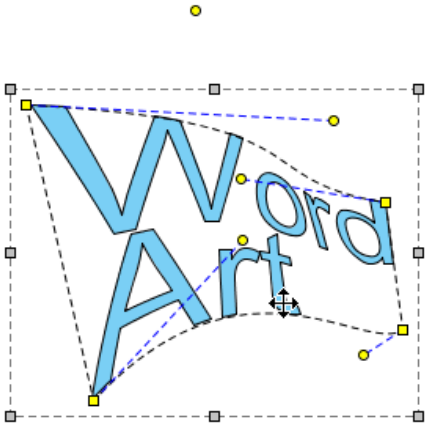
2. You can click **Ctrl** while pressing **Enter** to confirm editing.

Inserting word art

1. Click on the diagram to create a word art shape, and input the text. You can press **Enter** to insert line break.
2. You can click **Ctrl** while pressing **Enter** to confirm editor.

Freehand shape created

3. You can reshape it by dragging the fine editing selectors in yellow.



Editing word art

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Resource referencing

To include more information you can link to both internal and external material as reference. In this chapter, you will see how to refernece to shape, diagram, external file, folder and URL, as well as how to add sub-diagram.

Reference to external resources

File, folder and URL can be attached to a shape as references. This page will teach you how to do.

Reference to diagrams and shapes

In addition to external reference, internal reference can be added, too. You will learn how to refernece to another shape and diagram.

Elaborating model element with sub-diagram

Sub-diagram helps you describe a model element in detail by making use of a separate diagram.

Showing sub-diagrams and reference indicators

To indicate that a shape has sub-diagram or reference added, you can show the indicators on diagram. The indicators will show in exported image, too.

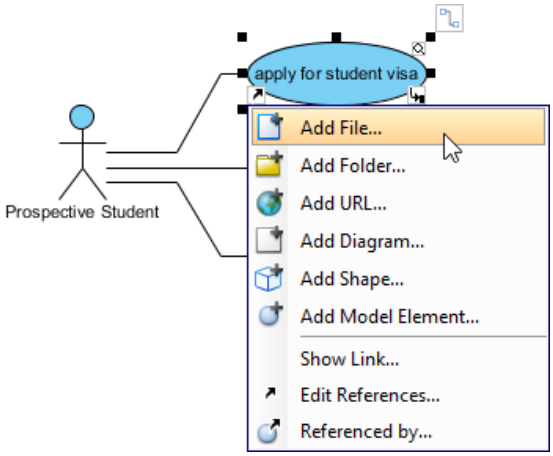
Reference to external resources

Additional references can be attached to a shape through resource icon **References**, such as inserting a file, a folder and a URL. After that, you can open and view the inserted references through resource icon **References**.

Adding external resources

Adding a file reference

1. Move the mouse over a shape to add reference, click the resource icon **References** and select **Add File...** from the pop-up menu.

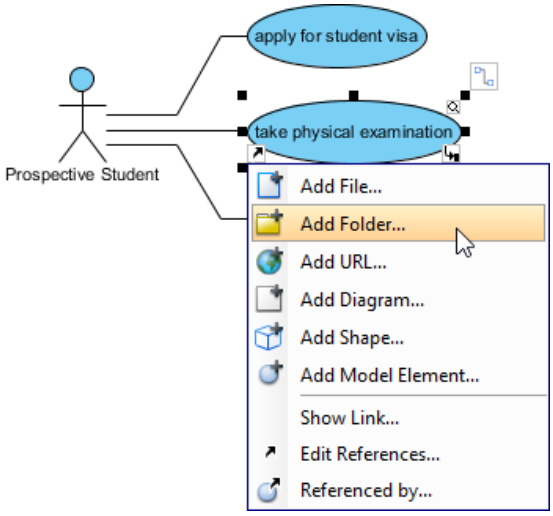


Click Add File...

2. In the **Select File** window, select the file(s) to reference to and click **OK**. Multiple file selection can be made by pressing **Ctrl** or **Shift** key. You may also enter the description for the file in **Description** field.

Adding a folder reference

1. Move the mouse over a shape to add reference, click the resource icon **References** and select **Add Folder...** from the pop-up menu.

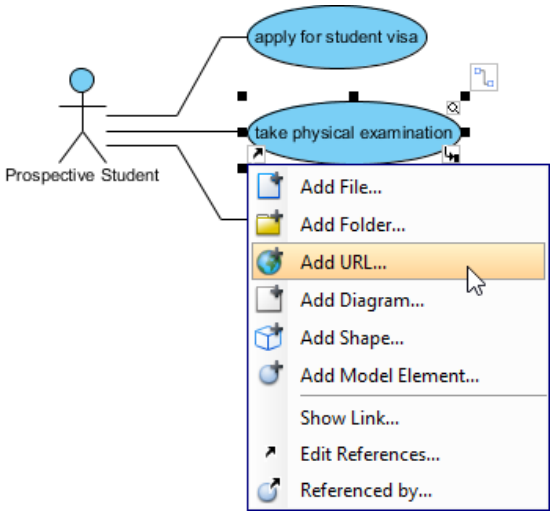


Click Add Folder...

2. In the **Select Folder** window, select the folder to reference to and click **OK**. Multiple folder selection can be made by pressing **Ctrl** or **Shift** key. You may also enter the description for the folder in **Description** field.

Adding a URL reference

1. Move the mouse over a shape to add reference, click the resource icon **References** and select **Add URL...** from the pop-up menu.



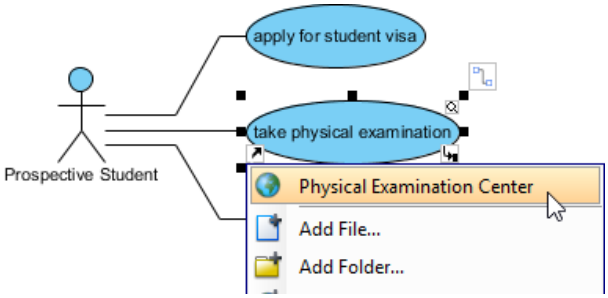
Click Add URL...

2. In the **Add URL** window, enter the URL, a brief description as its name and click **OK**. You may also enter the description for the URL in **Description** field.

Opening external resources

Move the mouse over a shape to open reference, click the resource icon **References** and select an external resource from the pop-up menu.

If you select a URL reference to open, it will be opened by default web browser. If you select a file reference to open, it will be opened by your system with the program used to open this kind of file. If you select a folder reference to open, it will be opened by your system automatically.



Open a URL reference

Editing references

1. Move the mouse over a shape and press it resource icon **References** and select **Edit References...** from the pop-up menu.
2. When the specification dialog box pops out, double click on the row of reference you want to enter its description or modify it.
3. Enter the description or modify it under **Description** column.
4. Finally, click **Enter** button to confirm editing.

Removing a reference

1. Move the mouse over a shape which has references, click its resource icon **References** and select **Edit References...** from the pop-up menu.
2. When the specification dialog box pops out, select a reference to be removed on the list and press **Remove** button to delete the selected reference.
3. Finally, click **OK** button to confirm the reference removal.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

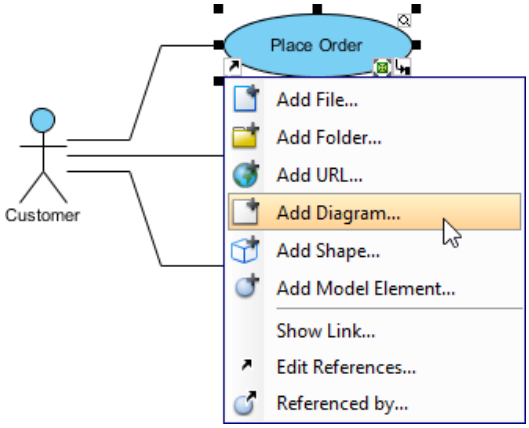
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reference to diagrams, shapes and model elements

Additional references can be attached to a shape through resource icon **References**, for example, inserting a diagram, a shape and a model element. After that, you can open and view the inserted references through resource icon **References**.

Reference to diagrams

1. Move the mouse over a shape, press its resource icon **References** and select **Add Diagram...** from the pop-up menu.



To add a diagram reference

2. In the **Select Diagram** window, select the diagram(s) to reference to and click **OK**. You may also enter the description for the diagram references in **Description** field.

Diagram Type vs Diagram Hierarchy view

The gear button at the top left of the **Select Diagram** window allows you to click on it to select a diagram view, either **Diagram Type** or **Diagram Hierarchy**. For **Diagram Type** view, diagrams are grouped by the types they belong to. For **Diagram Hierarchy** view, diagrams are grouped based on the model hierarchy. For example, if business process diagram D1 contains a sub-process S1, and S1 contains another diagram D2 as sub-diagram. By choosing **Diagram Hierarchy** view, you will see D2 appeared as a child node of D1.

Reference to shapes

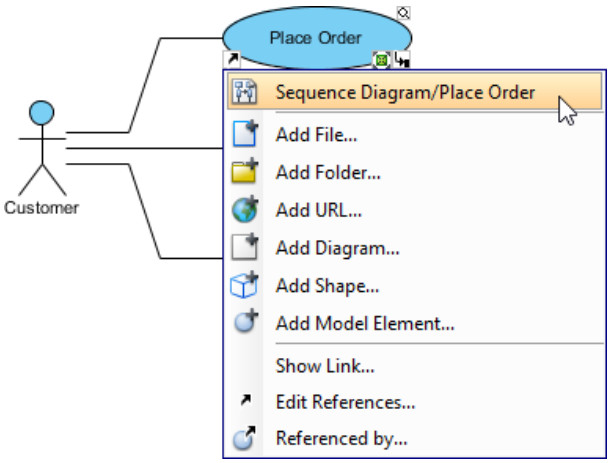
1. Move the mouse over a shape, press its resource icon **References** and select **Add Shape...** from the pop-up menu.
2. In the **Select Shape** window, select the shape(s) to reference to and click **OK**. You may also enter the description for the shape references in **Description** field.

Reference to model elements

1. Move the mouse over a shape, press its resource icon **References** and select **Add Model Element...** from the pop-up menu.
2. In the **Select Model Element** window, select the model element(s) to reference to and click **OK**. You may also enter the description for the element references in **Description** field.

Opening a reference

Move the mouse over a shape. Click its resource icon **References** and select a reference to open it. If you select a shape to open, it will switch to the diagram where the shape belongs to and the shape will be selected by filled-selector. If you select a diagram to open, it will switch to the selected diagram immediately. If you select a model element to open, it will open the specification of that model element.



Open a reference

Editing references

1. Move the mouse over a shape and press it resource icon **References** and select **Edit References...** from the pop-up menu.
2. When the specification window pops out, double click on the row of reference you want to enter its description or modify it.
3. Enter the description or modify it under **Description** column.

4. Finally, press **Enter** to confirm editing.

Removing a reference

1. Move the mouse over a shape which has references, click its resource icon **References** and select **Edit References...** from the pop-up menu.
2. When the specification window pops out, select a reference to be removed on the list and press **Remove** button to delete the selected reference.
3. Finally, click **OK** button to confirm the reference removal.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram-Based Reference Mapping Editor

You can add reference material such as an external file, a folder, a URL or even a model element or diagram to your design. If you use 'reference' frequently in your design, the coupling between your design and the referenced 'things' will be increased over time, making it uneasy to manage those references, nor to trace the usage of referenced material.

The Reference Mapping editor provides you with an overview of references added to a design. It is a diagram-based editor that lists the diagram itself and the containing elements on one side, and the referenced material on the other side, with mapping connectors linking between the model data and referenced material. With the Reference Mapping editor you can have a big picture of references used. You can also add new references, or to edit existing references.

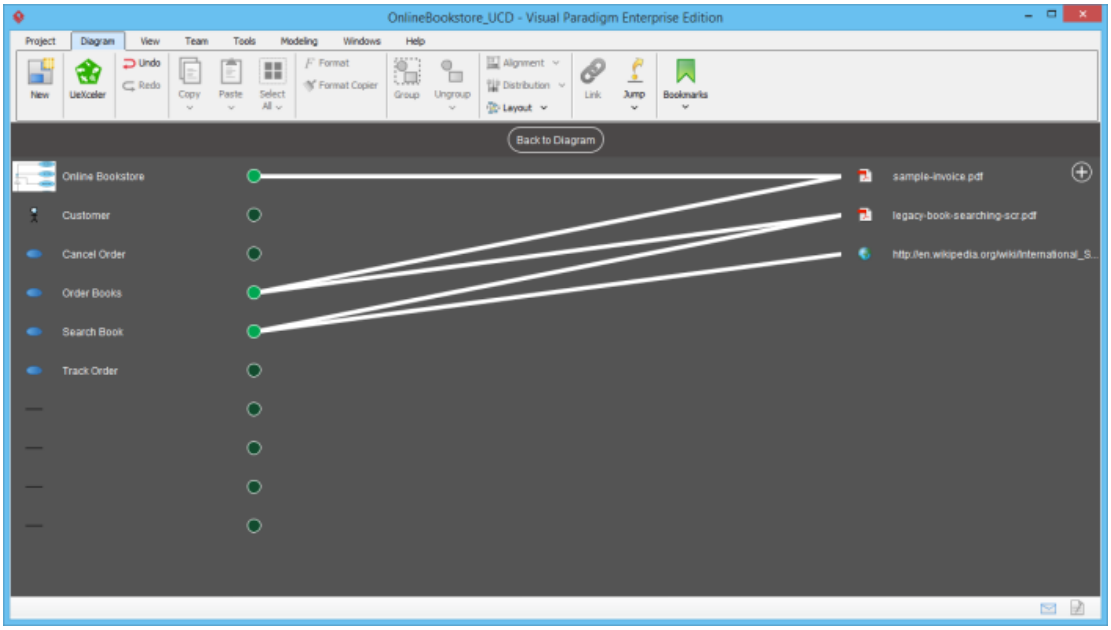
Opening the Reference Mapping Editor

To open **Reference Mapping** editor from a diagram, right click on the background of the diagram and select **Reference Mapping** from the popup menu.

Understanding the Reference Mapping Editor

The Reference Mapping editor consists of three main parts:

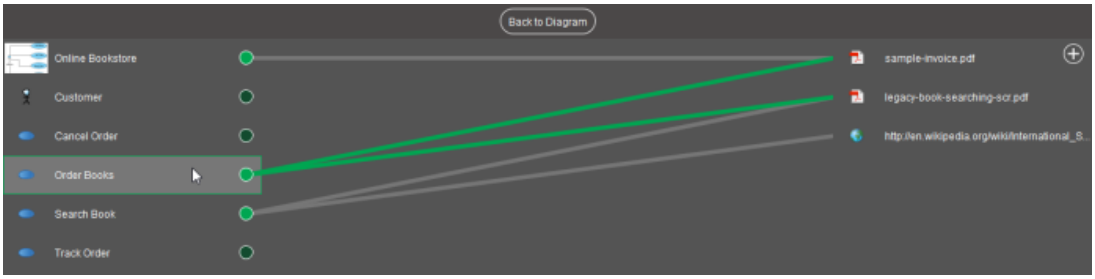
- Model Data - The diagram from which you open the Reference Mapping Editor as well as its containing diagram elements are listed on the left hand side.
- References - The file, folder, URL, diagram, shape and model element being added to any model data as references are listed on the right hand side. You can add extra reference material manually if you want to add references. For details, read the next section.
- Reference Mapping - References between model data and reference material are represented by mapping connectors that appear on the middle of the editor.



Reference Mapping editor

Knowing the Reference Material Added to a Model Element

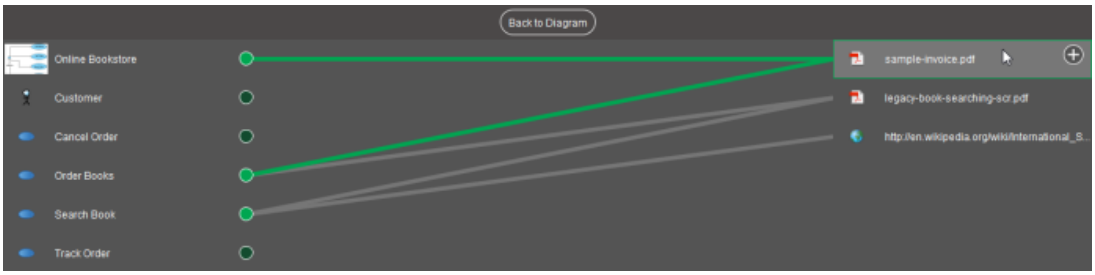
If you want to know the reference material added to a model element, click on the model element on the left hand side. Its references will be highlighted in green.



Knowing the Reference Material Added to a Model Element

Knowing the Utilization of a Piece of Reference Material

If you want to know the utilization of a piece of reference material, click on the reference material on the right hand side. The elements that reference the selected reference material will be highlighted in green.

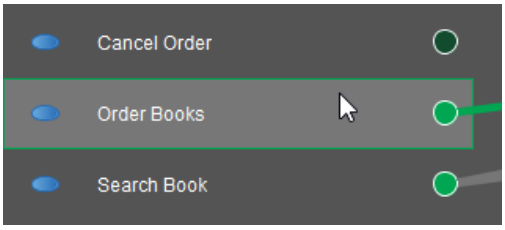


Knowing the Utilization of a Piece of Reference Material

Adding a Reference

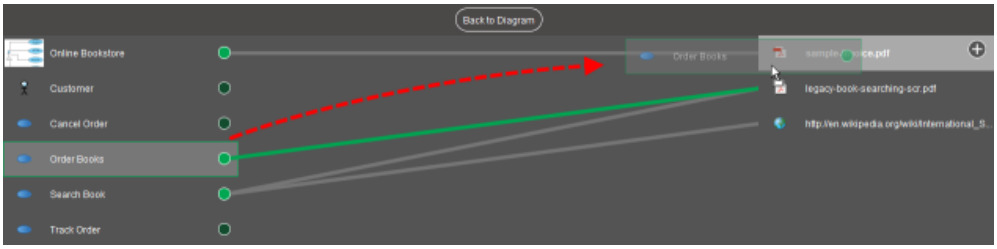
To add a reference to model data:

1. Select the model data on the left hand side. If you want to add reference to multiple items, press **Ctrl** or **Shift** for a multiple selection.



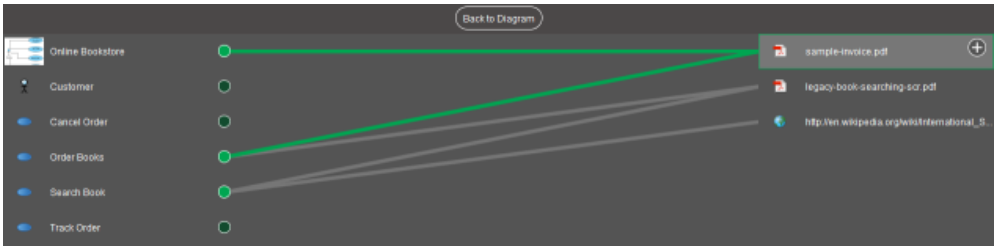
Selecting a use case to add reference to

2. Hold your mouse button and drag to the target reference material for creating a reference.



Adding a reference

3. Release the mouse button to create the reference. You will see a connector added between the selected model data and reference material.

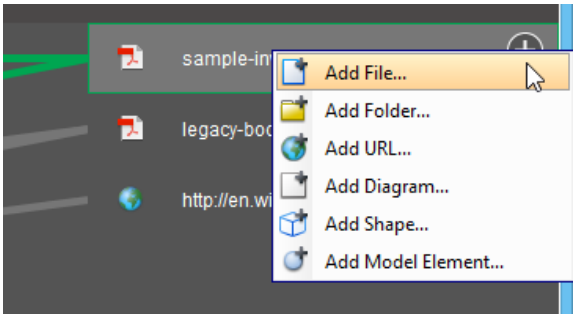


Reference added

Adding a Reference Material

You may want to add reference to a material that is not currently listed on the right hand side. In such case, you have to add the material manually first, and then add reference. To add a reference material:

1. Click the add button at the top of the reference material list on the right hand side.



Adding a file reference

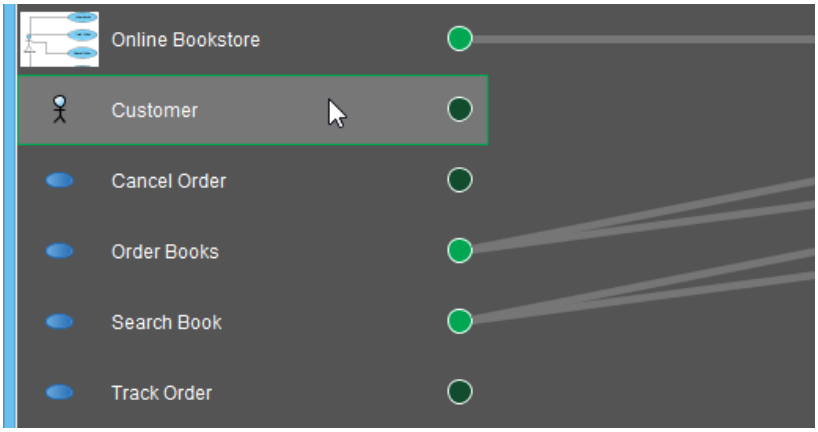
2. Select the kind of material to add from the popup menu.

3. Confirm your choice.

Re-ordering Model Data and Reference Material

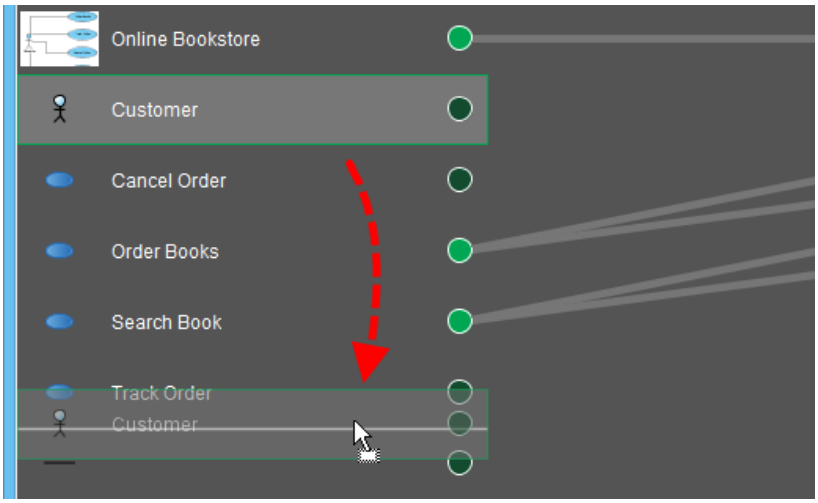
You can re-order model data and reference material to reduce the amount of crossing between mapping connectors. To re-order model data and reference material:

1. Select the model data or reference material. You can press **Ctrl** or **Shift** for a multiple selection.



Selecting an actor to re-order

2. Drag your selection to the desired position within the list.

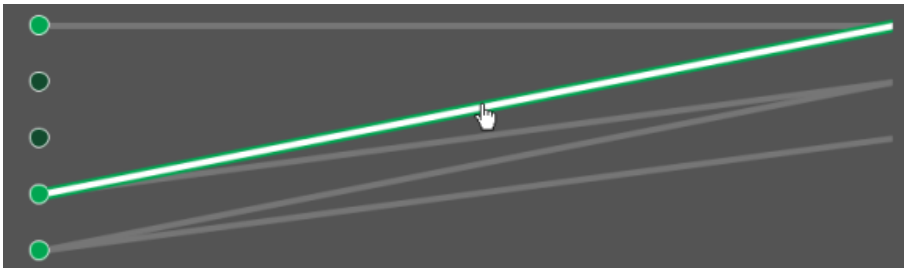


Re-order an actor

3. Release the mouse button.

Deleting References

To delete a reference, select it and press the **Delete** key to delete it. Again, you can delete multiple references at a time by pressing **Ctrl** or **Shift** key to select multiple references.



Selecting a reference to delete

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

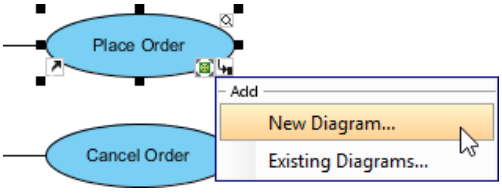
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Elaborating model element with sub diagram

Each notation has its own meaning and semantic. For example, you use a use case to present users' goals (system functions) but not how to achieve the goals. In order to model other aspects like the dynamic behavior of use case, you can elaborate a model element with a proper type of sub-diagram and contribute the details on the model element.

Creating a new sub diagram

1. Move the mouse over a shape, press its resource icon **Sub Diagrams** when it reveals and select **New Diagram...** from the pop-up menu.



Add a sub-diagram

2. In the **New Diagram** window, select the type of diagram to create. Note that the diagram types that are suitable for the selected shape are listed at the top of the window.
3. Enter the diagram name.
4. Click **OK**.

> **NOTE:** Inserting a sub diagram on a model element, all child model elements of the sub diagram will also be attached.

Adding an existing diagram as sub diagram

1. Move the mouse over a shape, press its resource icon **Sub Diagrams** when it reveals and select **Existing Diagram...** from the pop-up menu.
2. In the **Add Sub Diagrams** window, select the diagram(s) to add as the sub-digram of the selected shape.
3. Click **OK**.

Removing sub diagram

1. Move the mouse over a shape, press its resource icon **Sub Diagrams** and select **Manage...** from the pop-up menu.
2. In the specification window, select a sub diagram that you want it to be removed from the list and click **Remove**. When the **Confirm Remove** dialog box pops out, click **Yes** button to confirm the deletion.
3. Finally, click **OK** button to proceed.

Related Resources

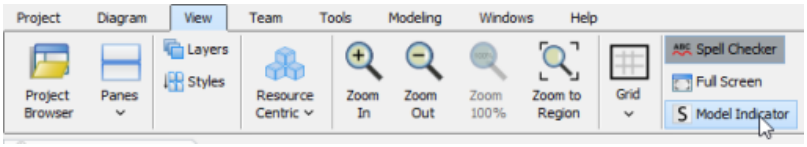
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Showing model indicator

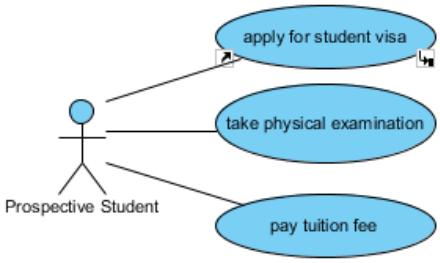
The indicator of sub-diagrams and references helps you to identify which shapes in diagram have sub-diagrams and/ or attached with references. To show the sub-diagrams resource icon:

1. Select **View > Model Indicator** from the toolbar.



Click Model Indicator button

2. As a result, references resource icon and sub-diagrams resource icon are shown.



References resource icon and sub-diagrams resource icon are shown

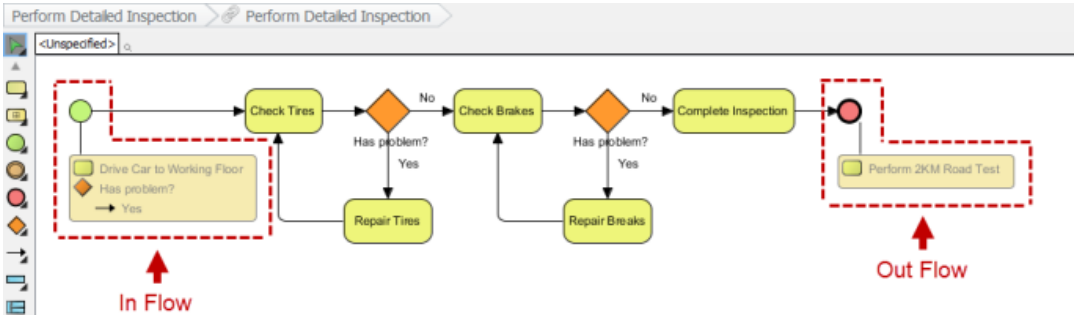
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

In/Out Flows in Sub-Diagram

Diagrams types like Business Process Diagram (BPD), (UML) State Machine Diagram and (UML) Activity Diagram are mainly for modeling 'flows'. You draw BPD for representing business workflow, state machine diagarm for representing state transition and activity diagram for representing flow of system actions.When you create sub-diagrams from these diagram types, you probably want the flow to continue flowing from the parent diagram to sub-diagram and from the sub-diagram back to the parent diagram. In order to keep the flow flows, you can make use of the in and out flow objects in sub-diagram.



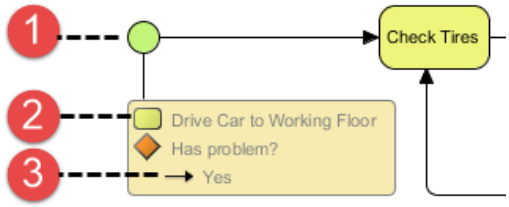
In Flow and Out Flow objects in a sub-diagram (BPD Example)

Instead of creating one yourself, the in/out flow objects will be created for you automatically in the following situations:

- Business Process Diagram - Creating a sub process BPD from a Sub-Process
- State Machine Diagram - Creating a sub state machine (diagram) from an Submachine state
- Actiivty Diagram - Creating Activity/State Machine as the 'behavior' of an Action

By performing any of these steps, you will see the in/out flow objects in the sub-diagarm, provided that a flow exist.

Reading In/Out Flows



An in flow

No.	Description
1	The initial node of the sub-diagram, which is conceptually connected from the parent diagarm.
2	The model element that connects to the parent element of the sub-diagarm.
3	The connector that connects to the parent element of the sub-diagram.

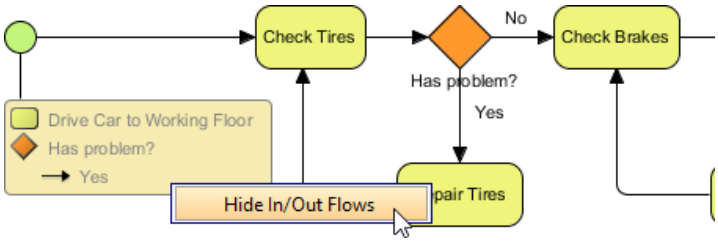
Description of an in flow

Modeling with In/Out Flows

To represent that a flow flows from the parent diagarm to content in sub-diagarm, simply connect the initial node with the sub-diagarm content. When the flow ends, connect the final element with the end node.

Hiding/Showing In/Out Flows

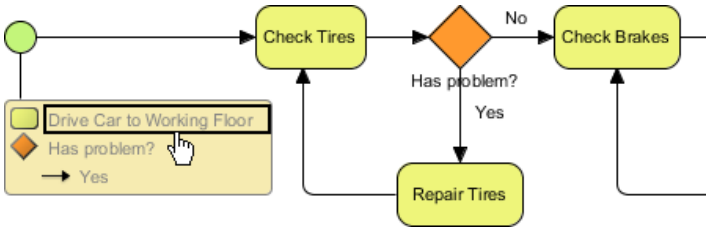
To hide in/out flows in sub-diagram, right click on it and select **Hide In/Out Flows** from the popup menu.



Hiding In and Out Flows

Going back to the parent diagram

To go back to the parent diagram, simply click on the model element link inside the in/out flow, like this:



Go back to parent diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using shape editor

Shape editor is a diagramming tool for you to design your own notation (stencil). In this chapter, you will learn how to make use of shape editor to create your own shapes.

Creating shape in shape editor

Teaches you how to start shape editor and create a shape in it.

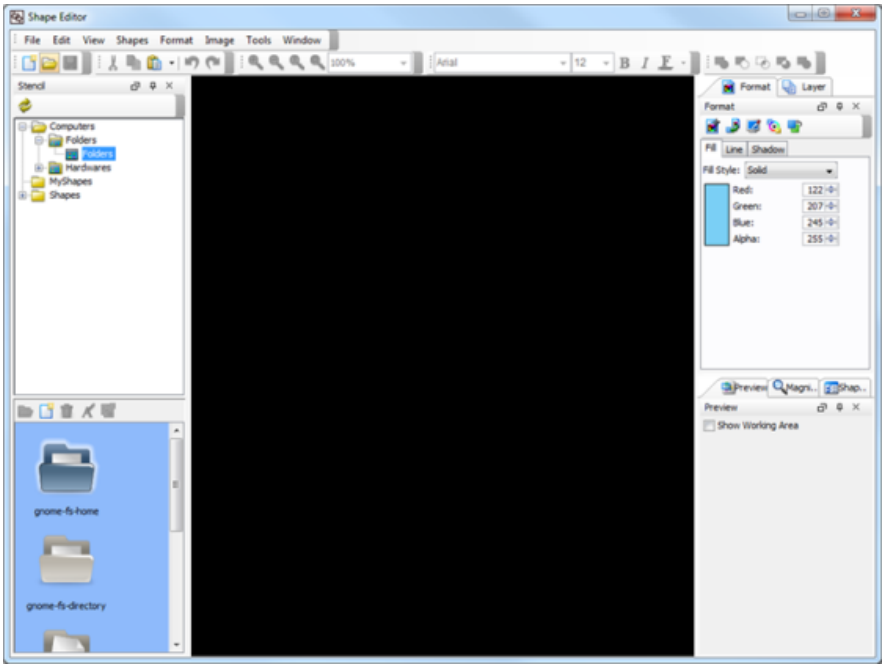
Creating shape from stencil pane

Shows you how to create a shape from a stencil.

Creating shape in shape editor

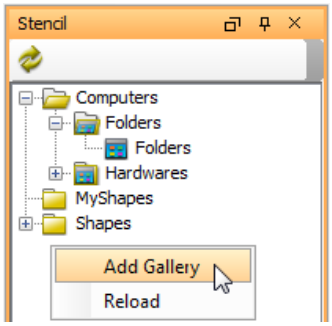
Although UML and BPMN are well-established notations, sometimes they still not rich enough to express domain specific idea. Shape Editor is a diagramming tool for you to design your own notation (stencil). Notations created can be incorporated into diagrams in Visual Paradigm. To use shape Editor:

1. To launch Shape Editor, select **Windows > Integration > Shape Editor...** from the toolbar.
2. When shape editor unfolds, you can create a shape by creating a gallery in advance. It is because a shape needs to be created under a stencil, while a stencil is put under a category of a gallery.



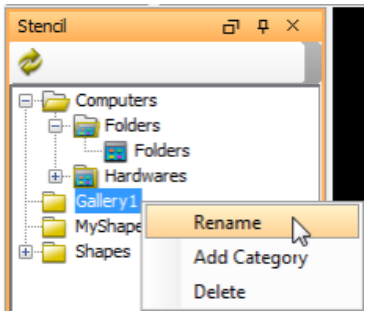
Shape Editor

3. A shape need to be created under a stencil, while a stencil is put under a category, under a gallery. To create a gallery, right click on the **Stencil** pane and select **Add Gallery** in the po-pup menu.



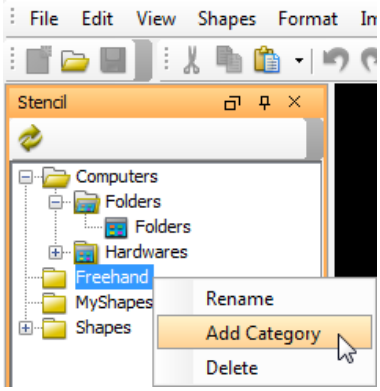
Add a gallery

4. To create a stencil, right click on a category and select **Add Stencil** from the pop-up menu. The newly created gallery is named as *Gallery1* by default. If you want to rename it, right click on it and select **Rename** from the pop-up menu. Enter your preferred gallery name in the pop-up **Input** dialog box and press **OK** button to confirm.



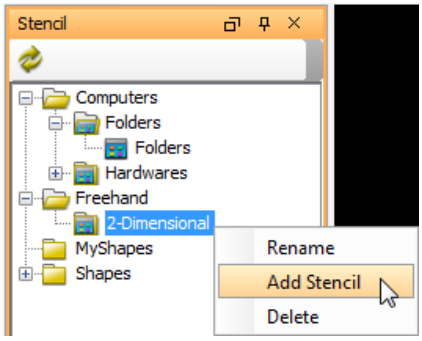
Naming a gallery

5. To create a category, right click on a gallery and select **Add Category** in the pop-up menu. Enter the category name in the pop-up **Input** dialog box and click **OK** button to confirm.



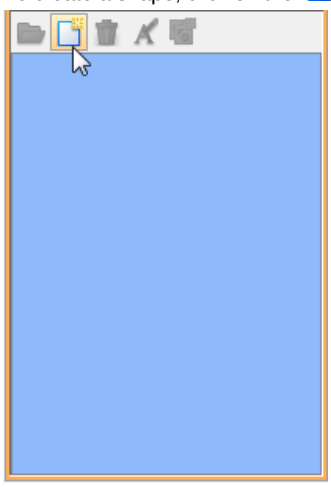
Add a category

6. To create a stencil, right-click on a category and select **Add Stencil** from the pop-up menu. The newly created stencil is named as *Stencil 1* by default. If you want to rename it, right click on it and select **Rename** from the pop-up menu. Enter your preferred stencil name in the pop-up **Input** dialog box and press **OK** button to confirm.



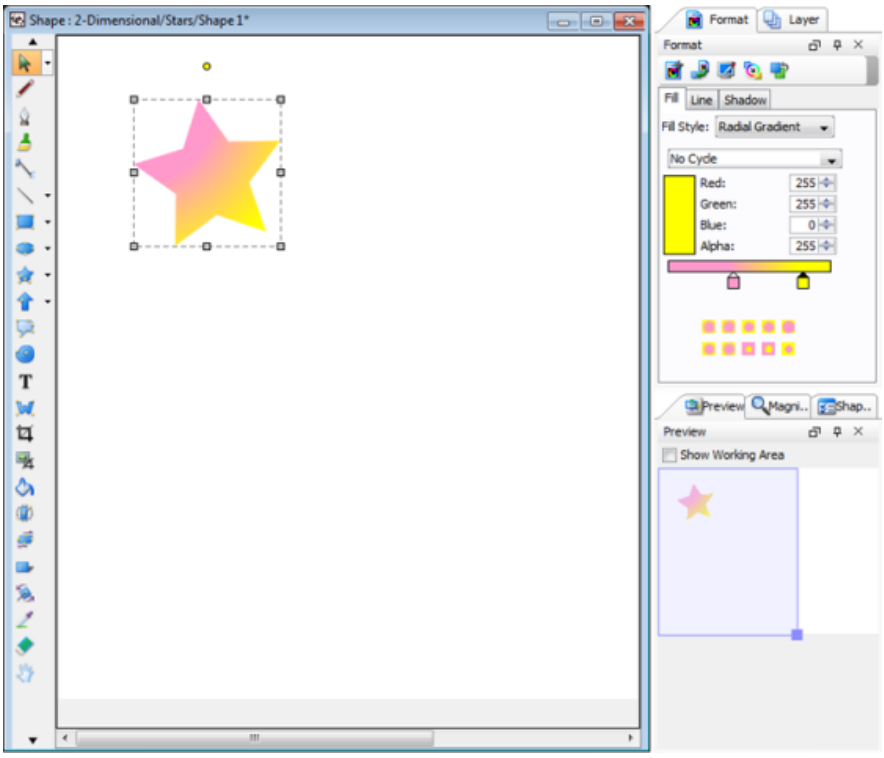
Add a stencil

7. To create a shape, click on the (**New Shape**) button in the bottom part of the **Stencil** pane to create a blank drawing for drawing the shape.



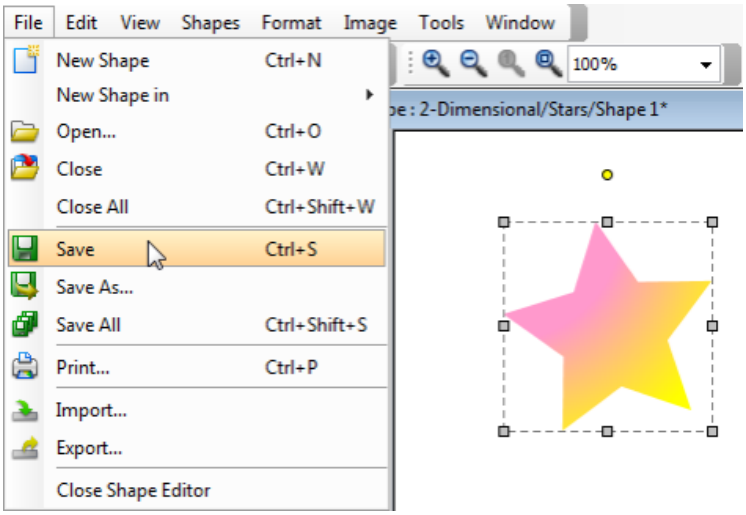
Add a shape

8. Create your preferred shape on the pane and you can also format its color in **Fill** tab of **Format** pane.



Draw a shape in drawing pane

9. To save the drawing by selecting **File > Save** from the main menu.



Save the drawing

Shapes created in Shape Editor can be used in Visual Paradigm. For details, please refer to the next page.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

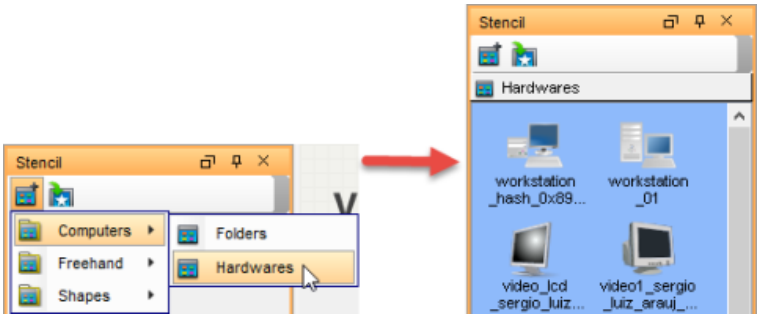
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating shape from stencil pane

The Stencil pane is where user-created stencil shapes are stored. User can create a stencil shape on diagram by first displaying a stencil, dragging and dropping a shape from **Stencil** pane to diagram. Below are the steps in detail.

1. Open the **Stencil** pane by selecting **View > Panes > Stencil** from the toolbar.

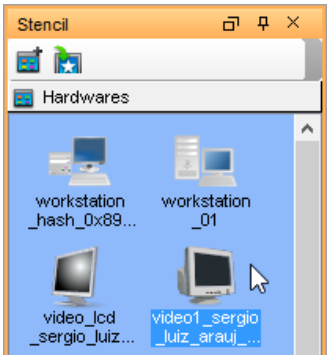
2. Click on the (Add Stencil) button in the top of **Stencil** pane. Select a category from the pop-up list of gallery. Select the stencil to add. The stencil is then added to the **Stencil** pane.



Add a stencil

> **NOTE:** You can add multiple stencil by repeating this step.

3. Press on a shape in the **Stencil** pane and drag it out of the **Stencil** pane and drop it on the diagram to create the shape.

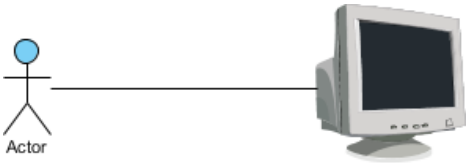


Dragging shape out of Stencil pane

4. You can also use generic connector to connect built-in notations shapes and stencil shapes. To do so, select **Generic Connector** in the diagram toolbar.

*Select **Generic Connector** from the diagram toolbar*

5. Press on the source shape, hold the mouse button, move the mouse cursor to the target shape and release the mouse button.



Connecting an Actor with a stencil shape

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Customizing user interface

You can hide away certain menu/toolbar button, or to configure the font of user interface through user interface customization.

Hiding user interface components

Hide away certain menu/toolbar button to ignore functions that you are not interested to use.

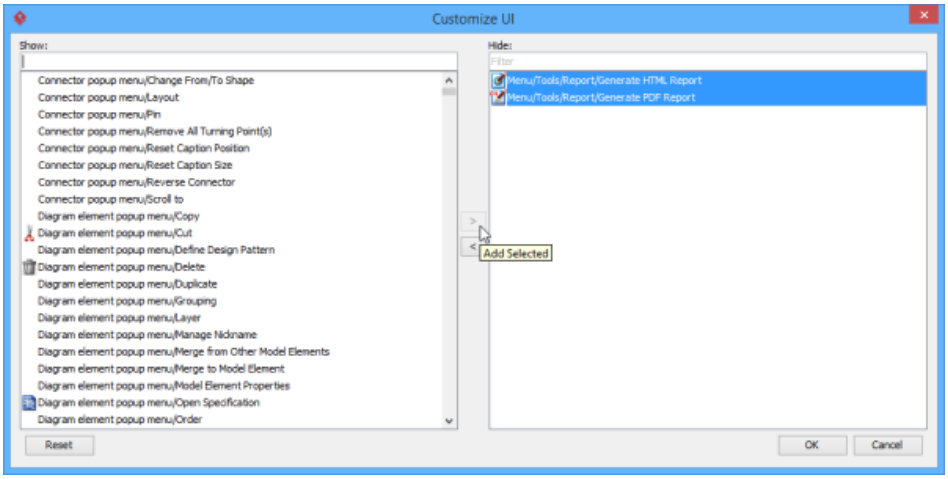
Adjusting user interface font

Adjust show the user interface with your favorite font.

Hiding user interface components

You may want to hide away some diagram types, menu items or toolbar items to avoid your team creating wrong types of model. This can be done by user interface (UI) customization.

1. Select **Help > Customize UI...** from the toolbar.
2. In the **Customize UI** window, select the menu(s)/toolbar button(s)/pop-up menu(s) to hide and click on the **>** button to hide them.



Select the menus to hide

3. Click **OK** button to confirm. By restarting the application, the selected user interface componenets will be hidden.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Adjusting user interface font

Visual Paradigm runs with a screen design that is friendly enough for most users, so that you can customize it to make it suit your preference. One of the possible customization is to adjust the font settings for text appears on user interface like the button caption for tools in toolbar, diagram editor tab's title, menus' captions, etc. The settings will be stored in workspace. Hence, you can keep the settings every time you run Visual Paradigm.

To adjust font settings:

1. Select **Windows > Application Options** from the toolbar.
2. Select **General** from the list on the left hand side.
3. Open the **Appearance** page.
4. Check **Change application font** in the **Font** section. Adjust the font type and size.
5. Click **OK**.
6. Restart the application to let the settings take effect.

Updated user interface

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Organizing works with model

A Model is a component that you can create in your project for organizing shapes and diagrams which acts like a folder. Modelers generally use models to differentiate stages or nature within project, such as an "as-is" model for storing diagrams and model elements about the current system, and a "to-be" model for recording blueprints of the system to be implemented. In this chapter, you will learn how to use model to organize your work.

Using model

Concepts about model will be discussed in this page.

Creating diagram under model

Shows to steps of creating model in Model Explorer.

Moving diagram to model

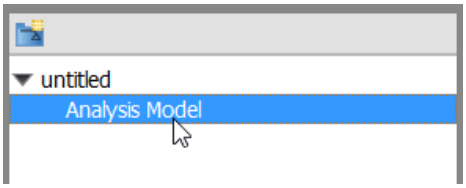
You can move a diagram to another move. This page shows you how to do.

Using Model

A model is a component that you can create in your project to organize shapes and diagrams which acts like a folder. Modelers generally use models to differentiate stages or nature within project such as an "as-is" model for storing diagrams and model elements about the current system and a "to-be" model for recording blueprints of the system to be implemented. The use of model improves not only the structuring of work but also the performance by reducing the number of root model elements to load. As a result, you can look up diagram or model element you need more easily.

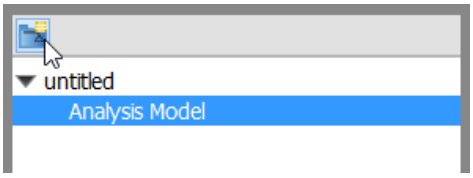
To create a model:

1. Open the **Project Browser** by selecting **View > Project Browser** from the toolbar.
2. Open the **Model Structure** view.
3. In the list of model and package, select the parent of the model to be created.



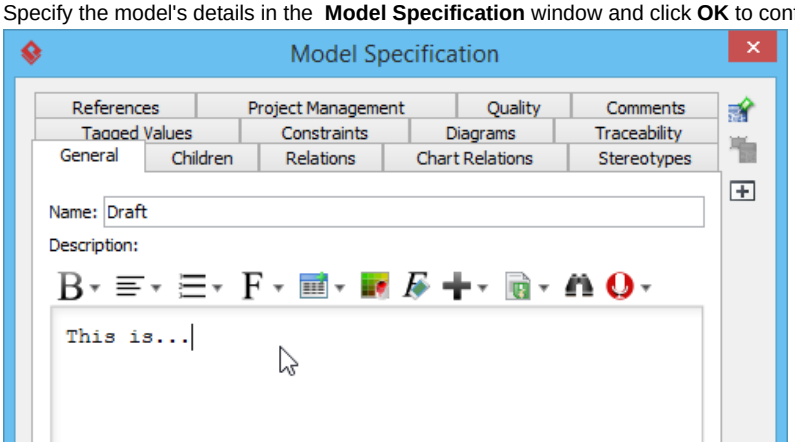
Selecting the parent of the model to be created

4. Click on the **New Model** button on top of the model list.



New model

5. Specify the model's details in the **Model Specification** window and click **OK** to confirm.



Filling in model specification

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating diagram under model

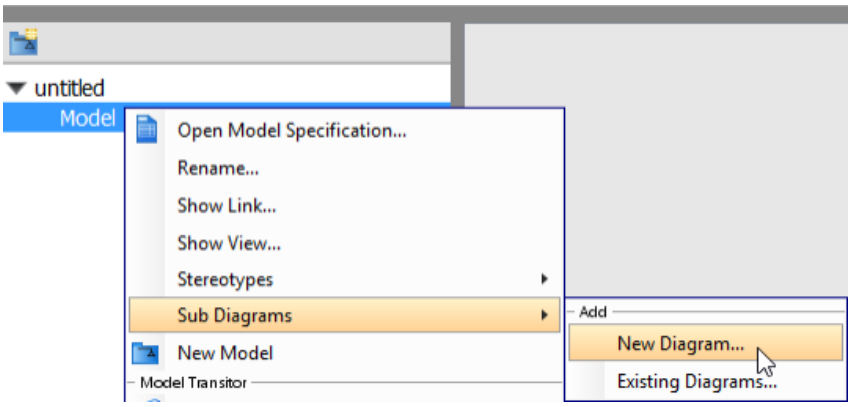
When a model has been created on the **Model Structure** view, you can start creating diagram(s) under the model. It is recommended to group diagrams using **Model** instead of laying them flat in the project. This can avoid accidentally loading diagrams and model elements that you never use and thereby can speed up project loading and saving.

To open the Model Structure View:

1. Select **View > Project Browser** from the toolbar to open the Project Browser.
2. In the Project Browser, select **Model Structure** view.

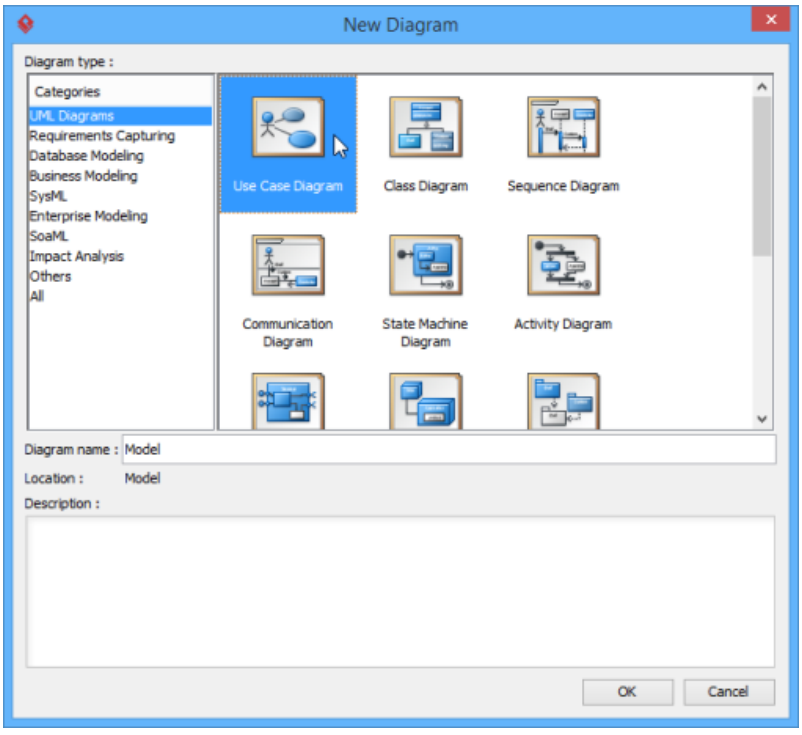
To create a diagram under model:

Right click on the target model and select **Sub Diagrams > New Diagram...** from the pop-up menu.



Create a new diagram under the model

When the **New Diagram** window pops out, select a specific category and the target diagram under the selected category. The new diagram name is *[Diagram Type]1* by default, you can rename it by entering in **Diagram name** text field. Click **OK** button to confirm and close the window.



***New Diagram** window*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Moving diagram to model

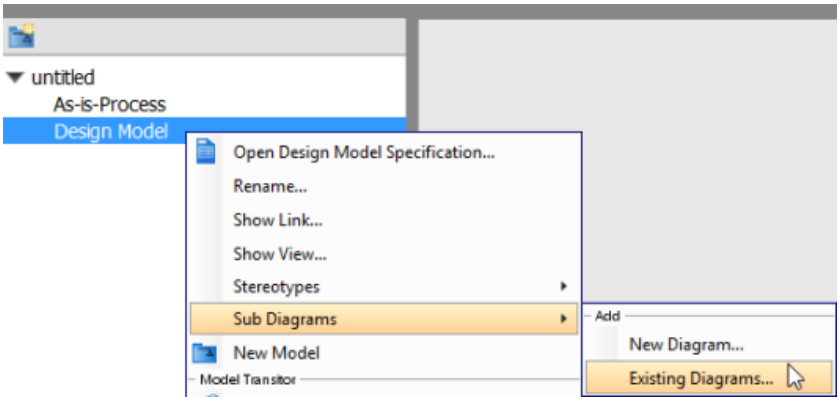
If you haven't organized project structure with model previously but want to do it at this stage, you can move a diagram from root into a model or transfer a diagram from one model to another.

To open the Model Structure View:

1. Select **View > Project Browser** from the toolbar to open the Project Browser.
2. In the Project Browser, select **Model Structure** view.

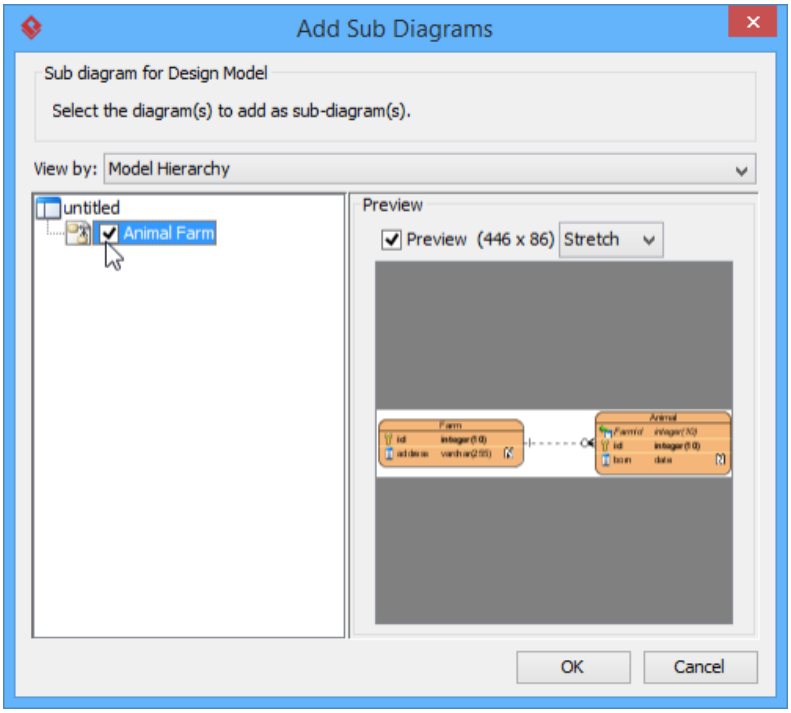
To move diagram from one model to another:

Right click on the target model and select **Sub Diagrams > Existing Diagrams...** from the pop-up menu.



Add existing diagram

When **Add Sub Diagrams** window pops out, check the diagram(s) you want to move and then click **OK** button to proceed.



*Check a diagram in **Add Sub Diagrams** window*

As a result, the selected diagram(s) will be moved to the target model.

NOTE: If you move a diagram which has the master view of model element(s), the model element(s) will be moved together with the diagram to the new model.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using stereotypes

A stereotype defines how a model element may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass. In this chapter, you will learn things about stereotypes and see how to apply stereotypes in your model.

Applying stereotype to model element

Tells you what stereotype is and how to apply to a model element.

Configure stereotypes

Shows you how to configure a stereotype like to define its color and add tagged values.

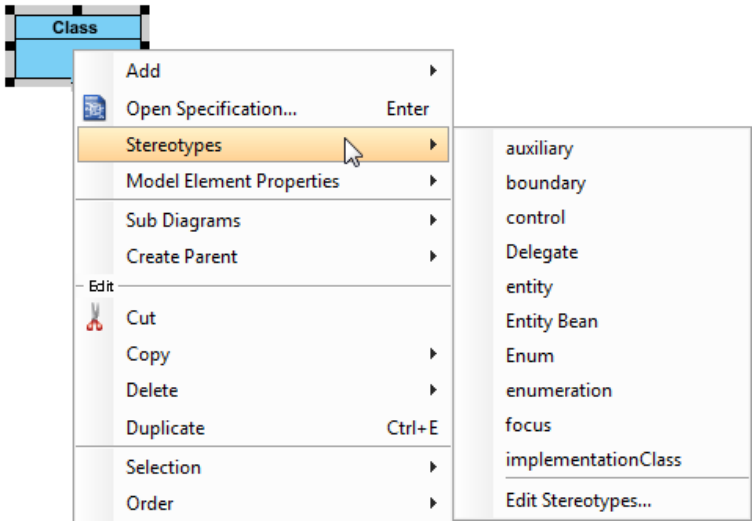
Shortcut of creating stereotyped model element

You can create a stereotyped element type easily through the diagram toolbar.

Applying stereotype to model element

A stereotype defines how a model element may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass. In Visual Paradigm, you can apply one or more stereotypes to model elements, and decide whether or not to visualize the stereotype or tagged values in views. To apply stereotype to model element:

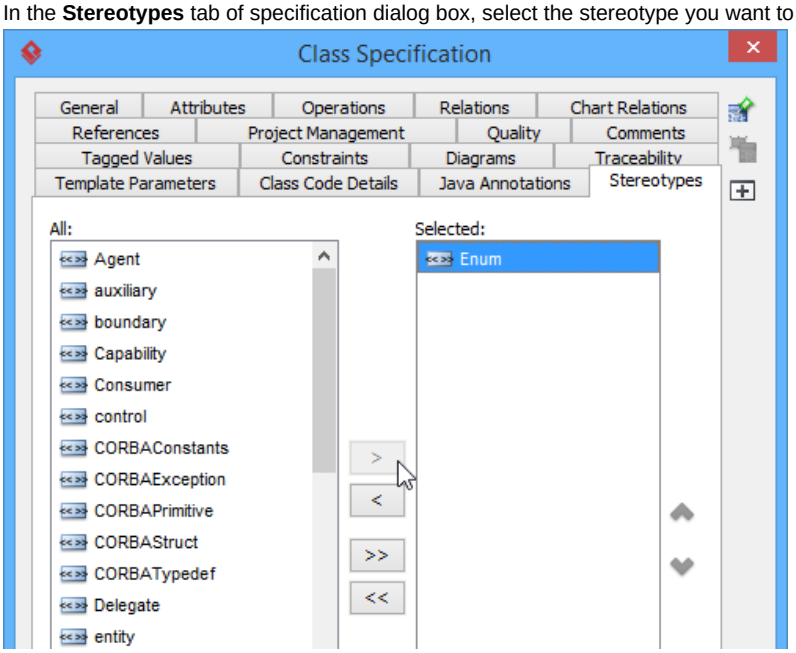
1. Right click on the model element, or the view of the model element that you want to apply stereotype to. Select **Stereotypes** from pop-up menu.



Select a stereotype

Depending on the type of model element you are selecting, there may be a list of suggested stereotypes listing in the menu popped up. It consists of both the recently used stereotypes and stereotypes that place at the top of stereotype list. If you see the stereotype you want to apply, select it. Otherwise, select **Stereotypes...** at the bottom of the menu to look for others.

2. In the **Stereotypes** tab of specification dialog box, select the stereotype you want to apply, then click **>** to assign it to the **Selected** list.

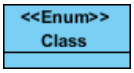


Stereotype Enum is selected

> **NOTE:** You can also double click on a stereotype to apply it.

> **NOTE:** While clicking on **>** applies the selected stereotype to model element, you can click **<** to remove a stereotype selected in Selected list.
>
> If you want to apply ALL available stereotypes to model element, click **>>**, and likewise, clicking on **<<** removes all the applied stereotypes.

3. Click **OK** to confirm. The stereotype will then be shown within a pair of guillemets above the name of the model element. If multiple stereotypes are applied, the names of the applied stereotypes are shown as a comma-separated list with a pair of guillemets.



Stereotype Enum is applied to a class

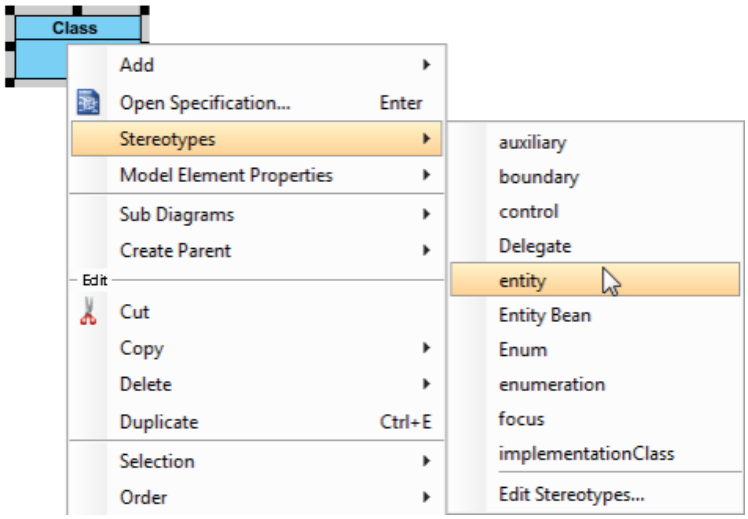
Robustness analysis icon

Robustness analysis helps to find out the relationships between actor, boundary, control and entity objects.

Robustness analysis icons

To draw a robustness analysis diagram with robust analysis symbols:

1. Create a class in diagram.
2. Depending on the type of robustness analysis symbol you want to create, apply either boundary, control and entity stereotype to the class.

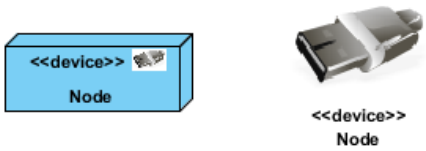


Apply entity stereotype to User class

NOTE: If you want to let a class display as traditional class shape instead of robustness analysis icons, right click on the class and de-select **Presentation Options > Display as Robustness Icon** from the popup menu.

Presenting a shape as stereotype icon

You can specify icon for a stereotype (Read the next chapter for details). When a stereotype is applied to a model element, you can let the stereotype icon show above the name of model element, which is the default presentation, or to make the model element show as the icon. To present a shape as stereotype icon, right click on the shape and select **Presentation Options > Stereotype Icon** from the popup menu.

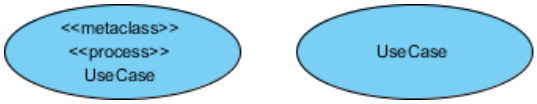


Different presentations of a model element with a stereotype that has icon defined

Showing or hiding stereotype

By default, applied stereotypes are shown within a shape. Yet, it is up to you whether to show or hide them. Furthermore, you can choose not to display the stereotypes, but to display only their tagged values.

To update the visibility of stereotypes, right click on the background of diagram where the shapes exist. Select/De-select **Presentation Options > Show Stereotypes** from the popup menu.



A use case with stereotype names shown and hidden

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Configure stereotypes

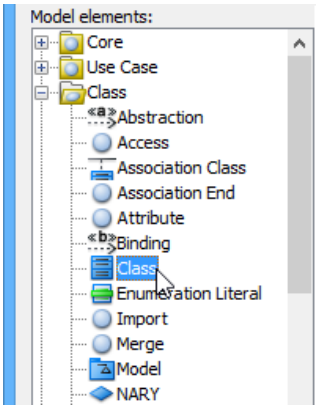
You can configure stereotypes, not just to create and name stereotypes for specific model element types, but also to format stereotypes like to set their colors, line formatting and font, and to define their tagged values. By configuring stereotypes, domain specific stereotype set can be built.

To configure stereotypes:

1. Select **Windows > Configuration > Configure Stereotypes...** from the toolbar.
2. Click on the drop down menu **Scope** at the top left corner of the **Configure Stereotypes** window, select whether to configure stereotypes in workspace or in the opening project.

> **NOTE:** Initially, stereotypes exist in workspace rather than in project. When you apply a stereotype to any model element, a copy of that stereotype will be made from workspace to project.
> By modifying stereotype in workspace, changes will not be applied to current project nor any project that has used the stereotype.
> To configure stereotype only in current project, you must select **Project** as scope.
> Alternatively, select **Workspace** but let the option **Apply changes to stereotypes in current project** on.

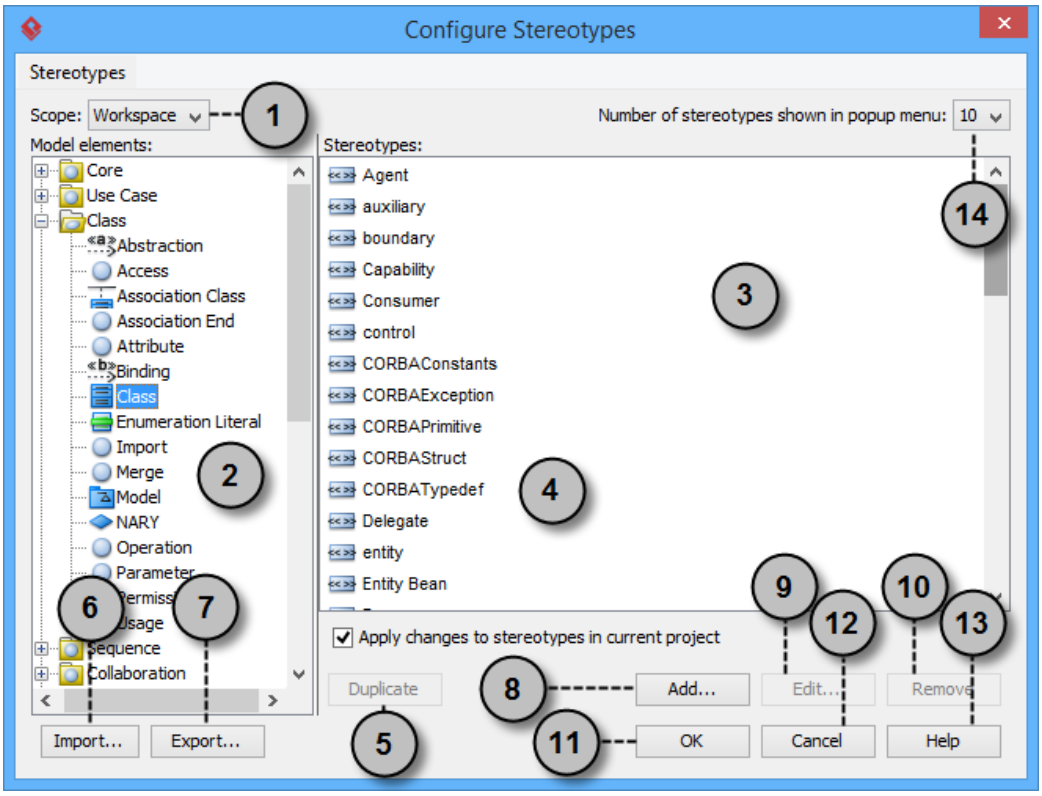
3. Select the type of model element that you want to add stereotype or edit its existing stereotypes.



Select class to edit its stereotypes

4. You may now perform any of the following action:
   - If you want to edit an existing stereotype, select the stereotype and click **Edit...**.
   - If you want to add a stereotype, click **Add...**.
   - If you want to remove a stereotype, select the stereotype and click **Remove**.
5. If you are adding or editing a stereotype, update its specification and click **OK** to confirm editing. For details about editing a stereotype, read the coming section.
6. Click **OK** to confirm.

An overview of Configure Stereotypes dialog box



*An overview of **Configure Stereotypes** dialog box*

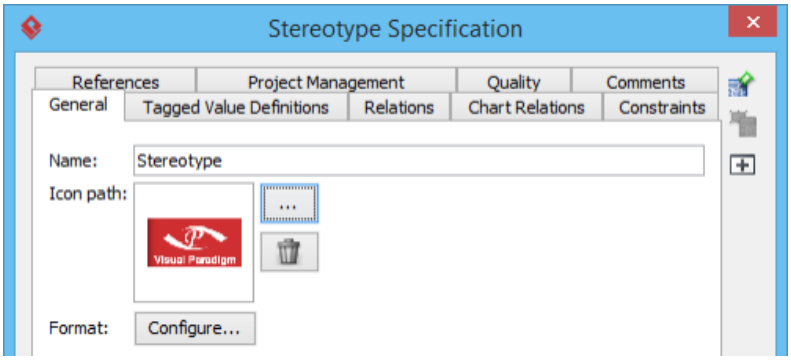
No.	Name	Description
1	Scope	Initially, stereotypes exist in workspace rather than in project. When you apply a stereotype to any model element, a copy of that stereotype will be made from workspace to project. By modifying stereotype in workspace, changes will not be applied to current project nor any project that has used the stereotype because the stereotype copied to project is being followed. If you want to configure stereotype only in current project, you must select **Project** as scope, or select **Workspace** but let the option **Apply changes to stereotypes in current project** on to make changes apply on both workspace and project.
2	Model element list	A list of categorized model element types. You can select a type to configure its stereotypes.
3	Stereotypes	A list of stereotypes of the selected model element type.
4	Apply changes to stereotypes in current project	Available only when scope is Workspace, this option cause the stereotype configuration applies to both stereotypes in workspace and project, when pressing **OK** .
5	Duplicate	Click to duplicate the stereotype selected in Stereotype pane.
6	Import	Click this to import stereotype configuration (an XML) produced by others. Once clicked, the **Import Stereotypes** dialog box will popup. You need to choose the XML file to import. At the bottom of the dialog box, there is an option **Add and update only (do not delete stereotypes)**. When checked, Visual Paradigm will only add and update stereotypes from XML. When unchecked, Visual Paradigm will add and update stereotypes, and additionally delete stereotypes that are not defined within the XML.
7	Export	Click to export stereotype configuration to an XML file.
8	Add	Click this to add a stereotype for the selected type of model element.
9	Edit	Click to edit the selected stereotype.
10	Remove	Click to delete the selected stereotype.
11	OK	Click to apply the configuration and close the dialog.
12	Cancel	Click to discard the changes (if any) and close the dialog box.
13	Help	Click to open the Help contents.

14	Number of stereotypes shown in popup menu	You can assign stereotype to a shape easily by right clicking on that shape and selecting **Stereotypes > %STEREOTYPE%** from the popup menu. This option is to control the number of stereotype to be listed for selection. The larger the number, the longer the popup menu, the easier you can find the desired stereotype.

*Description of **Configure Stereotypes** dialog box*

Editing stereotype

By adding or editing a stereotype, you can specify its icon and adjust its fill, line and font style in the **General** page within the **Stereotype Specification**.



Editing stereotype

By applying a stereotype that has icon defined to a model element, the icon above the name of model element, near the stereotype. You can optionally make the model element shown as the icon. For details, read the previous chapter. To specify icon, click on the **...** button near the preview of Icon. Then, select the image file of icon.

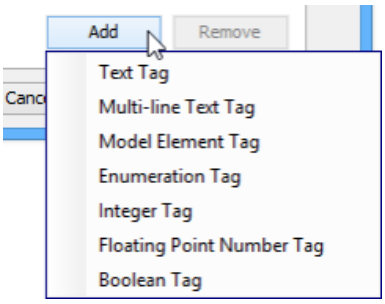
You can also configure the formatting properties of the stereotype by clicking **Configure...**. In the popup window, you can set the foreground, background and line styles by first checking **Use** and then start the editing. The settings will be applied automatically to model elements that have the stereotype applied.

Defining tagged values for stereotypes

A stereotype may have properties, which may be referred to as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred to as tagged values.

You can define tagged values for stereotypes. By doing so, when you apply the stereotype with tagged values defined to a model element, you can fill in the values for the model element.

1. Select **Windows > Configuration > Configure Stereotypes...** from the toolbar.
2. In the **Configure Stereotypes** dialog box, select the stereotype that you want to define tagged value and click **Edit**. If you want to add a new stereotype, select the base model type and click **Add...**
3. In the **Stereotype Specification** dialog box, open the **Tagged Value Definitions** tab.
4. Click **Add**. Select the type of tagged value to define. The type of tagged value limits the type of content user can enter for a tag.



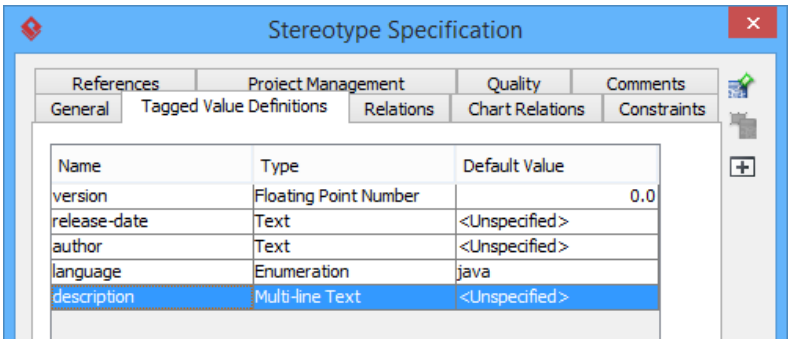
Adding a tag

Tag type	Description
Text	The most common and general type of tagged value that consists of words.
Multi-line Text	The value of tag is a text in multiple lines.
Model element	The value of tag is a model element in project.
Enumeration	The value of tag can be chosen from a list of possible values. For example, to select "red" out of values red, green and blue.
Integer	The value of tag must be a real number.
Floating point number	The value of tag must be a number that consists of one or more digits.

Boolean	The value of tag muse be either true or false.

Type of tags

5. Double click the name cell and enter the name of tag. Repeat step 4 and 5 to add all tagged values for this stereotype.



Tags defined

6. You can assign a default value to a tag by editing the **Default Value** cell. Usually, you give a tag a default value when the value is true in most cases. For example, a tag "in-door-temperature" can have "25" as default value.

Related Resources

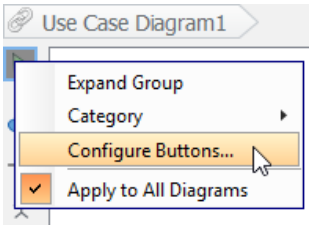
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Shortcut of creating stereotyped model element

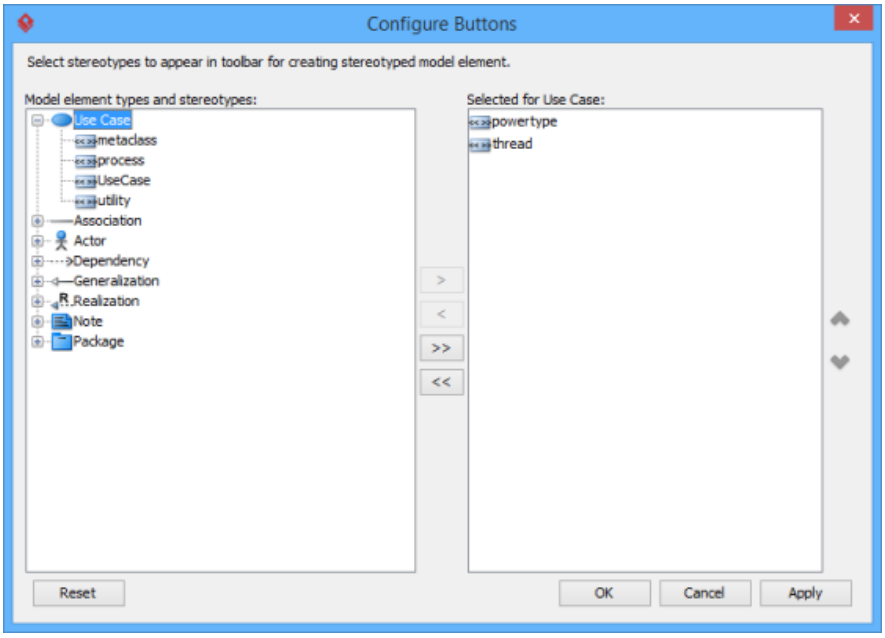
One of the ways of creating a shape is by selecting the element type in diagram toolbar, and clicking on diagram to create a shape of that type. If then you want to apply a stereotype to that model element, you will open the specification dialog box and make a stereotype selection. These steps can be simplified by adding a shortcut in diagram toolbar, for creating a type of model element with specific stereotype pre-set. To do so:

1. In any diagram, right click on the **Selector** from the diagram toolbar and then select **Configure Buttons**... from the popup menu.



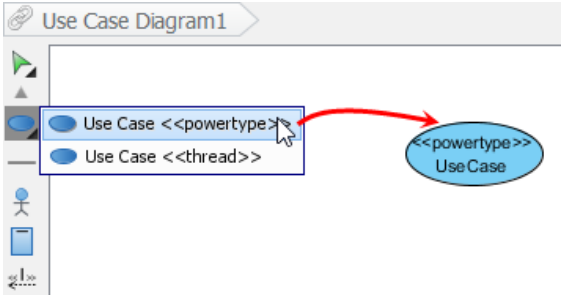
To configure buttons

2. In the **Configure Buttons** dialog box, expand the node(s) of model element type(s) that you want to add shortcut for. Select the stereotype that you need to apply to that type of model element in future. Click **>** or double click on it to assign. Click **OK** to confirm.



Configure buttons in diagram toolbar

3. After all, you can find the shortcuts in diagram toolbar, under the selected type(s) of model elements. By selecting a stereotyped model element type and click on diagram, you can create a shape with stereotype applied.



Create a stereotyped shape

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Collaborative Modeling in Visual Paradigm

Team collaboration is the practice of working in a team instead of by one person. Team members can work individually on their own specific parts of a project, and eventually combine works together to form a complete project. This chapter provides you with clear information on Visual Paradigm's team collaboration support.

Introduction to collaborative modeling

Describes the key concepts about team collaboration like administration, commit, update, revert, etc.

The teamwork client

Teamwork Client, where you can manage, checkout and open projects, involves all teamwork activities that you can perform with.

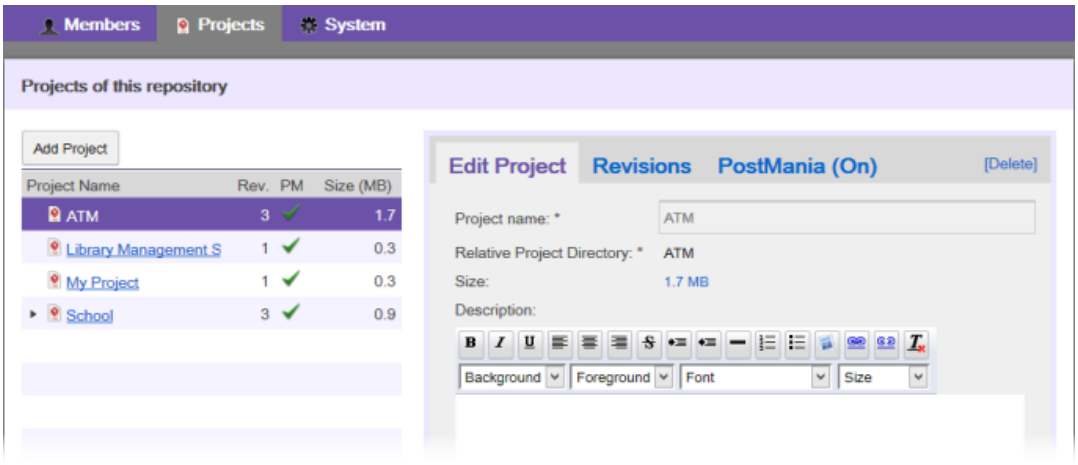
Introduction to team collaboration

Team collaboration is the practice of working in a team instead of by one person. Team members can work individually on their own specific parts of a project and eventually combine works together to form a complete project. As a result of collaborations between members by employing the unique skills of each, works can be done more effectively and in higher quality.

Visual Paradigm's team collaboration support provides access to a central repository for managing, sharing and versioning projects. You can allow team members to get project from repository, start working on their own parts and allow them to commit (i.e. upload) their work to server as well as update others' works. Visual Paradigm VPository and VP Teamwork Server are the supported standards of versioning systems. This chapter outlines some of the key concepts in team collaboration in Visual Paradigm.

Member and project management

Member and project management are processes of setting up members and projects and deciding the access rights of members against projects.



Edit project

Checkout and open project

Checkout project is the process to get project from repository to start working with. Team members can login to the server, then checkout the project(s) to work with, provided that they have the permission to do so, as granted by administrator. After that, open to start working on it.

Commit

Commit is the process to upload changes done in working copy to server. As team members make changes in a project, they can share their works by committing those changes to the server. Visual Paradigm will try to merge changes from working copy to server copy. When merging, there may be conflict when any changes a team member made to cause an unresolvable contradiction with changes made by others. Team member is required to decide whether to keep his/her change (i.e. overwrite) or to take the co-worker's change (i.e. revert). All conflicts must be resolved in order to proceed with committing.

Update

Update is the process of refreshing the working copy by merging changes that others had made and committed to server before. Similar to commit, update is a process of merging differences instead of overwriting. If your changes overlap the changes others had made, you will be asked to resolve conflict. All conflicts must be resolved in order to proceed with updating.

Conflict

Conflict is a situation that happens when committing or updating. It occurs during the merging between working and server copy of project, when a contradiction is detected between them. For example, a team member has renamed a shape from *A* to *B* and has committed the change. Then, another team member has renamed the same shape from *A* to *C* and attempt to commit. Due to difference in the name of shape, a conflict is occurred. Whenever a conflict occurres, you have to resolve it or else to abort before commit/update operation.

Branching

Branching is the process of creating a branch from trunk (i.e. main development flow), for isolating changes that are not supposed to be available on trunk, either at the moment or permanently. By working in a branch, team members can carry out half-broken or risky changes without worrying the risk of damaging the work in trunk. Once the works done in branch is examined and confirmed alright, team member can make changes available in trunk through merging. Merging can also be done from trunk to branch to ensure that the branch is always up-to-date.

Tagging

Tagging is the process of producing a snapshot (i.e. tag) of a project in time. People often create tags for archiving releases of works. Therefore, tags are often named as release-1.0 where 1.0 is the number of version. Since a tag is a snapshot, team members can never commit under a tag.

Revision history

Every time a team member performs a commit with success, a new revision is created as a snapshot of project. More and more revisions will be created through repeated committing. A list of revisions that shows the changes of project is called the revision history. In Visual Paradigm, you can review the works done in specific revision(s) by exporting them in project files. You can identify the differences between revisions by comparing them.

Revert changes

Revert is the process of undoing changes. In Visual Paradigm's team collaboration support, there are two kinds of revert actions that you can perform. The first one is to revert locally modified changes to make the working copy back to the original state. Another revert action is for undoing changes made in revisions. Team members can undo changes made in revisions by reverting them.

Related Resources

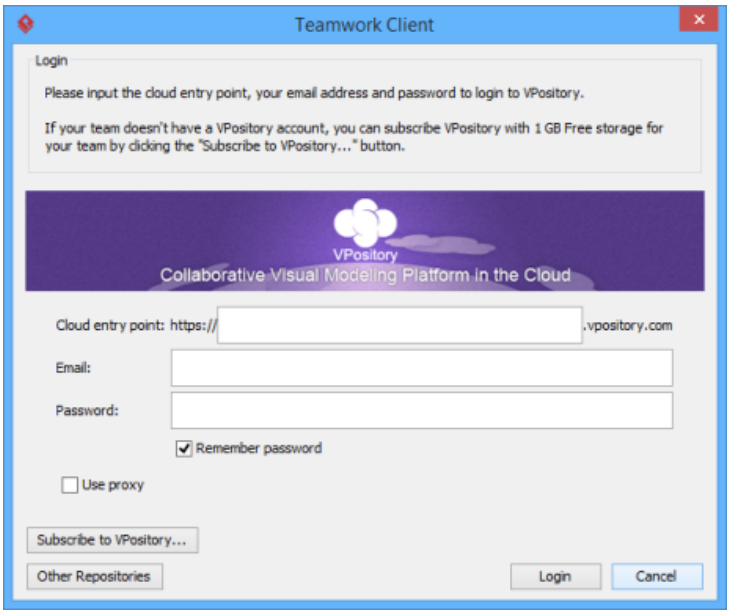
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

The Teamwork Client

Teamwork Client is where you can manage your projects. It enables you to select/de-select projects to work on with, and to provide you with access to all team operations like commit, update, branching, tagging, etc. To open **Teamwork Client**:

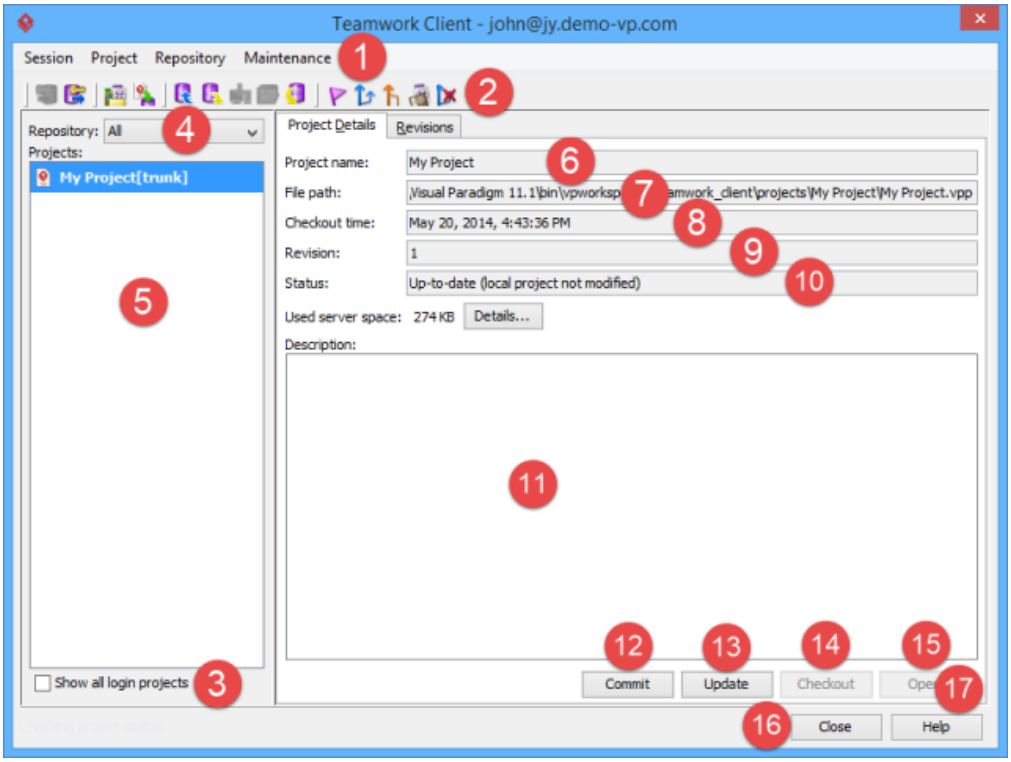
1. Select **Team > Login** from the toolbar of Visual Paradigm. If you have no prior connection to any server, you are then prompted to enter the login details of VPository. If your team have already subscribed to VPository, enter your login details. If you want to subscribe to VPository, click here for details . If you want to use Teamwork Server, click **Other Repositories** and enter the login details.



Login window

2. Click **Login**.
3. Since this is the first time you login to the server, you are prompted the **Open Project** window with available projects listed. Click **Cancel**.
4. Now, you are connected to the server. When you select **Team > Login** again, you can then open **Teamwork Client**.

Overview of Teamwork Client window



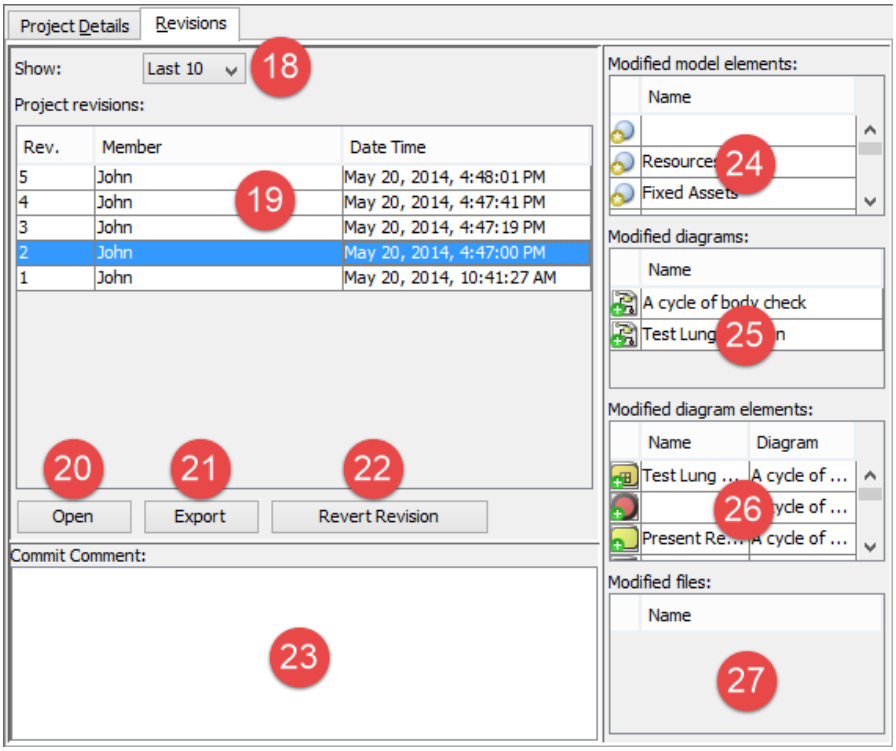
*The **Teamwork Client** window*

No.	Name	Description
1	Main menu	**Session**: It is a period of active connection with server. **Login**: Login the server. After you choose it, you will be able to execute all actions.

		Logout: Logout the server. After you choose it, you will not be able to execute any actions. **Project**: Provides an access to main functions, such as commit and update. **Manage Project**: Select a project that you get involved in. **Import Project to Repository**: Import a new project to the server. **Commit**: Commit your current modified project to the server. **Update**: Update the latest copy of project from the server to your computer. **Checkout**: Click it to checkout the project selected in Projects list. It will be disabled when the selected project has already been checked out. **Open**: Click it to open the checkout project on your computer. **Tag**: Create a new tag for your current project. It allows you to produce a static release version of project. **Branch**: Create a new branch for your current project. It becomes a duplication of project to perform isolated changes. **Merge**: Combine the selected branch(es) with the trunk (main project). When some changes made in branch, it will be made in trunk as well. **Switch**: Switch from a branch/ tag to another branch/ tag or from the trunk (main project) to a branch/ tag and vice versa. **Delete Branch**: Select a branch to delete, for preventing accidental modifications in branch. **Reset Password**: Reset your account's password. **Revert Local**: Click it to undo un-committed changes made on the local project copy. **Repository**: **Synchronize Design Pattern to Server**: Synchronize the template files stored in workspace with those stored in repository. When a conflict occurs, you will be asked which design template files to keep. **Maintenance:** All the functions under the Maintenance menu are prepared for diagnosis purposes. You should not run them unless you are requested by Visual Paradigm support team. And when you are requested to execute any maintenance function, you will be briefed.
2	Toolbar	**Login**: Log into the server. After you choose it, you will be able to execute all actions. **Logout**: Log out the server. After you choose it, you will not be able to execute any actions. **Manage Project**: Select a project that you get involved in. **Import Project to Repository**: Import a new project to the server on the list. **Update**: Update the latest copy of project from the server to your computer. **Commit**: Commit your current modified project to the server. **Checkout**: Click it to checkout the project selected in Projects list. It will be disabled when the selected project has already been checked out. **Open**: Click it to open the checkout project on your computer. **Revert Local**: Click it to undo un-committed changes made on the local project copy. **Check for Update**: Click it to check whether the project is up-to-date or not. **Tag**: Create a new tag for your current project. It allows you to produce a static release version of project. **Branch**: Create a new branch for your current project. It becomes a duplication of project to perform isolated changes. **Merge**: Combine the selected branch(es) with the trunk (main project). When some changes are made in branch, it will be made in trunk as well. **Switch**: Switch from a branch/ tag to another branch/ tag or from the trunk (main project) to a branch/ tag and vice versa. **Delete Branch**: Select a branch to delete, for preventing accidental modifications in branch.
3	Show all login projects	By checking it, projects that have been checked out before will all be displayed. By unchecking it, only projects checked out by the current member will be displayed.
4	Repository	It refers to the list of available project(s). Select **All** from the drop-down menu means all projects managed by all eligible members who have logged into server in this workspace will be listed. On the other hand, the project(s) managed by a specific member can be selected from the drop-down menu. If you uncheck **Show all login projects** and do not select the current member in **Repository**, no project will be listed.
5	Projects	It lists the project(s) you selected to manage.
6	Project name	The name of selected project.
7	File path	The path of the selected project file. It is shown only when project is checked out from the server.
8	Checkout time	It displays the date and time of your first checkout for the project.

9	Revision	It displays the revision of your local project copy. Note that the revision here does not always mean the latest revision on the server.
10	Status	It displays the status of selected project, such as "Not Checked Out" will be shown when the project has not been checked out yet.
11	Comment	It shows the textual description of selected project written by administrator when creating project.
12	Update	Update the latest project from the server to your computer.
13	Commit	Commit your current modified project to the server.
14	Checkout	Click it to checkout the selected project.
15	Open Project	Click it to open the checkout project on your computer. If the project has not checked out yet, it will perform a checkout prior to opening project.
16	Close	Click to close the **Teamwork Client**.
17	Help	Click it to get assistance from help system.

*The description of the **Teamwork Client** window*



*The **Revisions** tab of the Teamwork Client*

No.	Name	Description
18	Show drop-down menu	Select the number of latest project revision to view from the drop-down menu.
19	Project revisions	It lists all the latest project revisions. The number of revisions is in accordance with the show drop-down menu.
20	Open	Click it to open the selected revision of project.
21	Export	**Export selected revisions**: Export the selected revision(s) to a folder. **Export all revisions from repository**: Export all the projects in repository to a folder.
22	Revert Selected	Undo changes committed by the selected revisions.
23	Commit Comment	A textual description of commit given by you or your teammates before committing.
24	Modified model elements	It displays the modified model elements of the selected revision.
25	Modified diagrams	It displays the modified diagrams of the selected revision.
26	Modified diagram elements	It displays the modified diagram elements of the selected revision.
27	Modified files	It displays the modified Teamwork Files of the selected revision.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Basic Team Collaboration Features in Visual Paradigm

Check out the team collaboration features supported by Visual Paradigm.

Checkout project

Checkout project is a process that you download a managed project from repository to your computer and start working with.

Commit

Commit refers to the process of uploading local modifications to the server.

Support commit part of project to Teamwork

Other than committing the whole project, you may commit specific model elements/diagrams to Teamwork.

Update

Update is the process of refreshing your current copy by merging changes that others have made and committed previously to server.

Revert local modification

Since VP teamwork collaboration allows the feature of revert, you can undo all modifications you made in the local project copy when the changes haven't been committed yet.

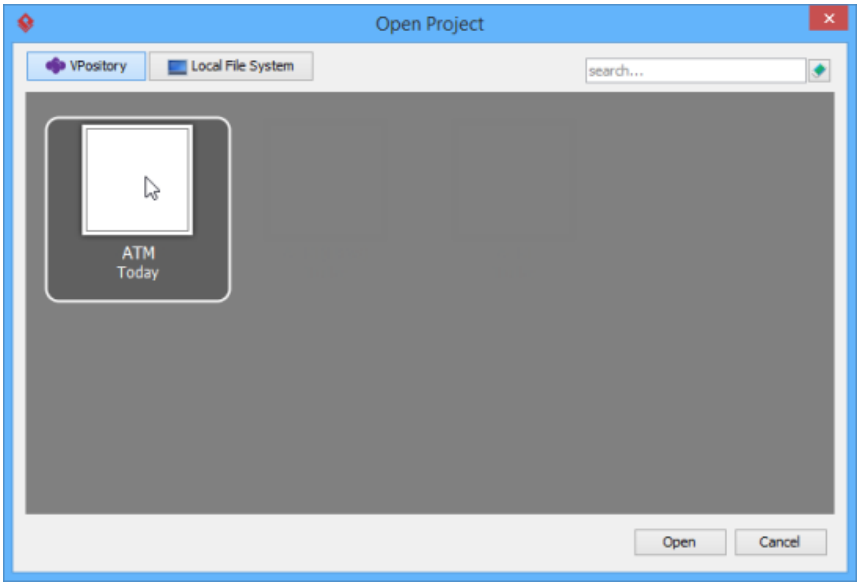
Import project

Import project refers to the process of adding project file in server so that team members can check that out and start working on it.

Checkout and open project

Checkout project is a process done by team members, for getting a project from repository to start working with. Team members can login into the server and then checkout the project(s) to work with, provided that they have the permission to do so, as granted by administrator. After that, open to start working on it.

1. Select **Team > Login** from the toolbar
2. Enter the cloud entry point of VPository, login email and password, and then click **Login**. If you are using Teamwork Server, click **Other Repositories**, select **VP Teamwork Server** as Server, enter host, email, password and then click **Login**.
3. Since this is the first time you login to the server, you are prompted the **Open Project** window with available projects listed. You have either or both read and update right to these projects. You can start working on a project by selecting it and click **Open** at bottom right, or simply by double clicking on the project thumbnail.



Checkout project

Checkout multiple projects

Instead of checking out a single project, you can checkout multiple project files at the same time, for those listed in the **Teamwork Client** window as managed projects.

1. Select **Team > Login** from the toolbar.
2. From the list on the left hand side, select the projects to checkout. You can perform a multiple selection by first pressing the **Ctrl** key.
3. Right click on the selection and select **Checkout** from the popup menu.

Related Resources

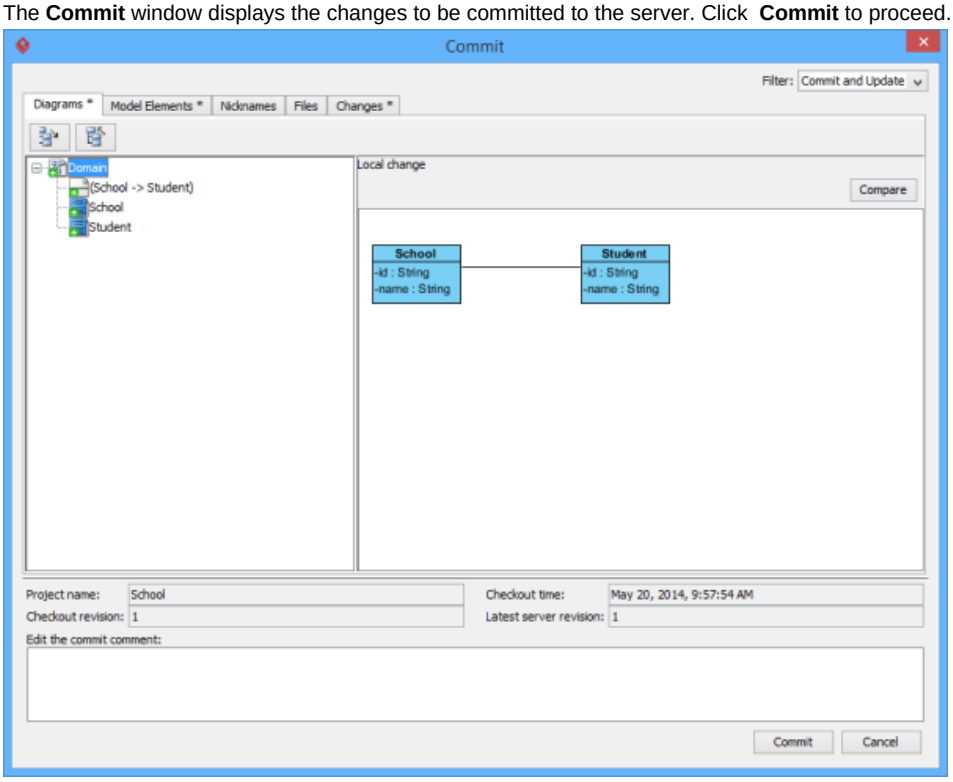
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Commit

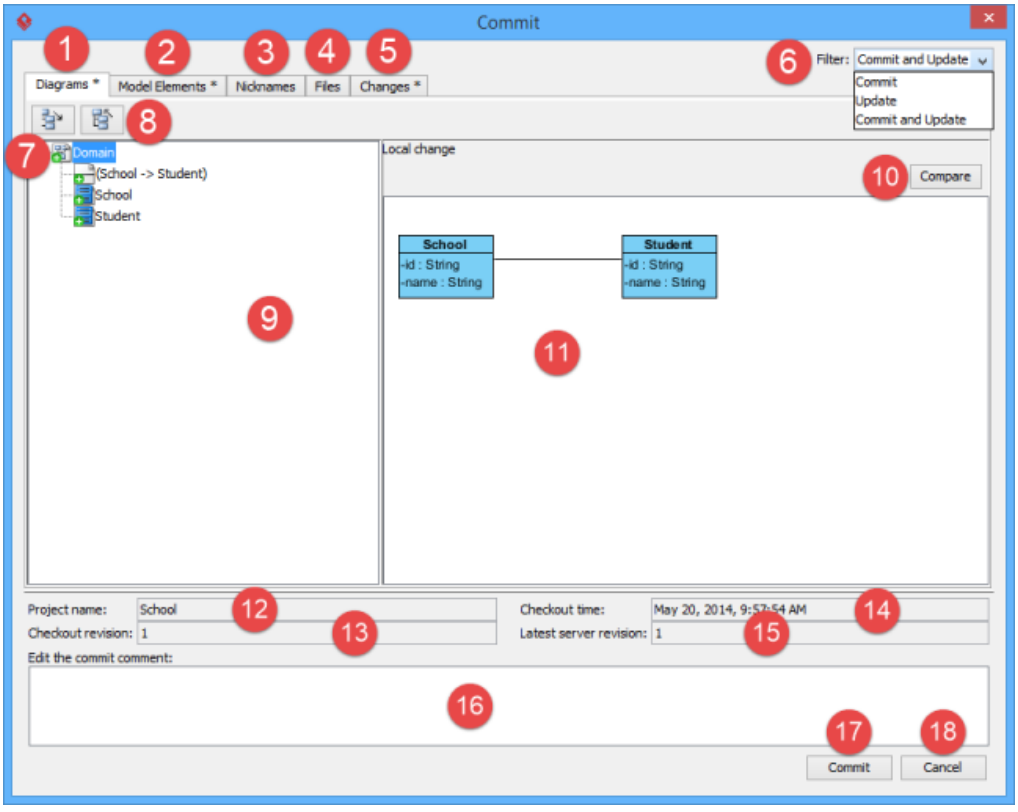
Commit refers to the process of uploading local modifications to the server. As team members make changes in a project, they can share their works by committing those changes to the server. By committing, changes are merged from working copy to server copy. During merging, a conflict may be caused when there is a contradiction between team members. Decision have to be made whether to keep the current modified copy (i.e. overwrite) or to accept others' copy (i.e. revert). All conflicts have to be solved before proceeding to commit. To commit changes:

1. Select **Team > Commit** from the toolbar.
2. If the change you made contradicts the change made by another team member, this will result in a conflict. You must resolve all the conflicts in order to continue. For details, read the Resolving conflicts section below. Clear the conflicts, if any, and continue.
3. The **Commit** window displays the changes to be committed to the server. Click **Commit** to proceed.



*The **Commit** window*

Overview of Commit window



*The **Commit** window*

No.	Name	Description
1	Diagrams tab	The diagram level changes to be performed when you execute commit.
2	Model Elements tab	The model element level changes to be performed when you execute commit.
3	Nicknames tab	The changes of nickname to be performed when you execute commit.
4	Files tab	The file changes to be performed when you execute commit.
5	Changes tab	All the changes to be performed when you execute commit.
6	Filter	When you commit, local changes will be merged to the server copy and meanwhile, changes in server copy will be merged to the local copy, too. The filter allows you to filter the tree on the left hand side to list the commit changes, which are changes to perform on server copy, and/or update changes, which are changes to perform on local copy.
7	Expand All	Expand the tree nodes in the tree below.
8	Collapse All	Collapse the tree nodes in the tree below.
9	Tree	List out the changes to be performed when you execute commit.
10	Compare	Click this button to compare local and server copy side by side.
11	Preview	The preview of the element as selected in the tree on the left hand side.
12	Project name	Name of current project.
13	Checkout revision	The number of current checkout revision.
14	Checkout time	The time for latest checkout.
15	Latest server revision	The number of latest revision in the server.
16	Commit comment	You can give a comment for your current commit by typing here.
17	Commit	Proceed committed
18	Cancel	Cancel committing and close the window.

*The description of **Commit** window*

Committing multiple projects

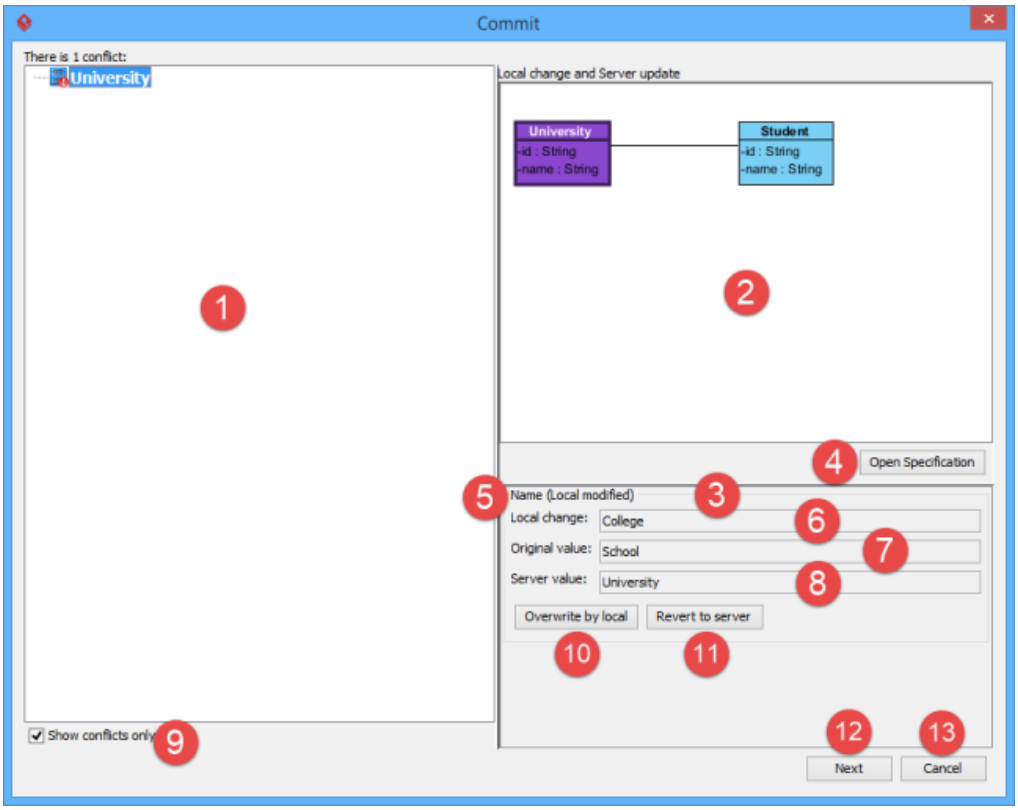
Instead of commtting a single project, you can commit multiple project files at the same time, for those listed in the **Teamwork Client** window as managed projects.

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. From the list on the left hand side, select the projects to commit.
3. Right click on the selection and select **Commit...** from the popup menu.

Resolving conflict

If the change you made contradicts with the change made by another team member, this will result in a conflict. For example, your colleague has renamed a class from School to University and performed a commit, and then you rename the same class to College and perform a commit. This produces a conflict.

When a conflict occur, you must resolve it in order to continue committing. You have to resolve conflict either by overwriting or reverting the change. Overwrite means to adopt the server copy's change, while Revert means to adopt the local's change.



Conflicts when committing

No.	Name	Description
1	List of change (action)	List of changes to be performed. Initially only conflicted changes are listed. You can uncheck **Show conflicts only** to list all changes.
2	Preview	The preview of the element as selected in the tree on the left hand side.
3	Conflicted properties	The properties that cause conflicts are listed in this panel.
4	Open Specification	Click on this button to open the specification of element selected in the tree on the left hand side.
5	Property name	The names of conflicted properties.
6	Local change	The value of property in local project copy.
7	Original value	The value of property before changed. In other words, it is the value in checkout copy.
8	Server value	The value of property in server project copy.
9	Show conflicts only	Check it to list only conflicted changes in the tree on the left hand side. Uncheck it to list all changes.
10	Overwrite by local	Click on this button to adopt the source copy.
11	Revert to server	Click on this button to adopt the target copy.
12	Preview	Click this button to continue committing.
13	Cancel	Cancel committing and close the window.

*The description of **Commit** window when have conflicts*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Commit a part of a project

In addition to committing all the changes you have made within the session, you can commit change(s) made on specific model element(s), shape(s) or diagram(s). This way of committing is called a *partial commit*. To work with partial commit, simply right click on the project data you want to commit and **Team > Commit** from the popup menu. You may select multiple items (e.g. two shapes) and commit them at once.

Related Resources

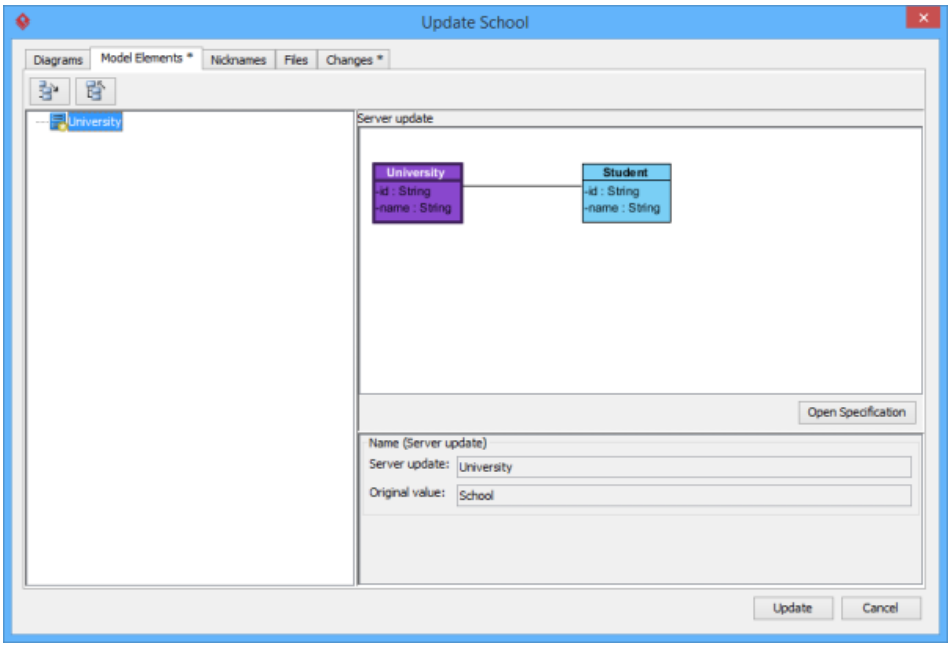
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Update

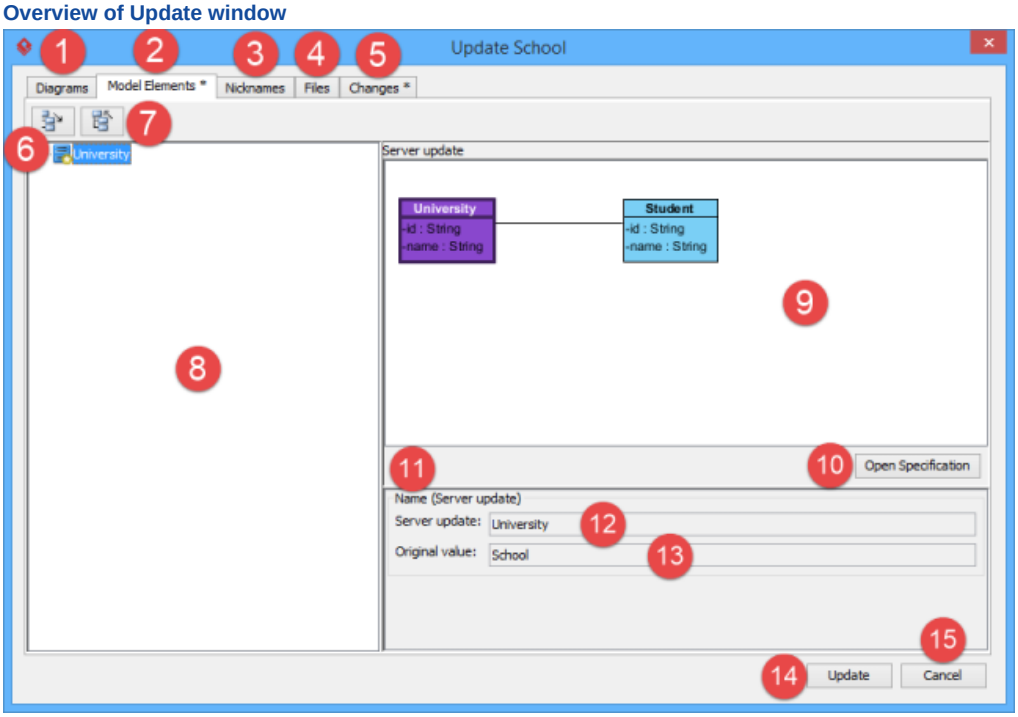
Update is the process of refreshing your current copy by merging changes that others have made and committed previously to server. Similar to commit, update is a process of merging differences instead of overwriting. When any of your changes contradicts the changes others have made, you will be asked to resolve conflict. All conflicts have to be resolved before proceeding with updating. To update:

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. If the change you made contradicts the change made by another team member, this will result in a conflict. You must resolve all the conflicts in order to continue. For details, read the Resolving conflicts section below. Clear the conflicts, if any, and continue.
3. The **Update** window displays the changes to be made upon updating. Click **Update** to proceed.



*The **Update** window*

Overview of Update window



*The **Update** window*

No.	Name	Description
1	Diagrams tab	The diagram level changes to be performed when you execute update.
2	Model Elements tab	The model element level changes to be performed when you execute update.
3	Nicknames tab	The changes of nickname to be performed when you execute update.
4	Files tab	The file changes to be performed when you execute update.

5	Changes tab	All the changes to be performed when you execute update.
6	Expand All	Expand the tree nodes in the tree below.
7	Collapse All	Collapse the tree nodes in the tree below.
8	Tree	List out the changes to be performed when you execute update.
9	Preview	The preview of the element as selected in the tree on the left hand side.
10	Open Specification	Click on this button to open the specification of element selected in the tree on the left hand side.
11	Property name	The name of conflicted property.
12	Server update	The value of property in server project copy.
13	Original value	The current value of property (before updating).
14	Update	Proceed updating
15	Cancel	Cancel updating and close the window.

*The description of **Update** window*

Updating multiple projects

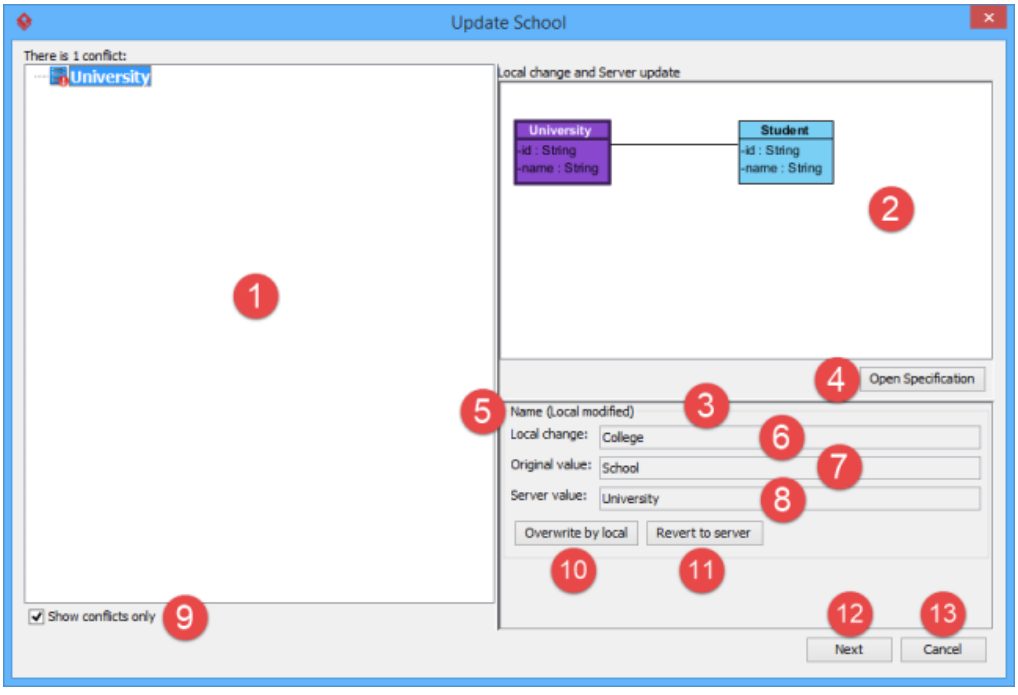
Instead of updating a single project, you can update multiple project files at the same time, for those listed in the **Teamwork Client** window as managed projects.

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. From the project list on the left hand side, select the projects to update.
3. Right click on the selection and select **Update...** from the popup menu.

Resolving conflict

If the change you made contradicts the change made by another team member, this will result in a conflict. For example, your colleague has renamed a class from School to University and performed a commit and then you rename the same class to College and perform an update. This produces a conflict.

When a conflict occurs, you must resolve it in order to continue updating . You have to resolve conflict by overwriting or reverting the conflicted change. Overwrite means to adopt the server copy's change, while Revert means to adopt the local's change.



Conflicts when updating

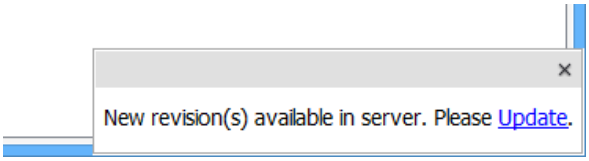
No.	Name	Description
1	List of change (action)	List of changes to be performed. Initially only conflicted changes are listed. You can uncheck **Show conflicts only** to list all changes.
2	Preview	The preview of the element as selected in the tree on the left hand side.
3	Conflicted properties	The properties that cause conflicts are listed in this panel.

4	Open Specification	Click on this button to open the specification of element selected in the tree on the left hand side.
5	Property name	The name of conflicted property.
6	Local change	The value of property in local project copy.
7	Original value	The value of property before changed. In other words, it is the value in checkout copy.
8	Server value	The value of property in server project copy.
9	Show conflicts only	Check it to list only conflicted changes in the tree on the left hand side. Uncheck it to list all changes.
10	Overwrite by local	Click on this button to adopt the source copy.
11	Revert to server	Click on this button to adopt the target copy.
12	Next	Click this button to continue updating.
13	Cancel	Cancel updating and close the window.

*The description of **Update** window when have conflicts*

Reminder of updating

When your teammates have committed their work to server, you will see a notification appear at the bottom right of Visual Paradigm window, reminding you to perform an update to obtain the changes. By clicking Update, the latest changes will be updated from server to your project. The effect is like performing a manual update but without the need to go through the menus/toolbars.



Update notification

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Revert local modifications

You can undo all the non-committed modification you made in the local project copy. The operation of giving up non-committed modifications is called *Revert*. To revert:

1. Select **Team > Utilities > Revert Local** from the toolbar.
2. Click **Yes** when you are asked for confirmation.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

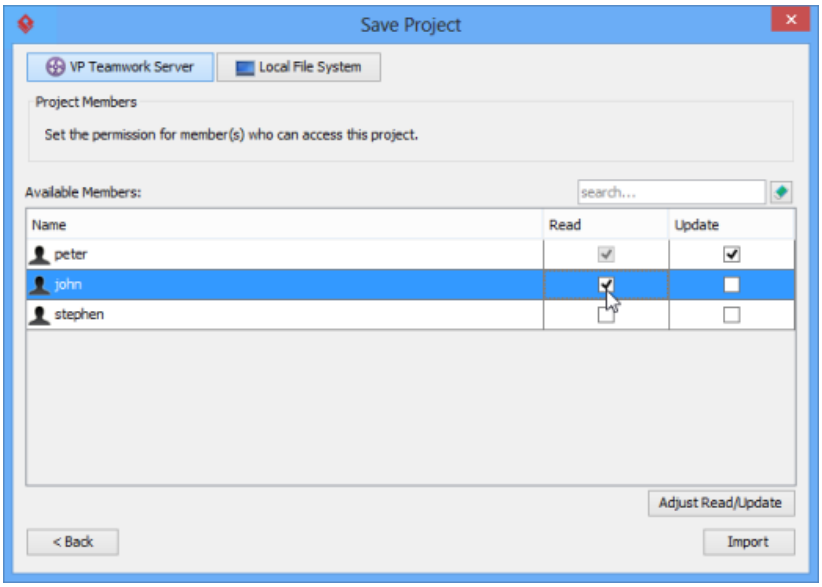
Import project to server

Import project refers to the process of adding project file in server, so that team members can check that out and start working on it.

Import project via project saving

This method of importing project allows you to save the currently opening project into the server. Perform the steps below to import the opening project to server.

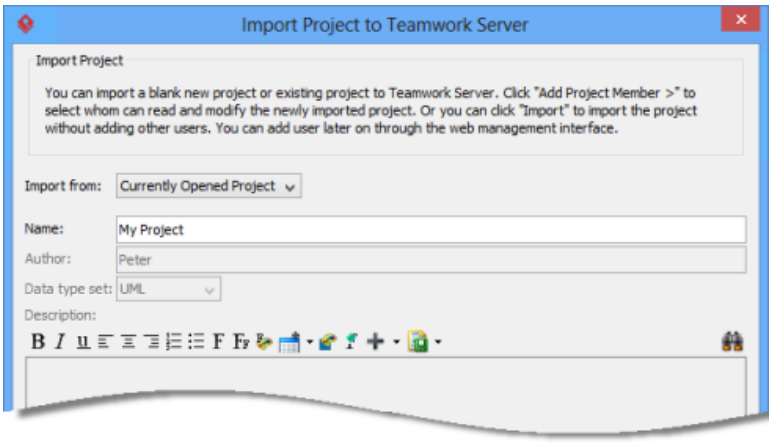
1. Select **Project > Save as** from the toolbar.
2. In the **Save Project** window, keep **VPository/ VP Teamwork Server** selected at the top of the window. If necessary, change the project name.
3. The **Enable PostMania** option allows you to enable the social networking platform bundled by VPository/VP Teamwork Server. If you want to allow you and your teammates view and comment on your design via PostMania, keep the option enabled.
4. Click **Import** to immediately create the opening project in server. By doing so, you will be the only one who can access the project. But of course, you or the manager can add other members to the project later on via the web interface.
5. If you want to assign members to your project now, click **Add Project Member>**. In the **Project Members** screen, grant either or both the **read** and **update** rights to the other members/groups. The **Read** permission means that member can only checkout the project and read its content. The **Update** permission means that member can both read the project content and commit changes to server. When ready, click **Import** to continue.



Granting member the read right to a new project

Import project via Teamwork Client

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. In the **Teamwork Client** window, select **Project > Import Project to Repository** from the main menu.
3. In the **Import Project window**, select the source of project file:
   Currently opened project - This will import the opening project to server as a new project.
   Blank new project - This will import a blank new project as a new project. If you select this option, enter/modify the author, data type set and description of project.
   Import existing project - This will start with an existing .vpp project file. If you select this option, click **...** and choose the .vpp file to import to server.



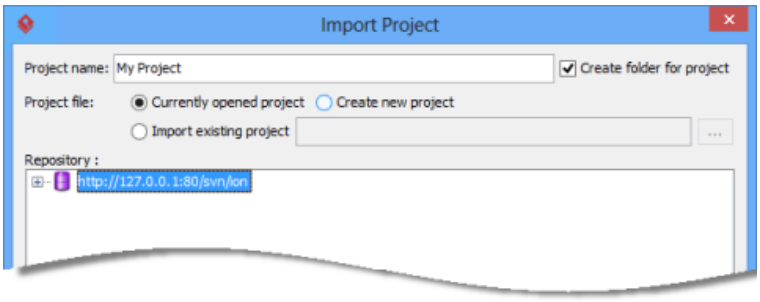
Import project

4. Enter/modify the project name.

5. Click **Import** to immediately create the opening project in server. By doing so, you will be the only one who can access the project. But of course, you or the manager can add other members to the project later on, via the web interface.
6. If you want to assign members to your project now, click **Add Project Member>**. In the **Project Members** screen, grant either or both the **read** and **update** rights to the other members/groups. The **Read** permission means that member can only checkout the project and read its content. The **Update** permission means that member can both read the project content and commit changes to server. When ready, click **Import** to continue.

Import project via Teamwork Client

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. In the **Teamwork Client** window, select **Project > Import Project to Repository** from the main menu.
3. In the **Import Project window**, enter the project name.
4. Next to the name field you can see the option **Create folder for project**. When checked, a folder with same name as project will be created to contain the project.
5. Select the source of project file:
   Currently opened project - This will import the opening project to server as a new project.
   Create new project - This will import a blank new project as a new project.
   Import existing project - This will start with an existing .vpp project file. If you select this option, click **...** and choose the .vpp file to import to server.



Project sources

6. In the **Repository** pane, select the folder where the project file will be stored.
7. You can optionally describe the import action by entering your comment in the **Comment** field.
8. Click **OK**. This will import the project.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Advanced Team Collaboration Features in Visual Paradigm

Check out the advanced team collaboration features supported by Visual Paradigm.

Branching

Branch is defined as a copy of work derived from a certain point in the trunk. It sets up an extra space for users to work on and make modifications without disturbing the trunk. As soon as the modifications in branch are done, you can merge it back to the trunk.

Tagging

Tag refers to a named version of your work at a point of time on the trunk.

Roll back past revision changes

Undo changes made in specific revision(s).

Export revision

Export revisions to project files.

Managing Teamwork Files

Add, commit teamwork files in Teamwork Files pane

Diagram protection

You can prevent a diagram from being edited by another team member by locking it. In this chapter, you will see how to protect diagram by locking.

See the evolution of design using Visual History

See the evolution of design. Visual History features a side-by-side viewer that allows the comparison of diagram over revisions.

Design recovery with Visual History

Recover your design by restoring your diagram from old revision. Run Visual History. Get changes back in an instant.

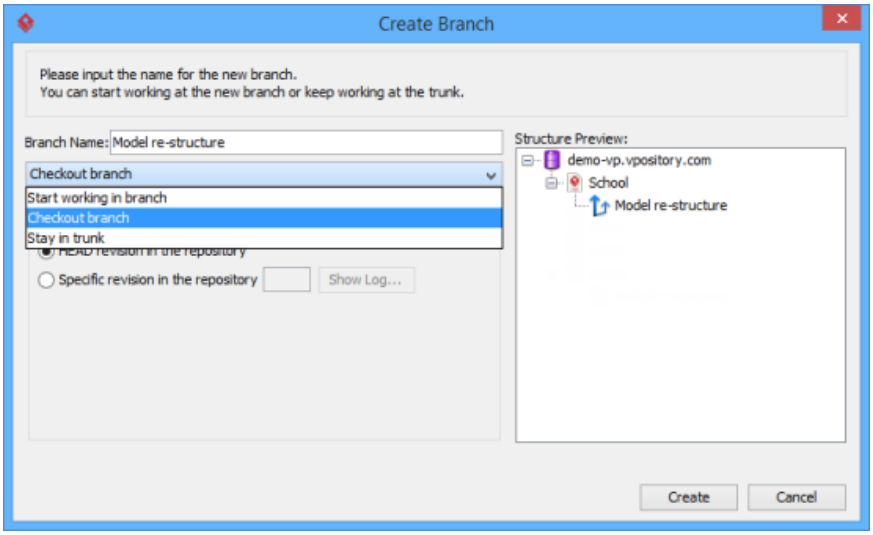
Branching

In terms of team collaboration, a trunk refers to the main stream of development. If you create a new project, you will spend the majority of your time making changes and committing them to the trunk of your repository.

Branch is defined as a copy of work derived from a certain point in the trunk. It sets up an extra space for users to work on and make modifications without disturbing the trunk. As soon as the modifications in branch are done, you can merge it back to the trunk.

Creating a branch

1. Select **Team > Utilities > Branch...** from the toolbar.
2. When the **Create Branch** window appears , enter its name in the **Branch Name** field for your new branch and select whether to start working on the branch or to stay in trunk. Click **OK** to confirm.



*The **Create Branch** window*

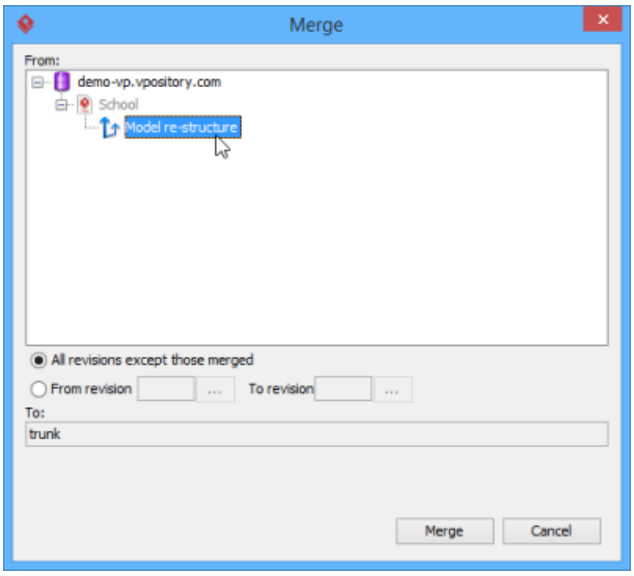
The three options in the drop-down menu are **Start working in branch**, **Checkout branch** and **Stay in trunk**. Select **Start working in branch** means you create, checkout and open a new branch and then start working on it. Select **Checkout branch** means you create and checkout a new branch but just keep it in repository for working on it later on. Select **Stay in trunk** means you create a new branch but do not check it out and do not show it in repository either.

> **NOTE:** For Teamwork Server, you can create branch from a specific revision of trunk by selecting **Specific revision in the repository** and entering the revision number. On the other hand, select **HEAD revision in the repository** if you want to create a branch from the latest revision in trunk. Finally, click **OK** button to confirm.

Merging a branch

When the development activity of branch has been completed, you can optionally merge the branch back to trunk. To merge:

1. Work in trunk. Select **Team > Utilities > Merge...** from the toolbar.
2. When the **Merge** window appears, select the branch you want to merge to the working trunk. A specified revision of project can be merged both from/to by selecting **From Revision** and entering the revision number of from and/or to revision. You can also click on the **Show Log...** button to popup a window with revisions listed and then select the revision from the list. Click **OK** to continue.



*The **Merge** window*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

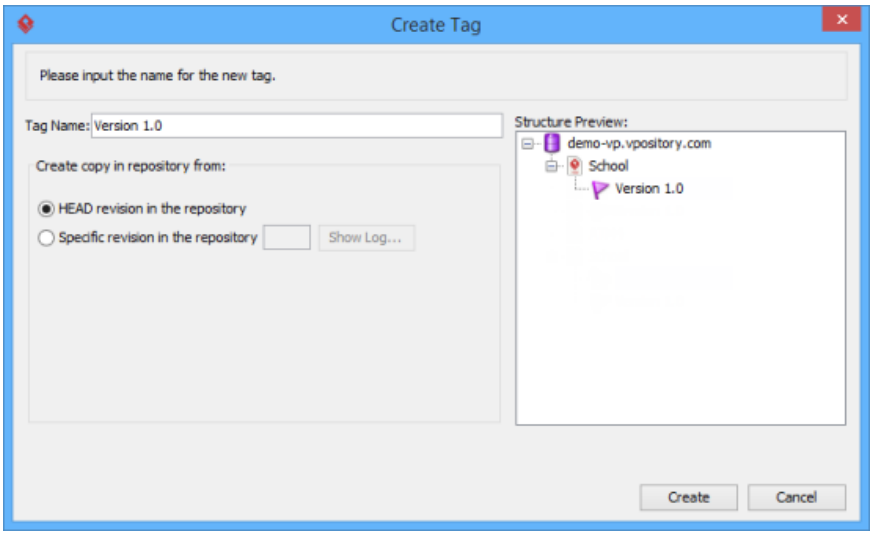
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Tagging

Tag refers to a named version of your work at a point of time on the trunk. The best application of a tag is to create it for every major release or milestone. Note that you cannot merge a tag to the trunk nor commit it to the server.

To create a tag:

1. Select **Team > Utilities > Tag...** from the toolbar.
2. In the **Create Tag** window, enter the name of the new tag in the **Tag Name** field. Click **OK** to confirm.



*The **Create Tag** window*

NOTE: For Teamwork Server user, check **HEAD revision in the repository** if you want to create a tag from the latest revision of trunk while check **Specific revision in the repository** if you want to create a tag from a specific revision.

Related Resources

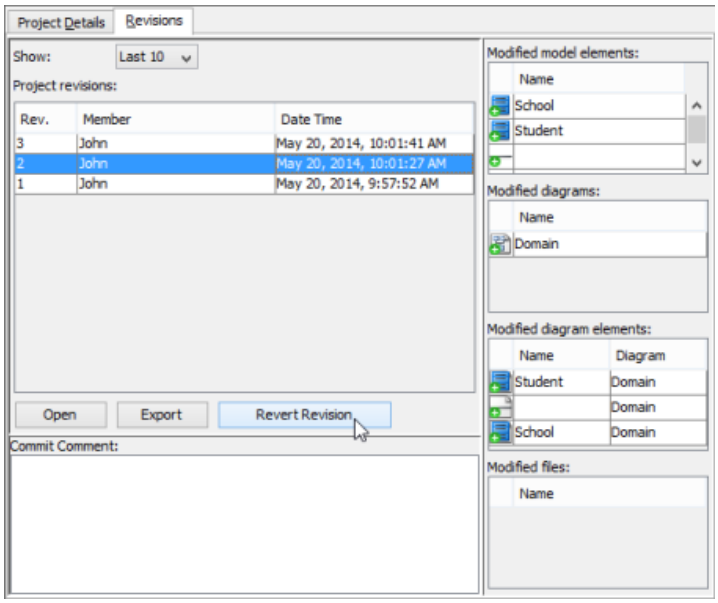
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Roll back past revisions changes

Visual Paradigm maintains the histories of changes team members made. If changes were made by mistake, you can undo it by performing a revert on the revision where the change was made in.

Select **Team > Utilities > Open Teamwork Client...** from the toolbar to open the **Teamwork Client**. In **Teamwork Client**, all your project's revisions are shown in **Project revisions** under **Revisions** tab. Select the number of project's revisions you want to list from the drop-down menu **Show**. Select the revision(s) that contain the unwanted changes and click **Revert Revisions.**



Revert a revision

A new revision will be created for the result of reverting. Therefore, you are asked to commit here. Sometimes, the undo action may cause a conflict with a more recent revision or revisions. Just like the normal commit process, you must resolve all the conflicts in order to continue committing. You have to resolve conflict either by overwriting or reverting the change. Overwrite means to adopt the server copy's change, while Revert means to adopt the local's change.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

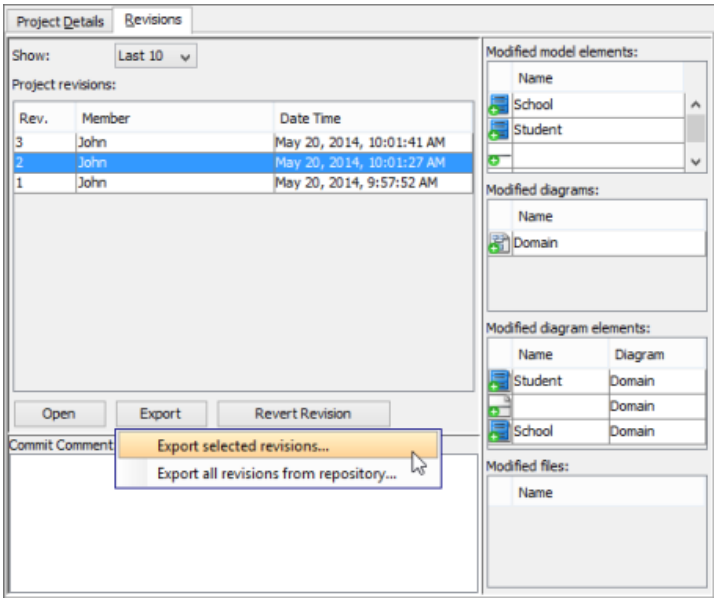
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export revision

You can export revision(s) from server to your machine for checking the project content of certain phase of development. You can export specific revision, or all revisions.

Export the selected revision

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. In the **Teamwork Client** window, open the **Revisions** tab.
3. Select the revisions(s) you want to export under **Project revisions**. Click **Export** and choose **Export selected revisions...** from the pop-up menu.



Export selected revision

4. In **Select Directory** window, select an existing folder or make a new folder for storing the selected revision(s). Click **OK** to proceed.

Export all revisions from repository

1. Select **Team > Utilities > Open Teamwork Client...** from the toolbar.
2. In the **Teamwork Client** window, open the **Revisions** tab.
3. Click **Export** and choose **Export all revisions from repository...** from the pop-up menu.
4. In **Select Directory** window, select an existing folder or make a new folder for storing all revisions. Click **OK** to proceed.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Managing Teamwork Files

When modeling, there may be external resources you want to attach to a model which help to describe it in detail or include data that cannot be modeled within the tool, like a text document. For example, you may want to attach a scanned image of a transaction receipt to a diagram that describes the transaction process so that the analyst can design the new system based on the image. Or maybe an image file showing the user's expectation of the user interface.

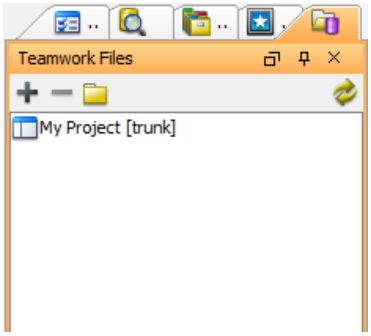
The reference function enables you to add file references to model elements and diagrams. You, as a user who work in a team-based environment with VPository/Teamwork Server do not need to copy any referenced files for other team members to open. Instead, you can commit your model along with the referenced files to the server by referencing a **teamwork file**. Teammates can then get the referenced files from server and open them in their environment.

In VP, there is a folder under the workspace for storing files that are readily being committed to VPository/Teamwork Server for versioning. Those files are called **teamwork files**. The file revision will follow the model, i.e. whenever you open a particular revision of work from server, the file of that revision will be obtained and thus there is always a consistency between the file and the design.

NOTE: "Teamwork Files" is supported by VPository and Teamwork Server Corporate Edition.

The Teamwork Files pane

Teamwork Files pane is a user interface component where you can manage and see the teamwork files. You may show it by selecting **Team > Teamwork Files** from the toolbar or press **Ctrl-Shift-F**.

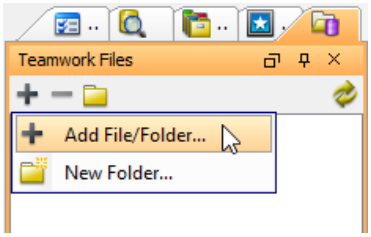


The Teamwork Files pane

Adding teamwork files/folder

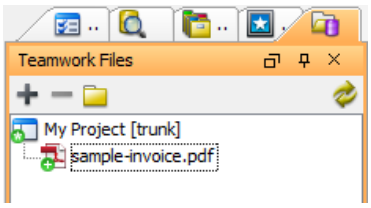
Adding a teamwork file/folder will copy that file/folder to under the workspace folder. Once added, the teamwork file/folder can be versioned by VPository/Teamwork Server. To add a teamwork file:

1. Select the node in **Teamwork Files** pane for creating file/folder. You may select the project root node or a folder node.
2. Click on the plus button in **Teamwork Files** pane and select **Add Files/Folder...** from the popup menu.



Add file/folder

3. In the **Add** window, select the file or folder to add and click **Open**.
4. You are prompted to confirm copying the file or folder to workspace. Click **OK**.



Teamwork file added

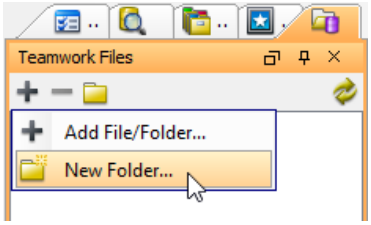
Opening the containing folder of file

You can open the containing folder of a teamwork file/folder so that you can access it with file explorer provided by the operating system. Right click on the file/folder and select **Show in Folder** from the popup menu.

Organizing teamwork files with folder

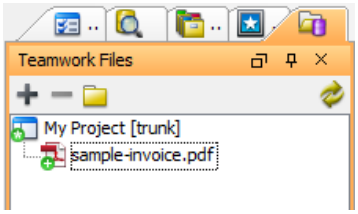
You may organize the added physical files/folder by adding folders. Note that when you create a folder, a physical folder will be created in your workspace folder. To add folder:

1. Select the node in **Teamwork Files** pane for creating folder. You may select the project root node or a folder node.
2. Click on the plus button in **Teamwork Files** pane and select **New Folder...** from the popup menu.



Create a folder

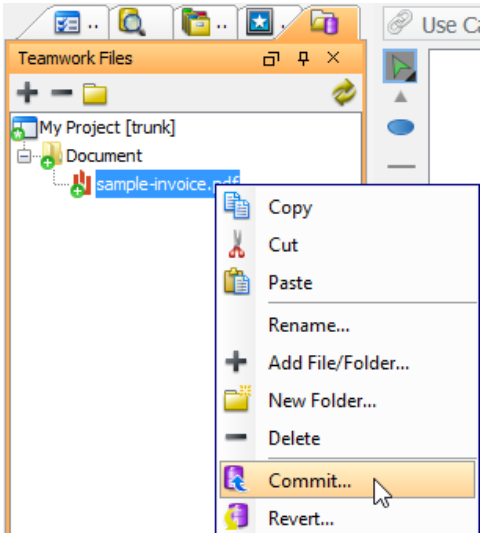
3. Enter the name of folder in the popup dialog box. Click **OK**. Once a folder is created, you may add files/folder in it or drag existing teamwork files/folders into it.



Folder added

Committing teamwork files

You may make use of the commit function to commit teamwork files to VPository/Teamwork Server. When you perform a commit globally (i.e. commit the whole project), teamwork files will get committed. If you want to perform commit on specific teamwork file/folder, right click on it and select **Commit...** from the popup menu.



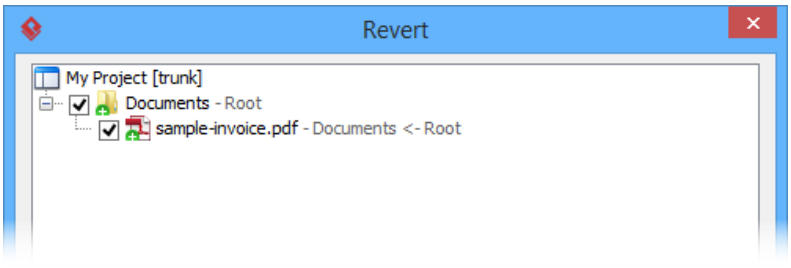
Commit teamwork files to server

Removing teamwork files

To remove a teamwork file from Teamwork Files pane, select the file/folder to remove and press **Delete**. Alternatively, right click on the file/folder and select **Delete** from the popup menu.

Revert changes

Revert is the process of discarding non-committed modifications. To revert a file/folder, right click on the file/folder and select **Revert** from the popup menu. In the Revert window, select the files to revert.



Revert changes of teamwork files

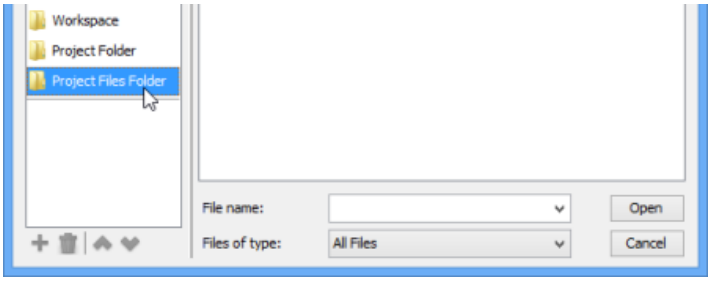
Copy, Paste and Cut

Copying, pasting and cutting of teamwork file can be done by two ways. One is by hotkey **Ctrl-C**, **Ctrl-V** and **Ctrl-X**. Another one is by right clicking on the file(s) and selecting **Copy**, **Paste** and **Cut** from the popup menu.

Referencing a teamwork file

When you try to add a file reference to a teamwork file, you may manually navigate through the folders to locate the teamwork file to add. Or to speed things up, Visual Paradigm allows you to create a shortcut that brings you to the folder where the teamwork files are stored. In order to have such shortcut, you need to define a user path of project files to create a corresponding shortcut in file chooser. After that, you may add the reference easily.

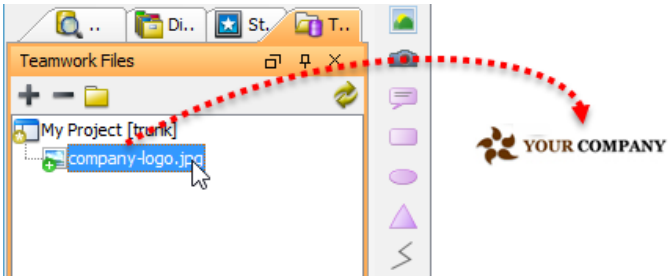
1. Select **Windows > Application Options** from the toolbar to open the Options window.
2. In the **Options** window, select **User Path** on the left hand side.
3. On the right hand side, click **Add...** and select **Project Files Path** from the popup menu.
4. Click **OK** at the bottom of the **Options** window to close it.
5. From now on, when you try to add file reference to model elements, you can select the shortcut Project Files Folder in file chooser to jump to the folder where the teamwork files are listed.



Project files folder

Creating image shape from a teamwork file

You can show a teamwork file in diagram if that file is in one of the supported images types - .jpg, .jpeg, .gif, .png, .bmp). To achieve this, simply drag and drop the image file from **Teamwork Files** pane to the diagram.

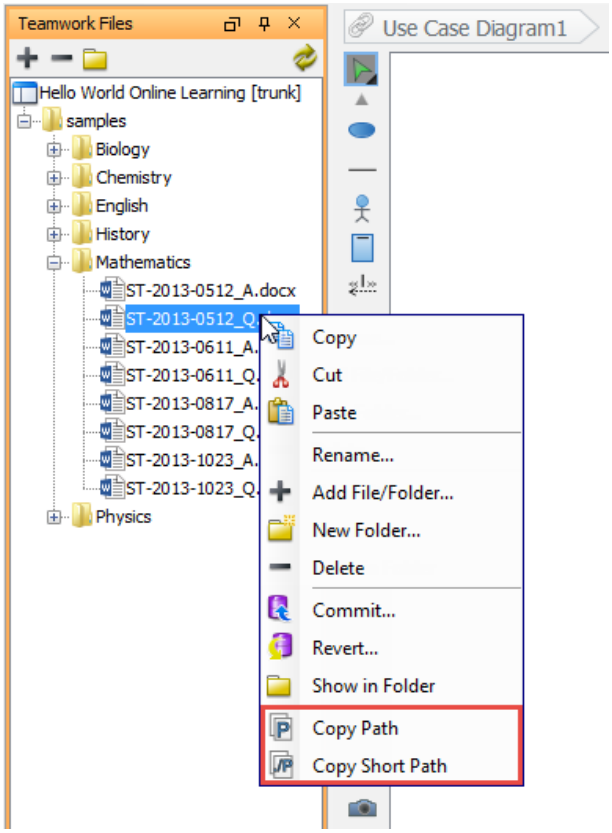


Dragging image file from Teamwork Files pane to diagram

Copying the path of a teamwork file

Sometimes, you may want to save a new file to a teamwork folder. In order to do this, you have to know the filepath of the folder, which is what the **Copy Path** feature can help. There are two kinds of path you can copy - full path and short path. Full path is what known as an absolute path. You can save a file to a specific location by providing its full path. On the contrary, a short path is a relative path, which is relative from the project. A sample of short path would be *MyProject\files\MyTWFile.txt*. Compare to full path, short path is short and machine-independent, making it a good method to express a filepath within a team.

To copy the path of a teamwork file, right click on that file and then select **Copy Path** or **Copy Short Path**. The path will then be copied to the system clipboard automatically.



The Copy Path menus

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram protection

When you are working on a project with your team members, you probably don't want your work to be modified by others by mistake. To protect your work, you can lock your current diagram with password to make it uneditable by others.

Locking diagram

1. To lock a diagram which you don't want it to be modified by others, right click on the diagram's background and select **Diagram Content > Lock ...** from the pop-up menu.

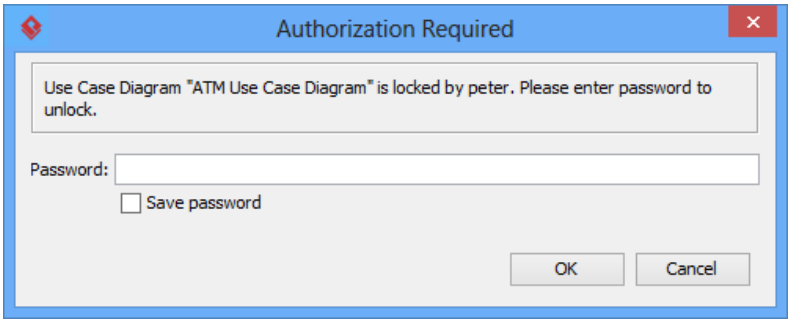
2. Enter your password in the pop-up **Lock** window and then click **OK** button. For your safety, always keep your password secret.

Enter password

Your diagram is then locked. Here is a detailed description of the options in the Lock window:
Lock by password: If you check this option, you need to set password for locking diagram. If you uncheck it, you don't have to enter password but you can still lock the diagram.
Save password: You can save your password in the project under your user account by checking this option. Thereafter, you don't have to enter password when you edit the diagram. Otherwise, you need to enter password to edit the diagram after you uncheck it.
Show password: Check this option to reveal the password. The password will be hidden by asteriks when you keep it unchecked.

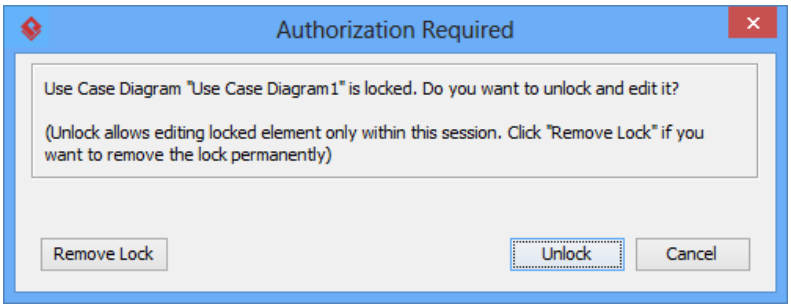
Trying to edit a locked diagram

If your team members want to modify any model elements of the locked diagram, they need to provide the password set by you in the pop-up **Authorization Required** window in order to proceed modification.



Require to enter password to proceed editing

If you try to modify any model elements of the diagram locked by you, an **Authorization Required** window will prompt out. Click **Remove Lock** button if you want to unlock the diagram permanently. Click **Unlock** button if you want to unlock it temporarily.



Unlock the diagram

NOTE: If you uncheck **Save password** when you lock the diagram, you will be asked to enter password to edit the locked diagram.

Moreover, you can also unlock the diagram permanently in **Configure Lock** window. Right click on the diagram's background and select **Diagram Content > Configure Lock...** from the pop-up menu. Choose **Remove lock** and enter password.

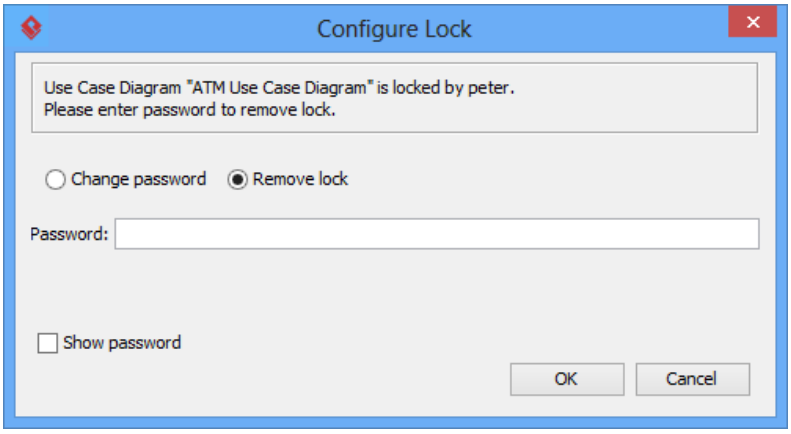
Removing lock

You can remove the lock or change the password after you lock the diagram.

1. Right click on the diagram's background and select **Diagram Content > Configure Lock...** from the pop-up menu.

> **NOTE:** **Configure Lock...** option is available on the pop-up menu only after you have locked the diagram.

2. In **Configure Lock** window, choose **Change password** if you want to change the current password and set a new password; otherwise, choose **Remove lock** if you want to remove the lock permanently. Click **OK** button to confirm.



Remove lock

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

See the evolution of design using Visual History

There are many reasons why you want to know how a diagram looked like at some point in time. First, you may find the latest design not feasible and you want to see how the feasible one looked like in the past. Second, you may spot a flaw/mistake and you want to recover it by referencing the correct design you did before. And for diagrams like business process diagram, ArchiMate and business motivation model, you can observe their changes to determine the changes the business has made over the years. You can easily tell what and why the design and hence the business was changed. Such information can help you to plan the future with minimized risks.

While seeing the evolution of diagram is pretty much the most common use case, you can also see the evolution of specific shape or even a model element, say, to know what a package contained at some point in time.

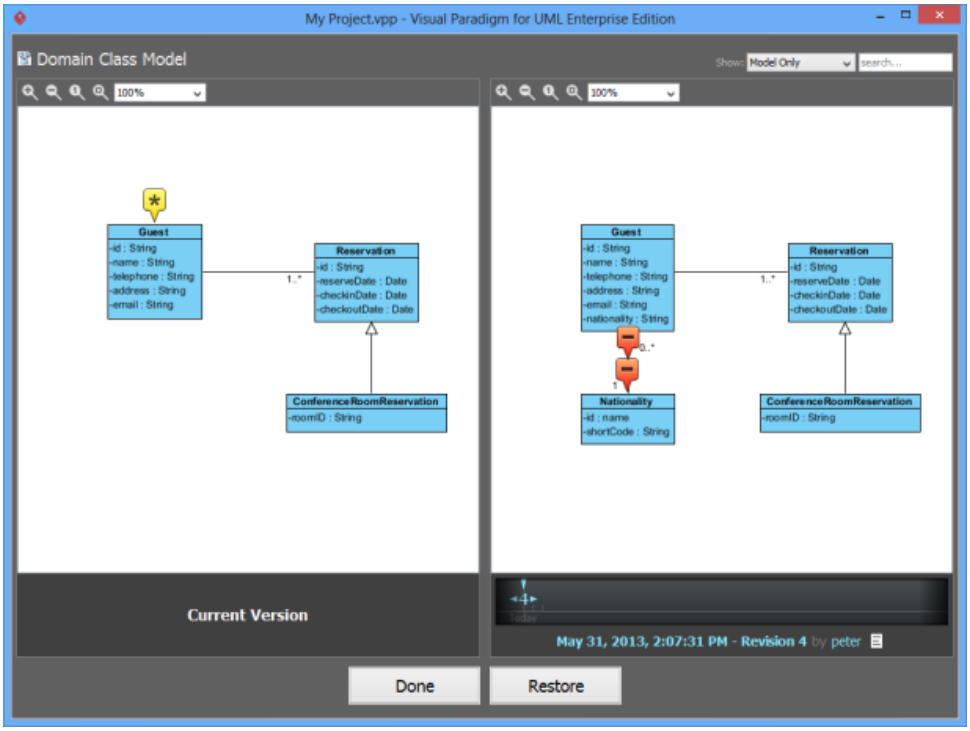
Visual History enables you to see the evolution of your design. It features a side-by-side diagram viewer that allows the comparison of diagram over revisions. With the revision slider, you can easily slide across revisions to compare the diagram in the latest revision and a specific revision in the past.

Another important function of Visual History is to help recovering your work by restoring from an old revision. You can read the next chapter for details.

Browse a diagram in old revision with Visual History

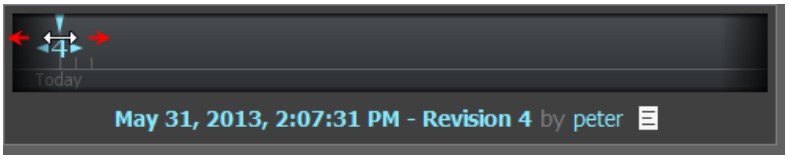
To see how a diagram looked like in the past, perform the steps below.

1. Right click on the background of a opening diagram and select **Visual History...** from the popup menu.
2. This opens the **Visual History** window. At the top of the window you can see the diagram name as well as the drop down menu **Show**, which controls the kind of difference you want to focus on when comparing revisions. We will talk about it in the coming sections. In the middle, you see two diagram viewers. The left hand side is the diagram in the latest revision, while the right hand side is the same diagram in an old revision. You can drag the viewers to pan the diagrams. If you are comparing a model element or a shape instead of a diagram, you will see the properties of that element/shape listed here.



The Visual History window

3. Under the diagrm viewer on the right, you see the time line. Press on the revision number and drag right or left to move to older or more recent revisions. The diagram viewer will be updated to show the diagram in the chosen revision. Note that only the revisions that reflect a change in diagram are listed on the time line. You can slide between revisions to see the evolution of diagram and to stop at certain revision to analyse and compare that revision with the latest revision, as shown in the viewer on left hand side.

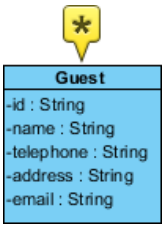


The time line

Compare diagrams with the help of visual indicator

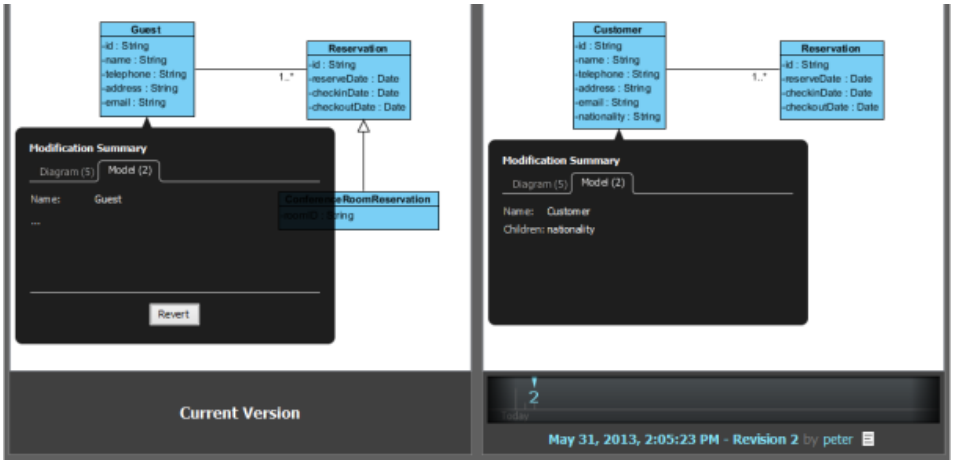
While visual difference of diagrams can be identified easily by bare eyes, you are unable to tell the changes of shape/model element properties that are not presented on diagram, such as the modification of element documentation. In order to know this kind of change, you need to check the visual indicators that appear on the diagram.

Each indicator represents a difference found when comparing the latest revision and the revision as selected in the time line. The pointer of indicator tells you where the difference is.



A modified class

By clicking on an indicator, you can see a list of modified properties, categorized by diagram level property (e.g. shape position, color, etc) and model level property (e.g. name, documentation of model element, etc).

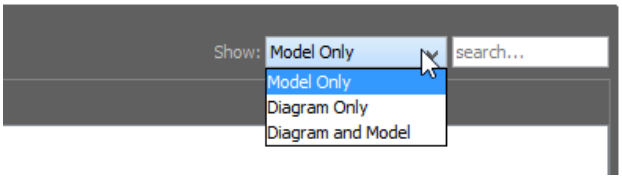


The 'Guest' class was named 'Customer' before, and owned an attribute nationality

Adjust the type of difference to indicate on diagram

There are two kinds of differences **Visual History** can identify when comparing diagrams - model and diagram. Model refers to all sort of model element properties change, such as the renaming of element, addition of file reference, modification of documentation, etc. Diagram refers to all sort of shape properties change, such as shape position, shape color, etc.

If you want to focus on model element properties rather than the view level properties, you can click on the drop down menu **Show** at the top right of the **Visual History** window and select **Model Only**. There are also options like **Diagram Only** and **Diagram and Model** to suit different needs. The diagram viewers will be updated to show or hide away indicators after considering or ignoring the differences base on your choice.



Show only model element properties changes

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Design recovery with Visual History

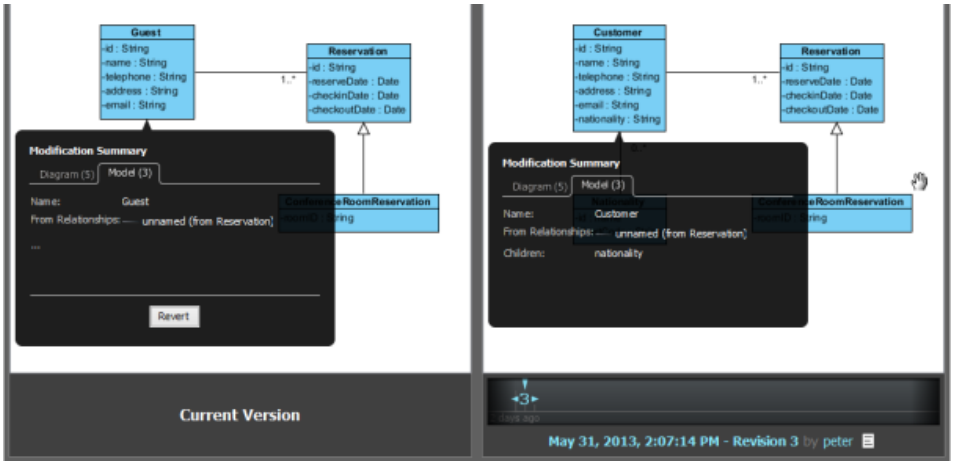
Aside from browsing a designs in old revision, **Visual History** also allows you to restore a diagram from old revision. Let's say you have deleted a shape by mistake and you want to get it back. You can locate in Visual History the revision that has the shape intact, and get it back by clicking the **Revert** button (You will learn how in this page). Besides restoring changes one by one, you can restore the whole diagram. By doing so, the whole diagram will be overwritten by the old version.

Revert modification

1. In the **Visual History** window, when you are comparing diagrams, there are indicators that appear on diagram showing the areas of changes. Click on an indicator to popup the modification summary.

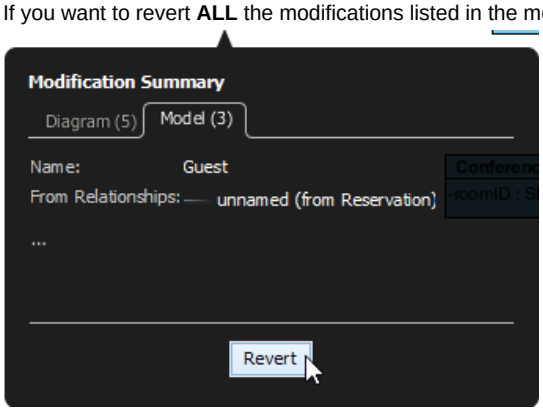
To review modifications

2. Review the modifications.



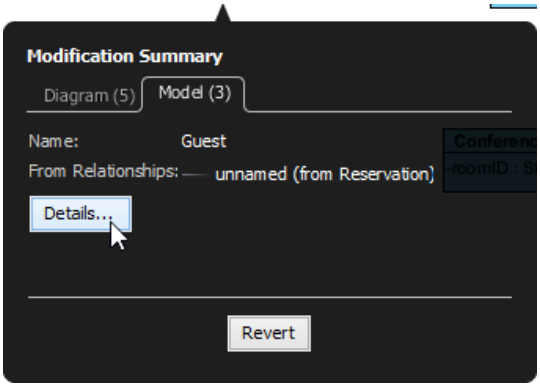
Review modifications

3. If you want to revert **ALL** the modifications listed in the modification summary, click the **Revert** button at the bottom of the summary popup.



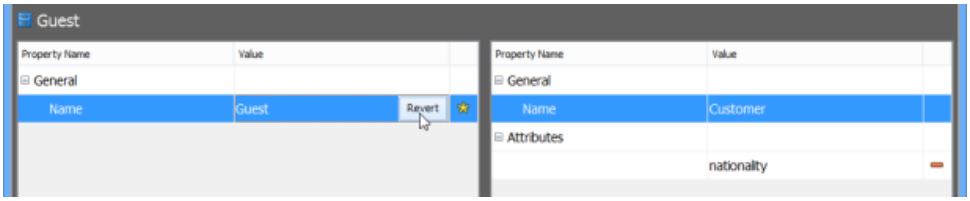
Revert modifications

4. If you want to revert a specific modification, move the mouse point below the last modification to show the **Details...** button. Click on it.



Review modification details

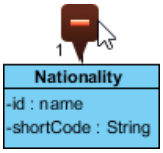
5. The modifications are listed in in a table. You can click on individual modification and then click the **Revert** button to revert it.



Revert a modification

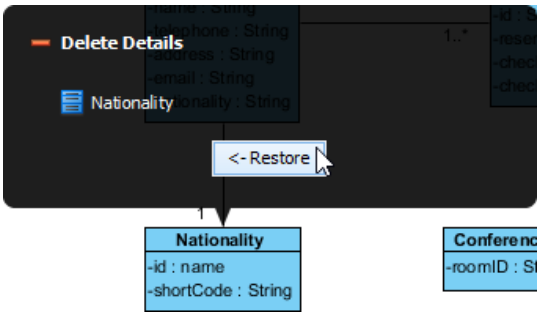
Restore deleted shape

1. In the **Visual History** window, when comparing diagrams, click on the indicator that appears on top of the deleted shape.



To get deleted shape back

2. Click the **Restore** button in the **Delete Details** popup to get the shape back.



Restore a deleted shape

Restore the whole diagram

If you want to restore the entire diagram from an old revision, simply click Restore at the bottom of the **Visual History** window.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using PostMania

PostMania Overview

Learn what PostMania is.

The PostMania Page

Learn how to start using PostMania.

Posting Discussion Topic in PostMania

Learn how to post a discussion topic.

Replying in PostMania

Learn how to reply a post.

Watching Diagram

Learn how to get notified by changes.

What is PostMania?

PostMania is a tool bundled with VPository that lets you share the diagrams that you've created in Visual Paradigm with others, and to collect their feedback after they have view your diagrams. Very often, you share diagrams with your partners, clients or colleagues to let them confirm your design or clarify design issues. They can view your diagrams on the web, without any rights to edit them. Here are some of the use cases of PostMania:

- Business user reviews and approves business process design
- Business user reads the latest business process design for working guidelines
- Business user request changes to process plan
- Developer views the latest class diagram to gain get up-to-date system specification and design guideline

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Sharing Diagram

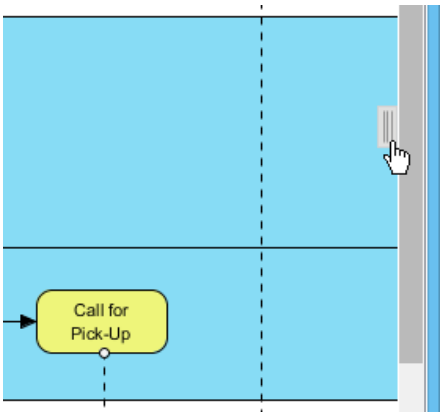
To share a diagram with teammate means to let him or her view the shared diagram, either through a web browser or through mobile device, and to allow him or her to give comments on the diagram, or shapes in the diagram.

Sharing works in diagram based, meaning that when you share a diagram with someone, he/she can only view that diagram, but not the other diagrams, unless the other diagrams have also been shared.

Sharing a diagram through invitation

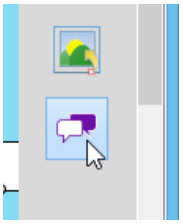
To share a diagram:

1. In Visual Paradigm, open the diagram you want to share.
2. Make sure the diagram have been committed to the server. You can only share a diagram that has been committed to server before.
3. Open the action bar by clicking on the tiny button on the right hand side of the diagram, near the scrollbar, if exist.



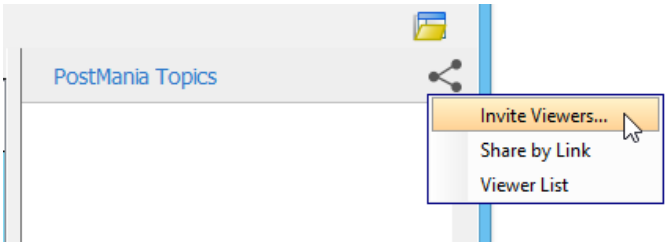
Open the action bar

4. Open the **PostMania Topic Pane** button by clicking on its button in the action bar.



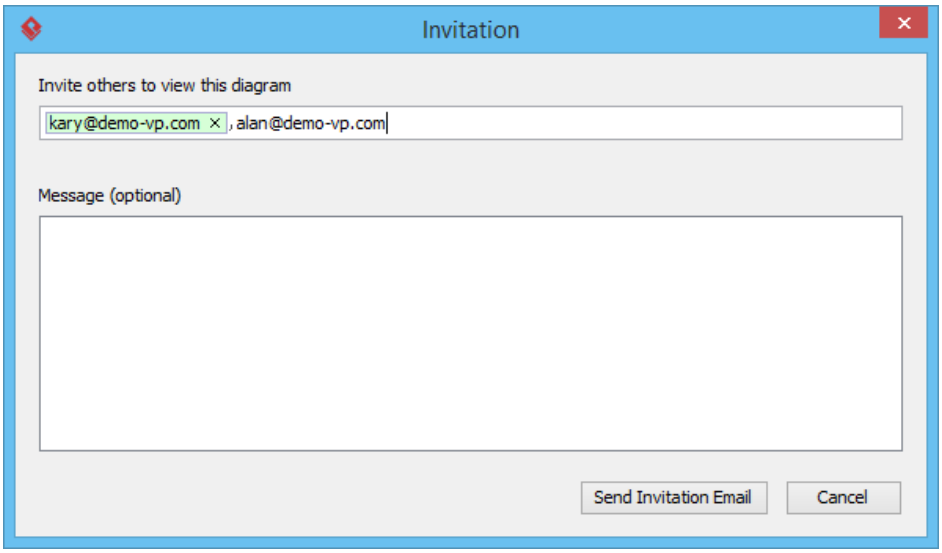
Opening PostMania Topic Pane

5. At the top of the **PostMania Topic Pane**, click **Share** and then select **Invite Viewers...** from the drop down menu.



Invite viewers

6. Enter the name and/or the Email address of the people to share with. If a person to invite is an existing viewer/member, you just need to enter his name and pick him up from the drop down menu. If he is not currently a viewer, please enter his email address. PostMania will send invitation Emails all people specified, to invite them to join PostMania for viewing and commenting on the diagram to be shared.



Entered the name and email address of invitees

7. You can optionally include a message, which will be included in the email PostMania send out.
8. Click **Send Invitation Email**. PostMania will send the invitees invitation Emails in three minutes.

> **NOTE:** If the invitee is an existing viewer, and has opened the notification page/view of PostMania, either in a web browser or in Android apps, he/she will not receive the email notification.

In three minutes, invitees will receive an Email, with subject " **%NAME% has shared a diagram with you**" where **%NAME%** is the person who shared the diagram. He/she has to:

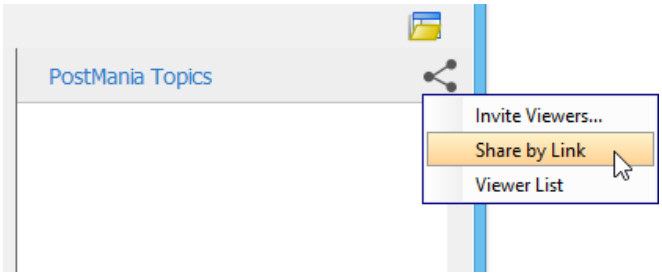
1. Click on **Accept Invitation** in the Email to open the **New Viewer Form** in web browser.
2. Enter the password and click **Accept Invitation** in the form. Then, he/she will be redirected to the diagram shared.

Sharing a diagram by link

To share a diagram with someone, you can enter his name and email address and send him an invitation, which was described above. Besides, you can give him a URL to open the sharing diagram in web browser. If he is already a viewer/member, he will be prompted to login to his VPository account when he visit the given link. Once logged in, he can view the diagram you shared. If he is not a viewer, he will be asked to join PostMania as a viewer to view the shared diagram.

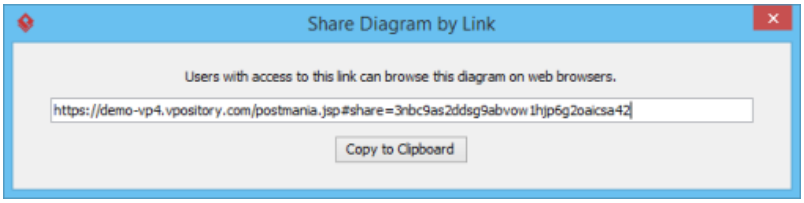
To share a diagram by link:

1. At the top of the **PostMania Topic Pane**, click **Share** and then select **Share by Link** from the drop down menu.



Share diagram by link

2. In the popup window, click **Generate Link**.
3. Copy the link by clicking **Copy to Clipboard**. Now, you can ask others to visit the link to view the diagram.



Copy diagram link

Disabling a shared link

For security reasons you may want to disable a shared link once it has been visited by the authorized person. To disable a shared link:

1. At the top of the **PostMania Topic Pane**, click **Share** and then select **Share by Link** from the drop down menu.

2. In the popup window, click **Disable**. Note that by disabling the URL, new attempts to browse the diagram with that URL will be denied. Yet, viewers who have browserd the diagram before will remain accessible to the diagram.

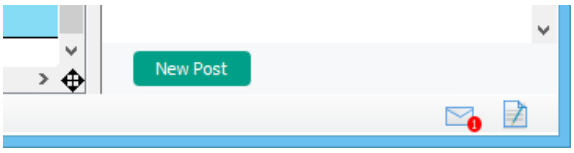
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Getting notified of new posts

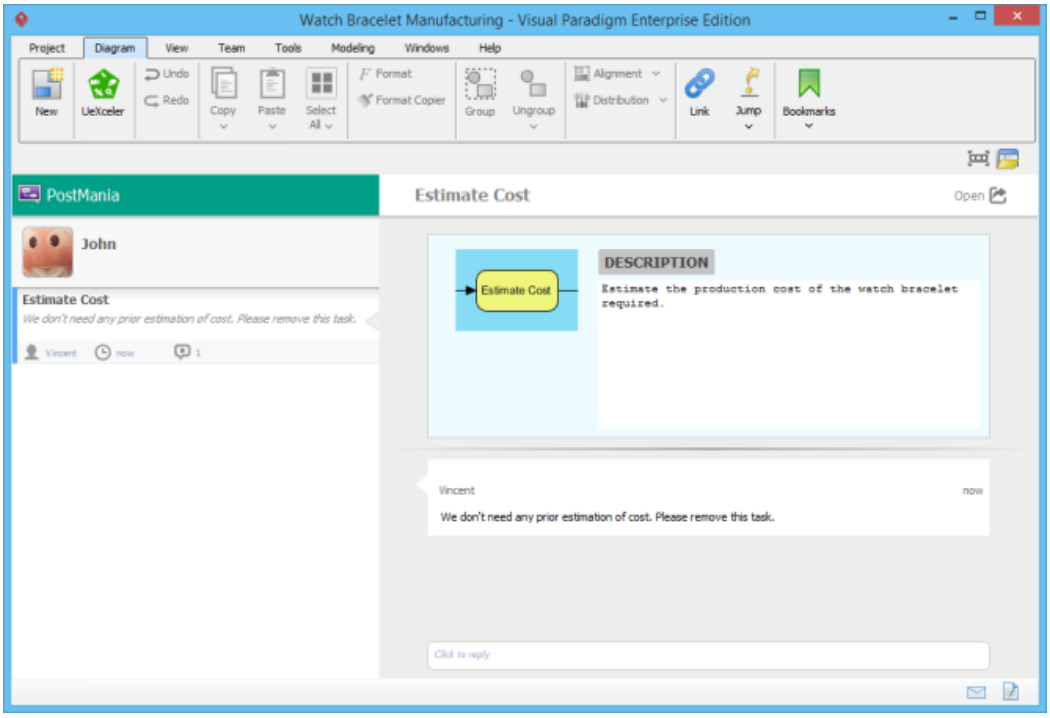
When a viewer has created/replied a post in a diagram shared by you, you will see a notification in the status bar, like this:



Notification received

The number indicates the number of unread notifications. The number will be decreased once you have checked a notification. To check a notification, click on the **Message** icon in the status bar to open PostMania.

The left hand side of PostMania lists the notifications you received, ordered by the receive date and time, with the latest notification appears at the top and the earliest one comes last. You can click on a notification to have an overview of that notification.



PostMania

The overview is displayed on the right hand side of PostMania. It consists of a thumbnail of diagram, and a chunk of unread conversation. Note that the content of a notification consists of only the unread portion of the entire conversation. If you want to view the full conversation, click **Open** at the top right corner of the PostMania. By opening a post, the related diagram will be opened, with the post opened in the **PostMania Topic Pane**, on the right of the diagram. If the post was created around a shape, that shape will be selected.

You can also reply to a post by entering your message below the conversation and then click **Post**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

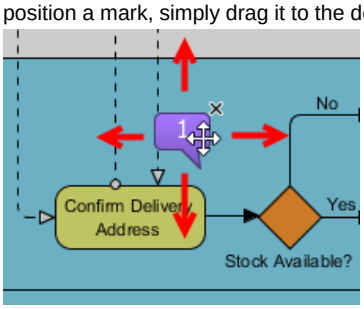
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Posting and replying

If you want to say something about a diagram or its content, no matter for what purpose, can create a post in PostMania. You can create a post to report status ("I have approved."), to ask for action ("Please implement this."), ask whatever questions ("When do you need this?"), seek for confirmation ("May I remove this from the model?"), request changes ("Fix it please"), etc.

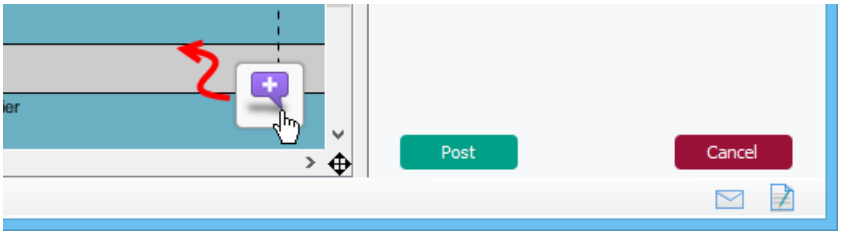
Posting to diagram

1. Open the diagram that you want to create a post in it.
2. Click on the diagram background to make sure no shape has been selected.
3. At the bottom of the **PostMania Topic Pane**, click **New Post**.
4. You see a purple mark symbol appear on the diagram. You can add marks to different parts of the diagram for the different areas of concern. To position a mark, simply drag it to the desired position.



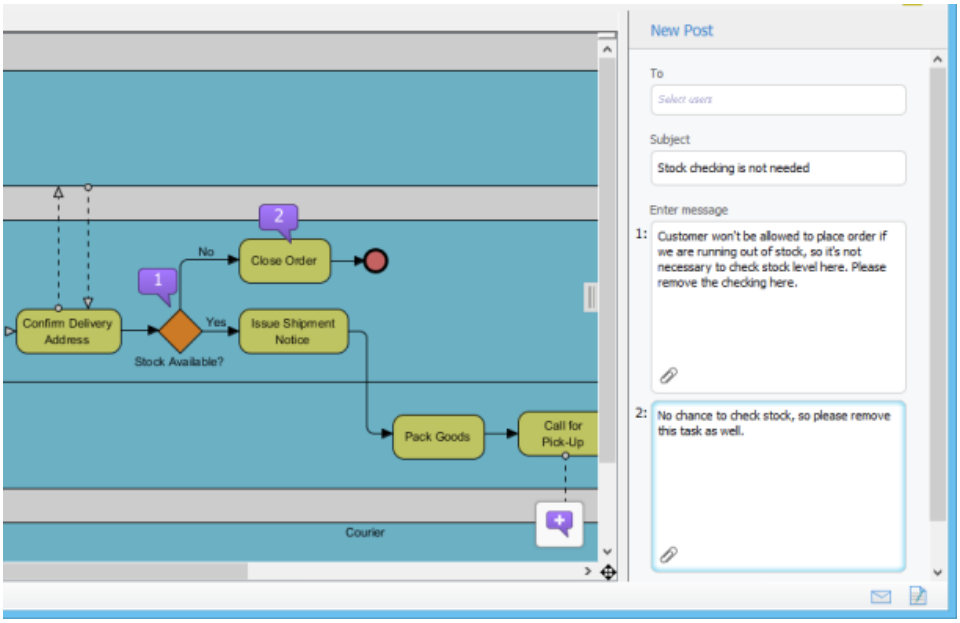
Repositioning a mark

To add a new mark, drag the mark symbol at the bottom right of the diagram, and release it at the desired position.



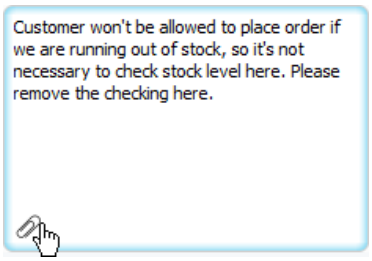
Create a mark

5. In the **PostMania Topic Pane**, enter the subject of post.
6. Enter the comments of the marks.



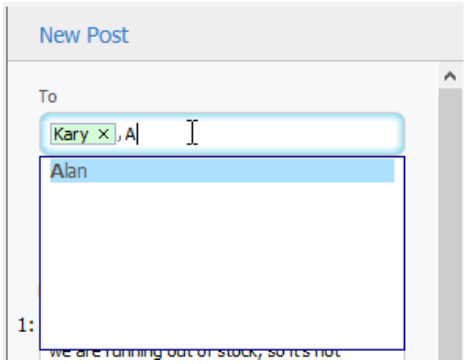
Entering comments

You can optionally attach file reference to each of your post content. To attach file(s), click on the attachment button at the bottom of the post box.



To attach a file to a post

7. Specify the member to notify in the **To** field. Selected members will receive Email notification about your post.

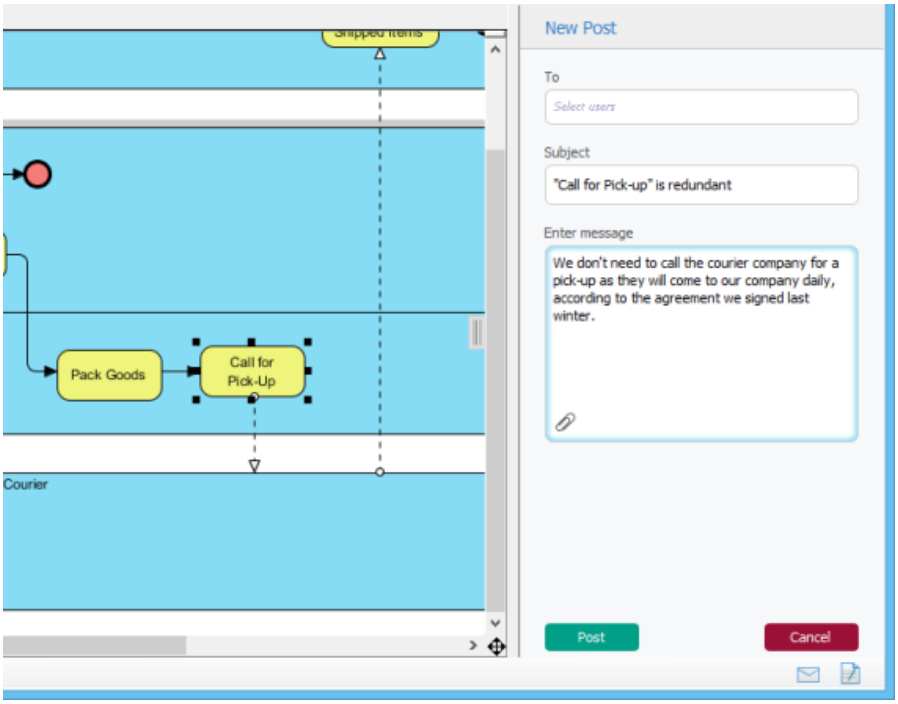


Specifying the team member(s) to notify

8. Click **Post**. Members selected to notify and members who followed the diagram will receive Email notification about your post.

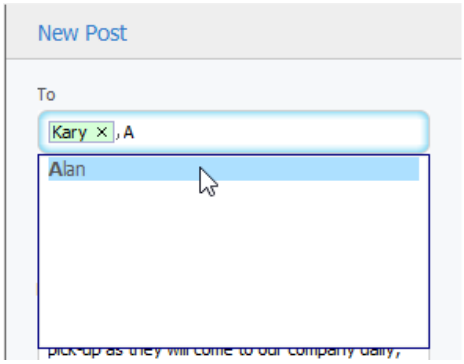
Posting to shape

1. Open the diagram that contains the desired shape.
2. Select the shape.
3. At the bottom of the **PostMania Topic Pane**, click **New Post**.
4. In the **PostMania Topic Pane**, enter the subject of post.
5. Enter the post content.



Entering post contents

6. Specify the member to notify in the **To** field. Selected members will receive Email notification about your post.



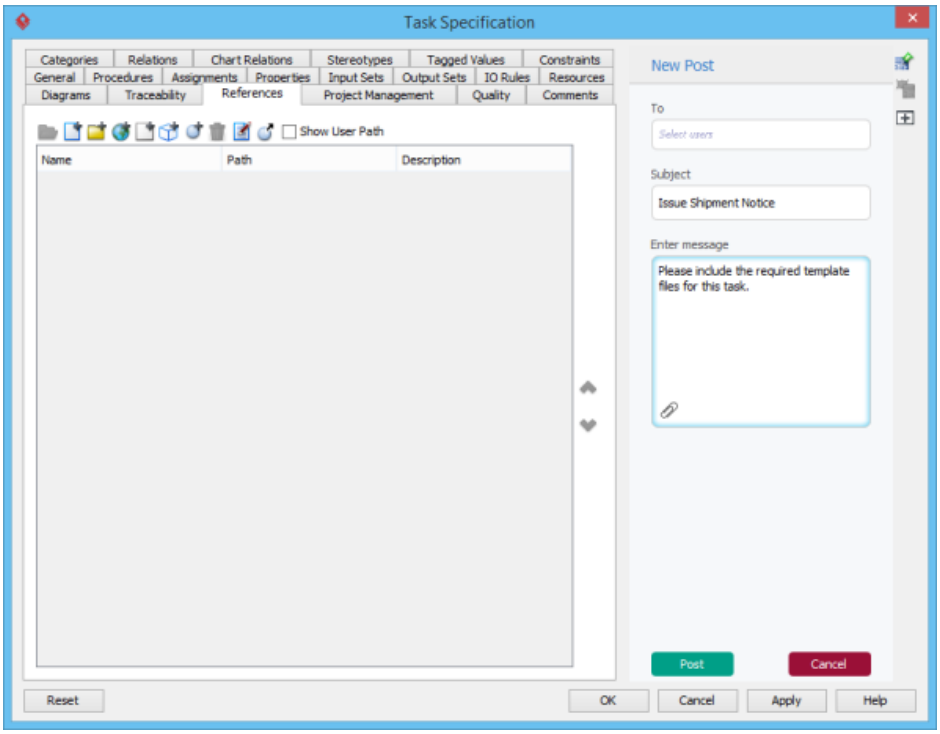
Specifying the team member(s) to notify

7. Click **Post**. Members selected to notify and members who followed the diagram will receive Email notification about your post.

Posting to references of a shape

You can also post comments to references of a shape, via the **References** tab of the specification window.

1. Open the specification window of shape by right clicking on the shape and selecting **Open Specification...** from the popup menu.
2. Open the **References** tab.
3. Create post in the **PostManic Topic Pane** on the right hand side of the specification window.

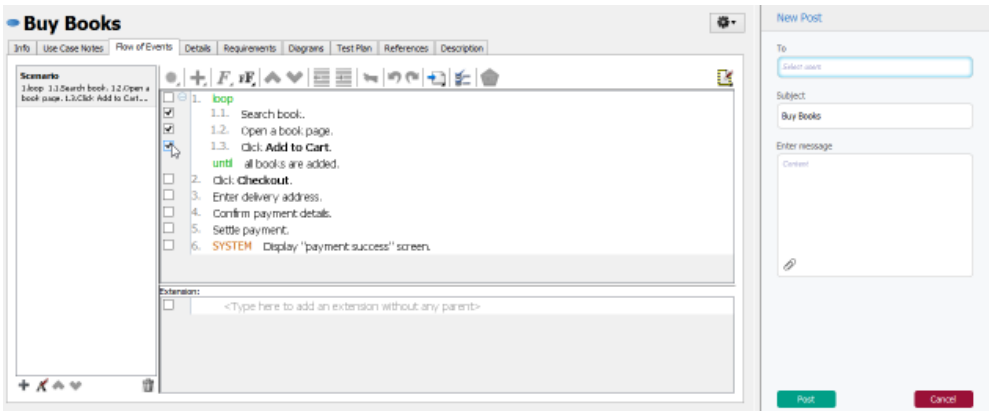


Creating a post to shape's references

Posting to use case flow of events

You can also post comments to the flow of events of use case.

1. Open the flow of events editor of use case by right clicking on the use case and selecting **Open Use Case Details...** from the popup menu.
2. Open the **Flow of Events** tab.
3. Create post in the **PostManic Topic Pane** on the right hand side of the procedure editor. You can select the step to comment, on the left hand side of the working procedure editor.

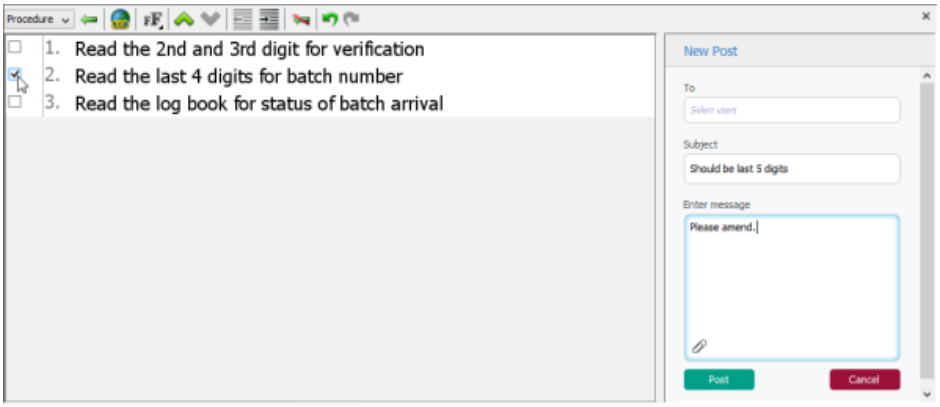


Select steps to comment

Posting to working procedure of BPMN task/sub-process

You can also post comments to working procedure of BPMN task/sub-process.

1. Open the working procedure editor by right clicking on the background of business process diagram and selecting **Show Procedure Editor** from the popup menu.
2. Select the desired task/sub-process to browse its working procedure.
3. Create post in the **PostManic Topic Pane** on the right hand side of the procedure editor. You can select the step to comment, on the left hand side of the working procedure editor.

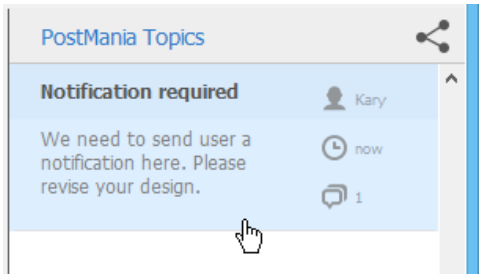


Creating a post to working procedure

Replying a post

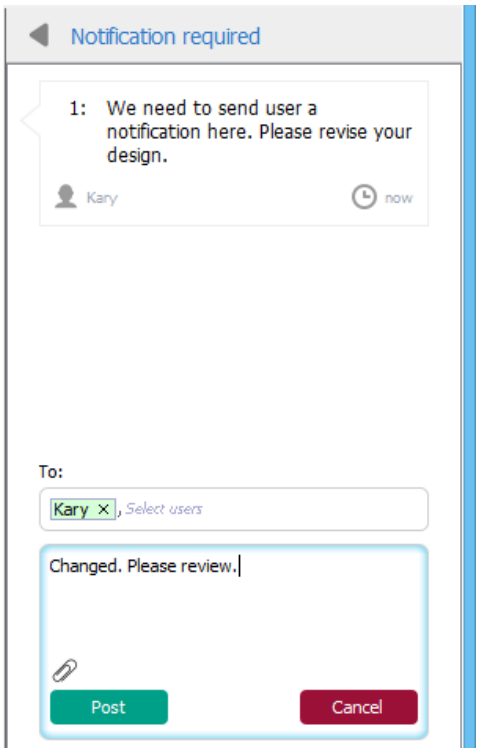
To reply a post:

1. Open the post to reply. You can open it from the **PostMania Topic Pane** or from the notification page.



Opening a post

2. Enter the content of reply.



Entering reply content

3. You may specify the member to notify in the **To** field. Selected members will receive Email notification about your post.
4. Click **Post**.

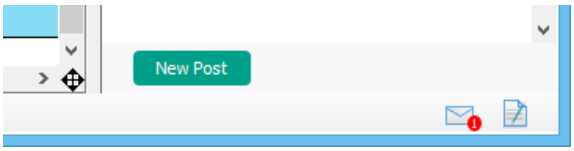
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Following Post

If you want PostMania to inform you when certain post has been replied, you can follow that post. By following a post, you will receive a notification when reply has been made to that post, like this:

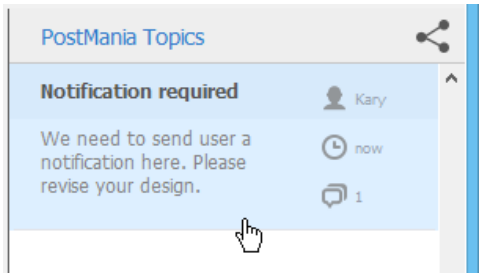


Notification received

The number indicates the number of unread notifications. The number will be decreased once you have checked a notification. To check a notification, click on the **Message** icon in the status bar to open PostMania.

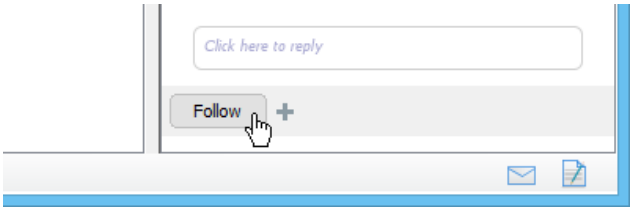
To follow a post:

1. Open the post to reply. You can open it from the **PostMania Topic Pane** or from the notification page.



Opening a post

2. At the bottom of the **PostMania Topic Pane**, click **Follow**.



Follow diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

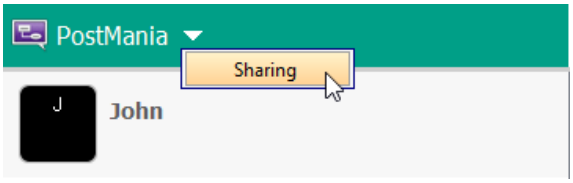
Managing Shared Diagrams (Viewer Based)

You may want to know if someone has viewed a diagram that you shared and if so, when did he/she last view the diagram. You may also want to unshare a diagram from someone who no longer take part in the development or discussion of that diagram. All these can be achieved by managing shared diagrams in PostMania.

Viewing the diagrams shared with a viewer

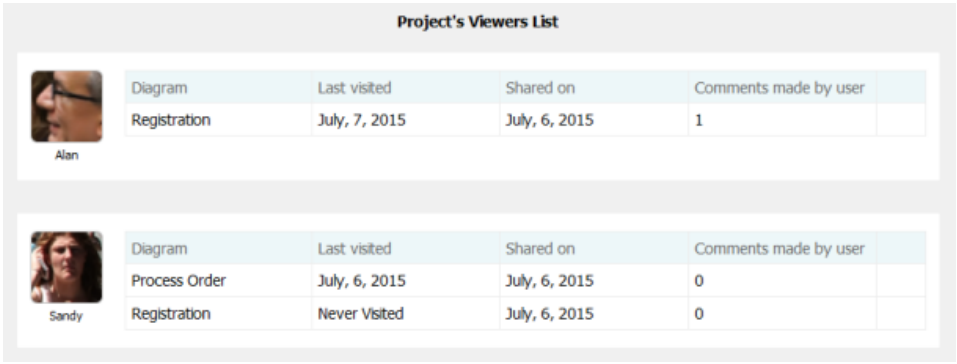
To know which diagram has been shared with a viewer, take the steps below.

1. Select **Team > PostMania** from the toolbar to open PostMania.
2. Click on the arrow button next to the word PostMania at the top left corner. of the screen. Select **Sharing** from the drop down menu.



Selecting Sharing

3. Now, you will see a list of viewers on the right hand side. Those are viewers (or members) who have been shared diagram(s). For each viewer, the diagram(s) that he/she can view are listed in the table.



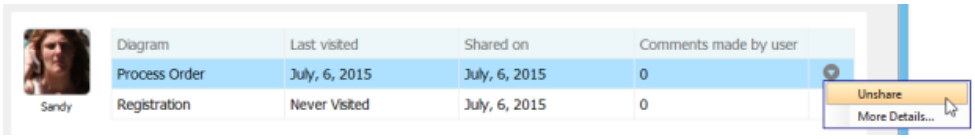
Project's Viewers list

For each diagram row listed in the table, the following information is presented:
Last visited: The date and time at which the viewer last viewed the diagram.
Shared on: The date and time at which the viewer has been shared the diagram.
Comments made by user: The number of post the viewer has made for this diagram, or any shape in this diagram.

Unshare a diagram from a viewer

If you don't want to let a viewer to view a shared diagram anymore, unshare that diagram from him/her. To unshare a diagram from a viewer:

1. Locate the viewer in the **Project's Viewers List**.
2. Select the row of diagram to unshare.
3. Click on the arrow button at the last column and select **Unshare** from the drop down menu.



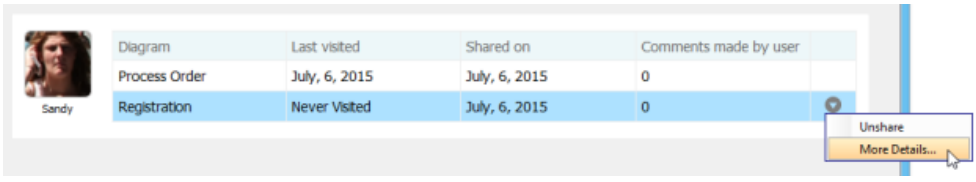
Unshare a diagram from a viewer

4. Choose Yes when are are prompted for confirmation. Note that the viewer will not be able to access the diagram anymore once you've chosen **Yes**. If you want to let him/her view the diagram again you need to share it again.

Managing the viewers of a diagram

You may also want to know the viewers of a particular diagram, or to unshare multiple viewers from a diagram, or to invite multiple viewers to view a diagram. All these can be achieved by taking the steps below.

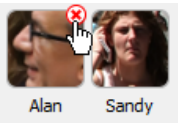
1. Find the row of diagram from any viewers listed in the **Project's Viewers List**.
2. Select that row.
3. Click on the arrow button at the last column and select **More Details...** from the drop down menu.



Unsharing diagram from individual viewer

To unshare the diagram from a particular viewer:

1. Move your mouse pointer over that viewer.
2. Click on the tiny cross appear at the top right corner of his/her profile picture.



Unshare diagram from viewer

3. Choose **Yes** when you are prompted for confirmation.

Unsharing diagram from all viewers

To stop sharing the diagram with anyone, click **Unshare All**. Choose **Yes** when you are prompted for confirmation.

Invite viewers

1. Click on **Invite others**.
2. Enter the name and/or the Email address of the people to share with. If a person to invite is an existing viewer/member, you just need to enter his name and pick him up from the drop down menu. If he is not currently a viewer, please enter his email address. PostMania will send invitation Emails all people specified, to invite them to join PostMania for viewing and commenting on the diagram to be shared.

Entered the name and email address of invitees

3. You can optionally include a message, which will be included in the email PostMania send out.
4. Click **Send Invitation Email**. PostMania will send the invitees invitation Emails in three minutes.

> **NOTE:** If the invitee is an existing viewer, and has opened the notification page/view of PostMania, either in a web browser or in Android apps, he/she will not receive the email notification.

In three minutes, invitees will receive an Email, with subject " **%NAME% has shared a diagram with you**" where **%NAME%** is the person who shared the diagram. He/she has to:

1. Click on **Accept Invitation** in the Email to open the **New Viewer Form** in web browser.
2. Enter the password and click **Accept Invitation** in the form. Then, he/she will be redirected to the diagram shared.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

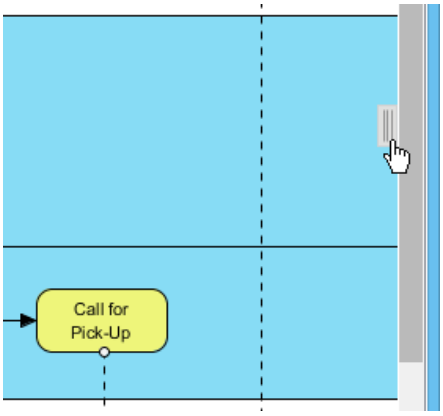
Managing Diagram Viewers

You may want to know if someone has viewed a diagram that you shared and if so, when did he/she last view the diagram. You may also want to unshare a diagram from someone who no longer take part in the development or discussion of that diagram. All these can be achieved by managing shared diagrams in PostMania.

Viewing the viewers of a diagram

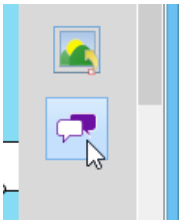
To know the viewers of a diagram, take the steps below.

1. Open the diagram you want to check.
2. Open the action bar by clicking on the tiny button on the right hand side of the diagram, near the scrollbar, if exist.



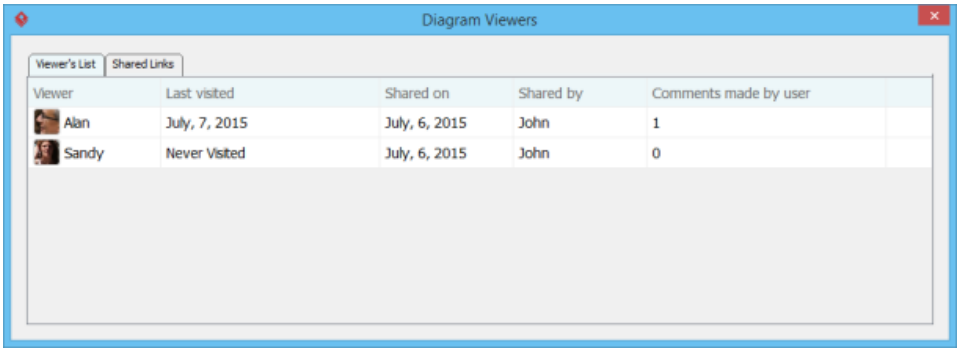
Open the action bar

3. Open the **PostMania Topic Pane** button by clicking on its button in the action bar.



Opening PostMania Topic Pane

4. At the top of the **PostMania Topic Pane**, click **Share** and then select **Viewer List** from the drop down menu. Now, you will see a list of viewers who can view this diagram.



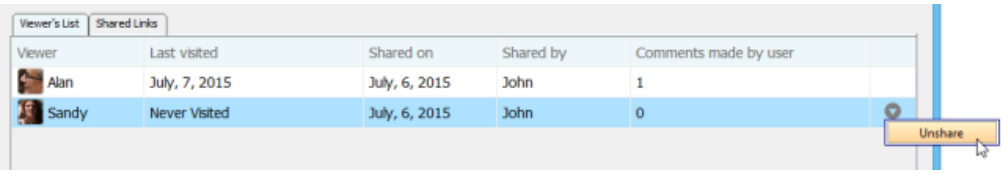
Viewer list

For each viewer row listed in the table, the following information is presented:
Last visited: The date and time at which the viewer last viewed the diagram.
Shared on: The date and time at which the viewer has been shared the diagram.
Shared by: The person who shared the diagram.
Comments made by user: The number of post the viewer has made for this diagram, or any shape in this diagram.

Unshare diagram from a viewer

If you don't want to let a viewer to view a shared diagram anymore, unshare that diagram from him/her. To unshare a diagram from a viewer:

1. Locate the viewer in the **Viewer List**.
2. Select the row of the viewer.
3. Click on the arrow button at the last column and select **Unshare** from the drop down menu.



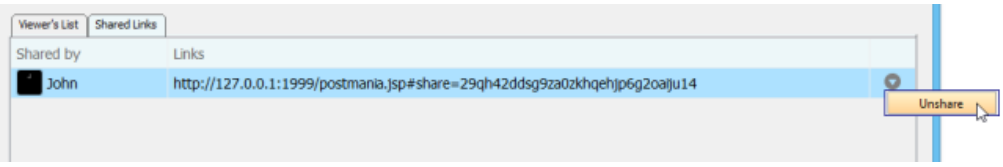
Unshare a diagram from a viewer

4. Choose Yes when are are prompted for confirmation. Note that the viewer will not be able to access the diagram anymore once you've chosen **Yes**. If you want to let him/her view the diagram again you need to share it again.

Disabling a shared link

For security reasons you may want to disable a shared link once it has been visited by the authorized person. To disable a shared link:

1. Open the **Shared Links** tab in the **Diagram Viewers** window.
2. Select the link to disable.
3. Click on the arrow button at the last column and select **Unshare** from the drop down menu.. Note that by disabling the URL, new attempts to browse the diagram with that URL will be denined. Yet, viewers who have browserd the diagram before will remain accessible to the diagram.



Disable diagram link

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

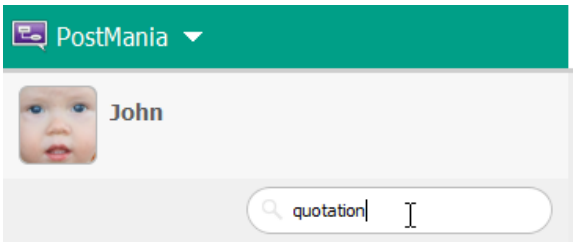
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to us

Searching a Post

When you want to read the conversation made earlier but have no idea in which diagram and shape the conversation was made, you can search it out with a keyword.

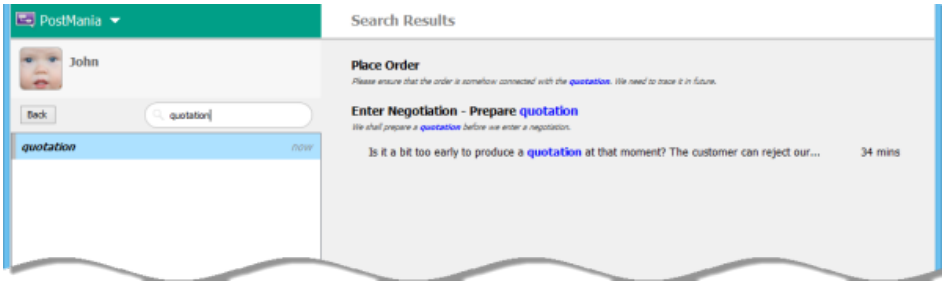
Searching a post

1. At the top left of the PostMania, enter the keyword in the **Search**... field.



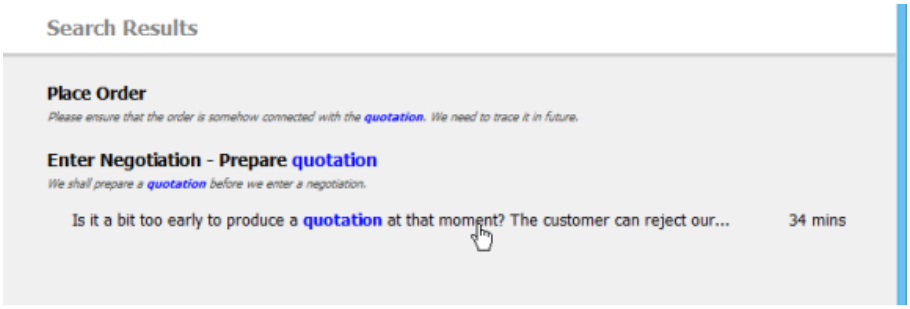
Searching for the word 'quotation'

2. Press the **Enter** key to perform a search. Search results are listed in the page.



Search results

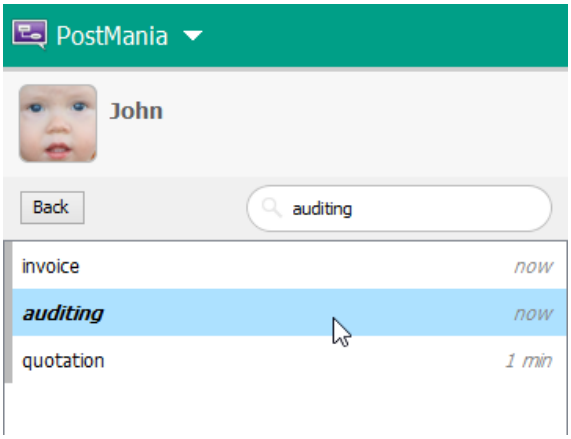
3. You can click on an entry to open it.



Opening a post from search resutls

Accessing search history

The history of your previous searches are listed on the left hand side of the search page. You can search a keyword again by clicking on it.



Search the word 'invoice' again

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to us

Instant Reverse

Instant reverse is a process to produce UML class model from a given input of source code. In this chapter, you will learn how to make use of Instant Reverse to reverse engineer UML class model from source code in specific language.

Instant reverse Java sources and classes

Reverse engineer class model from Java (.java, .class, .jar, zip)

Instant reverse C++ header files

Reverse engineer class model from C++ header files.

Instant reverse .NET dll and exe files

Reverse engineer class model from .NET dll/exe.

Instant reverse CORBA IDL Source files

Reverse engineer class model from CORBA IDL.

Instant reverse Ada 9X Source files

Reverse engineer class model from Ada 9x source files.

Instant reverse XML

Reverse engineer class model from XML files.

Instant reverse XML Schema

Reverse engineer class model from XML schema files (.xsd).

Instant reverse hibernate mapping files

Reverse engineer class model from Hibernate mapping files.

Instant reverse PHP 5.0 Source files

Reverse engineer class model from PHP 5.0.

Instant reverse Python

Reverse engineer class model from Python.

Instant reverse Objective-C

Reverse engineer class model from Objective-C.

Instant reverse Java sources to sequence diagram

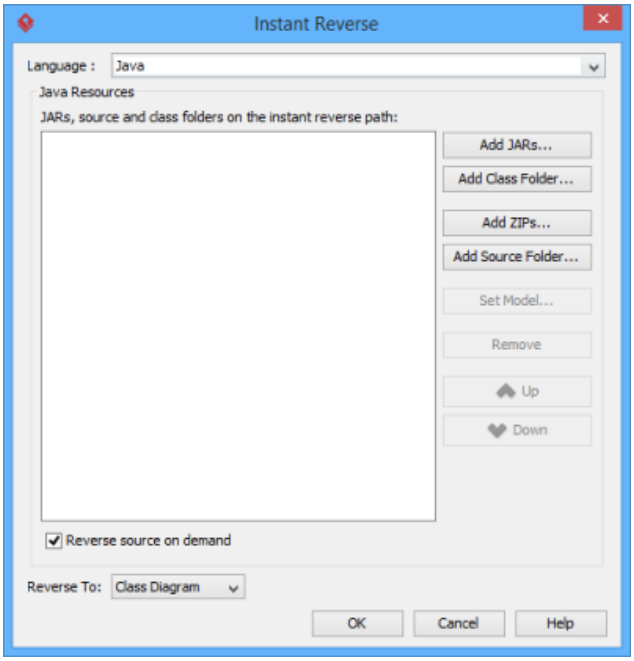
Reverse engineer sequence diagram from Java source files.

Instant reverse Java sources and classes

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes, and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of Java.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Java** as the **Language**.
3. Add the file or folder path of source by clicking on the appropriate **Add** button at the right hand side of the window. There are four kinds of supported sources: Jar file, class folder, a zip of source or a folder of source files.



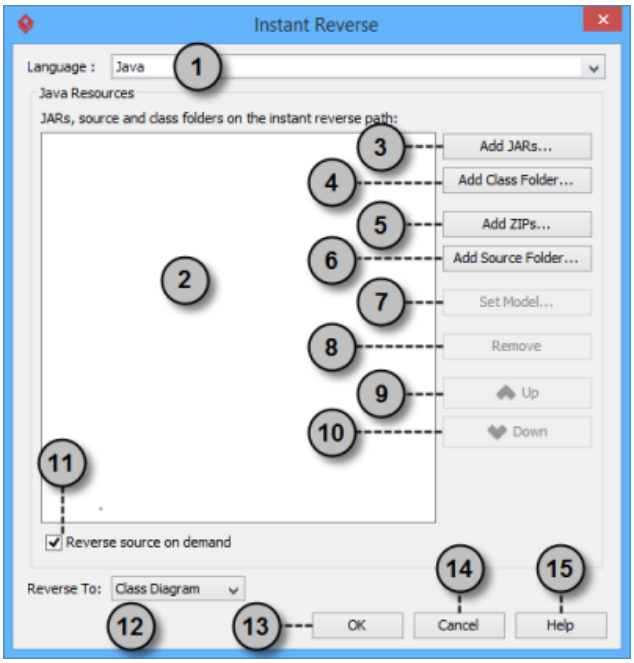
*The **Instant Reverse** window*

> **NOTE:** You can reverse multiple source paths by adding them one after the other.
> You can add different kinds of source. For example, you can reverse a jar as well as a folder of source file at the same time.

4. By default an on-demand reverse engineering will be carried out, which means to form indexes to the added path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below. If you want to carry out actual reverse engineering, uncheck **Reverse source on demand**.
5. Click **OK** to start reversing.
6. Upon finishing, the class repository will be popped up, listing the reversed classes (or indexes of classes if you are running an on-demand reverse engineering).

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



Overview of instant reverse window

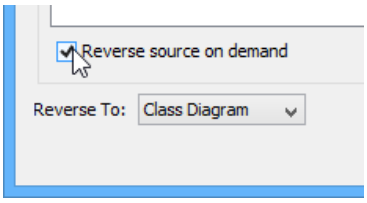
No.	Name	Description
1	Language	The programming language for reversing.
2	Source paths	A list of source paths to be reversed.
3	Add Jars	Add a path of Jar file for reversing.
4	Add class folder	Add a path of Java class folder for reversing.
5	Add zips	Add a path of a zipped source folder for reversing.
6	Add source folder	Add a path of Java source folder for reversing.
7	Set model	Set the model for placing the reversed UML classes into.
8	Remove	Remove selected source path(s) from the list of source paths.
9	Up	Move selected source path(s) upward in the path list. It just affects the order of reversing, and have no impact on the reversed UML classes.
10	Down	Move selected source path(s) downward in the path list. It just affects the order of reversing, and have no impact on the reversed UML classes.
11	Reverse source on demand	By checking, this means to form indexes to the source path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below.
12	Reverse to diagram	Analyze the sources and form class diagram/package diagram when reverse.
13	OK	Click to start reversing.
14	Cancel	Click to cancel reversing and exit.
15	Help	Click to read Help contents for instant reverse.

Description of instant reverse window

On-demand reverse engineering

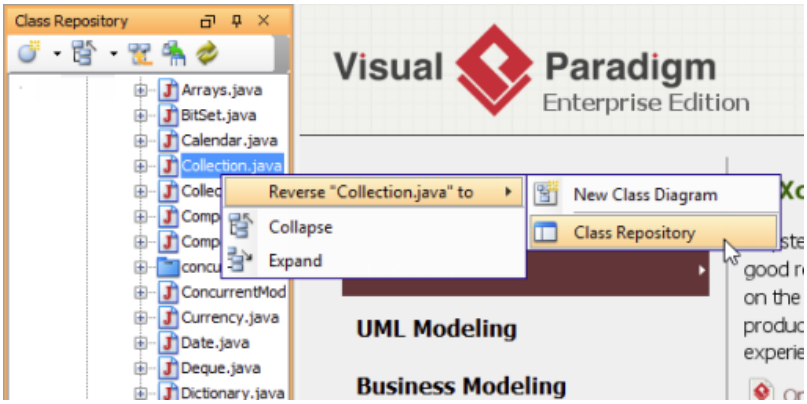
Consider if you have a zip file that contains million of Java source file, like the file src.zip of Java Development Kit (JDK), and now you want to make the class java.util.Collection appear as UML class so that you can extend it when developing your own collection framework. Now, if you try to reverse the zip file it will take you a long time to complete the reverse due to the amount of classes (and relationships) is just too many. With on-demand reverse engineering, you will reverse the sources as indexes, and obtain an index tree in class repository. No actual UML classes will be reversed until you trigger the reverse manually. This reduces the processing time significantly.

To perform on-demand reverse engineering, make sure the option Reverse source on demand is checked in the **Instant Reverse** window.



*The option **Reverse source on demand** that appear in reverse window*

When finished instant reverse, you can lookup the index tree in class repository. Then, right click on the class you want to reverse and select **Reverse Resources to** where **Resources** are the classes you have selected, and select either **New Class Diagram** or **Class Repository** from popup menu. Both options will result in reversing the selection to UML classes, while the option **New Class Diagram** will create a class diagram and place the classes in it.

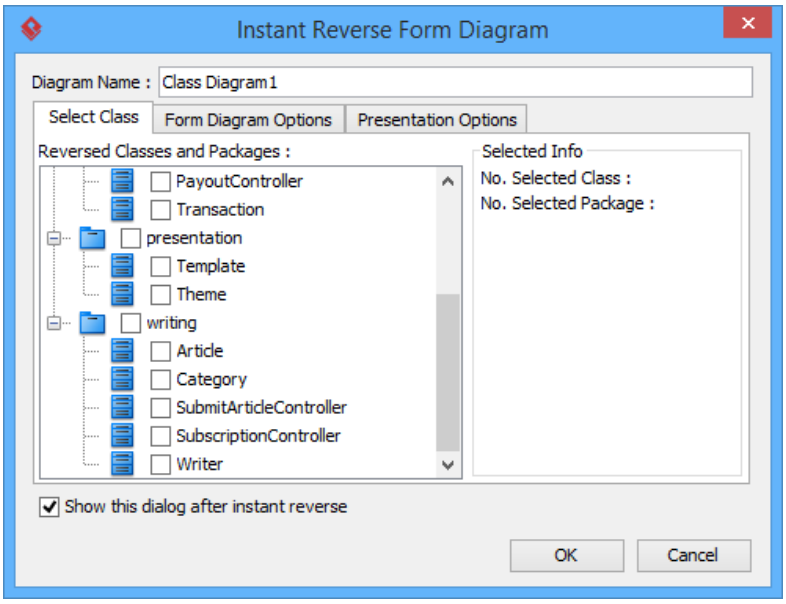


Reversing a java source file from index tree

NOTE: On-demand reverse engineering is only available for Java

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them, and confirm, a diagram will then be form.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

NOTE: The **Form Diagram** window will only pop up when you were reversing source with on-demand turned off. If you have performed an on-demand reverse engineering, you need to form diagram manually. For details, read the previous section.

Below is a description of this window, base on the tabs.

Select Class

The classes listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

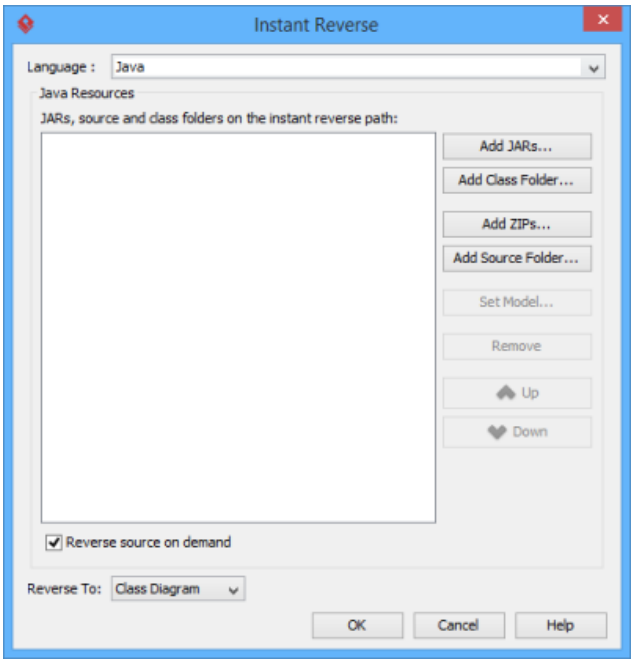
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Java** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.



The instant reverse window

4. Select **Package Diagram** for **Reverse To:**.
5. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

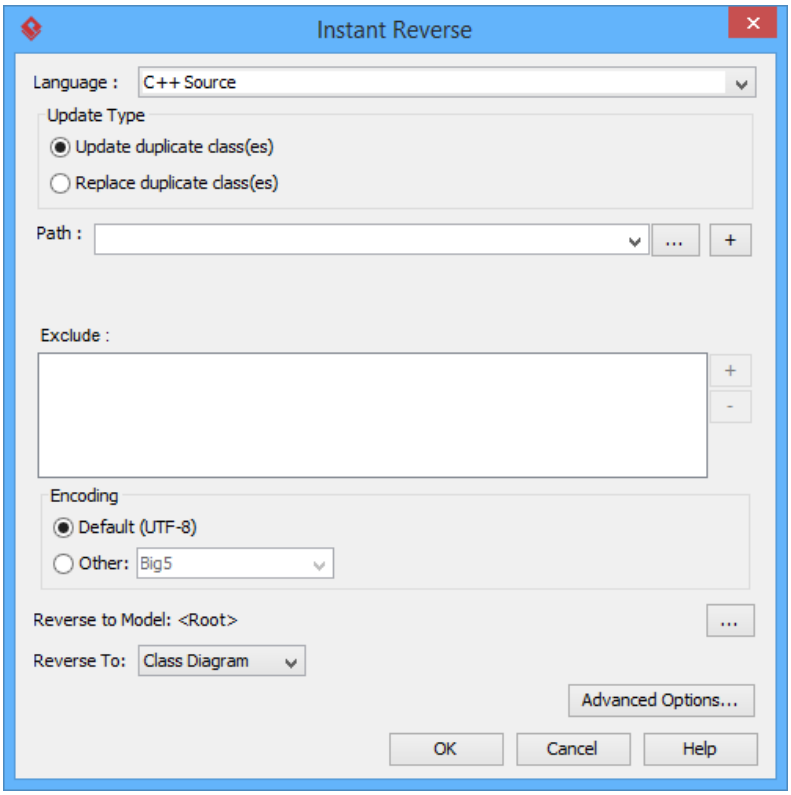
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse C++ source

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of C++ files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **C++ Source** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

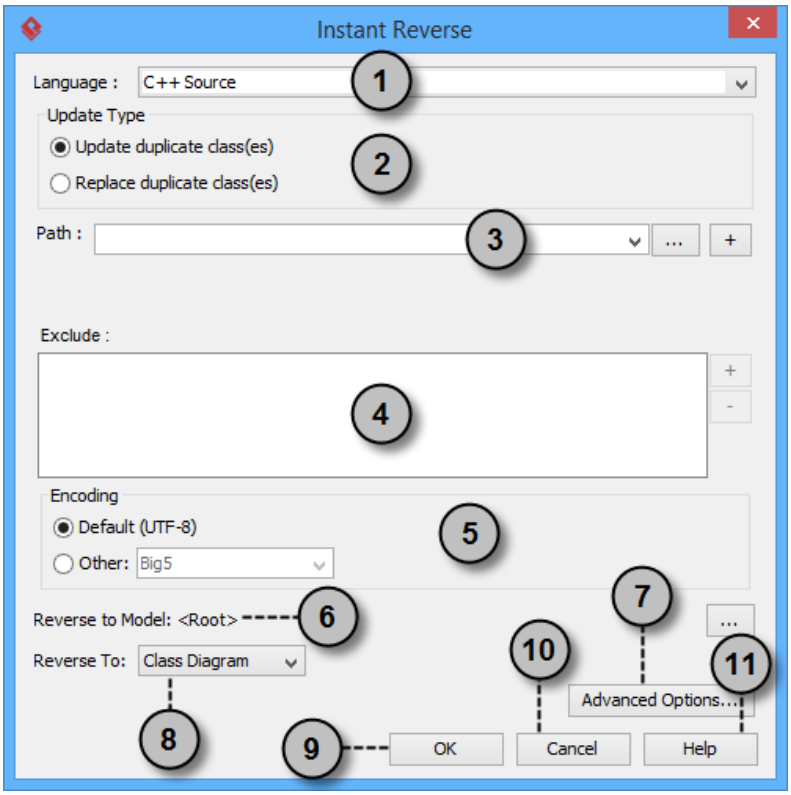


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository** and possibly form diagram later on manually.

Overview of Instant Reverse



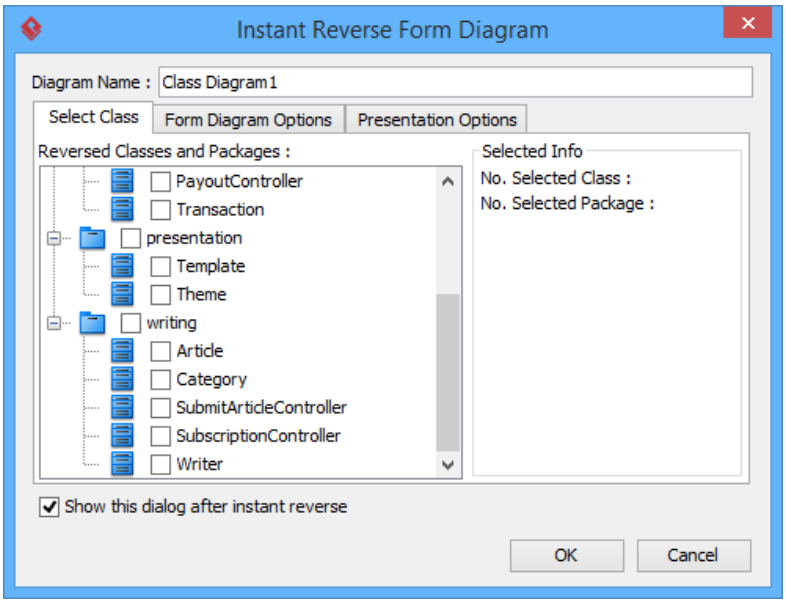
The instant reverse window

No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Exclude	The file(s) to skip during the reverse process.
5	Encoding	The encoding of source files.
6	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
7	Advanced Options	More configurable options related to C++ instant reverse.
8	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
9	OK	Click to start reversing.
10	Cancel	Click to close instant reverse.
11	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

Option	Description

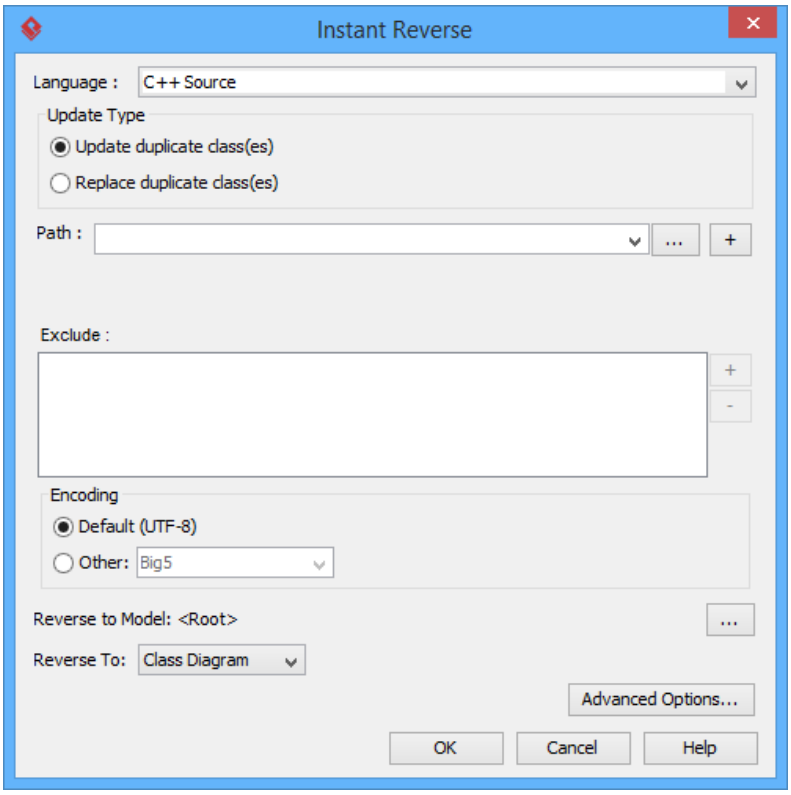
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **C++ Source** as the **Language**.
3. Specify the path of the source file, or the folder that contains those files.



The instant reverse window

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

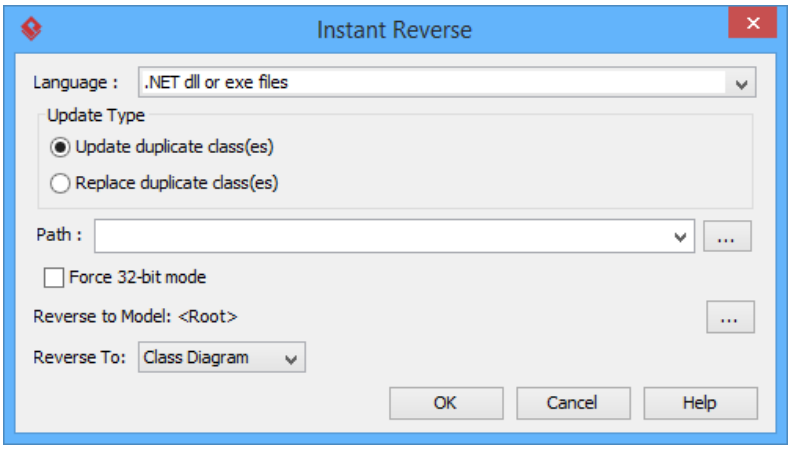
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse .NET dll and exe files

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of .NET dll and exe files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **.NET dll or exe files...** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

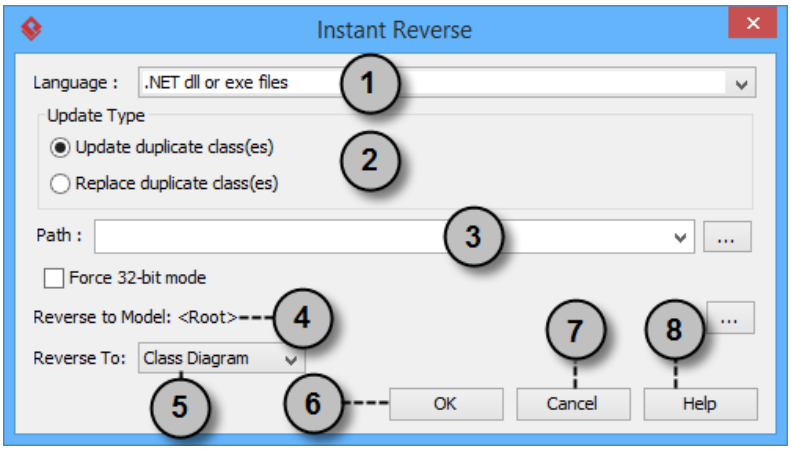


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



The instant reverse window

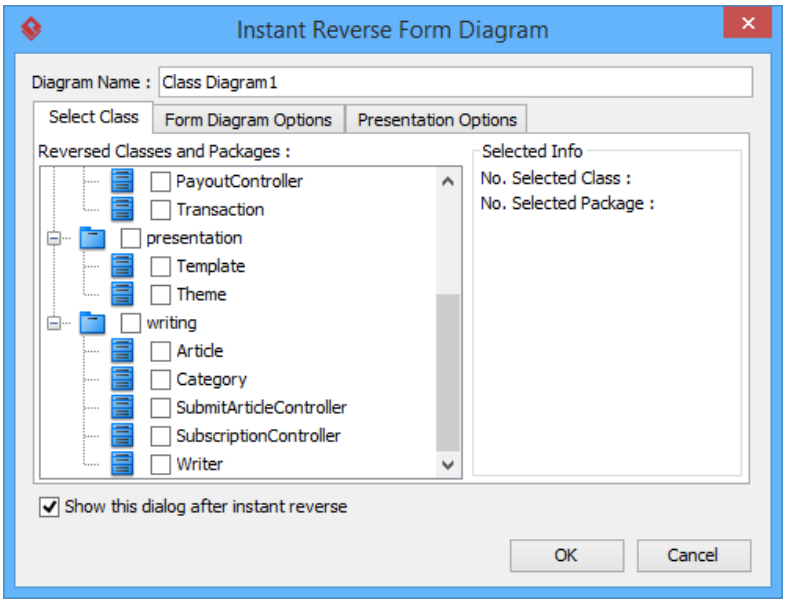
No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types:

		Update duplicate class(es) - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
6	OK	Click to start reversing.
7	Cancel	Click to close instant reverse.
8	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.

Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

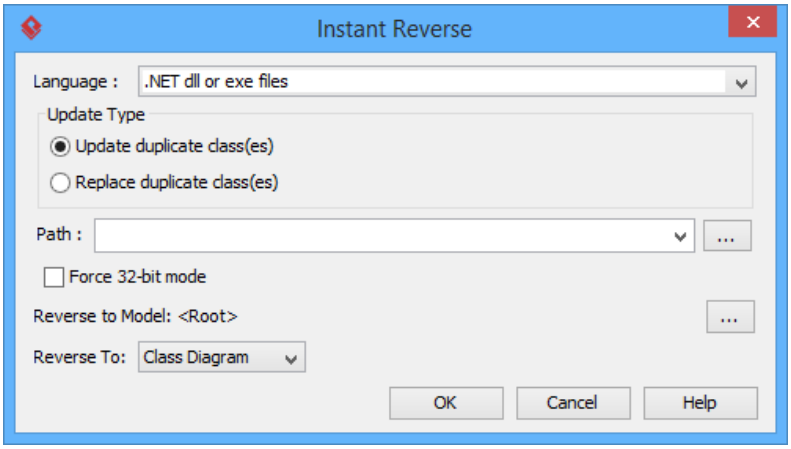
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **.NET dll or exe files...** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.



The instant reverse window

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.

3. Click **OK** to confirm.

6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

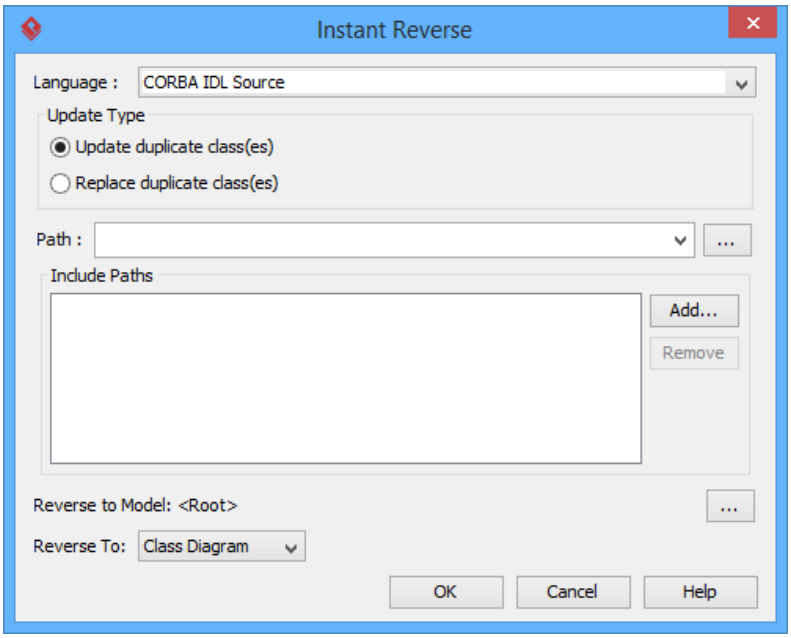
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse CORBA IDL source files

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of CORBA IDL source files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **CORBA IDL source files** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

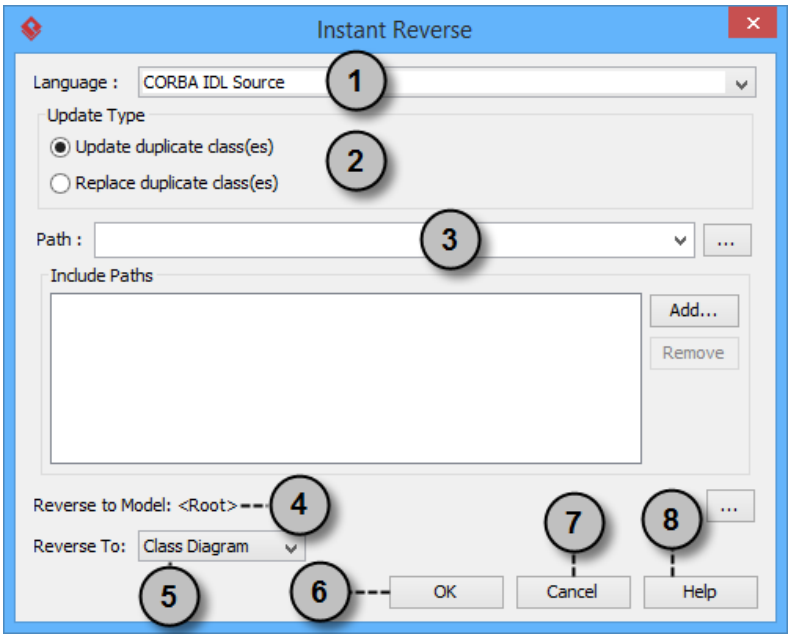


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository** and possibly form diagram later on manually.

Overview of Instant Reverse



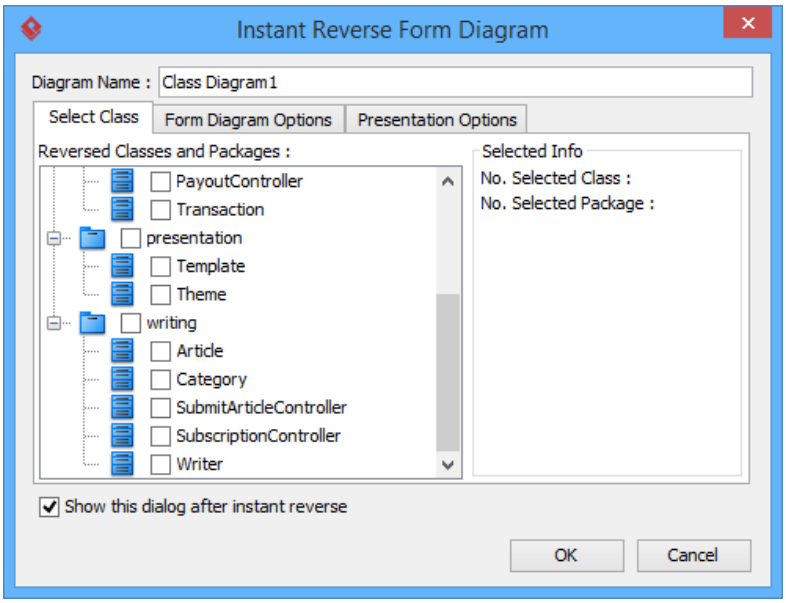
The instant reverse window

No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
6	OK	Click to start reversing.
7	Cancel	Click to close instant reverse.
8	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

Option	Description

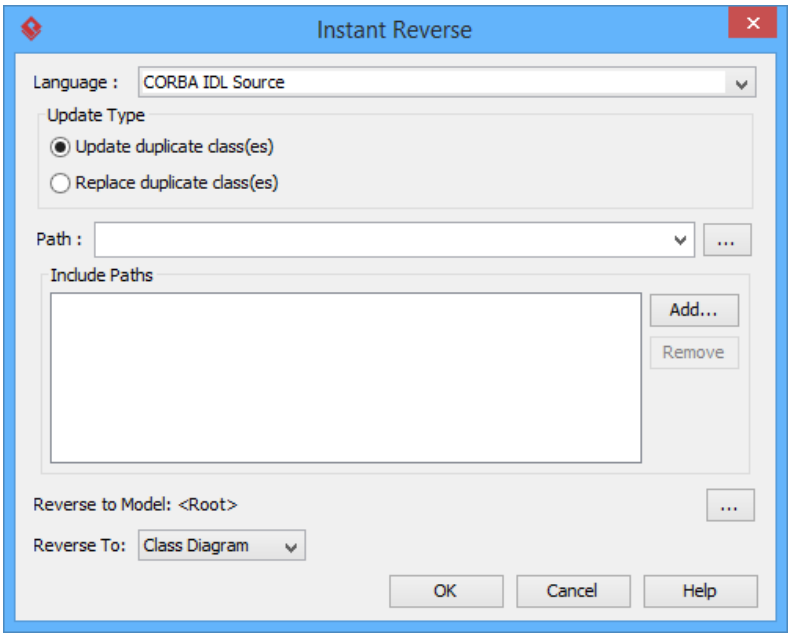
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **CORBA IDL source files** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.



*The **instant reverse** window*

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

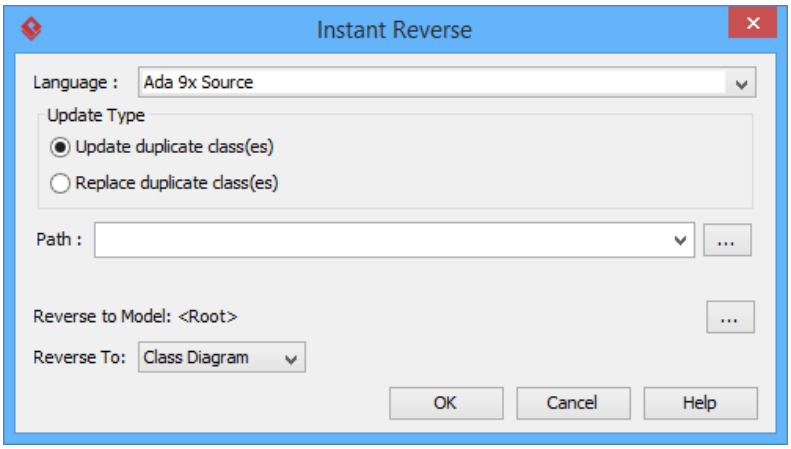
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse Ada 9X source files

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extendinDetermining the generic one. In this chapter, we will go through the instant reverse of Ada 9x source files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Ada 9x source files** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

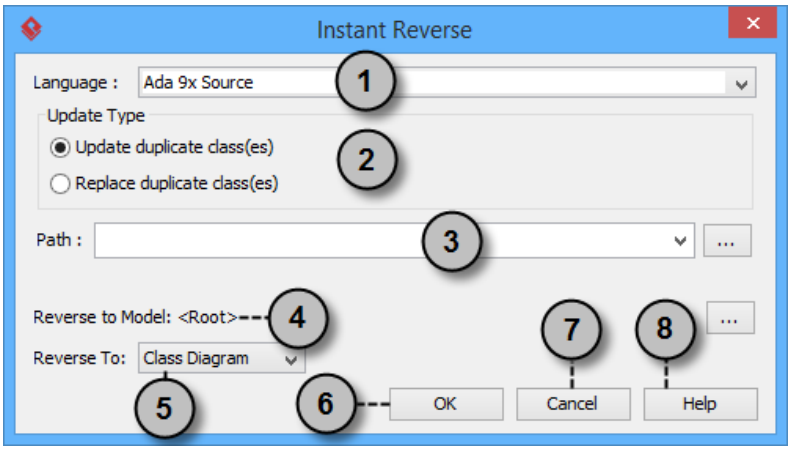


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



The instant reverse window

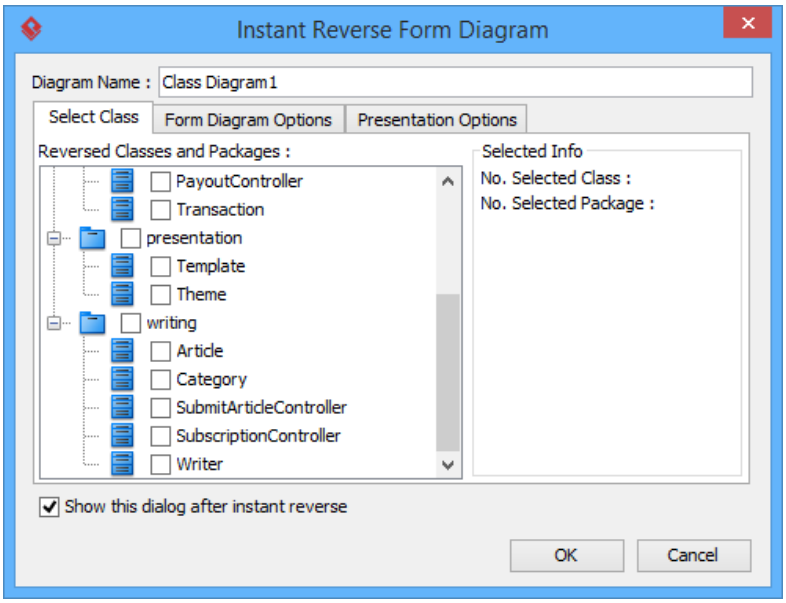
No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types:

		Update duplicate class(es) - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
6	OK	Click to start reversing.
7	Cancel	Click to close instant reverse.
8	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.

Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

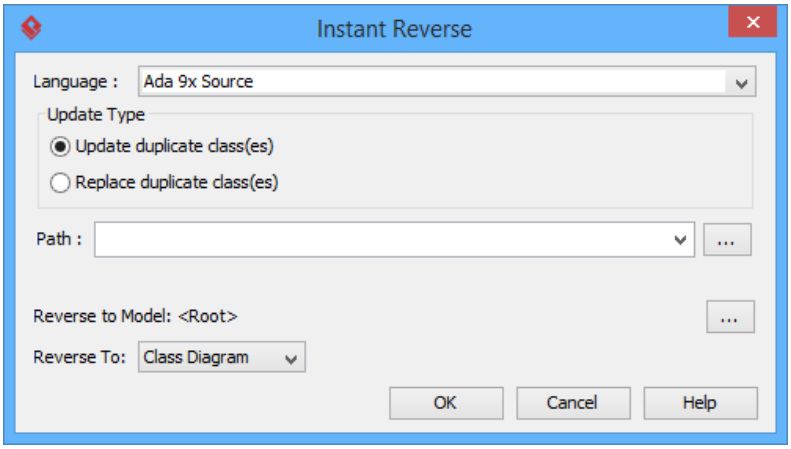
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Ada 9x source files** as the **Language**.
3. Specify the path of the source file, or the folder that contains those files.



*The **instant reverse** window*

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.

3. Click **OK** to confirm.

6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

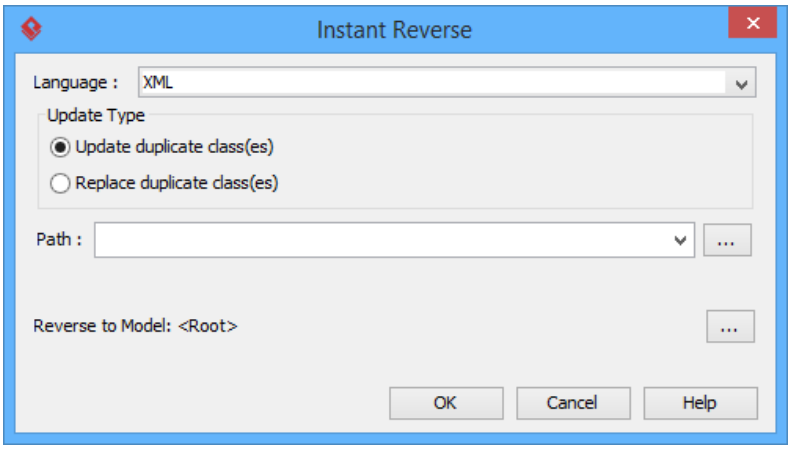
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse XML

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of XML files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **XML** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

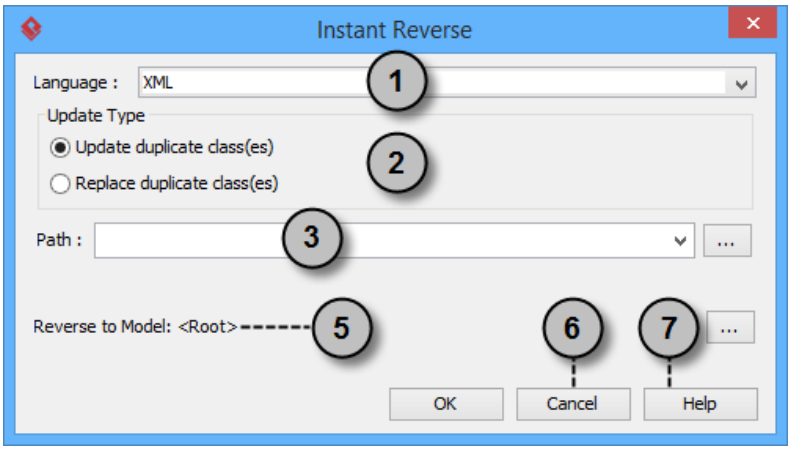


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository** and possibly form diagram later on manually.

Overview of Instant Reverse



The instant reverse window

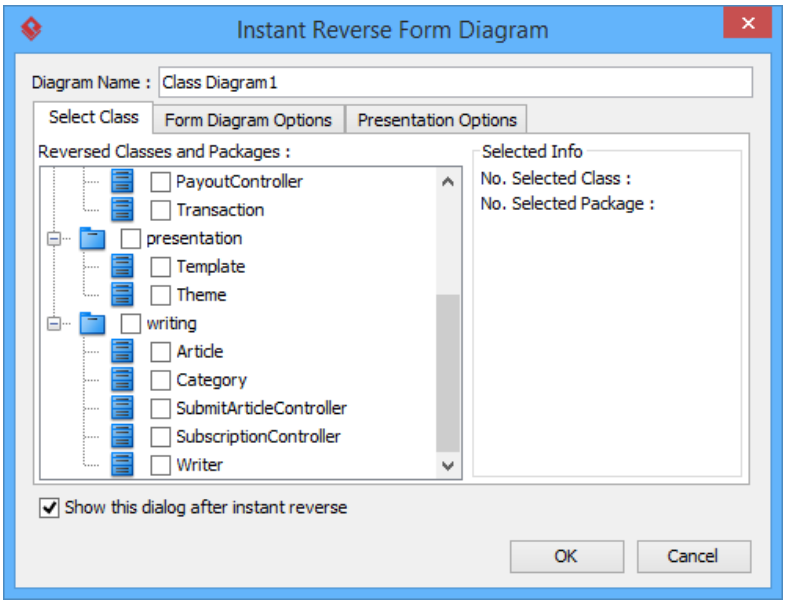
No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types:

		Update duplicate class(es) - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	OK	Click to start reversing.
6	Cancel	Click to close instant reverse.
7	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.

Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

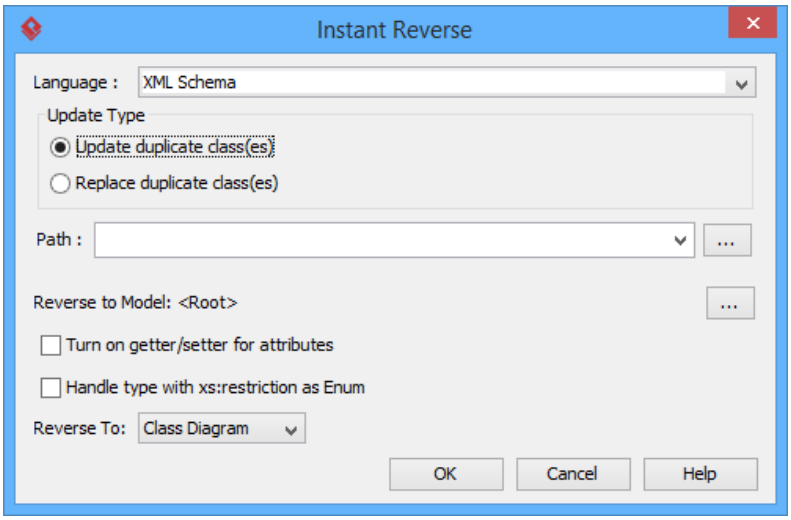
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse XML schema

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of XML Schema.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **XML Schema**as the **Language.**
3. Specify the path of the source file or the folder that contains those files.

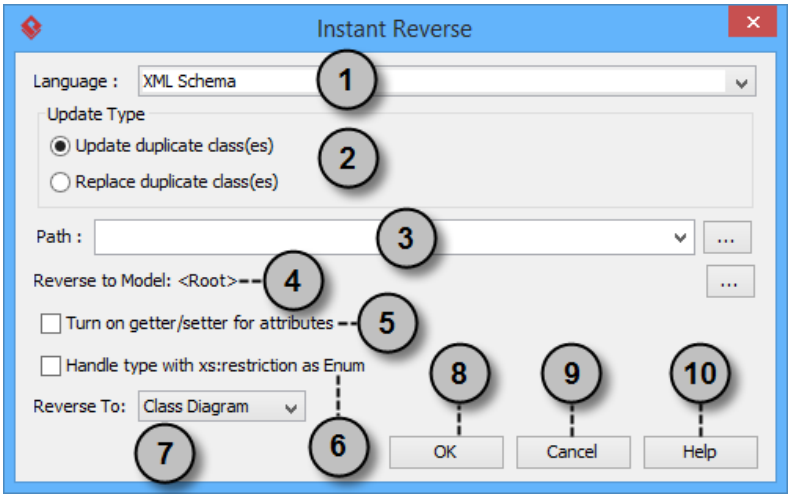


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



The instant reverse window

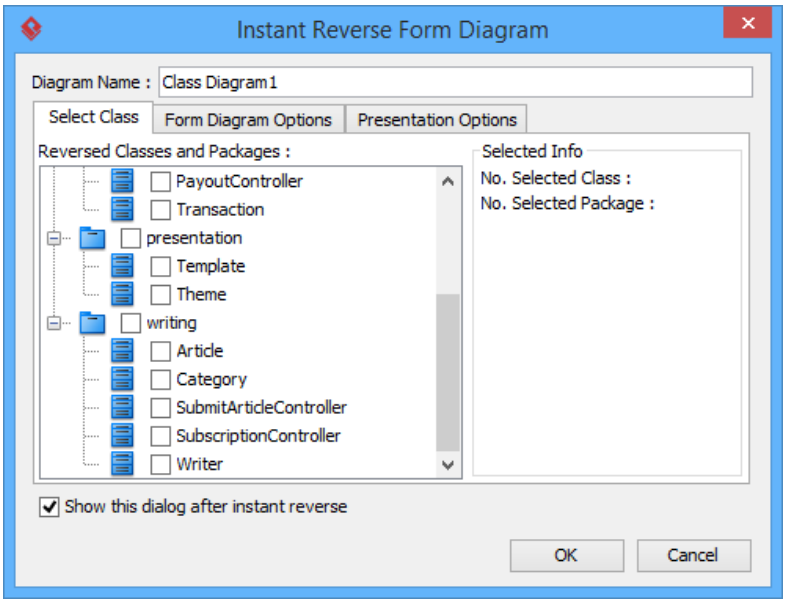
No.	Name	Description

1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	Turn on getter/setter for attributes	Set all attributes' getter and setter options to be true.
6	Handle type with xs:restriction as Enum	Whether or not to create Enum class for xs:restriction types.
7	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
8	OK	Click to start reversing.
9	Cancel	Click to close instant reverse.
10	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.

Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

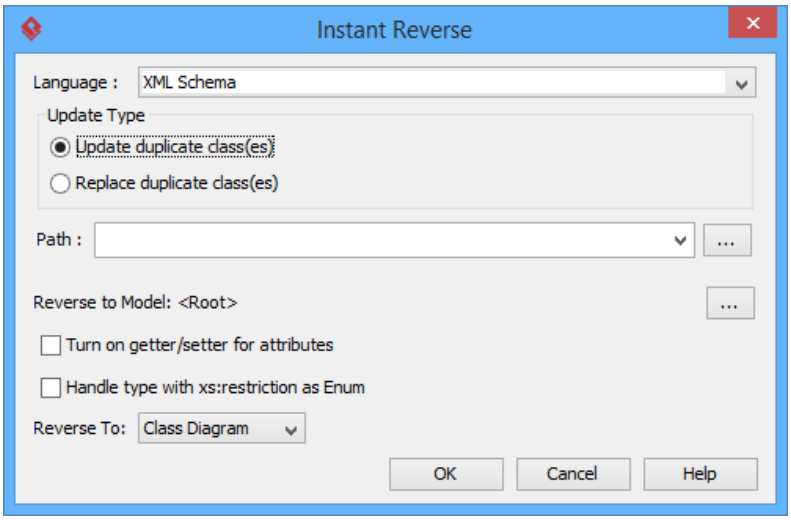
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **XML Schema** as the **Language.**
3. Specify the path of the source file, or the folder that contains those files.



*The **instant reverse** window*

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
    1. Click on the **...** button at the end of the **Reverse to Model** row.
    2. In the **Select Parent Model** window, either select an existing model or create one by clicking **New Model**.
    3. Click **OK** to confirm.
6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

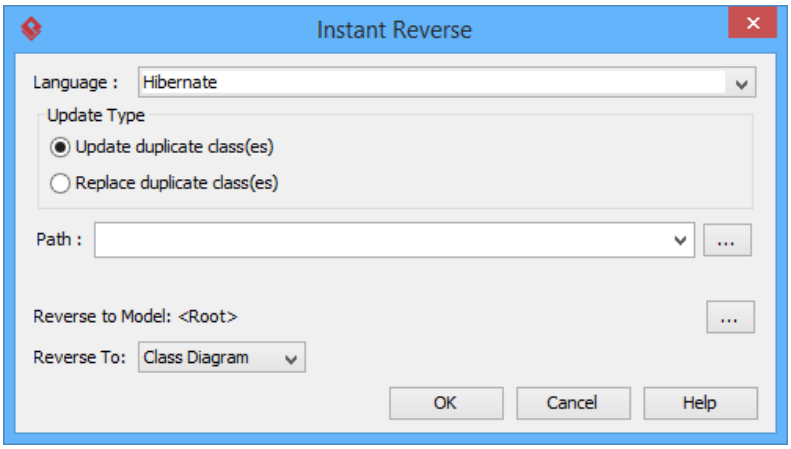
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse hibernate mapping files

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of Hibernate mapping files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Hibernate** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

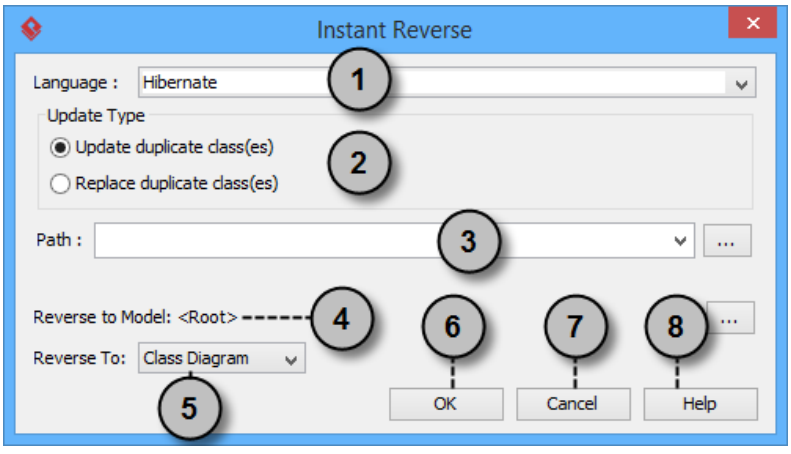


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository** and possibly form diagram later on manually.

Overview of Instant Reverse



The instant reverse window

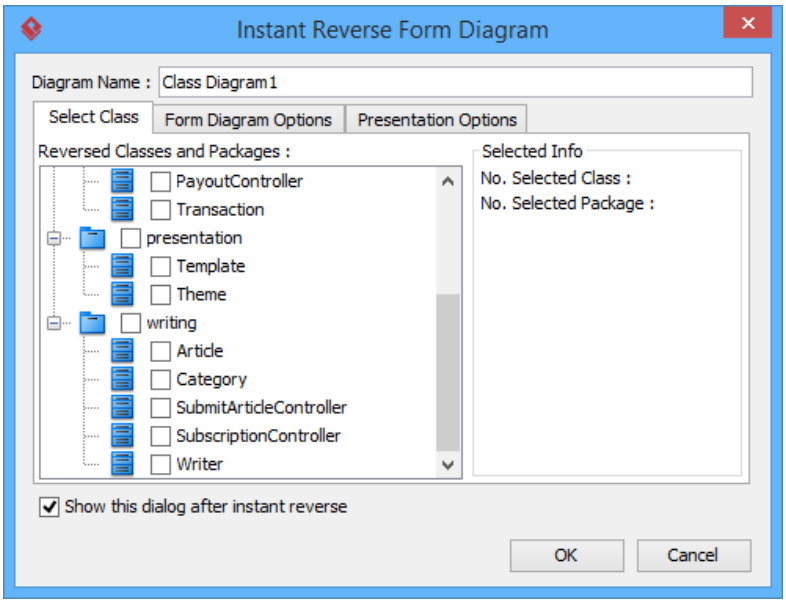
No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types:

		Update duplicate class(es) - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
5	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
6	OK	Click to start reversing.
7	Cancel	Click to close instant reverse.
8	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.

Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

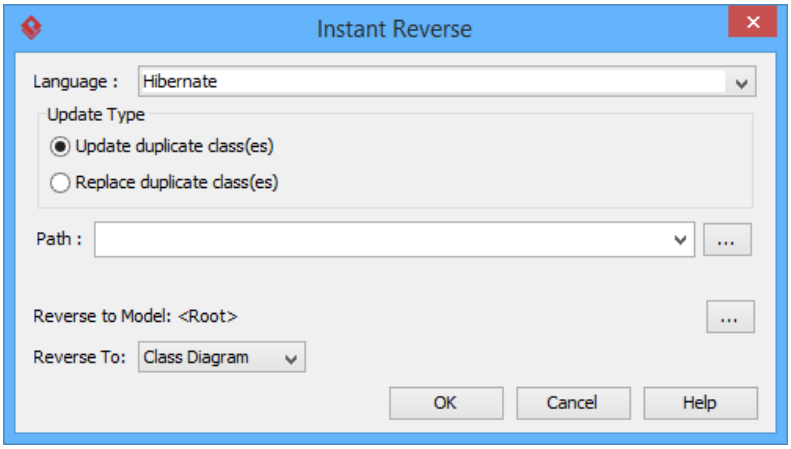
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Hibernate** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.



The instant reverse window

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.

3. Click **OK** to confirm.

6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

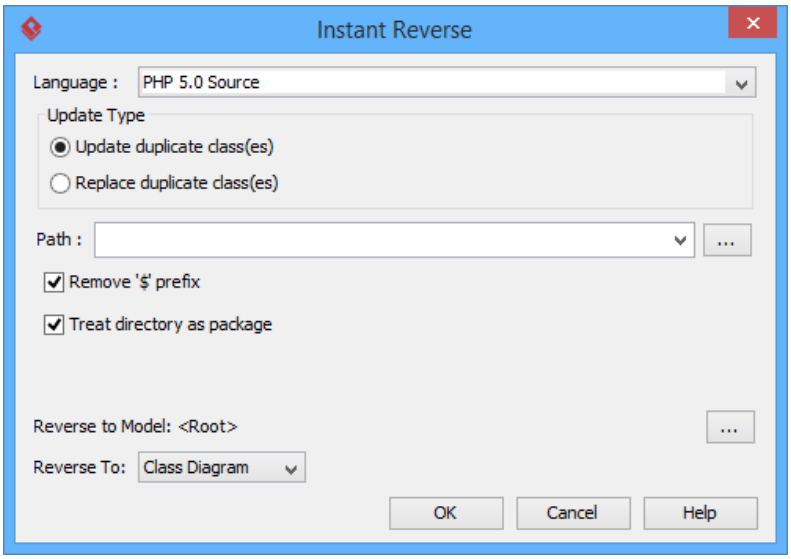
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse PHP 5.0 source files

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the Determingeneric one. In this chapter, we will go through the instant reverse of PHP 5.0 source files.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **PHP 5.0 source files** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

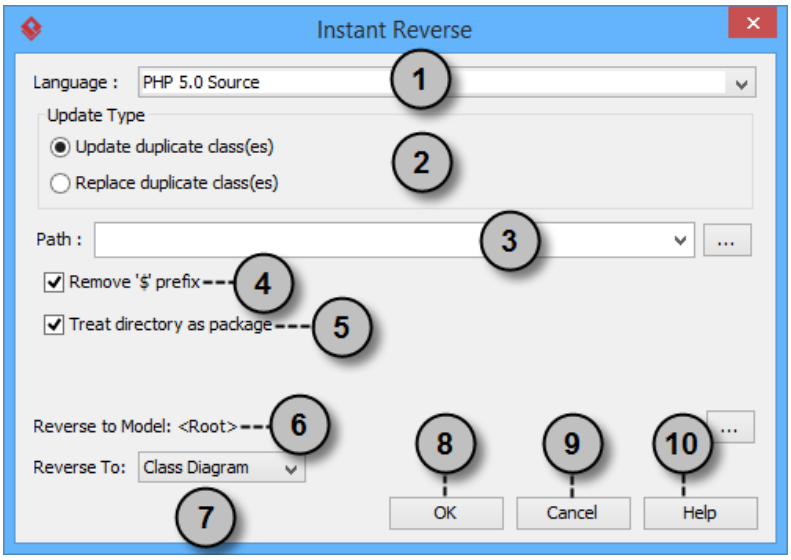


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



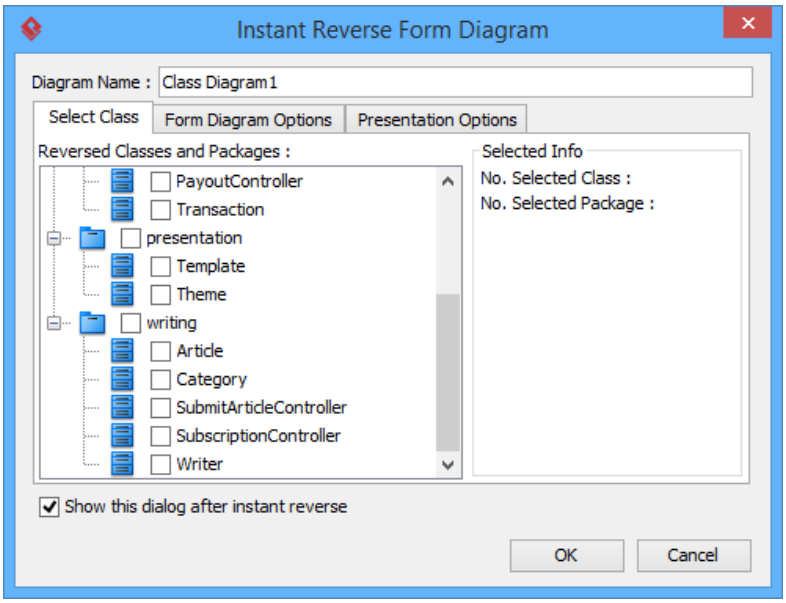
The instant reverse window

No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Remove '$' prefix	Ignore the dollar sign prefix for attributes.
5	Treat directory as package	Convert folders to UML packages
6	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
7	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
8	OK	Click to start reversing.
9	Cancel	Click to close instant reverse.
10	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes that are listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

Option	Description

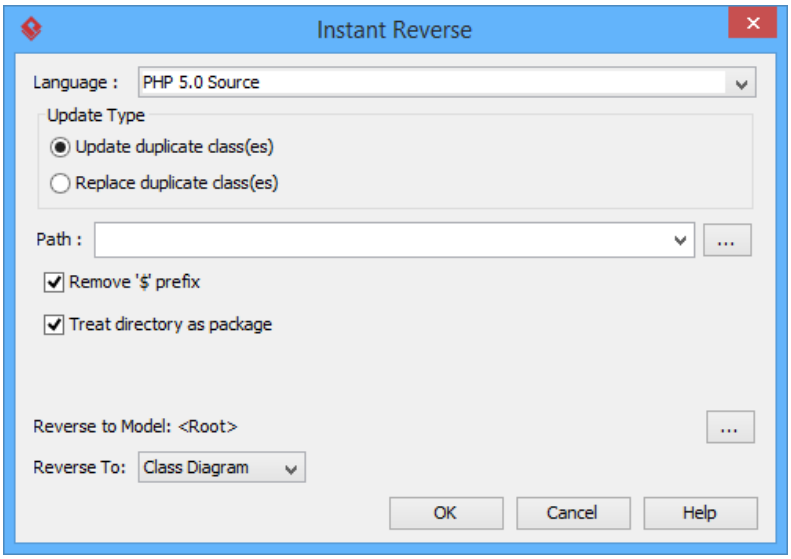
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **PHP 5.0 source files** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.



*The **instant reverse** window*

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

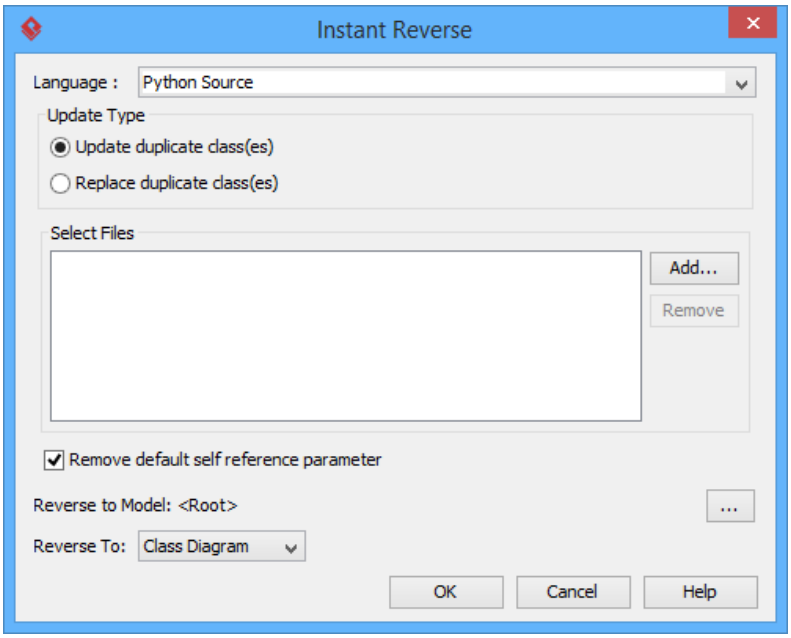
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse Python

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of Python.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Python Sourse** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

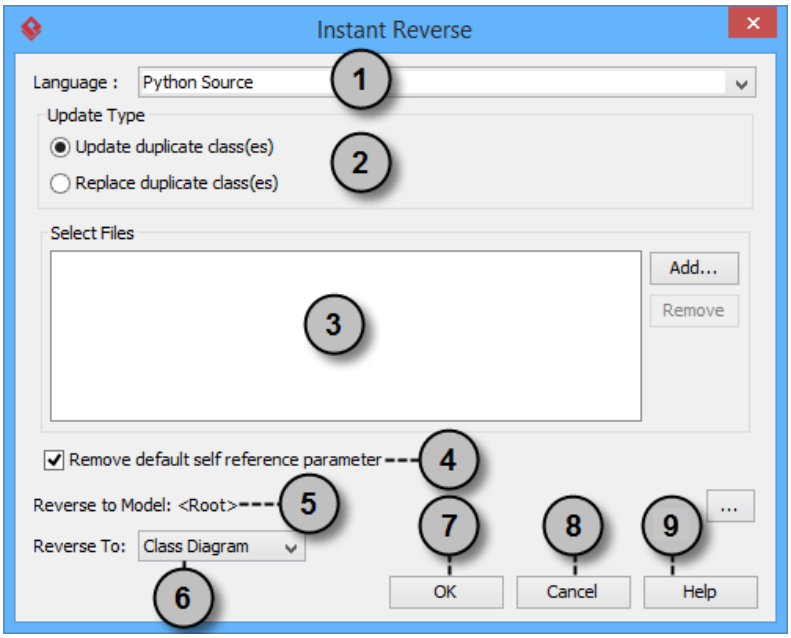


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



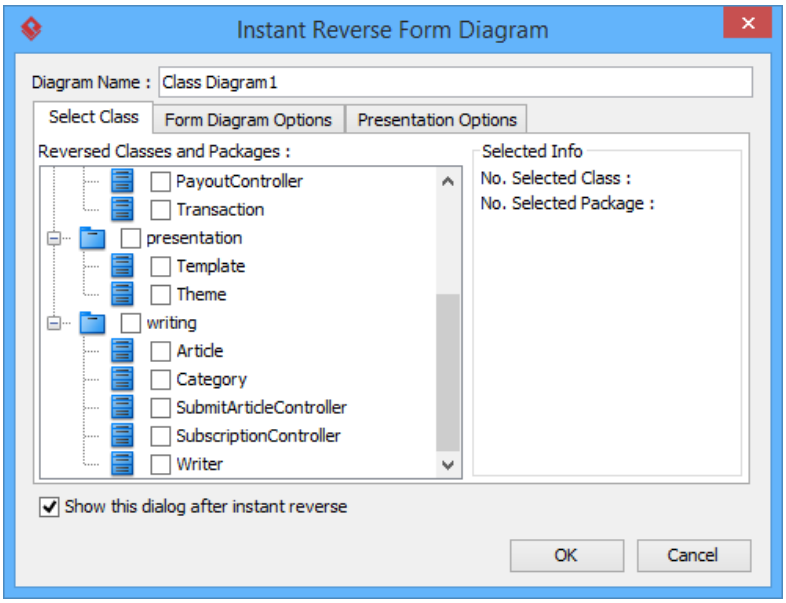
The instant reverse window

No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Select Files	The source files to reverse.
4	Remove default self reference parameter	Whether or not to remove default self reference parameter.
5	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
6	Reverse to diagram	Whether or not to reverse engineering class diagram/package diagram from source files. UML packages and the relationships in between will be produced.
7	OK	Click to start reversing.
8	Cancel	Click to close instant reverse.
9	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. **NOTE:** The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

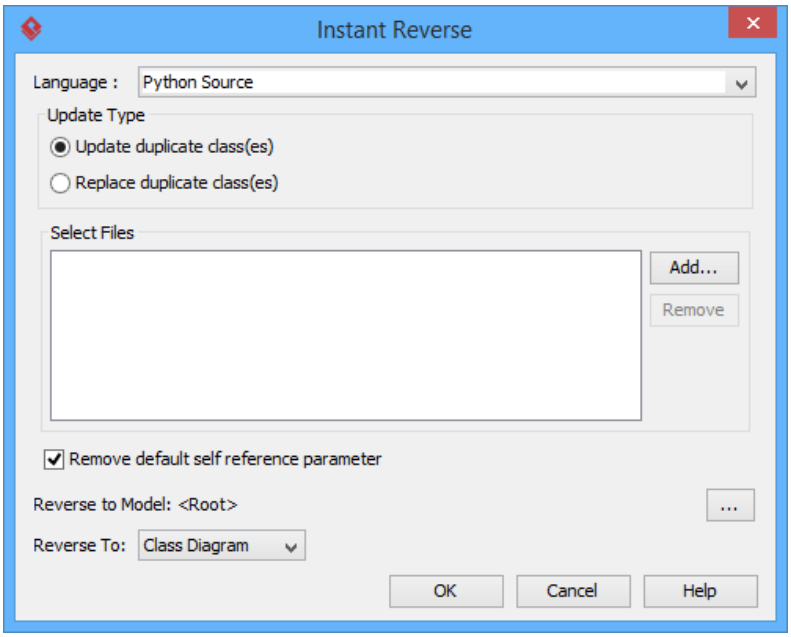
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Reverse engineer package diagram from source files

By reverse engineering package diagram from source files, UML packages and the relationships in between will be produced.

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Python Sourse** as the **Language**.
3. Specify the path of the source file, or the folder that contains those files.



*The **instant reverse** window*

4. Select **Package Diagram** for **Reverse To:**.
5. You can place reversed packages to specific model. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
6. Click **OK** to start reversing.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

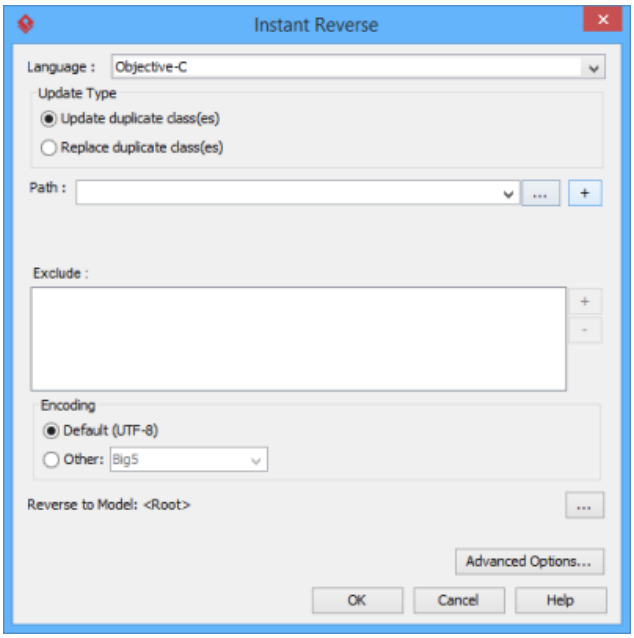
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse Objective-C

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. By bringing code content into visual UML model, this helps programmers or software engineers to review an implementation, identify potential bugs or deficiency and look for possible improvements. Apart from this, developers may reverse a code library as UML classes and construct model with them, like to reverse a generic collection framework and develop your own framework by extending the generic one. In this chapter, we will go through the instant reverse of Objective-C.

Reverse engineering UML classes from source files

1. Select **Tools > Code > Instant Reverse...** from the toolbar.
2. In the **Instant Reverse** window, select **Objective-C** as the **Language**.
3. Specify the path of the source file or the folder that contains those files.

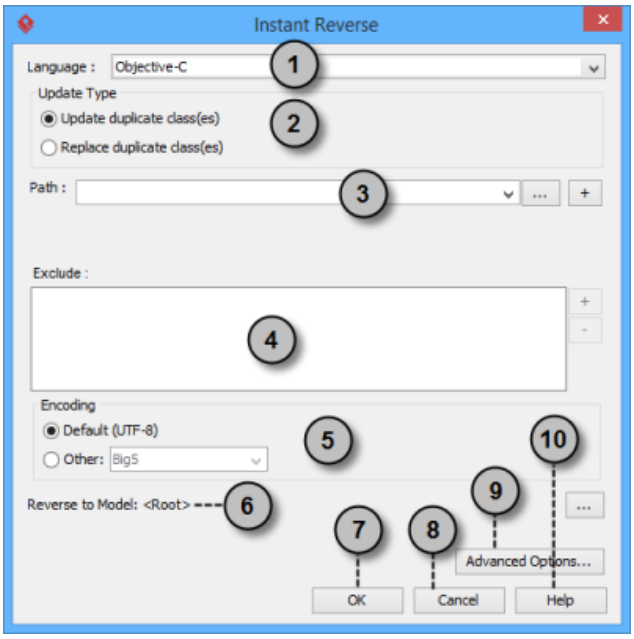


*The **Instant Reverse** window*

4. You can place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth. To do this:
   1. Click on the **...** button at the end of the **Reverse to Model** row.
   2. In the **Select Parent Model** window, either select an existing model, or create one by clicking **New Model**.
   3. Click **OK** to confirm.
5. Click **OK** to start reversing.
6. Upon finishing, you will see the **Instant Reverse Form Diagram** window appear. If you want to form a class diagram with the reversed classes, select the classes to form diagram, configure the options and click **OK** to proceed. Read the next section for detail. If you do not want to form diagram now, click **Cancel** to exit.

> **NOTE:** By cancelling from forming diagram, it just means you do not want to form diagram with the reversed classes for the time being. You can still look for the classes in **Model Explorer** or **Class Repository**, and possibly form diagram later on manually.

Overview of Instant Reverse



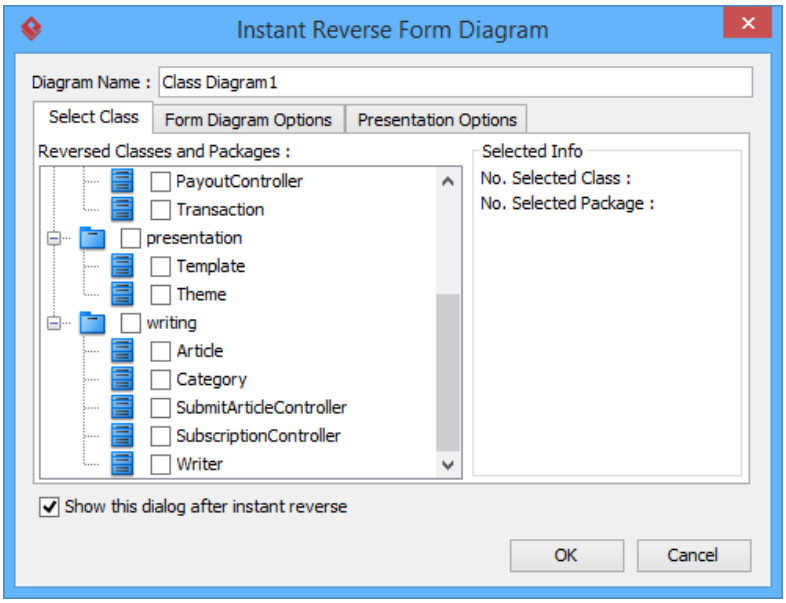
The instant reverse window

No.	Name	Description
1	Language	The language of source to reverse.
2	Update type	Determine how to handle duplicated classes by selecting the **Update Type**. Below is a description of the update types: **Update duplicate class(es)** - Update existing class(es) by source. **Replace duplicate class(es)** - Discard existing class(es), overwrite by reversed source.
3	Path	The path of source to reverse.
4	Exclude	The file(s) to exclude when reverse.
5	Encoding	The encoding of source files.
6	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named *Old*, to place system prototype to a model named *Prototype* and so forth.
7	OK	Click to start reversing.
8	Cancel	Click to close instant reverse.
9	Advanced Options	More configurable options related to Objective-C instant reverse.
7	Help	Click to read the Help contents.

Overview of instant reverse window

Forming class diagram from reversed classes

By the end of an instant reverse operation, you will be asked whether or not to form a class diagram with reversed UML classes. By selecting classes and configuring the way to present them and confirm, a diagram will then be formed.



*The **Instant Reverse Form Diagram** window*

NOTE: If you do not want Visual Paradigm to ask you for forming diagram next time you perform instant reverse, uncheck Show this window after instant reverse.

Below is a description of this window, base on the tabs.

Select Class

The classes listing in the tree are those reversed from your code-base. You must select at least one class in order to form a class diagram. Notice that forming diagram can be a costly operation if you have selected too many classes in forming diagram.

Form Diagram Options

Option	Description
Show superclasses	Show the generalization relationships between the selected elements and their super classes (ancestors) in the new diagram.
Show subclasses	Show the generalization relationships between the selected elements and their subclasses (descendants) in the new diagram.
Show suppliers (interface)	Show the realization relationships between the selected elements and their suppliers (interfaces) in the new diagram.
Show clients	Show the realization relationships between the selected elements and their clients (classes that implements them) in the new diagram.
Show navigable classes	Show the association relationships between the selected elements and their navigable classes (targets) in new diagram.
Show non-navigable classes	Show the association relationships between the selected elements and their non-navigable classes (sources) in the new diagram.
Show containers	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram.
Show residents	Show the containment relationships between the selected elements and their containers (e.g. packages) in the new diagram. Single level only - Show one level residents only in the new diagram. All levels in single diagram - Show all level residents in the new diagram. All levels in sub diagrams - Show all level residents in the new diagrams (multiple single level diagrams).
Show as containment relationships	Show the containment relationships as connectors in the new diagram. Notes: The containment relationships between classes are shown as connectors.

Description of form diagram options

Presentation Options

Option	Description

Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in the new diagram.
Operation options	Show all - Show all operations of classes in the new diagram. Public only - Show all public operations of classes only in the new diagram. Hide all - Hide all operations of classes in the new diagram. Initial values - Show initial values of operations of classes in the new diagram.
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.

Description of presentation options

Related Resources

The following resources may help you learn more about the topic discussed in this page.

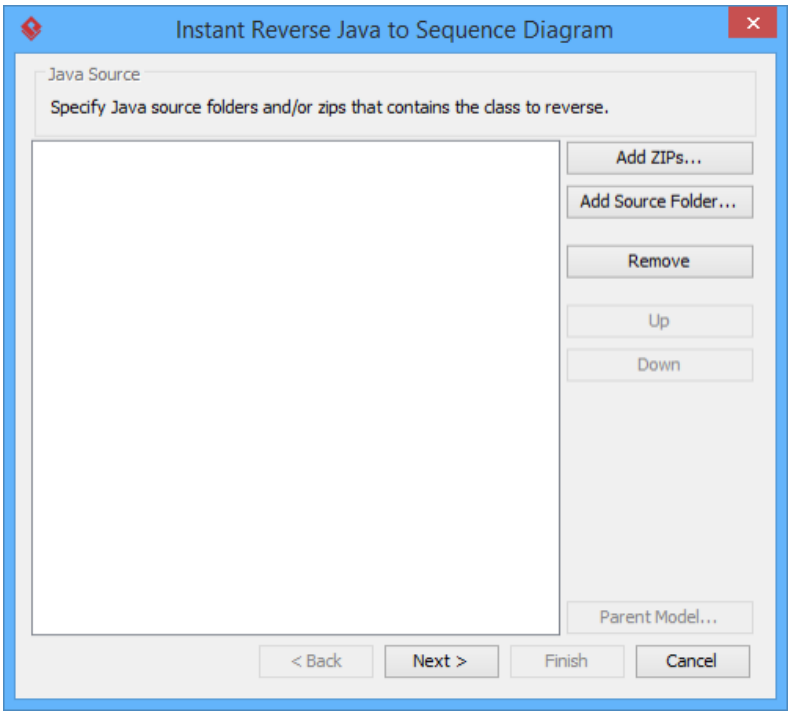
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant reverse Java sources to sequence diagram

Instant reverse is a process to produce UML class model from a given input of source code. With instant reverse, you can reverse a snap shot of your code-base to UML classes and form class diagram in further. Instant reverse can read the code body of operation in Java class (source file), analyze the method invocations and form the result on a sequence diagram. This allows you to study the runtime behavior of your application by means of a sequence diagram, which makes it easier to locate potential bottleneck and carry out changes.

Reverse engineering sequence diagram from source files

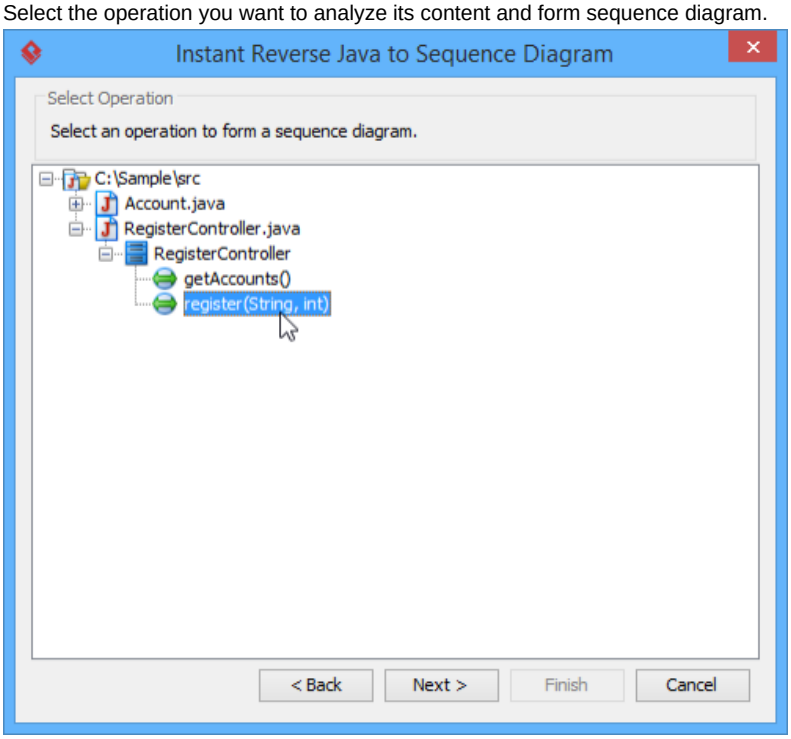
1. Select **Tools > Code > Instant Reverse Java to Sequence Diagram...** from the toolbar.
2. In the **Instant Reverse** window, add the zip file of source or folder path of source by clicking on the appropriate **Add** button at the right hand side of the window. Make sure the source folders include all the source files of all classes necessary for analyzing the traces of calls.



*The **Instant Reverse** window*

NOTE: You can reverse multiple source paths by adding them one after the other.

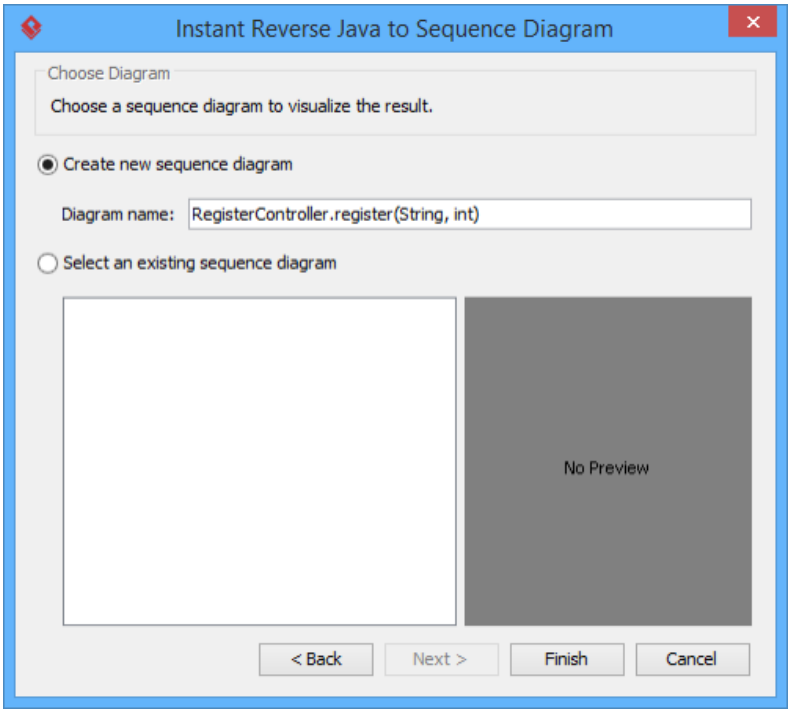
3. Click **Next**.
4. Select the operation you want to analyze its content and form sequence diagram.



Select an operating to analyze its code body and form diagram

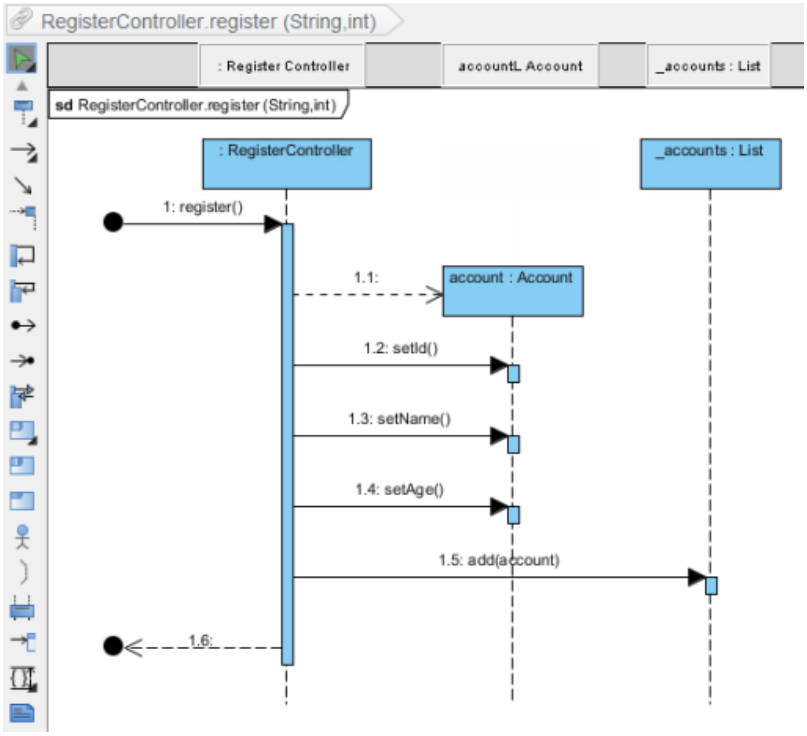
5. Click **Next**.

6. In the **Choose Diagram** page, select the diagram to visualize the result. You can either form a new sequence diagram by selecting **Create new sequence diagram** and entering the diagram name or select **Select an existing sequence diagram** and choose an existing sequence diagram to visualize the result.



Select a diagram to visualize the result

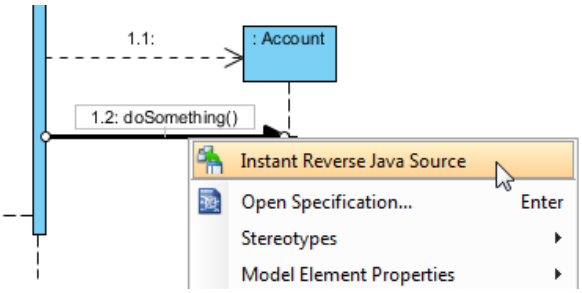
7. Click **Finish**. When the process is completed, you can obtain the result in sequence diagram.



Sequence diagram formed

Reversing Deeper Level of Code Details

Instant reverse does not drill inside method calls indefinitely. Instead, it reverse just the operation selected. If you want to reverse deeper level of details, right click on the target sequence message and select **Instant Reverse Java Source** from the popup menu.



Reverse Java source with a sequence message

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generation

Instant generator is the process of producing source code from class model. In this chapter, you will learn how to make use of Instant Generator to generate source code from UML classes.

Instant Generator for Java

Generate Java source file from UML classes.

Instant Generator C# source code

Generate C# source file from UML classes.

Instant Generator for VB.NET source code

Generate VB.NET source file from UML classes.

Instant Generator for PHP source code

Generate PHP source file from UML classes.

Instant Generator for ODL source code

Generate ODL source file from UML classes.

Instant Generator for ActionScript source code

Generate ActionScript source file from UML classes.

Instant Generator for IDL source code

Generate IDL source file from UML classes.

Instant Generator for C++ source code

Generate C++ source file from UML classes.

Instant Generator for Delphi source code

Generate Delphi source file from UML classes.

Instant Generator for Perl source code

Generate Perl source file from UML classes.

Instant Generator for XML Schema file

Generate XML Schema source file from UML classes.

Instant Generator for Python source code

Generate Python source file from UML classes.

Instant Generator for Objective-C source code

Generate Objective-C source file from UML classes.

Instant Generator for Objective-C 2.0 source code

Generate Objective-C 2.0 source file from UML classes.

Instant Generator for Ada95

Generate Ada95 source file from UML classes.

Instant Generator for Ruby

Generate Ruby source file from UML classes.

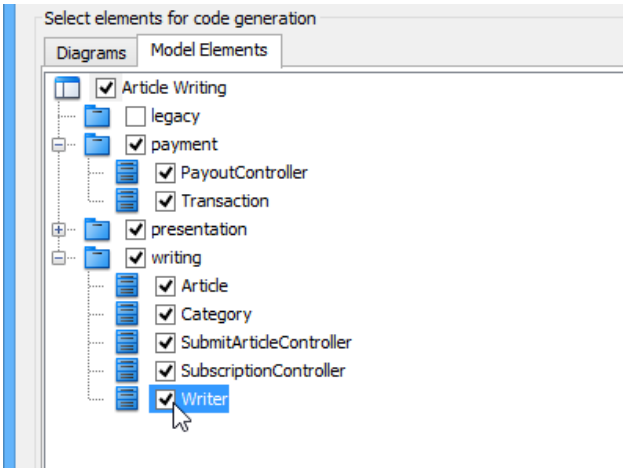
Customizing code generation

Customize the code generation template to control the output of instant generator.

Instant Generator for Java

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Java. To generate code by instant generator:

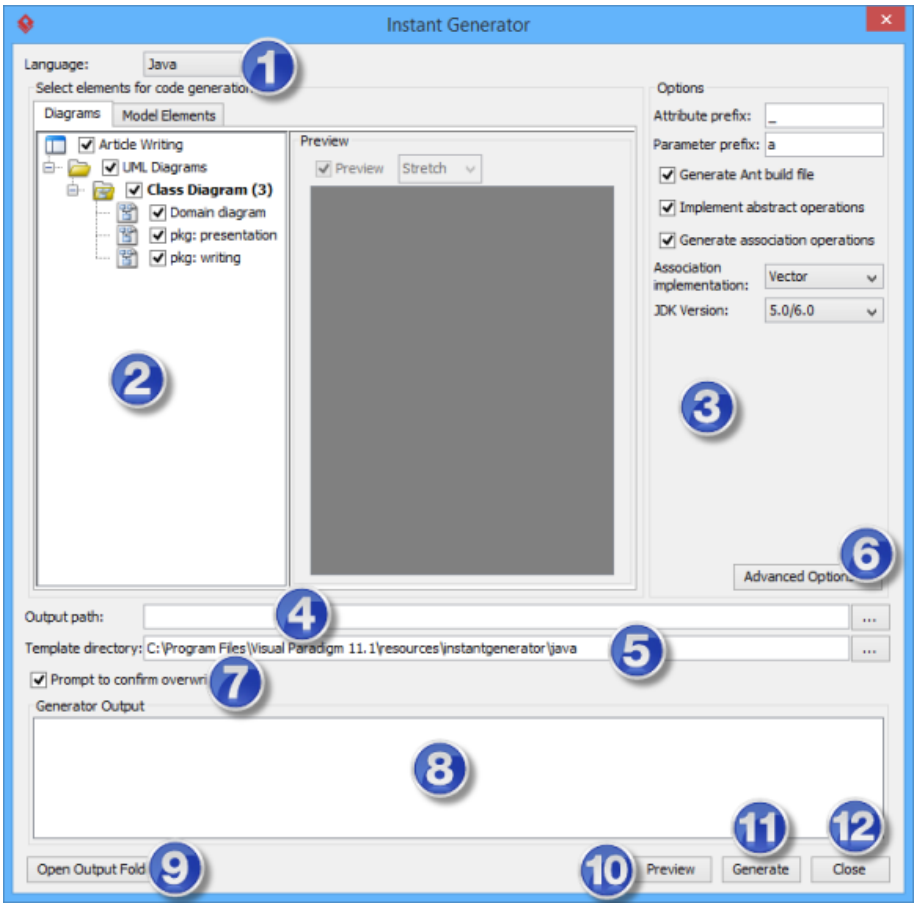
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Java** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

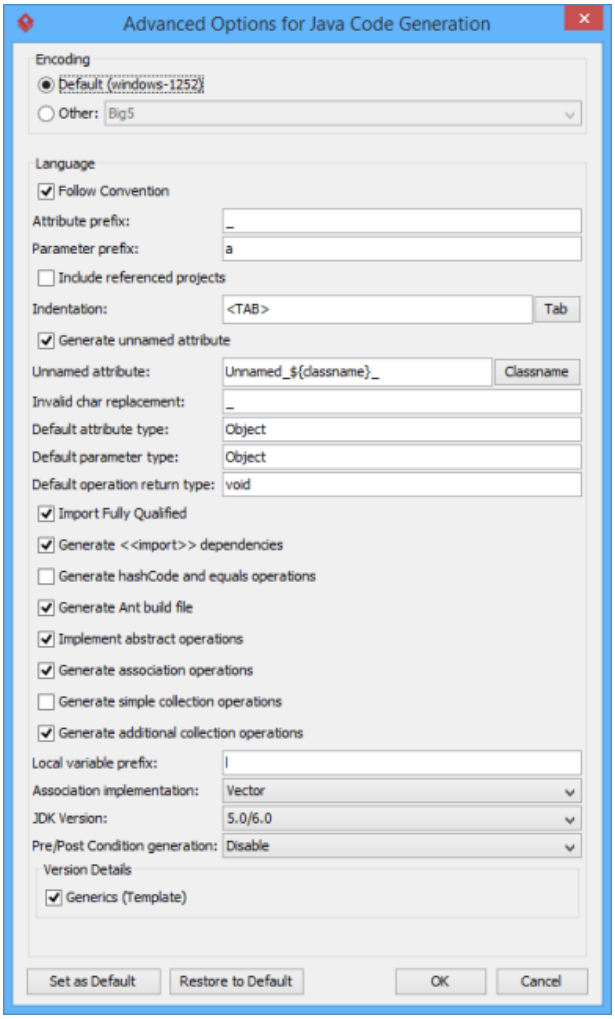
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Follow Convention	Whether to apply camel case Java naming convention.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Import Fully Qualified	Whether to state in import statement the class to import, or use asterisk to present an import on all classes in a package.
Generate hashCode and equals operations	Whether or not to generate hashCode() and equals() for each class.
Generate Ant build file	Whether or not to generate Ant build file.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.
Association implementation	The type of collection to be used for association.
JDK Version	Generate code in a specific standard of JDK.
Pre/Post Condition generation	You can defined pre and post conditions in class, attribute and operation specification. Check this option to generate them as comment in code.
Generics (Template)	Whether to generate template or not.

A description of advanced options

Related Resources

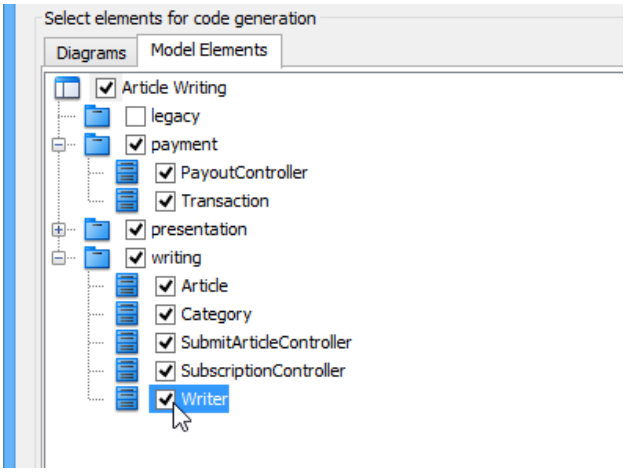
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator C# source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of C# source code. To generate code by instant generator:

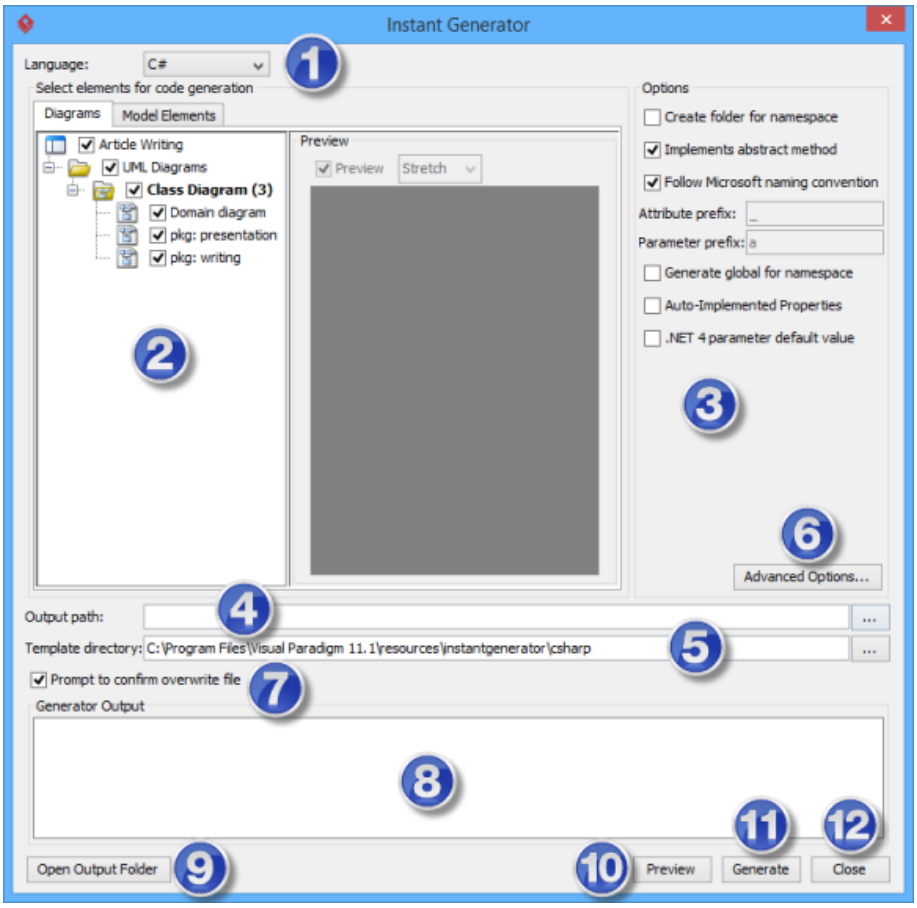
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **C#** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

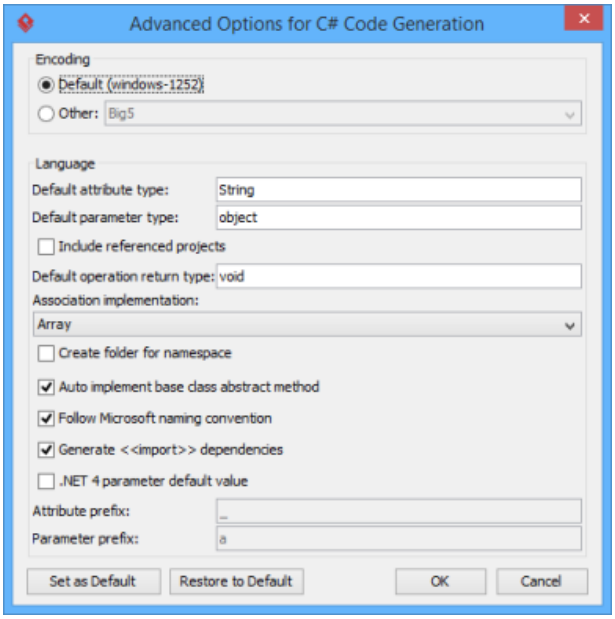
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Allow From Linked Project	Check to generate also classes in referenced project.
Default operation return type	Operation return type that will be used when operation has no return type specified.

Association implementation	The type of collection to be used for association.
Create folder for namespace	Create a directory in system for namespace
Auto implement base class abstract method	Whether or not to generate operations for implementing abstract operations defined in super class.
Follow Microsoft naming convention	Make the code convention follow Microsoft
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.

A description of advanced options

Related Resources

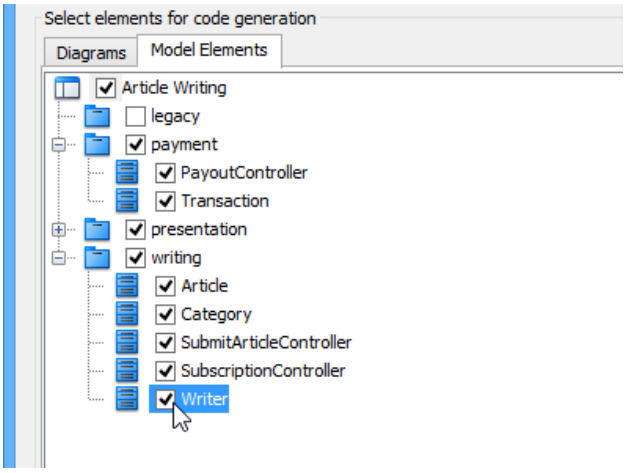
The following resources may help to you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for VB.NET source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of VB.NET. To generate code by instant generator:

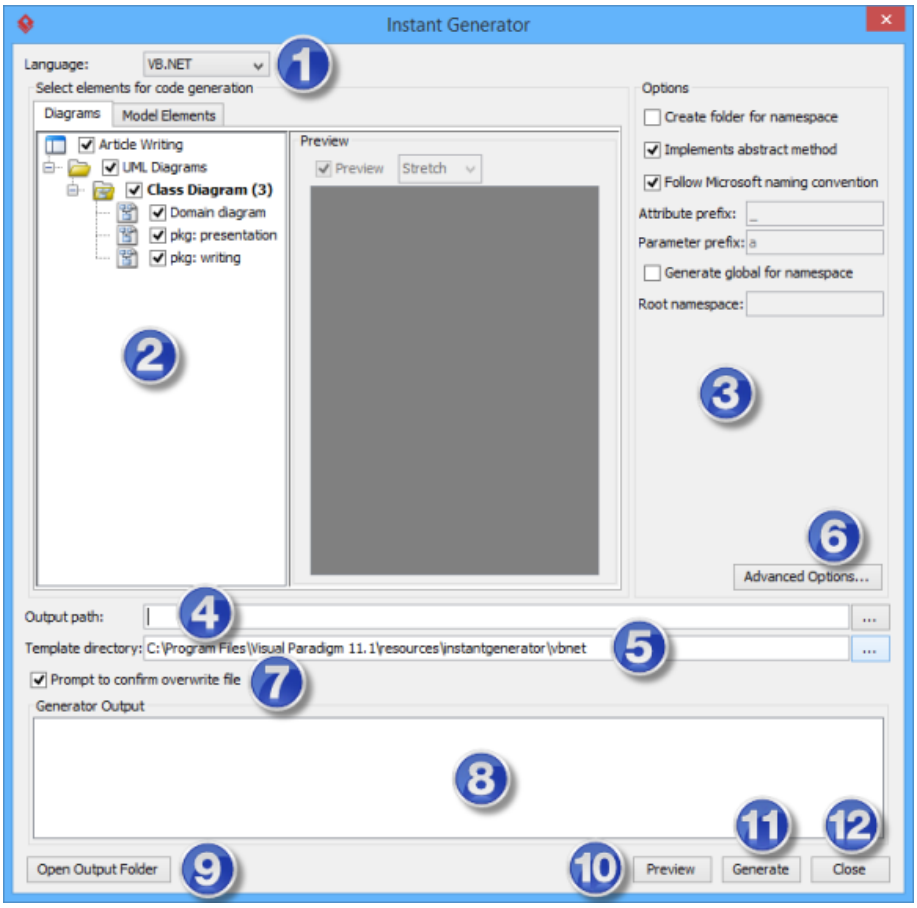
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **VB.NET** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

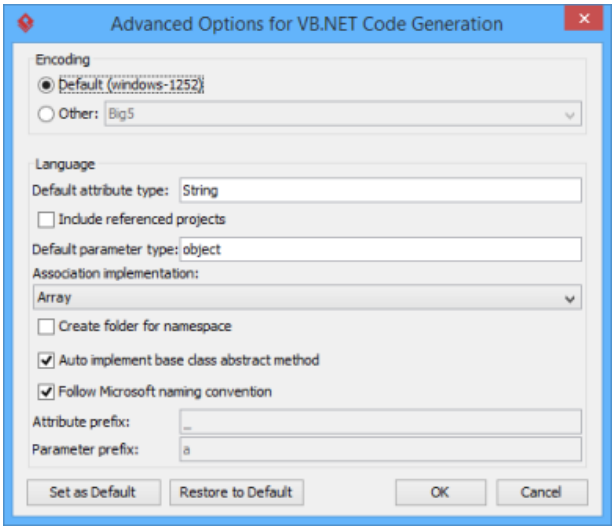
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Allow From Linked Project	Check to generate also classes in referenced project.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Association implementation	The type of collection to be used for association.
Create folder for namespace	Create a directory in system for namespace

Auto implement base class abstract method	Whether or not to generate operations for implementing abstract operations defined in super class.
Follow Microsoft naming convention	Make the code convention follow Microsoft
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.

A description of advanced options

Related Resources

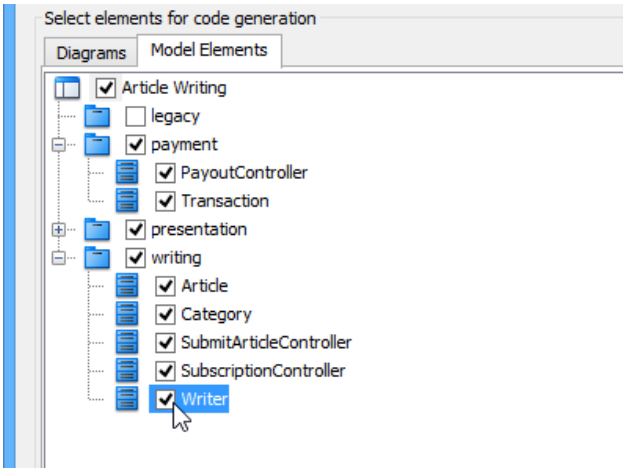
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for PHP source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of PHP. To generate code by instant generator:

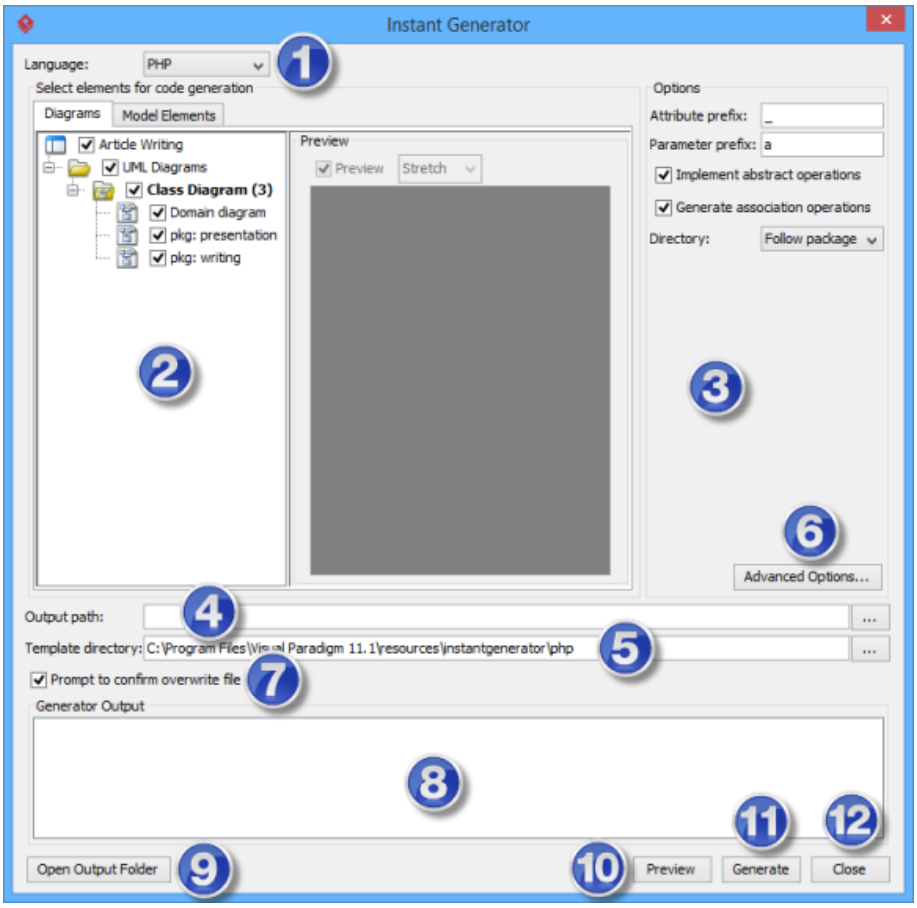
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **PHP** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

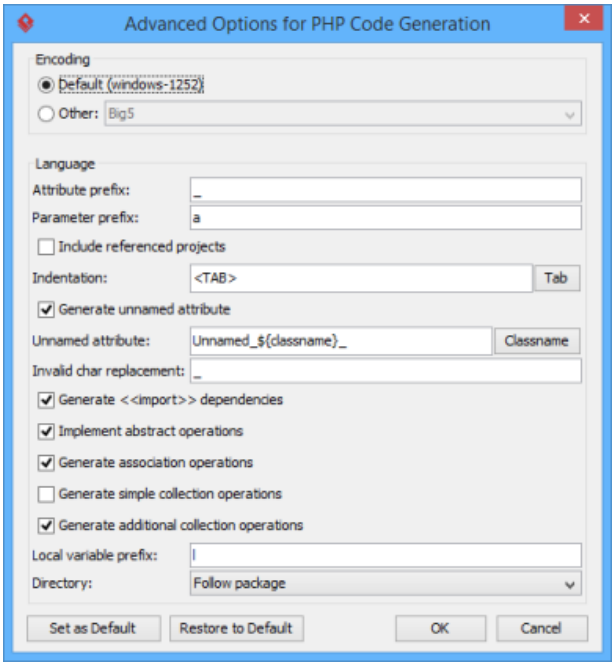
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.

Indentation	Character(s) used for indentation, default is Tab.
Generate unnamed attribute	Whether to generate nameless attributes.
Unnamed attribute	The naming pattern of nameless attributes.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing thsoe invalid characters by given character.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations in subclass.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.
Directory	**Follow package** - generate source in directory same as package's structure **Flat level** - generate source in same directory (only one directory)

A description of advanced options

Related Resources

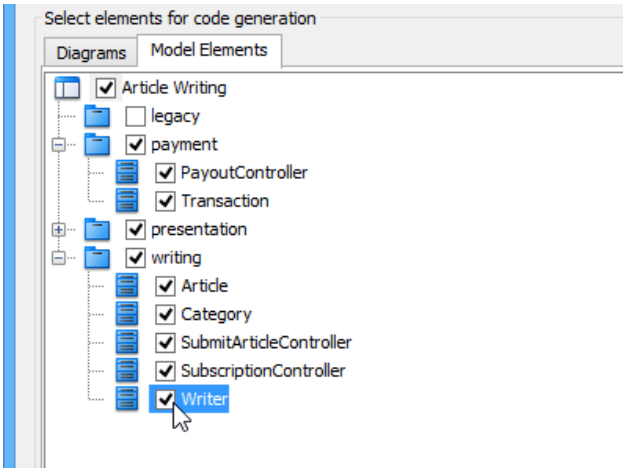
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for ODL source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of ODL. To generate code by instant generator:

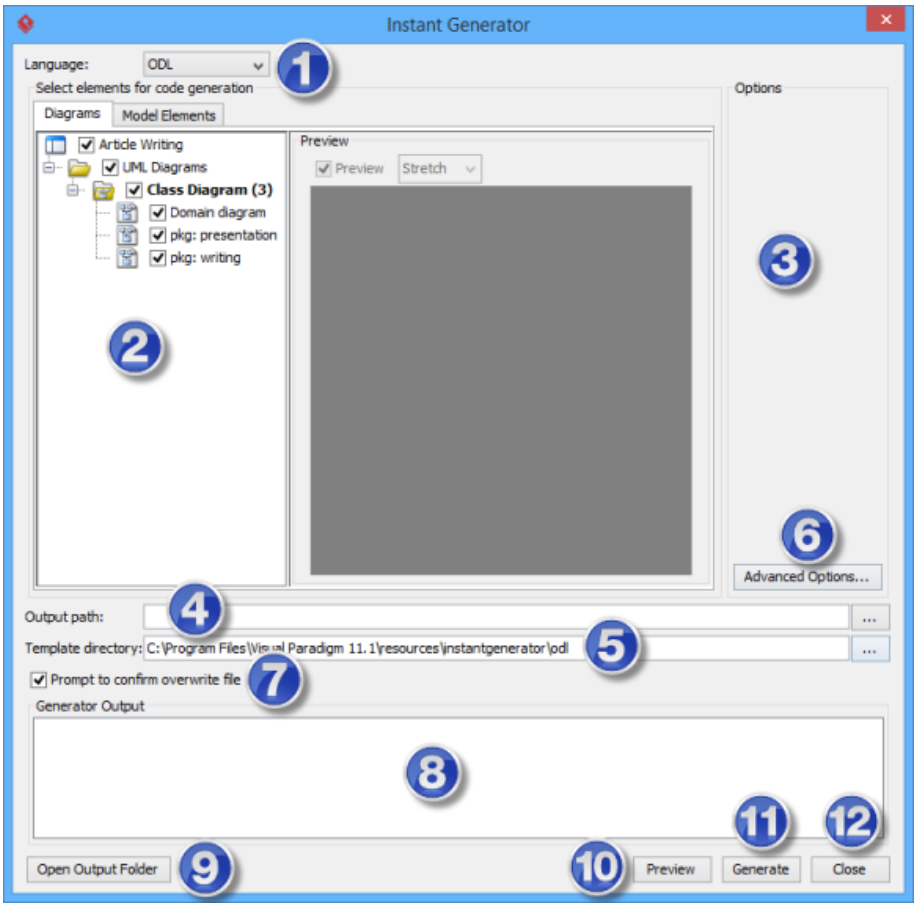
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **ODL** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

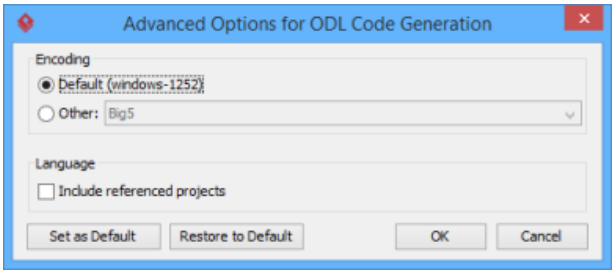
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Allow From Linked Project	Check to generate also classes in referenced project.

A description of advanced options

Related Resources

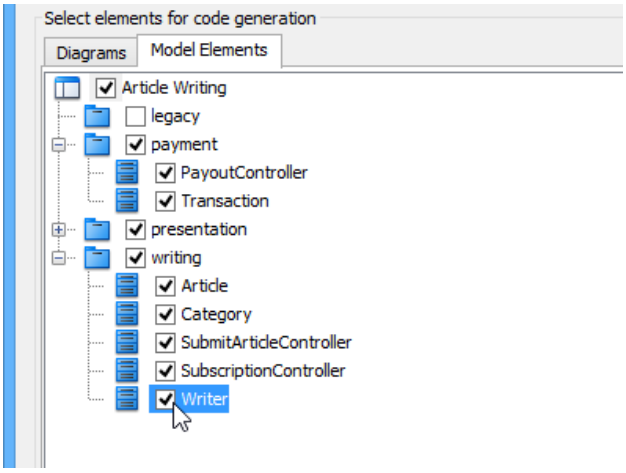
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for ActionScript source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of ActionScript. To generate code by instant generator:

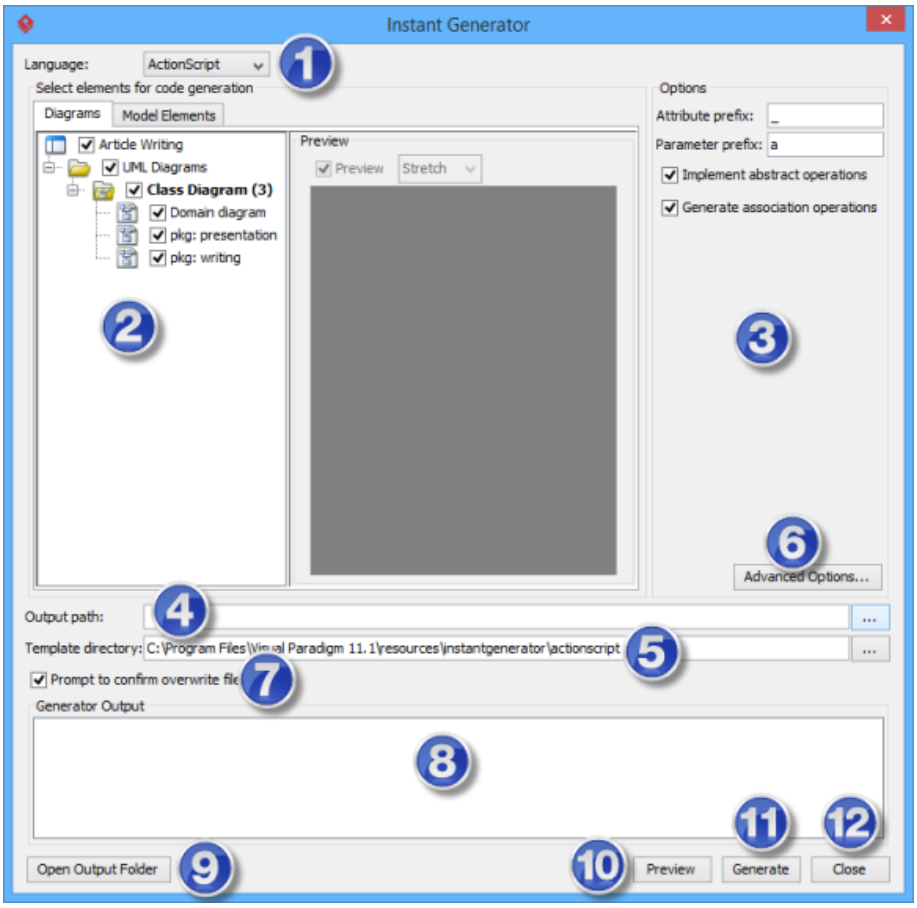
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **ActionScript** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

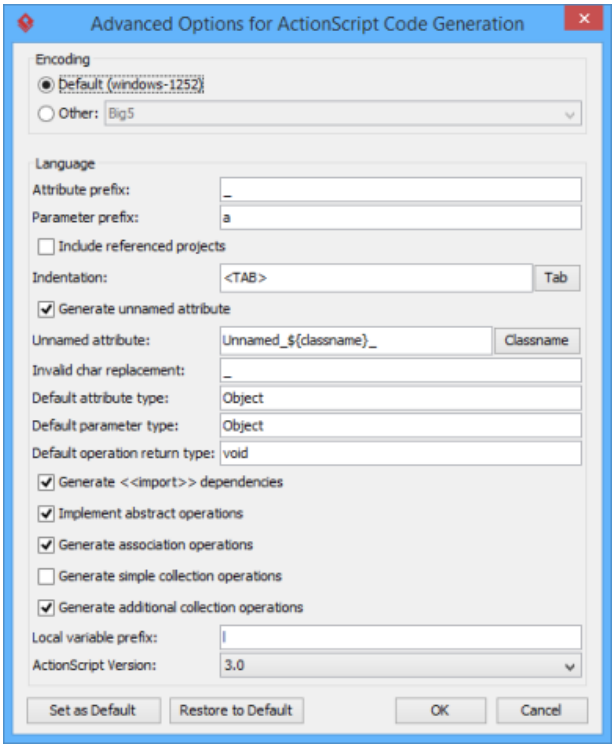
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.

Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.
ActionScript Version	Generate code in a specific standard of action script.

A description of advanced options

Related Resources

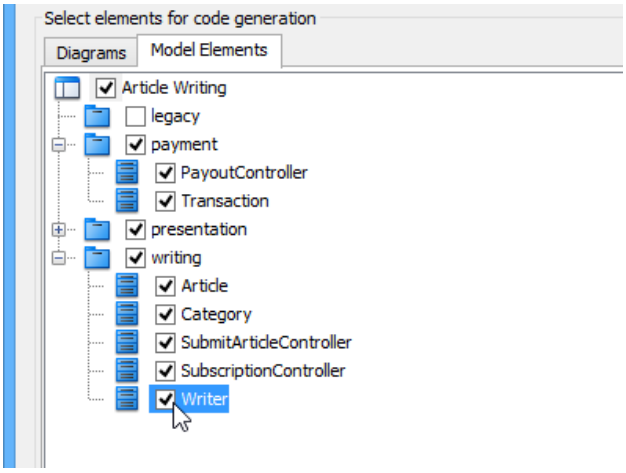
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for IDL source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of IDL. To generate code by instant generator:

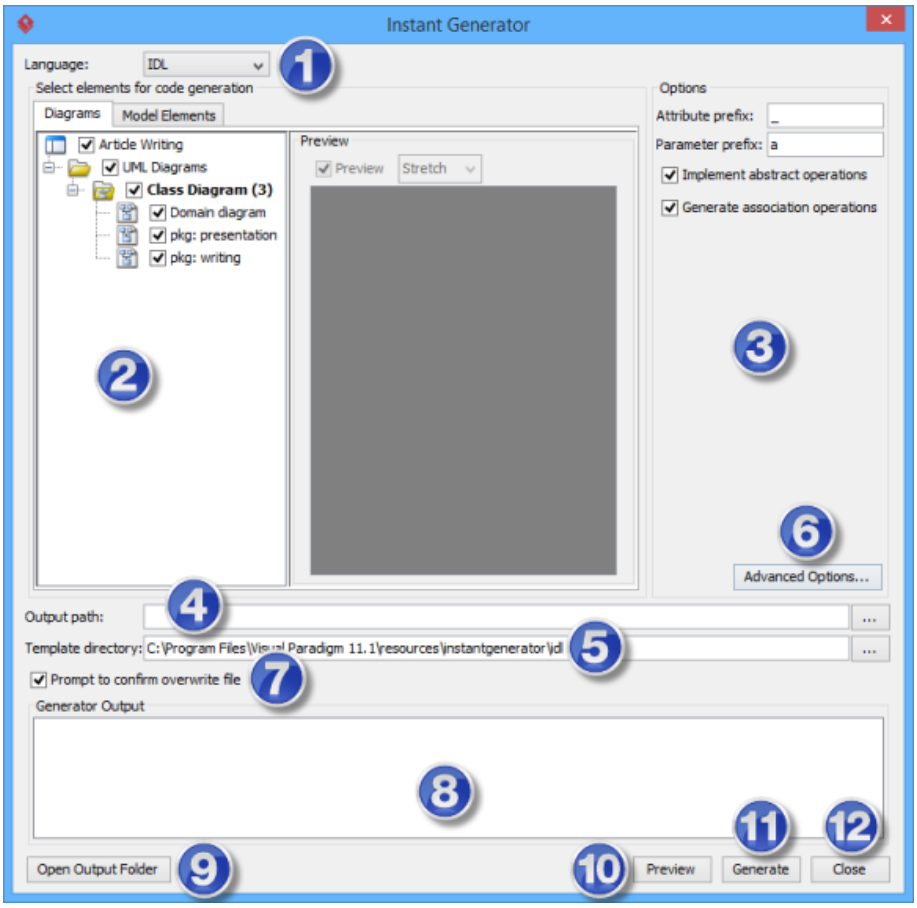
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **IDL** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

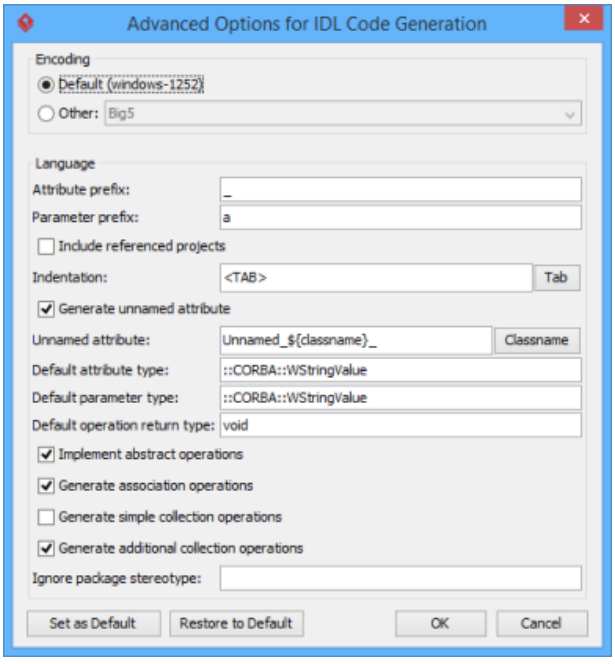
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.

Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.

A description of advanced options

Related Resources

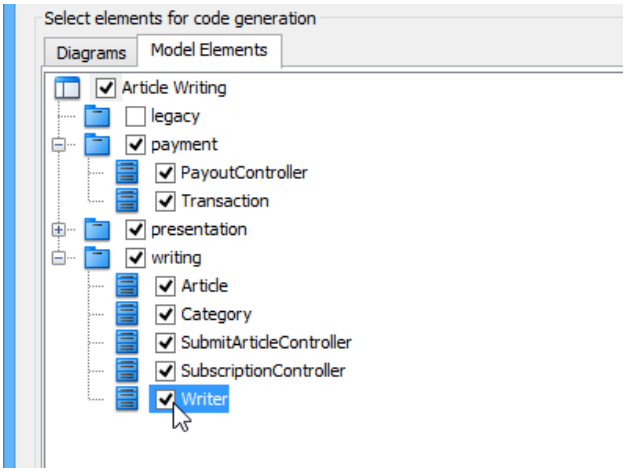
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for C++ source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of C++. To generate code by instant generator:

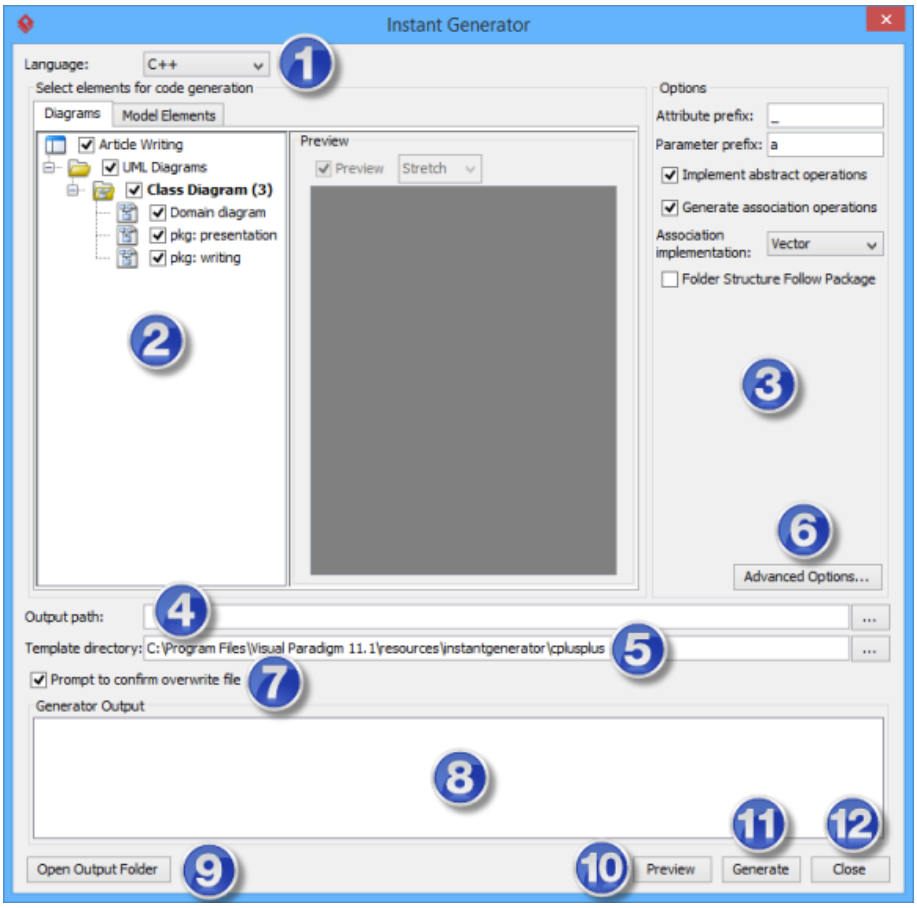
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **C++** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

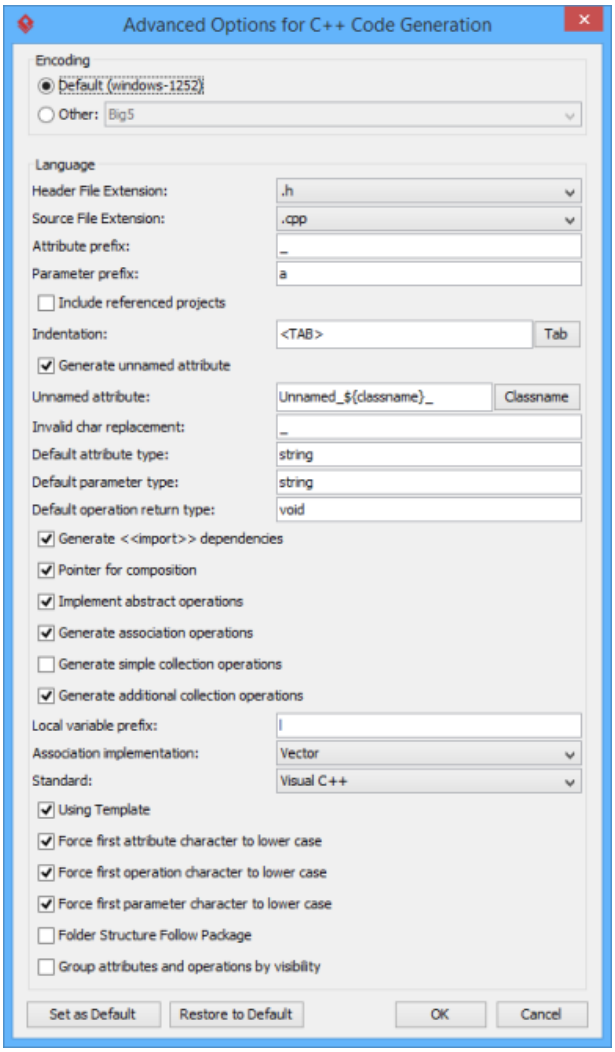
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Pointer for composition	When checked, generate attribute for linking composited class using pointer (by reference).
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.

Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.
Association implementation	The type of collection to be used for association.
Standard	**ANSI C++** - Most general standard of C++ **Visual C++** - Microsoft enhanced ANSI C++ and develop another standard
Using Template	Whether to generate template or not.
Force first attribute character to lower case	Force the first character in attribute name to be in lower case.
Force first operation character to lower case	Force the first character in operation name to be in lower case.
Force first parameter character to lower case	Force the first character in parameter name to be in lower case.
Folder Structure Follow Package	Generate folders according to package structure.
Group By Visibility	Group class members by their visibility.

A description of advanced options

Related Resources

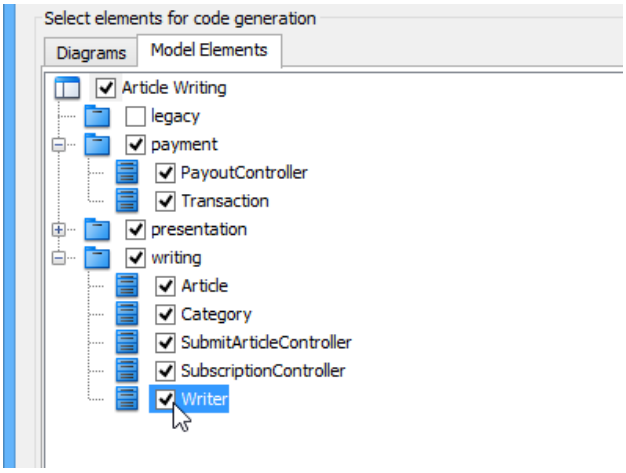
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Delphi source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Delphi. To generate code by instant generator:

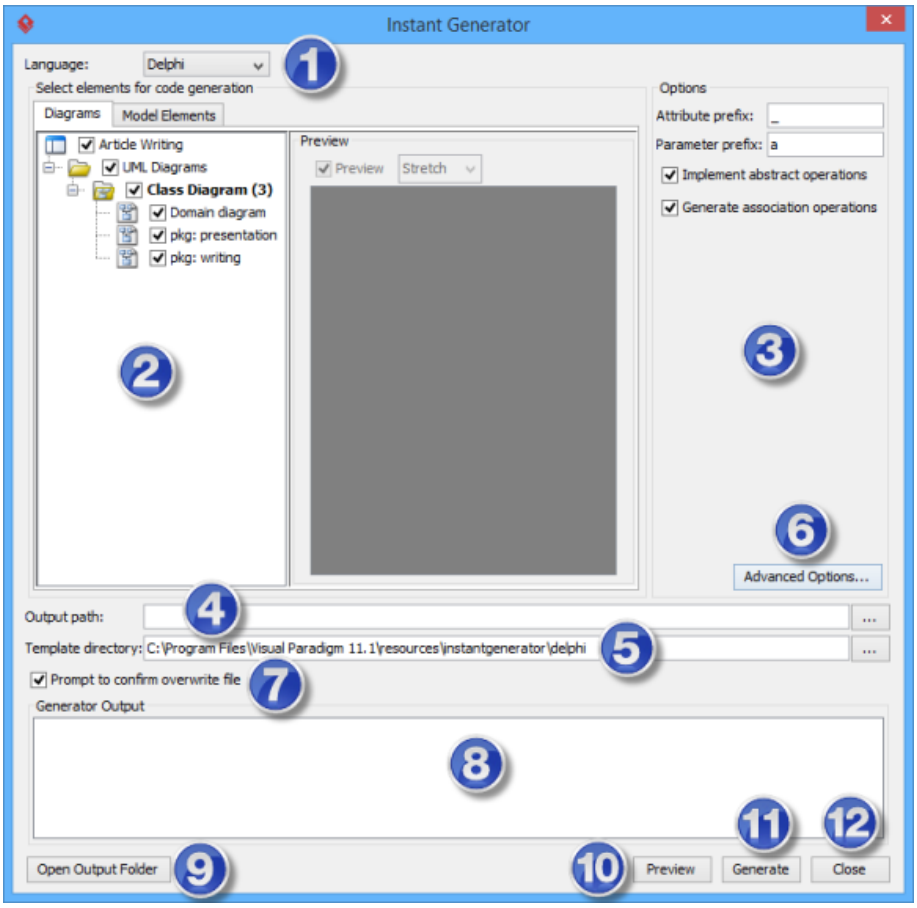
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Delphi** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

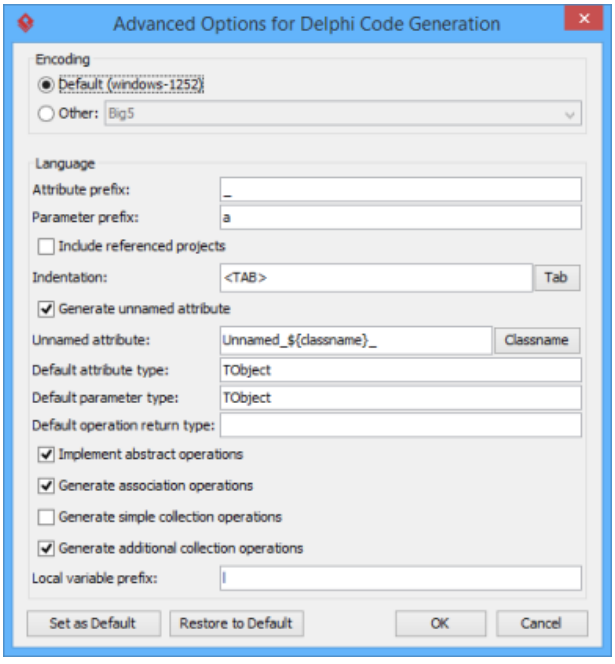
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.

Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.

A description of advanced options

Related Resources

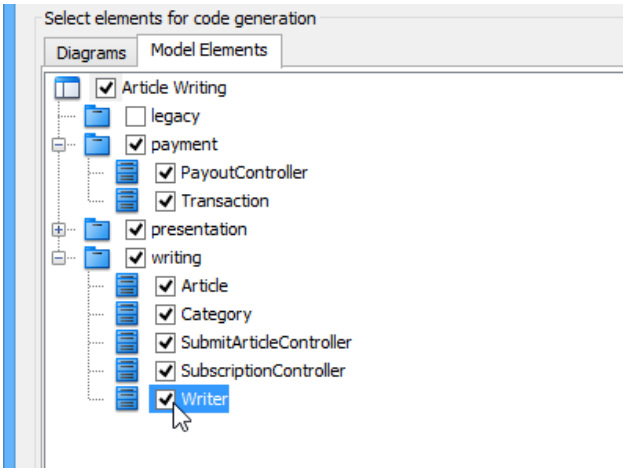
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Perl source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Perl. To generate code by instant generator:

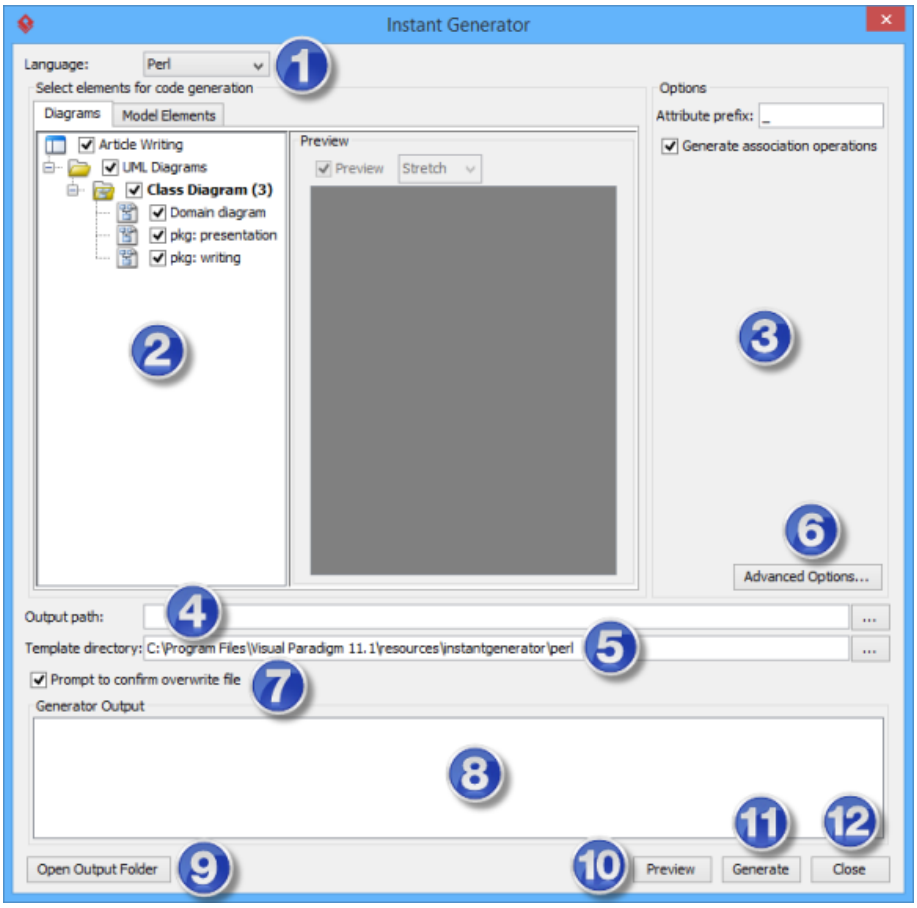
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Perl** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

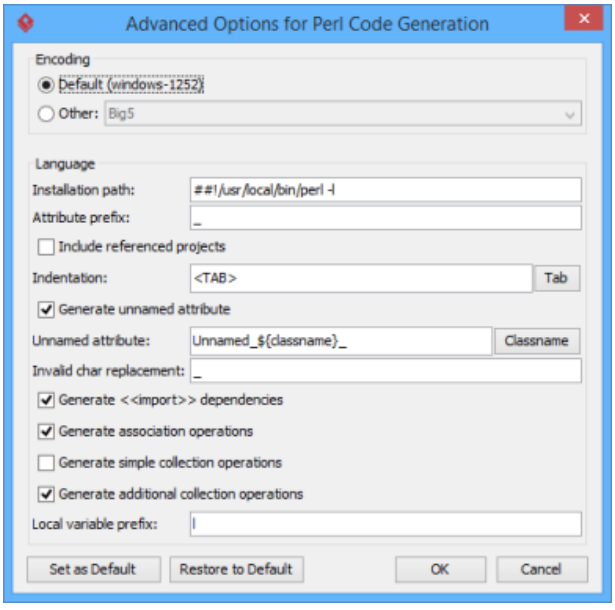
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Installation path	The Perl installation path
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.

Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.

A description of advanced options

Related Resources

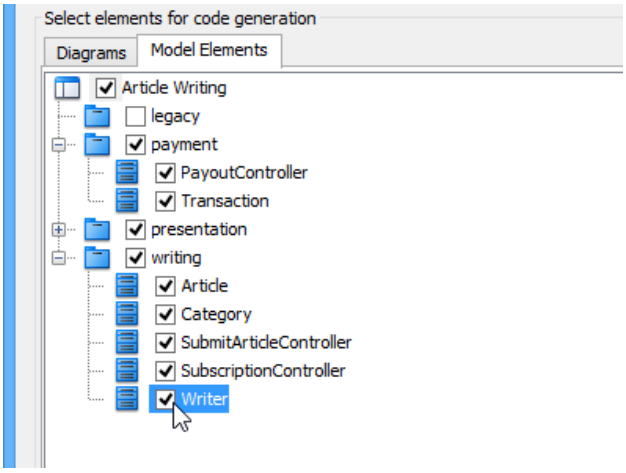
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for XML schema file

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of XML Schema. To generate code by instant generator:

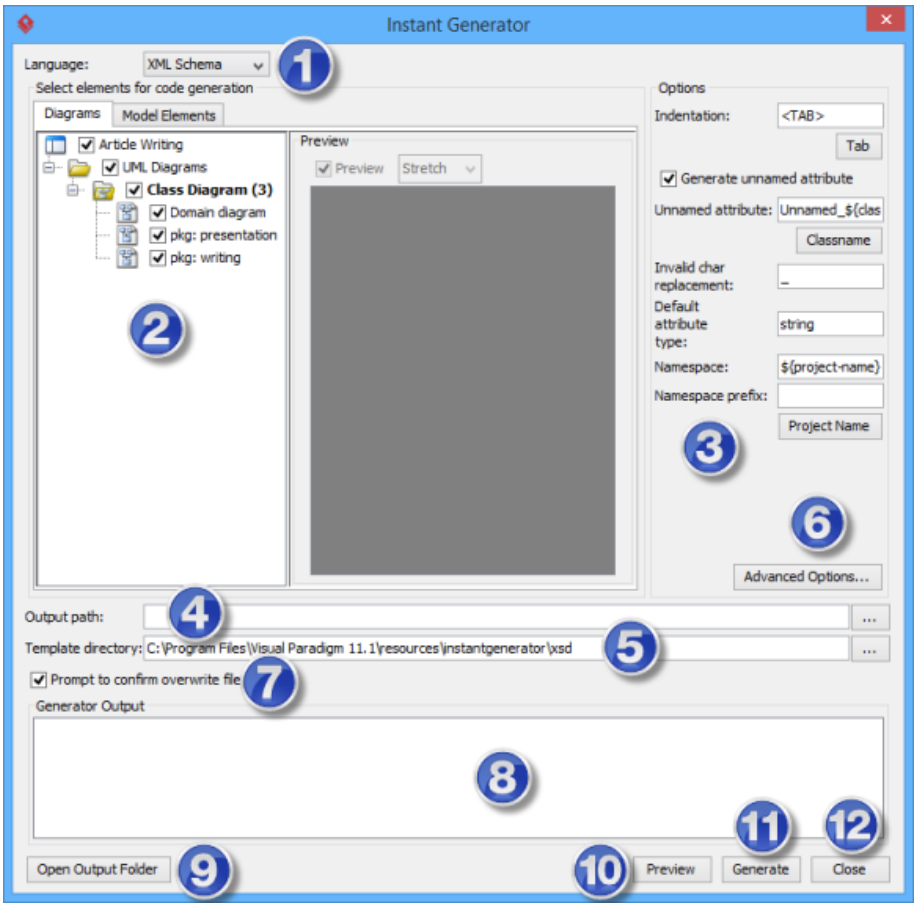
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **XML Schema** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

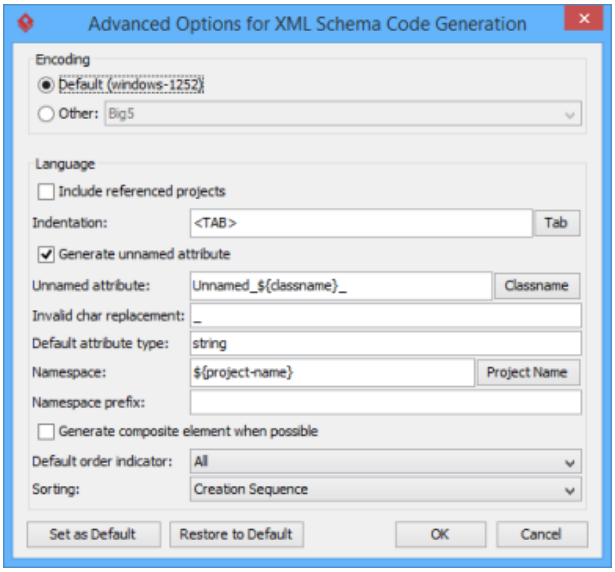
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab **.**
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.

Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Namespace	Define a namespace for generated code.

A description of advanced options

Related Resources

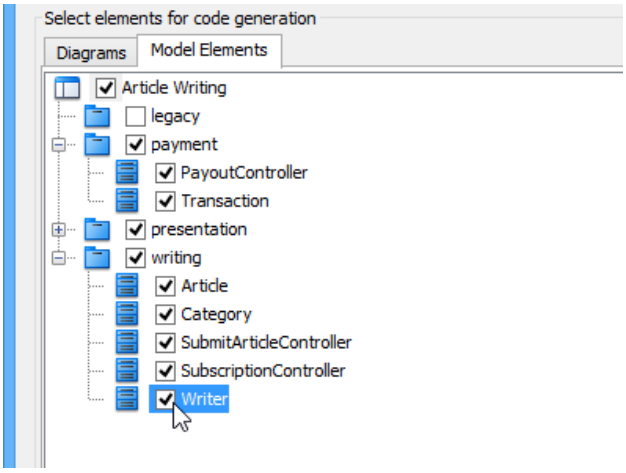
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Python source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Python. To generate code by instant generator:

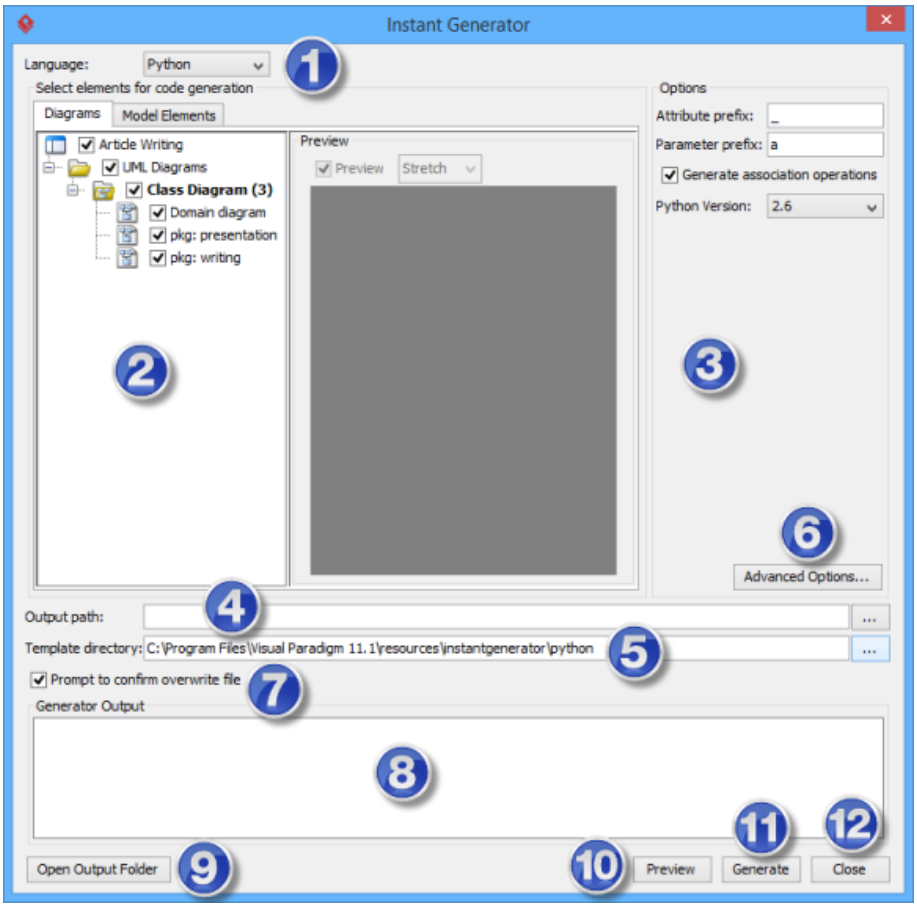
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Python** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

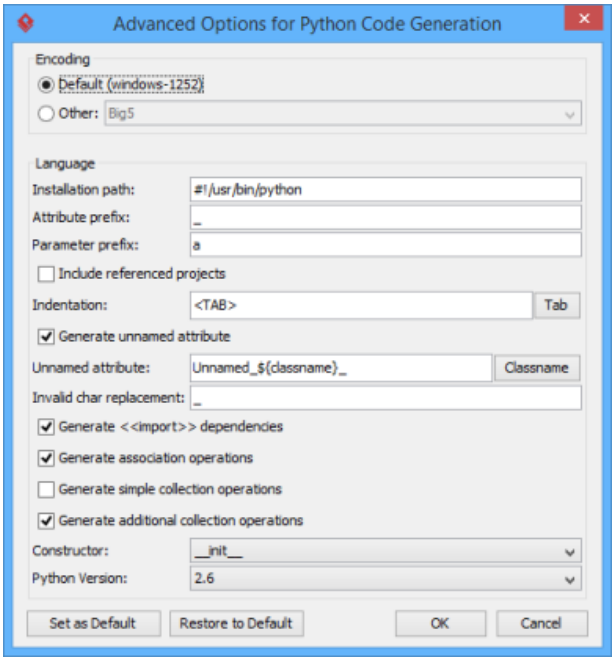
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Installation path	The installation path of Python.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.

Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Constructor	Select the constructor to use
Python Version	Generate code in a specific standard of Python.

A description of advanced options

Related Resources

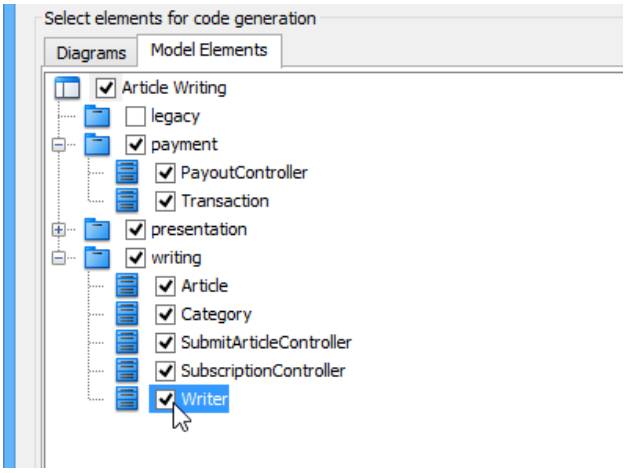
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Objective-C source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Objective-C. To generate code by instant generator:

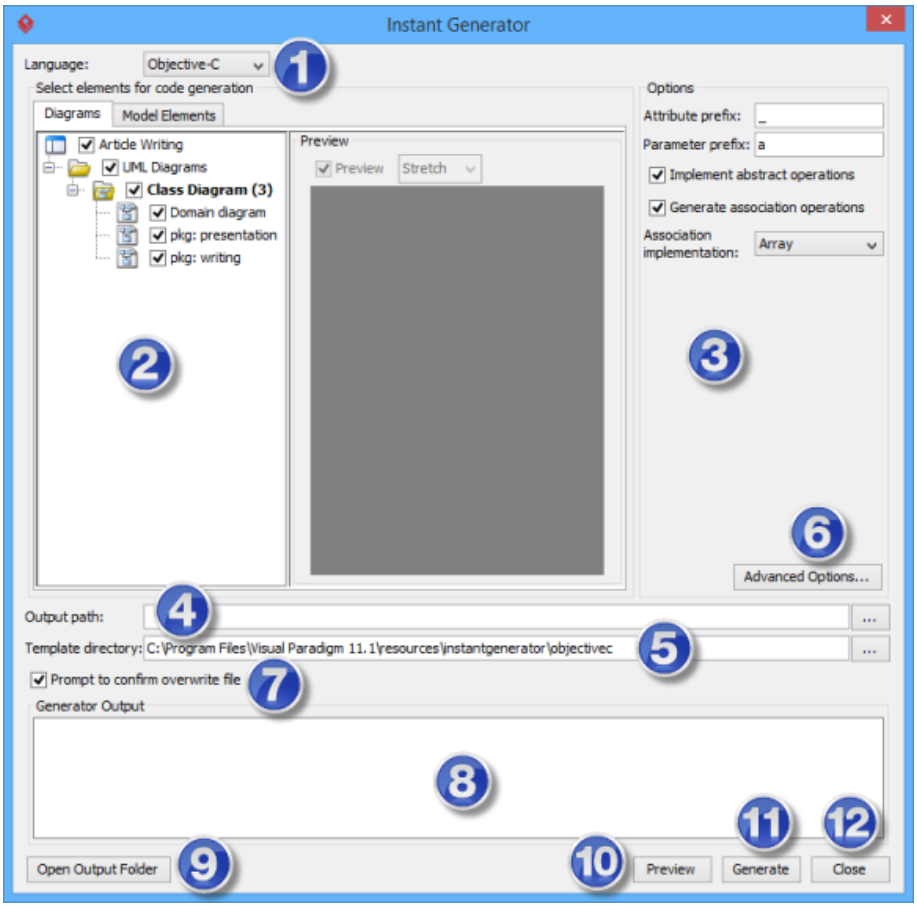
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Objective-C** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

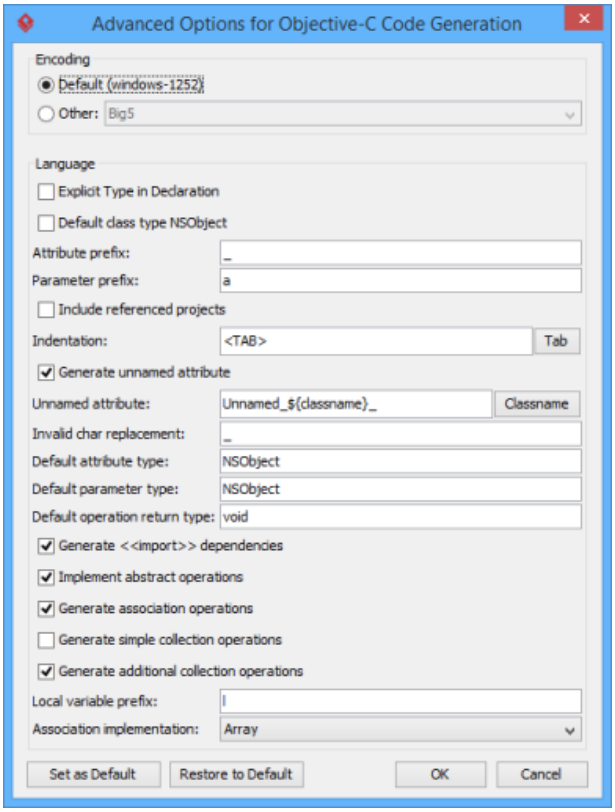
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.

Explicit Type in Declaration	hen checked, will generate attribute/parameter type with specified type or just id.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.
Local variable prefix	The characters to be appended to local variables.
Association implementation	The type of collection to be used for association.

A description of advanced options

Related Resources

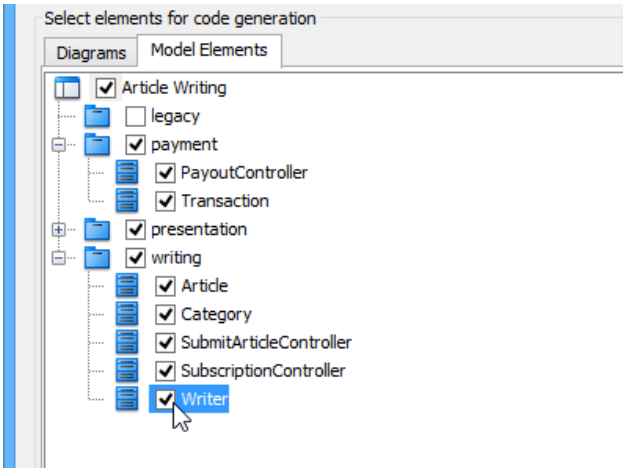
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Objective-C 2.0 source code

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Objective-C 2.0. To generate code by instant generator:

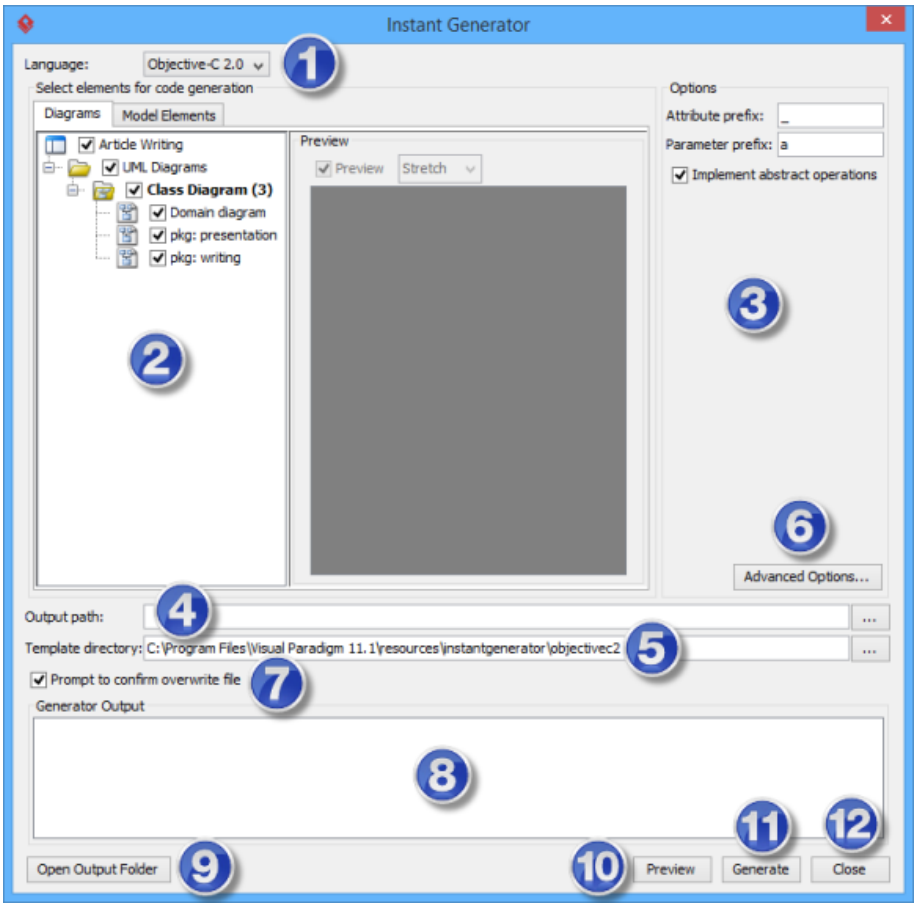
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Objective-C 2.0** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator window

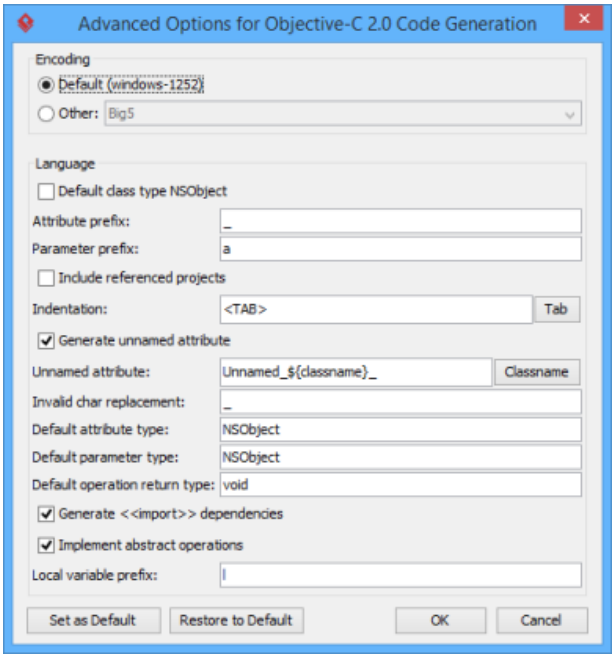
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.

Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Local variable prefix	The characters to be appended to local variables.

A description of advanced options

Related Resources

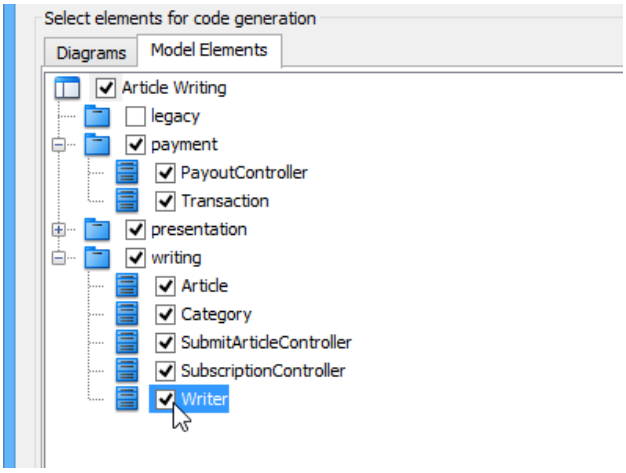
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Ada95

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Ada95. To generate code by instant generator:

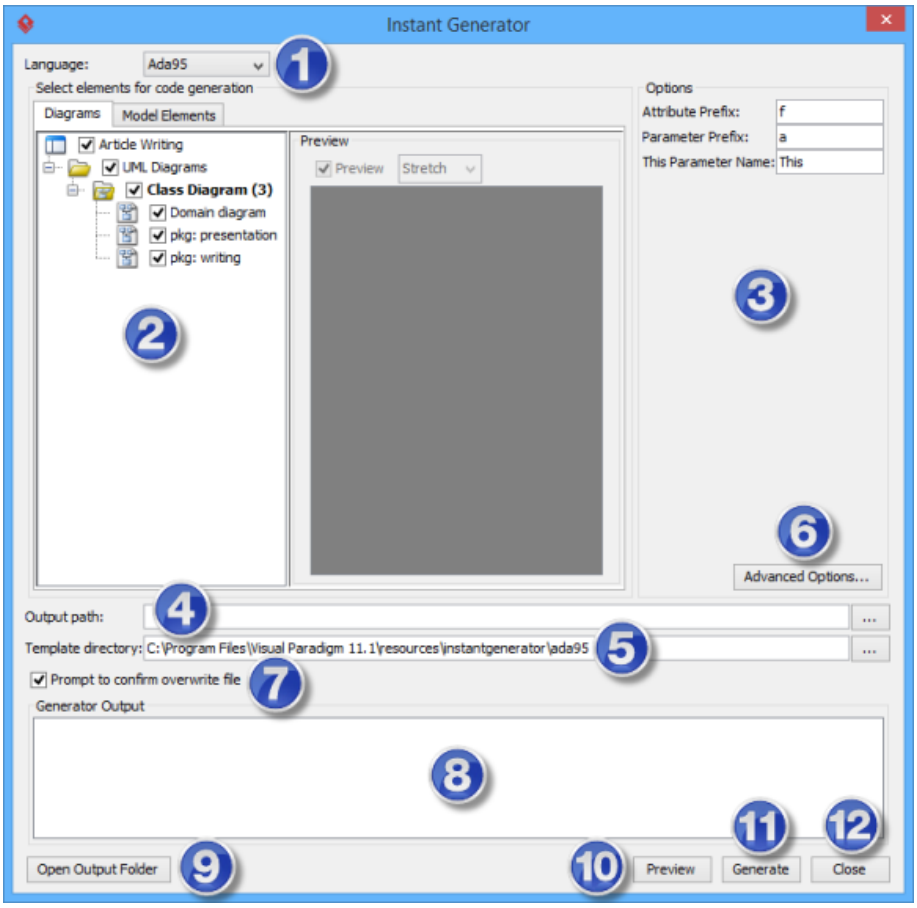
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Ada95** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of Instant Generator window

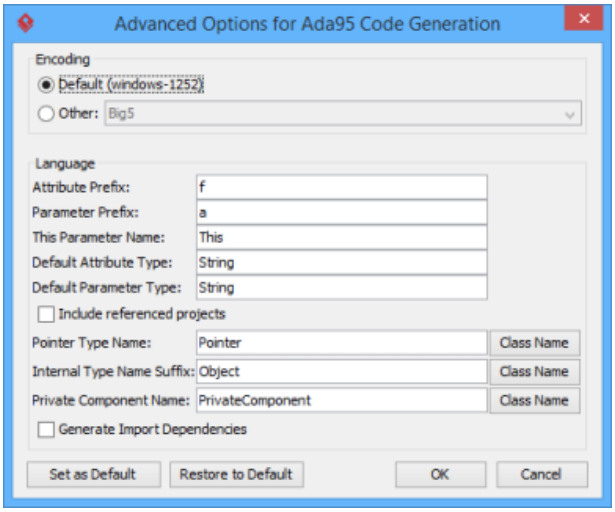
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new window.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator window

Generator options

On the **Instant Generator** window you can configure some of the common code options at the right of window. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options window

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
This Parameter Name	The name of the pointer which for accessing object itself.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Allow from linked project	Check to generate also classes in referenced project.

Pointer type name	The name of the pointer for accessing object's associated class.
Internal type name suffix	The name of the type which is generated for internal use.
Private component name	The name of the type which is used for containing the private member.

A description of advanced options

Related Resources

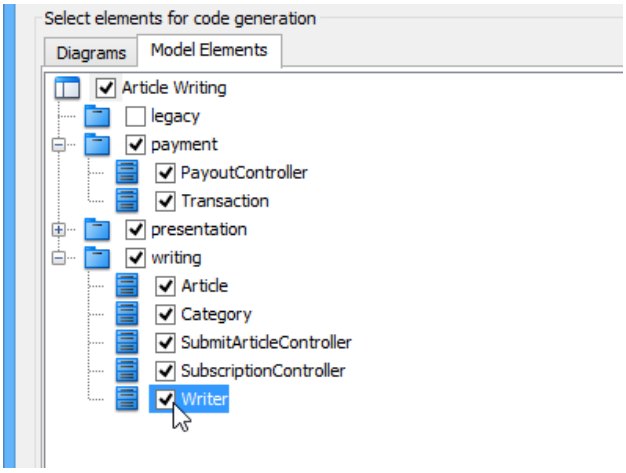
The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator for Ruby

Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling and eventually generate source code from implementation model. This chain makes building software faster and cheaper. In this chapter, we will go through the instant generation of Ruby. To generate code by instant generator:

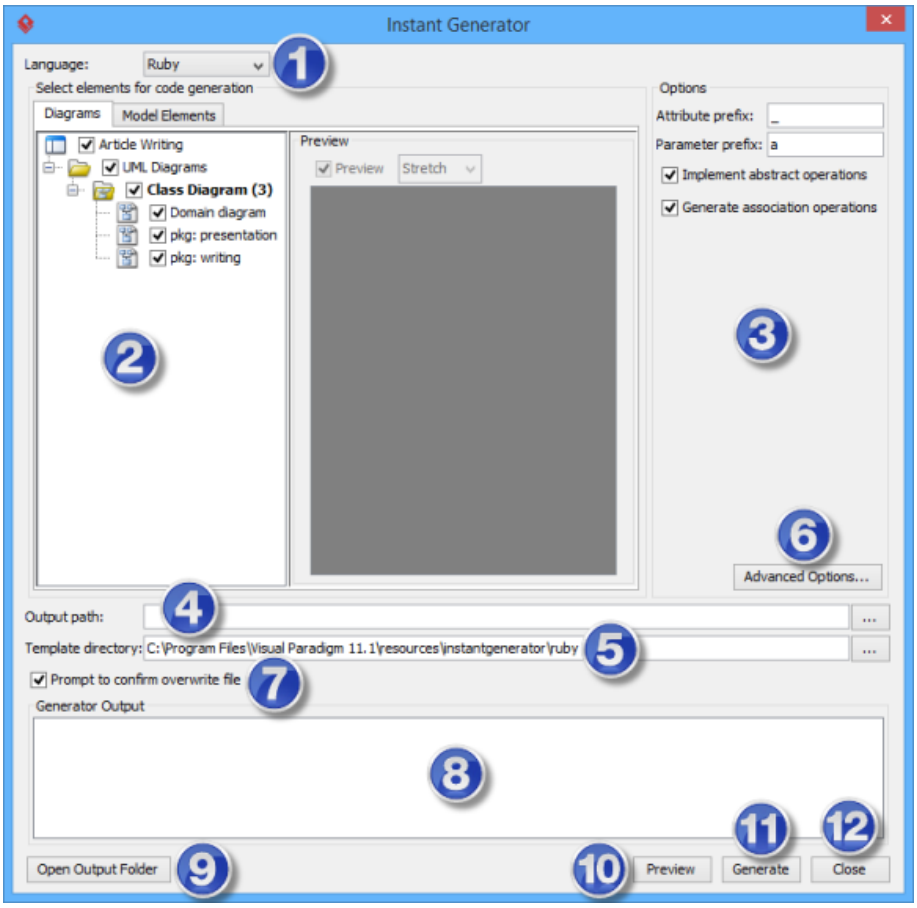
1. Select **Tools > Code > Instant Generator** from the toolbar.
2. In the **Instant Generator** window, select **Ruby** as the **Language**.
3. Fill in the **Output Path**, which is the directory where you want the code to generate to.
4. Select the classes to generate code. In the **Diagrams** tab, you can select the diagrams to generate code for classes in the selected diagrams. Alternatively, open the **Model Elements** tab and select the classes to generate code.



Selecting classes to generate code

5. Optionally configure the generator options. Read the section below for a description of options.
6. Click **Generate** to generate code.

Overview of Instant Generator



Overview of instant generator dialog box

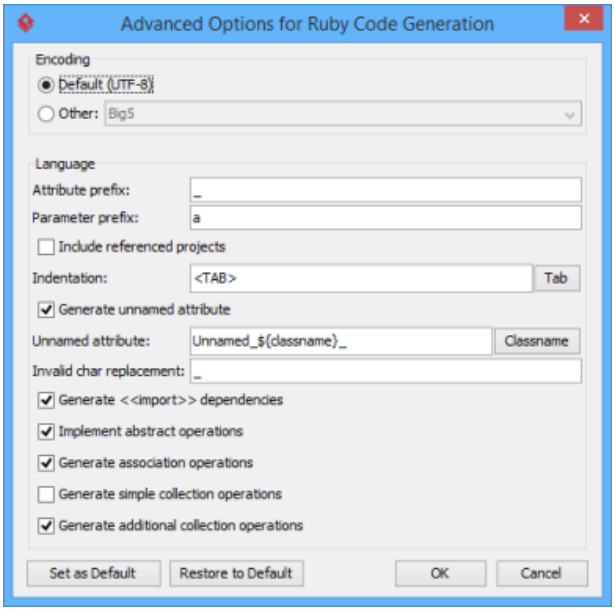
No.	Name	Description

1	Language	The programming language to generate.
2	Model element tree	A list of packages and classes that can be selected for code generation. You must select classes for code generation.
3	General options	Some of the common configurable options are shown here. You can configure them in advanced options.
4	Output path	The folder where you want the code files to be generated.
5	Template directory	Template governs how code will be generated from model to code. You can customize the template to suit your needs, such as to print company specific headers to each code file. If you want to use your own template, provide the template directory here. If you want to keep using the build in template, leave this option unchanged to let Visual Paradigm generate with build in template. To learn more about customization, read the final chapter of this part.
6	Advanced options	Click this button to configure any options related to code generation in a new dialog box.
7	Prompt to confirm overwrite file	If a code file instant generator going to generate is already exist, by checking this option you will be asked whether to overwrite that file or not. If you uncheck this option, it will help you to overwrite the existing file automatically.
8	Output pane	Any warning, error or progress about generation will be printed here.
9	Open output folder	Open the output path with system browser.
10	Preview	Click to preview the code content. It is just a preview and code will not be generated to the output path by previewing.
11	Generate	Click to start generation.
12	Close	Click to close the instant generator.

Description of instant generator dialog box

Generator options

On the **Instant Generator** dialog box you can configure some of the common code options at the right of dialog box. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.



Advanced Options dialog box

Below is a description of available options.

Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .

Generate unnamed attribute	When two classes are associated, checking this option will generate attributes in both classes with each other as type. When unchecked, attributes will not be generated to both of them.
Unnamed attribute	Pattern will be applied when generating name for those attribute without name.
Invalid char replacement	Invalid char refers to characters that will result in a compile error when compiling code. This option is for replacing those invalid characters by given one.
Implement abstract operations	Whether or not to generate operations for implementing abstract operations defined in super class.
Generate association operations	If you check this box, when a role is selected to provide setter/getter, the corresponding operation(s) will be generated for the role's attribute.
Generate simple collection operations	Whether or not to generate setter and getter for accessing attribute of associated class, when getter and setter are checked.
Generate additional collection operations	Whether or not to generate add, remove and to methods for accessing attribute of associated class, when getter and setter are checked.

A description of advanced options

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Customizing code generation

Instant generator allows you to generate programming source code from class models. Basically, the content of the generated code follows the common coding convention of the programming language. There are also advanced options for you to configure some of the specific settings in forming the code, like the use of prefix for attributes and parameters.
Although the built-in way of generating source code can satisfy most of the general needs, you may want to define something more specific. For example, you may need to print a copyright statement at the beginning of the code file, which is not a kind of customization being supported by Instant generator.
Fortunately, the way of how source code will be generated is handled by Apache Velocity engine, a templating engine, and the templates being used are fully opened for customization. In the following sections, we will explain how to customize a template to make the generated code follow your requirement.

Setting up development environment

The template files are put under the **resources/instantgenerator** folder of Visual Paradigm installation directory. It is absolutely alright to edit those files directly. However, it is recommended to setup your own development environment, copy the template files to there to perform further editing. There are two reasons for separating the development environment from Visual Paradigm:

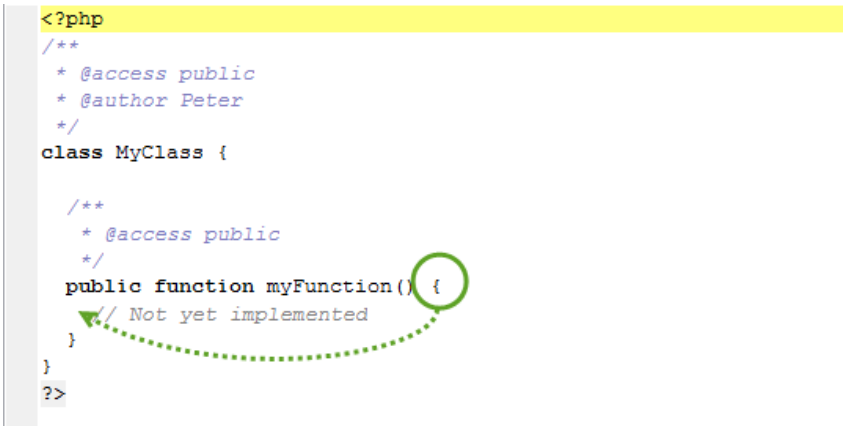
- Avoid the unexpected template removal by un-installing the Visual Paradigm.
- Avoid accidental file replacement by running product updates.

To setup your development environment:

1. Create a folder as working directory.
2. Explore **%Visual-Paradigm-Installation-Directory%/resources/instantgenerator**.
3. You will see a number of sub-folders that have the programming language as their names. Each of them contains the templates files for a specific programming language. Copy the folder(s) of the language(s) you need to customize and paste to the working directory.

Customizing template

By having the text editor and the development environment ready, it's time to get your hand dirty with editing the template. As mentioned before, Instant generator adopted the Apache Velocity engine in generating source code. For those who are interested in knowing how to write templates, please read Velocity's Users' guide at:
http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html .
The following example demonstrates how to edit the PHP code generation template to reposition the brace of operation blocks to a new line.

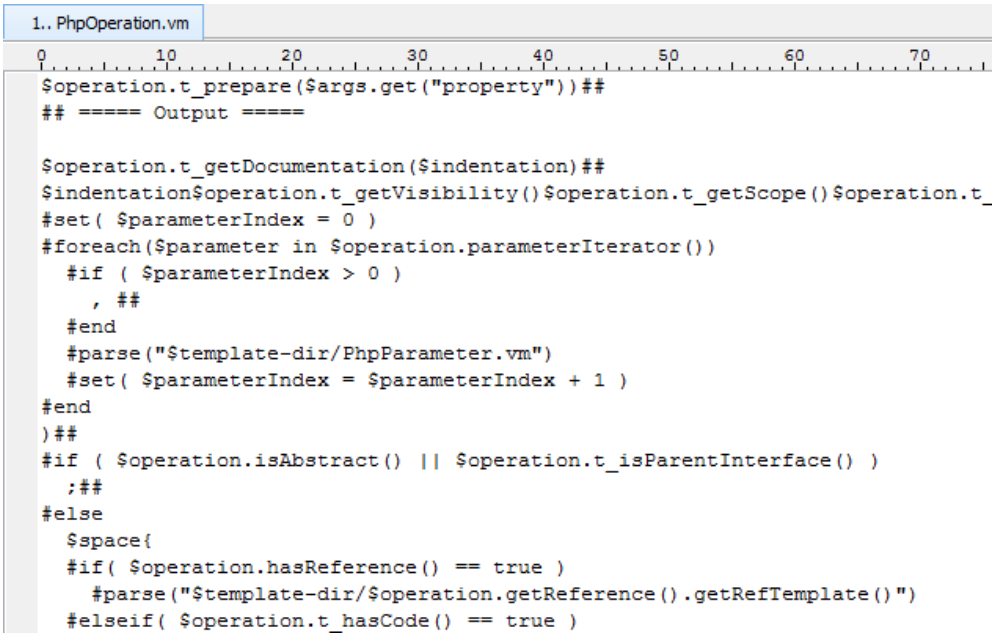


```php
<?php
/**
 * @access public
 * @author Peter
 */
class MyClass {

  /**
   * @access public
   */
  public function myFunction() {
    // Not yet implemented
  }
}
?>
```

Customization of operation in PHP class

1. Open the template you need to edit in text editor.



1.. PhpOperation.vm

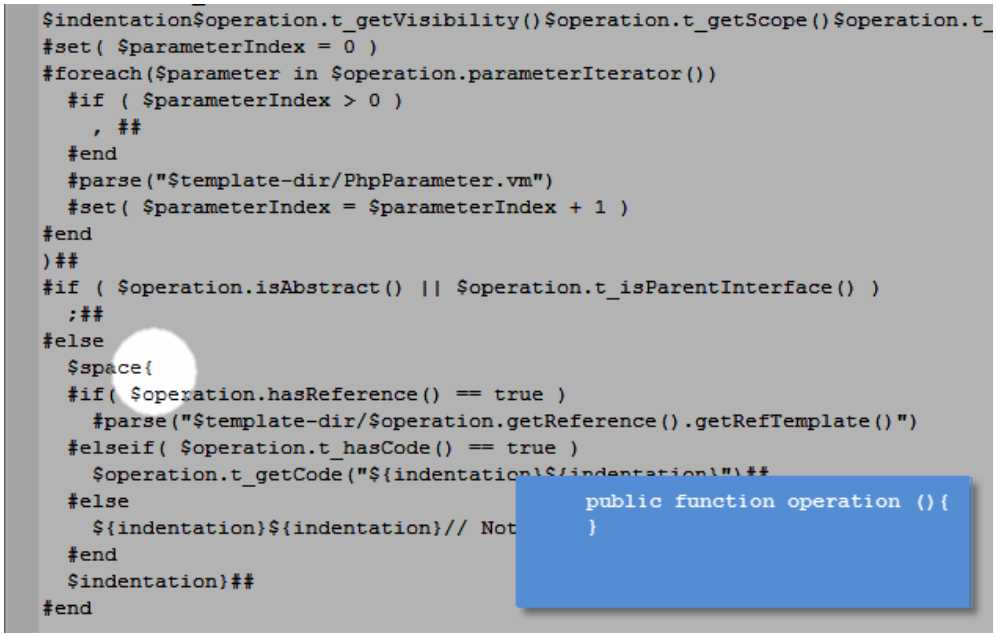
```
$operation.t_prepare($args.get("property"))##
## ===== Output =====

$operation.t_getDocumentation($indentation)##
$indentation$operation.t_getVisibility()$operation.t_getScope()$operation.t_
#set( $parameterIndex = 0 )
#foreach($parameter in $operation.parameterIterator())
  #if ( $parameterIndex > 0 )
    , ##
  #end
  #parse("$template-dir/PhpParameter.vm")
  #set( $parameterIndex = $parameterIndex + 1 )
#end
)##
#if ( $operation.isAbstract() || $operation.t_isParentInterface() )
  ;##
#else
  $space{
  #if( $operation.hasReference() == true )
    #parse("$template-dir/$operation.getReference().getRefTemplate()")
  #elseif( $operation.t_hasCode() == true )
```

Open PhpOperation.vm in text editor

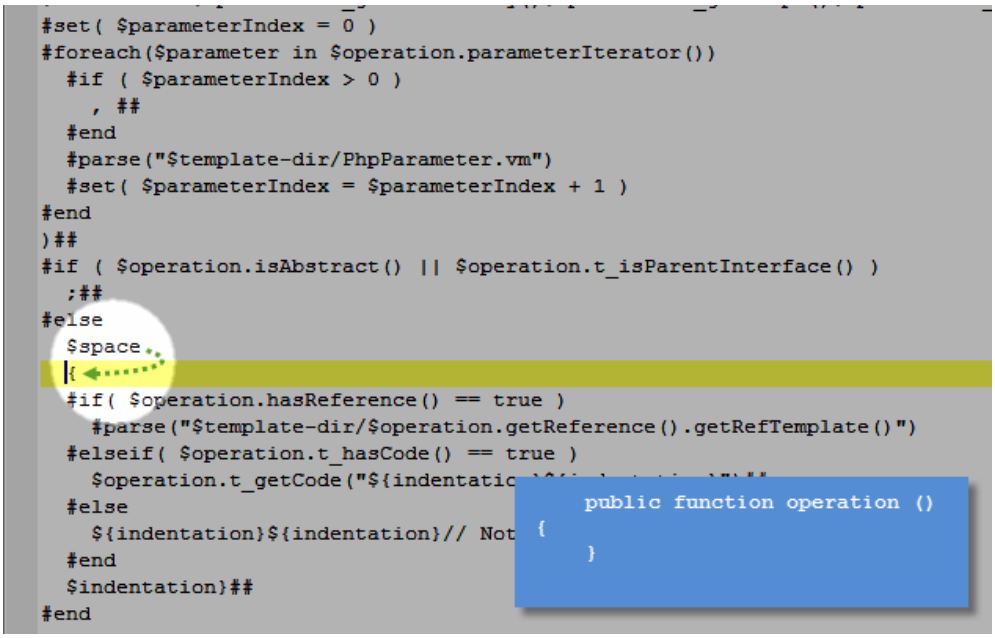
At the beginning, you may find the template a bit complex. But once you start working on it for a while, you'll find the syntax easy to understanding. In fact, it just composes of common programming construct like if-then-else statements, foreach and variables that programmers should find intuitive.

2. Look for the area that you need to edit.



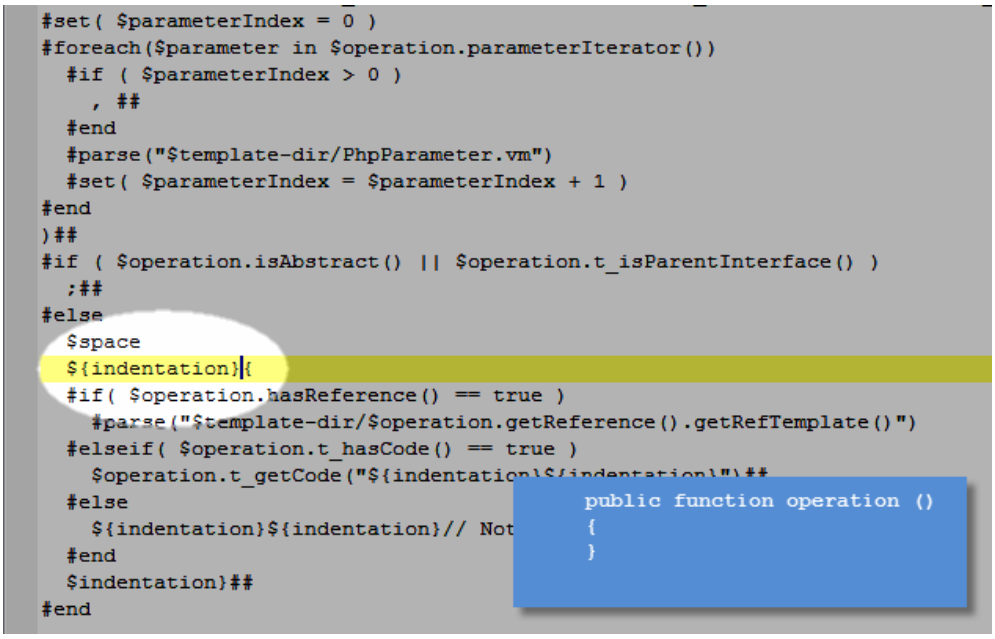
Search for the open branch

3. Make change.



Insert line breaks

4. Add a variable $indentation to indicate the need of printing indentation before the open brace.



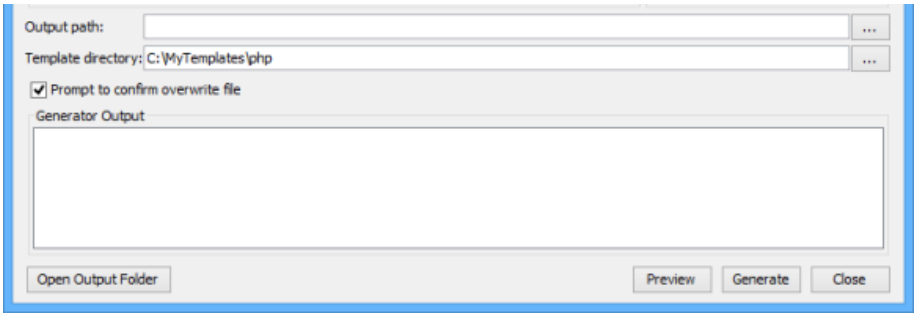
Add variable

5. Save the file.

Generate code with the customized template

To generate code with customized template:

1. In Visual Paradigm, select **Tools > Code > Instant Generator** from the toolbar, then the programming language that have the template customized.
2. Specify the **Template directory** where the customized templates are stored.



Specifying template directory

3. Select the classes to generate. Specify the output path. Click **Generate** to generate code. You may refer to previous chapters for details about instant generator.

List of API calls

The following table lists the available API calls for retrieving data from models.

Class	API	Return Value
Annotation	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue

	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	propertyArray()	Object[]
	propertyAt(int)	AnnotationProperty
	propertyCount()	int
	propertyIterator()	Iterator
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
AnnotationProperty	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getValue()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int

	taggedValueIterator()	Iterator
Association	associationClassArray()	AssociationClass[]
	associationClassAt(int)	AssociationClass
	associationClassCount()	int
	associationClassIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	fromAssociationClassArray()	Object[]
	fromAssociationClassAt(int)	AssociationClass
	fromAssociationClassCount()	int
	fromAssociationClassIterator()	Iterator
	getDocumentation()	String
	getFromElement()	Object
	getFromEnd()	AssociationEnd
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getToElement()	Object
	getToEnd()	AssociationEnd
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isAbstract()	boolean
	isDerived()	boolean
	isFromLinkedProject()	boolean
	isLeaf()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	toAssociationClassArray()	Object[]

	toAssociationClassAt(int)	AssociationClass
	toAssociationClassCount()	int
	toAssociationClassIterator()	Iterator
AssociationClass	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getFromElement()	Object
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getToElement()	Object
	getVisibility()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
AssociationEnd	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getAggregationKind()	String
	getDocumentation()	String
	getMultiplicity()	String
	getName()	String
	getNavigable()	int
	getReferencedAttribute()	Attribute
	getStereotype(String)	Stereotype

	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTypeModifier()	String
	getVisibility()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	isOrdered()	boolean
	isProvideGetterMethod()	boolean
	isProvideSetterMethod()	boolean
	isUnique()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Attribute	annotationArray()	Object[]
	annotationAt(int)	Annotation
	annotationCount()	int
	annotationIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDeclarativeAttribute()	String
	getDocumentation()	String
	getFieldType()	Object
	getInitialValue()	String
	getMetadataTag()	String
	getMultiplicity()	String
	getName()	String
	getScope()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype

	getStorage()	int
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo
	getType()	
	getTypeModifier()	String
	getVisibility()	String
	getXmlSchemaFieldType()	Object
	hasGetter()	boolean
	hasSetter()	boolean
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	hasXmlSchema()	boolean
	isAbstract()	boolean
	isConst()	boolean
	isDefault()	boolean
	isExtern()	boolean
	isFinal()	boolean
	isFromLinkedProject()	boolean
	isHasGetter()	boolean
	isHasSetter()	boolean
	isIndexer()	boolean
	isNew()	boolean
	isOrdered()	boolean
	isOverload()	boolean
	isOverride()	boolean
	isReadonly()	boolean
	isShadow()	boolean
	isTransient()	boolean
	isUnique()	boolean
	isUnsafe()	boolean
	isVirtual()	boolean
	isVisible()	boolean
	isVolatile()	boolean
	isWithEvent()	boolean
	propertyParameterArray()	Object[]
	propertyParameterAt(int)	Parameter
	propertyParameterCount()	int

	propertyParameterIterator()	Iterator
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
AttributeType	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getFixed()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getUse()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Class	annotationArray()	Object[]
	annotationAt(int)	Annotation
	annotationCount()	int
	annotationIterator()	Iterator
	associationArray()	Association[]
	associationAt(int)	Association

	associationClassArray()	AssociationClass[]
	associationClassAt(int)	AssociationClass
	associationClassCount()	int
	associationClassIterator()	Iterator
	associationCount()	int
	associationIterator()	Iterator
	attributeArray()	Attribute[]
	attributeAt(int)	Attribute
	attributeCount()	int
	attributeIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	containmentClassArray()	Class[]
	containmentClassAt(int)	Class
	containmentClassCount()	int
	containmentClassIterator()	Iterator
	fromAssociationArray()	Object[]
	fromAssociationAt(int)	Association
	fromAssociationClassArray()	Object[]
	fromAssociationClassAt(int)	AssociationClass
	fromAssociationClassCount()	int
	fromAssociationClassIterator()	Iterator
	fromAssociationCount()	int
	fromAssociationIterator()	Iterator
	generalizationArray()	Generalization[]
	generalizationAt(int)	Generalization
	generalizationCount()	int
	generalizationIterator()	Iterator
	getDeclarativeAttribute()	String
	getDocumentation()	String
	getManageType()	int
	getMetadataTag()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo

	getType()	Object
	getTypeModifier()	String
	getVisibility()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isAbstract()	boolean
	isActive()	boolean
	isFinal()	boolean
	isFromLinkedProject()	boolean
	isInterface()	boolean
	isLeaf()	boolean
	isNew()	boolean
	isNotInheritable()	boolean
	isRoot()	boolean
	isSealed()	boolean
	isShadow()	boolean
	isStatic()	boolean
	isStereotypeInterface()	boolean
	isStereotypeTypedef()	boolean
	isTypedef()	boolean
	operationArray()	Operation[]
	operationAt(int)	Operation
	operationCount()	int
	operationIterator()	Iterator
	realizationArray()	Realization[]
	realizationAt(int)	Realization
	realizationClassArray()	Object[]
	realizationClassAt(int)	Class
	realizationClassCount()	int
	realizationClassIterator()	Iterator
	realizationCount()	int
	realizationIterator()	Iterator
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue

	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateParameterArray()	TemplateParameter[]
	templateParameterAt(int)	TemplateParameter
	templateParameterCount()	int
	templateParameterIterator()	Iterator
	toAssociationArray()	Object[]
	toAssociationAt(int)	Association
	toAssociationClassArray()	Object[]
	toAssociationClassAt(int)	AssociationClass
	toAssociationClassCount()	int
	toAssociationClassIterator()	Iterator
	toAssociationCount()	int
	toAssociationIterator()	Iterator
Comment	commentCount()	int
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentIterator()	Iterator
	getAuthor()	String
	getContent()	String
	getDateTime()	String
	getDocumentation()	String
	getName()	String
	getSummary()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeCount()	int
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeIterator()	Iterator
	taggedValueCount()	int
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue

	taggedValueIterator()	Iterator
DataType	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateParameterArray()	Object[]
	templateParameterAt(int)	TemplateParameter
	templateParameterCount()	int
	templateParameterIterator()	Iterator
ElementType	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getBlock()	String
	getDocumentation()	String
	getForm()	String
	getName()	String
	getNillable()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue

	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Generalization	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getFromElement()	Object
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo
	getToElement()	Object
	getVisibility()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	isSubstitutable()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue

	taggedValueCount()	int
	taggedValueIterator()	Iterator
ImplModel	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getCode()	String
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Object	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean

	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Operation	annotationArray()	Object[]
	annotationAt(int)	Annotation
	annotationCount()	int
	annotationIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getAlias()	String
	getCharset()	int
	getDeclarativeAttribute()	String
	getDllName()	String
	getDocumentation()	String
	getImplModel()	ImplModel
	getMetadataTag()	String
	getMethodKind()	int
	getName()	String
	getOperatorType()	int
	getProcedureName()	String
	getReturnType()	Object
	getReturnTypeDocumentation()	String
	getReturnTypeModifier()	String
	getScope()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo
	getVisibility()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean

	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isAbstract()	boolean
	isConst()	boolean
	isDeclare()	boolean
	isDelegate()	boolean
	isExtern()	boolean
	isFinal()	boolean
	isFriend()	boolean
	isFromLinkedProject()	boolean
	isInline()	boolean
	isNative()	boolean
	isNew()	boolean
	isNotOverridable()	boolean
	isOverload()	boolean
	isOverridable()	boolean
	isOverride()	boolean
	isQuery()	boolean
	isReturnTypeConst()	boolean
	isSealed()	boolean
	isShadow()	boolean
	isSynchronized()	boolean
	isUnsafe()	boolean
	isVirtual()	boolean
	isVisible()	boolean
	parameterArray()	Object[]
	parameterAt(int)	Parameter
	parameterCount()	int
	parameterIterator()	Iterator
	postConditionArray()	Object[]
	postConditionAt(int)	Text
	postConditionCount()	int
	postConditionIterator()	Iterator
	preConditionArray()	Object[]
	preConditionAt(int)	Text
	preConditionCount()	int
	preConditionIterator()	Iterator
	raisedExceptionArray()	Object[]
	raisedExceptionAt(int)	Object
	raisedExceptionCount()	int

	raisedExceptionIterator()	Iterator
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateParameterArray()	Object[]
	templateParameterAt(int)	TemplateParameter
	templateParameterCount()	int
	templateParameterIterator()	Iterator
Package	classArray()	Class[]
	classAt(int)	Class
	classCount()	int
	classIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	packageArray()	Object[]
	packageAt(int)	Package
	packageCount()	int
	packageIterator()	Iterator
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator

	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateParameterArray()	Object[]
	templateParameterAt(int)	TemplateParameter
	templateParameterCount()	int
	templateParameterIterator()	Iterator
Parameter	annotationArray()	Object[]
	annotationAt(int)	Annotation
	annotationCount()	int
	annotationIterator()	Iterator
	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDeclarativeAttribute()	String
	getDefaultValue()	String
	getDirection()	String
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo
	getType()	Object
	getTypeModifier()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isConst()	boolean
	isFinal()	boolean
	isFromLinkedProject()	boolean
	isOptional()	boolean
	isParamArray()	boolean
	isParams()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype

	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Realization	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getFromElement()	Object
	getMapping()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTemplateTypeBindInfo()	TemplateTypeBindInfo
	getToElement()	Object
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Stereotype	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype

	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
TaggedValue	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getType()	int
	getValue()	Object
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int

	taggedValueIterator()	Iterator
TemplateParameter	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDefaultValue()	String
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateTypeBindInfoArray()	Object[]
	templateTypeBindInfoAt(int)	TemplateTypeBindInfo
	templateTypeBindInfoCount()	int
	templateTypeBindInfoIterator()	Iterator
	typeArray()	Object[]
	typeAt(int)	Object
	typeCount()	int
	typeIterator()	Iterator
	typeModifierArray()	Object[]
	typeModifierAt(int)	String
	typeModifierCount()	int
	typeModifierIterator()	Iterator
TemplateTypeBindDetails	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int

	commentIterator()	Iterator
	getArguments()	TemplateTypeBindInfo
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getWildcard()	int
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
TemplateTypeBindInfo	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	detailsArray()	Object[]
	detailsAt(int)	TemplateTypeBindDetails
	detailsCount()	int
	detailsIterator()	Iterator
	getBindedType()	Object
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	getTypeModifier()	String
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean

	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
Text	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String
	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
TextType	commentArray()	Comment[]
	commentAt(int)	Comment
	commentCount()	int
	commentIterator()	Iterator
	getDocumentation()	String
	getName()	String

	getStereotype(String)	Stereotype
	getStereotypeIgnoreCase(String)	Stereotype
	getTaggedValue(String)	TaggedValue
	getTaggedValueIgnoreCase(String)	TaggedValue
	hasStereotype(String)	boolean
	hasStereotypeIgnoreCase(String)	boolean
	hasTaggedValue(String)	boolean
	hasTaggedValueIgnoreCase(String)	boolean
	isFromLinkedProject()	boolean
	stereotypeArray()	Stereotype[]
	stereotypeAt(int)	Stereotype
	stereotypeCount()	int
	stereotypeIterator()	Iterator
	taggedValueArray()	TaggedValue[]
	taggedValueAt(int)	TaggedValue
	taggedValueCount()	int
	taggedValueIterator()	Iterator
	templateParameterArray()	Object[]
	templateParameterAt(int)	TemplateParameter
	templateParameterCount()	int
	templateParameterIterator()	Iterator

A list of API calls

Velocity syntax

The following lists the syntax that of statements that can be used in the template.

```
## ===== If =====
#if(...)
...
#end
## ===== If-then-Else =====
#if(...)
...
#else
...
#end
## ===== For-each =====
#foreach
...
#end
## ===== Continue with the template defined in (...) at the point where the call is made =====
#parse(...)
#set(...)
## ===== Comment =====
## ...
## ===== Comment =====
#* ... *#
## ===== Variable=====
${...}
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

- Contact us if you need any help or have any suggestion

Java Round-Trip

Round-trip engineering refers to the synchronization between source code in Java project and UML class model in VP-UML's modeling environment. In this chapter, you will learn how to perform round-trip engineering in Visual Paradigm.

Generate/Update Java code

To produce or update source files from UML class model.

Generate/Update UML classes from Java code

To produce or update UML class model from source files.

Generate/Update Java code

Round-trip engineering is the ability to generate model from source code and generate source code from model, and keep them synchronized. You can make use of round-trip engineering to keep your implementation model and source code up-to-date, so as to produce up-to-date description on your model.

Generating/Updating code from whole project

You can generate Java code from all classes in current project. To generate code from project:

1. Select **Tools > Code > Generate Java Code...** from the toolbar.
2. In the **Generate Code** dialog box, specify the mapping between model and source path. Model is a UML element that acts as a container of other elements. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.

The mappings between models and source paths are defined

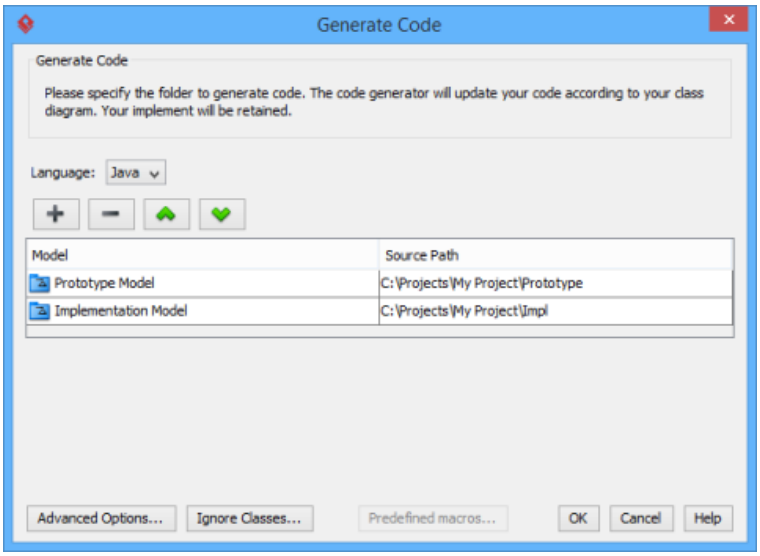
3. Optionally, configure the advanced code generation options by clicking **Advanced Options...**. Read the section *Advanced Options* in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Description in model elements is generated as comment in code.

Generating/Updating code from opening class diagram

You can generate Java code from an opening class diagram that contains the class(es) you want to generate code. To generate code from class diagram:

1. Right click on the class diagram background and select **Utilities > Java Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.



The mappings between models and source paths are defined

NOTE: If you have generated code for once, the **Generate Code** dialog box will not appear next time when you generate/update code, for any diagram. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Description in model elements is generated as comment in code.

Generating/Updating code from chosen classes

You can generate Java code from specific class or classes. To generate code from class/classes:

1. Select the class(es) and right click on them, then select **Java Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.

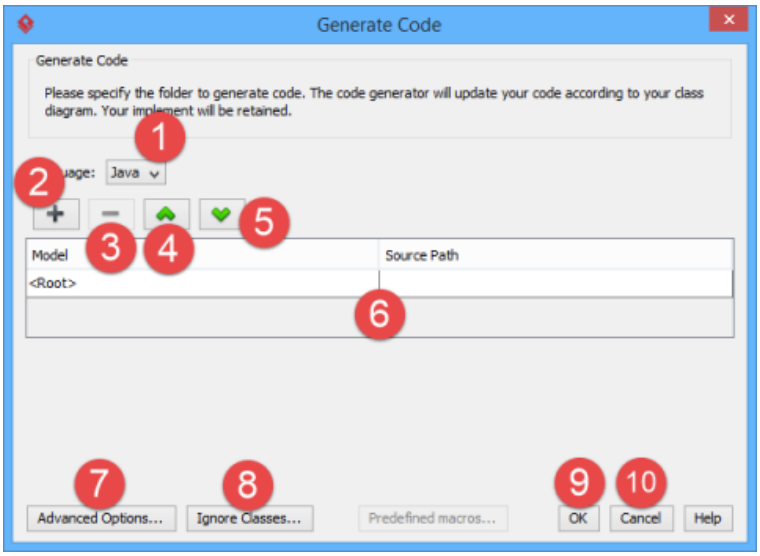
The mappings between models and source paths are defined

NOTE: If you have generated code for once, the **Generate Code** dialog box will not appear next time when you generate/update code, for any class selection. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Description in model elements is generated as comment in code.

An overview of Generate Code dialog box



*An overview of **Generate Code** dialog box*

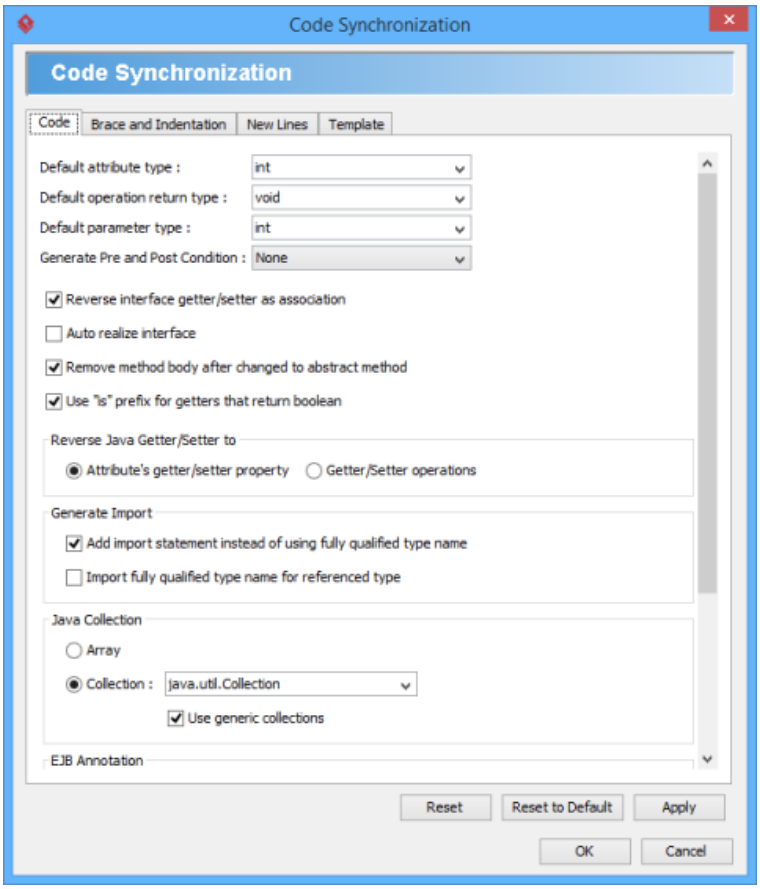
No.	Name	Description
1	Language	The programming language of the source code to generate.
2	Add model-to-source-path mapping	Click to add a new mapping between UML model and the source path where code will be generated to.
3	Remove model-to-source-path mapping	Click to remove chosen model-to-source-path mapping.
4	Move model-to-source-path mapping up	Click to move chosen model-to-source-path mapping one item upward.
5	Move model-to-source-path mapping down	Click to move chosen model-to-source-path mapping one item downward.
6	Model-to-source-path mapping	A list of mapping between UML model and source path.
7	Advanced options	Click to configure advanced code generation options. For details, read the section *Advanced Options* in this chapter.
8	Ignore classes	Click to organize the ignore list of classes to ignore in code generation. For details, read the section *To ignore classes in generation* in this chapter.
9	OK	Click to start generation.
10	Cancel	Click to close the **Generate Code** dialog without generating code.

*A description of **Generate Code** dialog box*

Advanced options

You can configure the advanced options for more control of the code by clicking the **Advanced Options...** button in **Generate Code** dialog box. In the **Code Synchronization** dialog box popped up, there are four categories (tabs) of settings you can configure. Below is a description.

Code



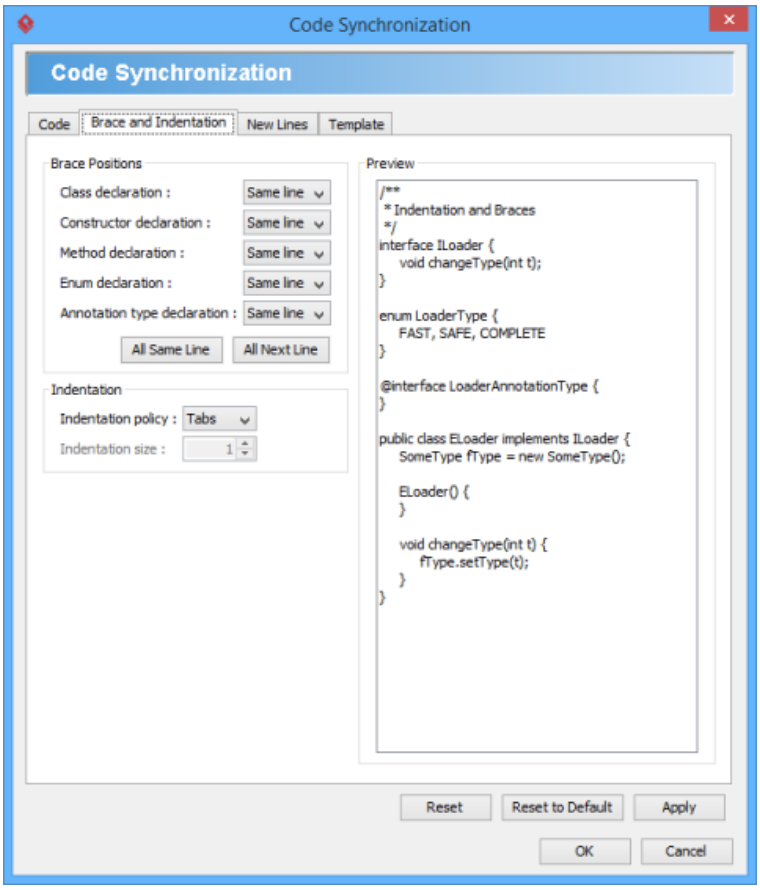
Code configuration

Option	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified
Default parameter type	(default int) Type that will be assigned to Parameter upon code generation when type is unspecified
Auto realize interface	(default false) Generate operations defined in interface in sub-classes
Remove method body after changed to abstract method	(default true) When an operation is set from non-abstract to abstract, updating code will remove the related method's body
Use "is" prefix for getters that return boolean	(default true) Generate getter's name as isXXXX() for getters that return a boolean value
Add import statement instead of using fully qualified type name	(default true) Add import statement for referencing classes in another package/namespace instead of using fully qualified name inline
Import fully qualified type name for referenced type	(default false) Use fully qualified type name in import statements instead of using wildcard character * to represent importing all classes in package
Java Collection	• Array - Generate one-to-many relationship as array • Collection - (default) Generate one-to-many relationship as collection
Use generic collections	(default true) Allow to use generic collection
Generate annotation on	• Property method - Generate annotation on property method • Field - Generate annotation on field

Generate annotation in code convention	(default true) Generate annotation in code convention
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files • Other -Specify an encoding for source files

A description of code configuration

Brace and Indentation

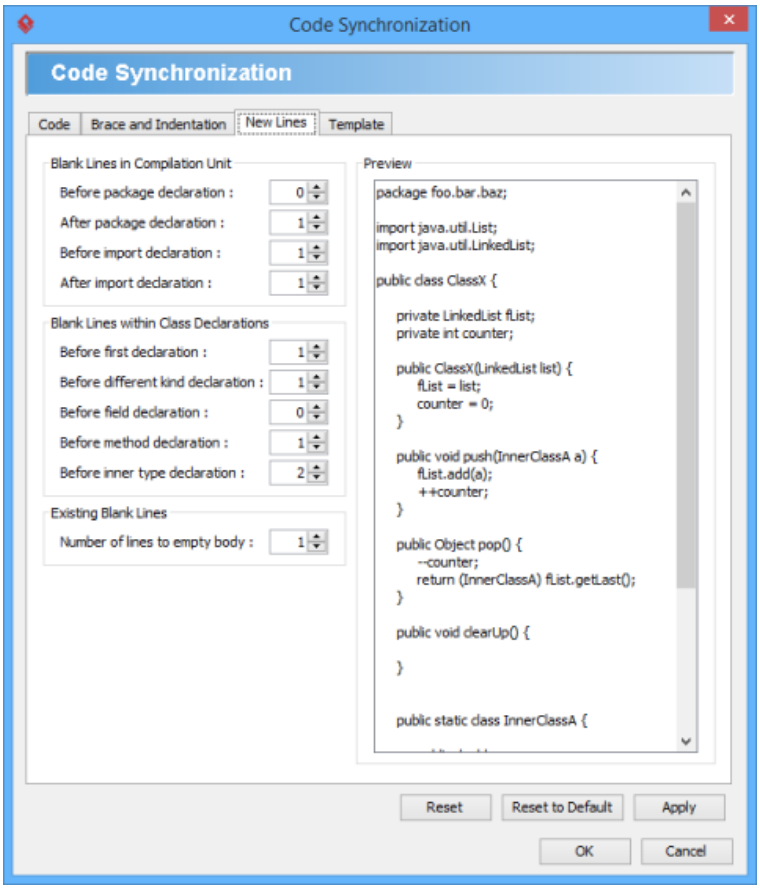


Brace and indentation configuration

Option	Description
Class declaration	• Same line - (default) Brace for class declaration appear at the same line as the declaration • Next line - Brace for class declaration appear at the line after the declaration
Constructor declaration	• Same line - (default) Brace for constructor appear at the same line as the declaration • Next line - Brace for constructor appear at the line after the declaration
Method declaration	• Same line - (default) Brace for method appear at the same line as the declaration • Next line - Brace for method appear at the line after the declaration
Enum declaration	• Same line - (default) Brace for enumeration appear at the same line as the declaration • Next line - Brace for enumeration tor appear at the line after the declaration
Annotation type declaration	• Same line - (default) Brace for annotation type appear at the same line as the declaration • Next line - Brace for annotation type appear at the line after the declaration
Indentation policy	• Tabs - (default) Use a tab of space as indentation • Spaces - Use spaces as indentation. The number of spaces can be defined below
Indentation size	The number of spaces to indent

A description of brace and indentation configuration

New Lines

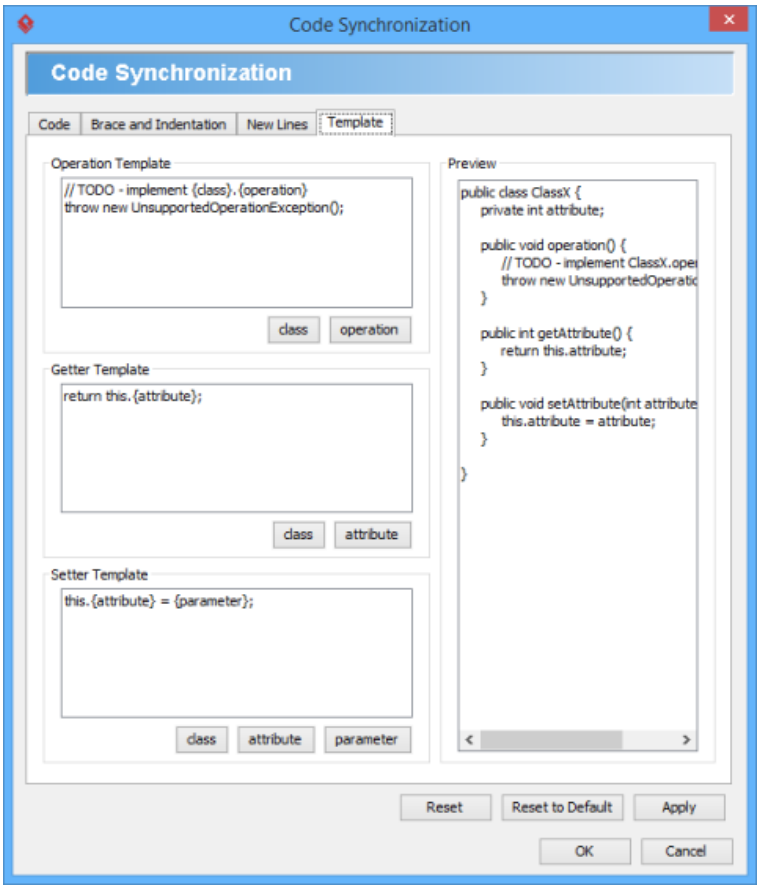


New lines configuration

Option	Description
Before package declaration	Number of blank lines to appear before Package declaration
After package declaration	Number of blank lines to appear after Package declaration
Before import declaration	Number of blank lines to appear before import statements
After import declaration	Number of blank lines to appear after import statements
Before first declaration	Number of blank lines to appear before the first declaration within Class declarations
Before different kind declaration	Number of blank lines to appear before a different kind of declaration
Before field declaration	Number of blank lines to appear before field declaration
Before method declaration	Number of blank lines to appear before method declaration
Before inner type declaration	Number of blank lines to appear before inner type declaration
Number of lines to empty body	Number of blank lines to appear in empty method body

A description of new lines configuration

Template



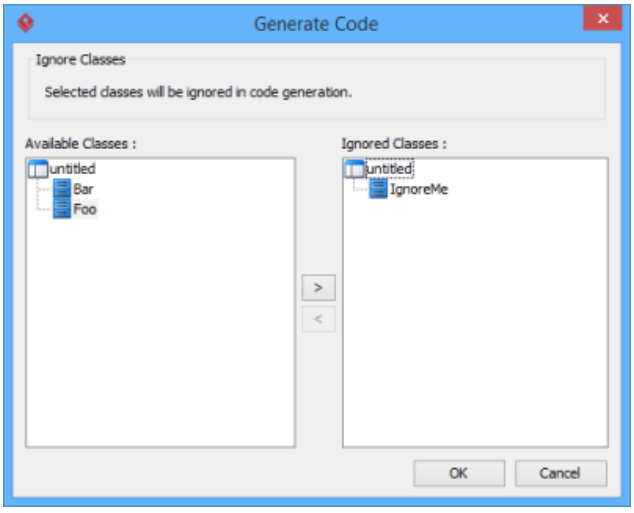
Template configuration

Option	Description
Operation Template	Defines a template of method body that will be applied when generating operations.
Getter Template	Defines a template of getter that will be applied when generating getter methods. Getter will be generated to attribute stereotyped as Property, or with property getter selected.
Setter Template	Defines a template of setter that will be applied when generating setter methods. Setter will be generated to attribute stereotyped as Property, or with property setter selected.

A description of template configuration

To ignore classes in generation

You can make certain UML class not to generate code against code generation by ignoring them. To ignore class(es), click **Ignore Classes...** in **Generate Code** dialog box. In the second Generate Code dialog box that popped up, select the class(es) to ignore and click **>** to move them to the ignore list. Click **OK** to confirm.



The class IgnoreMe is ignored

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

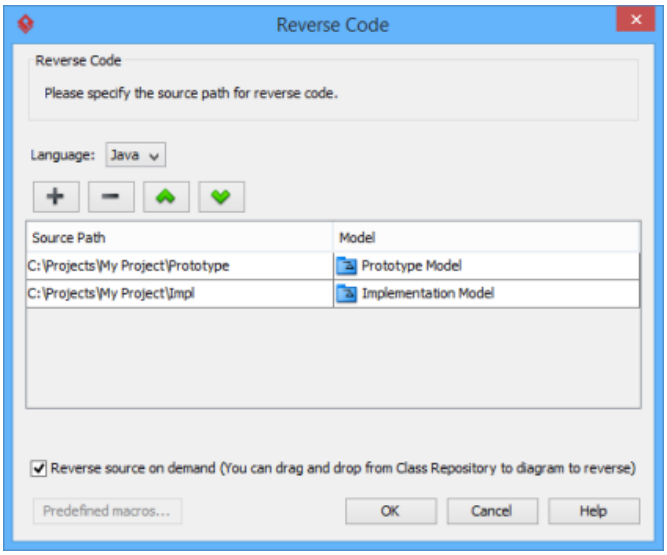
Generate/Update UML classes from Java code

Round-trip engineering is the ability to generate model from source code and generate source code from model, and keep them synchronized. You can make use of round-trip engineering to keep your implementation model and source code up-to-date, so as to produce up-to-date description on your model.

Generate/Update UML classes from code

You can produce UML classes from source code, or to update from code all the reversed UML classes in project. To do this:

1. Select **Tools > Code > Reverse Java Code...** from the toolbar.
2. In the **Reverse Code** dialog box, specify the mapping between source path and model. Model is a UML element that acts as a container of other elements. You can place the UML classes to be produced to specific model for better categorization. For example, you may create a Prototype model and an Implementation model for storing classes developed in prototype and implementation phrases respectively. Once a mapping is defined, round-trip engineering will be performed between the model and path as defined. You can add multiple Source-path-to-model mapping by pressing the **+** button. If you do not use model to structure your project, keep model to be **<root>**.



The mappings between source paths and model are defined

3. By default, an on-demand reverse engineering will be carried out, which means to form indexes to the added path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below. If you want to carry out actual reverse engineering, uncheck **Reverse source on demand**.
4. Click **OK** to proceed with reversal.

Updating UML classes on a class diagram from code

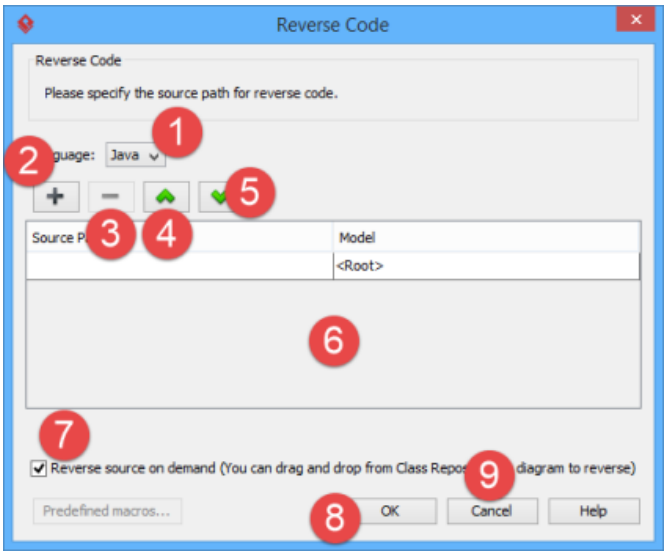
Once you have performed round-trip engineering for once, you can update UML class(es) on a diagram from source code for reflecting the changes made in code. To update, right click on the background of the class diagram for update and select **Utilities > Java Round-trip > Reverse Code** from the popup menu.

> **NOTE:** In order to trigger this function, make sure you have performed round-trip engineering at least for once and the diagram has at least one class.

Updating specific UML classes from code

Once you have performed round-trip engineering for once, you can update specific UML class(es) from source code for reflecting the changes made on that particular class(es) To update, select in class diagram the UML class(es) you want to update. Right click on them and select **Java Round-trip > Reverse Code** from the popup menu.

An overview of Reverse Code dialog box



*An overview of **Reverse Code** dialog box*

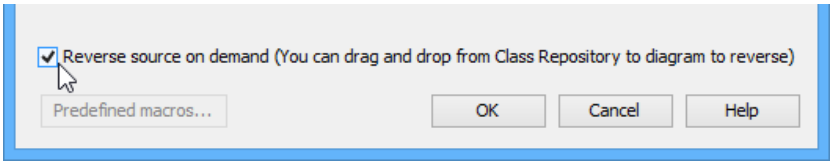
No.	Name	Description
1	Language	The programming language of the source code to reverse.
2	Add source-path-to-model mapping	Click to add a new mapping between source path where code will be reversed from and UML model.
3	Remove source-path-to-model mapping	Click to remove chosen source-path-to-model mapping.
4	Move source-path-to-model mapping up	Click to move chosen source-path-to-model mapping one item upward.
5	Move source-path-to-model mapping down	Click to move chosen source-path-to-model mapping one item downward.
6	Model-to-source-path mapping	A list of mapping between UML model and source path.
7	Reverse source on-demand	By checking, this means to form indexes to the source path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below.
8	OK	Click to start reversal.
9	Cancel	Click to close the **Reverse Code** dialog without reversing code.

*A description of **Reverse Code** dialog box*

On-demand reverse engineering

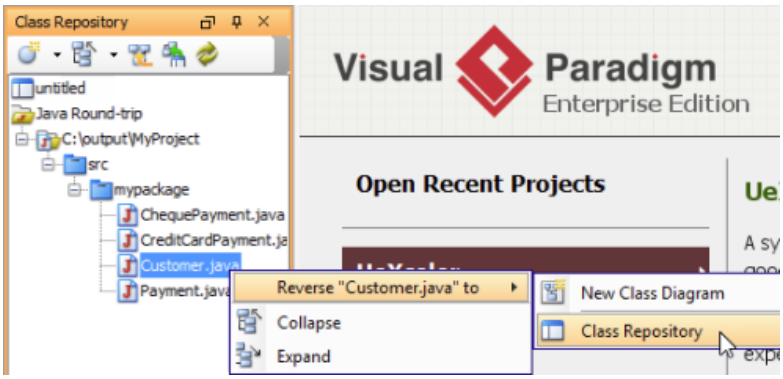
Consider if you have a project that contains million of Java source file, and now you want to re-develop just a few classes in it. If you try to reverse the whole project it will take you a long time to complete the reverse due to the amount of classes (and relationships) are just too many. With on-demand reverse engineering, you will reverse the sources as indexes, and obtain an index tree in class repository. No actual UML classes will be reversed until you trigger the reverse manually. This reduces the processing time significantly.

To perform on-demand reverse engineering, make sure the option Reverse source on demand is checked in the **Reverse Code** dialog box.



*The option **Reverse source on demand** that appear in reverse dialog box*

When finished reverse, you can lookup the index tree in class repository. Then, right click on the class you want to reverse and select **Reverse Resources to** where **Resources** are the classes you have selected, and select either **New Class Diagram** or **Class Repository** from popup menu. Both options will result in reversing the selection to UML classes, while the option **New Class Diagram** will create a class diagram and place the classes in it.



Reverse a java source file from index tree

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

C++ Round-Trip

Round-trip engineering refers to the synchronization between source code in Java project and UML class model in Visual Paradigm's modeling environment. In this chapter, you will learn how to perform round-trip engineering in Visual Paradigm.

Generate/Update C++ code

To produce or update source files from UML class model.

Generate/Update UML classes from C++ code

To produce or update UML class model from source files.

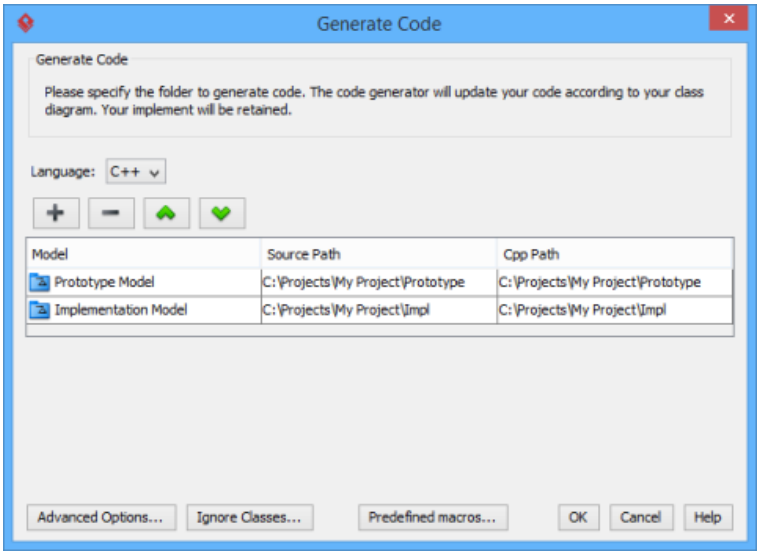
Generate/Update C++ code

Round-trip engineering is the ability to generate model from source code and generate source code from model and keep them synchronized. You can make use of round-trip engineering to keep your implementation model and source code up-to-date, so as to produce up-to-date description on your model.

Generating/Updating code from whole project

You can generate C++ code from all classes in current project. To generate code from project:

1. Select **Tools > Code > Generate Java Code...** from the toolbar. .
2. Select **C++** as the **Language**.
3. In the **Generate Code** dialog box, specify the mapping between model and source path. Model is a UML element that acts as a container of other elements. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.



The mappings between models and source paths are defined

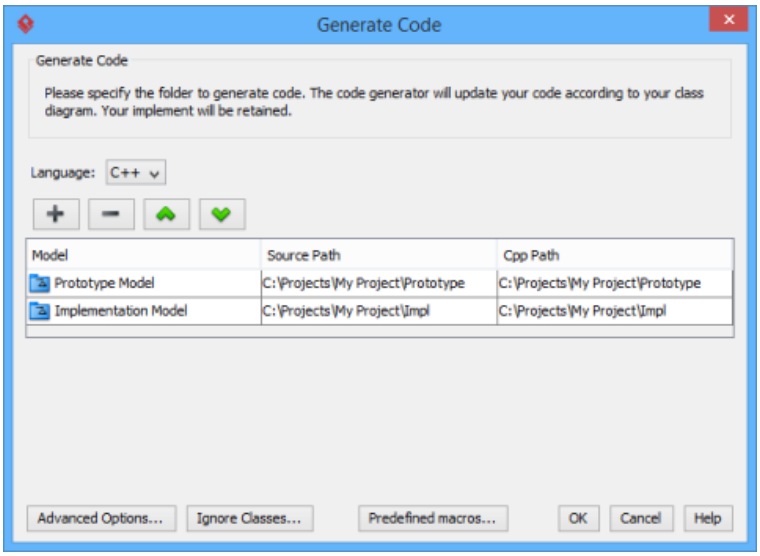
4. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
5. Click **OK** to proceed with generation.

NOTE: Description in model elements is generated as comment in code

Generating/Updating code from opening class diagram

You can generate C++ code from an opening class diagram that contains the class(es) you want to generate code. To generate code from class diagram:

1. Right click on the class diagram background and select **Utilities > C++ Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.



The mappings between models and source paths are defined

> **NOTE:** If you have generated code for once, the **Generate Code** dialog box will not appear next time when you generate/update code, for any diagram. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

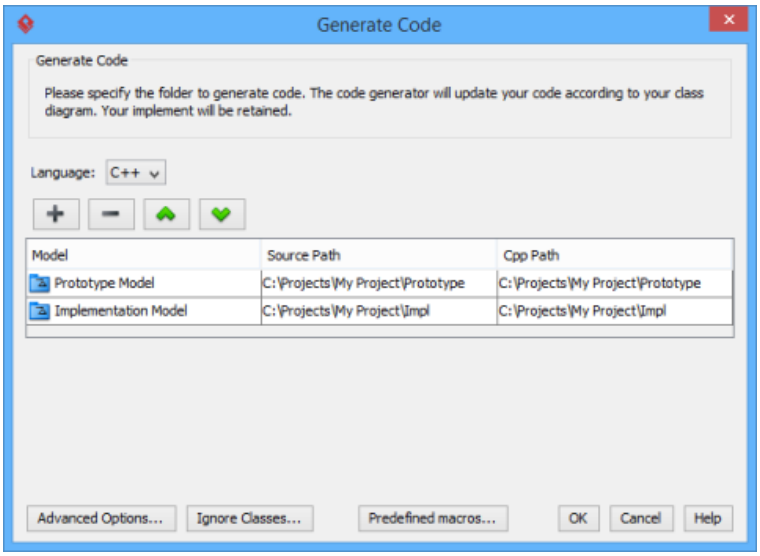
3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

> **NOTE:** Description in model elements is generated as comment in code

Generating/Updating code from chosen classes

You can generate C++ code from specific class or classes. To generate code from class/classes:

1. Select the class(es) and right click on them, then select **C++ Round-trip > Generate Code** from the popup menu.
2. In the **Generate Code** dialog box, specify the source path where you want the code to be generated. Model is a UML element that acts as a container of other elements. Notice that source path is set for model, not for diagram. Classes and packages under a model will be generated to the mapped source path. You can add multiple model-to-source-path mapping by pressing the **+** button. If you are not using model to structure your project, or if you want to generate all classes in project to the same folder, regardless of their parent model, keep model to be **<root>**.



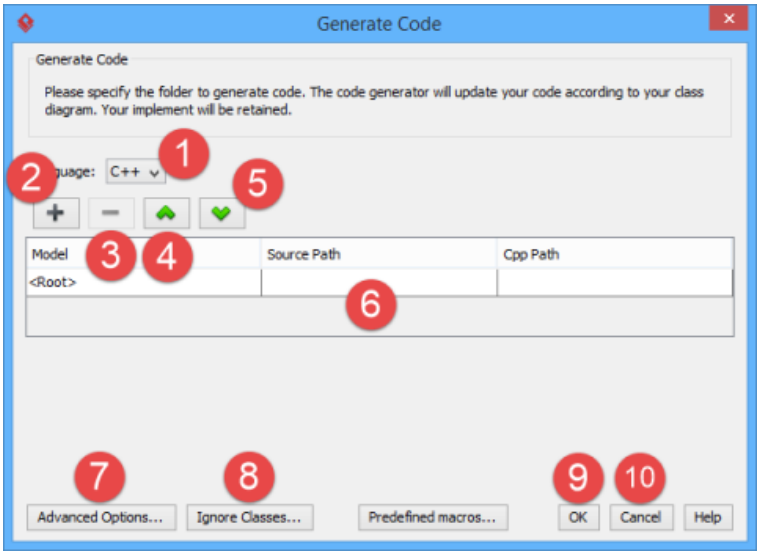
The mappings between models and source paths are defined

> **NOTE:** If you have generated code for once, the **Generate Code** dialog box will not appear next time when you generate/update code, for any class selection. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Description in model elements is generated as comment in code

An overview of Generate Code dialog box



*An overview of **Generate Code** dialog box*

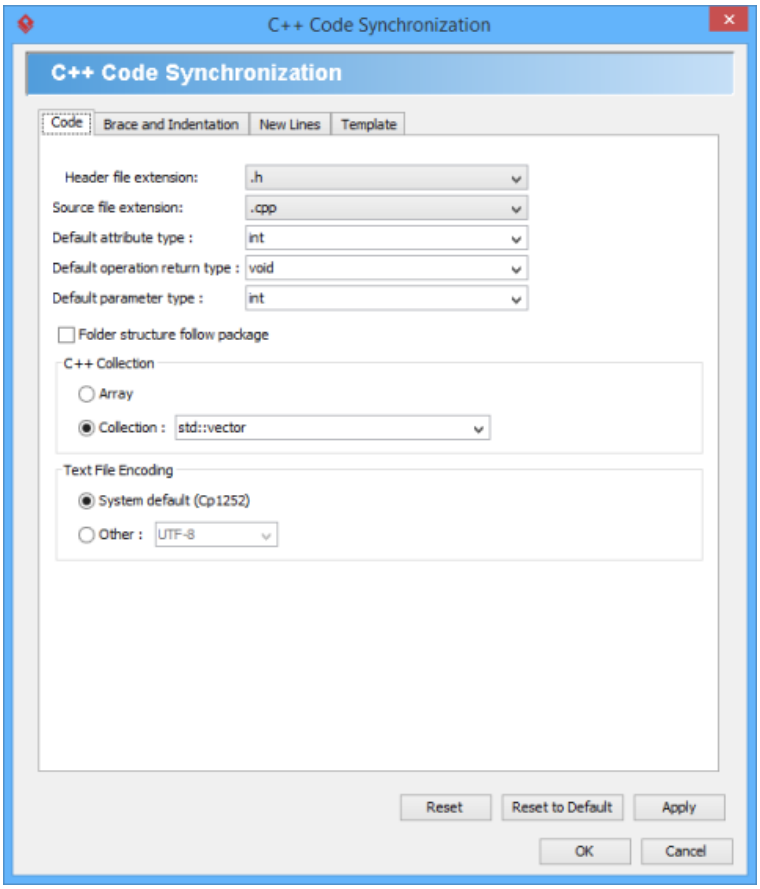
No.	Name	Description
1	Language	The programming language of the source code to generate.
2	Add model-to-source-path mapping	Click to add a new mapping between UML model and the source path where code will be generated to.
3	Remove model-to-source-path mapping	Click to remove chosen model-to-source-path mapping.
4	Move model-to-source-path mapping up	Click to move chosen model-to-source-path mapping one item upward.
5	Move model-to-source-path mapping down	Click to move chosen model-to-source-path mapping one item downward.
6	Model-to-source-path mapping	A list of mapping between UML model and source path.
7	Advanced options	Click to configure advanced code generation options. For details, read the section *Advanced Options* in this chapter.
8	Ignore classes	Click to organize the ignore list of classes to ignore in code generation. For details, read the section *To ignore classes in generation* in this chapter.
9	OK	Click to start generation.
10	Cancel	Click to close the **Generate Code** dialog without generating code.

*A description of **Generate Code** dialog box*

Advanced options

You can configure the advanced options for more control of the code by clicking the **Advanced Options...** button in **Generate Code** dialog box. In the **Code Synchronization** dialog box popped up, there are four categories (tabs) of settings you can configure. Below is a description.

Code

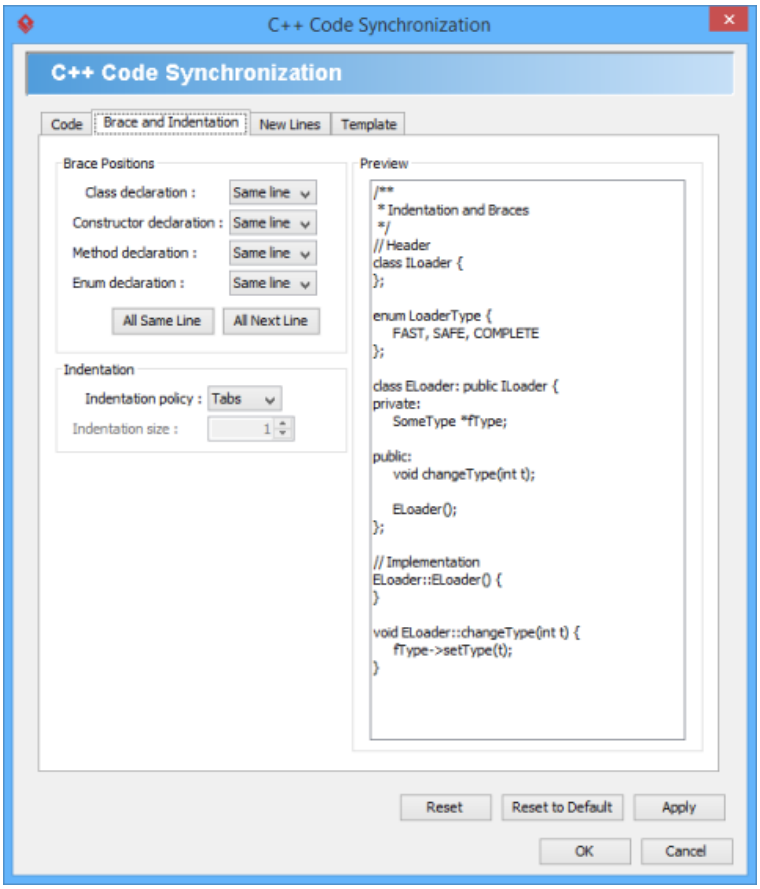


Code configuration

Option	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified
Default parameter type	(default int) Type that will be assigned to Parameter upon code generation when type is unspecified
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files • Other -Specify an encoding for source files

A description of code configuration

Brace and Indentation

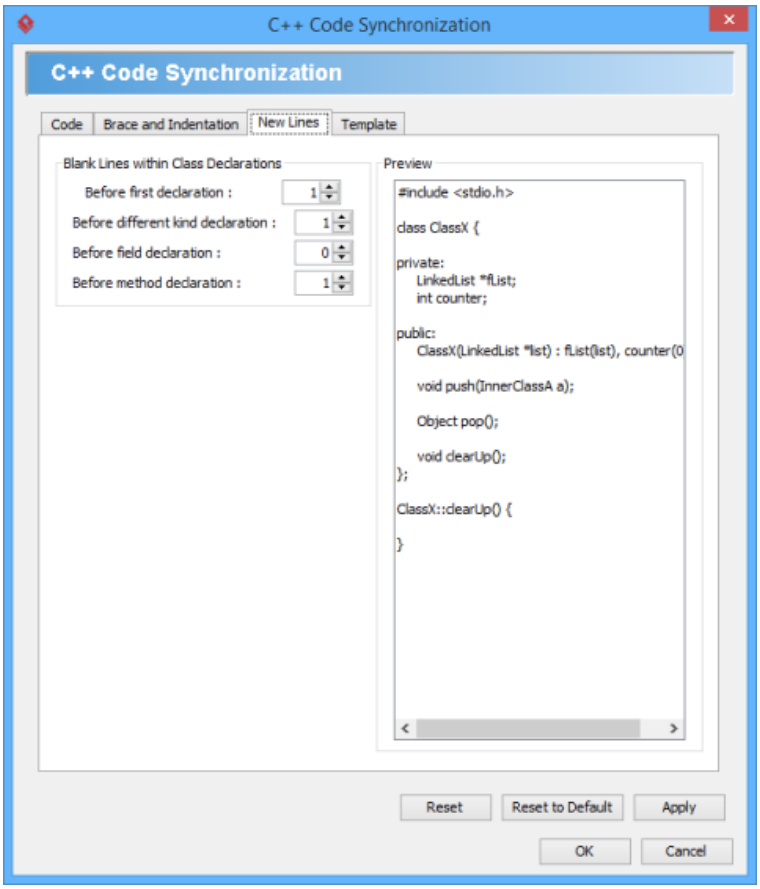


Brace and indentation configuration

Option	Description
Class declaration	• Same line - (default) Brace for class declaration appear at the same line as the declaration • Next line - Brace for class declaration appear at the line after the declaration
Constructor declaration	• Same line - (default) Brace for constructor appear at the same line as the declaration • Next line - Brace for constructor appear at the line after the declaration
Method declaration	• Same line - (default) Brace for method appear at the same line as the declaration • Next line - Brace for method appear at the line after the declaration
Enum declaration	• Same line - (default) Brace for enumeration appear at the same line as the declaration • Next line - Brace for enumeration tor appear at the line after the declaration
Indentation policy	• Tabs - (default) Use a tab of space as indentation • Spaces - Use spaces as indentation. The number of spaces can be defined below
Indentation size	The number of spaces to indent

A description of brace and indentation configuration

New Lines

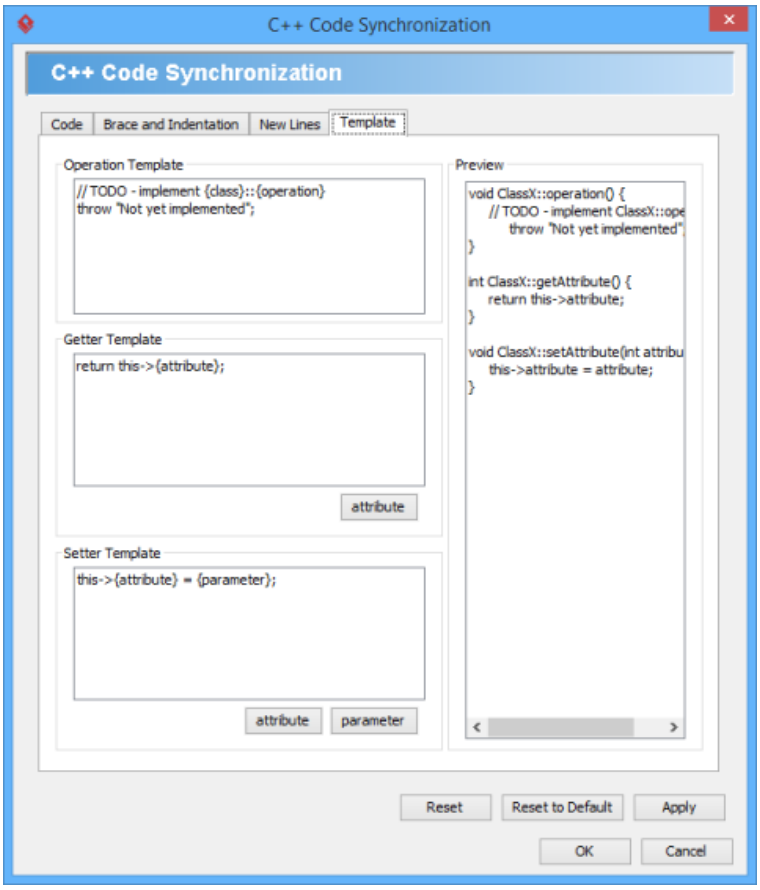


New lines configuration

Option	Description
Before first declaration	Number of blank lines to appear before the first declaration within Class declarations
Before different kind declaration	Number of blank lines to appear before a different kind of declaration
Before field declaration	Number of blank lines to appear before field declaration
Before method declaration	Number of blank lines to appear before method declaration

A description of new lines configuration

Template



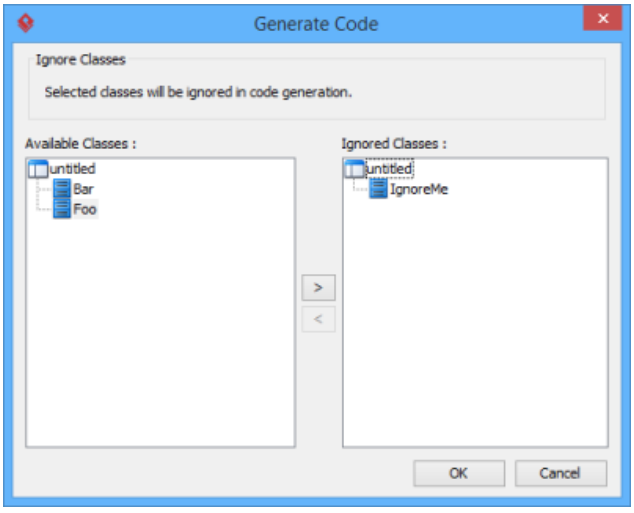
Template configuration

Option	Description
Operation Template	Defines a template of method body that will be applied when generating operations.
Getter Template	Defines a template of getter that will be applied when generating getter methods. Getter will be generated to attribute stereotyped as Property, or with property getter selected.
Setter Template	Defines a template of setter that will be applied when generating setter methods. Setter will be generated to attribute stereotyped as Property, or with property setter selected.

A description of template configuration

To ignore classes in generation

You can make certain UML class not to generate code against code generation by ignoring them. To ignore class(es), click **Ignore Classes...** in **Generate Code** dialog box. In the second Generate Code dialog box that popped up, select the class(es) to ignore and click **>** to move them to the ignore list. Click **OK** to confirm.



The class IgnoreMe is ignored

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

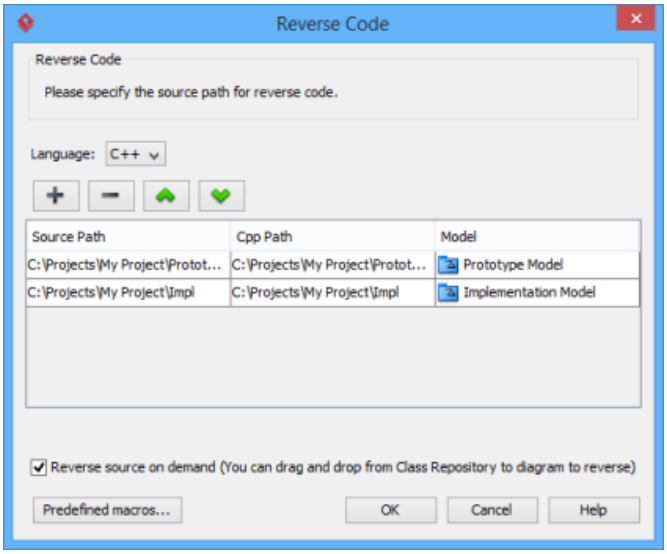
Generate/Update UML classes from C++ code

Round-trip engineering is the ability to generate model from source code and generate source code from model, and keep them synchronized. You can make use of round-trip engineering to keep your implementation model and source code up-to-date, so as to produce up-to-date documentation on your model.

Generate/Update UML classes from code

You can produce UML classes from source code, or to update from code all the reversed UML classes in project. To do this:

1. Select **Tools > Code > Reverse Java Code...** from the toolbar.
2. Select **C ++** as the **Language**.
3. In the **Reverse Code** dialog box, specify the mapping between source path and model. Model is a UML element that acts as a container of other elements. You can place the UML classes to be produced to specific model for better categorization. For example, you may create a Prototype model and an Implementation model for storing classes developed in prototype and implementation phrases respectively. Once a mapping is defined, round-trip engineering will be performed between the model and path as defined. You can add multiple Source-path-to-model mapping by pressing the **+** button. If you do not use model to structure your project, keep model to be **<root>**.



The mappings between source paths and model are defined

4. By default, an on-demand reverse engineering will be carried out, which means to form indexes to the added path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below. If you want to carry out actual reverse engineering, uncheck **Reverse source on demand**.
5. Click **OK** button to proceed with reversal.

Updating UML classes on a class diagram from code

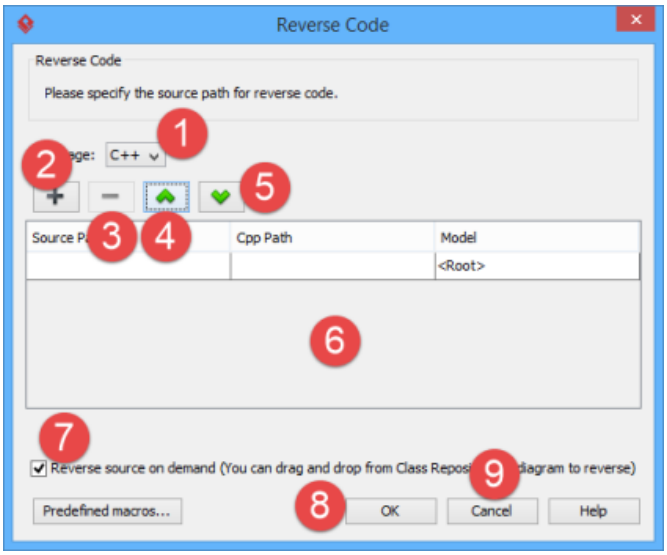
Once you have performed round-trip engineering for once, you can update UML class(es) on a diagram from source code for reflecting the changes made in code. To update, right click on the background of the class diagram for update and select **Utilities >** C++ **Round-trip > Reverse Code** from the pop-up menu.

NOTE: In order to trigger this function, make sure you have performed round-trip engineering at least for once and the diagram has at least one class.

Updating specific UML classes from code

Once you have performed round-trip engineering for once, you can update specific UML class(es) from source code for reflecting the changes made on that particular class(es) To update, select in class diagram the UML class(es) you want to update. Right click on them and select **C++ Round-trip > Reverse Code** from the pop-up menu.

An overview of Reverse Code dialog box



*An overview of **Reverse Code** dialog box*

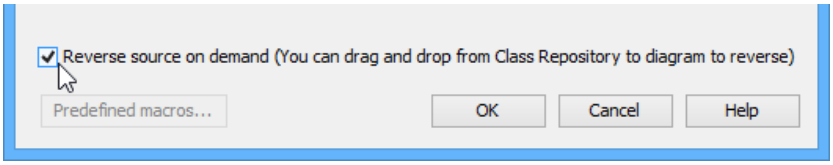
No.	Name	Description
1	Language	The programming language of the source code to reverse.
2	Add source-path-to-model mapping	Click to add a new mapping between source path where code will be reversed from and UML model.
3	Remove source-path-to-model mapping	Click to remove chosen source-path-to-model mapping.
4	Move source-path-to-model mapping up	Click to move chosen source-path-to-model mapping one item upward.
5	Move source-path-to-model mapping down	Click to move chosen source-path-to-model mapping one item downward.
6	Model-to-source-path mapping	A list of mapping between UML model and source path.
7	Reverse source on-demand	By checking, this means to form indexes to the source path(s) instead of actually reversing them. For details about on demand reverse engineering, refer to the section below.
8	OK	Click to start reversal.
9	Cancel	Click to close the **Reverse Code** dialog without reversing code.

*A description of **Reverse Code** dialog box*

On-demand reverse engineering

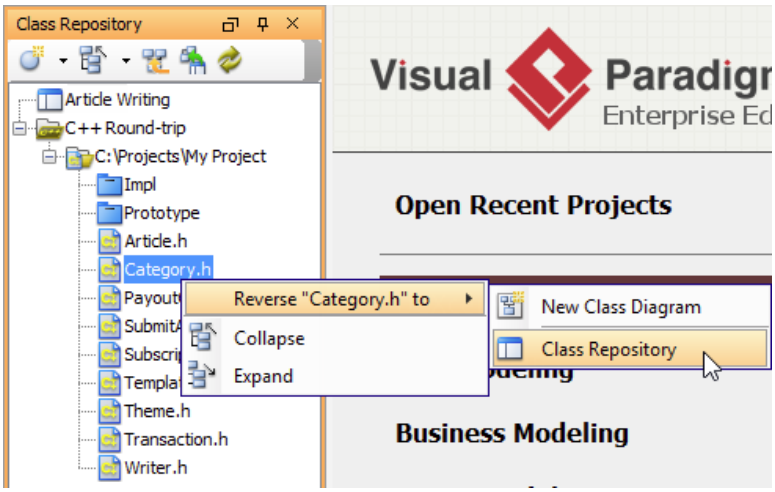
Consider if you have a project that contains million of C++ source file, and now you want to re-develop just a few classes in it. If you try to reverse the whole project it will take you a long time to complete the reverse due to the amount of classes (and relationships) are just too many. With on-demand reverse engineering, you will reverse the sources as indexes, and obtain an index tree in class repository. No actual UML classes will be reversed until you trigger the reverse manually. This reduces the processing time significantly.

To perform on-demand reverse engineering, make sure the option Reverse source on demand is checked in the **Reverse Code** dialog box.



*The option **Reverse source on demand** that appear in reverse dialog box*

When finished reverse, you can lookup the index tree in class repository. Then, right click on the class you want to reverse and select **Reverse Resources to** where **Resources** are the classes you have selected, and select either **New Class Diagram** or **Class Repository** from popup menu. Both options will result in reversing the selection to UML classes, while the option **New Class Diagram** will create a class diagram and place the classes in it.



Reversing a C++ source file from index tree

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse ORM POJO Classes

You can generate ORM classes which has POJO be the persistent API. On the contrary, those generated POJO classes can be reversed back to class model. In this chapter, you will learn how to reverse engineer ORM from POJO classes.

Reversing ORM POJO classes

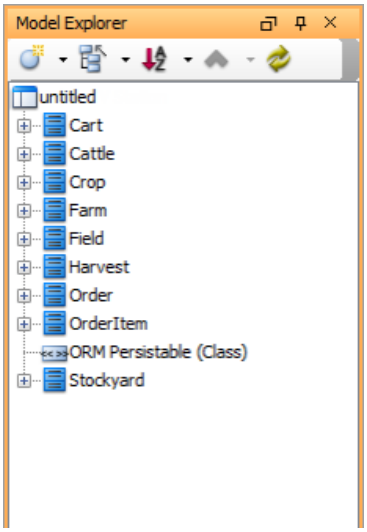
Reverse engineer ORM persistable classes from POJO classes.

Reversing ORM POJO classes

You can generate ORM classes which has POJO be the persistent API. On the contrary, those generated POJO classes can be reversed back to class model. This is particularly useful when you want to produce a class diagram from legacy ORM classes (code).

To reverse engineer class model from ORM POJO classes:

1. Select **Tools > Hibernate > Reverse Java Classes...** from the toolbar.
2. In the **Reverse Java Classes** dialog box, click **Add** to add the classpaths where the ORM classes exist.
3. From the **Available Classes** pane, select the classes you want to reverse and click **>**.
4. Click **OK**. You can find the reversed classes in the **Model Explorer**.



ORM classes reversed

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating Object-Relational Mapping Code

In this chapter, you will learn how to generate object-relational mapping (ORM) code for accessing database.

Generating code and database

Generate ORM tier and the necessary Java source files for accessing database.

Lazy collection setting

Description of lazy collection setting - a setting for improving application performance by loading up less objects to memory when necessary.

Persistent API

Introduce several available types of persistent API.

Using generated code

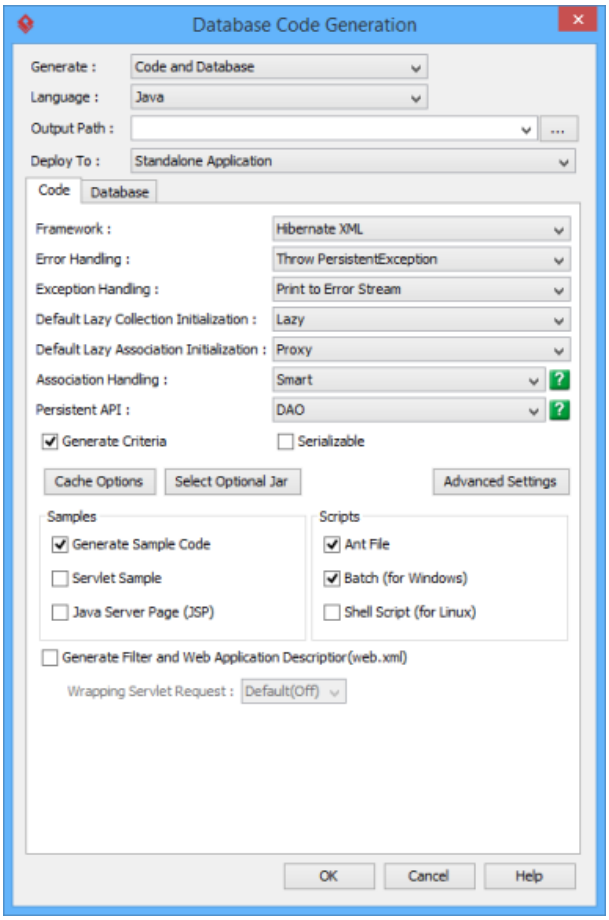
Provide you with samples of using the generated code.

Customizing getter and setter body

Specify the method body of getter and setter for persistent classes to be generated upon ORM code generation.

Generating code and database

1. Select **Tools > Hibernate > Generate Code...** from the toolbar.
2. Fill in the **Output Path** and select code generation options.



Database code generation

Option	Description
Framework	Determines how the mapping between class model and ER model is to be recorded. **Hibernate XML** - Hibernate mapping file (.hbm.xml) will be generated. It contains the mapping between class model and ER model in XML format. **JPA** - Not to generate the Hibernate mapping file but to store the mapping information directly in the generated Java source file as annotations.
Error Handling	For Error Handling, select the way to handle errors. The possible errors include PersistentException, GenericJDBCException, and SQLException. **Return false/null** - It returns false/null in the method to terminate its execution. **Throw PersistentException** - It throws a PersistentException which will be handled by the caller. A try/catch block has to be implemented to handle the exception. **Throw RuntimeException** - It throws a RuntimeException which will be handled by the caller. A try/catch block has not been implemented to handle the exception. The exception will be caught at runtime. **Throw User Defined Exception** - It throws an exception that is defined by user. Select this to input the fully qualified name of the Exception class.
Exception Handling	For Exception Handling, select how to handle the exception. **Do not Show** - It hides the error message. **Print to Error Stream** - It prints the error message to the Error Stream. **Print to log4j** - It prints the error message to the log4j library.
Default Lazy Collection Initialization	Lazy collection initialization avoids the associated objects from being loaded when the main object is loaded. With lazy collection initialization, all associated object (1 to many) will not be loaded until you access it (e.g. getFlight(0)). Enabling this option can usually reduce more than 80% of the database loading. **Lazy** - You must update both ends of a bi-directional association manually to maintain the consistency of the association. Besides, casting of object(s) to its corresponding persistence class is required when retrieving object(s) from the collection.

	Extra - When you update one end of a bi-directional association, the generated persistent code is able to update the other end automatically. Besides, you do not need to cast the retrieved object(s) into its corresponding persistence class when retrieving object(s) from the collection. **Non-lazy** - Load the associated objects when the main object is loaded.
Default Lazy Association Initialization	Each association can set lazy. This setting is for those associations that has lazy set as Unspecified. **Proxy fetching** - A single-valued association is fetched when a method other than the identifier getter is invoked upon the associated object. **"No-proxy" fetching** - A single-valued association is fetched when the instance variable is accessed. Compared to proxy fetching, this approach is less lazy; the association is fetched even when only the identifier is accessed. It is also more transparent, since no proxy is visible to the application. This approach requires buildtime bytecode instrumentation and is rarely necessary. **False** - Not to set lazy.

Association Handling	**Smart association** handling, updating either side of a bi-directional association will automatically trigger an update on the other side. For example:
	many.setOne(one);
	or
	one.many.add(many);
	It also provides static type checking by using strong type collection.
	With **Standard** association handling, you will need to update both sides of a bi-directional association to maintain consistency. For example:
	many.setOne(one);
	one.getMany().add(many);
	With **Generics** association handling, you will need to update both sides of a bi-directional association to maintain consistency, which is the same as the Standard association handling as described above, except that for Generics association handling, generics (e.g. Set<Account>) has been used for type specialization.

Persistent API	For Persistent API, select the type of persistent code to be generated, either Static Methods, Factory Class, DAO or POJO. The decision to which API to select depends on the practice of coding. **Static Methods** - Client can create, retrieve and persist with the PersistentObject directly. **Factory Class** - FactoryObject class will be generated for client to create and retrieve the PersistentObject. Client can directly persist with the PersistentObject. **DAO** - The PersistentObjectDAO class helps client to create, retrieve and persists to PersistentObject. **DAO (with Interface)** - A variation of DAO. In DAO (with Interface), the DAO class become an interface, and the implementation is moved to the DAOImpl class. Instance methods are used instead of Static methods. With these changes, you can define your own DAO objects and swap between DAO implementations easily. **POJO** - The PersistentManager helps client to retrieve and persist with PersistentObject. **Mapping only** - Just to generate the mapping without generating any code file.
Generate Criteria	Check the option to generate the criteria class for each ORM Persistable class. The criteria class is used for querying the database in object-oriented way.
Serializable	Generate implement Serializable in Java, [Serializable] in C#.
Cache Options	Configure Second Level Cache to improve performance.
Selected Optional Jar	Select the libraries and JDBC driver to be included in the generation of the orm.jar file (Persistent Library).
Advanced Settings	Click to edit some of the advanced settings: **Default Order Collection Type** - Select the type of ordered collection to be used in handling multiple cardinality relationship, either List, Array or Map. **Default Un-Order Collection Type** - Select the type of un-ordered collection to be used in handling the multiple cardinality relationship, either Set or Bag. **Override toString Method** - Select the way that you want to override the toString method of the object. There are three options provided to override the toString method as follows: • **All Properties** - the toString method returns a string with the pattern • **ID Only** - the toString method returns the value of the primary key of the object as string. "Entity[<column1_name>=<column1_value><column2_name>=(column2_value>...]" • **Business Key** - You can specify business key per class, inside the Business Key tab of class specification. Business keys comprise of attributes. The toString method will return a string of business keys. • **No** - the toString method will not be overridden **Flush Mode** - Select the Flush Mode to be used in flushing strategy. User can select Auto, Commit, Always and Never. • **Auto** - The Session is sometimes flushed before executing query. • **Commit** - The Session is flushed when committing Transaction.

- **Always** - The Session is flushed before every query.
- **Never** - The Session is never flushed unless the flush method is called.

Mapping File Column Order - <to-be-entered>

Getter/Setter Visibility - Set the visibility of getter and setter methods for attributes in persistable classes. By selecting **Public**, public getters and setters will be generated. By selecting **Follow attribute**, the visibility will follow the visibility of attributes.

Generate Mapping - Overwrite or update XML mapping files

Mapping type for date - Select hibernate type for mapping date type: Date, Time, Timestamp

Composite ID with Association - Generate <key-many-to-one> or <key-property> for composite ID, <key-many-to-one> has some bugs and <key-property> is recommended.

Generate property constant - Generate constant for each properties, best for building customized HQL or Criteria.

Generate lower case package - It is preferred to use lower case for package name in Java. Check this to enforce the use of lower case for package name no matter the package element in project has lower or upper case as name.

Separate subclass mapping file - Generate subclass's XML mapping file in separate file, instead of generating in parent class mapping file.

Attribute Prefix - Prefix of names of attributes in persistable classes.

Persistent API return type - DAO method return <<ORM Implementation>> subclass.

Public ID setter - Generate ID setter as public.

Public Version setter - Generate Version property setter as public.

Generate custom annotation - Generate Java annotations defined in model specification.

Generate non-persistable association - Generate non-persistable association in persistable class.

Generate @SuppressWarning("all") annotation - Generate @SuppressWarning("all") annotation to reduce compiler warnings.

Generate sessionless methods - Generate DAO methods without session parameter.

Property access - Hibernate access attribute by getter/setter methods.

Generate validator constraints - Generate hibernate validator annotation for validating String length.

Encoding - Encoding for generating source files.

Persistent Manager Package - Customize the package for PersistentManager or using default.

Persistent Collection - Customize persistent collection's implementation class for each collection type.

Non Persistent Collection - Customize non-persistent collection implementation class.

Samples	Sample files, including Java application sample, Servlet sample and Java Server Page (JSP) sample are available for guiding you through the usage of the Java persistence class. You can check the option(s) to generate the type of sample files you need.
Scripts	Check the option(s) to generate Ant File/Batch/Shell scripts, for the direct execution of generated code.
Generate Filter and Web Application Descriptor (web.xml)	Determines whether to generate the file web.xml essential in Web application development.

Available options in ORM code generation

3. Click **OK** button to start code generation.

Related Resources

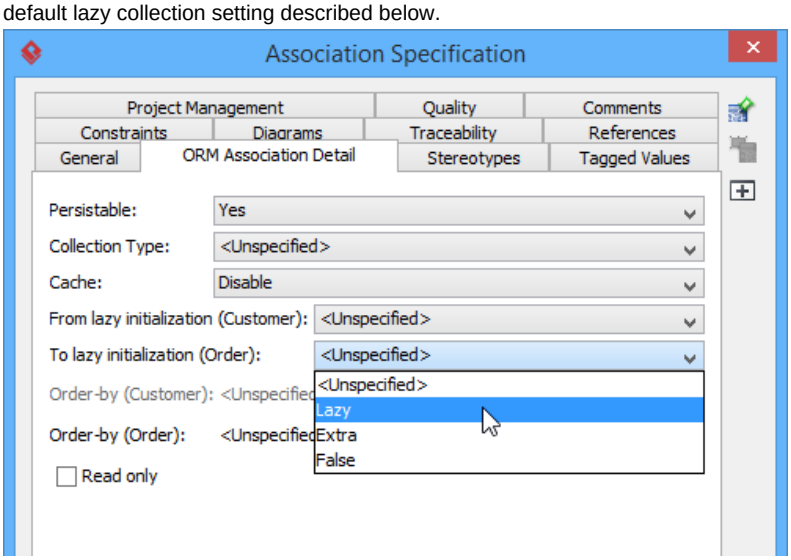
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Lazy collection setting

Setting lazy collection for association

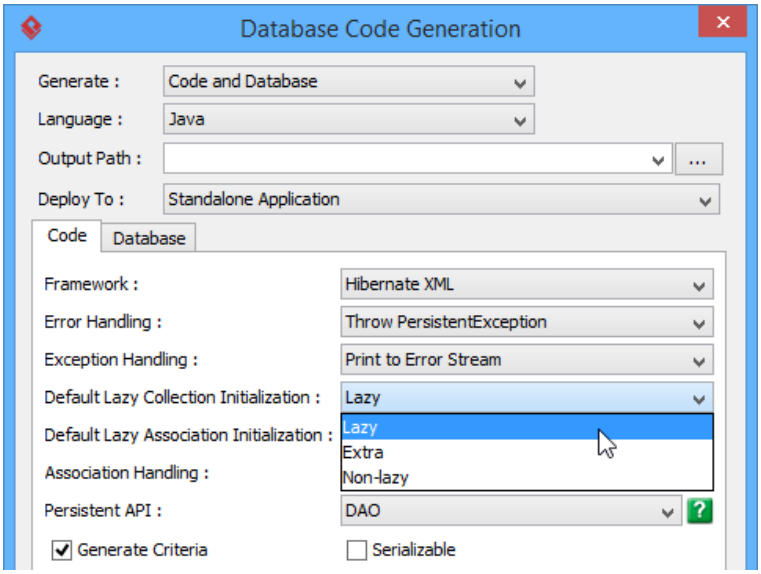
1. Open specification dialog box of association.
2. Switch to ORM Association Detail tab, select **Lazy** or **Extra** for **From lazy initialization** or **To lazy initialization**, depending on which side multiplicity is *. Lazy collection is fetched when the application invokes an operation upon that collection. Extra lazy supports individual elements of the collection are accessed from the database as needed, rather than fetch the whole collection. If the value is **Unspecified**, it will follow the default lazy collection setting described below.



Lazy collection setting

Setting default lazy collection when generating ORM

1. Open **Database Code Generation** dialog box.
2. Specify a value for **Default Lazy Collection Initialization**.



Default lazy collection setting

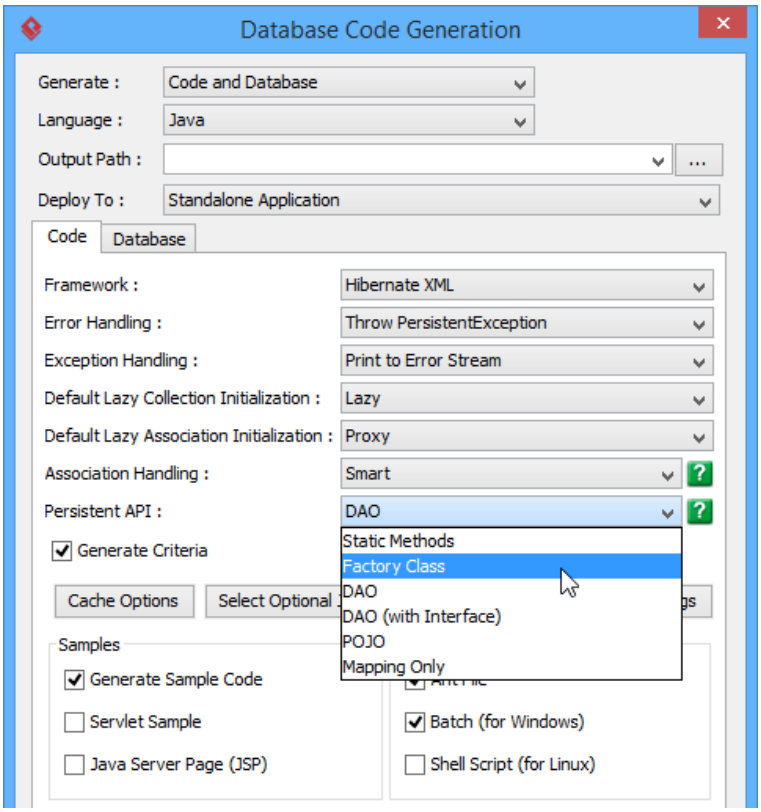
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Persistent API

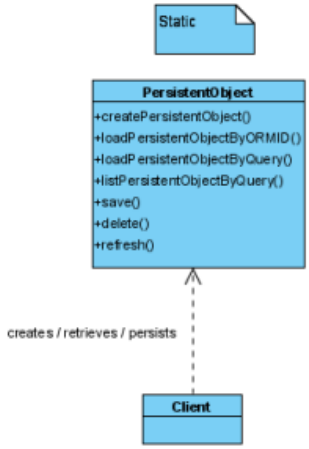
The persistent API setting supports various styles for generated code. It can be configurated in **Database Code Generation** dialog box.



Persistent API setting

Static methods

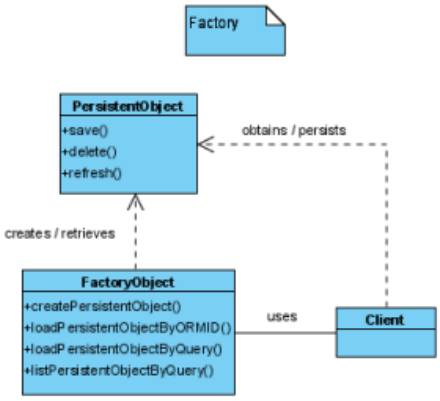
Static methods generate all persistent methods in the persistent class, client can access the methods in the same persistent object.



Static methods

Factory class

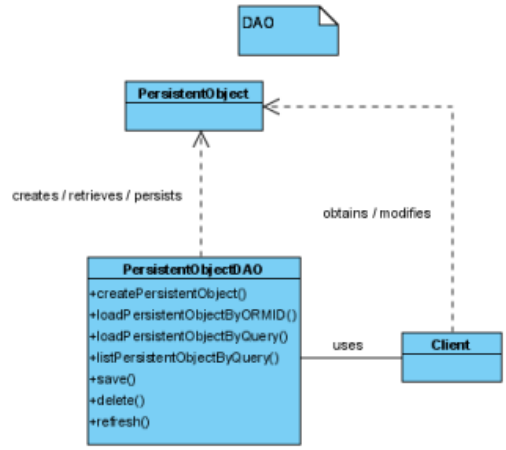
Factory class generates save/delete/refresh methods in persistent class, other persistent methods that return persistent object are generated in factory class.



Factory class

DAO

DAO generates all persistent methods in DAO class, a DAO class is generate for each persistent class.



DAO

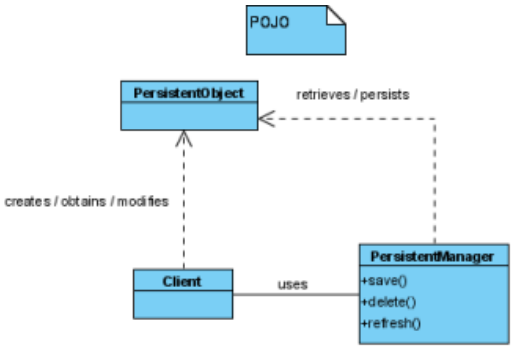
DAO (with interface)

DAO (with Interface) generates all persistent methods signature in DAO interface. A DAO interface is generate for each persistent class, and a corresponding DAO implementation class is generated with default persistent implement.

DAO (with Interface)

POJO

POJO generate persistent object in Plain Old Java Object style, without generating any persistent methods. Client can access persistent methods in PersistentManager object.



POJO

Mapping only

Mapping only does not generate any code, it only generates the XML mapping file required for ORM.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using generated code

The following sections demonstrate how to use the generate ORM code with factory method persistent API.

Inserting records

1. Create persistent object with factory create method.
2. Save persistent object with save method.

The following codes demonstrate how to insert a *Product* record:

```
PersistentTransaction t = ErdPersistentManager.instance().getSession().beginTransaction();
try  {
Product product = ProductFactory.createProduct();
product.setName( "ABC Keyboard");
product.setPrice(24.5);
product.save();
}
catch  (Exception e) {
t.rollback();
}
```

Selecting records

Factory method provides a convinient listByQuery method, accept condition and order by as parameter` and return array of persistent object.

The following codes demonstrate how to select a list of *Product* records, null for condition parameter will select all records, null for order by parameter does not sort in any order:

```
Product[] products = ProductFactory.listProductByQuery(  null ,  null );
for  (  int  i = 0; i < products.length; i++) {
System.out.println(products[i]);
}
```

Another useful method to select a persistent object by ID is loadByORMID. The following codes demonstrate how to select a l *Prouct* record by ID.

```
Product product = ProductFactory.loadProductByORMID(  1);
```

Updating records

1. Select a persistent object from database.
2. Update the persistent object.
3. Save persistent object with save method.

The following codes demonstrate how to update a *Product* record:

```
PersistentTransaction t = ErdPersistentManager.instance().getSession().beginTransaction();
try  {
Product product = ProductFactory.loadProductByORMID(1);
product.setName( "DEF Keyboard");
product.save();
}
catch  (Exception e) {
t.rollback();
}
```

Deleting records

1. Select a persistent object from database.
2. Delete persistent object with delete method.

The following codes demonstrate how to delete a *Product* record:

```
PersistentTransaction t = ErdPersistentManager.instance().getSession().beginTransaction();
try  {
Product product = ProductFactory.loadProductByORMID(1);
product.delete();
}
catch  (Exception e) {
t.rollback();
}
```

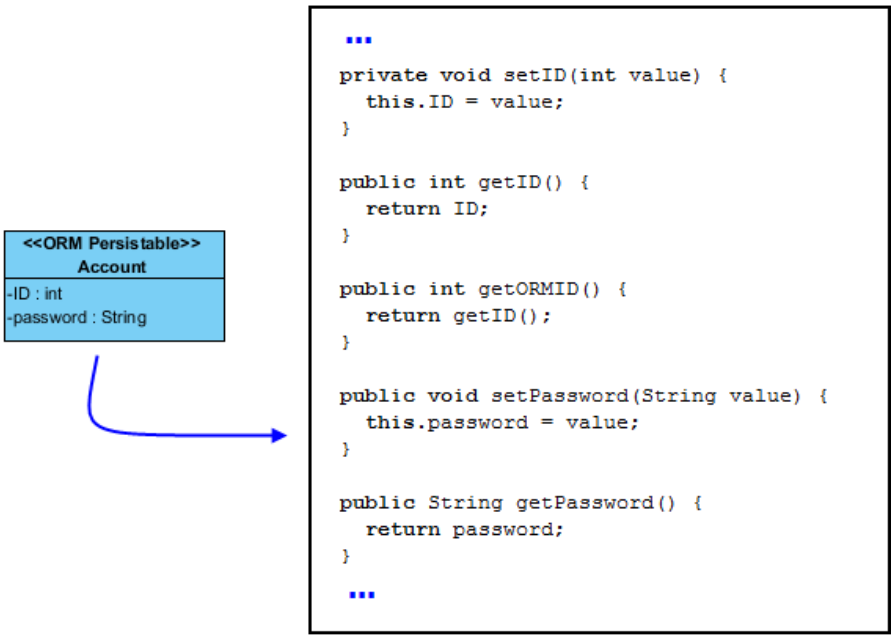
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Customizing getter and setter body

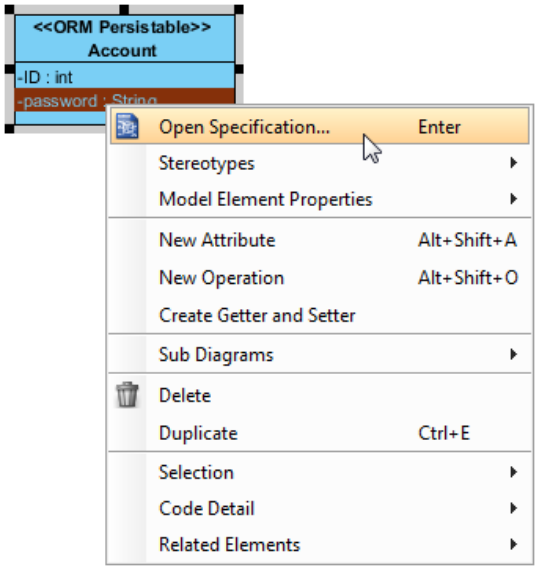
In generated ORM code, getters and setters will be generated for attributes added to every ORM Persistable class. Sometimes, you may want to customize the method body of those getters and setters, like to apply security checking or to print a statement upon the updating of data. In these cases, you can customize the getter and setter of attribute to add the code you want.



A part of the generated code, showing the getters and setters generated from attributes of an ORM Persistable class

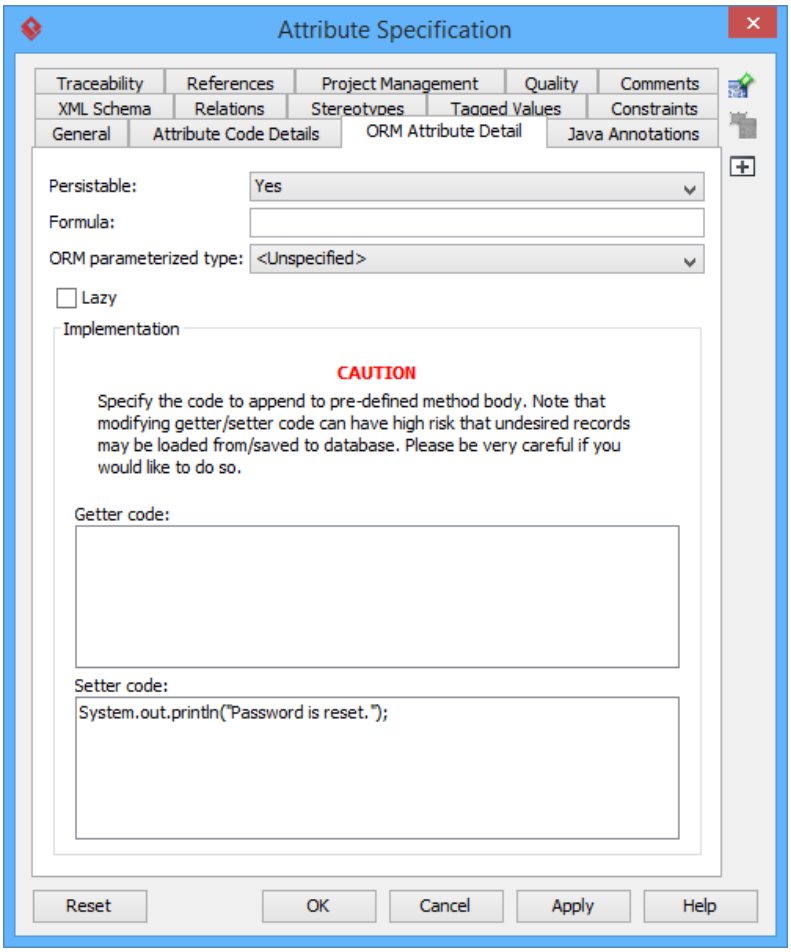
To customize getter/setter of attribute:

1. Right click on the attribute that you want to customize its getter or setting and select **Open Specification...** from the popup menu.



Opening the specification of attribute

2. Open the **ORM Attribute Detail** tab and enter the code body in **Getter/Setter** code sections.



Customizing the setter of attribute

When you generate code, you will see the entered code appended to the generated getter or setter.

```
...
public int getORMID() {
  return getID();
}

public void setPassword(String value) {
  this.password = value;
  System.out.println("Password is reset");
}

public String getPassword() {
  return password;
}

public String toString() {
  return String.valueOf(getID());
}

}
```

Customized setter

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

State Machine Diagram Code Generation

You can model the state machine of your system or application, and generate the code file from your design. In this chapter, you will learn both the modeling part which involve class and state machine diagram, and code generation.

Modeling guidelines

Model state machine with class and state machine diagram.

Generating state machine code

Generate state machine code from your design.

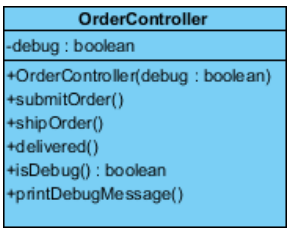
Reverse state machine code

Modeling guidelines

A state machine involves a number of states as well as the transition of states. You can generate source code for a state machine by first creating a controller class, then create sub-state machine diagram from the controller class, model the state machine. In this chapter, you will see how to model a state machine readily for code generation. For the steps of code generation, read the next chapter.

Step 1 - Modeling controller class

A controller class is a class that is used for controlling and managing the activities within a use case. It also manage the states within the use case or the system.



A controller class

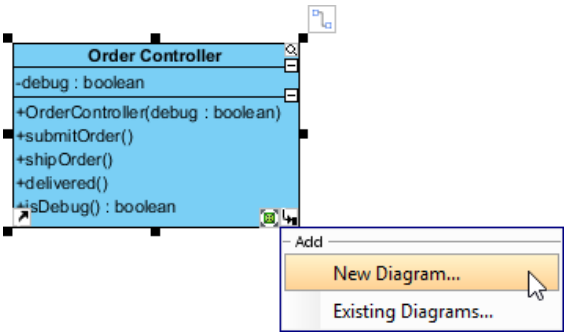
You can create a controller class by selecting **Class** from diagram toolbar and click on the diagram. Name the class properly to represent the nature of controller class. Very often, people name controller class as *SomethingController* where the *Something* refers to a use case or the model that the controller need to manage. For example, a *PhoneController* is for controlling operations of a telephone and managing its states like waiting, dialing, etc.

You can add attributes to the class by right clicking on it and selecting **Add > Attribute** from the popup menu. Attributes defined will be generated to code. However, you do NOT need to add attributes for states nor attributes for remembering states. Everything about states will be managed by the state machine in state machine diagram.

Add operations to the class by right clicking on it and selecting **Add > Operation** from the popup menu. There should be operations that may update the state.

Step 2 - Modeling state machine

You need to create a sub state machine diagram from the controller and model the state machine there. To create a sub state machine diagram, move the mouse pointer over the controller class, click on the resource icon at bottom right corner and select **New Diagram...** from the popup menu. In the **New Diagram** window, select **State Machine Diagram**. Click **Next** and then **OK** to confirm.



To create a sub state machine diagram from controller class

In the state machine diagram, draw the states as well as the transition of states. Since the states will be generated to source code, you are adviced to consider the naming convention of the programming language you want to generate when naming states.

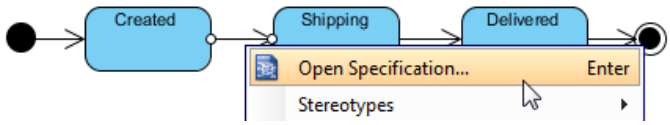
A state machine

You do not need to name the transitions as we will assign operations to them. But if you want you can do this. It will not affect the code that will be generated.

Step 3 - Assigning operations to transitions

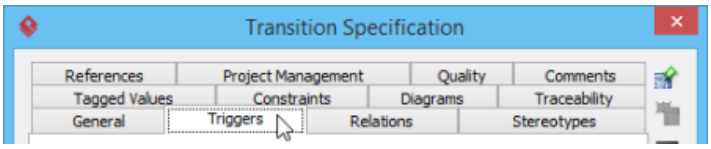
A transition is a relationship between two states, representing the update of states. Previously you have defined operations in controller class. Now, you need to assign those operations to the transitions to indicate the cause of state change. To assign an operation to transition:

1. Right click on a transition and select **Open Specification...** from the popup menu.



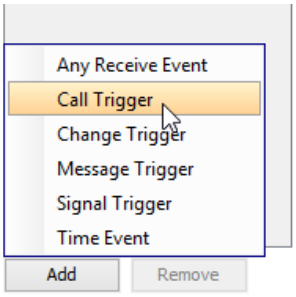
Open specification of transition

2. Open the **Triggers** tab.



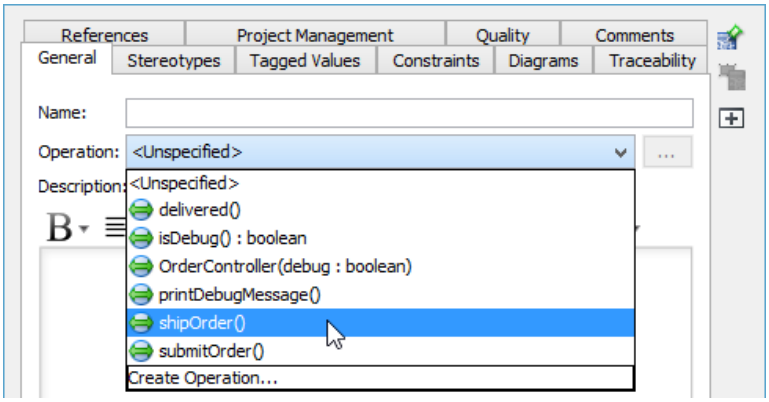
Open Triggers tab

3. Click **Add** and select **Call Trigger** from the popup menu.



Add a Call Trigger

4. In the **General** tab, specify the operation from the **Operation** drop down menu.



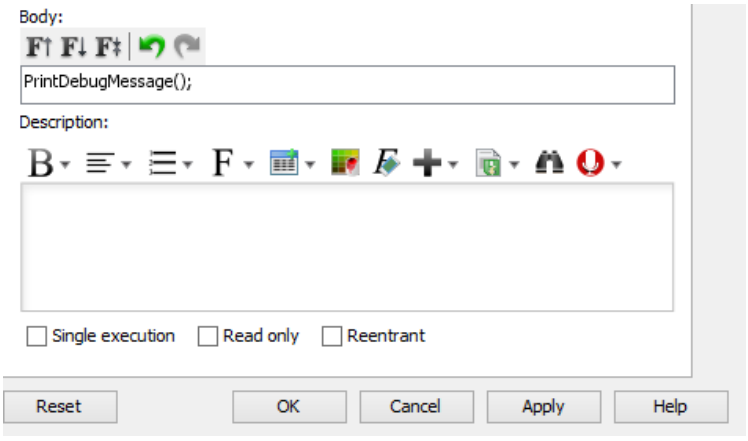
Select operation

Repeat the steps to assign operations to all transitions.

Step 4 - Specifying method body for the entry/exit of state

You can specify the invokcation of method call when entering and exiting a state by updating the Entry properties of state. To do this:

1. Right click on the state and select **Open Specification...** from the popup menu.
2. Click on the drop down menu next to the **Entry** field, then select **Create Activity...**.
3. In the specification, fill in the **Body** field by entering the methods to invoke. Click **OK** to confirm.



Specifying method for Entry

4. Repeat the steps on **Exit**.

Step 5 - Specifying method body for operation

Part of the method body of operations being assigned to transitions can be defined by editing the **Exit** property of a transition. To do this:

1. Right click on the transition where operation was assigned.

2. Click on the drop down menu next to the **Exit** field, then select **Create Activity...**.
3. Fill in the **Body**. Click **OK** to confirm.
4. Click **OK** to confirm and go back to diagram.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating state machine code

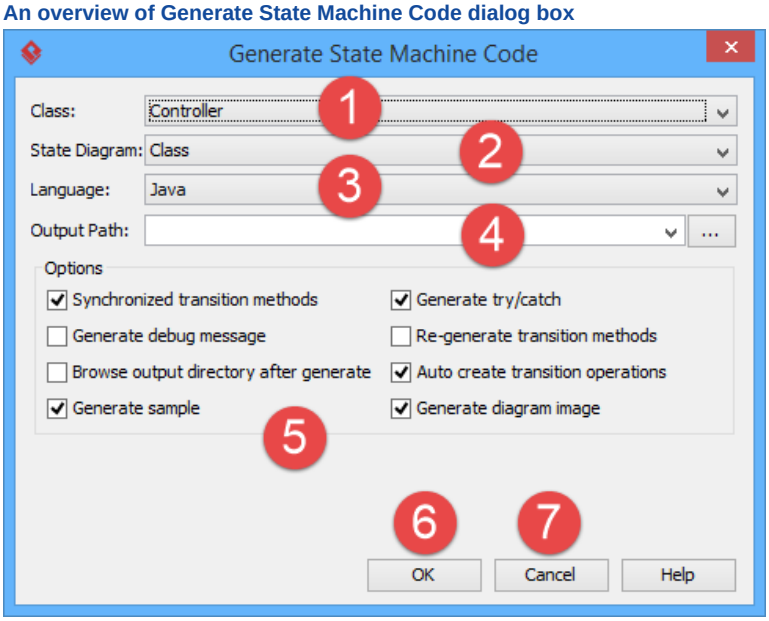
Once the controller class and state machine are modeled, you can generate state machine code for the controller and state machine. With the generated state machine, you can run instant generator to produce other classes, like the model and view classes, and incooperate with the state machine code.

To generate state machine code:

1. Select **Tools > Code > Generate State Machine Code...** frrom the toolbar.
2. In the **Generate State Machine** dialog box, select the controller class for generating state machine.
3. Select the state machine in the drop down menu **State Diagram** for generating code.
4. Select the programming language of the code.
5. Specify the output path to save the generated code to.
6. Optionally configure the generator options.
7. Click **OK** to generate.

NOTE: There must be at least one class that contain sub state machine diagram in order to open the **Generate State Machine Code** dialog box.

An overview of Generate State Machine Code dialog box



*An overview of **Generate State Machine Code** dialog box*

No.	Name	Description
1	Class	The controller class for generating state machine.
2	State Diagram	The state machine (in the form of state machine diagram) to generate. It must be a sub-class of the chosen controller class.
3	Language	The programming language of code to generate.
4	Output Path	The output path of state machine code.
5	Options	Options for code generation. Below is a description: **Synchronized transition methods** - By checking, it causes the generated code to: • Java: add the synchronized keyword to the transition method declarations. • VB.net: encapsulate the transition method's body in a SyncLock Me, End SyncLock block. • C#: encapsulate the transition method's body in a lock(this) {...} block. **Generate try/catch** - Uncheck to not generate try/catch code. You are recommended to keep this checked. Uncheck only in C ++ applications where exceptions are not used. **Generate debug message** - Adds debug output messages to the generated code. **Re-generate transition methods** - Check to overwrite the transition methods in code, including the implementation. **Browse output directory after generate** - Open the output path. **Auto create transition operations** - If a transition is named, but does not have Operation assigned. By checking this option operation will be created to the parent class, named as the transition name. **Generate sample** - Generate sample files to guide you how to work with the generated file.

		Generate diagram image - Generate PNG image for chosen state machine diagram.
6	OK	Click to start code generation.
7	Cancel	Click to close the **Generate State Machine Code** dialog box.

*A description of **Generate State Machine Code** dialog box*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

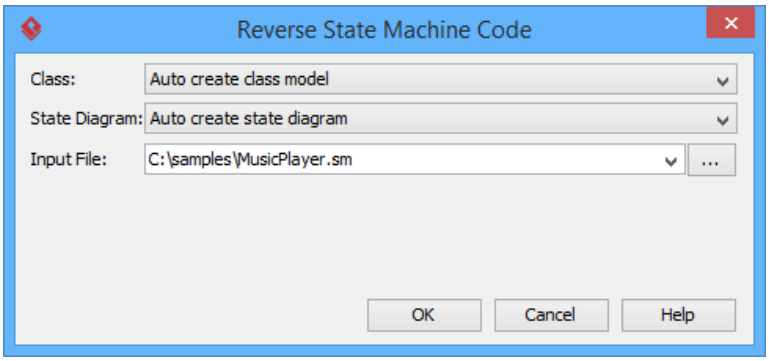
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse state machine code

If you have a state machine definition (.sm) and you want to visualize the state machine with a UML state machine diagram, you can make use of the reverse function to produce the class and state diagram essential to visually represent the definition.

To reverse state machine definition (.sm):

1. Select **Tools > Code > Reverse State Machine Code...** from the toolbar.
2. Specify the class, state diagram and the .sm file in the input field. Click **OK** to continue.



Reverse state machine

Field	Description
Class	The controller class for managing the state. Only classes that have a state machine diagrams as sub-diagrams would be listed in the drop down menu. You can select an existing class for managing the state machine. If such a class is not available, leave the option **Auto create class model** selected.
State Diagram	The diagram where the state machine definition to be reversed will be visually presented at. State machine diagrams that are sub-diagram of classes are listed in the drop down menu. You can select the one for visualizing the state machine definition or create a new one by selecting Auto create state diagram.
Input File	The state machine definition (.sm) file to be visualized.

An overview of Reverse state machine window

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating REST API

The word REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style used when designing networked applications. Web service APIs that conform to the REST architectural constraints are known as RESTful, or REST API. Visual Paradigm supports modeling the underlying communication model of REST API, as well as the generation of REST API and API documentation.

Overview of REST API Generation

REST is an architectural style used when designing networked applications. In this page you will learn what REST and REST API is and how Visual Paradigm supports REST.

Modeling REST API

You can design your REST API by drawing a class diagram that represents your resource, the request and response body. You will learn how to draw such a class diagram in this page.

Generating REST API

Once you have finished the modeling of your REST Resource(s), you can generate the API and, optionally, the API documentation. This page will show you how to generate REST API.

Using REST API (as a service consumer)

Consumers of RESTful service have to go through a series of steps in order to obtain the API code required for accessing a REST Resource. This page will show you the steps in detail.

Overview of REST API Generation

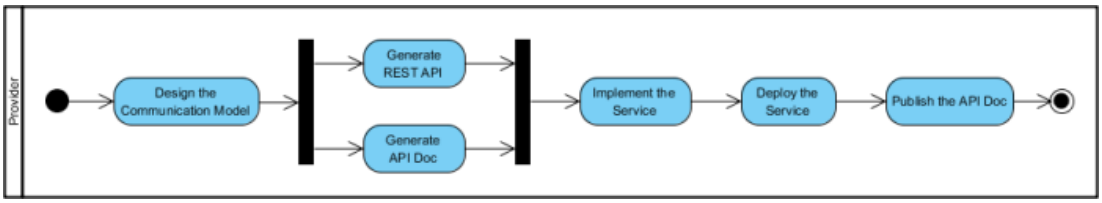
What is REST API?

The word REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style used when designing networked applications. Web service APIs that conform to the REST architectural constraints are known as RESTful, or REST API.

How Visual Paradigm supports REST API?

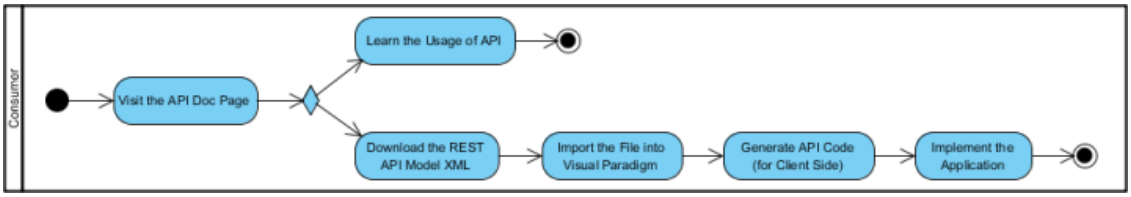
Visual Paradigm supports modeling the underlying communication model of REST API, as well as the generation of REST API and API documentation.

The following Activity Diagram shows you the steps that a provider will take in order to produce the REST API and related API documentation. First of all, the service provider will design the communication model with a class diagram that visualize the REST service, the request and response body. He can then generate the REST API and API documentation from the class diagram. After that, the provider can continue with programming the service logic. When finished, he can deploy the service and publish the API on their website.



Activity Diagram - How can a provider design and produce the REST API?

The following Activity Diagram shows you the steps that a consumer will take in order to use the service. Consumer of service can visit the API documentation page, download an XML file and then import the XML file into Visual Paradigm. By doing so, they can then generate the source code and the API required in accessing the service. The final step would be to program the application that uses the service with the generated source code.



Activity Diagram - How can a client access a service with REST API?

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Modeling REST API

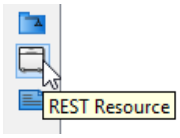
You can design your REST API by drawing a class diagram that represents your resource, the request and response body.

Drawing REST Resource

A REST resource is the fundamental unit of a web service that conforms to REST. It is an object with a URI, the http request method, associated parameters and the request/response body. Each of the REST resources represents a specific service available on the path specified by its URI property. Therefore, if you want to model multiple services, please draw multiple REST resources.

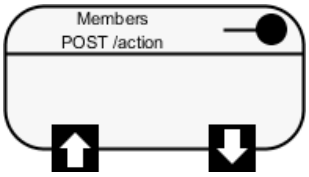
To draw a REST resource:

1. Select **Diagram > New** from the toolbar. In the **New Diagram** Window, select **Class Diagram** and the click **Next**. Enter the diagram name and description and then click **OK**.
2. Select **REST Resource** in the diagram toolbar.



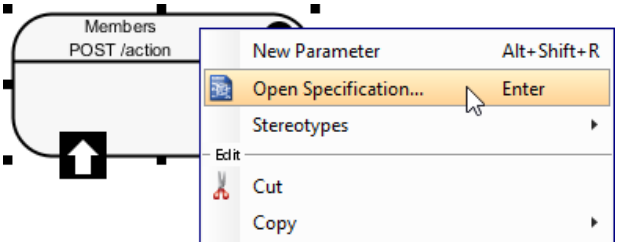
Select REST Resource in diagram toolbar

3. Click on the diagram to create a REST Resource. Name the resource by giving it a short and meaningful name.



REST Resource created

4. Right click on the REST Resource and select **Open Specification...** from the popup menu.

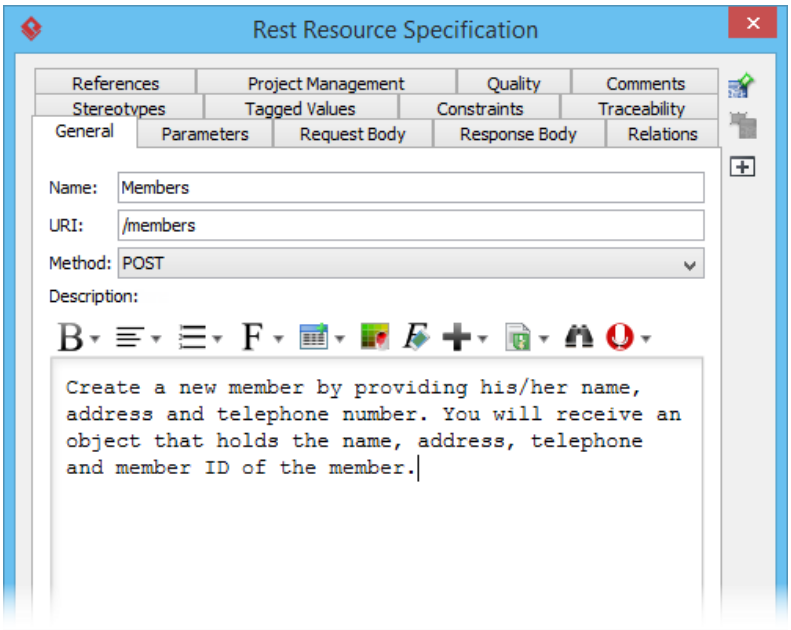


Opening the specification of REST Resource

5. In the **General** tab, fill in the following:

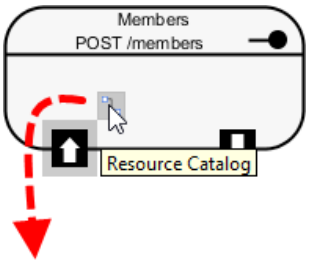
Property	Description
URI	Each REST Resource has its own URI. Consumers access to URL to access for the REST Resource. Typically, a RESTful URI should refer to a resource that is a thing instead of referring to an action. Therefore, when you are deciding the URI, try to use a noun instead of a verb.
Method	Specifies the action to act on the resource. For details, please read the section Methods (HTTP methods) below.
Description	Description of resource that will appear in generated API documentation. It is recommended to provide clear description of the service, so that the consumer know what the service is and how to operate with it.

General properties of a REST Resource



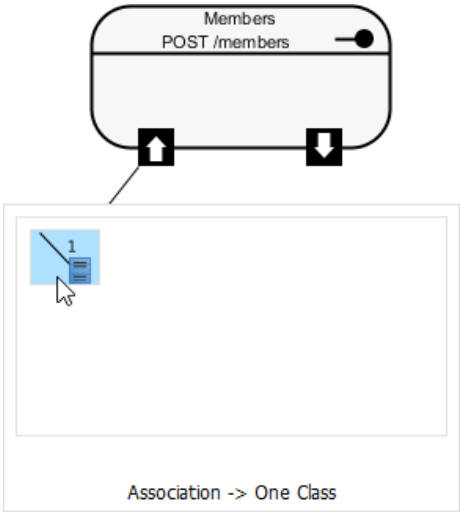
URI, method and description filled

6. Click **OK** to confirm the changes.
7. If the REST Resource uses a POST, PUT or DELETE method and if parameter is required in using the REST Resource, model the parameters by drawing class(es). Move your mouse pointer over the **REST Request Body** icon. Press on the **Resource Catalog** button and drag it out.



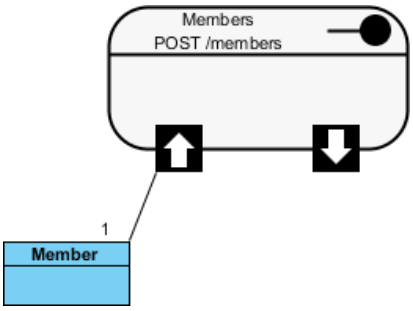
Create class from REST Request Body

8. Release the mouse button and select **Association -> One Class** from Resource Catalog.



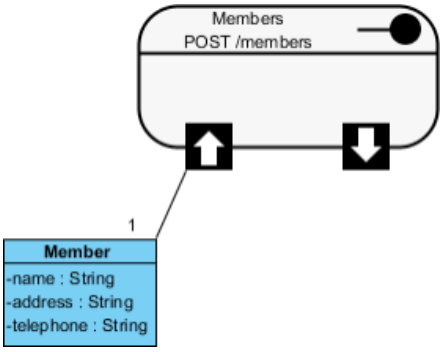
Select One Class

9. Release the mouse button to create the Request class. The class is named based on the REST Resource by default. You can rename it if you like. For instance, if you are going to create a member via the /members REST Resource, you probably need to send the member's details to the server for creating a member record. Therefore, name the class *Member* for storing member's details.



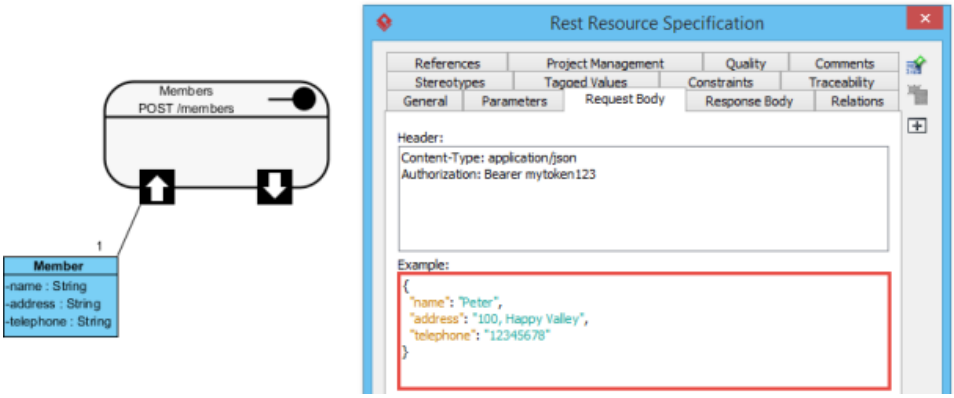
Class created from REST Request Body

10. Add the attributes into the classes. These attributes will hold the data that sends to the server.



Attributes added

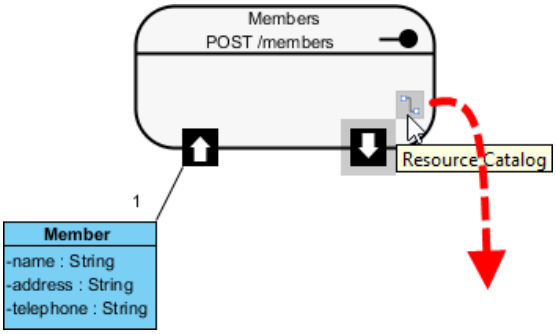
Here is a comparison between the class model and the representation of request body in JSON.



Comparison between class model and Request Body in JSON

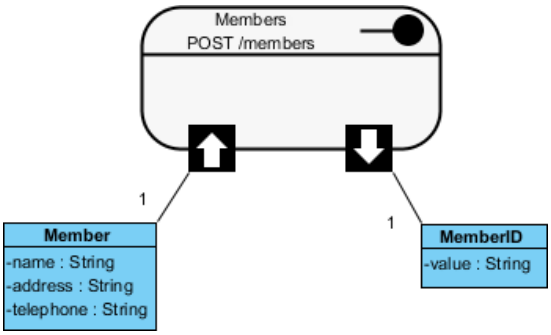
Note that you are free to create a more complex structure by creating more associated classes, but normally you don't need to do this.

11. Now, you can move on to designing the response part of the REST Resource. Move your mouse pointer over the **REST Request Body** icon. If the service will return a simple data value or object, press on the **Resource Catalog** button and drag it out. Then, select **Association -> One Class** from Resource Catalog. If the service will return an array of object, select **Association -> Many Class** from Resource Catalog.



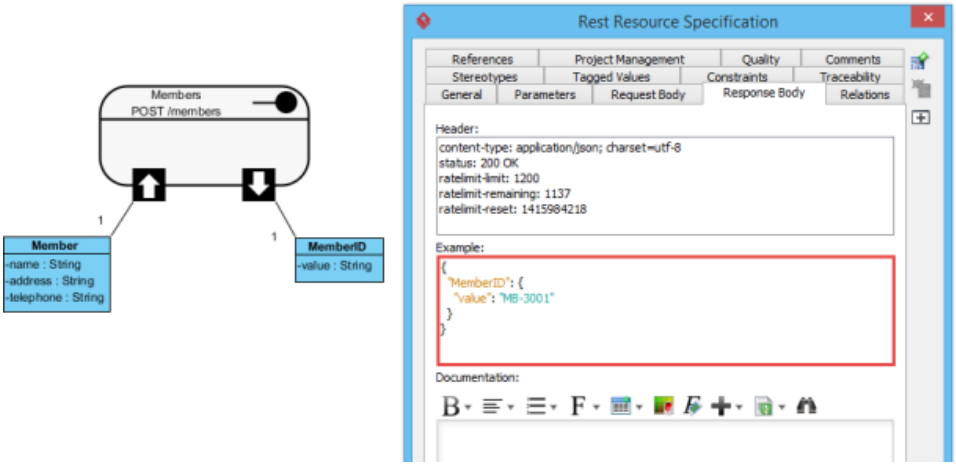
Create class from REST Response Body

12. Name the class and add the attribute into the class.



Class created from REST Response Body

Here is a comparison between the class model and the representation of response body in JSON.



Comparison between class model and Response Body in JSON

Specifying parameters for REST Resource that uses GET

Parameters are referring to query parameters used for passing data to a service. For example, when you use a 'currency converter' service, you probably need to pass the amount to convert, the current currency and the target currency to the service, in return for the converted amount. The amount to convert, the current and target currency are therefore the parameters of service.

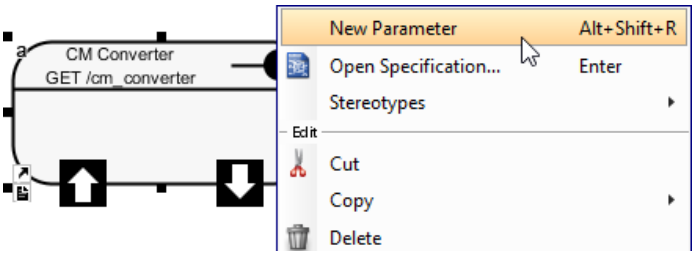
A feature of parameters is that they are optional. Another character of parameters is that they are non-unique, which means that you can add the same parameter multiple times.

Parameters are appended to the path of a URL when submitting a HTTP request. A URL with parameters could look like this:

http://www.example.com?age-limit=18

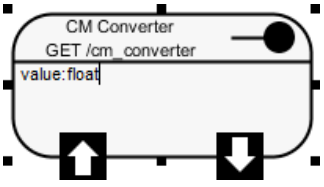
To add parameters to a REST Resource:

1. Right click on the REST Resource and select **New Parameter** from the popup menu.



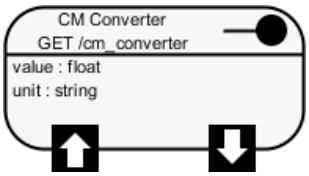
New parameter

2. Enter the name of the parameter. If you like, you can specify the type as well. Note that the specification of type is just for documentation purpose. While it helps the consumer to understand what kind of data is expected, it won't have any effect in code level. In coding, parameters are always put into a Map that takes string as both key and value.



Parameter created

3. Press **Enter**.
4. Repeat step 2 and 3 to create all parameters. Press **Esc** when finished creating all parameters.

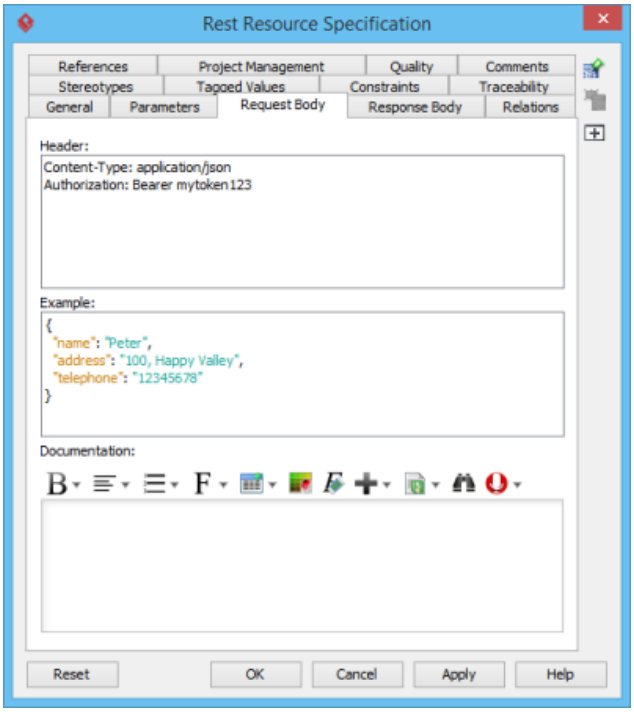


Parameters created

Specifying the Request Header and Request Example

A HTTP message consists of a HTTP request line, a collection of header fields and an optional body. In order for consumers to access a REST Resource, you have to specify the request headers and request (body) example. By doing so, the request header and example will be presented in the generated API documentation. Consumer can then follow the specification in using the service.

1. Right click on the REST Resource and select **Open Specification...** from the popup menu.
2. Open the **Request Body** tab.
3. Enter the **Header**. As we said in the Overview of REST API page that REST is not a standard but architectural style. REST makes use of HTTP standard, so, any REST call header is in fact HTTP header. For more details about headers, read the Headers (HTTP headers) section below.
4. Enter the **Example** in JSON.



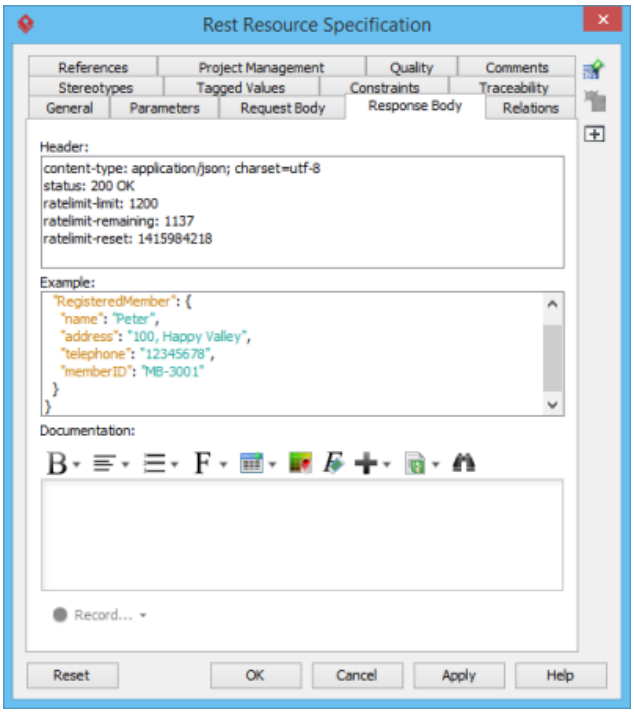
Request header and example specified

5. Click **OK** to confirm the changes.

Specifying the Response Header and Request Example

A HTTP message consists of a HTTP request line, a collection of header fields and an optional body. In order for consumers to access a REST Resource, you have to specify the response headers and response (body) example. By doing so, the response header and example will be presented in the generated API documentation. Consumer can then follow the specification in using the service.

1. Right click on the REST Resource and select **Open Specification...** from the popup menu.
2. Open the **Response Body** tab.
3. Enter the **Header**. As we said in the Overview of REST API page that REST is not a standard but architectural style. REST makes use of HTTP standard, so any REST call header is in fact HTTP header. For more details about headers, read the Headers (HTTP headers) section below.
4. Enter the **Example** in JSON.



Response header and example specified

5. Click **OK** to confirm the changes.

Headers (HTTP headers)

HTTP headers are the core component of any HTTP requests and responses, and they define the operating parameters of any HTTP transactions. When you visit a URL in your web browser, your web browser sends an HTTP request and it may look like this:

```
GET / HTTP/1.1 Host: www.visual-paradigm.com User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:33.0)
Gecko/20100101 Firefox/33.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-
Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Cookie: landing=b7b93a316f374b13af4d5904c9797dcc;
__utma=... Connection: keep-alive Pragma: no-cache Cache-Control: no-cache
```

After the request, your web browser receives an HTTP response and the HTML content to be displayed.

As we said in the Overview of REST API page that REST is not a standard but architectural style. REST makes use of HTTP standard. Therefore, any REST call headers are in fact HTTP headers.

Methods (HTTP methods)

HTTP methods, or sometimes known as HTTP verbs, specify the action to act on a resource. The most commonly used HTTP methods are GET, PUT, POST and DELETE, which correspond to read, update, create and delete operations, respectively.

Method	Description
GET	A GET method (or GET request) is used to retrieve a representation of a resource. It should be used SOLELY for retrieving data and should not alter.
PUT	A PUT method (or PUT request) is used to update a resource. For instance, if you know that a blog post resides at http://www.example.com/blogs/123, you can update this specific post by using the PUT method to put a new resource representation of the post.
POST	A POST method (or POST request) is used to create a resource. For instance, when you want to add a new blog post but have no idea where to store it, you can use the POST method to post it to a URL and let the server decide the URL.
DELETE	A DELETE method (or DELETE request) is used to delete a resource identified by a URI.

Description of different HTTP methods

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

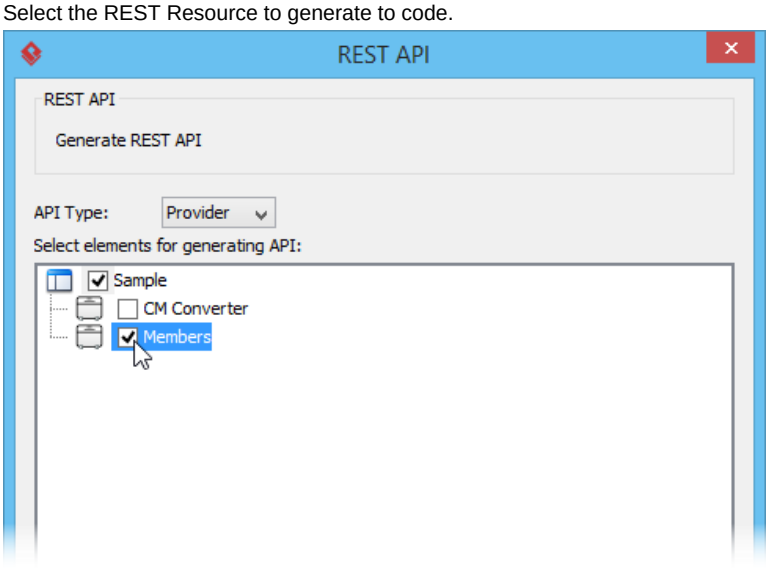
Generating REST API as service provider

Once you have finished the modeling of your REST Resource(s), you can generate the API and optionally, the API documentation.

Generating REST API

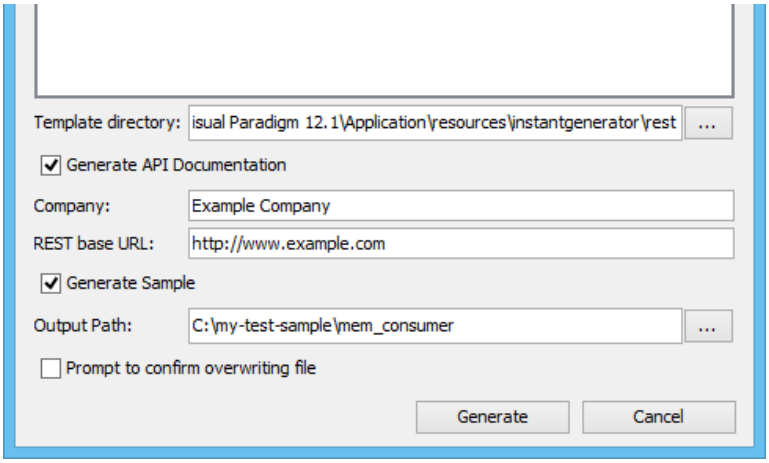
To generate REST API:

1. Select **Tools > Code > Generate REST API...** from the toolbar.
2. In the **REST API** window, keep **Provider** selected for **API Type**. By doing so, you will be able to generate API documentation as well as the server sample code that guides you in programming your service (logic).
3. Select the REST Resource to generate to code.



Select the REST Resource to be generated

4. The generator will use the templates stored in the **Template directory** for code generation. You may edit the templates or select another directory as template directory.
5. Check **Generate API Documentation** to generate the HTML files that shows how to use the selected REST Resource(s). Supposedly, you will publish the generated API documentation in your website so that the consumers of your service can read through it to know how to access for your service.
6. Enter your company name, which will be presented in the API documentation.
7. Enter the base URL of your services.
8. Check **Generate Sample** to generate the source code that teaches you how to program your service. The sample code is rich and informative. Therefore, instead of programming from scratch, we strongly recommend you to generate the sample code and modify its content to fit your needs.
9. Enter the output path of the code.



Output path entered

10. Click **Generate**. Depending on the option checked/unchecked, you may see the following folders in the output directory.

Folder	Description
doc	The API documentation. You should publish the API documentation in your website, so that the consumers of your service can check the documentation to learn the API.
lib	In order for the generated code to work, the Google Gson library must be presented in your class path. To avoid any compatibility issues, please download library manually:

	https://code.google.com/p/google-gson/ and then place the file in the lib folder.
sample_src	The sample code of client and servlet. It shows you how to access as client and how to react to a request as provider. We strongly recommend you to copy the code and modify it by filling in your own service logic.
src	The source code of the communication model. Do not modify the file content or else the code may not be able to function properly.

Description of generated files

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

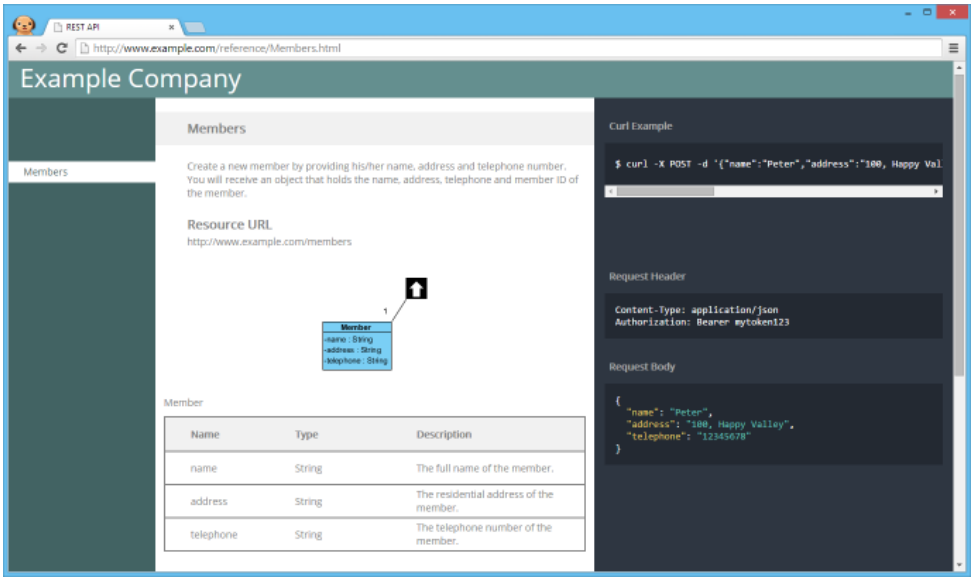
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using REST API (as a service consumer)

Consumers of RESTful service have to go through a series of steps in order to obtain the API code required for accessing a REST Resource. This page will show you the steps in detail.

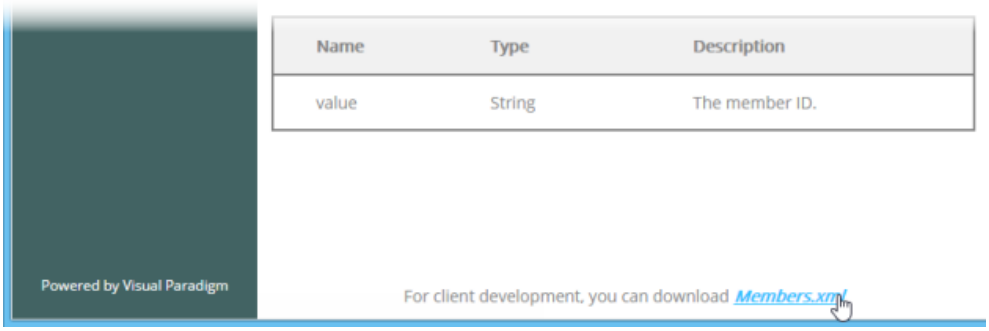
To obtain and use the REST API code as a service consumer:

1. Visit the API documentation of the service published by service provider. The API documentation should look like this:



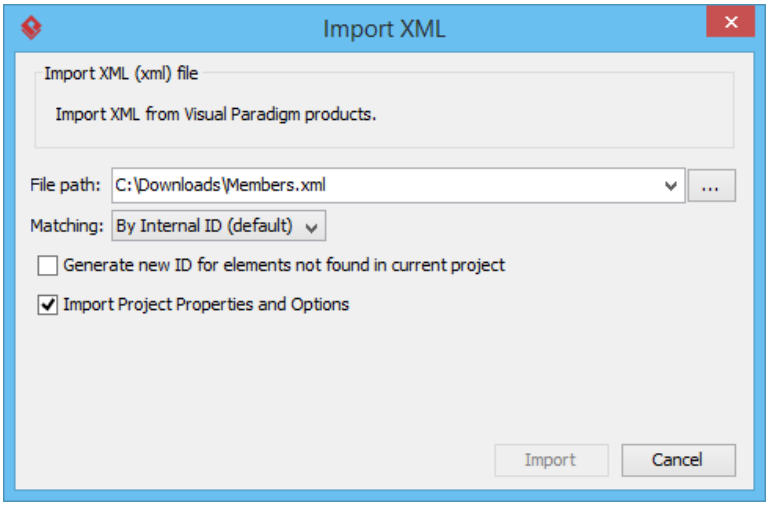
REST API documentation

2. You can learn the usage of the REST resource by reading through the API documentation. To obtain the API code, scroll down to the bottom of the API documentation. Click on the download link of the REST API model XML file at the bottom of the page.



Download REST API model XML

3. Download Visual Paradigm . Install and run it.
4. Start Visual Paradigm.
5. Import the REST API model XML file into Visual Paradigm by selecting **Project > Import > XML...** from the toolbar.
6. In the **Import XML** window, enter the file path of the XML file and click **Import**.

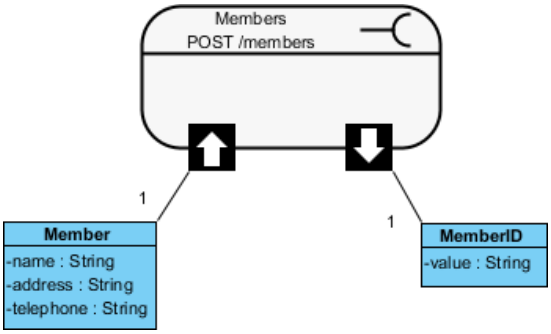


The Import XML window

7. In the **Diagrams** tab of the **Project Browser**, double click on the class diagram created by importing the XML file.

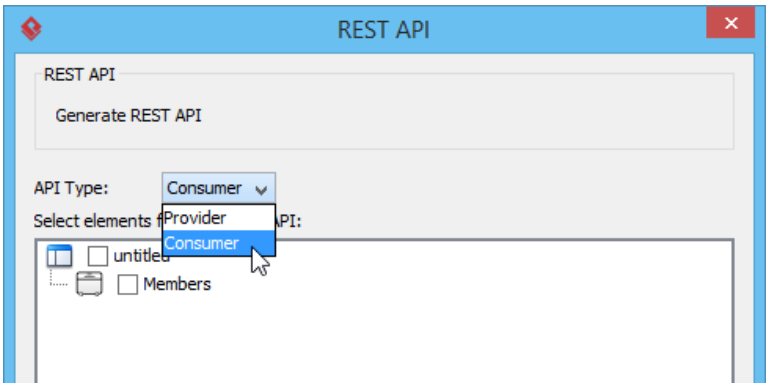
Open the class diagram

8. You can now see the communication model of the REST Resoure which looks like this:



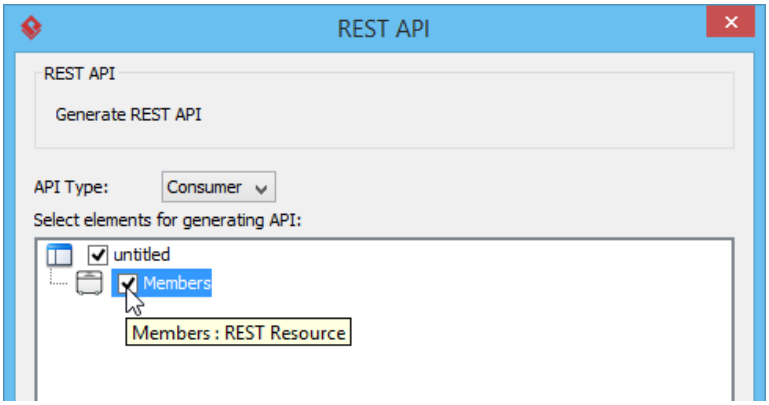
The communication model

9. Now, you can generate the API code. Select **Tools > Code > Generate REST API...** from the toolbar.
10. In the **REST API** window, select **Consumer** as **API Type**.



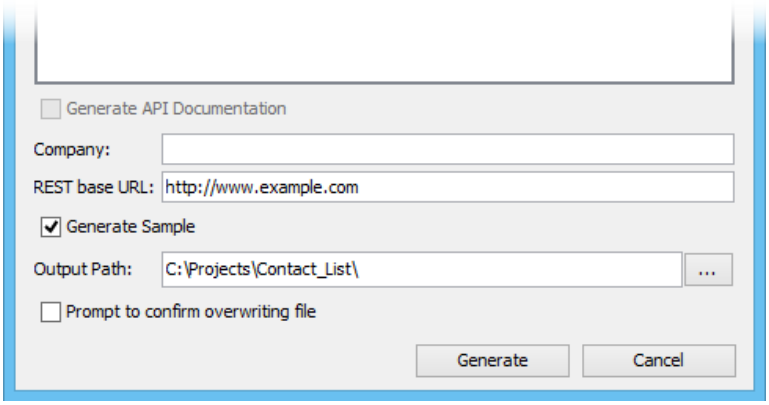
Select Consumer to be API Type

11. Select the REST Resource to generate code.



Select the REST Resource to be generated

12. Skip the **Company** field as you don't really need it in programming.
13. Enter the base URL of the service.
14. Check **Generate Sample** to generate the source code that teaches you how to access the service. The sample code is rich and informative. Therefore, instead of programming from scratch, we strongly recommend you to generate the sample code and modify its content to fit your needs.
15. Enter the output path of the code.



Output path entered

16. Click **Generate**. Depending on the option checked/unchecked, you may see the following folders in the output directory.

Folder	Description
lib	In order for the generated code to work, the Google Gson library must be presented in your class path. To avoid any compatibility issues, please download library manually: https://code.google.com/p/google-gson/ and then place the file in the lib folder.
sample_src	The sample code that shows you how to access the service. We strongly recommend you copy the code and modify it by filling in your own application logic.
src	The source code of the communication model. Do not modify the file content or else the code may not function properly.

Description of generated files

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Introduction of Database Design and Management Features

Overview of Database Design & Management with Visual Paradigm

Check out how Visual Paradigm supports your database design and management needs.

Benefits of Designing Database with Visual Paradigm

In this page you will see a list of key benefits of using Visual Paradigm in designing database.

Database Configuration in Visual Paradigm

Database configuration is to setup the connection details required to connect to your database from Visual Paradigm. In this page you will learn the properties you have to set in database configuration.

Sharing Database Configuration between Projects

See how to share database configuration from project to project through the export and import feature.

Supported Database, JDBC Drivers and .NET Drivers

Check out the list of databases supported in data modeling, database generation and database reversal.

Overview of Database Design & Management with Visual Paradigm

Software applications are most likely to be developed with a database such that all data working with the application system can be retained, resulting in information and knowledge. Hence, database application is widely adopted by businesses of all sizes. In order to access and manipulate the relational database, a standard computer language, Structured Query Language (SQL) has to be used. The use of SQL statements was nearly the only choice when developing database application.

Taking a trading system as an example, if the end-user wants to update a Sales Order record, the system has to retrieve the corresponding record from the Sales Order table and display to the end-user. After the end-user confirms the modification of data, the system has to update the record accordingly. It is noticeable that a database application requires a lot of coding for handling SQL statements so as to access and manipulate the database.

Hence, it is inevitable that developers spend almost 50% of development time for implementing the code with SQL statement. Moreover, mapping between the persistent code and database table is maintained throughout the development life cycle. Once there is a change in the structure of a database table, SQL statements which related to the modified table have to be re-written. Developers have to keep an eye on every change in the database schema.

Visual Paradigm provides a solution to develop database application. Visual Paradigm features Object-Relational Mapping tool which provides an ease-to-use environment bridging between object model, data model and relational database. Visual Paradigm not only provides you a visual modeling for both logical data design and physical data design, but also automates the mapping between object model and data model.

Visual Paradigm generates not only Java and .NET persistent code, but also a cost-effective, reliable, scalable and high-performance object to relational mapping layer. The generated mapping layer includes the support of transaction, cache and other optimized feature. Visual Paradigm increases the productivity and significantly reduces the risk of developing the mapping layer manually.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Benefits of Designing Database with Visual Paradigm

Visual Paradigm provides the following key features so as to help you simplify your application development:

Persistance Made Easy

Traditionally developers spend a lot of effort in saving and loading objects between memory and database which makes the program complicated and difficult to maintain. Visual Paradigm simplifies these tasks by generating a persistence layer between object and data models.

Sophisticated Object-Relational Mapping Generator

Visual Paradigm generates object-relational mapping layer which incorporates prime features such as transaction support, pluggable cache layer, connection pool and customizable SQL statement. With this mapping layer, developers can keep away from mundane implementation work and focus on the business requirements.

Model Driven Development

Visual Paradigm provides a true model driven platform for application development. Visual Paradigm allows developer not only to start from creating the models by using Class Diagram or Entity Relationship Diagram to generating the executable persistence layer from the models, but also to modify the entity-relational model which comes from reverse engineering an existing database, transform into object model and generate persistence layer. With the sophisticated model-code generator, the persistent model will be updated automatically according to any modification.

Extensive Database Coverage

Visual Paradigm supports a wide range of database, including Oracle, DB2, Cloudscape/Derby, Sybase Adaptive Server Enterprise, Sybase SQL Anywhere, Microsoft SQL Server, PostgreSQL, MySQL and more. Visual Paradigm also promotes an easy migration between databases by enabling the same set of ORM Java objects to work with different databases and transforms the proprietary data type that suit the default database specified.

Database Reverse Engineering

Visual Paradigm allows you to reverse engineering an existing database through JDBC into the entity-relational model. Developers can transform the entity-relational model to object model and redesign the database for further development

Class Reverse Engineering

Visual Paradigm allows you to reverse engineering Java classes and Hibernate models into the persistent object model. Developers can transform the persistent object model to data model and redesign the models for further development.

IDE Integration

Visual Paradigm is not only a standalone application, but also integrated to the major Integrated Development Environments (IDEs), including Eclipse, NetBeans, IntelliJ IDEA and Visual Studio .NET, which results in streamlining the entire model code- deploy software development process.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

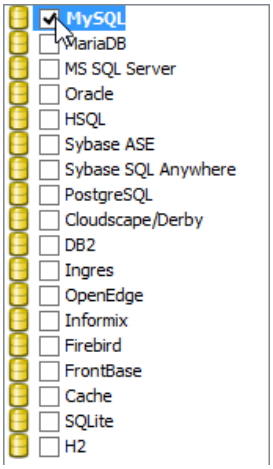
Database Configuration in Visual Paradigm

Database configuration is to setup the connection details required to connect to your database from Visual Paradigm. Database configuration is required because:

- It allows you to use DBMS specific data types in your ER model
- Visual Paradigm has to access the target database when you perform database generation

To configure database:

1. Select **Tools > DB > Database Configuration...** from the toolbar.
2. At the top left corner of the **Database Configuration** window, set the (Programming) **Language** of your project.
3. Check all the DBMS(s) you use from the list of vendor on the left hand side. When you are you drawing an Entity Relationship Diagram (ERD), you will be able to select data types supported by the DBMS selected here.



Select DBMS

4. If you use multiple DBMS, you have to select the default DBMS by right clicking on the DBMS and selecting **Set as default** from the popup menu. Database and DDL generation follow the default DBMS selection made here.
5. Enter the **Driver Setting**.

Property	Description
Version	There might be a slightly difference in database setting for different database versions. In order for the database connectivity to work properly, choose the version of your database here.
Driver	A driver is a software component essential for us to connect with your database. There are different driver suppliers on the market. Choose the one you like or leave it as default if you are uncertain.
Driver file	JDBC is a popular option in database connectivity. We use JDBC in connecting with database. In order for DB generation to function, you have to specify the JDBC driver here. It can be downloaded automatically by clicking  . If it failed to find one, you will be redirected to the download URL.
Connection URL	Enter the information required for the JDBC driver to connect to the database.
User	The username of the user who has the access right to the database.
Password	The corresponding password for the user input in the **User** field.
Engine (MySQL and Maria DB only)	MySQL and MariaDB supports different types of table engines. The two most commonly used engines are **Innodb** and **MyISAM**.

Driver settings

6. Click **Test Connection**. If the database settings are confirmed alright, you will be prompted yb a dialog box showing **Connect Successful**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

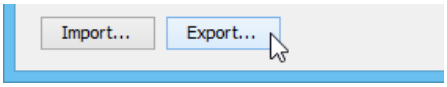
- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Sharing Database Configuration between Projects

Export and import database configuration allows you to reuse the same database configuration across projects, without the needs to re-define the configuration in each project.

To export database configuration:

1. Select **Tools > DB > Database Conguration...** from the toolbar to open **Database Configuration** window.
2. Click the **Export...** button at the bottom left of the window.



Export database configuration

3. The configuration will be saved in an XML file. Enter the filename of the XML in the **Save** window and click **Save** to export it.

To import database configuration:

1. In the target project, select **Tools > DB > Database Conguration...** from the toolbar to open **Database Configuration** window.
2. Click the **Import...** button at the bottom left of the window.
3. Select the XML exported before and click **Open** to select the file and apply the configuration.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Database, JDBC Drivers and .NET Drivers

Visual Paradigm provides a visual modeling environment for modeling software system. By connecting the relational database to Visual Paradigm, Visual Paradigm can automate the mapping between models and relational database. Visual Paradigm supports the most common relational databases, including Oracle, DB2, Microsoft SQL Server, Sybase Adaptive Server Enterprise, Sybase SQL Anywhere, MySQL, HSQLDB, Cloudscape/ Derby and PostgreSQL. Their relative JDBC Drivers and .NET Drivers are listed in the following tables.

In order to connect to any of the supported database, the relative JDBC and .NET Drivers are required for configuring the database connection. All of the required JDBC and .Net Drivers will not be bundled with Visual Paradigm. Yet, you can get the driver files by using the automatic download function provided, or to have them manually downloaded.

Table shows the supported DBMS and their corresponding JDBC Drivers.

DBMS	JDBC Driver Name
Cache	InterSystems Cache
Cloudscape/Derby 10.6.2.1 or lower	Cloudscape/Derby (Embedded) Cloudscape/Derby (Server)
Cloudscape/Derby 10.7 or above	Derby 10.7 (Embedded) Derby 10.7 (Server)
DB2 7/8	DB2 (Type 4 Driver) DB2 (App Driver) DB2 (Net Driver) DB2 (AS/400 Toolbar for Java JDBC Driver) DB2 (DataDirect Connect Driver)
Firebird	Firebird
FrontBase	FrontBase
H2	H2
HSQLDB 1.61 - 1.8	HSQLDB (In-process) HSQLDB (Server)
IBM Informix	IBM Informix (Client) IBM Informix (Server) Informix (DataDirect Connect Driver)
Ingres	Ingres
MariaDB	MariaDB
MS SQL Server 2005 or lower	MS SQL Server (jTDS Driver) MS SQL Server (Microsoft Driver) MS SQL Server 2005 (Microsoft Driver) MS SQL Server (SequeLink Driver) MS SQL Server (DataDirect Connect Driver) MS SQL Server (WebSphere Connect Driver) MS SQL Server (JSQL Driver) MS SQL Server (JTURBO Driver) MS SQL Server (Weblogic Connect Driver)
MS SQL Server 2008 or higher	MS SQL Server (jTDS Driver) MS SQL Server 2005 (Microsoft Driver) MS SQL Server (SequeLink Driver) MS SQL Server (DataDirect Connect Driver) MS SQL Server (WebSphere Connect Driver) MS SQL Server (JSQL Driver) MS SQL Server (JTURBO Driver) MS SQL Server (Weblogic Connect Driver)
MySQL 5.0.4 or lower	MySQL (Connector/J Driver) MySQL 5 (Connector/J Driver)
MySQL 5.0.5 or higher	MySQL 5 (Connector/J Driver)
OpenEdge	OpenEdge
Oracle	Oracle (DataDirect SequeLink Driver)
Oracle 8i	Oracle8i (THIN JDBC Driver)
Oracle 9i	Oracle9i/10g (or above)
PostgreSQL	PostgreSQL
SQLite	SQLite

Sybase Adaptive Server Enterprise 12.5	Sybase Adapter Server Enterprise (jConnect Driver) Sybase Adapter Server Enterprise (JTDS Driver) Sybase (DataDirect Driver)
Sybase SQL Anywhere 9	Sybase SQL Anywhere (jConnect Driver)

Supported DBMS and their corresponding JDBC Drivers

Table shows the supported DBMS and their corresponding .NET Drivers.

DBMS	.NET Driver Name
Cache	InterSystems Cache
DB2	DB2 (IBM DB2 Data Provider)
Firebird	Firebird (Firebird .NET Data Provider) Firebird (Firebird Client)
Ingres	Ingres
MS SQL Server 2005 or lower	MS SQL Server
MS SQL Server 2005 or higher	MS SQL Server
MySQL 5.0.4 or lower	MySQL (MySQL Connector/Net 1.0) MySQL (MySQL Connector/Net 6.1)
MySQL 5.0.5 or higher	MySQL (MySQL Connector/Net 1.0) MySQL (MySQL Connector/Net 6.1)
Oracle	Oracle (.NET Framework Data Provider) Oracle (Oracle Data Provider for .NET)
PostgreSQL	Postgre (Npgsql)
SQLite	SQLite (SQLite .NET Data Provider)

Supported DBMS and their corresponding .NET Drivers

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Visual Paradigm Database Designer Guides

Drawing Entity Relationship Diagram

Learn how to draw entity, how to add column and how to create relationship between entities.

Conceptual, Logical and Physical Data Model

Conceptual, logical and physical model are three different ways of modeling data in a domain. In this page you will learn what they are and how to transit from one model to another.

Using Entity Domain

An entity domain is a named set of property values that can be re-used by an entity. In this article you will learn how to define and use entity domain.

Using Column Domain

A column domain is a named set of property values that can be re-used by a column. In this article you will learn how to define and use column domain.

Applying Default Schema on ERD

A default schema can be used to easily move entities on an ERD to a common schema. You will know how it works by reading this page.

Using Auto Column

Auto Column provides you with a quick way in creating entities in ERD. This page describes how to configure auto column and how to use it in creating entities in an ERD.

Entering Sample Table Records for Entities

Enter sample table records in ERD, and have the records exported to your database in database generation. This article shows you how it works.

Modeling Database View

A database view is the result of a query on the data stored in a database. This page describes how to create a database view.

Modeling Stored Procedures in ERD

In Visual Paradigm, stored procedure is modeled in form of a procedure container. This article shows you how to create stored procedure.

Modeling Triggers in ERD

In Visual Paradigm, trigger is modeled in form of a trigger container. This article shows you how to create trigger.

Working with Unique Constraint

Unique constraints help to enforce the uniqueness of specific columns. In this page, you will see how to create unique constraints for entities in ERD.

Using ID Generator

Some databases support controlling how primary key value can be generated, through the use of an ID generator. In this article you will learn how to select an ID generator for a primary key.

Different Inheritance Strategies

Inheritance strategy is used to define the way how entities should be created and structured to respect an object model. You will learn what inheritance strategy is and how to set it in this page.

Using Discriminator Column

Discriminator column contains a unique value that can be used for identifying the entity in which a table record belongs to. In this article you will learn what discriminator column is.

Using Array Table

Retrieve objects in the form of primitive array, instead of a collection when handling a data column with cardinality of many. This article shows you how Array Table works in Visual Paradigm.

Using Partial Table

Optimize the size of database, and minimizes the redundant persistent classes for handling one-to-one identifying relationship. In this page, you will learn how to work with partial table.

Drawing Entity Relationship Diagram

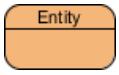
ERD, short form for Entity Relationship diagram is a kind of diagram for presenting the properties as well as the relationships between data or participants. Database designer uses ERD to model physical structure of a relational database, while business analyst uses ERD to model the data that is logically required or produced by processes. In this page you will learn how to draw entity, how to add column and how to create relationship between entities.

Creating Entity Relationship Diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Entity Relationship Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.
6. This creates an Entity Relationship Diagram. At the top right corner of the diagram, select the Data Model. All entities created in this diagram will be set to the chosen data model. And note that only entities in physical model will be included in generating database/DDL.

Drawing an entity

To draw an entity, select from the diagram toolbar and then click on the diagram. An entity will be created.

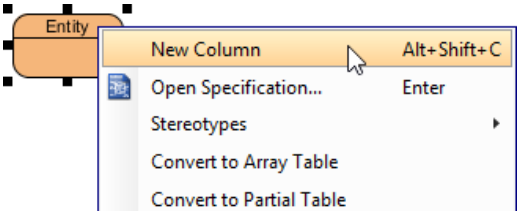


Entity created

Adding column into entity

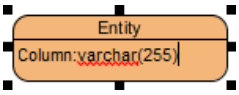
To add column into entity:

1. Right click ont he entity and select **New Column** from the popup menu.



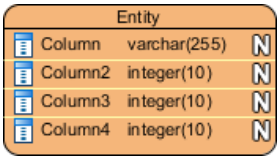
To create a new column

2. A column is added. Enter its name in the pattern *COL_NAME : COL_TYPE* where *COL_TYPE* is the data type of column.



Naming a new column

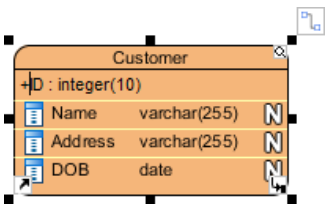
3. Press **Enter** to confirm.
4. Now, you can repeat step 2 and 3 to add more columns. When finished editing, press **Esc** to confirm.



Columns created

Specifying primary key

There are several ways you can take to specify a column as a primary key. When inline editing, you can type + before the column name to indicate that the column is a primary key column.



Specifying a primary key

Alternatively, right click on a column and select **Include in Primary Key** to set the column as primary key or include it as part of a composite key. Finally, you can also find and check the **Include in Primary Key** option in the **Column Specification** window. To open the window, right click on a column and select **Open Specification...** from the popup menu.

Clustered and non-clustered primary key

The use of clustered primary key may make the querying of data more efficient. To make a primary key of an entity a clustered/non-clustered primary key:

1. Right click on that entity and select **Open Specification...** from the popup menu.
2. Open the **Columns** tab.
3. Select **Clustered**/ **Non-Clustered** for **Primary key clustered**.
4. Click **OK**.

Hiding nullable icons in ERD

In case you want to hide the nullable icon (as represented by symbol N) in ERD, you can follow the steps below: Right click on the diagram > **Presentation Options > Entity Columns Display Options> Column Constraints Presentation Option>** uncheck **Show Nullable**.

Selecting all columns in an entity

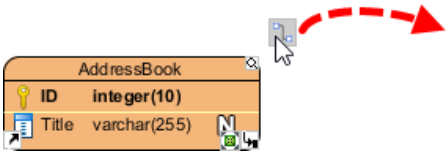
To select all columns within an entity, select any column first, and then press **Ctrl-A** to select the rest.

Working with relationships

Creating an entity with relationship

Relationship shows how the entities are related to each other. You can create a related entity by performing the steps below:

1. Move your mouse pointer over the source entity.
2. Press on the **Resource Catalog** button and drag it out.

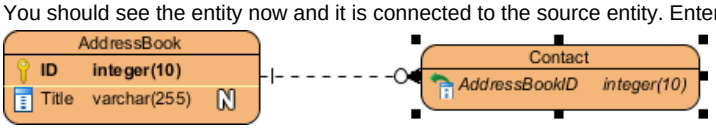


Using Resource Catalog

3. Release the mouse button at the place where you want the entity to be created.
4. In Resource Catalog, select the kind of relationship to be created. If you want to create an entity with a one-to-many relationship, select **One-to-Many Relationship -> Entity**.

To create an entity

5. You should see the entity now and it is connected to the source entity. Enter its name and press **Enter** to confirm editing.

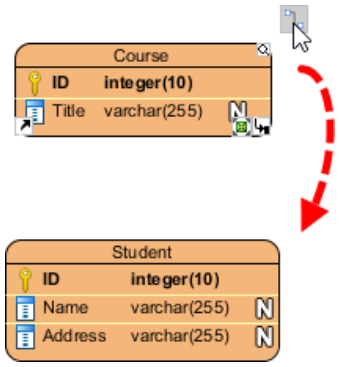


Entity created

Connecting to existing entity

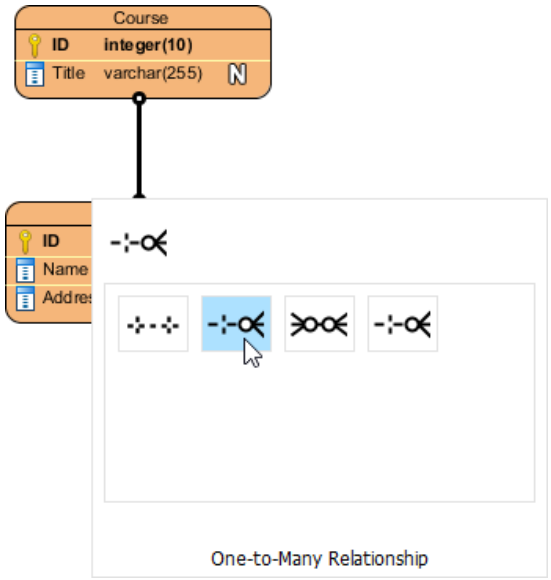
To connect to en existing entity:

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.



Using Resource Catalog

3. Release the mouse button at the target entity.
4. In Resource Catalog, select the kind of relationship to be created.

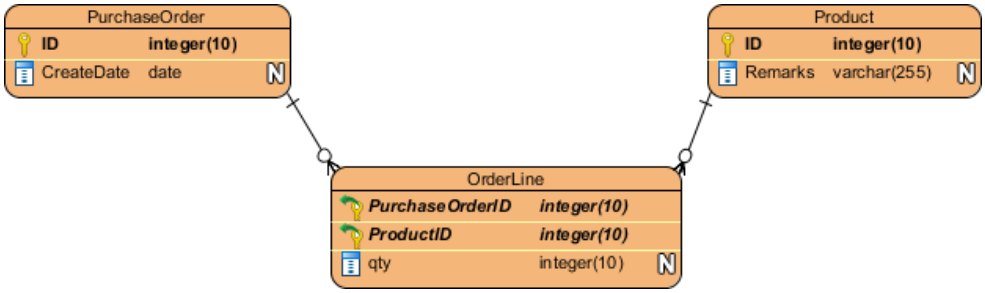


To create a one-to-many relationship between entities

The entities are now connected with the relationship you chose.

Linked entity in many-to-many relationship

When you create a many-to-many relationship, a linked entity will be created, with two one-to-many relationships connected to it from the source entities.



Linked entity

Identifying and non-identifying relationships

There are two types of relationships - identifying and non-identifying.

Identifying relationship specifies the part-of-whole relationship. It means that the child instance cannot exist without the parent instance. Once the parent instance is destroyed, the child instance becomes meaningless.

Non-identifying relationship implies weak dependency relationship between parent and child entities. There are two kinds of non-identifying relationships, including optional and mandatory. The necessity of the parent entity is "exactly one" and "zero or one" in the mandatory and optional non-identifying relationship respectively.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Knowhow - Using business key in ERD
- FREE Online Training - Database Design and Management
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

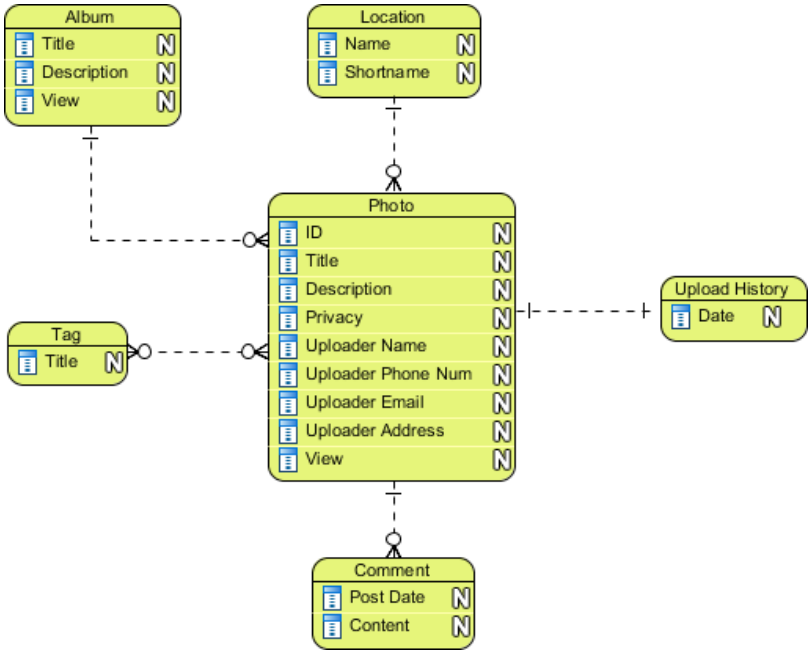
Conceptual, Logical and Physical Data Model

Conceptual, logical and physical model are three different ways of modeling data in a domain. While they all contain entities and relationships, they differ in the purposes they are created for and audiences they are meant to target. A general understanding to the three models is that, business analyst uses conceptual and logical model for modeling the data required and produced by system from a business angle, while database designer refines the early design to produce the physical model for presenting physical database structure ready for database construction.

With Visual Paradigm, you can draw the three types of model, plus to progress through models through the use of Model Transitor.

Conceptual Model

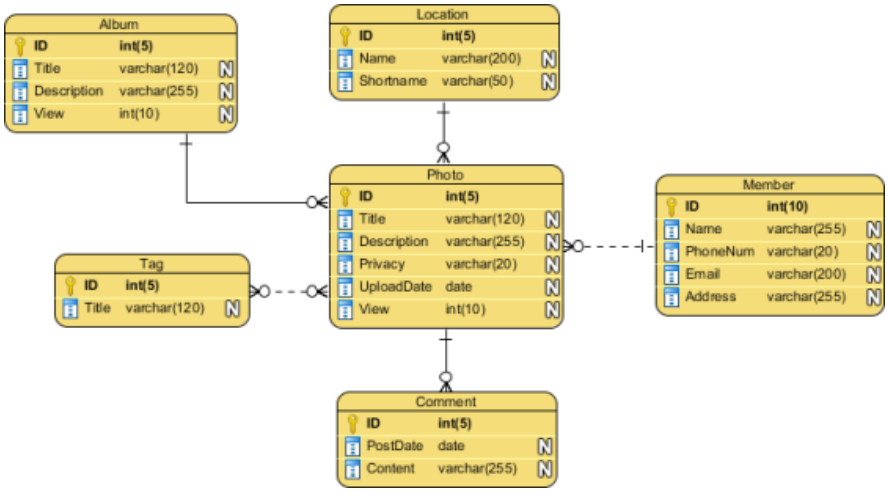
Conceptual ERD models information gathered from business requirements. Entities and relationships modeled in such ERD are defined around the business's need. The need of satisfying the database design is not considered yet. Conceptual ERD is the simplest model among all.



Conceptual ERD example

Logical Model

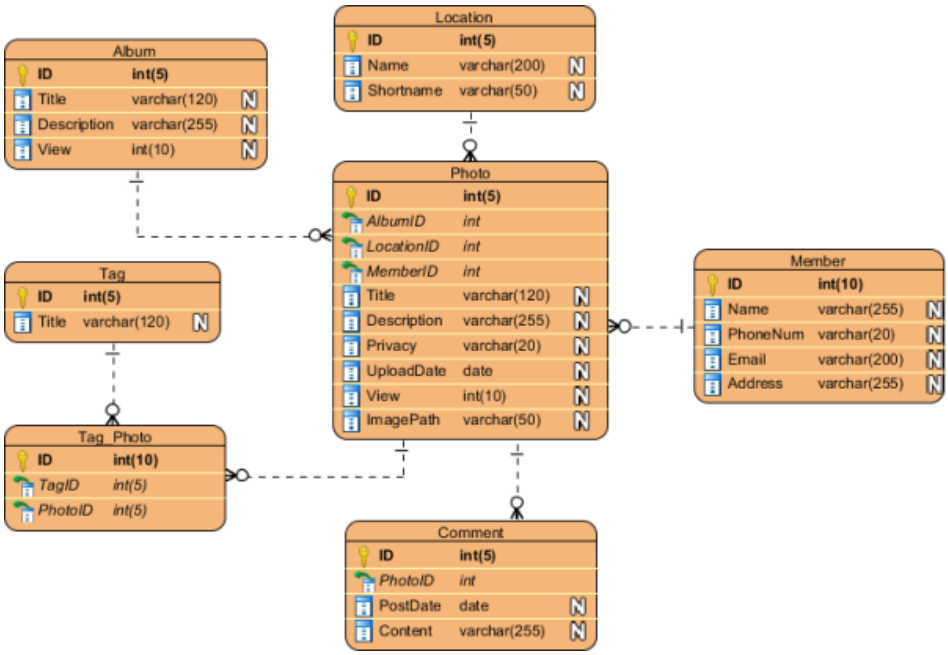
Logical ERD also models information gathered from business requirements. It is more complex than conceptual model in that column types are set. Note that the setting of column types is optional and if you do that, you should be doing that to aid business analysis. It has nothing to do with database creation yet.



Logical ERD example

Physical Model

Physical ERD represents the actual design blueprint of a relational database. It represents how data should be structured and related in a specific DBMS so it is important to consider the convention and restriction of the DBMS you use when you are designing a physical ERD. This means that an accurate use of data type is needed for entity columns and the use of reserved words has to be avoided in naming entities and columns. Besides, database designers may also add primary keys, foreign keys and constraints to the design.



Physical ERD example

Transiting from Conceptual/Logical to Physical ERD

Model Transitor enables you to transit a logical ERD to a physical ERD and with the transition relationship maintained. To transit, right click on the background of your conceptual/logical ERD, and then select **Utilities > Transit to Logical/Physical ERD...** from the popup menu. This will form a new ERD with new entities there. You can make changes like to rename the entities and columns, or to add extra entities in the new ERD.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using Entity Domain

If you create ERD for serious database development purpose, you may need to specify a lot of details for entities in addition to their names. For example, the schema the entities belong to, their tablespace, DDL clauses, description, indices, etc.

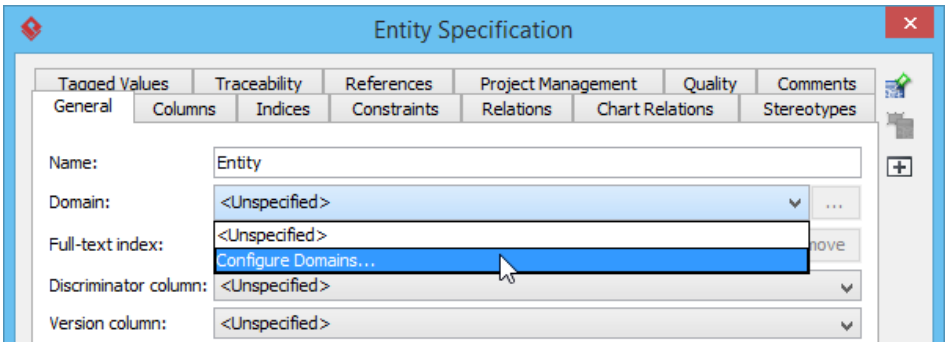
To specify these properties can be a time consuming task. Visual Paradigm supports the use of entity domain in re-using specification details between entities.

Each domain is a named set of property values that can be re-used by an entity. You can create a domain for commonly entered values such as schema, description template and default column assignment, and re-use these setting in other entities when needed. This saves you time in creating similar entities as well as to keep your data model consistent.

Defining an entity domain

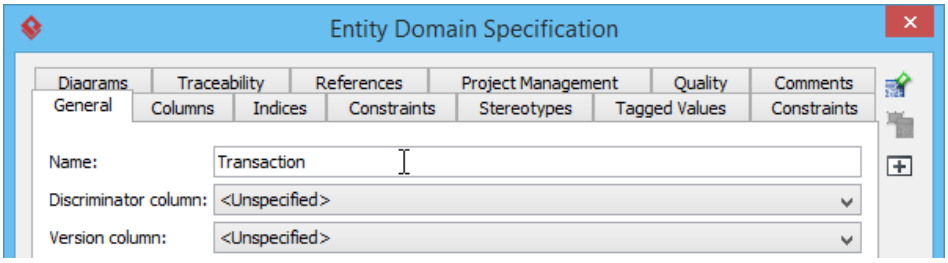
You can define an entity domain from the **Entity Specification** window of any entity. To define an entity domain:

1. Right click on any entity and select **Open Specification...** from the popup menu.
2. Click on the drop down menu of the **Domain** row and select **Configure Domains...** from the drop-down menu.



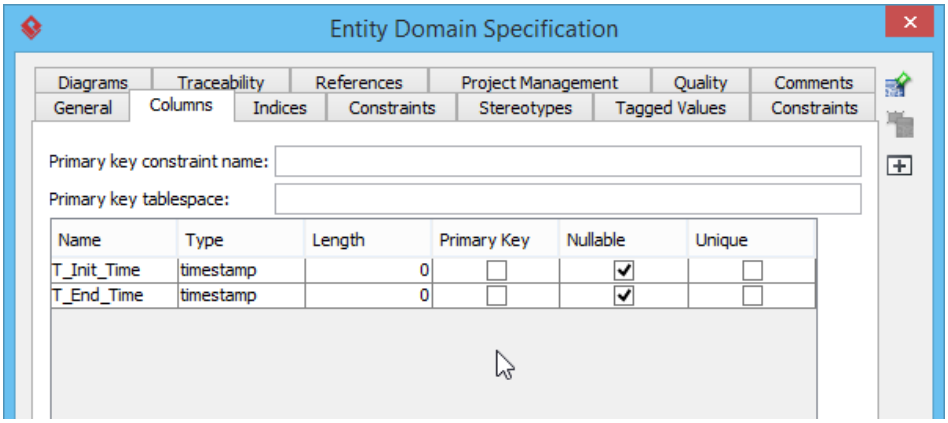
Configure domains

3. In the **Configure Entity Domains** window, click **Add...**.
4. Enter the name of the domain.



Entering the name of entity domain

5. Specify the properties. The properties you specified will be re-used by entities that use this domain.



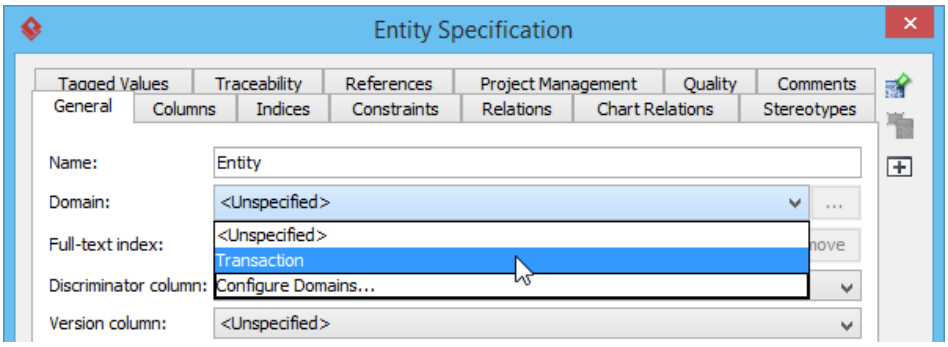
Specifying the properties of the domain

6. Click **OK** to confirm.

Using an entity domain

To use an entity domain:

1. Open the specification of the entity to which you want use a domain. You can open entity specification by right clicking on that entity and selecting **Open Specification...** from the popup menu.
2. Click on the drop down menu of the **Domain** row.
3. Select the domain to use from the drop-down menu.



Selecting an entity domain

4. Click **OK** to confirm. By doing so the entity will be updated to apply the property values specified in the domain.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using Column Domain

If you create ERD for serious database development purpose, you may need to specify a lot of details for entity columns in addition to their names. For example, the type, length, default value, etc.

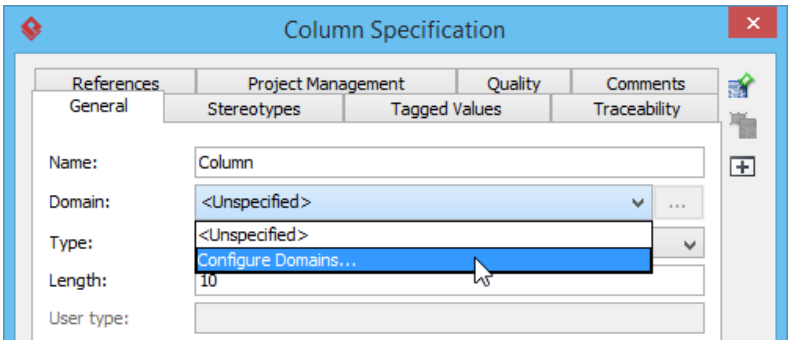
To specify these properties can be a time consuming task. Visual Paradigm supports the use of column domain in re-using specification details between columns.

Each domain is a named set of property values that can be re-used by a column. You can create a domain for commonly entered values such as type, length and default value, and re-use these setting in other columns when needed. This saves you time in creating similar columns as well as to keep your data model consistent.

Defining a column domain

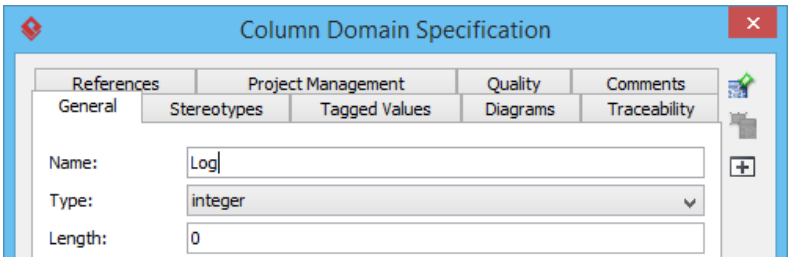
You can define a column domain from the **Column Specification** window of any column. To define a column domain:

1. Right click on any column and select **Open Specification...** from the popup menu.
2. Click on the drop down menu of the **Domain** row and select **Configure Domains...** from the drop-down menu.



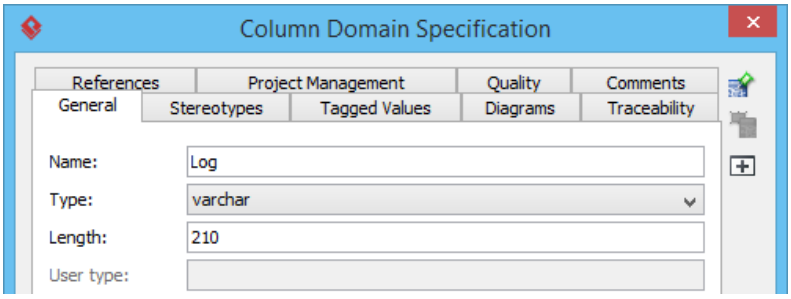
Configure domains

3. In the **Configure Column Domains** window, click **Add...**.
4. Enter the name of the domain.



Entering the name of column domain

5. Specify the properties. The properties you specified will be re-used by columns that use this domain.



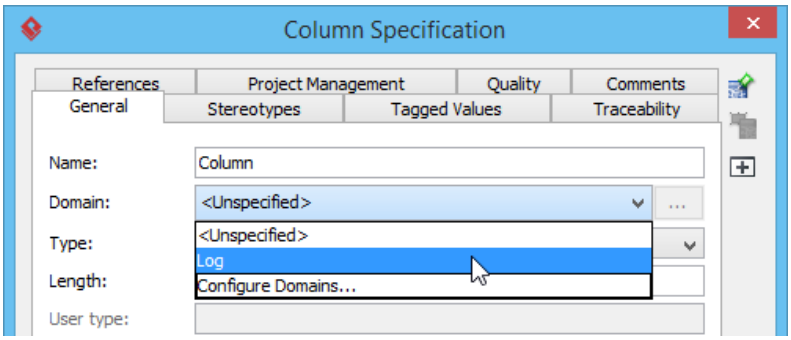
Specifying the properties of the domain

6. Click **OK** to confirm.

Using a column domain

To use a column domain:

1. Open the specification of the column to which you want use a domain. You can open column specification by right clicking on that column and selecting **Open Specification...** from the popup menu.
2. Click on the drop down menu of the **Domain** row.
3. Select the domain to use from the drop-down menu.



Selecting an column domain

4. Click **OK** to confirm. By doing so the column will be updated to apply the property values specified in the domain.

Related Resources

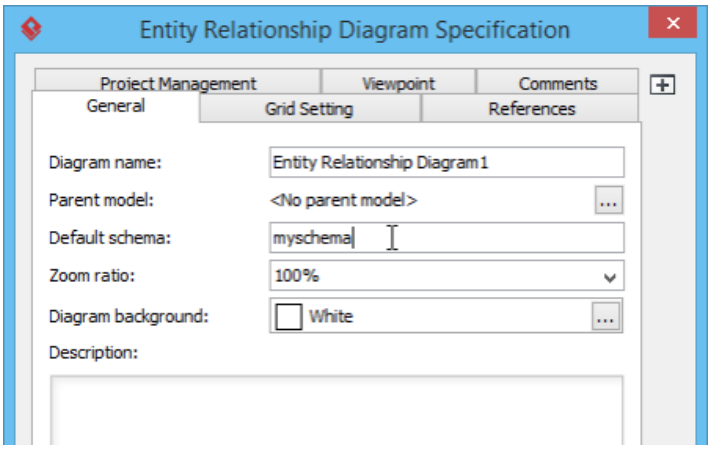
The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Applying Default Schema on ERD

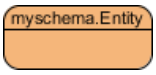
A default schema can be set on diagram. Entities to be created on the diagram will all share the same schema. To define a default schema on diagram:

1. Right click on the diagram background and select **Open Specification...** from the pop-up menu.
2. Enter the default schema in the **Entity Relationship Diagram Specification** window.



Entering default schema

3. Click **OK** to confirm the change. Now, when you create an entity on the diagram, the entity will be under the default schema.



An entity with myschema as its schema

In order to move existing entities in a diagram to a schema, define default schema on the Entity Relationship Diagram first, then right click on the diagram and select **Apply Default Schema** from the pop-up menu. This will make all entities that have master view on the diagram to move to the schema defined.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

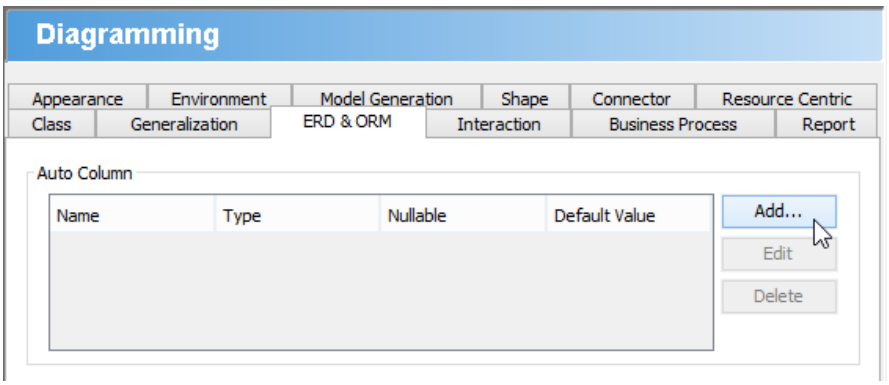
Using Auto Column

Auto Column provides you with a quick way in creating entities in ERD. Just by entering the name of a column when you create it, its type, nullable and default value will be specified, based on the pre-defined column schema.

Configuring Auto Column

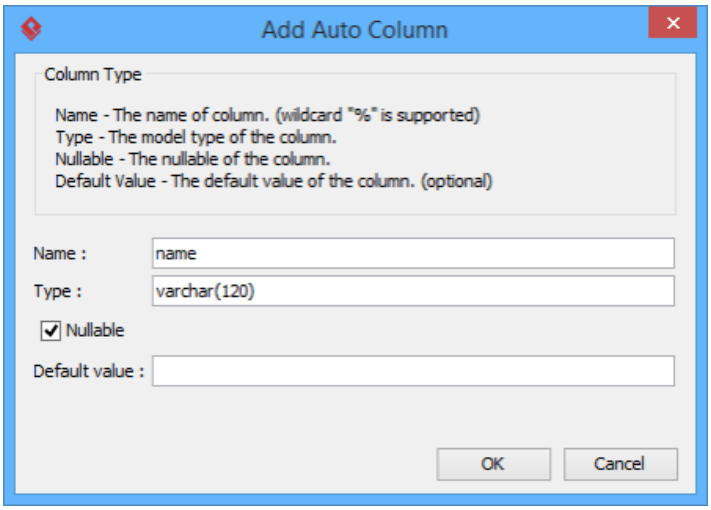
You need to configure the auto columns in order to use them in ERD. To configure auto columns:

1. Select **Windows > Application Options** from the toolbar.
2. In the **Application Options** window, open **Diagramming > ERD & ORM**.
3. Click **Add...**.



Add an auto column

4. Define the auto column by entering its name, type, nullable and default value. When you create columns by using the name specified, the type, nullable and default value will follow the definition here. Note that you can specify the length of datatype when specifying **Type**.



Defining an auto column

5. Click **OK** to confirm.
6. Click **OK** again to close the **Application Options** window.

Pattern matching using wildcard

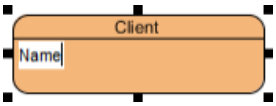
Sometimes, columns are named based on the entities they are in. For example, for entity, *Teacher*, there may be a column named *TeacherID* and for entity, *Student,* there may be another column named *StudentID*. Although these columns are named differently, they may share the same type, nullable and default values. In such case, you can make use of pattern matching in defining an auto column.

When naming an auto column, you can use the '**%**' character to indicate the occurrence of ANY character. For example, naming an auto column *%Name* will result in causing *TeacherName* and *StudentName* apply the definition specified.

Using pattern matching in defining an auto column

Adding a column to entity with Auto Column

When you create a column in entity, give it a name that follows an Auto Column defined. When you confirm editing, the type, nullable and default values will be automatically set for you. Note that if you enter a type yourself, Auto Column will NOT function.



Naming a column

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

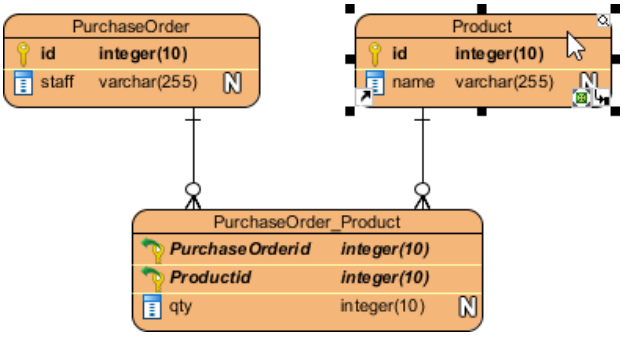
Entering Sample Table Records for Entities

Visual Paradigm allows you to design your database with entity relationship diagram (ERD) and finally generate the design to database as database schema. On top of the schema that will be generated, you can also specify default data to add to database upon database generation. Please note that although you can specify default data for conceptual, logical and physical data model, only entities under physical data model will be considered in database generation.

Entering default data

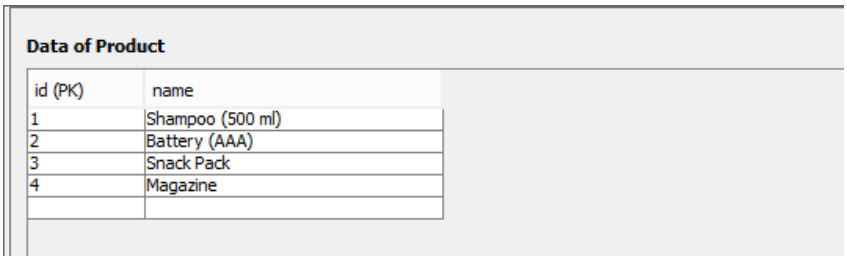
Default data has to be entered in the table record editor. To enable the editor, right click on the background of your ERD and select **Show Table Record Editor or View Editor** from the popup menu.

Then, select in diagram the entity you want to enter its default data.



Selecting entity

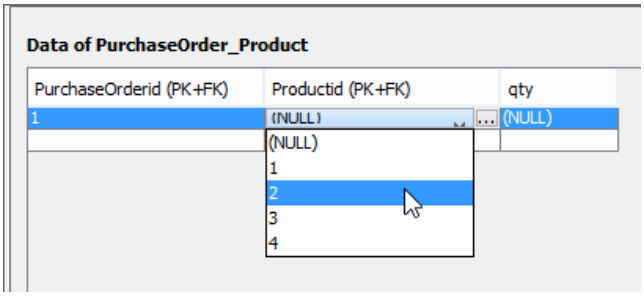
Enter the data in the editor. Double click on a cell to start editing. Click **Enter** to end editing.



Edited sample data

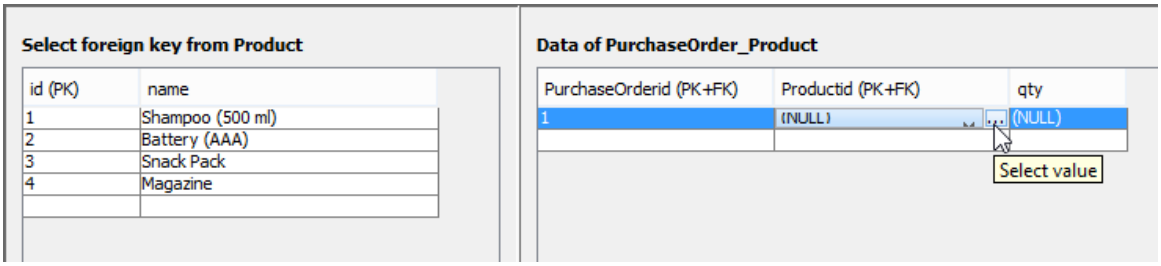
Selecting foreign key value

To select a foreign key value, click on the cell to popup the drop down menu and select the value from the menu.



Selecting FK value

Sometimes, you may be uncertain to what the foreign values represent. You can click on the **...** button to show the additional pane and select the proper record from the pane.



Selecting record

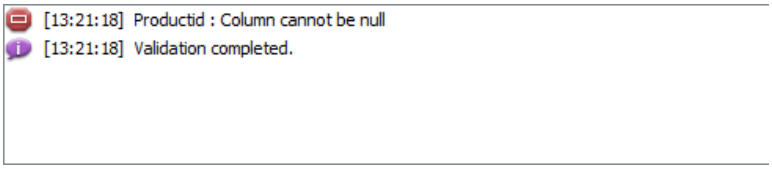
Removing a record

To remove a record, select the record you want to remove and click  at the top right of the editor.

Validating records

Record validation helps to verify the correctness of entered data. To validate, click  at the top right of the editor.

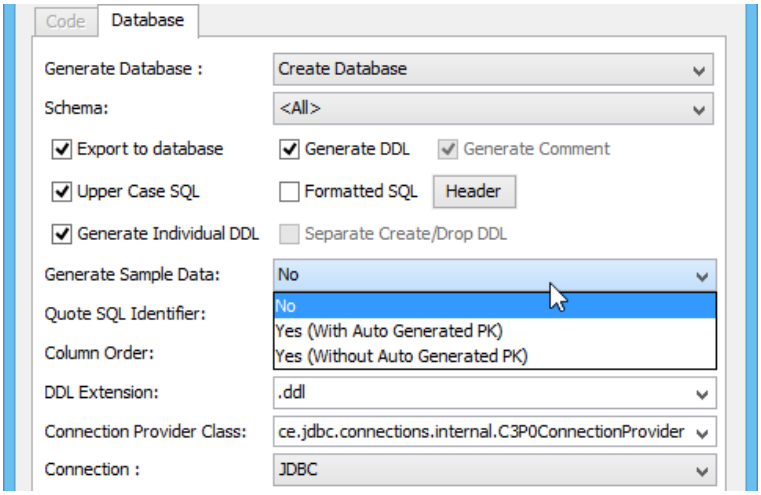
If anything wrong is detected, the **Message** pane will popup and a message will appear in it.



Message appear when something goes wrong

Generating default data

In order for the sample records to be exported to database in database generation, you have to set the **Generate Sample Data** option in the **Database Code Generation** window.



Setting the Generate Sample Data option

Here is a description of the available choices:

- No - Not to generate sample table records
- Yes (With Auto Generated PK) - Generate sample table records. For primary keys, the value you entered in sample table records editor will be used.
- Yes (Without Auto Generated PK) - Generate sample table records. For primary keys, the value you entered in sample table records editor will be ignored, leaving the DBMS handle the generation of primary key.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Modeling Database View

A database view is the result of a query on the data stored in a database. You can select columns, specify where and join statements to a view and present the data as if the data were coming from one single table. In Visual Paradigm, you can edit database view in a visual editor.

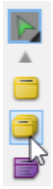
Creating database view

A database view is represented visually with a View shape. You can create a database view from the diagram toolbar or from the entities involved in the database view.

From diagram toolbar

You can create a database view and then edit it by specifying the entities involved. To create a database view:

1. Select **View** from the diagram toolbar.



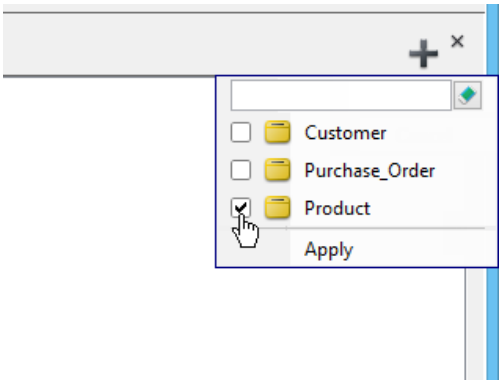
Select View from diagram toolbar

2. Click on the diagram to create a view.
3. Enter its name and confirm.

Database view created

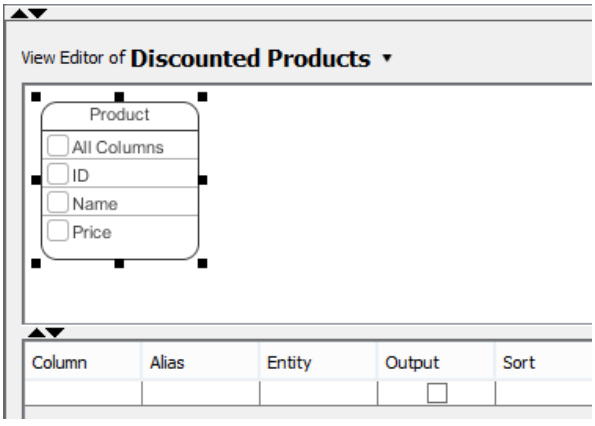
You can now specify the entities involved in the view via the **View Editor**. To open the **View Editor**, right click on the background of ERD and select **Show Table Record Editor or View Editor** from the popup menu.

4. On the right hand side of the **View Editor**. Click on the add button. Then, select the entities that are involved in the view and click **Apply**.



Add some entities to a view

Now, you can specify the view. For details, read the following sections.

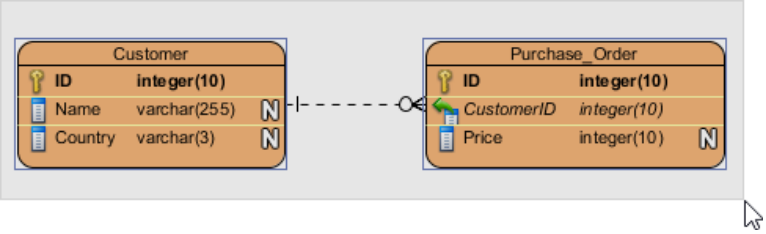


Entity added to view

From entities

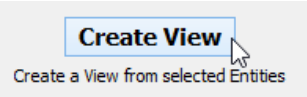
Instead of creating a blank view and adding entities into the view, you can create a view directly from entities that are involved in the view. To create a database view from entities involved:

1. Select the entities in diagram.



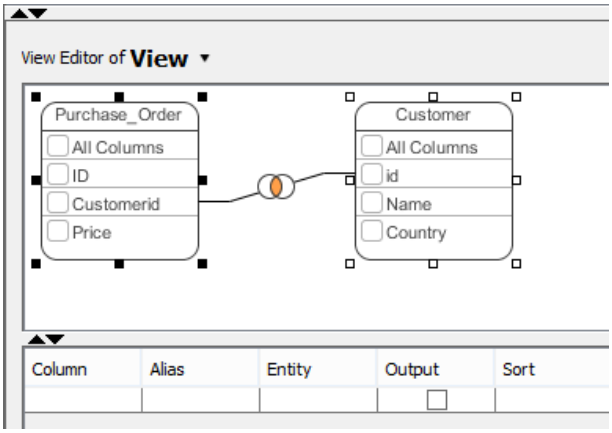
Select entities to create view

2. Click **Create View** in the **View Editor**. The **View Editor** is placed under the ERD. If you do not see it, right click on the background of ERD and select **Show Table Record Editor or View Editor** from the popup menu.



To create view

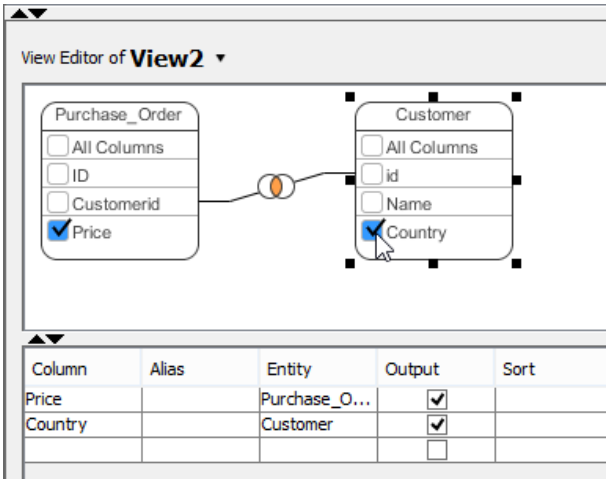
Now, you can specify the view. For details, read the following sections.



View formed from multiple entities

Column selection

A database view contains rows (i.e. the results) and columns, just like a database table. To select columns, simply click on the checkbox of the desired column. Alternatively, select the column in the **Column** column in the **View Editor**.

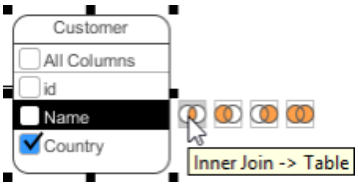


Selecting column

Joining columns

Joining of columns between entities can be done by the resource-centric interface. To do this:

1. Click on the source column in the entitiy in **View Editor**.
2. Press on the desired resource and drag it out.

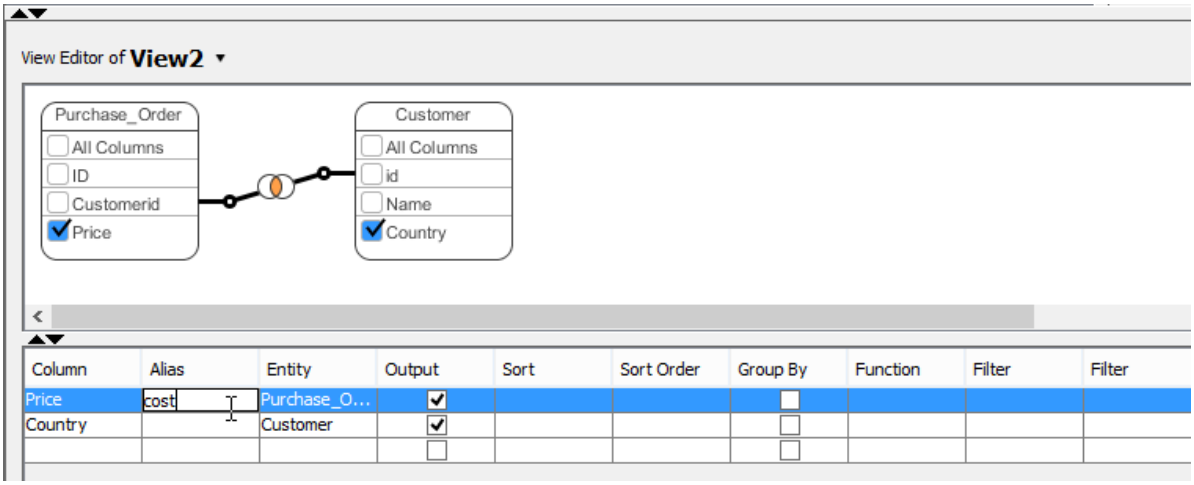


To join columns

3. Drag to the target column and release the mouse button.

Editing database view in View Editor table

There is a table under the visual view editor where you can configure a view's column alias, sort order, grouping, function, filter, etc.



Editing the alias of column

Property	Description
Column	Correspond to the SELECT clause of a view creation statement.
Alias	The displayed name of column.
Entity	The entity where the column come from.
Output	Check it to include the column into the creation statement of view.
Sort	Specify whether to sort the column ascendingly or descendingly
Sort Order	Specify the sort column. Records are sort following the order, with smaller number sort first.
Group By	Check it if you want the results be grouped by the column.
Function	Apply function to the column.
Filter	Add filter for the column.

Database view table editor

Supported fitlers

The following table lists out the filters you can enter in the Filter field.

Filter	Description	Sample
=	Equals	= 10
<>	Does not equal	<> 250
<	Less than	< 80
<=	Less than or equal to	<= 200
>	Greater than	> 0
>=	Greater than or equal to	>= 18
LIKE	Search for a pattern	LIKE '%PRIORITY%'
IN	To specify multiple possible values	IN ('Administrator', 'Manager')

BETWEEN	Between an inclusive range	BETWEEN 10 AND 20
IS NULL / IS NOT NULL	IS NULL: To select records with null IS NOT NULL: To select records with filled values	IS NULL / IS NOT NULL
NOT	Only when no row is returned from the sub query.	NOT EXISTS (SELECT NAME FROM USERS)
EXISTS	Only when at least one row is returned from the sub query.	EXISTS (SELECT NAME FROM USERS)
ALL	Compares a scalar value with a single-column set of values.	>= ALL (SELECT STOCK FROM PRODUCTS)
ANY	Compares a scalar value with a single-column set of values.	>= ANY (SELECT STOCK FROM PRODUCTS)

Supported filters

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Modeling Stored Procedures in ERD

A stored procedure is a pre-written procedure code that allows you to execute over and over again for validation or quick retrieval of data. The use of stored procedure helps maintain a consistent implementation of logic across program modules and applications. It also makes the design, coding and testing easier because the logic is put in a single place - the stored procedure.

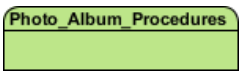
In Visual Paradigm, stored procedure is modeled in form of a procedure container. You can create stored procedures (mind the 's' here) shape, and add stored procedure to the procedures shape as rows in the procedures shape.

Creating a Stored Procedure

1. Select **Stored Procedures** from diagram toolbar.

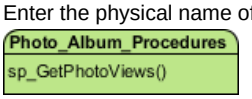
Select Stored Procedures

2. Click on the diagram to create a stored procedures shape. Note again that this is a container of stored procedures, not the procedure itself.
3. Enter its name.



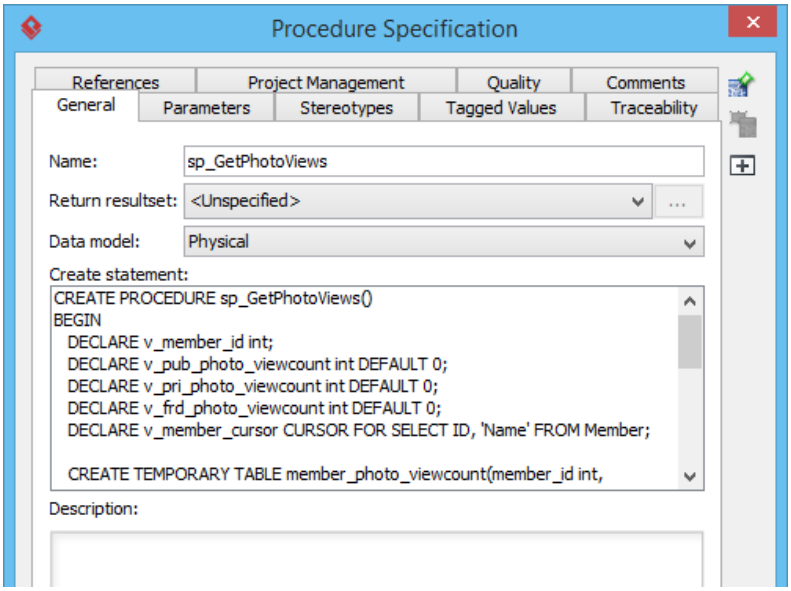
Stored procedures created

4. To create a stored procedure, right click on the stored procedures shape and select **New Procedure** from the popup menu.
5. Enter the physical name of the procedure .



Stored procedure added

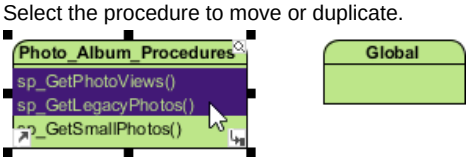
6. Right click on the procedure and select **Open Specification...** from the popup menu.
7. Enter the **Create statement** of procedure. The create statement entered here will be executed in database generation, so make sure it's in correct syntax.



Create statement of stored procedure

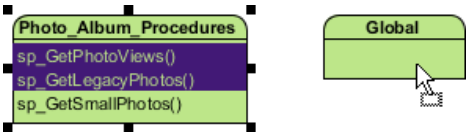
Moving or duplicating a procedure to another procedure container

1. Select the procedure to move or duplicate.



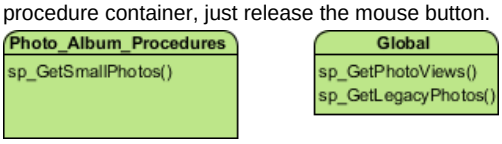
Select procedures to move or duplicate

2. Drag over the target procedure container.



Drag procedure towards the target procedures container

3. If you want to duplicate the procedures, press on the **Ctrl** key and release the mouse button. If you want to move them from source to target procedure container, just release the mouse button.



Procedures are moved

Creating stored procedure resultset

1. Select **Stored Procedure ResultSet** from diagram toolbar.

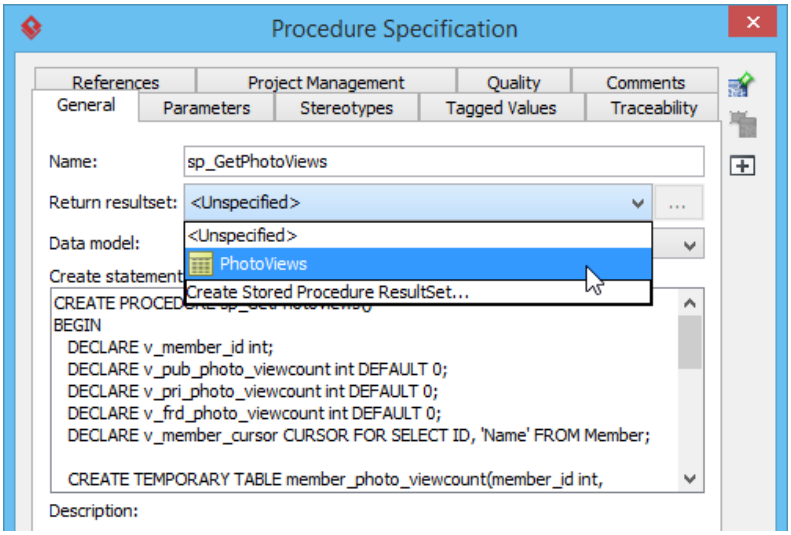
Select Stored Procedure ResultSet

2. Click on the diagram to create a stored procedure resultset shape.
3. Enter the name of the resultset.

Resultset created

Assigning stored procedure resultset to stored procedure

1. Right click on the stored procedure and select **Open Specification...** from the popup menu.
2. In the **Procedure Specification** window, specify the **Return resultset**.



Assigning stored procedure resultset to stored procedure

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

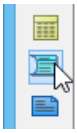
- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Modeling Triggers in ERD

A database trigger is a procedure that is automatically executed in response to certain events on a database table. A common use of database trigger is for auditing database. A trigger that records the insertion, modification and deletion of important data will let you know when and why certain change has been made to the database.

Creating a Trigger

1. Select **Triggers** from diagram toolbar.

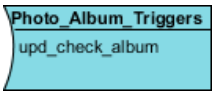


Select Triggers

2. Click on the diagram to create a triggers shape. Note again that this is a container of triggers, not the trigger itself.
3. Enter its name.

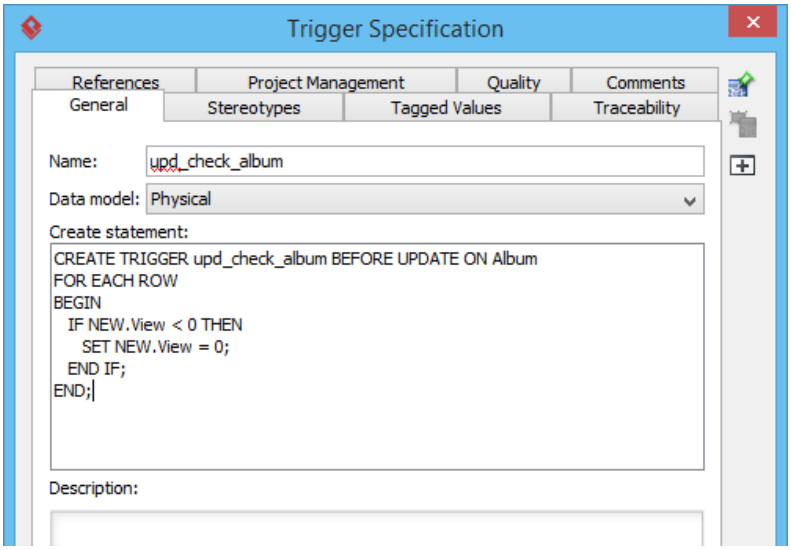
Triggers created

4. To create a trigger, right click on the triggers shape and select **New Trigger** from the popup menu.
5. Enter the physical name of the trigger.



Trigger added

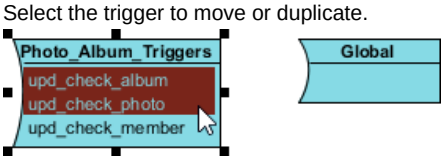
6. Right click on the trigger and select **Open Specification...** from the popup menu.
7. Enter the **Create statement** of trigger. The create statement entered here will be executed in database generation, so make sure it's in correct syntax.



Create statement of trigger

Moving or duplicating a trigger to another trigger container

1. Select the trigger to move or duplicate.

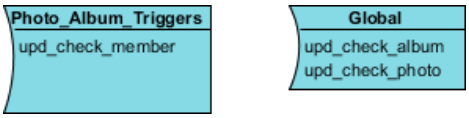


Select triggers to move or duplicate

2. Drag over the target trigger container.

Drag trigger towards the target triggers container

3. If you want to duplicate the triggers, press on the **Ctrl** key and release the mouse button. If you want to move them from source to target trigger container, just release the mouse button.



Triggers are moved

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

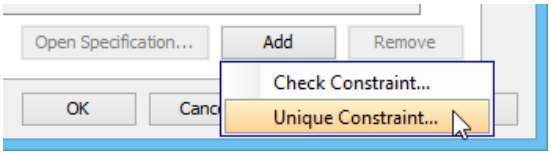
Working with Unique Constraint

Unique constraints help to enforce the uniqueness of specific columns. You can add unique constraints to an entity to make sure that no duplicate values are entered in specific columns. A unique constraint may consist of a single column, or combination of columns.

Creating a unique constraint

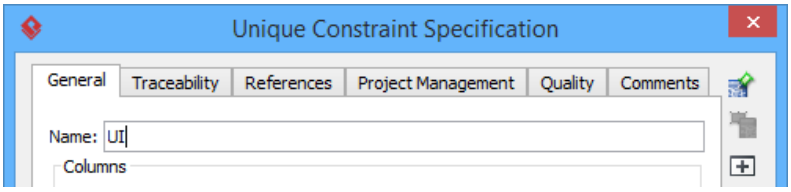
To create a unique constraint:

1. Right click on the entity and select **Open Specification...** from the popup menu.
2. Open the **Constraints** tab.
3. At the bottom right of the specification window, click **Add > Unique Constraint...**.



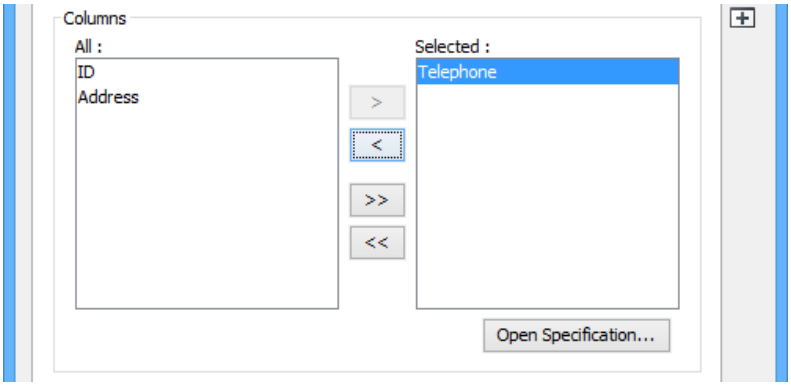
Add a unique constraint

4. In the **Unique Constraint Specification** window, enter the name of the constraint.



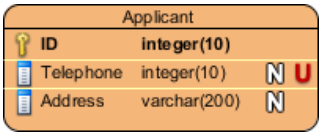
Name the unique constraint

5. Select the columns to be included and click **>**.



Add columns to the constraint

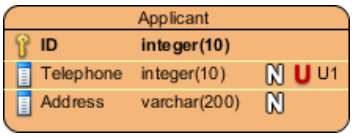
6. Click **OK** to return to the **Entity Specification**.
7. Click **OK** to return to the diagram. You can see the selected column(s) is marked unique.



Columns marked unique

Showing the name of unique constraints in entity

To show the name of unique constraint in entities in an ERD, right click on the ERD and select **Presentation Options > Entity Columns Display Options > Show Column Unique Constraint Name** from the popup menu. With enough shape width, you can see the constraint names of columns.



Constraint name shown

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

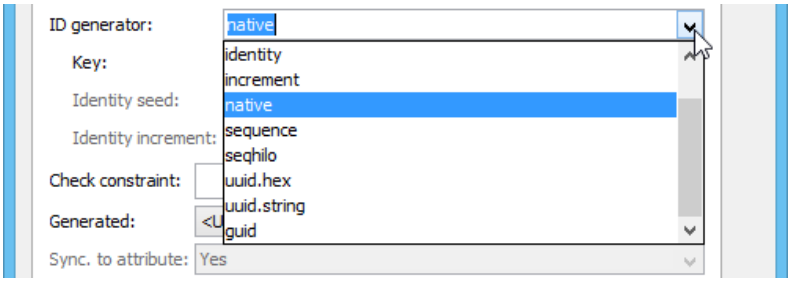
Using ID Generator

In relational database, records are identified by a unique value, called the primary key. Some databases support controlling how this unique value can be generated, through the use of an ID generator. In this article you will learn how to select an ID generator for a primary key. And since ID generator is designed to serve primary key columns, it is only available to columns that are included in primary keys.

Setting ID generator for a column

ID generator is a column-specific option that can be set in the **Column Specification** window. To set ID generator:

1. Right click on the primary key column of an entity and select **Open Specification...** from the popup menu.
2. Select the ID generator from the drop-down menu of **ID Generator**. If you select sequence, native, seqhilo or hilo as ID Generator, you have to enter the key for the sequence/table name.



Selecting ID generator

3. Click **OK** to confirm editing.

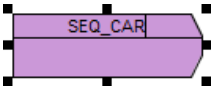
Using sequence as ID generator

You can model a sequence and use it as the ID Generator. To model a sequence:

1. Select **Sequence** from diagram toolbar.

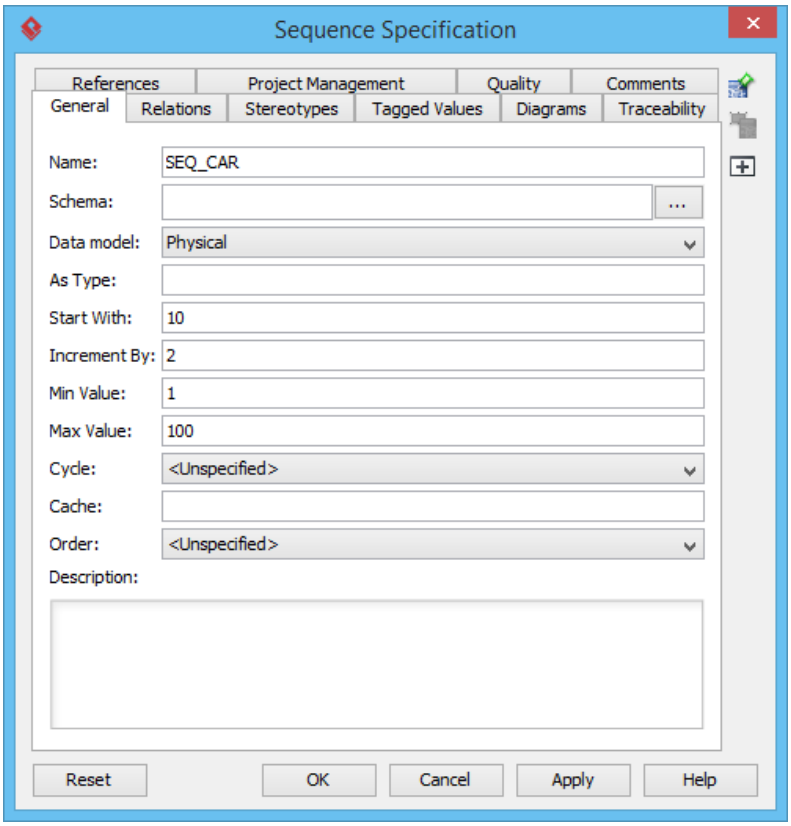
Select Sequence

2. Click on the diagram to create a sequence shape.
3. Enter its name.



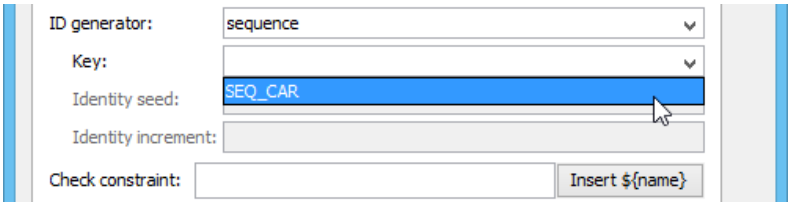
Sequence created

4. Specify the details of the sequence. Right click on the shape and select **Open Specification...** from the popup menu.
5. Fill in the details of your sequence, such as **Start With**, **Increment By**, **Min Value**, **Max Value**, etc.



Sequence Specification window

6. Click **OK** to confirm editing.
7. Now, you can select this sequence as an ID generator of primary key column.



Selecting sequence as ID generator

Description of common ID generators

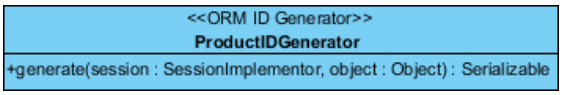
ID Generator	Description
assigned	allows the application to assign an identifier to the object before save() is called.
guid	uses a database-generated GUID string on MS SQL Server and MySQL.
hilo	uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a table and column as a source of hi values. The hi/lo algorithm generates identifiers that are unique only for a particular database.
identity	supports identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL. The returned identifier is of type long, short or int.
increment	generates identifiers of type long, short or int that are unique only when no other process is inserting data into the same table. Do not use in a cluster.
native	(default) picks identity, sequence or hilo depending upon the capabilities of the underlying database.
seqhilo	uses a hi/lo algorithm to efficiently generate identifiers of type long, short or int, given a named database sequence.
sequence	uses a sequence in DB2, PostgreSQL, Oracle. The returned identifier is of type long,short or int

Description of common ID generators

Customizing ID generator

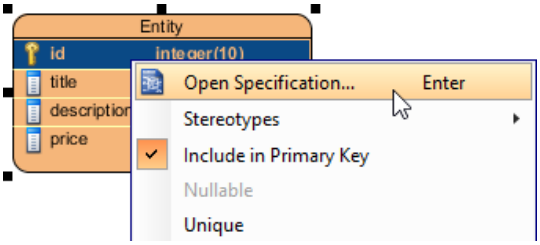
Besides the built-in strategies for generating ID, users can implement how ID will be generated by customizing an ID generator.

1. In Class Diagram, create the ID generator class and stereotype it as **ORM ID Generator**.



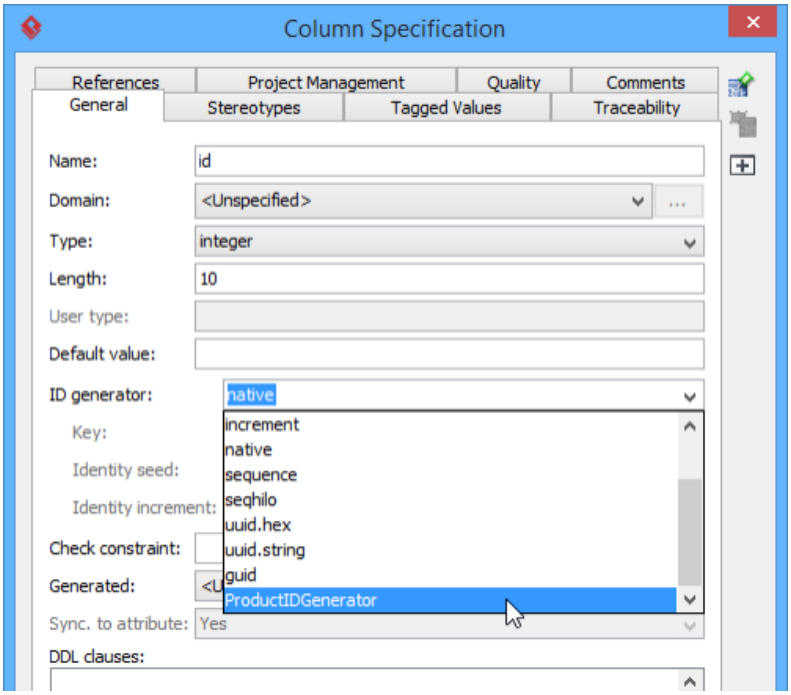
An ID generator class

2. Right click on the primary key column that you want to select an ID generator for and select **Open Specification...** from the pop-up menu.



Click Open Specification... from the pop-up menu

3. In the **Column Specification** window, select the class in the **ID Generator**.



Select an ID generator in Column Specification window

4. Click **OK** to confirm.
5. After generated the ORM code, look for the ID generator class and implement the **generate** method by returning an Integer or Long.

```
/**
 * Licensee: VP Development
 * License Type: Purchased
 */
import java.io.Serializable;
import org.hibernate.engine.SessionImplementor;
import org.hibernate.id.IdentifierGenerator;
public class ProductIDGenerator implements IdentifierGenerator {
    public Serializable generate(SessionImplementor session, Object object) {
        //TODO: Implement Method
        throw new UnsupportedOperationException();
    }

}
//ORM Hash:fae9faed19486e5f2b85c9d2d0d52cd9
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

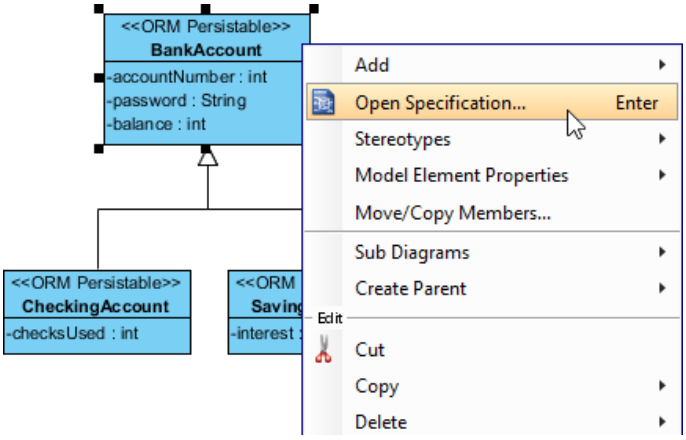
- Contact us if you need any help or have any suggestion

Different Inheritance Strategies

In UML world, you can model classes with similar characteristics with a generalization hierarchy, which groups the common attributes and behaviors into a class known as the superclass, leaving the distinctions in different subclasses that inhertis the features of the superclass. Unlike UML, ERD, as a language for designing relational mapping, has no direct way of representing a generalization hierarchy. In order for an object model to map with and conform to a data model upon synchronization, inheritance strategy has to be chosen to define the way how entities should be created and structured to represent the generalization hierarchy modeled in object model. In Visual Paradigm, there are three strategies you can choose. Your selection will affect the way how entities are created and structured, ultimately affecting the way how data will be stored in database.

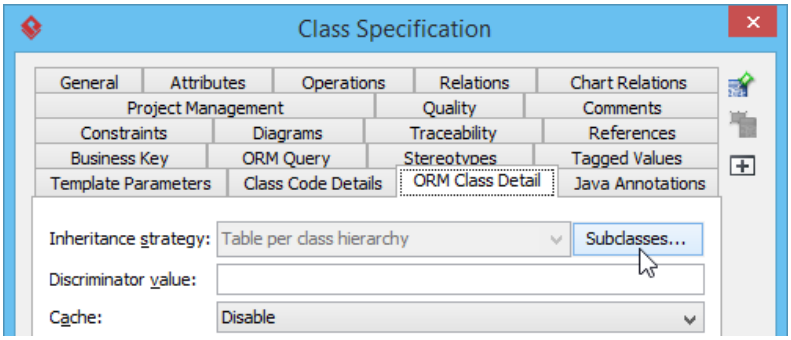
Choosing an inheritance strategy

1. Right click on the superclass within a generalization hierarchy and select **Open Specification...** from the popup menu.



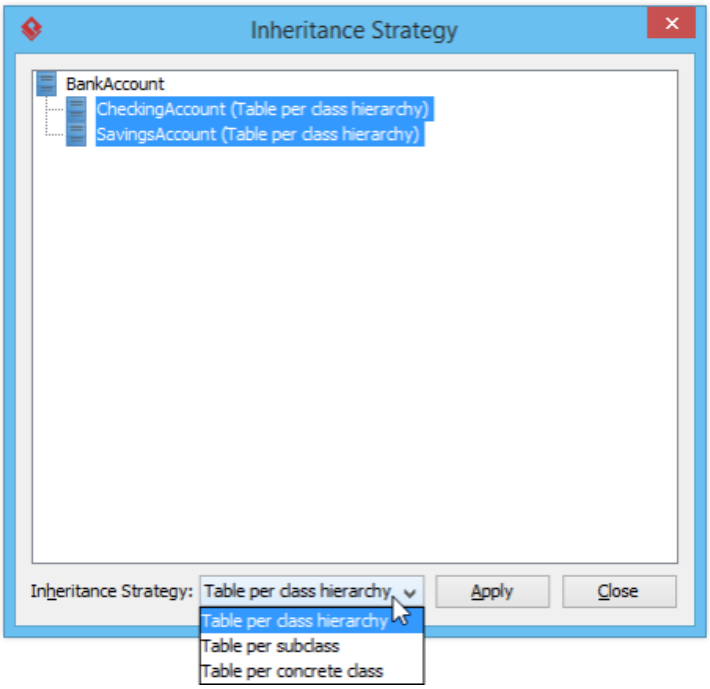
Opening class specification

2. Open the **ORM Class Detail** tab.
3. Click **Subclasses**....



Clicking on Subclasses

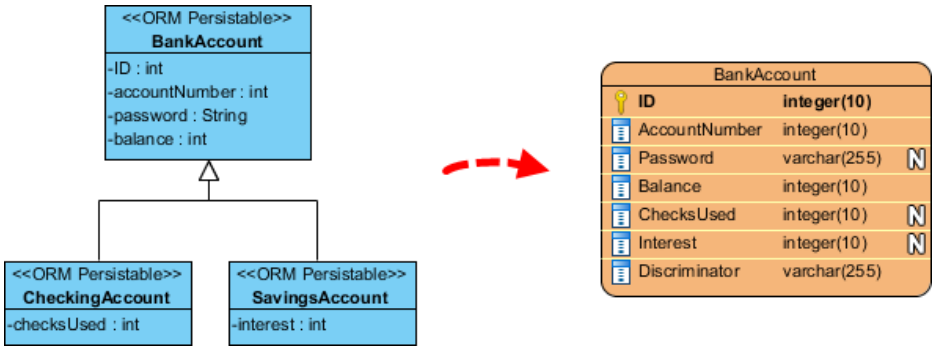
4. Select the sub-classes. Press **Shift/Ctrl** to perform a multi selection.
5. Select the **Inheritance Strategy**.



6. Click **Apply**.
7. Click **Close**.
8. Click **OK**.

NOTE: Different inheritance strategies can be applied to different subclasses within a generalization hierarchy in Java project. Applying multiple strategies to different subclasses within a generalization in .NET project will result in error when the generation of code and database.

When you synchronize class diagram to ERD, you will see the entity(ies) created or re-created to respect the inheritance strategy you chose.



Synchronizing class diagram to ERD

Strategy 1 - Table per class hierarchy

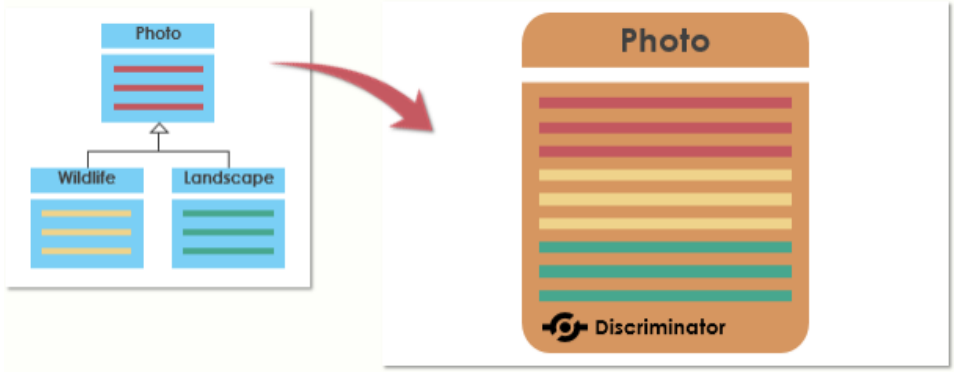


Table per class hierarchy

With this strategy applied, a single entity will be created for all classes within a hierarchy. Attributes from superclass and all subclasses will become columns of that entity. In order to differenetiate the type of database records in runtime, an extra discriminator column is used to store a value for identification.

Strategy 2 - Table per subclass

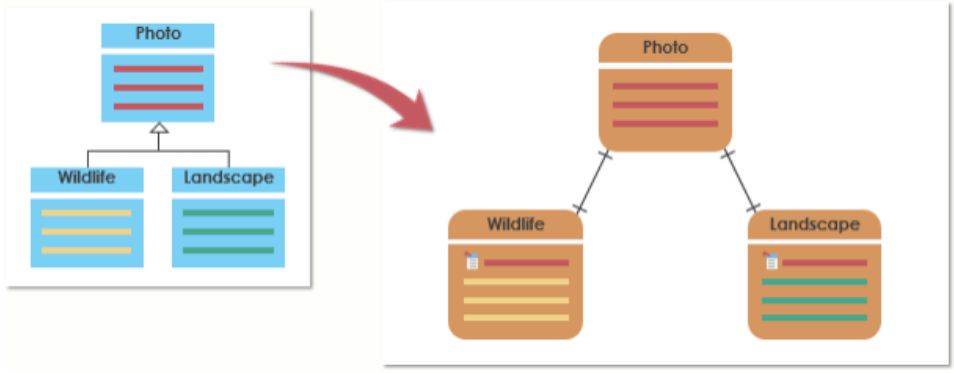


Table per subclass

With this strategy applied, separate entities will be created for each subclass. A foreign key is included in each "sub-entity" for referencing the "super-entity".

Strategy 3 - Table per concrete class

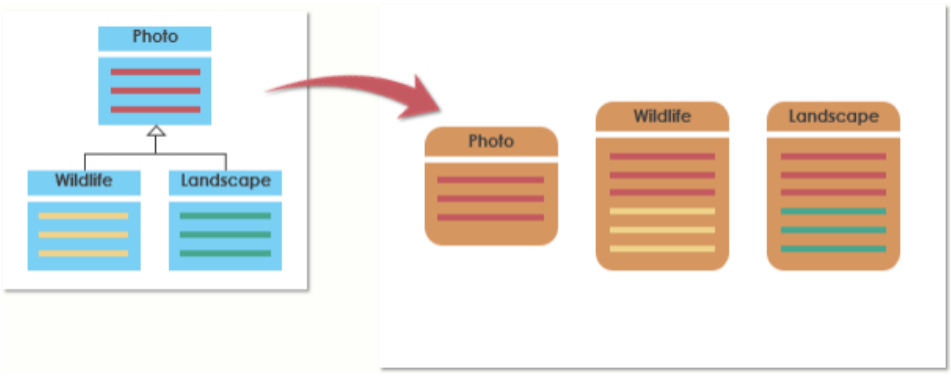


Table per concrete class

With this strategy applied, separate entities will be created for each class within the generalization hierarchy. Each entity contains columns that are produced by the corresponding class plus its superclass(es) along the hierarchy.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

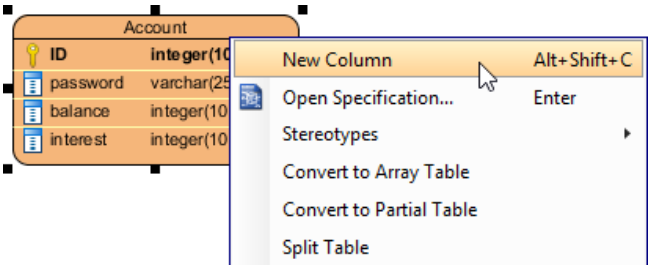
Using Discriminator Column

In a generalization hierarchy, the superclass distributes its commonalities to its subclasses. The subclasses inherit all superclass' attributes and may contain their own attributes. Visual Paradigm combines the entities within the hierarchy into one single entity that contains all the attributes, plus an additional discriminator column, which contains a unique value that can be used for identifying the entity in which a record belongs to. Discriminator column is used when "table per class" is chosen as inheritance strategy. In order to make discriminator works, you need to define the discriminator in the entity and the discriminator values in the object model.

Adding the discriminator column for entity

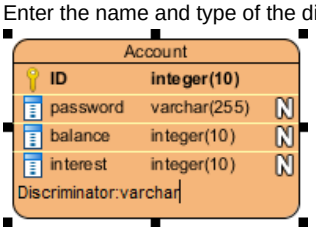
You can add a column to an entity as the discriminator column. To add a discriminator column:

1. Right click on an entity and select **New Column** from the popup menu.



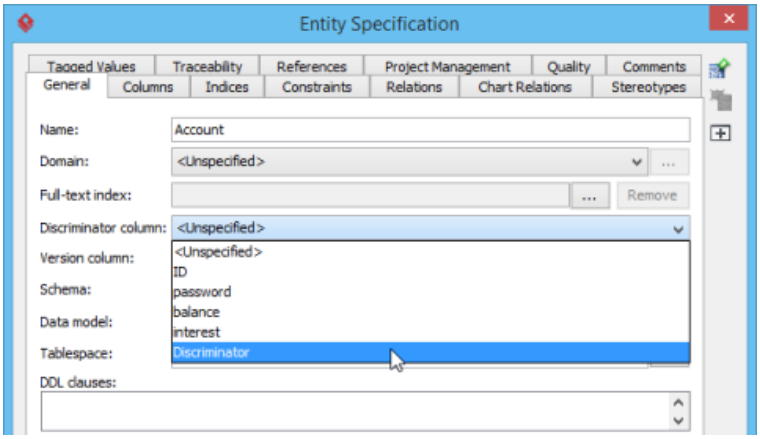
Create a column in entity

2. Enter the name and type of the discriminator.



Enter the name of discriminator column

3. Right click on the entity and select **Open Specification...** from the popup menu.
4. Select the column just created from the **Discriminator Column** drop-down menu.



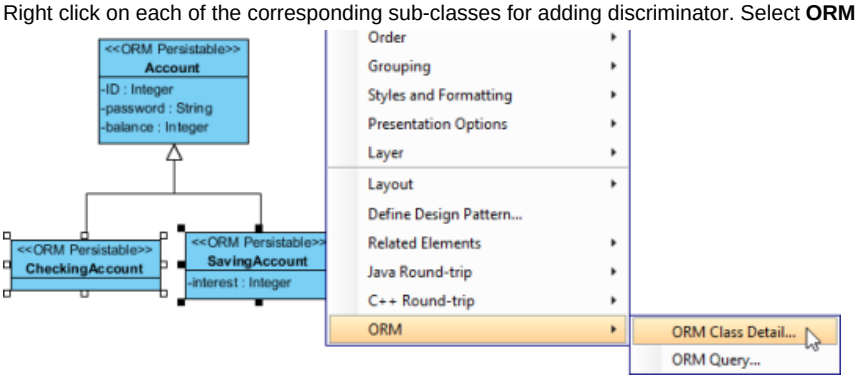
Select discriminator column

5. Click **OK**.

Defining Discriminator Value for Class

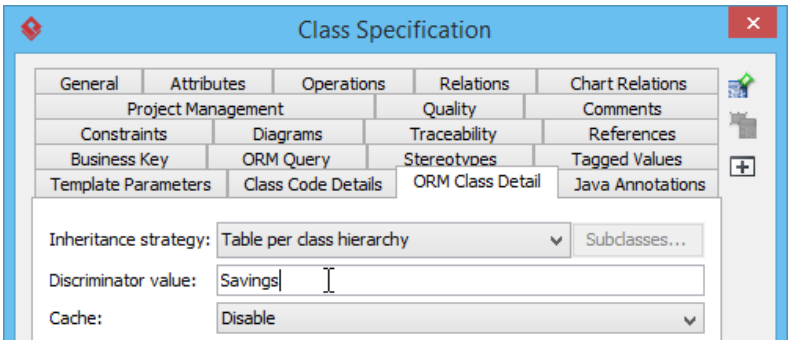
You can specify the discriminator value for each sub-class.

1. Right click on each of the corresponding sub-classes for adding discriminator. Select **ORM > ORM Class Detail...** from the popup menu.



To edit ORM class detail

2. In the **Class Specification** window, enter the discriminator value that represent the sub-class. This value will be stored in the corresponding database record under the discriminator column.



Enter discriminator value

3. Click **OK** to confirm the change.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

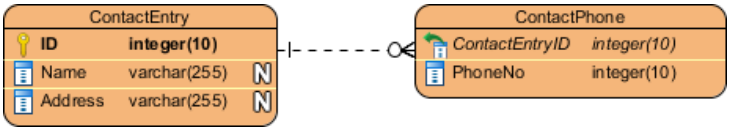
Using Array Table

In a one-to-many relationship, a collection is used for handling the multiple objects such that it is simpler to retrieve each object from the collection one by one. Visual Paradigm promotes the idea of Array Table which allows users to retrieve objects in the form of primitive array, instead of a collection when handling a data column with cardinality of many. Visual Paradigm allows you to create an array table in the entity and define an array type in the classes.

Converting an Entity to an Array Table

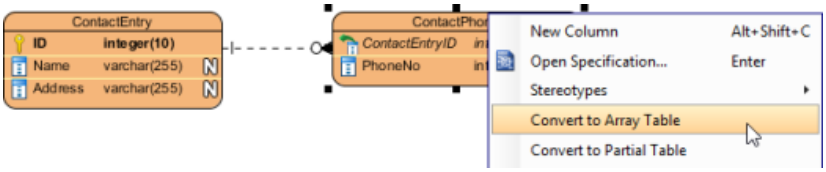
You can create an Array Table for the Entity with a column containing more than one instance of data.

1. Locate the "many" side of an one-to-many relationship. In the following example, the phone book has a contact entry for each contact person. Each contact person may have more than one phone numbers. A one-to-many relationship between contact entry and contact phone is established.



Entities with One-to-many relationship

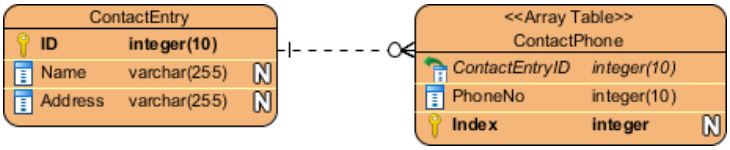
2. Right-click on the entity for the data column with cardinality of many, select **Convert to Array Table** from the popup menu.



Convert to Array Table

3. A warning message will be displayed, showing that the listed constraints are not satisfied for converting to array table. Click **Yes** to let Visual Paradigm to resolve the constraints automatically. Click **No** to cancel the conversion to array table.

4. The conversion to Array Table is completed and the entity for the data column is stereotyped as <<Array Table>>.



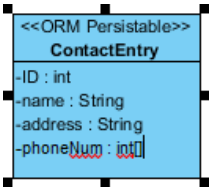
Array Table formed

Defining an Array Type for Attribute in Class

A class with an attribute of array type modifier means that the attribute may contain more than one data; thus it implies the idea of Array Table. You can define the array type for the attribute in one of the two ways:

Inline Editing

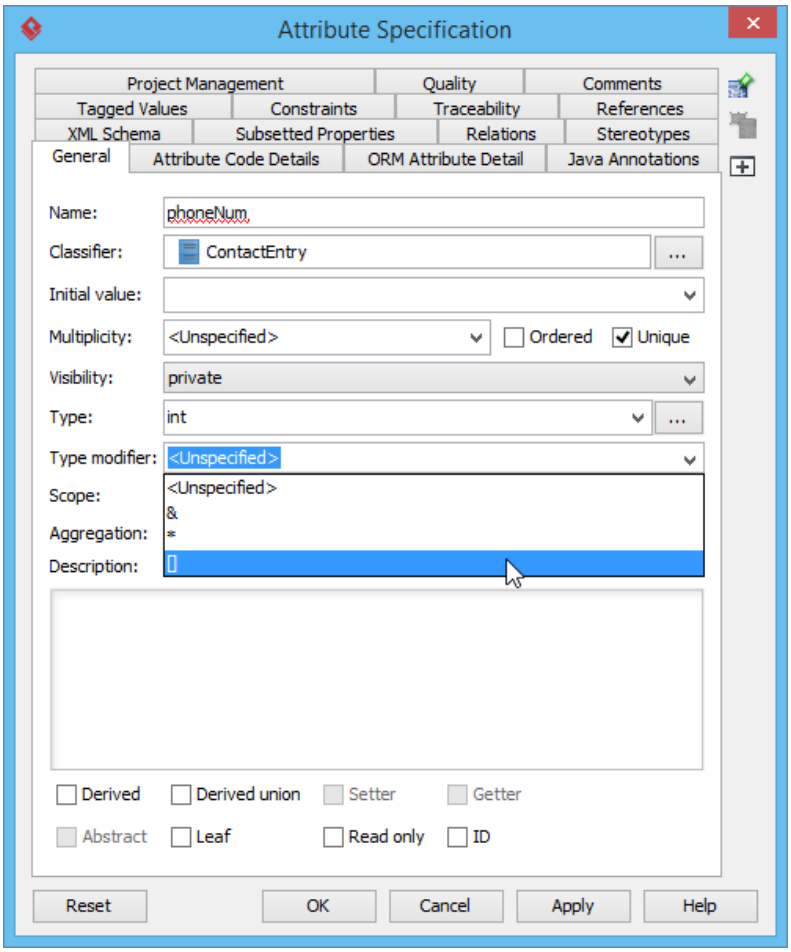
Double click on an attribute to edit it inline. Then, edit its type by appending the **[]** sign, which indicates that the attribute is an array.



Editing attribute inline

Editing Type Modifier in Attribute Specification Window

Right click on an attribute and select **Open Specification...** from the popup menu. In the **Attribute Specification** window, choose **[]** as **Type modifier**.



Selecting [] as Type Modifier

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using Partial Table

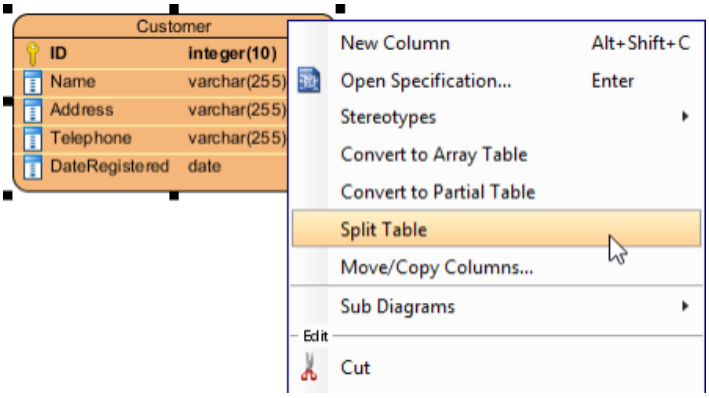
In a one-to-one identifying relationship, an entity may be a subordinate of the related entity; that is, the subordinate entity has columns which also belong to its superior entity in the real world situation. Visual Paradigm promotes the idea of Split Table with stereotype of Partial which allows developers to optimize the size of database, and minimizes the redundant persistent classes for handling one-to-one identifying relationship. In order to reduce the risk of appending a new column to an existing database table, Split table supports developers to add new columns to the partial table with a one-to-one identifying relationship linked to the existing table. Visual Paradigm allows you to split the entity into two and convert the subordinate entity to be a Partial Table in a one-to-one identifying relationship.

Splitting Table

You can split an entity into two associated entities with a one-to-one identifying relationship. There are two ways you can take to split a table.

Using popup menu

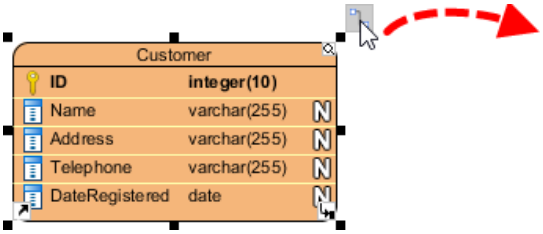
1. Right click on an entity.
2. Select **Split Table** from the popup menu.



Split table

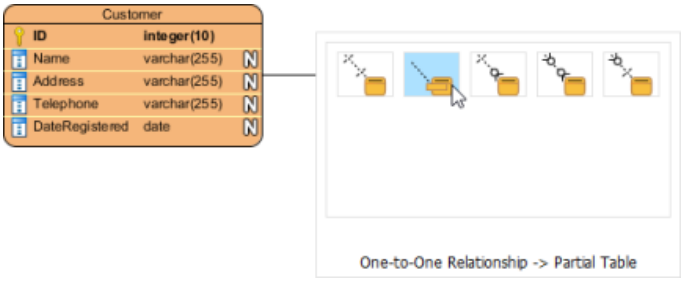
Using Resource Catalog

1. Move the mosue pointer over an entity.
2. Press on the **Resource Catalog** button and drag it out.



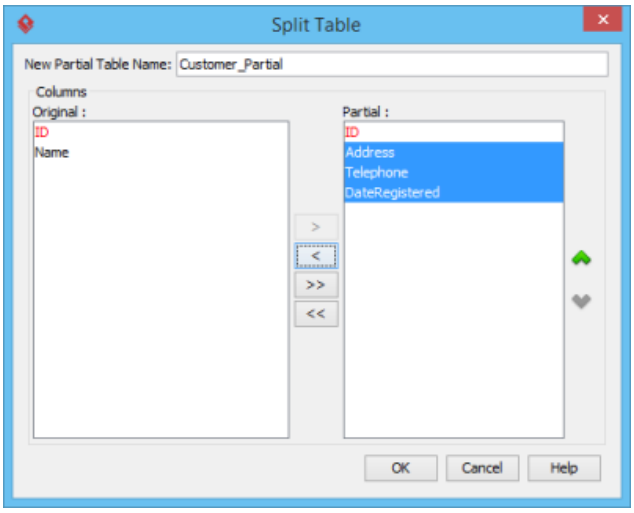
Split table using resource-centric interface

3. Release the mouse button and select **One-to-One Relationship -> Partial Table** from Resource Catalog.



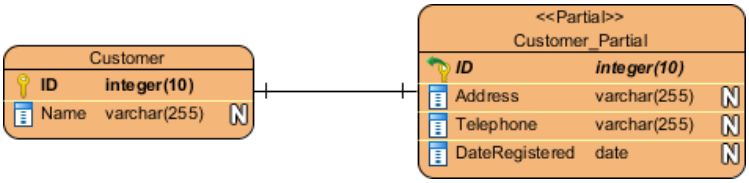
Select One-to-One Relationship -> Partial Table

Both ways will result in popping up the **Split Table** window. In the window, enter the name of the new entity and select the columns from the list of **Original** to **Partial**, and click **OK**.



Split Table window

An entity stereotyped <<Partial>> is created.

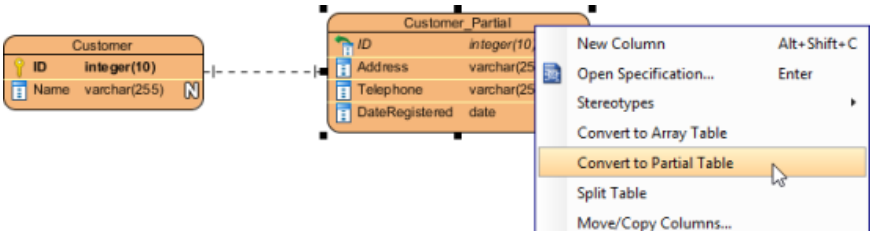


Partial table created

Converting to a Partial table

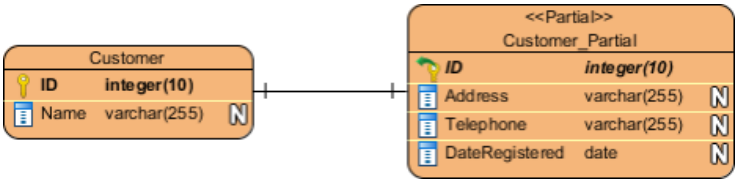
You can convert an entity to a Partial Table in a one-to-one identifying relationship.

1. Right click on the entity.
2. Select **Convert to Partial Table** from the popup menu.



Convert to Partial Table

The entity is stereotyped <<Partial>>.



Partial table converted

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

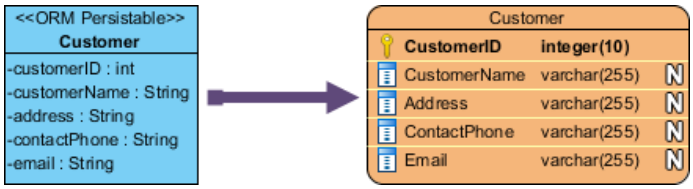
- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Mapping Object Model to Data Model

Visual Paradigm supports Object Relational Mapping (ORM) which maps object models to entity relational models and vice versa. Visual Paradigm helps mapping between Java objects to relational database. It not only preserves the data, but also the state, foreign/primary key mapping, difference in data type and business logic. Thus, you are not required to handle those tedious tasks during software development.

Mapping classes to entities

Generally speaking, the mapping between class and entity is a one-to-one mapping, meaning that one class in object model maps with one entity in data model. Classes that map with entities are represented by the stereotype <<ORM Persistable>>.

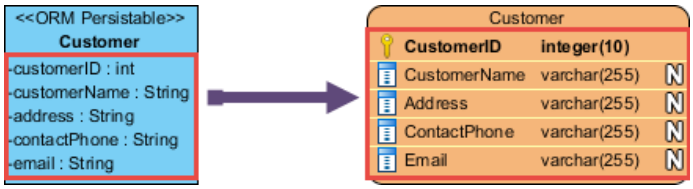


Mapping class

In the above example, the *Customer* class is the object equivalent of the *Customer* entity. This means that in application development or in runtime, an instance of *Customer* (class) stores the information of a customer retrieved from the *Customer* table of database.

Mapping attributes to columns

Since the persistent classes map to the entities, persistent attributes map to columns accordingly. Visual Paradigm ignores all non persistent attributes such as derived values.



Mapping attributes

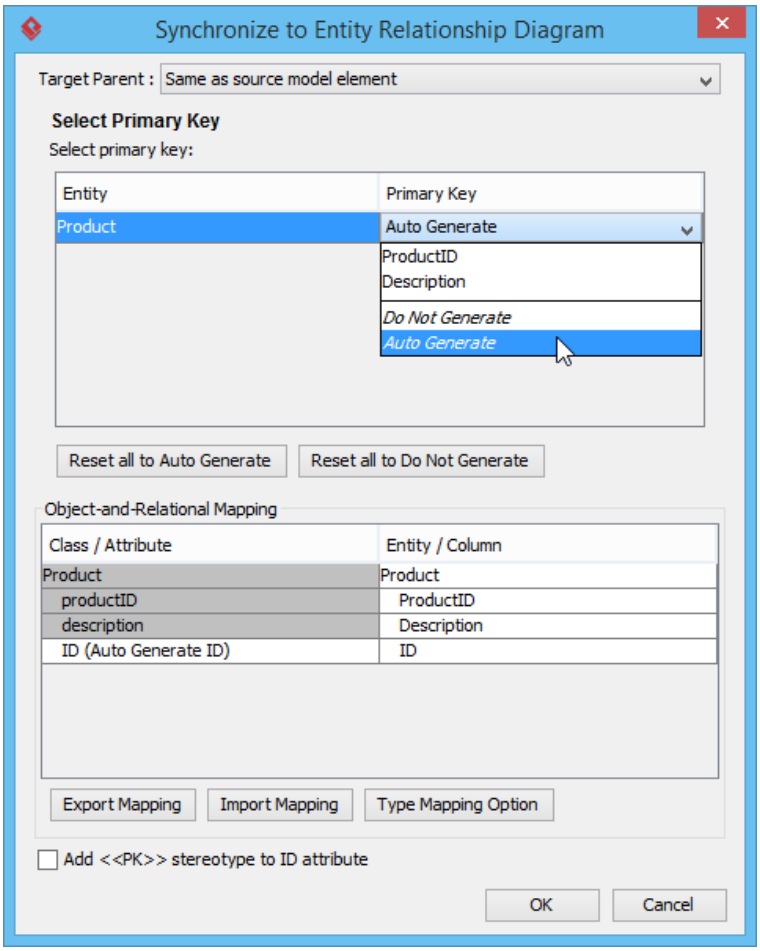
Mapping data type

Persistent attribute types are automatically mapped to appropriate column data types of the database you desired. The following table lists out the typical mapping between object model and data model. Note that the actual data type to map to depends on the default database you selected in database configuration.

Object Model	Data Model
Boolean	Bit (1)
Byte	Tinyint (3)
Byte[]	Binary (1000)
Blob	Blob
Char	Char (1)
Character	Char (1)
String	varchar (255)
Int	Integer (10)
Integer	Integer (10)
Double	Double (10)
Decimal	Integer
Bigdecimal	Decimal (19)
Float	Float (10)
Long	Bigint (19)
Short	Smallint (5)
Date	Date
Time	Time (7)
Timestamp	Timestamp (7)

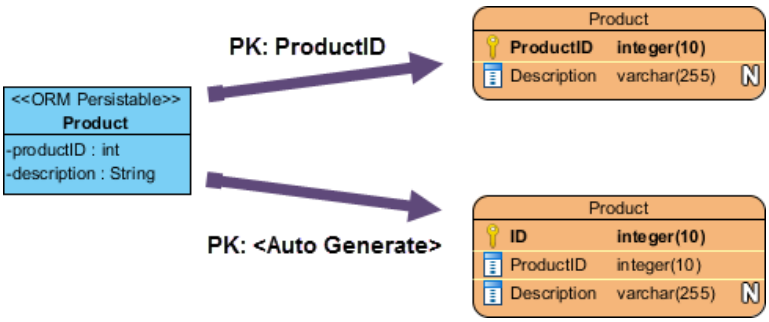
Mapping primary key

You can map an attribute to a primary key column. When you synchronize the ORM-Persistable Class to the ERD, you will be prompted by a window to select primary key.



Selecting the way to map primary key

You can select an existing attribute as primary key, let us generate one for you. or select **Do Not Generate** to leave the generated entity without primary key.



Mapping primary key

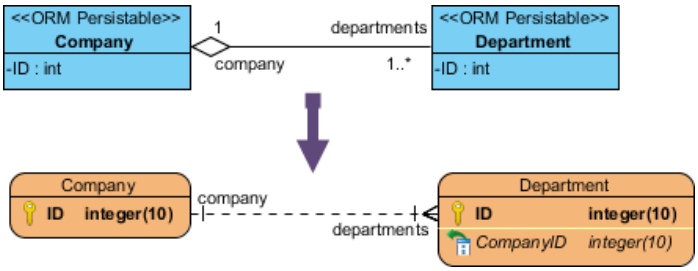
The above diagram shows if you assign *ProductID* as primary key, the *ProductID* of the generated entity, *Product* will become bold; whereas if you select **Auto Generate** for the primary key, Visual Paradigm generates an additional attribute *ID* as the primary key of the *Product* entity.

Mapping association

Association represents a binary relationship among classes. Each class of an association has a role. A role name is attached at the end of an association line. Visual Paradigm maps the role name to a phrase of relationship in the data model.

Mapping aggregation

Aggregation is a stronger form of association. It represents the "has-a" or "part-of" relationship.



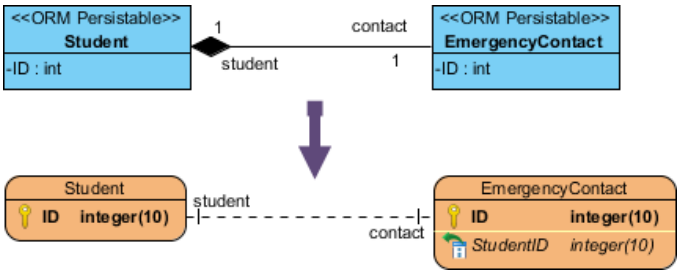
Mapping aggregation

In the above example, it shows that a company consists of one or more department while a department is a part of the company.

> **NOTE:** You have to give the role names, "ConsistsOf" and "is Part Of" to the classes, *Company* and *Department* in the association respectively in order to proceed to the generation of code.

Mapping composition

Composition implies exclusive ownership of the "part-of" classes by the "whole" class. It means that parts may be created after a composite is created, meanwhile such parts will be explicitly removed before the destruction of the composite.



Mapping composition

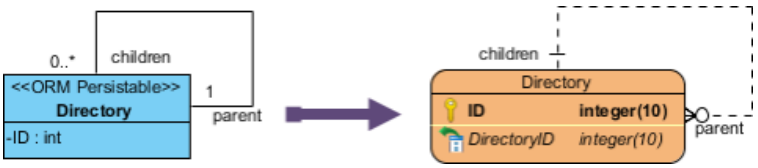
In the above example, Visual Paradigm performs the Primary/Foreign Key Column Mapping automatically. *ID* of the *Student* entity is added to the entity, *EmergencyContact* as primary and foreign key column.

Mapping multiplicity

Multiplicity refers to the number of objects associated with a given object. There are six types of multiplicity commonly found in the association. The following table shows the syntax to express Multiplicity.

Type of Multiplicity	Description
0	Zero instance
0..1	Zero or one instance
0..*	Zero or more instances
1	Exactly one instance
1..*	One of more instances
*	Unlimited number of instances

Description of multiplicities

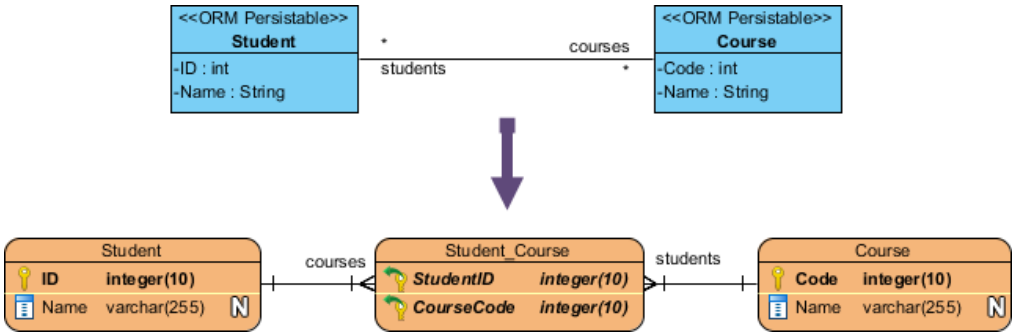


Mapping multiplicity

In the above example, it shows that a parent directory (role: parent) contains zero or more subdirectories (role: children).

Mapping many-to-many association

For a many-to-many association between two classes, Visual Paradigm will generate a Link Entity to form two one-to-many relationships in-between two generated entities. The primary keys of the two entities will migrate to the link entity as the primary/foreign keys.



Mapping many-to-many association

In the above example, Visual Paradigm generates the link entity, *Student_Course* between entities of *Student* and *Course* when transforming the many-to-many association.

Mapping inheritance/generalization

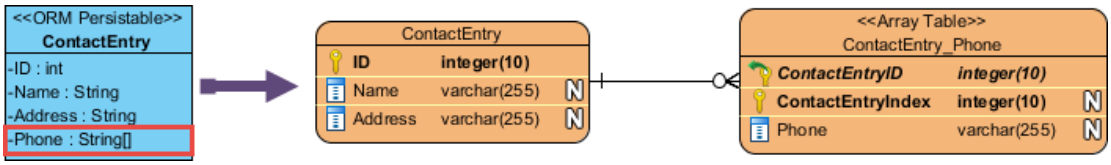
Generalization distributes the commonalities from the superclass among a group of similar subclasses. The subclass inherits all the superclass's attributes and it may contain specific attributes.

We provide multiple strategies for transforming the generalization hierarchy to relational model. Click here to learn more about the various inhertance strategies.

Mapping collection of objects to Array Table

For a persistent class acting as persistent data storage, it may consist of a persistent data containing a collection of objects. Visual Paradigm promotes the use of Array Table to allow users retrieve objects in the form of primitive array.

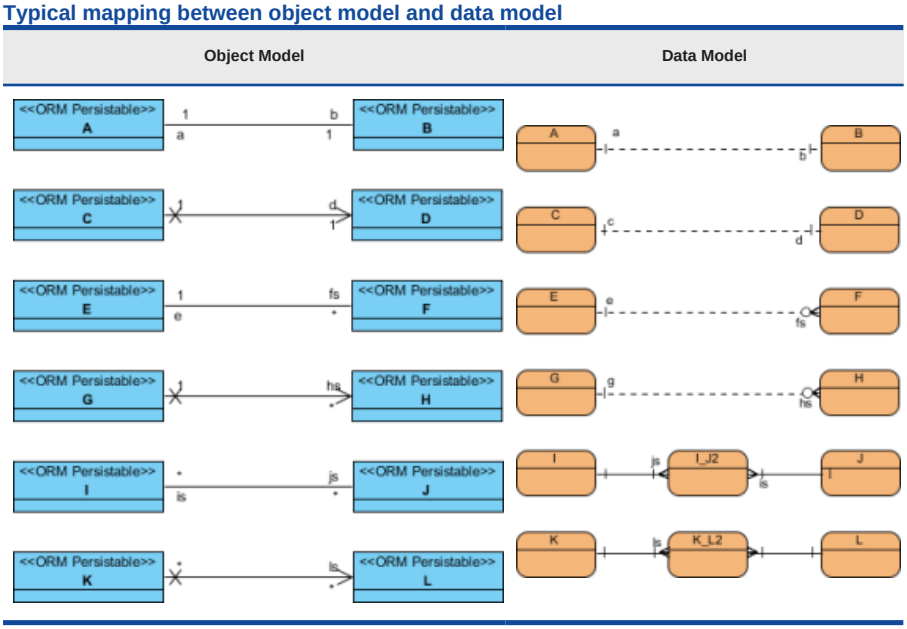
When Visual Paradigm transforms a class with an attribute of array type modifier, this attribute will be converted to an Array Table automatically. The generated entity and the array table form a one-to-many relationship.



Mapping collection of objects to Array Table

In the above example, each contact person may have more than one phone numbers. In order to ease the retrieval of a collection of phone objects, Visual Paradigm converts the phone attribute into a *ContactEntry_Phone* array table.

Typical mapping between object model and data model



Typical mapping between object model and data model

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

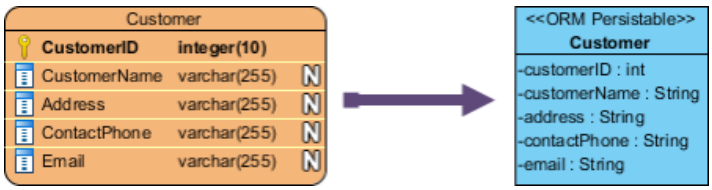
- Contact us if you need any help or have any suggestion

Mapping Data Model to Object Model

Visual Paradigm supports Object Relational Mapping (ORM) which maps data model to object model and vice versa. Visual Paradigm helps mapping entities to classes. Source files can then be generated from classes for use in software development. The mapping to object model preserves the data, but also the state, foreign/primary key mapping, difference in data type and business logic. Thus, you are not required to handle those tedious tasks during software development.

Mapping entities to classes

All entities map one-to-one to persistent classes in an object model. Classes that map with entities are represented by the stereotype <<ORM Persistable>>.

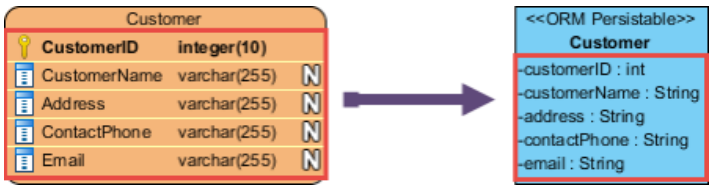


Mapping class

In the above example, the *Customer* entity map one-to-one the *Customer* class as the *Customer* instance can store the customer information from the *Customer* Entity.

Mapping columns to attributes

Since all entities map one-to-one to persistent classes in an object model, columns in turn map to attributes in a one-to-one mapping. Visual Paradigm ignores all specialty columns such as computed columns and foreign key columns.



Mapping columns

Mapping data type

Column types are automatically mapped to appropriate attribute types. The following table lists out the typical mapping between data model and object model.

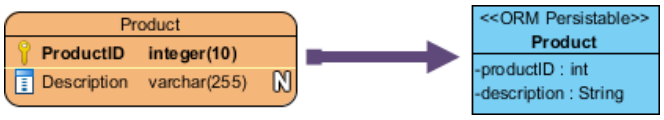
Data Model	Object Model
bigint	Long
binary	Byte []
bit	Boolean
blob	Blob
varchar	String
char	Character
char (1)	Character
clob	String
date	Date
decimal	BigDecimal
double	Double
float	Float
integer	Integer
numeric	BigDecimal
real	Float
time	Time
timestamp	Timestamp
tinyint	Byte

smallint	Short
varbinary	Byte []

Mapping of data types between data and object model

Mapping primary key

As the columns map to attributes in a one-to-one mapping, primary key columns in the entity map to attributes as a part of a class.



Mapping primary key

In the example, the primary key of entity *Product*, *ProductID* maps to an attribute *ProductID* of the class *Product*.

Mapping relationship

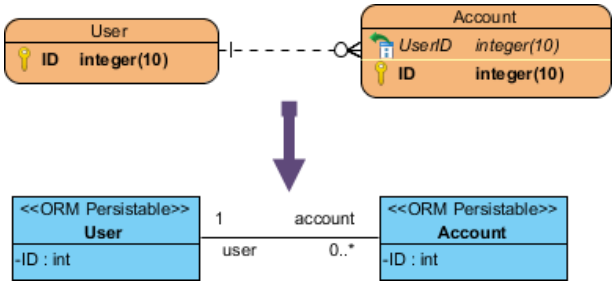
Relationship represents the correlation among entities. Each entity of a relationship has a role, called Phrase describing how the entity acts in it. The phrase is attached at the end of a relationship connection line. Visual Paradigm maps the phrase to role name of association in the object model.

Mapping cardinality

Cardinality refers to the number of possible instances of an entity relate to one instance of another entity. The following table shows the syntax to express the Cardinality.

Type of cardinality	Description
	Zero or one intance
	Zero or more instances
	Exactly one instance
	One or more instances

Description of cardinalities

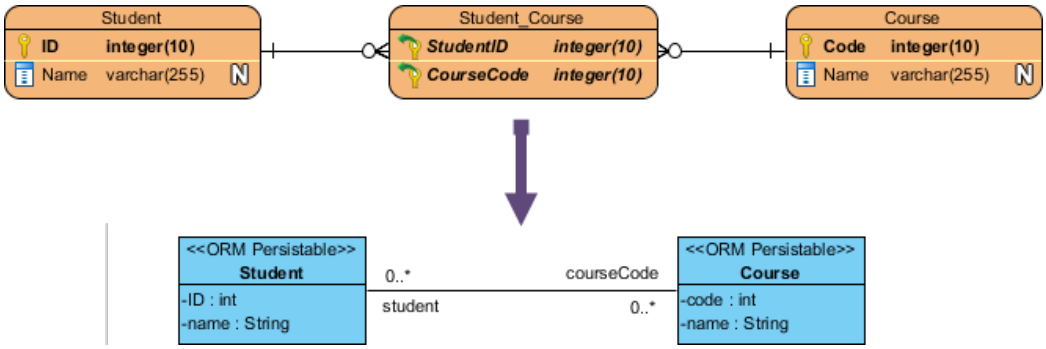


Mapping cardinality

In the above example, it shows that a user contains multiple accounts.

Mapping many-to-many relationship

For each many-to-many relationship between entities, Visual Paradigm generates a link entity to form two one-to-many relationships in between. The primary keys of the two entities will automatically migrate to the link entity to form the primary/foreign keys.



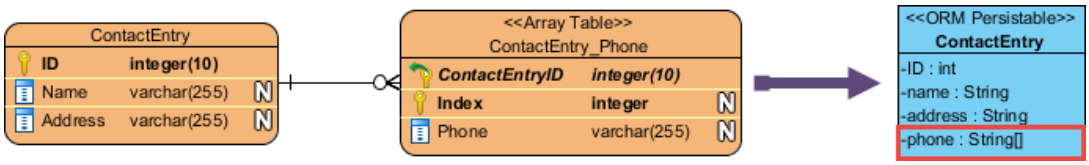
Mapping many-to-many relationship

In the above example, Visual Paradigm generates the link entity once a many-to-many relationship is setup between two entities. To transform the many-to-many relationship, Visual Paradigm maps the many-to-many relationship to many-to-many association.

Mapping Array Table to collection of objects

Visual Paradigm promotes the use of Array Table to allow users retrieve objects in the form of primitive array.

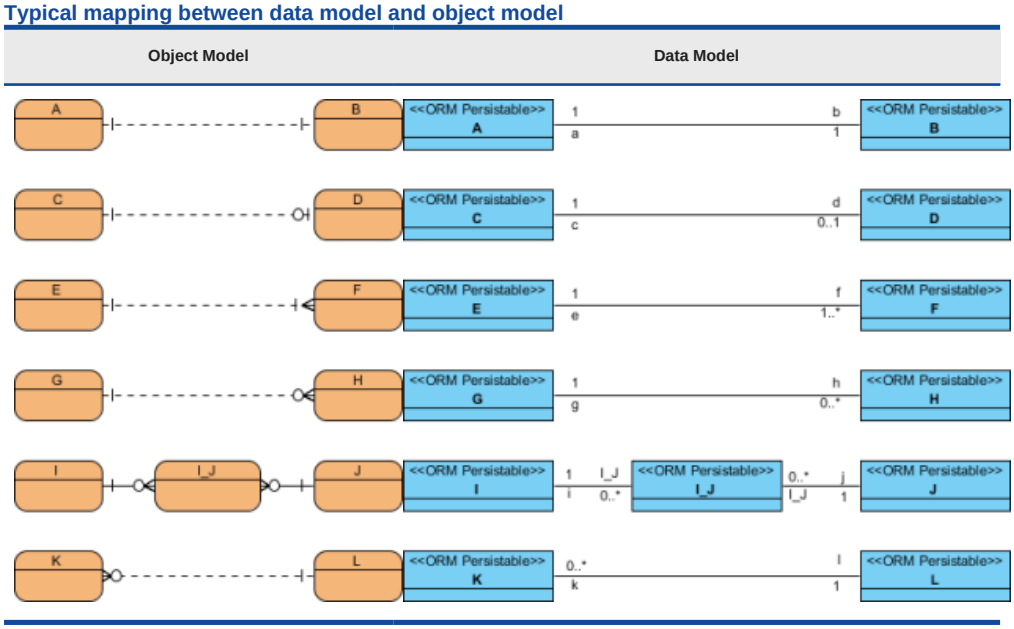
When Visual Paradigm transforms an array table, the array table will map to an attribute with array type modifier.



Mapping Array Table to collection of objects

In the above example, each *Contact* may have more than one phone numbers, stored in *ContactEntry_Phone*. The array table of *ContactEntry_Phone* maps into the phone attribute with array type modifier in the *ContactEntry* class.

Typical mapping between data model and object model



Typical mapping between data model and object model

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Synchronizing object model and data model

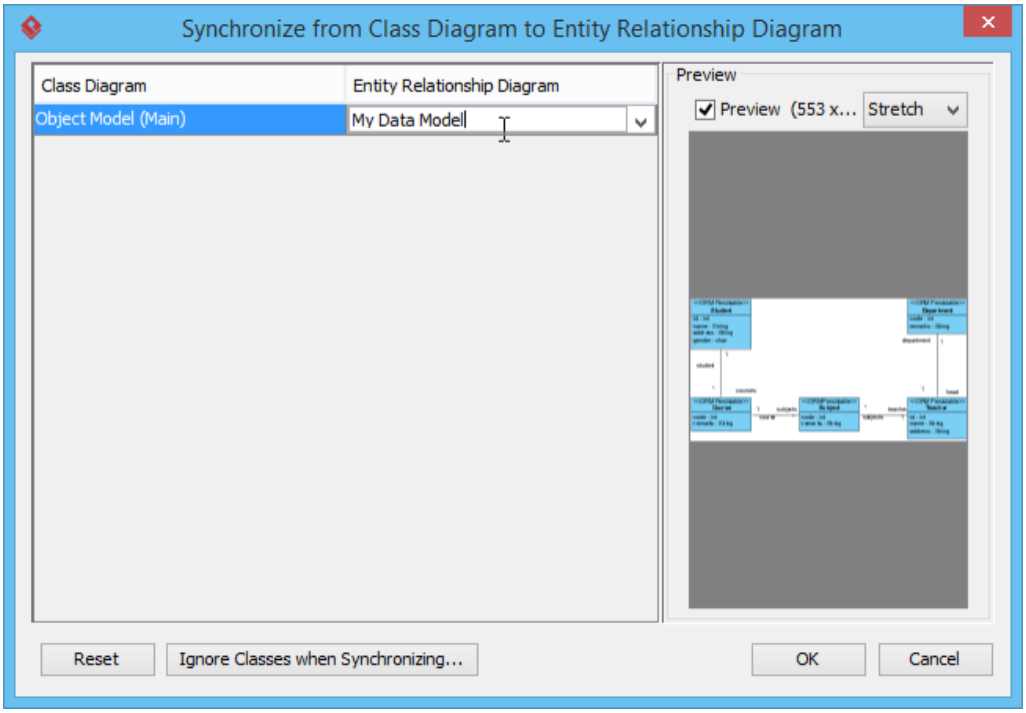
Visual Paradigm supports mapping object model (class diagram) with data model (Entity Relationship Diagram). Such mapping enables the production of programming source code from object model, which can be used in developing applications that require accessing database.

In order for the application to work properly, it is important that the object model and data model are conformed with each other. In Visual Paradigm, you can make them conformed by synchronizing them manually.

Synchronize object model to data model

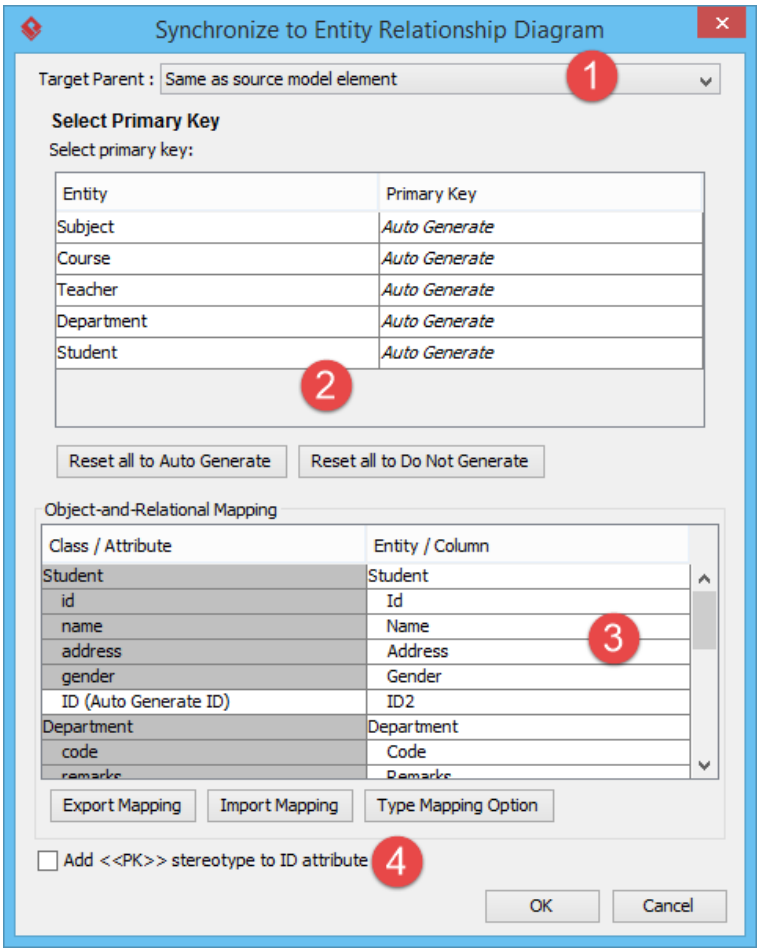
To synchronize object model to data model, perform the steps below.

1. Select **Tools > Hibernate > Synchronize to Entity Relationship Diagram** from the toolbar.
2. One class diagram maps with one ERD. If any of your class diagram has not been mapped with an ERD, you will be prompted to select the ERD to map with. Upon finishing, entities will be visualized in the ERD selected, following the corresponding class diagram. You can select an existing ERD from the drop down menu, or enter a new diagram name to create a new ERD.



Naming target ERD

3. Click **OK**.
4. This shows the Synchronize to Entity Relationship Diagram. There are primarily four actions you can/have to do in this window, as described in the table below.



The Synchronize to Entity Relationship Diagram window

No.	Name	Description
1	Target Parent	Click on the drop down menu to specify the placement of entities to be created, if any. **Root** - Place the new entities directly under project root. **Same as source model element** - Place the new entities to the same parents as the corresponding classes. **Specify** - Choose a model to place the entities yourself.
2	Select Primary Key	Select primary keys for entities without primary keys specified yet. You can choose a column of an entity as its primary key, or select **Auto Generate** to let Visual Paradigm generate a new primary key column for you, namely *ID*, or select **Do Not Generate** to leave the entity without primary.
3	Object-and-Relational Mapping	Confirm the naming for new entities and/or columns. By default, the entities and columns will be named following the corresponding classes and attributes. If you want different names you can double click on the corresponding cells and enter yourself. You can also select the column type to map to for string, boolean and byte[] attributes, by clicking **Type Mapping Option** and choosing the target types in the popup dialog box. Besides, you can export the mapping as an Excel file by clicking **Export Mapping**, and then edit the mapping externally and import it back by clicking **Import Mapping**.
4	Add <<PK>> stereotype to ID attribute	In ERD, primary keys are represented with a key icon, but in class diagram, there is no indication for attributes that map to a primary key column. If this information is important to you, you can check this option to let us stereotype attributes that map to primary keys as <<PK>>. The stereotype will be visualized in diagram so that you can differentiate such attributes from other attributes.

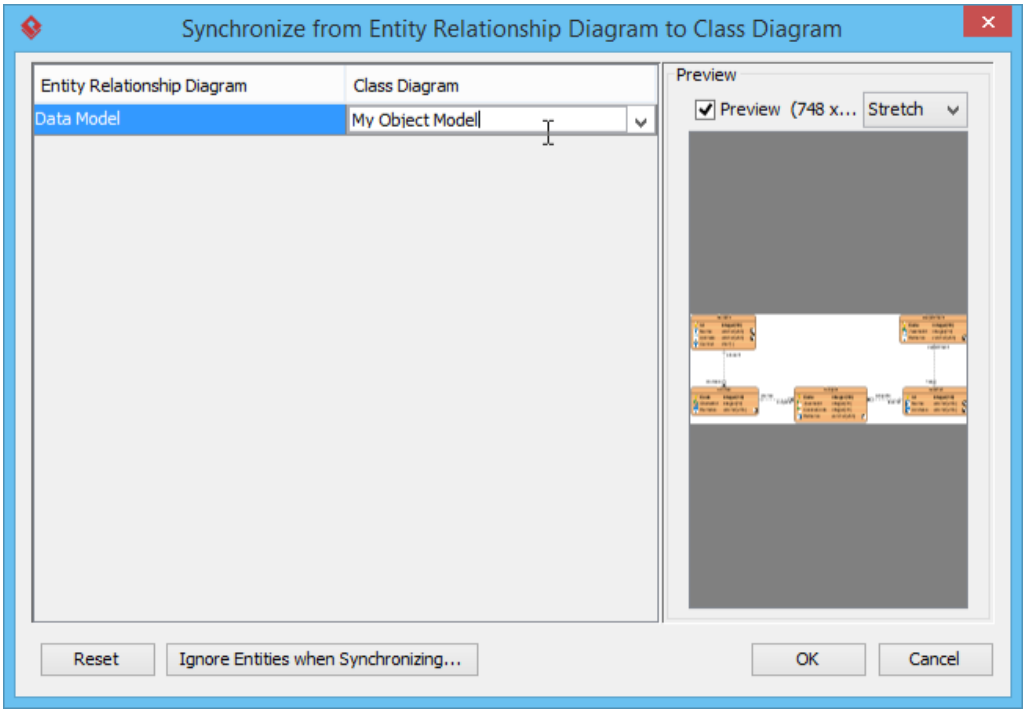
Description of the Synchronize to Entity Relationship Diagram window

5. Click **OK**. Your object model is now synchronized to data model. Click here if you want to learn more about the detailed mapping from object model to data model.

Synchronize data model to model model

To synchronize data model to object model, perform the steps below.

1. Select **Tools > Hibernate > Synchronize to Class Diagram** from the toolbar.
2. One ERD maps with one class diagram. If any of your ERD has not been mapped with a class diagram, you will be prompted to select the class diagram to map with. Upon finishing, classes will be visualized in the class diagram selected, following the corresponding ERD. You can select an existing class diagram from the drop down menu, or enter a new diagram name to create a new class diagram.



Naming target class diagram

3. Click **OK**.
4. This shows the Synchronize to Class Diagram. There are primarily three actions you can/have to do in this window, as described in the table below.

The Synchronize to Class Diagram window

No.	Name	Description
1	Target Parent	Click on the drop down menu to specify the placement of classes to be created, if any. **Root** - Place the new classes directly under project root. **Same as source model element** - Place the new classes to the same parents as the corresponding entities. **Specify** - Choose a model to place the classes yourself.
3	Object-and-Relational Mapping	Confirm the naming for new classes and/or attributes. By default, the classes and attributes will be named following the corresponding entities and columns. If you want different names you can double click on the corresponding cells and enter yourself. You can also select the attribute type to map to for columns in decimal and numeric types by clicking **Type Mapping Option** and choosing the target type in the popup dialog box. Besides, you can export the mapping as an Excel file by clicking **Export Mapping**, and then edit the mapping externally and import it back by clicking **Import Mapping**.
4	Add <<PK>> stereotype to ID attribute	In ERD, primary keys are represented with a key icon, but in class diagram, there is no indication for attributes that map to a primary key column. If this information is important to you, you can check this option to let us stereotype attributes that map to primary keys as <<PK>>. The stereotype will be visualized in diagram so that you can differentiate such attributes from other attributes.

Description of the Synchronize to Class Diagram window

5. Click **OK**. Your data model is now synchronized to object model. Click here if you want to learn more about the detailed mapping from object model to data model.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Visual Paradigm Database Management Guides

Reverse Engineering ERD from Database

Reverse engineering ERD from existing database. By visualizing a database schema in ERD, you can re-edit it and patch changes back to the database. You will learn how the reverse engineering of ERD works in this article.

Generating Database from ERD

Once you have designed your database with ERD, you can generate the actual database from it, or to generate the DDL file for manual database creation. This page shows you how to generate database from ERD.

Patching Design Changes to Database

Compare your ER design and database, and patch the differences to your database accordingly. This article will show you how database patching works.

Copying SQL Statements from Entities in ERD

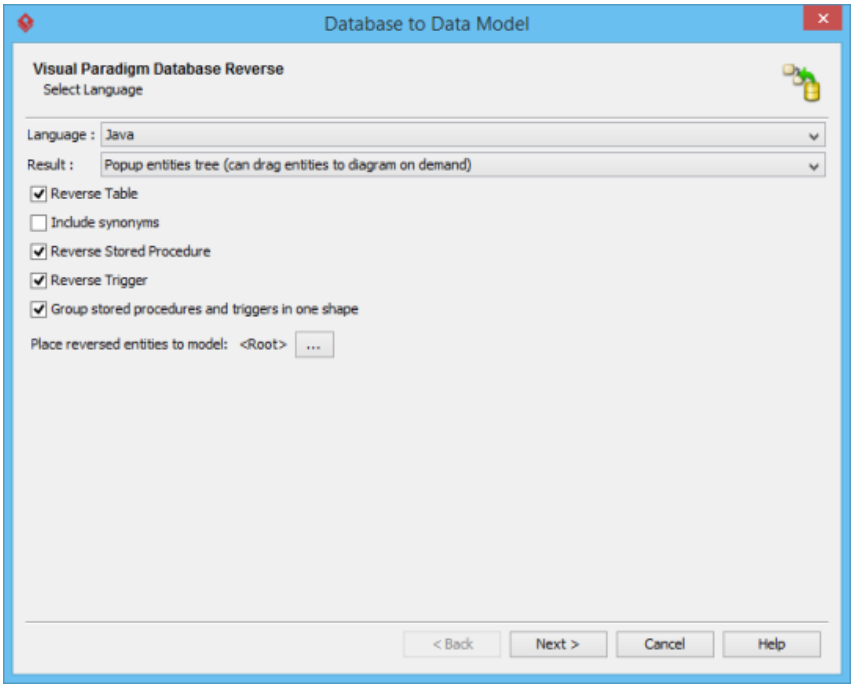
Copy the SQL statements you need in creating, altering and deleting data in database. In this page you will see how the copy SQL feature works.

Reverse Engineering ERD from Database

Visual Paradigm supports reverse engineering ERD from existing database. By visualizing a database schema in ERD, you can re-edit it and patch changes back to the database, or you can produce a data specification that acts as a reference for programmers who need to access the database. Besides, Visual Paradigm automates object-relational mapping, object model can thus be generated from the data model produced by reverse engineered.

To reverse engineering ERD from database:

1. Select **Tools > DB > Reverse Database...** from the toolbar. This opens the **Database to Data Model** window.



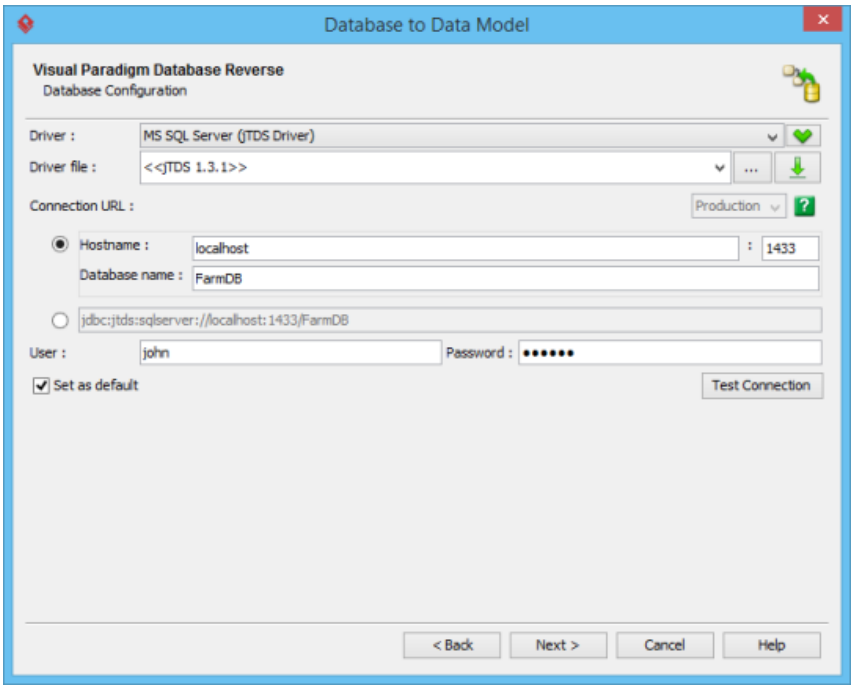
The Data to Data Model window

2. In the **Database to Data Model** window, set the language, which is the method to use in connecting to your database. By selecting Java, JDBC will be used. By selecting C#, adapter file will be used. If you are uncertain about this option, just leave it unchanged.
3. Select the Result, which is the post-action of database reversal. Here is a list of available options:

Result	Description
Popup entities tree	You will be presented a window with reversed entities listed. You can form ERD(s) manually by dragging and dropping entities from that window onto diagram(s).
Form a new diagram with the reversed entities	Automatically create an ERD and place all the reversed entities in it. Make sure you are clear about the amount of entities that will be reversed and the complexity of diagram that will be formed. Having too many entities on one diagram may result in a poor database design that is hard to read and maintain.
Add reversed entities to ERD	Automatically place all the reversed entities onto the first ERD in your project. Again, make sure the amount of entities to be reversed and the complexity of diagram is under your control.
Do not form diagram with the reversed entities	Just produce entities without visualizing them with any ERD. You can visualize them later on if you want to.

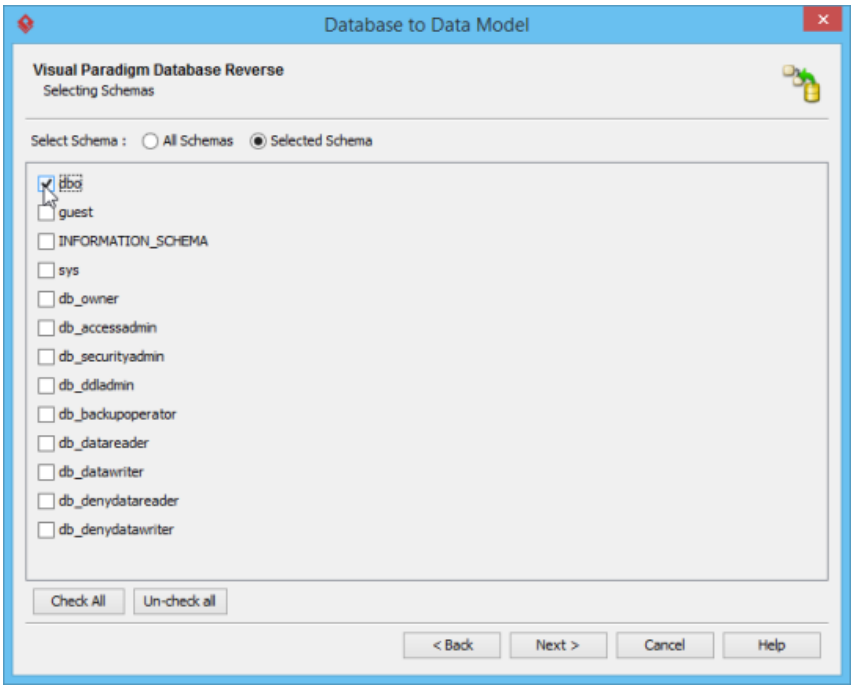
List of available results of database reversal

4. Sometimes, you may just want to reverse engineer stored procedures or triggers. If that's the case, uncheck **Reverse Table**.
5. Synonym is an alternative name for objects such as tables, view, etc. If you are going to reverse an Oracle database, you can optionally check **Include synonyms** to reverse synonyms as entities.
6. Keep **Reverse Stored Procedure** checked if you want to restore stored procedures.
7. Keep **Reverse Trigger** checked if you want to restore database triggers.
8. By checking **Group stored procedures and triggers in one shape**, we will create at most one stored procedures shape and one triggers shape to hold all the reversed procedures and triggers. When unchecked, one procedures shape will be created for each procedure being reversed.
9. You can also place the reversed entities into a dedicated model by clicking **...** next to **Place reversed entities to model** to select the model in which you want the entities be placed.
10. Click **Next**.
11. Configure the database connection and click **Next**.



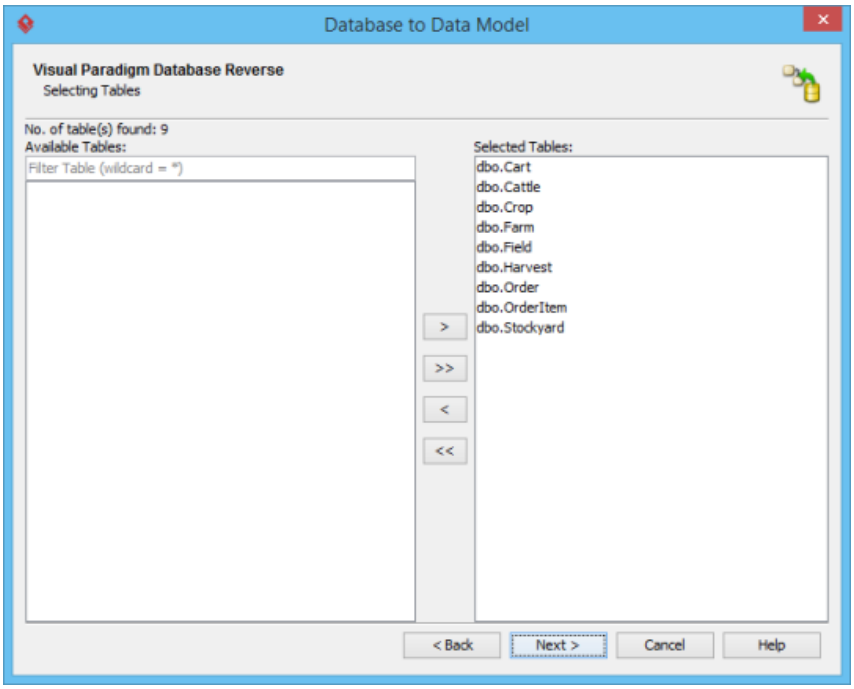
Configure database connection

12. For DBMS like MSSQL, you can select the schema to reverse. Make your choice and click **Next**. Skip this step if you aren't prompted to select schema.



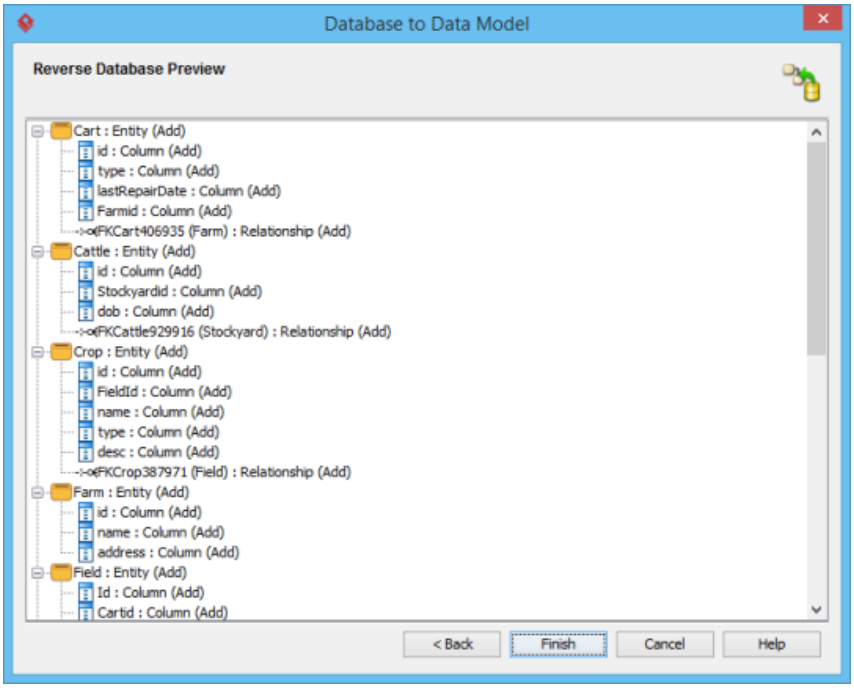
Select schema

13. By default, all database tables in the specified database (or schema) will be reversed. If you want to reverse just some tables, select the tables you do not want and click **<** to remove them from the list of Selected Tables. Click **Next**.



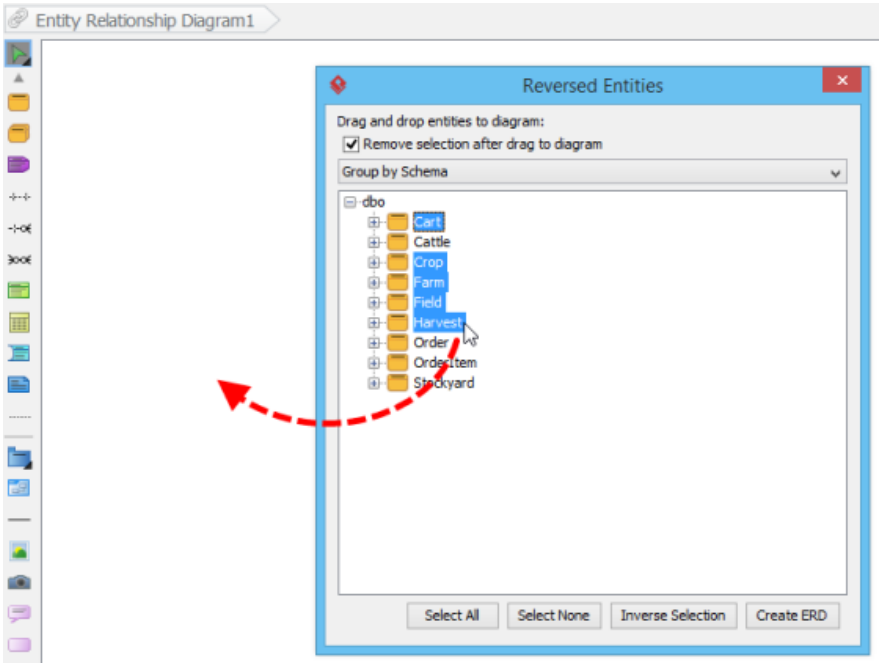
Select tables to be reversed

14. Preview the entities that will be created and click **Finish** to execute the reversal.



Preview the entities to be created

15. When finished, you can optionally form an ERD from the reversed entities. If you have selected Popup entities tree as result in step 2, you are now prompted a window with all the reversed entities listed. You can form diagram by draggin them onto an ERD.



Form ERD with the reversed entities

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

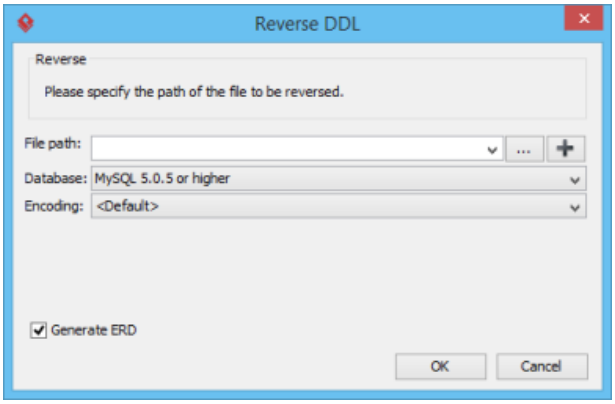
- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Reverse Engineering ERD from DDL

Visual Paradigm supports reverse engineering ERD from .ddl and .sql. It forms nice Entity Relationship Diagram out of the create and alter statements written in DDL (Data Definition Language). With an ERD, you can produce a data dictionary, or to revise the design and generate a database, etc.

To reverse engineering ERD from DDL:

1. Select **Tools > DB > Reverse DDL...** from the toolbar. This opens the **Reverse DDL** window.



The Reverse DDL window

2. In the **Reverse DDL** window, enter the file path of the .sql or .ddl file to reverse.
3. Select the kind of database that supports your DDL file.
4. Select the encoding of the .sql or .ddl file. Leave it **<Default>** if you are uncertain.
5. If you want to form an Entity Relationship Diagram automatically after reversal, keep Generate ERD reversed. If you uncheck it, entity elements will be formed without creating any diagrams. No matter what, you can visualize entities later on if you want to.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

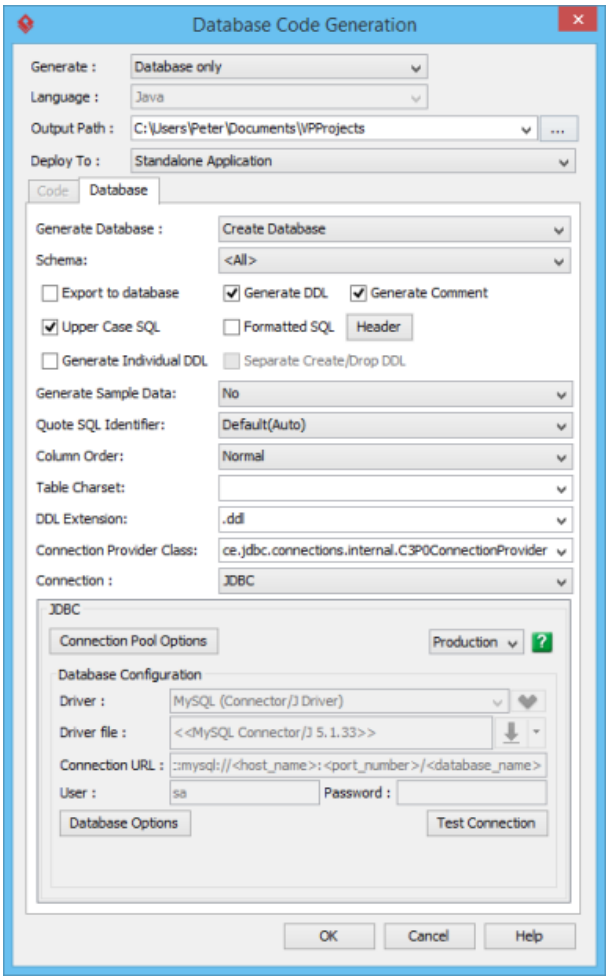
- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Generating Database from ERD

Once you have designed your database with ERD, you can generate the actual database from it, or to generate the DDL file for manual database creation.

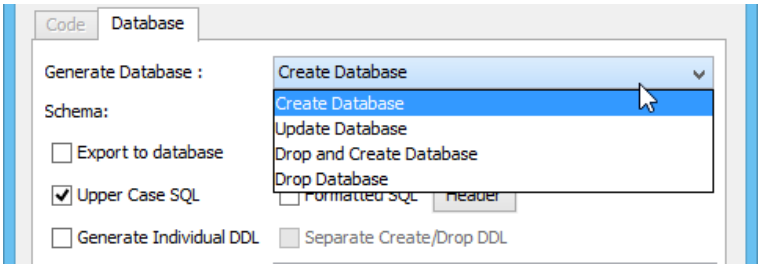
Generating Database from ERD

1. Select **Tools > DB > Generate Database...** from the toolbar. This opens the **Database Code Generation** window.



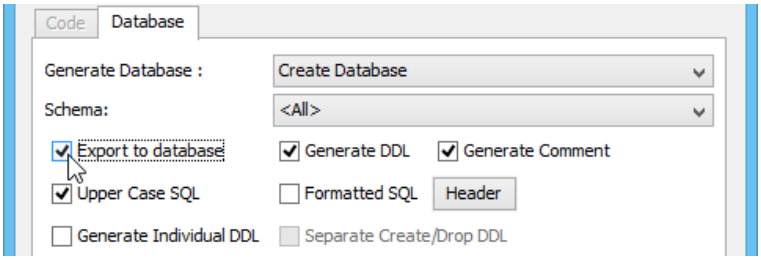
Database Code Generation

2. Select **Create Database** for **Generate Database**.



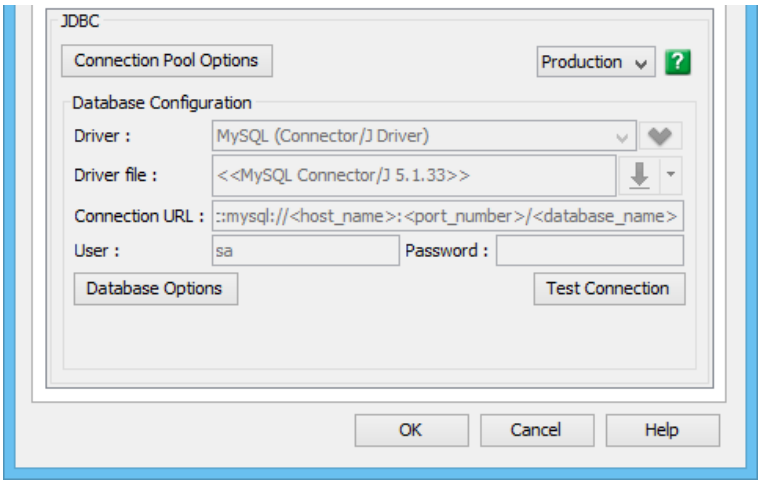
Create database

3. Check and/or uncheck **Export to database** and **Generate DDL**. Sometimes, you may want to review and modify the DDL scripts for database creation, instead of having the database be created completely based on your design. If this is what you want, uncheck **Export to database** and check **Generate DDL**. If you want database to be created automatically based on your ERD, check **Export to database**.



Select Export to database

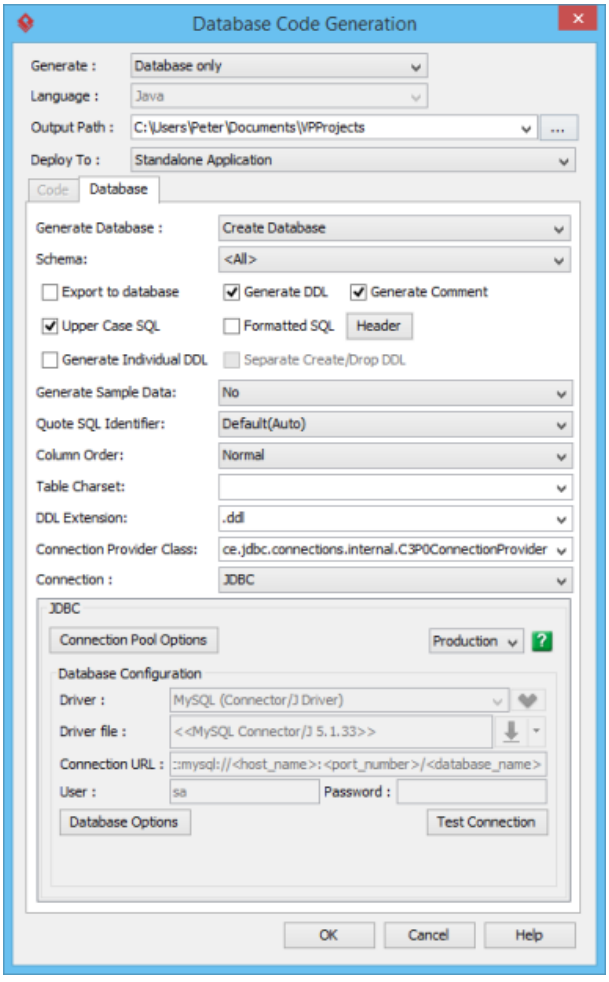
4. Make sure you have chosen and configured the default database. You can only generate database if you have chosen the default database.



Default database specified

5. Configure the other generation options. Read the next sections for details about those available options.
6. Click **OK**. If succeed, and if you have chosen **Export to database**, you will see the database schema be created in the chosen database. If you have chosen **Generate DDL**, you can find the DDL file in the folder specified in the **Output Path** field.

Overview of Database Code Generation



Database Code Generation window

Option	Description
Generate	Create Database - generate create statements only. Update Database - query existing object in database, generate create and alter statement depends on object not exists or outdated in database, or do nothing if database is up-to-date. Drop and Create Database - generate drop statements first, then generate create statements. Drop Database - generate drop statements only.
Language	The language of your application to be developed. Just leave it as-is if you are uncertain or not going to develop any applications.

Output Path	The folder to store the generated DDL files, source files and library files (if any).
Deploy To	Specify the template setting for the purpose of the generated code. This setting affects the generated optional Jar and Datasource setting in database connection. You can select Standalone Application, WebLogic Application Server 8.1/9.0, WebSphere Application Server Community Edition 1.0, JBoss Application Server and Generic Application server.
Generate Database	The kind of action to perform, which will influence the outcome. For example, if you select **Create Database**, a new database will be created base on your ERD. If you select **Drop Database**, database tables modeled in ERD will all be dropped in database.
Schema	Select the schema for database generation.
Export to database	Check this option to create tables in the **actual** DBMS.
Generate DDL	DDL (Data Definition Language) is a syntax for defining data structure. E.g. CREATE TABLE PHOTO (...); DROP TABLE PHOTO; Very often, you don't have direct access to the production database during designing/development. By using a DDL, you can execute the table create scripts on the production database manually, and to automate the table creation process as part of the database re-initialization routine.
Generate Comment	Generate entity and column description as comments of tables and their columns. (Only available to My SQL, DB2, Oracle, Postgre SQL)
Upper Case SQL	Force the content of the generated DDL to be in upper case.
Formatted SQL	Apply proper line breaking and indentation to make the content looks prettier.
Formatted SQL - Header	Optionally include the date and time of generating DDL, the author as well as custom description. You can also enter custom DDL for stating the assumption or the DB configuration required for executing the DDL script (e.g. must run in a blank DB).
Generate Individual DDL	Split table creation statements and foreign key constraint creation statements into two files.
Separate Create/Drop DDL	Split table creation and drop statements into two files.
Generate Sample Data	You can enter sample table records for entities. This option enables the generation of sample table records into database. This allows the same team to share a common set of sample data when in development and testing.
Quote SQL Identifier	Words like SELECT, ORDER are known as reserved words. Reserved words are permitted as identifiers if they are quoted in SQL statements. E.g. CREATE TABLE 'Order'... You can keep this option "Auto" or "Yes" to let us add proper quotes for you. However, we don't recommend the use of reserved words. This is to avoid potential errors.
Column Order	By default, we generate CREATE TABLE statement by following the column order presented in ER model, You can enforce the column order by having key and index column created before the other columns. It can be a good practice to have key and index generated first as this may avoid potential problems in data insertion.
Table Charset	Select the table charset to use, such as UTF8, BIG5, GB2312 and LATIN1. Only available for MySQL users.
DDL Extension	Generate the DDL as .ddl file(s) or .sql file(s).
Connection Provider Class	A strategy for obtaining JDBC connections. Just leave it as-is if you are unsure.
Connection	For use with Hibernate (ORM). JDBC - standard Java database API. Datasource - use database connection from application server. JDBC + Datasource - generate two configuration files for JDBC and database.
JDBC	Connection Pool Options - Connect to and disconnect from the database is an expensive operation, using the connection pool to share the opened connection can dramatically increase the application performance. You can deselect the Use connection pool option to disable the use of connection pool. Production - Normally, database connectivity is stored in project and shared among team members in a team environment. If you have your own database connection setting for your environment, you can select **Personal** and enter the connection URL there. The setting you filled will not be committed to server, which means that you can keep your own set of database connection setting. For details, please read the next section. Database Options - Set and configure database. Note that you need to select a default DBMS in order for database generation to function. Test Connection - Click to verify the connection settings entered.

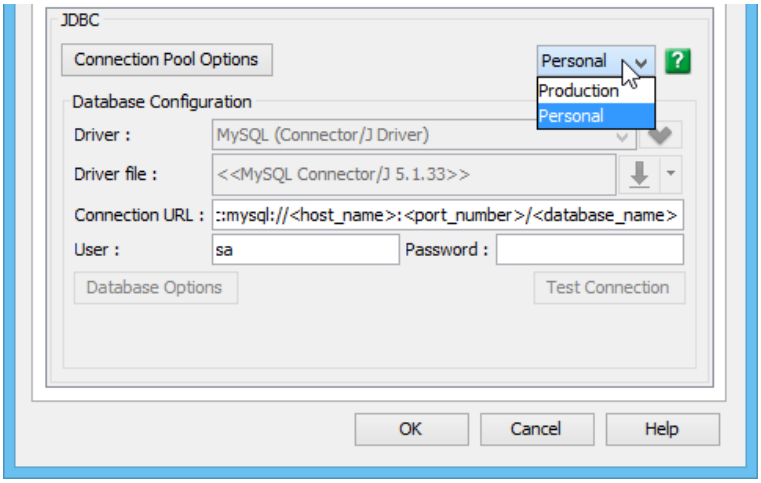
Overview Database Code Generation

Personal/Production Connectivity

Database is one of the key components of an information system. In order to keep a system stable, separate databases are used for development and production so that database changes can be made freely. And when tested, changes can be patched to the production database.
Visual Paradigm enables you to share database design through VPository.com, a cloud -based repository of software design projects. This allows development and production environments to work consistently with the same and latest set of database design. Although different environments can share the same database design, it is impractical to have them share the same set of database connection information. For instance, developer Peter may have 'demo' as database name, with 'peter' as login, while developer Mary may have 'dev' as database name and 'mary' as login. The production environment, certainly won't name the database as 'demo' or 'dev'.

Due to the fact that different environments can have their own set of database settings, it is a bad idea to bind the database connection setting required in generating ORM code and updating database along with the design. If that's the case, all users will require a re-configuration of setting for their own environments prior to generating ORM code and/or updating database schema. This is not just cumbersome, but may corrupt important data in database if a wrong connection setting is supplied by mistake. For these reasons, Visual Paradigm supports personalizing the database connection information so that each user can have their own set of database connection setting. To be specific, the personalized setting is stored within workspace instead of project. Therefore, it will not affect any other user and the production environment. It affects only the generated Hibernate files and the database in specific environment.

To use this feature, simply click on the drop-down menu labeled **Production**, inside the JDBC section. Then, select **Personal** and enter your own connection URL. Database generation will apply the connection URL you supplied.



Select Personal database setting

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

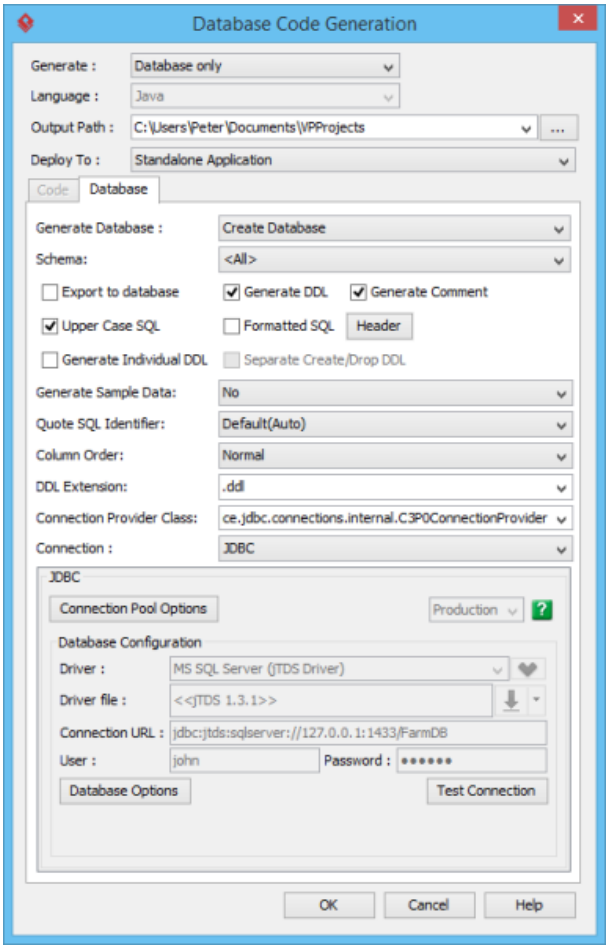
Patching Design Changes to Database

Information systems keep changing at all time. Very often, those changes involves the introduction of new business rules, which requires storing new data, and thus requires the introduction of new database tables, columns and relationships. Since database is an integral part of an information system, to modify database structure we cannot just throw away the old one an re-create everything from scratch. Instead, we need to take good care of existing data and structure to ensure the system will remain working properly both throughout and after the transition.

Visual Paradigm supports the patching of design changes to existing database. We compare the differences between your design and a database, and patch the differences to your database accordingly. Throughout the process, your existing data will remain intact. You don't need to worry about data lost or system stability.

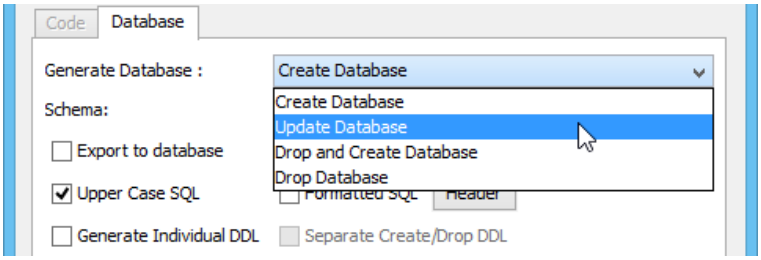
Patching Database from ERD

1. Select **Tools > DB > Generate Database...** from the toolbar. This opens the **Database Code Generation** window.



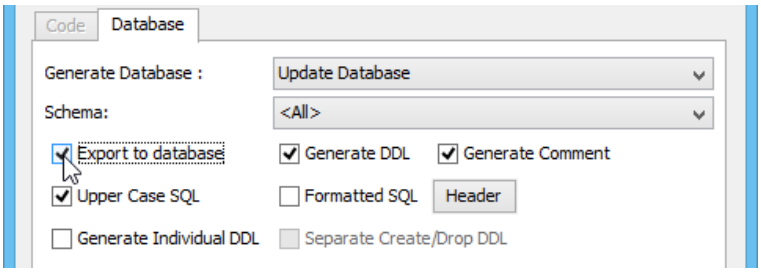
Database Code Generation window

2. Select **Update Database** for **Generate Database**.



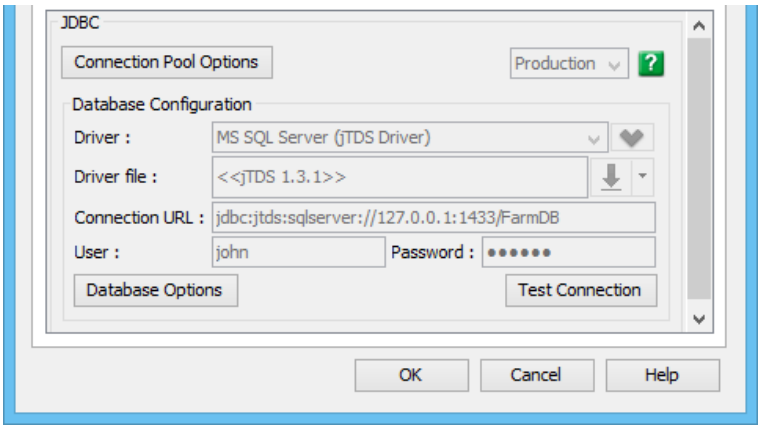
Update database

3. Check and/or uncheck **Export to database** and **Generate DDL**. Sometimes, you may want to review and modify the DDL scripts for database patching, instead of having the database be altered completely based on your updated design. If this is what you want, uncheck **Export to database** and check **Generate DDL**. If you want database to be patched automatically based on your ERD, check **Export to database**.



Select Export to database

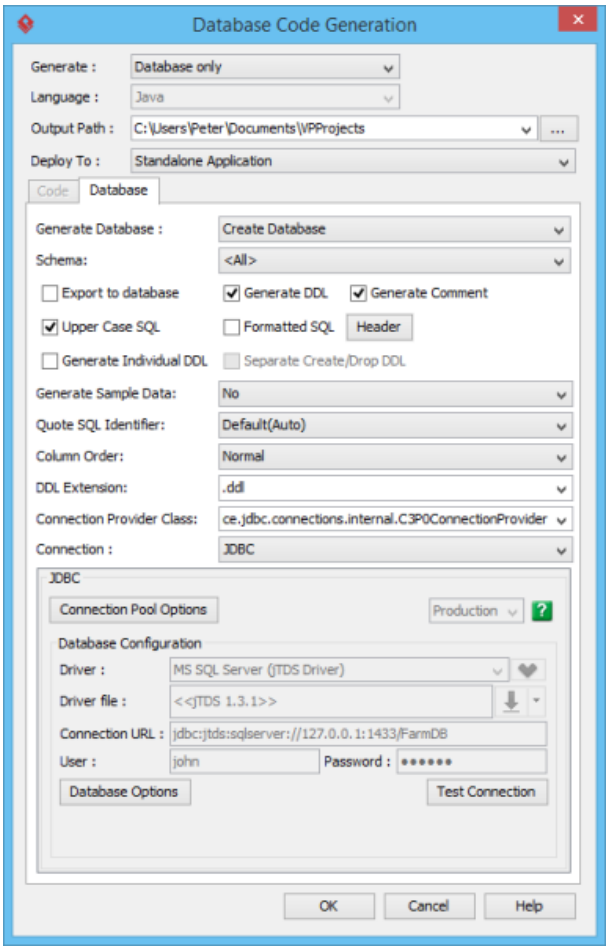
4. You can check **Include drop table/columns that not exist in ER model** if you want the patching process to drop tables and/or columns that do not exist in your ER diagram. Note that dropping a column or table will disregard the data contained. Therefore, think carefully if you want to enable this option.
5. Make sure you have chosen and configured the default database. You can only patch database if you have chosen the default database.



Default database specified

6. Configure the other generation options. Read the next sections for details about those available options.
7. Click **OK**. If succeed, and if you have chosen **Export to database**, you will see the database be patched. If you have chosen **Generate DDL**, you can find the DDL file in the folder specified in the **Output Path** field.

Overview of Database Code Generation



Database Code Generation window

Option	Description
Generate	Create Database - generate create statements only. Update Database - query existing object in database, generate create and alter statement depends on object not exists or outdated in database, or do nothing if database is up-to-date. Drop and Create Database - generate drop statements first, then generate create statements. Drop Database - generate drop statements only.
Language	The language of your application to be developed. Just leave it as-is if you are uncertain or not going to develop any applications.
Output Path	The folder to store the generated DDL files, source files and library files (if any).
Deploy To	Specify the template setting for the purpose of the generated code. This setting affects the generated optional Jar and Datasource setting in database connection. You can select Standalone Application, WebLogic Application Server 8.1/9.0, WebSphere Application Server Community Edition 1.0, JBoss Application Server and Generic Application server.
Generate Database	The kind of action to perform, which will influence the outcome. For example, if you select **Create Database**, a new database will be created base on your ERD. If you select **Drop Database**, database tables modeled in ERD will all be dropped in database.
Schema	Select the schema for database generation.
Export to database	Check this option to create tables in the **actual** DBMS.
Generate DDL	DDL (Data Definition Language) is a syntax for defining data structure. E.g. CREATE TABLE PHOTO (...); DROP TABLE PHOTO; Very often, you don't have direct access to the production database during designing/development. By using a DDL, you can execute the table create scripts on the production database manually, and to automate the table creation process as part of the database re-initialization routine.
Generate Comment	Generate entity and column description as comments of tables and their columns. (Only available to My SQL, DB2, Oracle, Postgre SQL)
Upper Case SQL	Force the content of the generated DDL to be in upper case.

Formatted SQL	Apply proper line breaking and indentation to make the content looks prettier.
Formatted SQL - Header	Optionally include the date and time of generating DDL, the author as well as custom description. You can also enter custom DDL for stating the assumption or the DB configuration required for executing the DDL script (e.g. must run in a blank DB).
Generate Individual DDL	Split table creation statements and foreign key constraint creation statements into two files.
Separate Create/Drop DDL	Split table creation and drop statements into two files.
Generate Sample Data	You can enter sample table records for entities. This option enables the generation of sample table records into database. This allows the same team to share a common set of sample data when in development and testing.
Quote SQL Identifier	Words like SELECT, ORDER are known as reserved words. Reserved words are permitted as identifiers if they are quoted in SQL statements. E.g. CREATE TABLE 'Order'... You can keep this option "Auto" or "Yes" to let us add proper quotes for you. However, we don't recommend the use of reserved words. This is to avoid potential errors.
Column Order	By default, we generate CREATE TABLE statement by following the column order presented in ER model, You can enforce the column order by having key and index column created before the other columns. It can be a good practice to have key and index generated first as this may avoid potential problems in data insertion.
Table Charset	Select the table charset to use, such as UTF8, BIG5, GB2312 and LATIN1. Only available for MySQL users.
DDL Extension	Generate the DDL as .ddl file(s) or .sql file(s).
Connection Provider Class	A strategy for obtaining JDBC connections. Just leave it as-is if you are unsure.
Connection	For use with Hibernate (ORM). JDBC - standard Java database API. Datasource - use database connection from application server. JDBC + Datasource - generate two configuration files for JDBC and database.
JDBC	Connection Pool Options - Connect to and disconnect from the database is an expensive operation, using the connection pool to share the opened connection can dramatically increase the application performance. You can deselect the Use connection pool option to disable the use of connection pool. Production - Normally, database connectivity is stored in project and shared among team members in a team environment. If you have your own database connection setting for your environment, you can select **Personal** and enter the connection URL there. The setting you filled will not be committed to server, which means that you can keep your own set of database connection setting. For details, please read the next section. Database Options - Set and configure database. Note that you need to select a default DBMS in order for database generation to function. Test Connection - Click to verify the connection settings entered.

Overview Database Code Generation

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

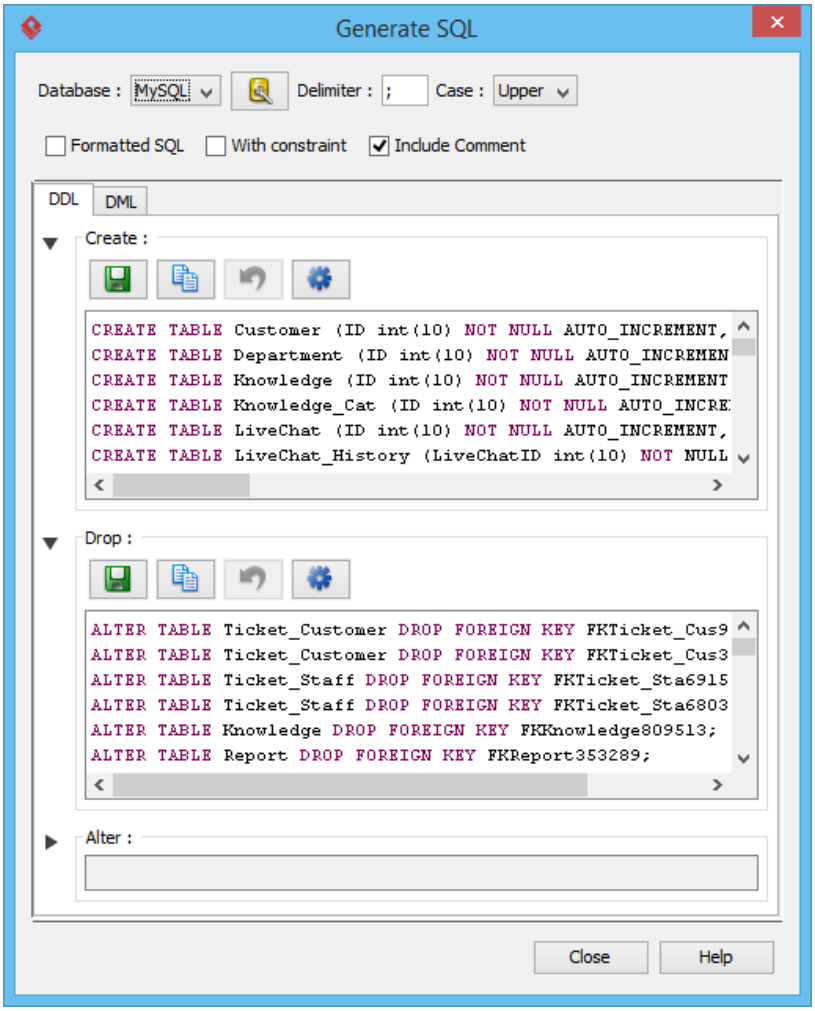
Copying SQL Statements from Entities in ERD

Visual Paradigm provides function of copying SQL statements from entities in ERD. It provides you with a handy approach to copy the SQL statements you need in creating, altering and deleting data in database. By copying the SQL statements you can make necessary change to the statements and then execute them in your database.

Note that in order to use this function, you must configure the database setting in advance as Visual Paradigm will generate the SQL statements according to the default DBMS selected.

Copying SQL statements for all entities in ERD

1. Right click on the background of an ERD and select **Utilities > Generate SQL...** from the popup menu.
2. In the **Generate SQL** window, you can copy the **DDL** statements (Create/Drop/Alter) or **DML** statements (Select/Insert/Update/Delete). If you have configured multiple DBMS, select the desired DBMS from the **Database** drop down menu. The corresponding SQL statements will be displayed accordingly.



Generate SQL window

The Generate SQL window also provides you with several configuration options to the statements.

Option	Description
Delimiter	Delimiter of each SQL statement. Use \n to represent line break, \\n to represent '\n'.
Case	Force the content of the generated DDL to be in upper or lower case.
Formatted SQL	Apply proper line breaking and indentation to make the statements formatted prettier.
With constraint	Let's say, for example if entity X has a FK that references entity Y, if you only selected to generate SQL for entity X, with this option checked, alter statements will also generated for adding the necessary constraint to entity Y.
Include Comment	Generate entity and column description as comments of tables and their columns. (Only available to My SQL, DB2, Oracle, Postgre SQL)

Configuration options

Besides copying SQL statements, you can execute them directly from the **Generate SQL** window by clicking  above the statements.

Copying SQL statements from a selected scope

An ERD may be large and contains many entities, or even all entities of your database. Sometimes, you may want to get SQL statements from only some of the entities in an ERD, but not all. Visual Paradigm provide you with two options:

Some entities in an ERD

Select the entities and relationships on the ERD. Right click on the selection and select **Generate SQL...** from the popup menu. In the Generate SQL window the DDL and DML statements of the chosen entities and relationships are presented.

Relationship(s) in an ERD

If you want to obtain the SQL statements for creating or dropping constraints, choose this option. Select the desired relationships. Right click on them and select **Generate SQL...** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Programming Guides (ORM) for Visual Paradigm Users

PersistentManager and Transaction

PersistentManager is used to manage database connection, state of the persistent objects and transaction when the application is running. In this article you will learn more about PersistentManager.

Using ORM Qualifier

ORM Qualifier allows you to specify extra data retrieval rules apart from the system pre-defined rules. In this article we will show you how to define and use ORM Qualifier.

Using ORM Criteria

ORM Criteria is one of the handy approaches that aids data retrieval in writing a program. In this article we will show you how it works.

What is Object Relational Mapping (ORM)?

In programming, Java developers used to take the JDBC approach for data persistence, which involves writing and executing SQL statements in retrieving/updating data in database. This approach was popular, but probably not a good one because SQL can be difficult to read and write, error prone and hard to debug.

To make programming easier and reduce the chance of making mistakes, many developers prefer not to execute SQL statements directly, but to build an object model that reflects the data structure. In runtime, data will be retrieved from database and filled into the object model. Developers can then work entirely with objects, without writing any SQL statements. The technique to convert data between object model and relational database is known as object-relational mapping (ORM, O/RM and O/R mapping).

Hibernate

A well-built object model can be very useful both in programming and debugging. However, to build an object model is not that easy. While programming language like Java is object-oriented, which represents data as interconnected graph of objects, relational database, on the contrary, represents data in tabular format (like spreadsheet). You need to tackle the mismatch between object model and relational model if you decide to write such a tier, not to mention the amount of time and work required in programming and debug the model, which is always propotional to the scale of your system. All these make data persistence difficult.

There are both free and commercial packages you can use to perform object-relational mapping. Hibernate is one of the most popular packages. Visual Paradigm supports the generation of Hibernate ORM from your database design (ERD). Visual Paradigm can also synchronize your database design to object model (Class Diagram). You can then generate Java objects to use in your program.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

PersistentManager and Transaction

PersistentManager is used to manage database connection, state of the persistent objects and transaction when the application is running. A database transaction is a unit of the application. It is used to keep integrity of your data in database. Our Persistent Library provides a transaction API for you to create, commit and rollback the transaction for your application.

Start Transaction

When you need to start a transaction, you need to get the session from the PersistentManager and call the beginTransaction function (PersistentManager.instance().getSession().beginTransaction();). Here is a sample:

```
PersistentTransaction t = PersistentManager.instance().getSession().beginTransaction();
```

Commit Transaction

After you have inserted/updated/deleted data, you can call the commit function of the transaction object to commit your changes to Database.

```
t.commit();
```

Rollback Transaction

In case there are any exception, you can call the rollback function to cancel all the operation.

```
t.rollback();
```

Sample Code

The following is an example of using the transaction API.

```
private Course fireOK() throws PersistentException {
   PersistentTransaction t = SchoolSystemPersistentManager.instance().getSession().beginTransaction();
   try {Course lCourse = CourseFactory.createCourse();
      lCourse.setTitle(getTitleTextField().getText());
      lCourse.setDescription(getDescriptionTextField().getText());
      lCourse.setTeacher(_teacher);
      _teacher.save();
      t.commit();
      return lCourse;
   } catch (Exception e) {
      e.printStackTrace();
      t.rollback();
   }
   return null;
}
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

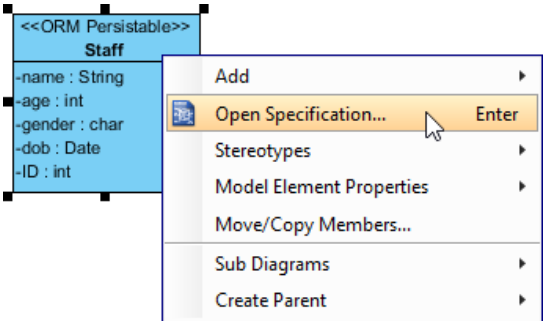
Using ORM Qualifier

ORM Qualifier allows you to specify extra data retrieval rules apart from the system pre-defined rules. With ORM Qualifier, you can define retrieval rules that match your business logic. Let's say your program requires retrieving staff records for specific gender. Without using ORM Qualifier, you will need to retrieve all records, and check record by record to find out the staff that match the gender you look for. With ORM Qualifier, you can retrieve data with specialized methods like listByGender(char gender). This not only makes the code looks tidier but also reduce the chance of making error.

Defining ORM Qualifier in object model

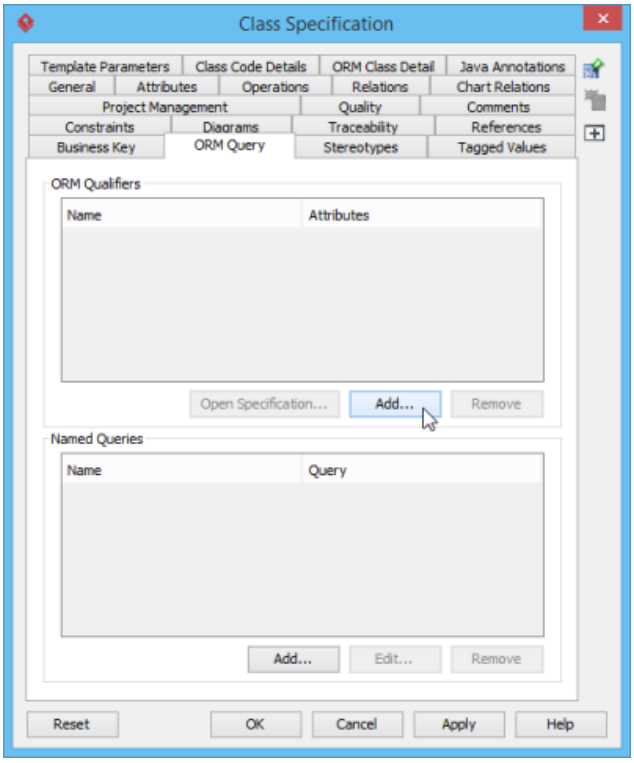
In order to use ORM Qualifier you need to define the qualifier(s) in your object model by selecting the attribute(s) to be included in qualifier.

1. Right click on the ORM Persistable class in which you want to define an **ORM Qualifier**.



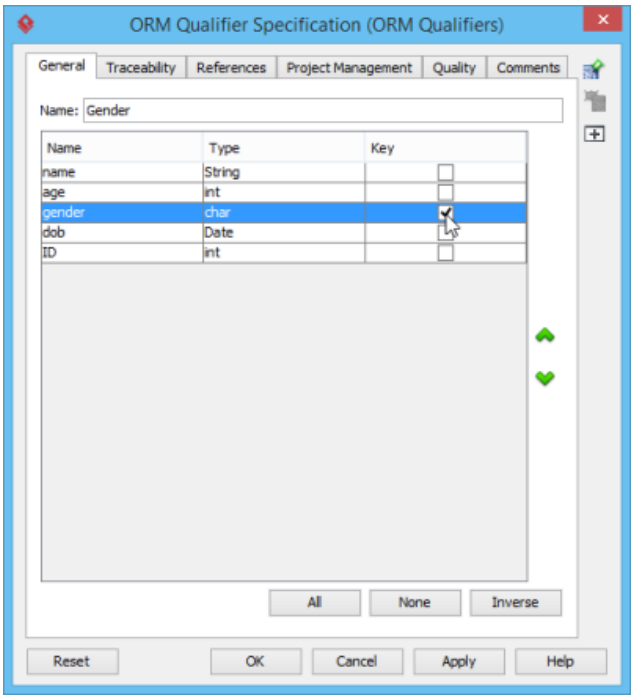
Open class specification

2. Open the **ORM Query** tab.
3. Click **Add...** in the **ORM Qualifiers** section.



Add a ORM Qualifier

4. In the **ORM Qualifier Specification** window, enter the name of the ORM Qualifier and select the key attribute. The name you defined will be appended to method names in generated code. For example, an ORM qualifier named *Gender* will result in generating methods like *loadByGender(...)*, *listByGender(...)*, etc.



Defining an ORM Qualifier

5. Click **OK** to confirm.
6. Click **OK** to return to diagram.

ORM Qualifier (generated code)

ORM Qualifier methods will be generated in Persistent class according to the selected Persistent API. For example, if you selected Factory class as Persistent API, then the following methods will be generated in the StaffFactory class.

Return Type	Method Name	Sample	Description
Class	loadByORMQualifier (DataType attribute)	loadByGender(char gender)	Retrieve the first record that matches the specified value with the attribute defined in the ORM Qualifier.
Class	loadByORMQualifier (PersistentSession session, DataType attribute)	loadByGender(PersistentSession session, char gender)	Retrieve the first record that matches the specified value with the attribute defined in the ORM Qualifier and specified session.
Class[]	listByORMQualifier (DataType attribute)	listByGender(char gender)	Retrieve the records that match the specified value with the attribute defined in the ORM Qualifier.
Class[]	listByORMQualifier (PersistentSession session, DataType attribute)	listByGender(PersistentSession session, char gender)	Retrieve the records that match the specified value with the attribute defined in the ORM Qualifier and specified session.

Methods of a typical ORM Qualifier class

Using ORM Qualifier in programming

You can use the qualifier methods to load or list data from database. The following examples show how to load or list via ORM qualifier.

Load

```
System.out.println(com.PersonFactory.loadByGender('m'));
```

By executing the code the **FIRST** occurrence of 'm' gender column in the Staff table will be loaded to a Staff object. Here is the result:

```
Person[ Name=Paul Age=12 Gender=m Dob=1993-11-07 ID=1 ]
```

List

```
com.Staff[] staffs = com.StaffFactory.listByGender('f');
for (int i = 0; i < staffs.length; i++){
   System.out.println(staffs[i]);
}
```

By executing the code, ALL rows that contain 'f' in the gender column in the Staff table will be retrieved and stored in an array of Staff object.

```
Staff[ Name=Erica Age=33 Gender=f Dob=1972-11-07 ID=2 ]
Staff[ Name=Peggy Age=45 Gender=f Dob=1960-11-07 ID=3 ]
```

Related Resources

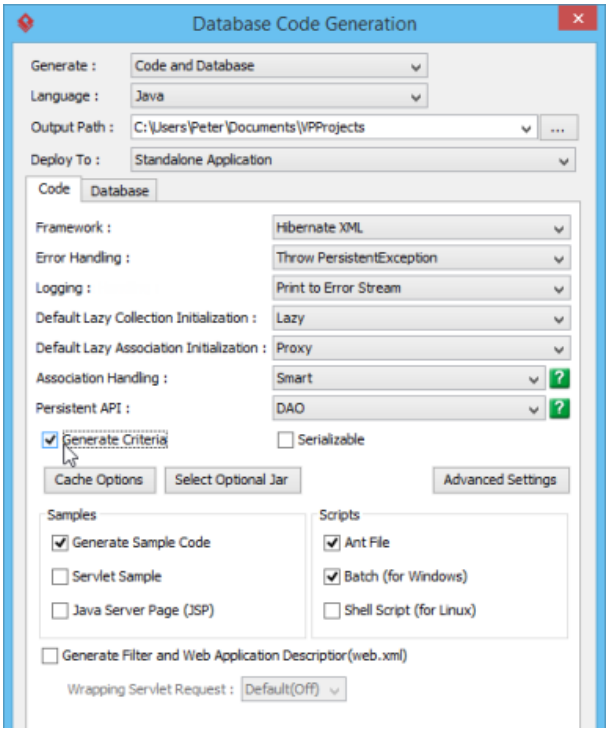
The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Using ORM Criteria

ORM Criteria is one of the handy approaches that aids data retrieval in writing a program. Instead of writing a lot of if-then-else in checking and filtering the records you need, ORM Criteria groups all the required conditions as a single criteria object, and you can retrieve data by loading the records from that criteria object.

In order to use ORM Criteria, make sure you have checked **Generate Criteria** in the **Database Code Generation** window.



Checking Generate Criteria in Database Code Generation window

Understanding the Criteria class

The following is an example of ORM Criteria class. Its name and attributes respect the corresponding ORM Persistable class in your object model.

```
public class StaffCriteria extends AbstractORMCriteria {
  public final StringExpression name;
  public final IntegerExpression age;
  public final CharacterExpression gender;
  public final DateExpression dob;
  public final IntegerExpression ID;
  public StaffCriteria(PersistentSession session) {
    super(session.createCriteria(Staff.class));
    name = new StringExpression("name", this);
    age = new IntegerExpression("age", this);
    gender = new CharacterExpression("gender", this);
    dob = new DateExpression("dob", this);
    ID = new IntegerExpression("ID", this);
  }
  public StaffCriteria() throws PersistentException {
    this(com.UntitledPersistentManager.instance().getSession());
  }
  public Staff uniqueStaff() {
    return (Staff) super.uniqueResult();
  }
  public Staff[] listStaff() {
    return (Staff[]) super.list().toArray(new Staff[super.list().size()]);
  }
}
```

Using ORM Criteria in programming

To use an ORM Criteria, you need to first specify the conditions and then retrieve data from the Criteria class. To specify conditions, you need to write the following in source code:

```
criteria.property.expression(parameter);
```

where criteria is the instance of the criteria class; property is the property of the criteria; expression is the expression to be applied on the property; parameter is the parameter(s) of the expression. The table below shows the expression that can be used for specifying the condition for query.

Expression	Description

eq(value)	The value of the property is equal to the specified value.
ne(value)	The value of the property is not equal to the specified value.
gt(value)	The value of the property is greater than to the specified value.
ge(value)	The value of the property is greater than or equal to the specified value.
lt(value)	The value of the property is less than the specified value.
le(value)	The value of the property is less than or equal to the specified value.
isEmpty()	The value of the property is empty.
isNotEmpty()	The value of the property is not empty.
isNull()	The value of the property is NULL.
isNotNull()	The value of the property is not NULL.
in(value)	The value of the property contains the specified values in the array.
between(value1, value2)	The value of the property is between the two specified values, value1 and value2.
like(value)	The value of the property matches the string pattern of value; use % in value for wildcard.
ilike(value)	The value of the property matches the string pattern of value, ignoring case differences.

Expression of ORM Criteria

Here is an example of specifying conditions with ORM criteria:

```
staffCriteria.age.ge(13);
```

There are two types of ordering to sort the retrieved records, that is, ascending and descending order. To sort the retrieved records with respect to the property, use the code template:

```
criteria.property.order(ascending_order);
```

where the value of ascending_order is either true or false. True refers to sort the property in ascending order while false refers to sort the property in descending order. For example:

```
staffCriteria.age.order(true);
```

To set the range of the number of records to be retrieved by using one of the two methods:

- setFirstResult(int i) - Retrieve the i-th record from the results as the first result.
- setMaxResult(int i) - Set the maximum number of retrieved records by specified value, i.

For example:

```
staffCriteria.setMaxResults(100);
```

The StaffCriteria class contains two methods to load the retrieved record(s) to an object or array.

- uniqueClass() - Retrieve a single record matching the specified condition(s) for the criteria; Exception will be thrown if the number of retrieved record is not equal to 1.
- listClass() - Retrieve the records matched with the specified condition(s) for the criteria.

For example:

```
com.Staff[] lcomStaffs = staffCriteria.listStaff();
```

Comparison between Criteria class and SQL query

Both SQL Query and Criteria Class can help you find records from databas. However, SQL Query is usually long and complex. It is easy to make syntax mistake when writing SQL Query and when a mistake happens, it is hard to debug, too. On the contrary, Criteria class is easy to use. It also supports getting persistent objects directly from database while SQL Query can only retrieve individual data from database.

Using Criteria	Using SQL Query

StaffCriteria staffCriteria = new StaffCriteria();
staffCriteria.name.eq("Paul");
Staff[] lcomStaffs = staffCriteria.listStaff();
int length = (lcomStaffs == null) ? 0 :
Math.min(lcomStaffs.length, 100);
for (int i = 0; i < length; i++) {
	SELECT * FROM staff WHERE name = 'Paul';

System.out.println(lcomStaffs[i]); } System.out.println(length + " Staff record(s) retrieved.");	

StaffCriteria staffCriteria = new StaffCriteria();
staffCriteria.dob.between(new GregorianCalendar(1970, 1, 1).getTime(),
new GregorianCalendar(1985, 12, 31).getTime());
Staff[] lcomStaffs = staffCriteria.listStaff();
int length = (lcomStaffs == null) ? 0 :
Math.min(lcomStaffs.length, 100);
for (int i = 0; i < length; i++) {
System.out.println(lcomStaffs[i]);
}
System.out.println(length + " Staff record(s) retrieved.");
	SELECT * FROM staff WHERE dob > '1970-01-01' AND dob < '1985- 01-01';

StaffCriteria staffCriteria = new StaffCriteria();
staffCriteria.age.in(new int[]{18, 22});
staffCriteria.gender.eq('m');
Staff[] lcomStaffs = staffCriteria.listStaff();
int length = (lcomStaffs == null)? 0 :
Math.min(lcomStaffs.length, 100);
for (int i = 0; i < length; i++) {
System.out.println(lcomStaffs[i]);
}
System.out.println(length + " Staff record(s) retrieved.");
	SELECT * FROM staff WHERE age = 18 OR age = 22 AND gender = 'm';

StaffCriteria staffCriteria = new StaffCriteria();
staffCriteria.name.like("P%");
staffCriteria.age.lt(50);
staffCriteria.name.order(true);
Staff[] lcomStaffs = staffCriteria.listStaff();
int length = (lcomStaffs == null)? 0 :
Math.min(lcomStaffs.length, 100);
for (int i = 0; i < length; i++) {
System.out.println(lcomStaffs[i]);
}
System.out.println(length + " Staff record(s) retrieved.");
	SELECT * From staff WHERE age < 50 AND name LIKE 'p%';

Comparison between Criteria class and SQL Querying

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- FREE Online Training - Database Design and Management
- Know-how - Personalize Database Connection Settings to Aid in Team Development
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Maintaining project reference

To reference another project enables you to link to another project, and use its model elements in developing your own project. In this chapter you can see how to add and maintain referenced project.

Referencing another project

Shows you how to add a referenced project.

Referencing other projects' model elements

Having added a referenced project, you can make use of referenced elements in current project.

Mirroring model element

Mirror is a technique to create a partial copy of referenced element in current project. This make it possible to create shapes in a mirroed container shape like package.

Viewing referenced diagrams

You can read the diagrams in referenced project without actually opening it. Click on the diagram in Diagram Navigator to open it.

Duplicating element from linked project

You can clone an element from referenced project through the duplicate function.

Referencing another project

To reference another project enables you to link to another project and use its model elements in developing your own project. You can organize your model elements in a more disciplined approach by having one project per library project. This also helps you to "slim up" projects through breaking down a project into smaller pieces. Moreover, you can reference to other projects and create an overview project for them.

To reference to another project:

1. Select **Project > Referenced Project** from the toolbar.
2. In the **Manage Referenced Projects** window, click **Add**.
3. In the **Open** window, choose the Visual Paradigm project file (*.vpp) to reference to and click **Open**.
4. When you return the **Manage Referenced Projects** dialog box, click **Close**.

Related Resources

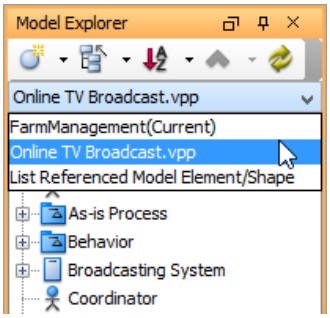
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Referencing other projects' model elements

Once a referenced project has been established, you can develop your model using model elements in referenced project. To do this:

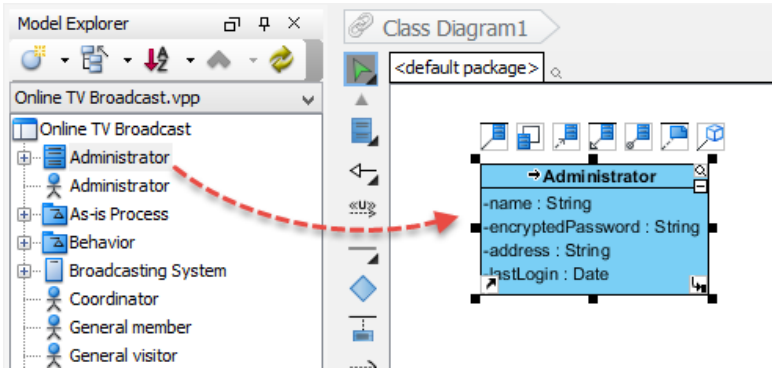
1. In **Model Explorer**, click on the drop-down menu at the top of the pane and select the project that contains the model elements you want to use. If the **Model Explorer** is hidden, open the **View** tab of the toolbar and select **Panes > Model Explorer** to show it.



Select a referenced project

> **NOTE:** By selecting the first selection **(Current)**, model elements in the current project will be listed.

2. In the model element list, drag the target model element(s) and drop it/ them on diagram. This creates views from them. In addition, you may continue modeling with the referenced elements. Note that Views for referenced project are read-only (i.e. non edit-able).

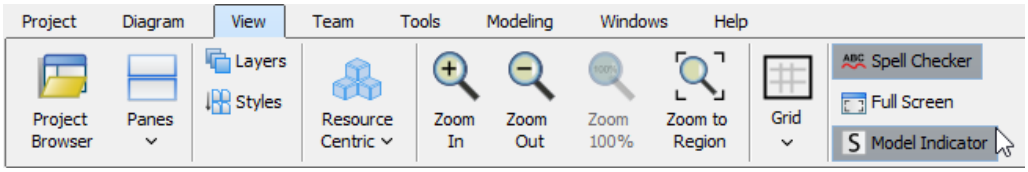


Drag model elements from referenced project

> **NOTE:** Alternatively, you can right click on the model elements and select **Create View in Active Diagram** from the pop-up menu.

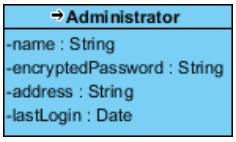
Indicating referenced elements on diagram

In order to know which shape(s) on a diagram comes from a referenced project, you can enable to Model Indicators. Click on the **Model Indicator** under the **View** tab of the toolbar.



To show Model Indicator

After that, you can see the indicator, which is a small arrow, appears at the shapes that are referencing a referenced project.



Model indicator appears

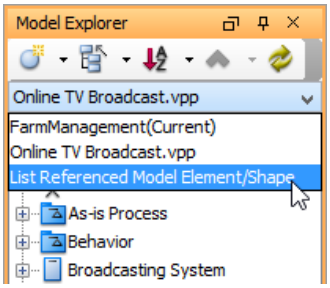
Automatic conversion to mirror for container shape-types

You may want to add shape into a referenced container shape like a referenced package or a referenced BPMN pool. Theoretically, referenced elements are not editable. In order to make it possible for the referenced shape to contain the shape you try to add, you must set it as a mirror. To set a referenced shape as mirror grants that shape the right to contain shapes. You can set a shape as mirror manually by right clicking on the shape in diagram and selecting **Convert to Mirror** from the popup menu. But if you do not do this, the shape will still be converted to mirror when you try to add shape into it. You may click here for details about mirror.

Locating referenced element

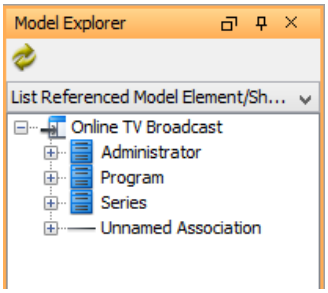
You can go to a shape that references a model element from referenced project by walking through the steps below:

1. Open the **Model Explorer**.
2. From the drop down menu at the top of the Model Explorer, select **List Referenced Model Element/Shape**.



List referenced model elements and shapes

3. A tree of model elements are listed. Those are came from the referenced project(s) and are used by the currently opening project. The first level of the tree lists the model elements in used, while the second level represent the usage. For example, **View ([DIAGRAM NAME])** means that the model element has been visualized in the diagram specified. **[MODEL ELEMENT NAME] (Documentation)** means that the model element has been added as a link in the model element specified.



Referenced model elements listed

Related Resources

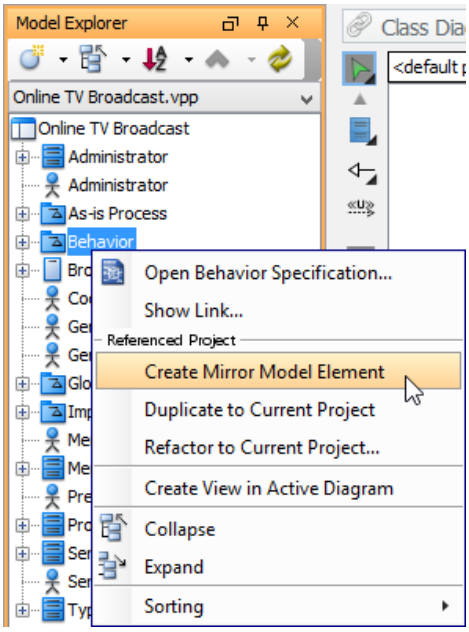
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Mirroring model element

Due to views of referenced model elements are read-only, you cannot add shapes in it. This may be a problem when you want to use referenced packages in your project and to add model elements such as classes in it. To overcome this problem, you can create a mirror of container-typed referenced model element. By mirroring, a referenced element is localized partially by keeping a mirrored copy in your project which echoes the element in referenced project. The mirrored copy can be accessed in **Model Explorer** that lists model elements in current project, but not editable.

To create a mirror from a container-typed referenced model element (e.g. package, pools/lanes), right click on the element in **Model Explorer** and select **Create Mirror Model Element** from pop-up menu. If the **Model Explorer** is hidden, open the **View** tab of the toolbar and select **Panes > Model Explorer** to show it.



Mirror a referenced package

By doing so, you can use it in your project and add shapes in it. If you have already created a view for a non-mirrored element and you want to create a mirror, you may right click on the shape and select **Convert to Mirror** from pop-up menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Viewing referenced diagrams

Sometimes, you may want to take a look at the design created in referenced project to make yourself familiar with it. To view the diagrams in referenced project, open the **Project Browser** by selecting **View > Project Browser** from the toolbar. Under the **Diagrams** page, select the desired project from drop down menu at the top left corner of the page. Thumbnails of the diagrams in that project will be listed in the page and you can double click on a thumbnail to open the diagram. Note that the diagrams are read-only.

Selecting a referenced diagram in Project Browser

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

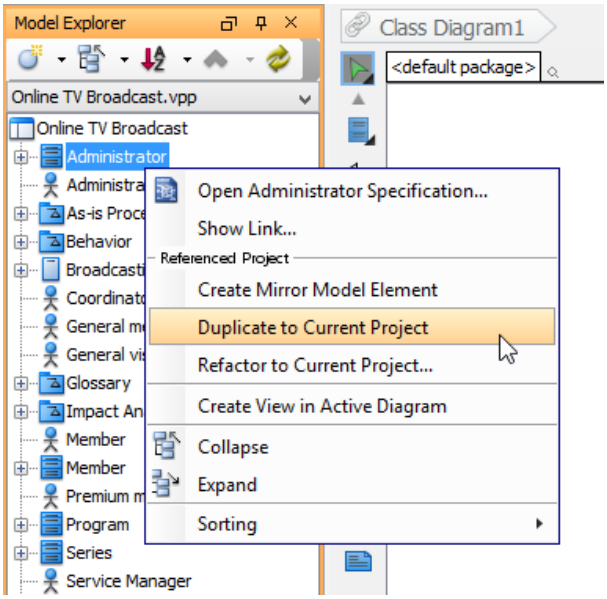
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Duplicating element from referenced project

When we have added a referenced project, we say we are referencing that project. By reference, it's like a linkage that points to the actual data stored in referenced project. Those elements borrowed from referenced project are not editable by current project.

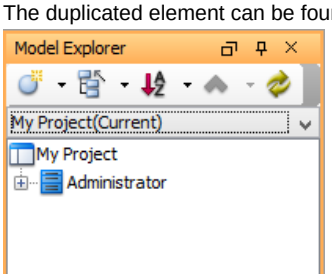
Sometimes, you need the elements created in another project,not to appear in the form of references but to make it become your own data. You will need to duplicate element from linked project. To duplicate a model element:

1. Open the **Model Explorer** by opening the **View** tab of the toolbar and selecting **Panes > Model Explorer** to show it.
2. Right click on the element you want to duplicate and select **Duplicate to Current Project** from the popup menu.



Duplicate a model element

The duplicated element can be found in **Model Explorer**, under current project.



Duplicated element appear in Model Explorer

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

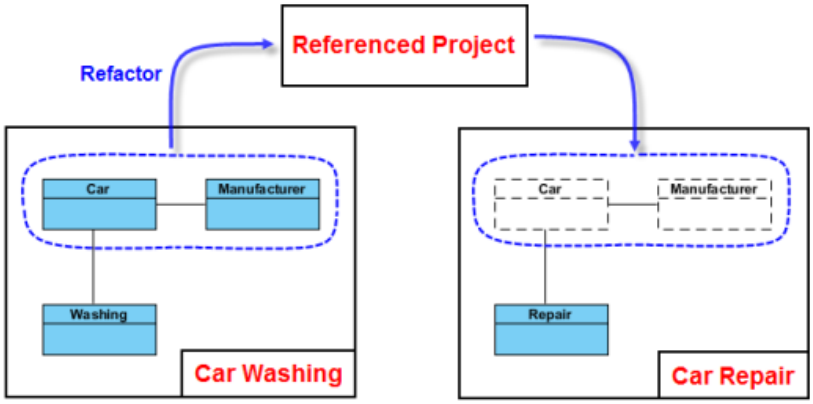
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Refactoring

Refactoring refers to the action to move project data like diagrams and/or model elements between projects that have a reference relationship in between.

Refactoring diagrams and/or model elements to another project helps in the re-structuring of project and re-organizing project data. Through refactoring, generic project data is moved to a project file, generally known as a library project, readily for other project files to access through project referencing, such that they can get the common project data included without the need of re-definition. Another benefit of refactoring is that it guarantees the correctness of model definition by enforcing common project data to be defined just once. This is also, in fact, the benefit of using project reference.

Let's say, for example, you are modeling a vehicle maintenance company. You have created two projects for the distinct parts of the business - Car washing and car repair. When modeling car washing with class diagram, you have found that classes like Car and Manufacturer are also needed by the car repair model. You then refactor them to a referenced project so that the car repair model can link with it and have the classes included.



What is refactoring?

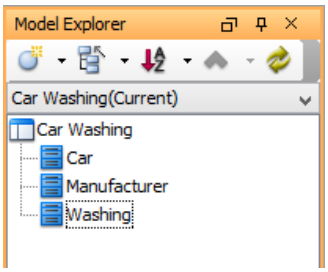
Refactoring also works in the opposite way, that is, to move diagram/model elements from the generic library project back to the 'consumer' project that use the data. While it is not so common, you may need it when you discover that certain diagram or model element in the generic project is not really that general, and you want to move the diagram/model element back to the only, specific project that use it.

How to refactor?

Keep in mind that refactoring only works with projects that have a reference in between. If you want to refactor diagram or model element to a library project but such project is either not existing or not a referenced project of your source project, get ready for the project and the source project reference to it. If you are unclear about project referencing, please read the previous chapters.

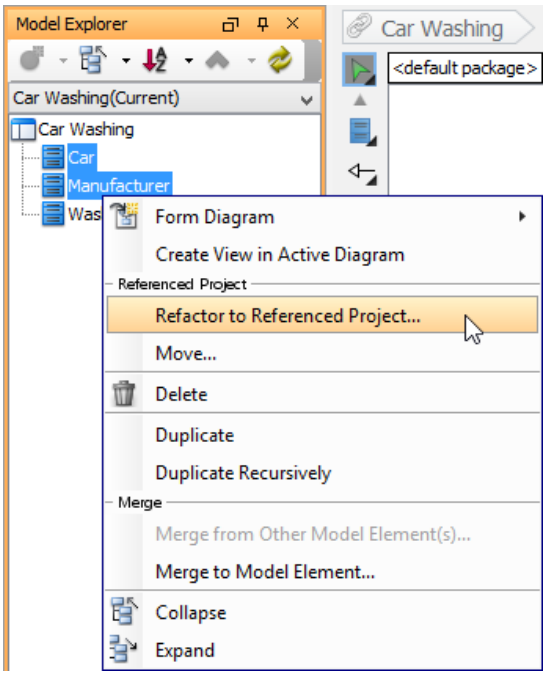
Refactor model elements to reference project

1. Open the **Model Explorer** by opening the **View** tab of the toolbar and selecting **Panes > Model Explorer** to show it.



Model Explorer

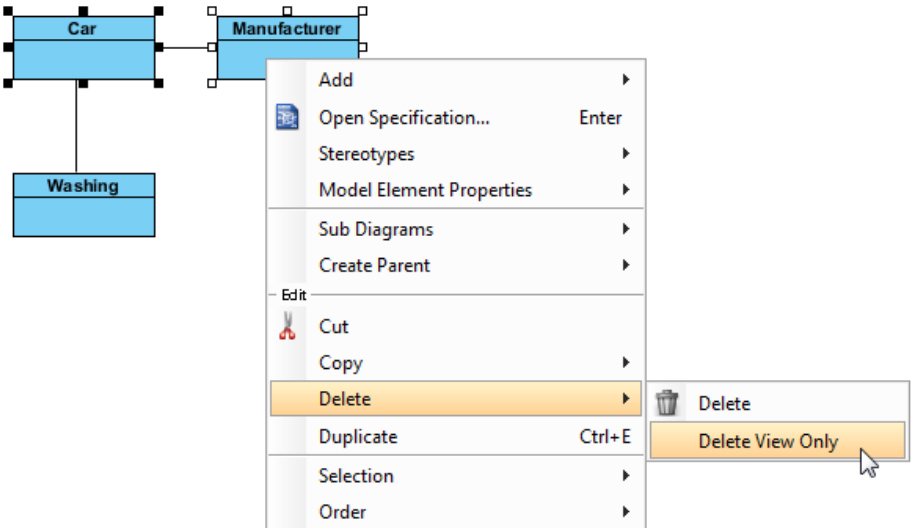
2. Select and right click on the model elements you want to refactor to referenced project. Select **Refactor to Referenced Project...** from the popup menu.



Refactor classes

3. You are prompted for saving project. Click **Yes**. If you click **No**, refactor will stop.

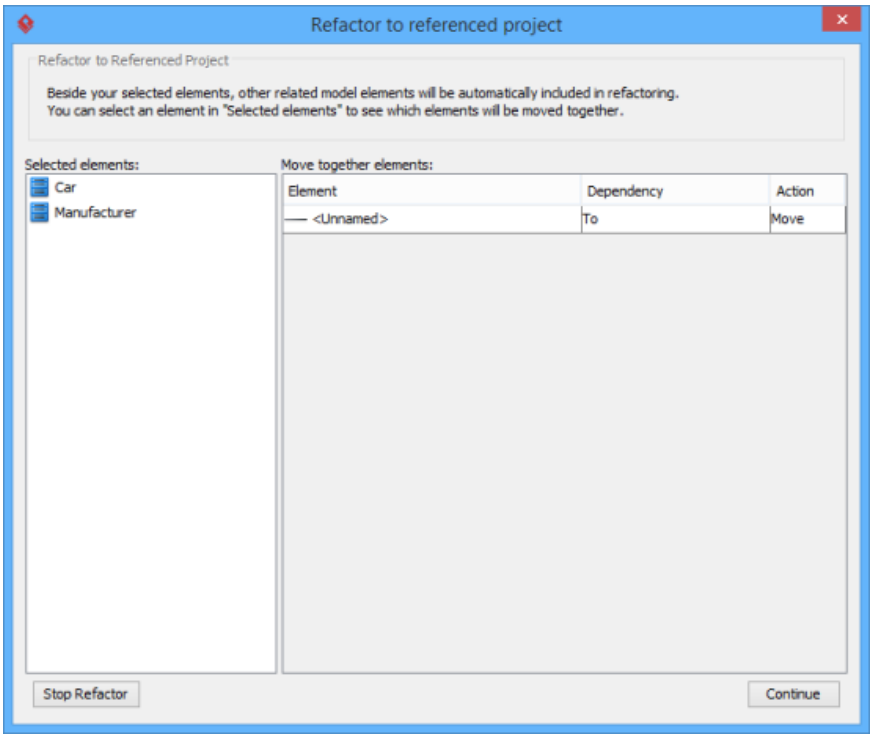
4. If the **Include Related Diagrams** window appears, this means that one or more of the selected elements has been visualized in at least one diagram (as listed on the left of window) with master view created.
Here you may take any of the following actions to continue.
   - **Include the diagrams** - Click **Include Diagram(s)** at the bottom right corner. This will move not only the model elements but also the diagrams listed on the left hand side of window to the referenced project. Since the action is not undoable, think carefully before continuing.
   - **Remove the master views** - Click **Stop Refactor** at the bottom left corner. Open the diagrams where the master views of the selected model elements exist. Delete them. When delete, make sure you are deleting the **VIEW** instead of model element. Once all master views are removed, re-run step 1.



Delete only views of element, not model elements

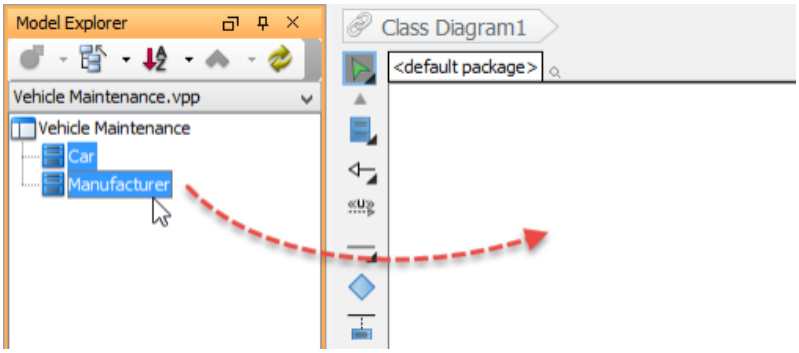
5. Refactor not only moves the selected elements to referenced project, but also those related elements. Here are two examples:
   - Connectors that connect between the selected elements.
   - Class being selected as type of attribute of selected class.

   If such related elements exist, you will be listed with those elements. Check them. If you accept moving them to referenced project, click **Continue** at the bottom right corner to continue. Otherwise, click **Stop Refactor** at the bottom left corner to terminate the refactoring.



Elements to refactor (including those related elements)

6. Once the refactoring is completed, the project will reopen itself. From now on, you can access and use the refactored elements through the **Model Explorer**. Select the referenced project under the drop down menu. Drag to diagram the element(s) to use.



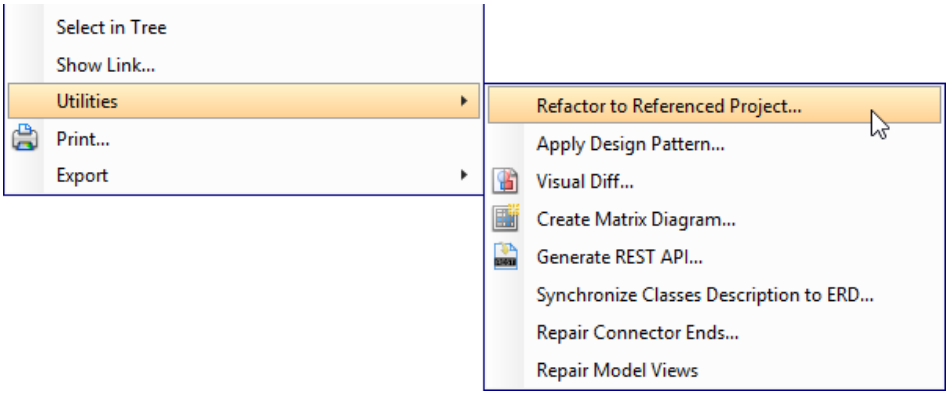
To re-use referenced classes on current project

Note that auxiliary views of refactored elements are now referencing the elements in referenced project.

Refactor diagram to reference project

To refactor diagram means to refactor the diagram as well as the elements on the diagram. As the steps are pretty close to refactoring model element, as described above, please read refactor model elements before reading this section.

1. Right click on the diagram that you want to refactor and select **Utilities > Refactor to Referenced Project...** from the popup menu.



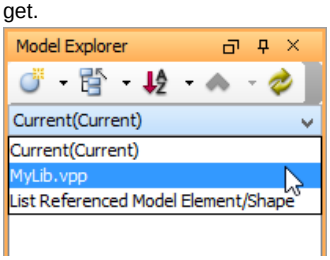
Refactor a class diagram

2. You are prompted for saving project. Click **Yes**. If you click **No**, refactor will stop.
3. If the **Include Related Diagrams** window appears, this means that one or more of the elements in the selected diagram has been visualized in another diagram (as listed on the left of window) with master view created. Read step 4 in the previous section to learn how to resolve the relationships.

4. Refactor not only moves the selected elements to referenced project but also those related elements. Read step 5 in the previous section to learn how to carry on.
5. Once the refactoring is completed, the project will reopen itself. From now on, you can access and use the refactored elements through the **Model Explorer**. Select the referenced project under the drop down menu. Drag to diagram the element(s) to use. Note that auxiliary views of refactored elements are now referencing the elements in referenced project.

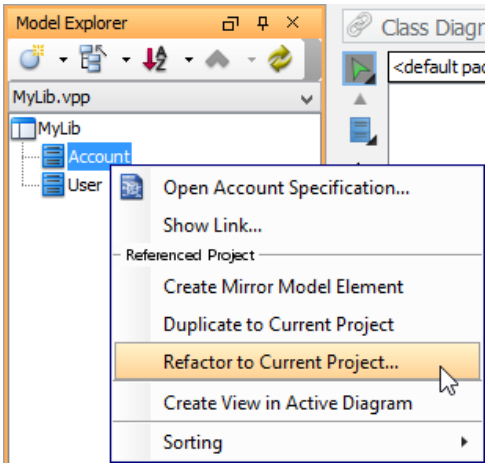
Refactor model elements from reference project

1. Open the **Model Explorer** by opening the **View** tab of the toolbar and selecting **Panes > Model Explorer** to show it.
2. Click on the drop down menu at the top of the **Model Explorer** and choose the reference project that contains the model element(s) you want to get.



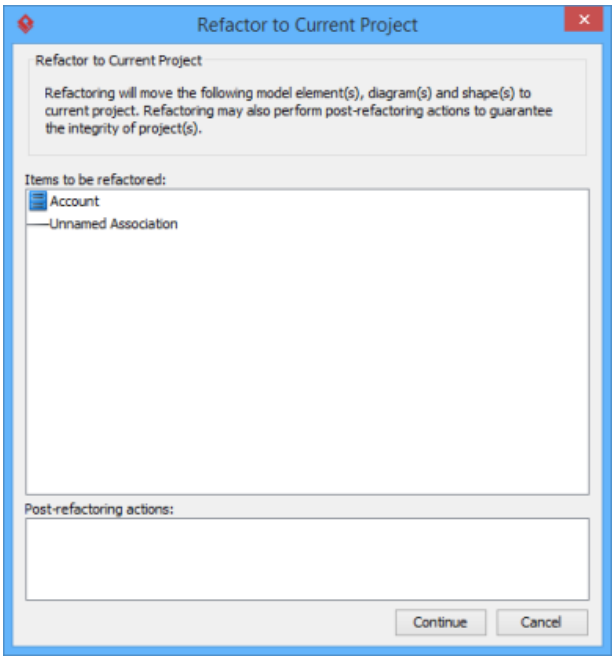
Browsing model elements in reference project

3. Select and right click on the model elements you want to get from the reference project. Select **Refactor to Current Project...** from the popup menu.



Refactoring a class

4. You are prompted to a note with some points you should pay attention to before continue. Let's revise them in brief here:
   - Refactoring cannot be undone.
   - Project will be saved before refactoring.
   - Make sure all the 'related project files' are in your workspace folder, or any sub-directory under the workspace folder. 'Related projects' means the project files that are referencing the current or reference project, either with a direct reference (project A references B) or a transitive reference (project A references A1, A1 references B). Refactoring will scan through the project files in (and only in) your workspace. The reason is that the model element(s) you attempt to refactor might be in-used by another project file. Refactoring may need to modify such project by adding/updating the project and element reference, or to stop you from continuing if a potential error may occur.
   - If you work as a team, and are using a version control system (VPository/Teamwork Server/Other repository types), make sure the 'related projects' are all checked out. Refactoring will not and cannot detect the project files in your server.
5. Click **Yes** to continue.
6. You are prompted to preview the changes. Click **Yes** if you want to preview changes. If you click **No**, refactoring will continue. If you click **Cancel**, refactoring will be cancelled.
7. If you clicked **Yes**, you see the window below, listing out the items to be refactored and the actions that will be performed after the refactoring.



Previewing changes

Refactor not only moves the selected elements to referenced project but also those related elements. Here are several typical cases:

- Connectors that connect from/to the selected element.
- Model elements that takes the selected element as type (e.g. If class 'X' has an attribute 'Xa', which has the selected element as type, class 'X' will be refactored).
- Member-typed elements (e.g. Refactoring a class will cause its attributes and operations to be refactored. Refactoring an entity will cause its columns to be refactored).

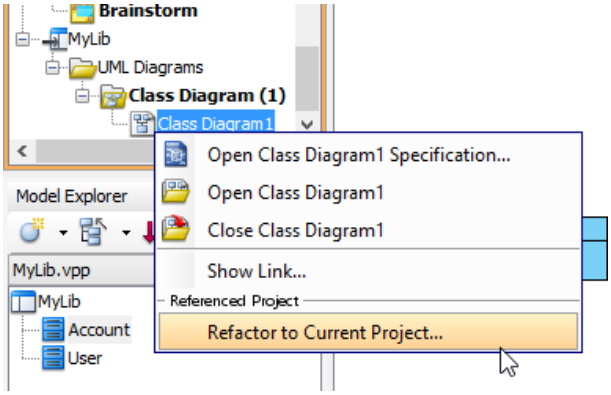
Notice that the searching of relevant project data is done recursively, meaning that whenever a model element is found relevant to any element selected to be refactored, that element will undergo the same checking performed to look for the other relevant elements, until the end. If you accept the changes, click **Continue** at the bottom right corner to continue. Otherwise, click **Cancel** at bottom right to terminate the refactoring.

8. Once the refactoring is complete, the project will reopen itself. The refactored element is now in your project. You can find them in the **Model Explorer**.

Refactor diagram from reference project

To refactor diagram means to refactor the diagram as well as the shapes on the diagram. As the steps are pretty close to refactoring model element, as described above, please read refactor model elements before reading this section.

1. In **Diagram Navigator** or **Model Explorer**, locate the diagram you want to get from reference project. If either the **Diagram Navigator** or the **Model Explorer** is hidden, show it by opening the **View** tab of the toolbar and select **Panes > Diagram Navigator** or **Panes > Model Explorer** to show it.
2. Right click on the diagram node and select **Refactor to Current Project...** from the popup menu.



Refactor class diagram

3. Click **Yes** when you are prompted to the note.
4. You are prompted to preview the changes. Click **Yes** if you want to preview changes. If you click **No**, refactoring will continue. If you click **Cancel**, refactoring will be cancelled.
5. Refactor not only moves the selected elements to referenced project but also those related elements. If you clicked **Yes**, read step 6 in the previous section to learn how to carry on.
6. Once the refactoring is completed, the project will reopen itself. The refactored diagram is now in your project. Notice that the shapes in diagram should be referencing the model elements in reference project.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model element nicknaming

Nickname enables you to creat multiple name and documentation set for your model. You can keep multiple language sets of your model by using the translation feature. In this chapter, both the nickname and translation feature will be covered.

What is nickname?

Introduce element nickname support.

Configure nickname

Shows you how to create a nickname or a language.

Using nickname

Shows you how to work with a new nickname and how to switch between nicknames.

Translation

Shows you how to make use of the translation feature to maintain model with multiple language.

Export and import word document of nickname

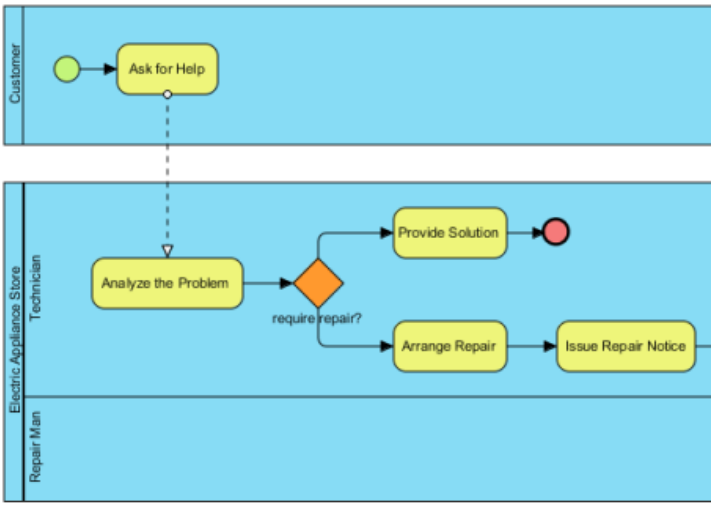
Export a Word document of nickname and nick-documentation, give it to a translator to perform translation, and import the work back to the tool.

What is nickname?

We all have a name and we may have multiple names such as nicknames or names in other languages. This is the same for your Visual Paradigm project content. While we have applied certain language in naming and describing model elements, we may have the need to model with another language to satisfy the readers of model. The nickname feature is designed to let you define multiple language sets for a model. Further to the definition of nickname, you also can make use of the translate function to translate your work into another language.
One model element can have one Original name and multiple nicknames and the same for description. With nickname, you can define and view different names without affecting the original name of model elements. You can disable the effect of nickname anytime by switching to Original nickname. Features that related to code generation will always use Original name, i.e. changing Class's name in other nicknames will not affect the generated code.

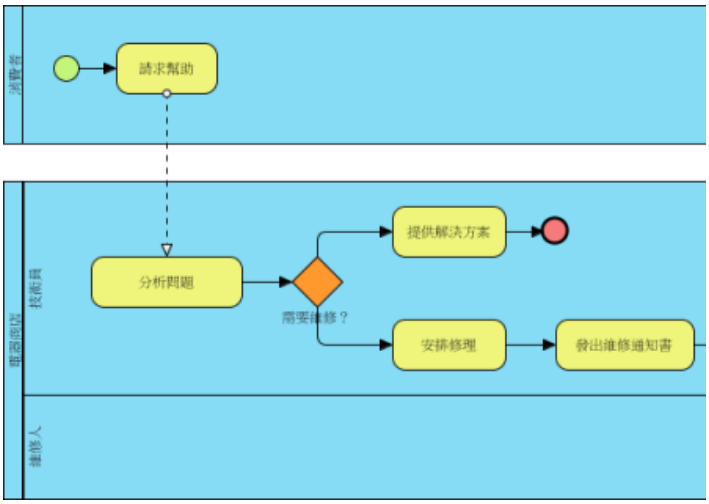
Multi-national team

If you are working in a team and your members are using different languages, you can define model elements name and description in multiple languages. Each member can choose their own language for modeling or view diagrams. The following example demonstrates the Business Process Diagram in English and Traditional Chinese respectively:



Original version of a business process

You can create a *Traditional Chinese* nickname and rename the model elements:

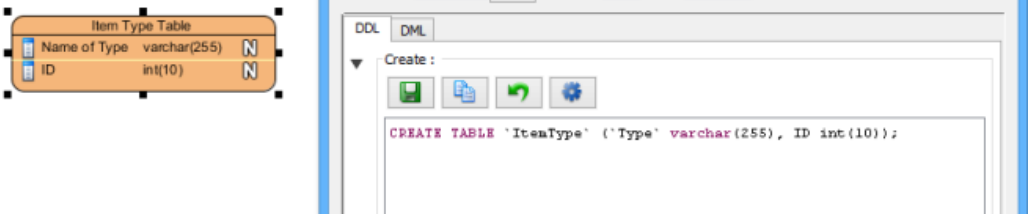


Traditional Chinese version of a business process

Now, you can switch between English (Original) and Traditional Chinese anytime, or even create more nicknames.

Increasing readability of entity relationship diagram

The name of **Entity** will be used to generate SQL but Database Management System (DBMS) has many constraints on the name of Entity, Column, etc and each DBMS are different. These constraints include the length of the name, reserved keywords, special characters, etc. They restricted the database designer to create an **Entity Relationship Diagram** (ERD) with meaningful names. With nickname, you can freely change any names to create a high readability ERD without affecting the generated SQL. The following diagram displays **ERD** in nickname but generate SQL in original name:



Generate SQL with original name

Related Resources

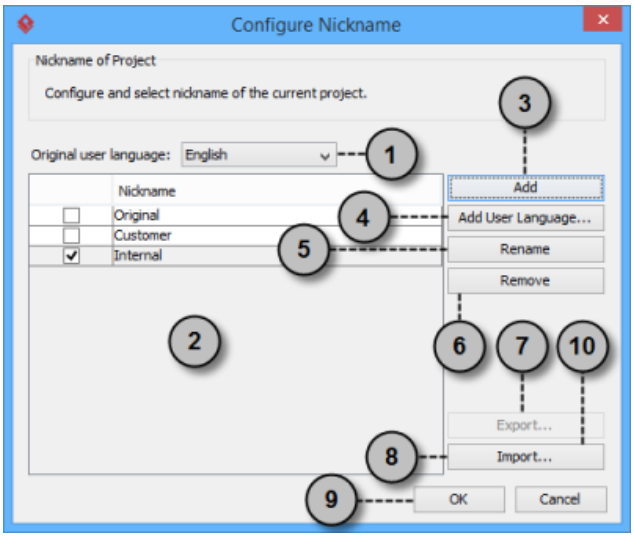
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Configure nickname

You can add a nickname under **Modeling** in the toolbar. By adding a nickname, you can start editing the names and descriptions of model elements under the new nickname.

Overview of Configure Nickname window



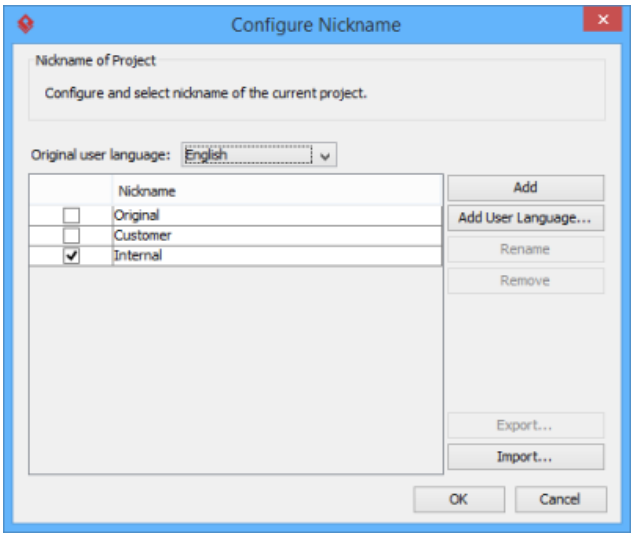
An overview of Configure Nickname window

No.	Name	Description
1	Original user language	The language (e.g. English) of the Original nickname The selection only affects the outcome of translation.
2	Nickname list	List all the nicknames.
3	Add	Click to add a nickname with a name.
4	Add user language	Click to add a nickname with selected language.
5	Rename	Click to rename a chosen nickname.
6	Remove	Click to delete a chosen nickname.
7	Export	Click to export an XML file that contains information about the original name and nickname of the name and description of model elements that are named differently in nickname.
8	Import	Click to import the XML exported from **Configure Nickname** window.
9	OK	Click to apply the nickname configuration and close this window.
10	Cancel	Click to close the window without applying the changes.

Description of Overview window

Adding nickname

1. Select **Modeling > Nickname > Configure Nicknames...** from the toolbar.
2. The current working copy is by default in **Original** nickname, with English as user language. Click **Add** in the **Configure Nickname** window.
3. In the **Input** dialog box, enter the name of the nickname set and click **OK** to confirm. Click **OK** to close the **Configure Nickname** window.



Two nicknames are added

Renaming nickname

1. Select **Modeling > Nickname > Configure Nicknames...** from the toolbar.
2. In the **Configure Nickname** window, select the nickname to rename. Click **Rename**.

> **NOTE:** You are not allowed to rename the original nickname.

3. In the **Input** dialog box, enter the name of the nickname set and click **OK** to confirm. Click **OK** to close the **Configure Nickname** window.

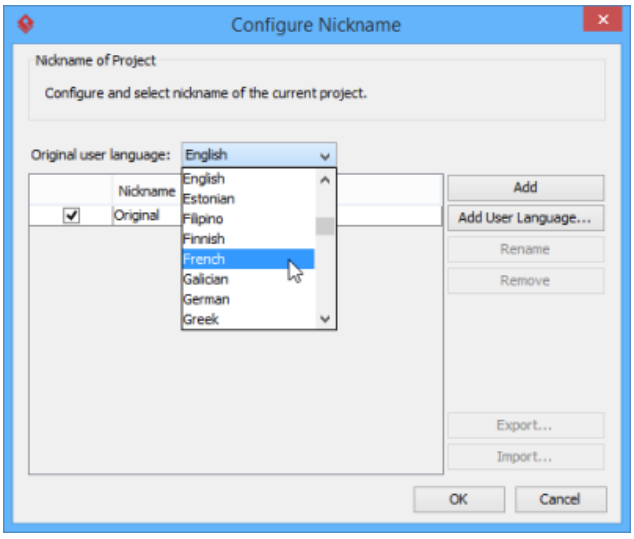
Removing nickname

1. Select **Modeling > Nickname > Configure Nicknames...** from the toolbar.
2. In the **Configure Nickname** window, select the nickname to remove. Click **Remove**. Click **Yes** when you are asked for confirmation

> **NOTE:** You are not allowed to remove the original nickname.

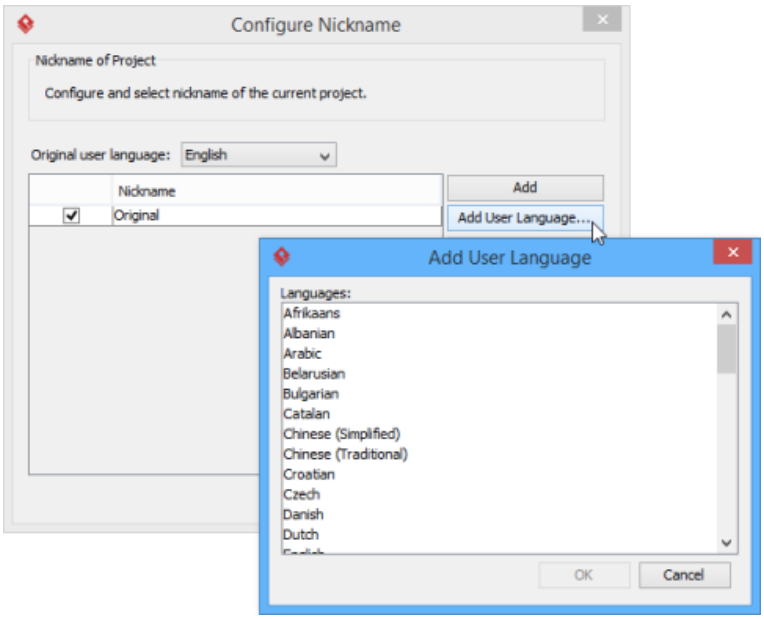
Specifying user language for the nickname

There must be an original nickname in every project, namely *Original* which represent the most standard version of language of project. You can specify the language for the original nickname, such as German. The language you have set affects the outcome of translation.



Select original user language

To add a nickname in specific language, click on the **Add User Language...** button and select a language.



Add a user language

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

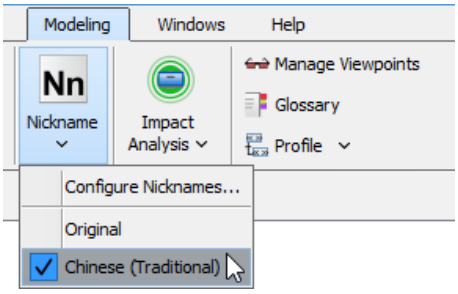
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using nickname

Once you have defined a nickname, you can start updating your model by entering the new names and description of model elements.

Start updating elements' nickname

1. Select **Modeling > Nickname** and then select a nickname to work with from the toolbar.



Select a nickname to work with

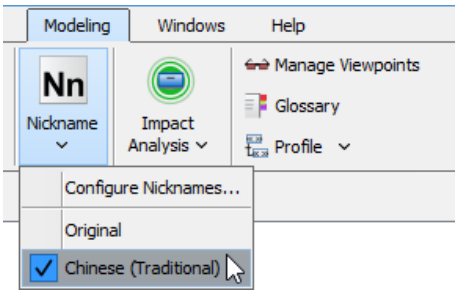
2. Start renaming model elements and updating their description. The changes you make will only be applied to the selected nickname.

Update the name and description of elements under a nickname

Switching between nicknames

The names and description of model elements are language specific. This means that, the change you make applies only to a specific nickname. Once you have switched to another nickname, the names and description of model elements will be updated to show the definition under the new nickname.

To switch between nicknames, select **Modeling > Nickname** and then select a nickname to switch to from the toolbar.



Switch nickname

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import word document of nickname

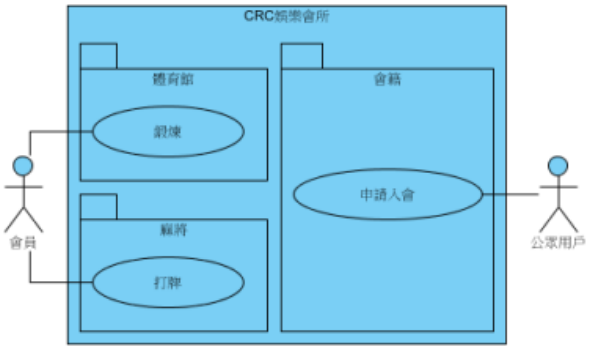
Exporting and importing the nickname of your diagrams to word document enables you to manage translation with ease. After all, you can update the nickname of exported Word document and import it back to synchronize the changes to the nickname of model elements, especially when you ask your team member (or your translator) to translate the project without providing the project to him/her.

To export your project to word document:

1. Select **Modeling > Nickname > Configure Nicknames...** from the toolbar.
2. In **Configure Nickname** window, select the desired nickname
3. Click **Export...**.
4. In **Save** dialog box, select a directory for saving it as word document in your computer. Click **Save** after you've named the file.

Now, you send the word file to your team member (or your translator). To import the translated word document back to Visual Paradigm:

1. Select **Modeling > Nickname > Configure Nicknames...** from the toolbar.
2. In **Configure Nickname** dialog box, click **Import...**.
3. In **Open** dialog box, select the directory where you save your translated word document and click **Open** button.
   As a result, you can see your project is updated.



The name of your project's model elements has been updated to show the selected nickname

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Visual diff

You can use visual diff for comparing two diagrams, and to show their differences. The comparison can be made between diagrams in the same or different. In this chapter, you will see how to compare as-is and to-be BPD, logical and physical ERD with visual diff.

What is visual diff?

You will learn what is visual diff, and have a look at its configuration options.

Comparing as-is and to-be business process diagram

An example of comparing as-is and to-be BPD will be presented to help you understand how to use visual diff in practice.

Comparing logical and physical ERD

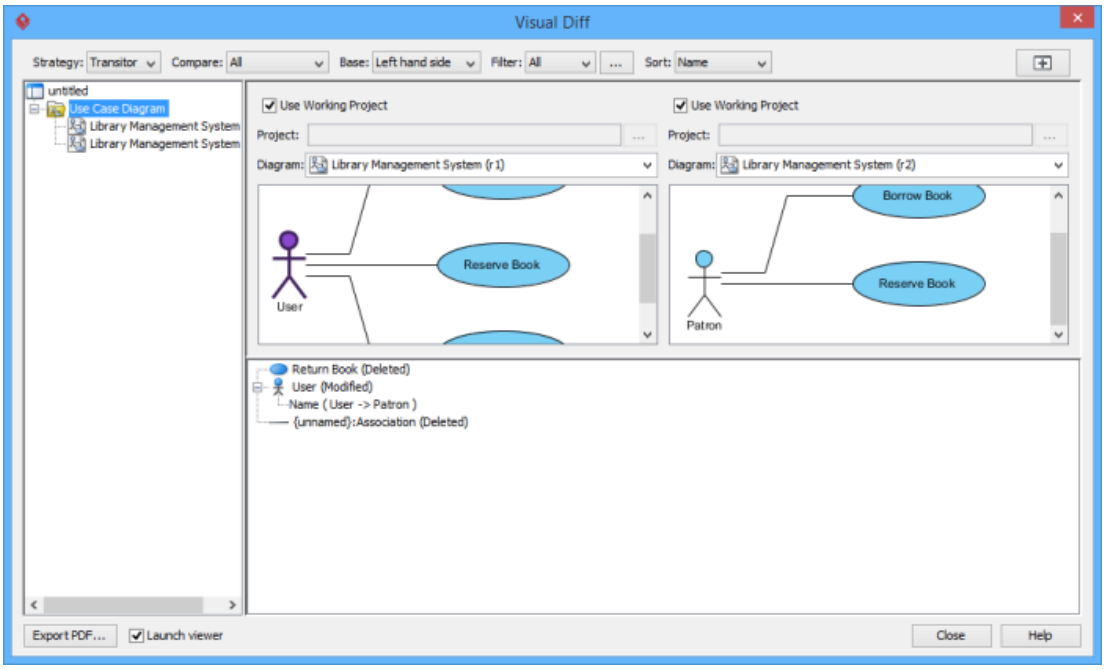
An example of comparing logical and physical ERD will be presented to help you understand how to use visual diff in practice.

What is Visual Diff?

The situation of comparing two diagrams is common. For example, compare an ERD of conceptual model with an ERD of physical model and comparing a domain class diagram with a class diagram ready for implementation. Visual Paradigm allows you to compare the differences between diagrams in order to trace the changes between them.

Diagram comparison

With the feature of **Visual Diff**, two diagrams can be compared to recognize their differences. Changes, such as modification of properties (e.g. name) and addition/removal of containing models can thus be found easily.



The overview of Visual Diff window

Various comparison strategies

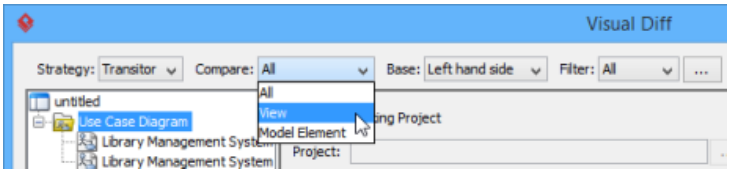
A comparison strategy determines how two diagrams will be compared. Each strategy is used for its own specific purpose. You can select the appropriate strategy that suits your case most. The following is the description of strategies:

Strategy	Description
ID	Shapes will be matched base on their internal ID. Differences between shapes that have same ID will be displayed in the result pane. This strategy is usually used to visualize the changes of same shapes in two projects.
Name	Shapes will be matched base on their names. This strategy is useful when visualizing differences for external works. One of typical examples is to compare databases and class models.
Transitor	Shapes will be matched base on their transition established by Model Transitor. This way of comparison is usually used to visualize the differences between model elements.

Description of comparison strategies

Comparing view only, model element only, or both

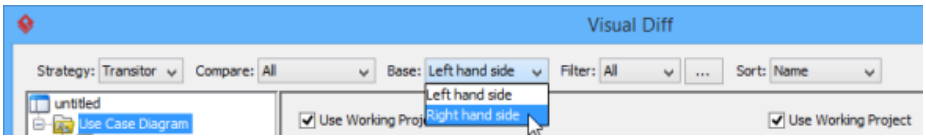
Comparison is divided into view, model element and both of them. The selected option from the drop-down menu of **Compare** determines the display for comparison. By selecting **View**, the differences in view settings, such as the coordination of shapes will be considered as changes. By selecting **Model Element**, the differences in model element level, such as their names will be considered as changes. By selecting **All**, the differences in both view settings and model element level are displayed.



Change to compare view

Comparing from the left to the right and vice versa

Comparisons are made between two diagrams, which are put on the left and the right hand side in the **Visual Diff** window respectively. By default, comparison is based on left hand side, which means that if a shape does not exist on the left hand side but on the right hand side, the shape will be considered as newly added in the result pane. The base can be swapped from the right to the left and vice versa. By doing so, the absence of shape on the left hand side will result in a document of deleted shape.



Changing the base to "Right hand side"

Sorting the data on result pane

The selected option from the drop-down menu of **Sort** determines the order of data on the result pane. You can select sort by type, by name or by change type.

Exporting a document of differences

A document of the current comparing diagrams can be saved in your computer as PDF. Click **Export PDF...** button at the bottom left corner of the window. Select a directory for storing in **Export PDF** dialog box and then click **Save** button.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

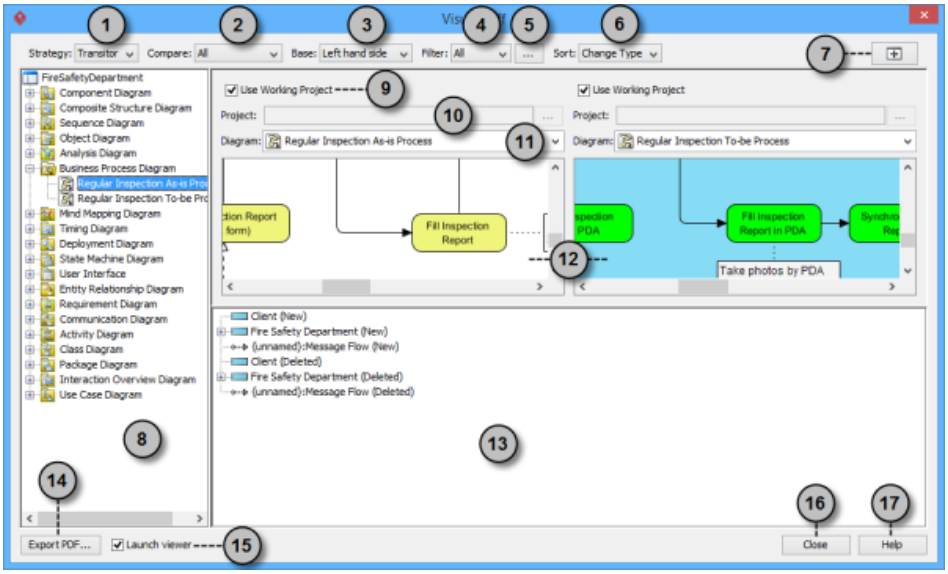
Comparing as-is and to-be business process diagram

In this page, two Business Process Diagrams are compared: one for modeling the As-is Process and another for modeling the To-be Process. The features of **Visual Diff** are applied in order to find the differences between them.

1. In *As-Is Process* diagram, select **Modeling > Visual Diff** from the toolbar.

> **NOTE:** Alternatively, you may right click on diagram background and select **Utilities > Visual Diff...** from the pop-up menu:

The overview of **Visual Diff** window:



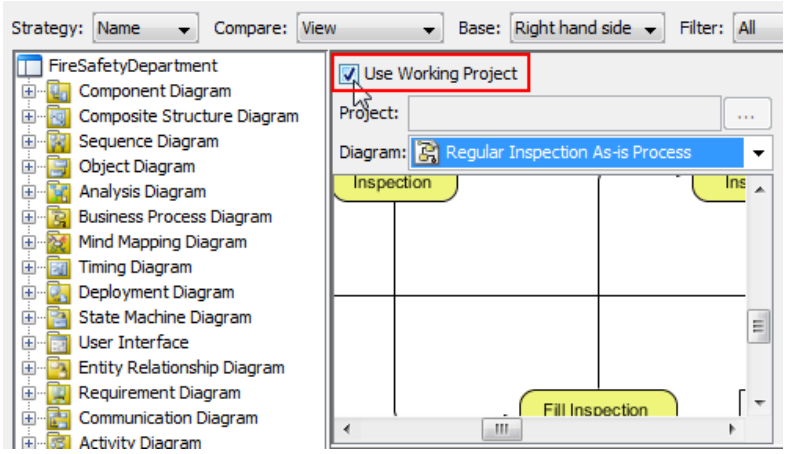
*The **Visual Diff** window*

No.	Name	Description
1	Strategy	It determines how two diagrams will be compared. Select the appropriate strategy that suits your application most. **ID** : Shapes will be matched base on their internal ID. Differences between shapes that have same ID will be displayed in the result pane. This strategy is usually used to visualize the changes of same shapes in two projects. **Name** : Shapes will be matched base on their names. This strategy is useful when visualizing differences for external works. One of typical examples is to compare databases and class models. **Transitor** : Shapes will be matched base on their transition established by Model Transitor. This way of comparison is usually used to visualize the differences between model elements.
2	Compare	It determines how two diagrams will be displayed for comparison. **All**: Both view and model elements are displayed. **View** : Differences, such as coordinates, width, height and color of shapes, are displayed. **Model Element** : Differences, such as model name, are displayed.
3	Base	It determines which diagram is used as a base for comparison. **Left hand side**: Comparison is based on the diagram on the right hand side. For example, if there is a shape absent on the left hand side, but appear on the right hand side, the shape is said to be a new shape. **Right hand side** : Comparison is based on the diagram on the right hand side. For example, if there is a shape absent on the left hand side, but appear on the right hand side, the shape is said to be a removed shape.
4	Filter	It determines what kind of comparing data will be displayed on the result pane. **All**: Display all kinds of differences including the addition, modification and removal of shapes. **New**: Display only results about the addition of shape and then hide the rest. **Modified**: Display only results about the modification of shapes and then hide the rest. **Deleted**: Display only results about the removal of shapes and then hide the rest.
5	Filter Model Type...	It determines what type of model elements will be shown on the result pane. When **Filter Model Type** window pops out, uncheck the type of model elements you don't want to be shown; otherwise, check the type of model elements you want it to be shown.
6	Sort	It determines how the result of comparison be displayed on Result pane. You can select sort by Type, by Name or by Change Type. **Name**: All model elements are sorted by their name. **Change Type**: All model elements are changed their type. **Type**: All model elements are sorted by their type.
7	Maximum	Press this button to enlarge the **Visual Diff** window to the maximum screen size. Press it again to reduce the window to the default size.

8	Diagram list	It lists all diagrams on projects selected in the left hand side and the right hand side respectively.
9	Use Working Project	Check it if you want to select a diagram for comparison within the current working project.
10	Project	Select and open a project directory for comparison.
11	Diagram	Select and open a specific diagram for comparison after selecting a project.
12	Display pane	A pane that has two sides. Each side displays a diagram out of a project.
13	Result pane	The differences of the two projects are shown here. When the word *New* is shown behind a task that means it is newly added, the word *Deleted* is shown behind a task that means it is deleted.
14	Export PDF...	Click it to save a document of the two compared diagrams in your computer as PDF.
15	Launch viewer	Open the exported file after you have export a PDF file. Check it to open the PDF file. If you uncheck it, nothing will happen even after you have exported a PDF file.
16	Compare	Click it to start comparing two diagrams.
17	Close	Close **Visual Diff** window.

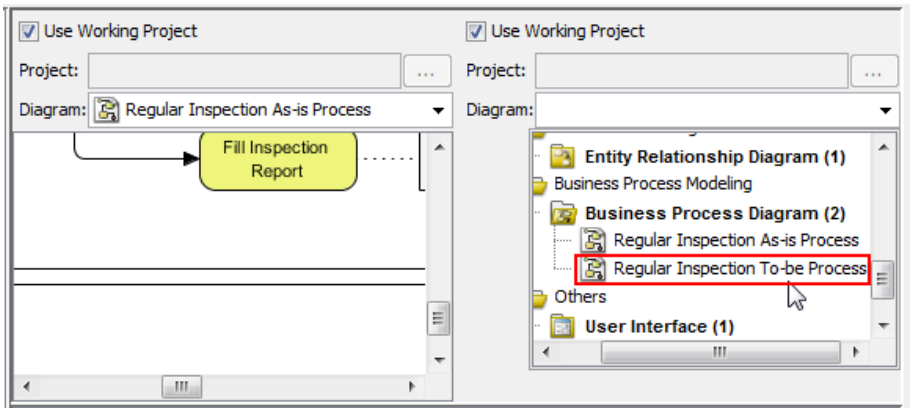
*The description of features in **Visual Diff** window*

2. In **Visual Diff** window, the left hand side window shows the currently opened diagram by default. You may remain it unchanged; otherwise uncheck **Use Working Project** on the left hand side if you want to select a diagram in other projects to compare with. Click **...** button in **Project** to select the directory of other projects. Similarly, check/ uncheck **Use Working Project** on the right hand side window.



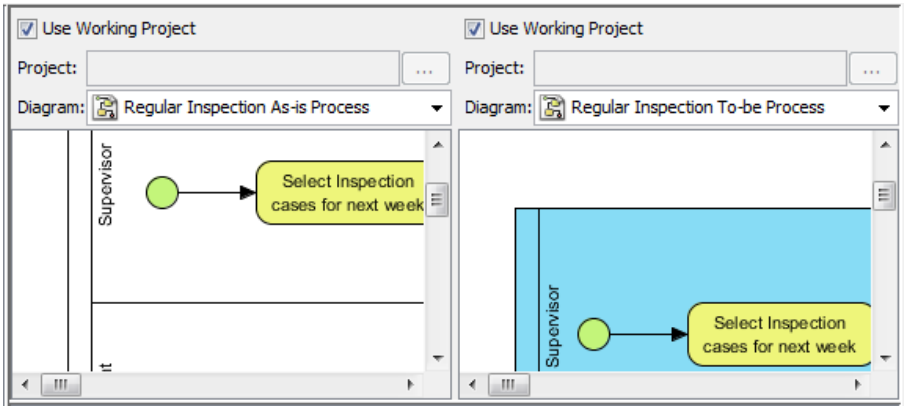
*Check **Use Working Project***

3. Select the *To-be Process* diagram to compare with from the drop-down menu of **Diagram**.



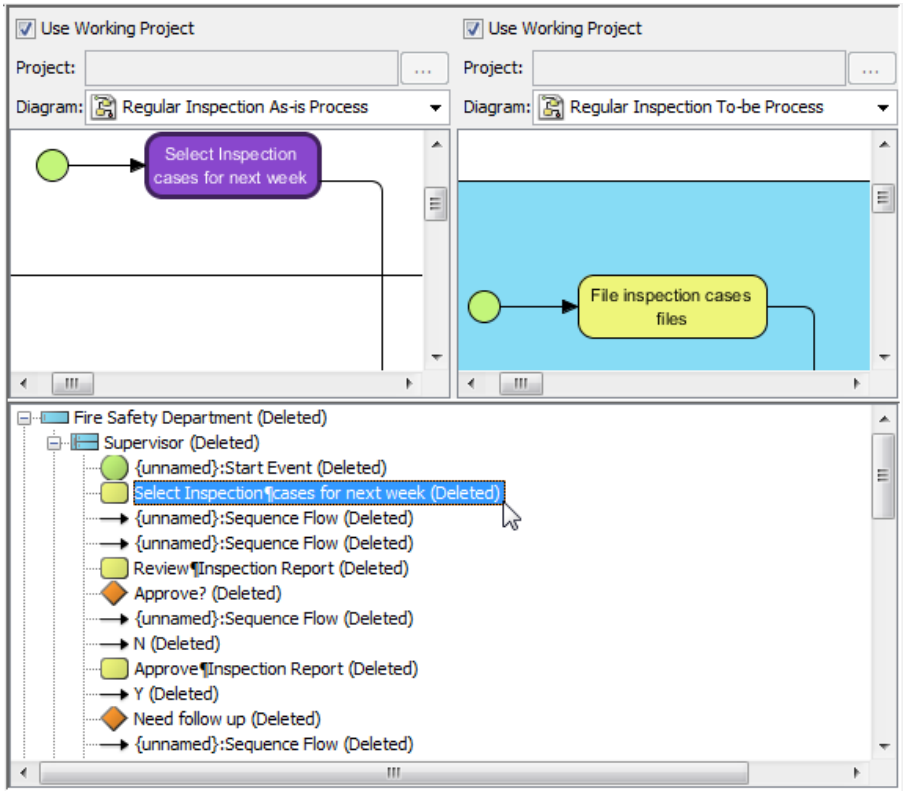
Select a diagram for comparison

The two diagrams are shown side by side on display pane. However, the ways of comparison has not yet been configured. Let's configure them one by one in the following steps.



Two diagrams are selected

4. At the top of the Visual Diff window, adjust the **Strategy**, **Compare**, **Base** and **Filter**.
5. Once everything is set, the differences of the two diagrams will be shown on the result pane.
6. If you want to view a specific shape, you may click its node on the result pane and then, the selected shape will be painted in dark purple on the display pane.



Select a node on the result pane

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Comparing logical and physical ERD

Entity relationship diagram (ERD) represents a detailed picture of the entities needed for a business. In forward engineering, ERD will be transformed into a relational database eventually. There are at least two types of ERD – Logical and Physical. They are used in different stages of development and are inter-related.

Logical ERD models information gathered from business requirements. Entities and relationships modeled in such ERD are defined around the business's need. The need of satisfying the database design is not considered yet.

Physical ERD represents the actual design of database. It deals with conversion from logical design into a schema level design that will be transformed into relational database. When modeling a physical ERD, Logical ERD is treated as base, refinement occurs by defining primary keys, foreign keys and constraints. Sometimes, relationships need to be resolved by introducing additional tables, like a Linked table for a many to many relationship.

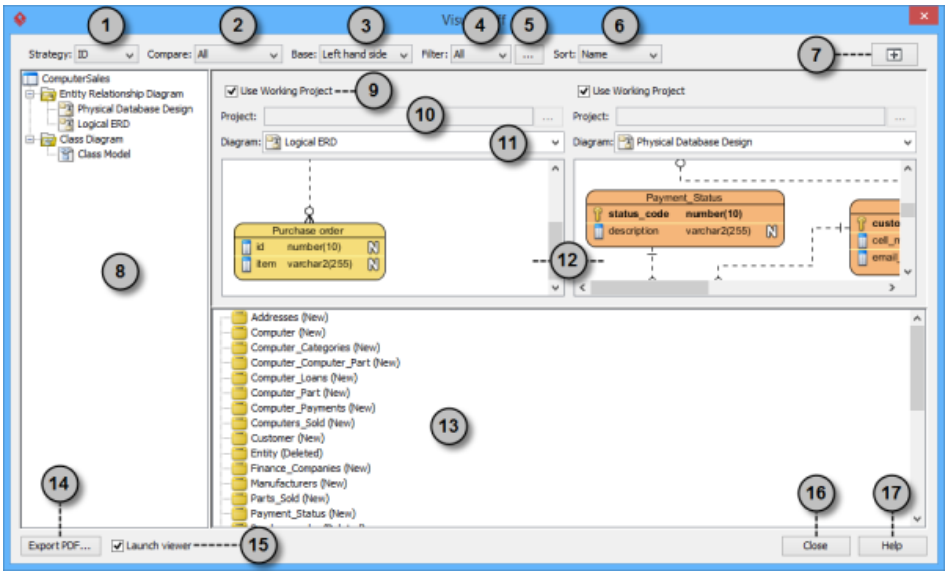
Since physical ERD and logical ERD represent the business requirement and database schema respectively, comparing physical and logical ERD helps to find out the differences between them in order to confirm the database is exactly following the initial business requirements regardless of the changes.

Two ERDs are compared: one for modeling the Logical Model and another one for modeling the Physical Model. With the features of **Visual Diff**, the differences between Logical and Physical ERD can be found easily.

1. In *Logical ERD* diagram, select **Modeling > Visual Diff** from the toolbar.

> **NOTE:** Alternatively, you can right click on diagram background and select **Utilities > Visual Diff...** from the pop-up menu.

The overview of **Visual Diff** window:



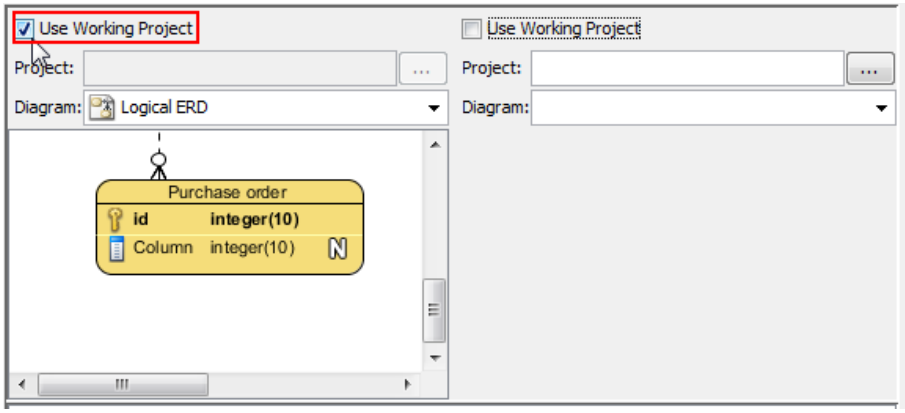
*The **Visual Diff** window*

No.	Name	Description
1	Strategy	It determines how two diagrams will be compared. Select the appropriate strategy that suits your application most. **ID**: Shapes will be matched base on their internal ID. Differences between shapes that have same ID will be displayed in the result pane. This strategy is usually used to visualize the changes of same shapes in two projects. **Name**: Shapes will be matched base on their names. This strategy is useful when visualizing differences for external works. One of typical examples is to compare databases and class models. **Transitor**: Shapes will be matched base on their transition established by Model Transitor. This way of comparison is usually used to visualize the differences between model elements.
2	Compare	It determines how two diagrams will be displayed for comparison. **All**: Both view and model elements are displayed. **View**: Differences, such as coordinates, width, height and color of shapes, are displayed. **Model Element**: Differences, such as model name, are displayed.
3	Base	It determines which diagram is used as a base for comparison. **Left hand side**: Comparison is based on the diagram on the right hand side. For example, if there is a shape absent on the left hand side, but appear on the right hand side, the shape is said to be a new shape. **Right hand side**: Comparison is based on the diagram on the right hand side. For example, if there is a shape absent on the left hand side, but appear on the right hand side, the shape is said to be a removed shape.
4	Filter	It determines what kind of comparing data will be displayed on the result pane. **All**: Display all kinds of differences including the addition, modification and removal of shapes. **New**: Display only results about the addition of shape and then hide the rest. **Modified**: Display only results about the modification of shapes and then hide the rest. **Deleted**: Display only results about the removal of shapes and then hide the rest.

5	Filter Model Type...	It determines what type of model elements will be shown on the result pane. When **Filter Model Type** window pops out, uncheck the type of model elements you don't want to be shown; otherwise, check the type of model elements you want it to be shown.
6	Sort	It determines how the result of comparison be displayed on Result pane. You can select sort by Type, by Name or by Change Type. **Name**: All model elements are sorted by their name. **Change Type**: All model elements are changed their type. **Type**: All model elements are sorted by their type.
7	Maximum	Press this button to enlarge the Visual Diff window to the maximum screen size. Press it again to reduce the window to the default size.
8	Diagram list	It lists all diagrams on projects selected in the left hand side and the right hand side respectively.
9	Use Working Project	Check it if you want to select a diagram for comparison within the current working project.
10	Project	Select and open a project directory for comparison.
11	Diagram	Select and open a specific diagram for comparison after selecting a project.
12	Display pane	A pane that has two sides. Each side displays a diagram out of a project.
13	Result pane	The differences of the two projects are shown here. When the word *New* is shown behind a task that means it is newly added, the word *Deleted* is shown behind a task that means it is deleted.
14	Export PDF...	Click it to save a document of the two compared diagrams in your computer as PDF.
15	Launch viewer	Open the exported file after you have export a PDF file. Check it to open the PDF file. If you uncheck it, nothing will happen even after you have exported a PDF file.
16	Compare	Click it to start comparing two diagrams.
17	Close	Close **Visual Diff** window.

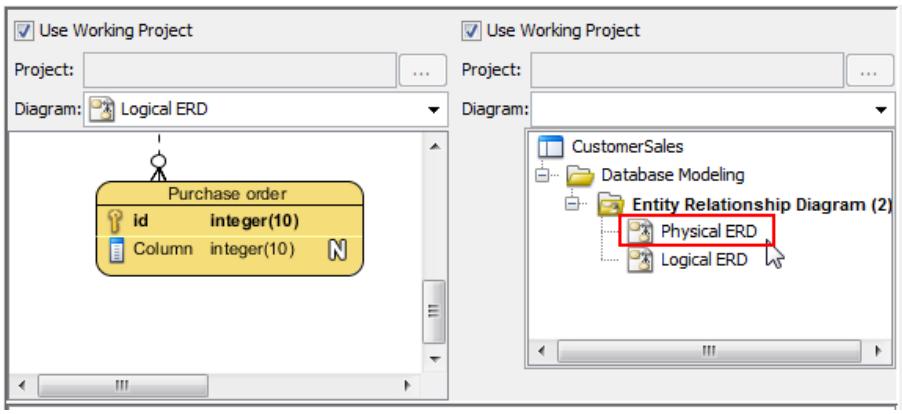
Description of various parts of Visual Diff window

2. In **Visual Diff** window, the left hand side window shows the currently opened diagram by default. You may remain it unchanged; otherwise uncheck **Use Working Project** on the left hand side if you want to select a diagram in other projects to compare with. Click **...** button in Project to select the directory of other projects. Similarly, check/ uncheck **Use Working Project** on the right hand side window.



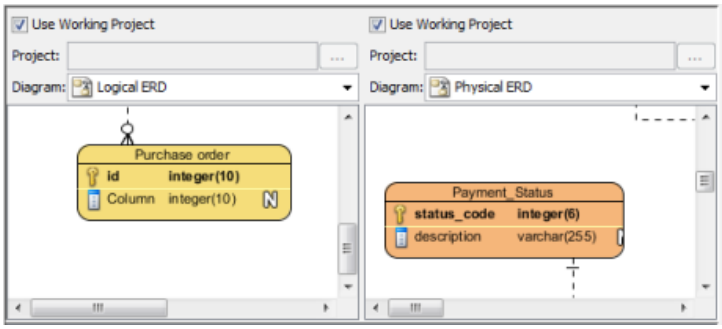
*Select **Use Working Project***

3. Select the *Physical ERD* to compare with from the drop-down menu of **Diagram**.



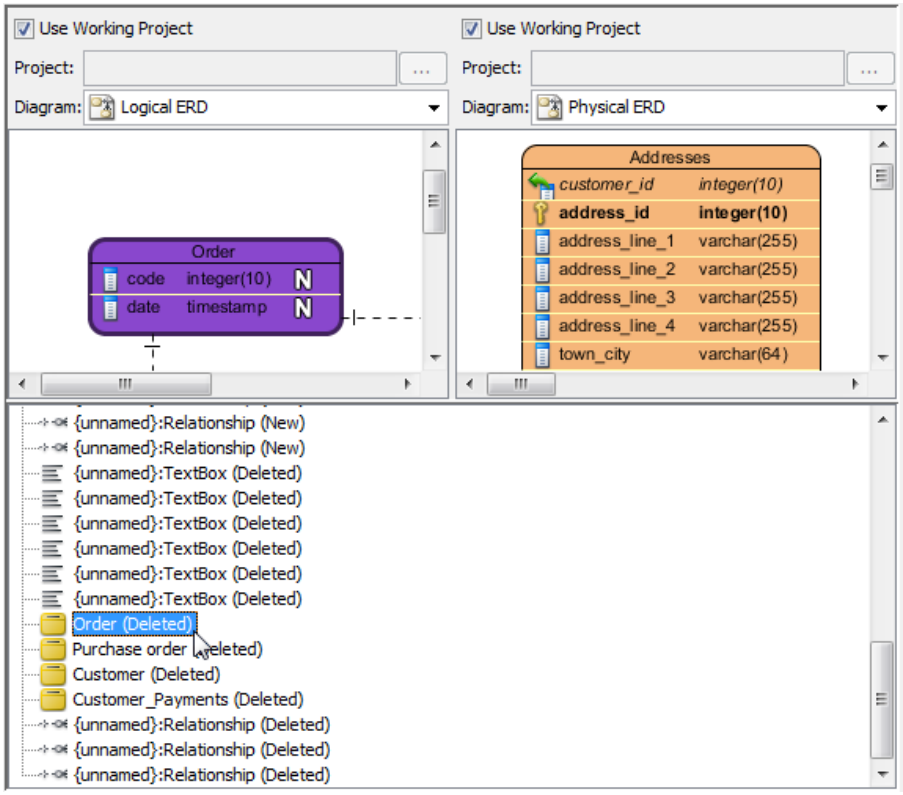
Select a diagram

The two diagrams are shown side by side on display pane. However, the ways of comparison has not yet been configured. Let's configure them one by one in the following steps.



Two diagrams are selected

4. At the top of the Visual Diff window, adjust the **Strategy**, **Compare**, **Base** and **Filter**.
5. Once everything is set, the differences of the two diagrams will be shown on the result pane.
6. If you want to view a specific shape, you may click its node on the result pane and then the selected shape will be painted in dark purple on the display pane.



Select a node in the result pane

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Using design pattern

Design pattern is a part of diagram that can be used in many different diagrams. In this chapter, you will learn what design pattern is, and how to apply it in your design.

Defining design pattern

You need to draw the pattern, and define it afterwards.

Applying design pattern

Defined patterns can be applied to your project, or another project (through an export and import of pattern).

Synchronize design pattern with teamwork server

Patterns can be shared among team members through the team collaboration support.

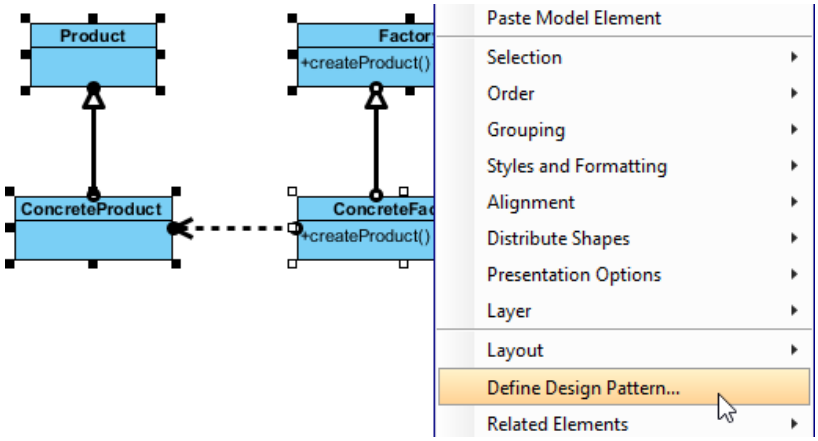
Defining design pattern

In Visual Paradigm, design pattern is a part of diagram that can be used in many different diagrams, thus form a pattern. Design pattern typically shows the shapes and more importantly, the relationships between the shapes. You can define and reuse design pattern in your project, across projects or share with your team members. You can define and apply design patterns on any kinds of diagram.

In order to apply a pattern, you need to define it first and save it as a pattern file ready for being used. To define a pattern, draw the pattern on diagram. After that, you can save the pattern, which is the diagram content as a pattern file.

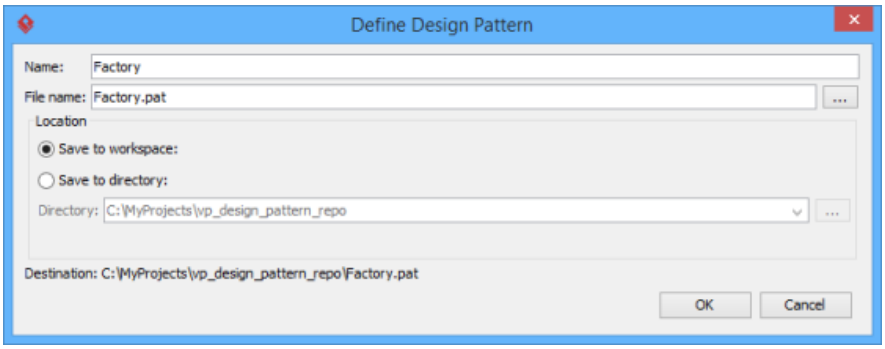
Defining design pattern

1. In the diagram where the pattern was drew, select the shapes to be involved in pattern.
2. Right click on any selected shapes, select **Define Design Pattern...** from the popup menu.



Defining design pattern

3. A design pattern is needed to save as a file. In the **Define Design Pattern** window, specify the name and file name for the pattern, with **.pat** as extension. You can save the pattern file to workspace for ease of sharing with other projects that will be opened in current workspace. Besides, you can save to another directory and share the **.pat** file with your team member for reusing. Click **OK** button to finish defining design pattern.



Naming design pattern

Related Resources

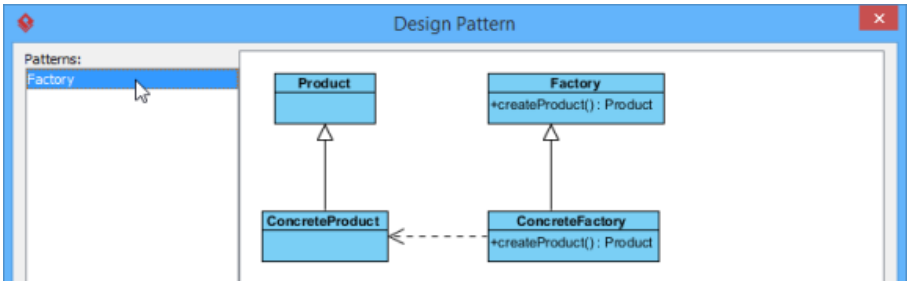
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Applying design pattern

You can apply a previously defined design pattern into your diagram and modify it to make it fit into your design. To apply a design pattern:

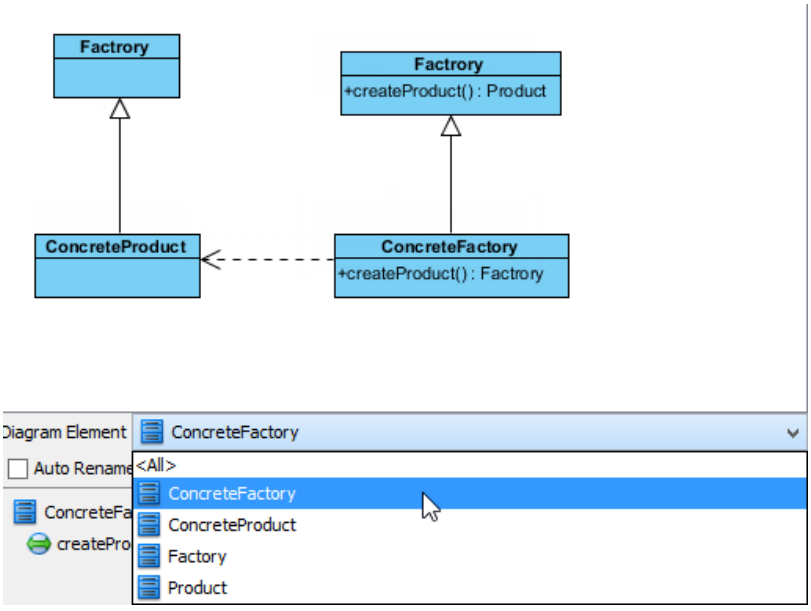
1. Open an existing diagram where you want to apply a design pattern or create a new diagram.
2. Right click on diagram, select **Utilities > Apply Design Pattern...** from the pop-up menu.
3. In **Design Pattern** window, you can see a list of defined patterns, select the desired pattern from the list.



Select pattern

If you have a **.pat** file, click **Add** button to import into the list.

4. For searching a specify shape on the defined pattern, select the shape from the drop-down menu of **Diagram Element** to filter the list.



Searching an element

You can also click on the shape or select a diagram element from the **Diagram Element** combo box to filter the list.

5. Finally, click **OK** button. The pattern will be applied to the diagram.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

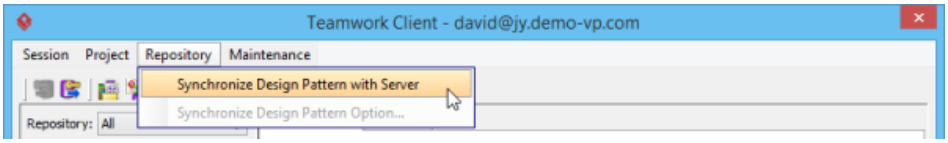
Synchronize design pattern with server

In Visual Paradigm, design pattern is a part of diagram that can be used in many different diagrams, thus form a pattern. Design pattern typically shows the shapes and more importantly, the relationships between the shapes. You can define and reuse design pattern in your project, across projects or share with your team members. You can define and apply design patterns on any kinds of diagram.

Once you have defined a design pattern, you can share it with your teammates through synchronizing it to server. Note that the synchronization of design pattern is independent from project commit/update, which means that the only way to have your design patterns available in server is by synchronizing them to server.

To synchronize design patterns:

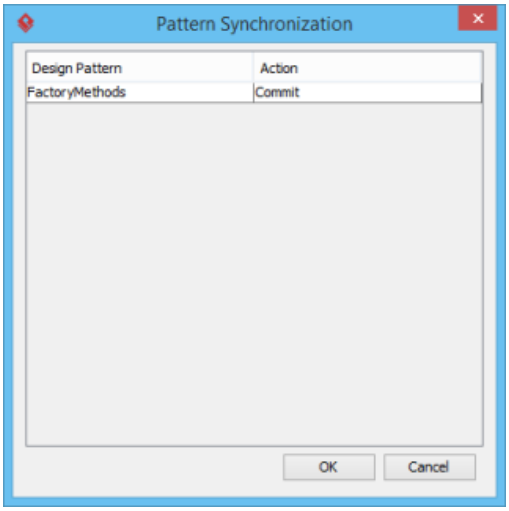
1. Select **Team > Utilities > Open Teamwork Client...** from the main menu.
2. In the **Teamwork Client window**, select **Repository > Synchronize Design Pattern with Server** from the menu.



Synchronize Design Patterns with Server

> **NOTE:** Only members who are permitted to configure design pattern can synchronize design patterns. Click here for details about configuring members' permissions .

3. Confirm the changes and click **OK** in the **Pattern Synchronization** window.



Pattern Synchronization

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model transitor

Model transitor helps define the transition of phrasees of work and maintain the traceability in between. Both the use of model transitor for shape and diagram will be covered in this chapter.

Model transitor for shape

Shows you the steps of adding and maintaining transition between shapes.

Model transitor for diagram (diagram transitor)

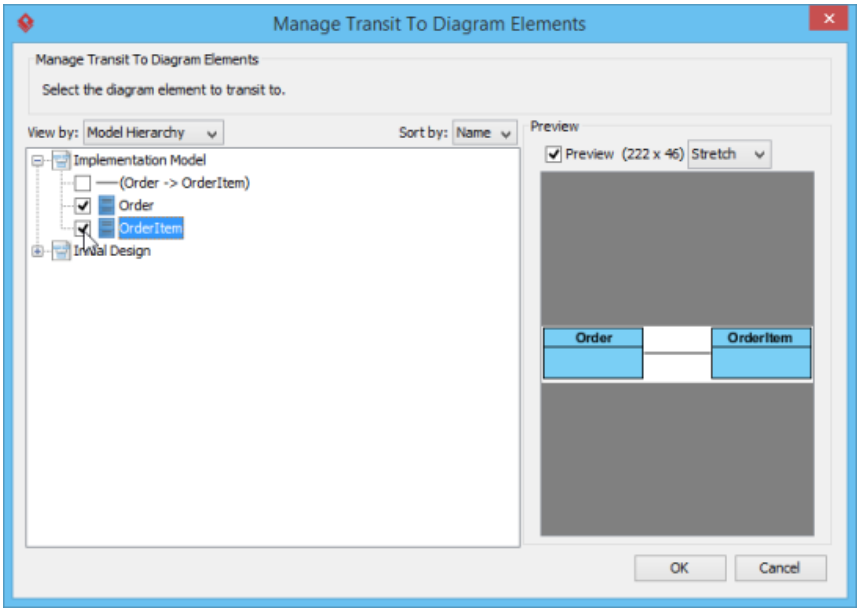
Shows you the steps of adding and maintaining transition between diagrams.

Model transitor for shape

You can use different diagrams for modeling different phases of development, such as a diagram for modeling the current system, and another diagram for modeling to system to be implemented. In order to trace the evolution of model elements across diagrams, you can make use of model transitor. Model transitor enables you to establish transit relationship between shapes. With the transition relationship, you can trace between shapes across diagrams.

Adding a transition between shapes

1. Right click on the shape you want to add a transition. It can be the source or target shape within the transition to add. If you are right clicking on the source shape, select **Related Elements > Transit To > Manage Transit To...** from the pop-up menu. If you are right clicking on the target shape, select **Related Elements > Transit To > Manage Transit From...** from the pop-up menu.
2. In the **Manage Transit To/From Model Elements** dialog box, select the shape(s) you want to transit to/from. You can select multiple shapes to transit to/from. For example, an initial *Purchase Order* class will be transited to an *Order* class and an *OrderItem* class.



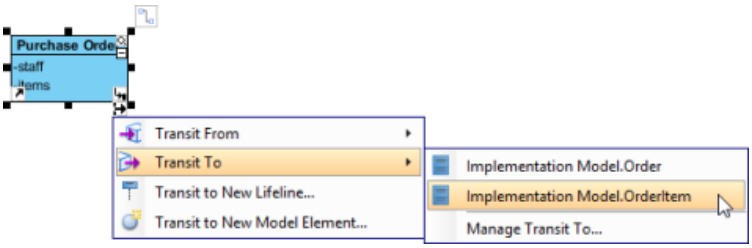
Select shapes to transit to

Navigating transited shape

Once a transition is added between two shapes, you can navigate between them. There are two methods to navigate to a transited shape. The first way is to go through the transitor popup menu of a shape. Right click on a shape and select **Related Elements > Transit From/To** from the popup menu, then the shape to navigate to.

Alternatively, make use of resource centric interface by following the steps below:

1. Move the mouse pointer over the shape that you want to navigate to its transited shape.
2. Press on the **Model Transitor** resource icon below the shape. Select **Transit From/To**, then the shape to navigate to.

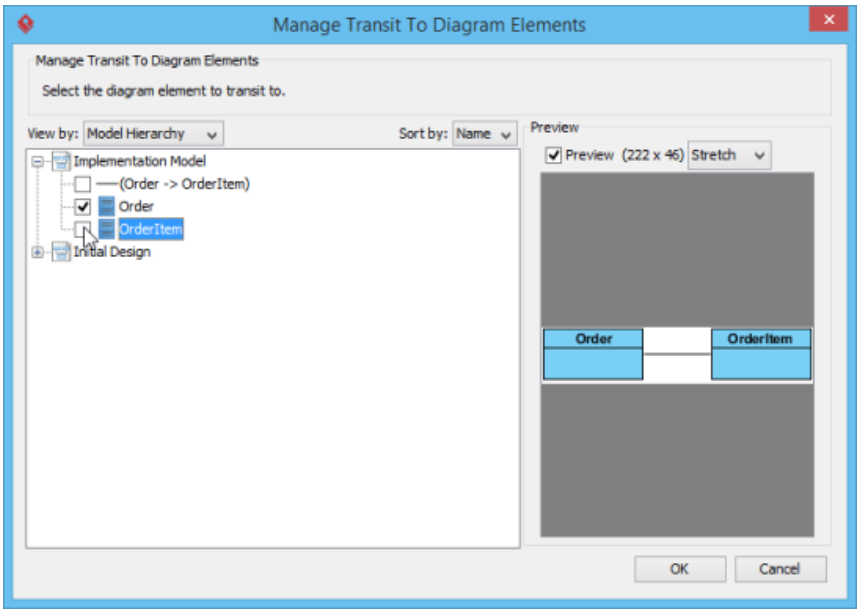


Navigate to a transited shape through the resource centric interface

Removing a transition

To remove a transition between shapes:

1. Right click on a shape and select **Related Elements > Transit From/To > Manage Transit From/To** from the popup menu.
2. In the **Manage Transit To/From Model Elements** dialog box, de-select the shapes that you do not want to transit with. Click **OK** to confirm.



De-select shapes to withdraw from transition

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

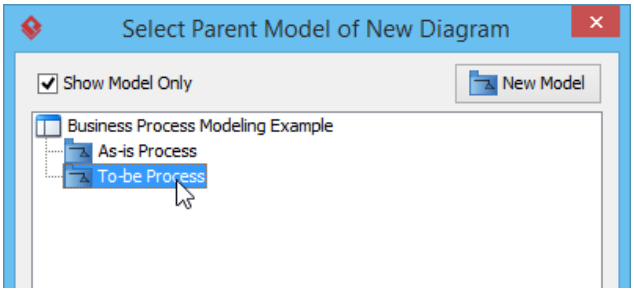
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model transitor for diagram (diagram transitor)

You can use different diagrams for modeling different phases of development, such as a diagram for modeling the current system, and another diagram for modeling to system to be implemented. Sometimes, diagrams across phrases are similar, but little variation. Model transitor enables you to duplicate a diagram with transition added in between. You can then continue modeling in the new diagram by using the original diagram's content as base.

Transiting to a new diagram

1. Right click on the background of diagram that you want to transit from. Select **Utilities > Transit to New Diagram...** from the pop-up menu.
2. The **Select Parent Model of New Diagram** window appears, enabling you to select a model for storing diagram. Visual Paradigm encourages structuring project with model for easier accessing of model elements and increasing application performance. If you want to place the new diagram in a model, select one or click **New Model** button at the top right to create one and select it. If you do not want to store diagram inside any model, do not make any model selection. Click **OK** button to continue.



Selecting a model for storing the new diagram

3. You can then start editing the new diagram.

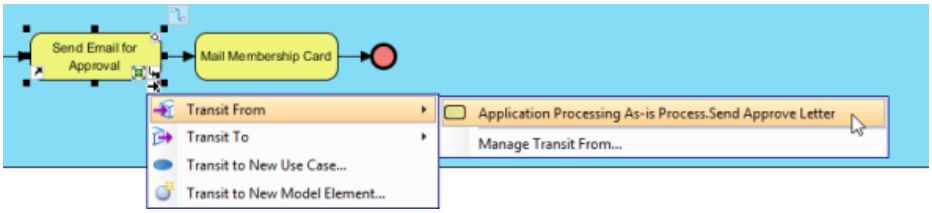
Editing transited diagram

Navigating transited shape

By adding a transition between diagrams, transitions are automatically added between shapes in the two diagrams. With the transition between shapes, you can navigate between them. There are two methods to navigate to a transited shape. The first way is to go through the transitor popup menu of a shape. Right click on a shape and select **Related Elements > Transit From/To** from the popup menu, then the shape to navigate to.

Alternatively, you can make use of the resource icon by following the steps below:

1. Move the mouse over the shape that you want to navigate to its transited shape.
2. Press on the **Model Transitor** resource icon below the shape. Select **Transit From/To**, then the shape to navigate to.



Navigate to a transited shape

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Customizing elements with profile

A profile provides a generic extension mechanism for customizing model elements for different purposes. It is closely related to stereotype and tagged value. Details will be covered in this chapter.

Creating a profile

You will see how to create a profile in Model Explorer.

Drawing a profile diagram

A profile can be modeled through a profile diagram.

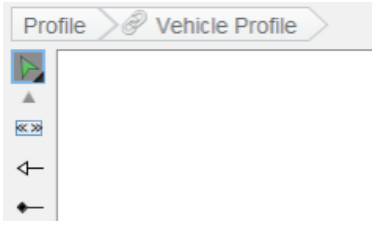
Drawing a profile diagram

A profile diagram enables you to create domain and platform specific stereotypes and define the relationships between them. You can create stereotypes by drawing stereotype shapes and relate them with composition or generalization through the resource-centric interface. You can also define and visualize tagged values of stereotypes.

Creating a profile diagram

To create a profile diagram:

1. Select **Modeling > Profile > New Profile** from the toolbar
2. Name the diagram and press **Enter** to confirm.



Naming a profile diagram

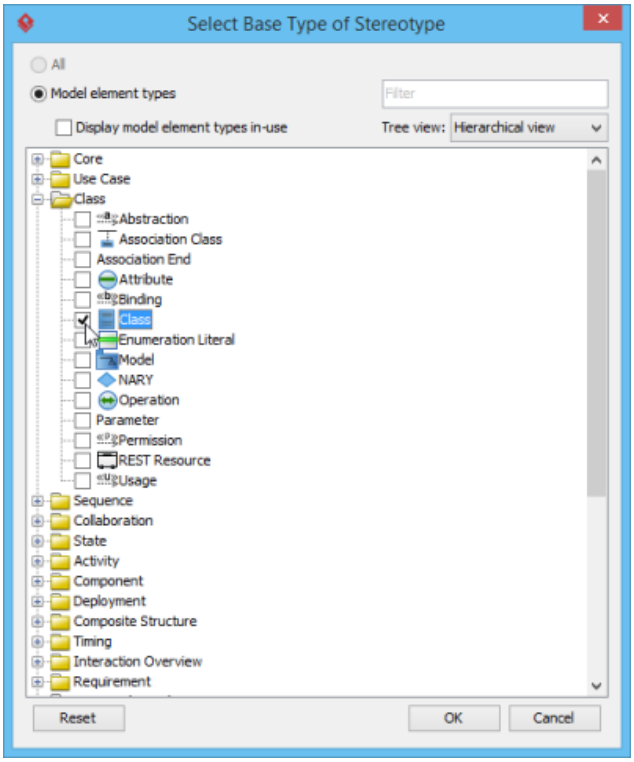
Drawing a stereotype

To draw a stereotype in profile diagram:

1. Select **Stereotype** in diagram toolbar.

*Selecting **Stereotype** in diagram toolbar*

2. Click on the diagram to create a stereotype.
3. In the **Select Base Type of Stereotype** window, select the base type of stereotype from the model type tree. A base type is the type of model element that the stereotype will extend.



Selecting a base type

NOTE: You can check **Display model element types in-use** to list only the types of model elements used in project.
The text box **Filter** enables you to filter model element type base on the type name (e.g. enter class to list only class)

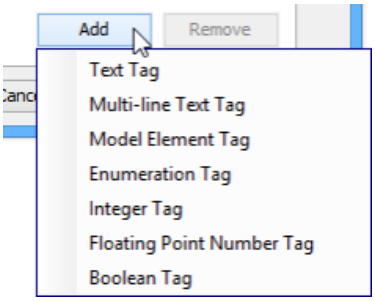
4. Click **OK**. Name the stereotype and press **Enter** to confirm creation.

Defining tagged values for stereotypes

A stereotype may have properties, which may be referred as tag definitions. When a stereotype is applied to a model element, the values of the properties may be referred as tagged values.

You can define tagged values for a stereotypes. By doing so, when you apply the stereotype with tagged values defined to a model element, you can fill in the values for the model element.

1. Right click on a stereotype shape and select **Open Specification...** from the popup menu.
2. In the **Stereotype Specification** window, open the **Tagged Value Definitions** tab.
3. Click **Add**. Select the type of tagged value to define. The type of tagged value limits the type of content user can enter for a tag.

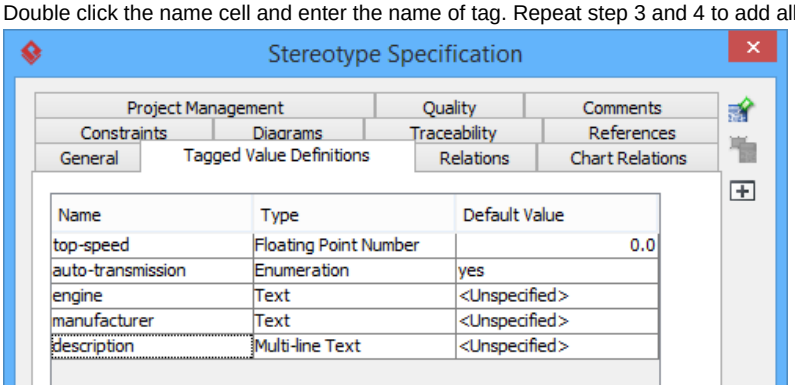


Add tag

Tag type	Description
Text	The most common and general type of tagged value that consists of words.
Multi-line Text	The value of tag is a text in multiple lines.
Model element	The value of tag is a model element in project.
Enumeration	The value of tag can be chosen from a list of possible values. For example, to select "red" out of values red, green and blue.
Integer	The value of tag must be a real number.
Floating point number	The value of tag must be a number that consists of one or more digits.
Boolean	The value of tag is a boolean (true/false).

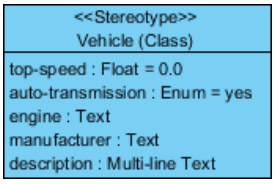
Type of tags

4. Double click the name cell and enter the name of tag. Repeat step 3 and 4 to add all tagged values for this stereotype.



Tags defined for an Vehicle stereotype

5. You can assign a default value to a tag by editing the **Default Value** cell. Usually, you give a tag a default value when the value is true in most cases. For example, a tag "in-door-temperature" can have "25" as default value. By confirming changes, you can see the stereotype show on diagram, with tagged values show below the stereotype name.



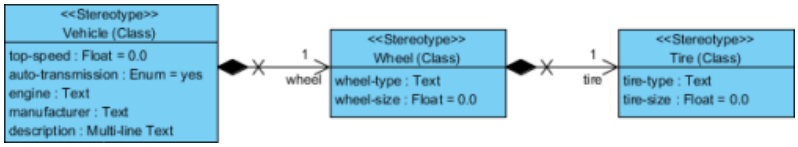
Stereotypes with tagged values defined

Relating stereotypes

Stereotypes can be related with each other by composition or generalization. Relating stereotypes not just affect the modeling in profile, but also the model elements that the stereotypes will be applied to.

Composition

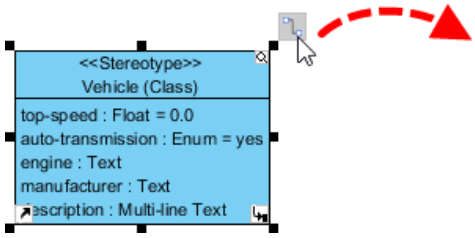
A composition relationship shows a "part of" relationship between stereotypes. The composite stereotype has responsibility for the existence and storage of the composed stereotype.



Composition between stereotypes

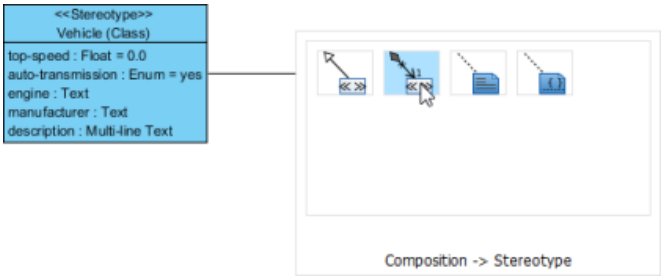
To create a composed stereotype:

1. Move the mosue pointer over a stereotype. Press on the **Resource Catalog** button and drag it out.



To create a composition

2. Select **Composition -> Stereotype** from Resource Catalog.



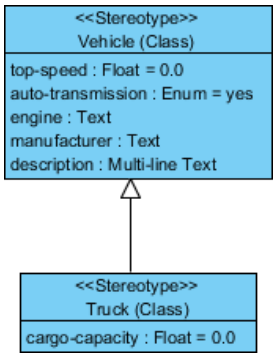
Create Composition

3. Name the stereotype and press **Enter**.

Since composition models a "part of" relationship, when you apply a composite stereotype to a model element, you can add tagged value defined in composed stereotype in the model element. For example, stereotype *Vehicle* is composed of stereotype *Wheel*. If you apply stereotype *Vehicle* to a class, you can specify the properties of tagged values as defined by both stereotype *Vehicle* and *Wheel*.

Generalization

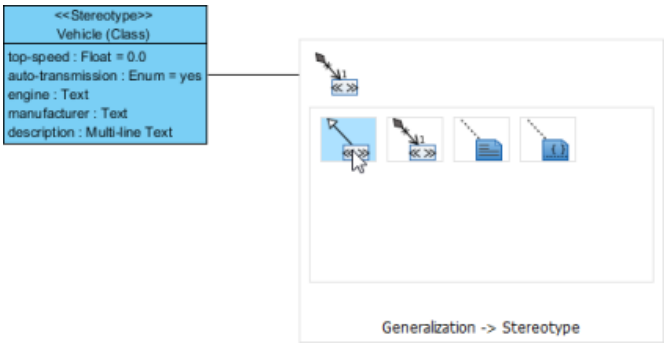
A generalization relationship shows a "kind of" relationship between stereotypes.



A generalization relationship

To create a specific stereotype from a general stereotype:

1. Move the mosue pointer over a stereotype. Press on the **Resource Catalog** button and drag it out.
2. Select **Generalization-> Stereotype** from Resource Catalog.



Create Generalization

3. Name the stereotype and press **Enter**.

Since generalization models a "kind of" relationship, when you apply a specialized stereotype to a model element, you can add tagged value defined in general stereotype in the model element. For example, stereotype *Vehicle* is a generalized stereotype of *Truck*. If you apply stereotype *Truck* to a class, you can specify the properties of tagged values as defined by both stereotype *Vehicle* and *Truck*.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Mind Mapping Diagram

Mind mapping is a tool to help you in brainstorming and organizing ideas, concepts, words, tasks through a visual note-taking way. It sometimes helps to capture requirements and business process, too. You will see how to draw and edit mind mapping diagram in VP-UML.

Drawing mind mapping diagram

Draw mind mapping diagram and create mind mapping nodes in diagram.

Formatting nodes

Decorate nodes for better organization of similar ideas.

Linking nodes

Relating concepts with link connectors.

Reference to resources

Maintain reference between node and other artifacts in project.

Relocating a branch

Restruct mind mapping diagram by relocating a branch of nodes.

Layout diagram

Make diagram looks tidier by performing layout.

Drawing mind mapping diagram

Mind mapping is a tool to help you in brainstorming and organizing ideas, concepts, words, tasks through a visual note-taking way. It sometimes helps to capture requirements and business process, too. Modelers can create and link model element (such as task, use case, classes) with mind mapping node. The traceability can be kept between initial idea (mind mapping node) and detail design elements (e.g. class).

Creating mind mapping diagram

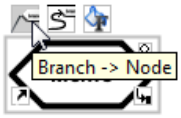
1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Mind Mapping Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.
6. This create a mind mapping diagram with a central idea node appears in it. Immediately name the central idea node and press **Enter** to confirm. You can then start drawing the diagram by branching nodes from the central idea nodes.

Naming central idea node

Creating branch with resource centric interface

Mind mapping is formed by nodes that represent ideas or concepts. They are connected with each other, showing the flow of thinking. To create a new branch of nodes from an existing node:

1. Move the mouse pointer over a node. Press on the resource icon **Branch -> Node**.

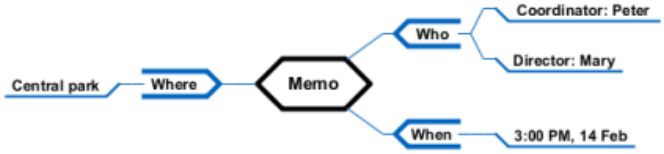


To create a node

2. Drag it out. Release the mouse button to create the node.

A node is created

3. Name the node and press **Enter** to confirm. Create the other nodes by repeating the same steps.

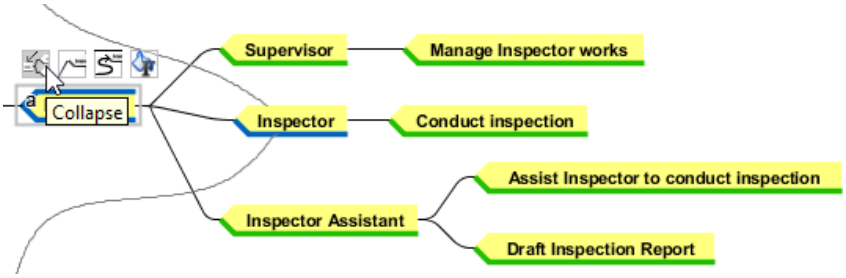


A mind mapping diagram is created

Collapsing/Expanding a branch

When reading a mind map, you may want to collapse some branches to make a diagram cleaner and easier to read. To collapse a branch:

1. Move the mouse pointer over a node.
2. Click on the resource icon **Collapse**.



To collapse a branch of node

Similarly, you can expand a collapsed branch by clicking on the **Expand** resource that appear when moving your mouse pointer over a collapsed node.

Identifying collapsed nodes

To locate the collapsed nodes on a mind map, simply turn on the Model Indicator by selecting **View > Model Indicator** from the toolbar. By doing so, those collapsed nodes will have the **Expand** resource appear above them.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

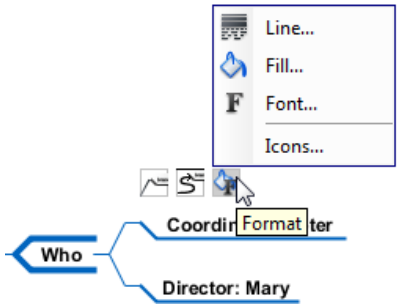
Formatting nodes

You can set colors to nodes to represent different kinds of idea and concepts. You can also set icon(s) to a node to represent the nature of a node, such as a telephone icon for concepts related to contacting some body.

Changing the line, fill or font style of node

1. Select the node(s) that you want to format. Multiple node selection can be made by a range of selection or by pressing the **Ctrl** key and select the nodes subsequently.

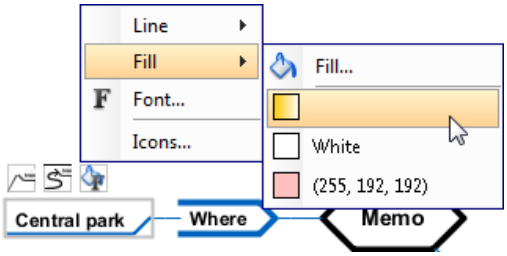
2. Move the mouse pointer to a node within the selection. Click on the **Format** resource icon.



To format nodes

3. Select either **Line...**, **Fill...** or **Font...** from the popup menu to change specific type of format. (Read the coming sub-sections for details about line, fill and font settings)

Once you have confirmed your selection, your choice will be memorized. When you want to apply the settings on other nodes, you can select the new nodes, re-open the same popup menu and select the setting through the popup menu.



Choosing a color for node

Line

Line settings control the appearance of border around node(s). You can adjust the style (e.g. dash, solid), the weight, which is the thickness of line, the color and the level of transparency.

Nodes with brown border

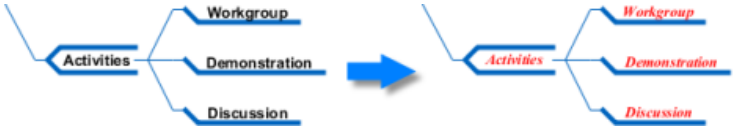
Fill

Fill settings control the background color of node(s). You can apply solid and gradient colors as well as to control the transparency.

Nodes with gray background

Font

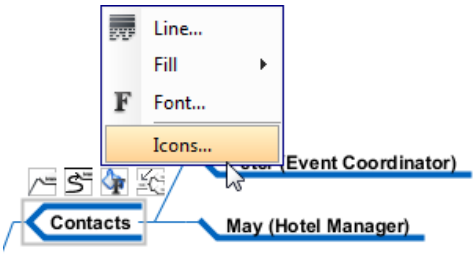
Font settings control the font style, size, type and color of text appear on a node.



Nodes with Times new roman text

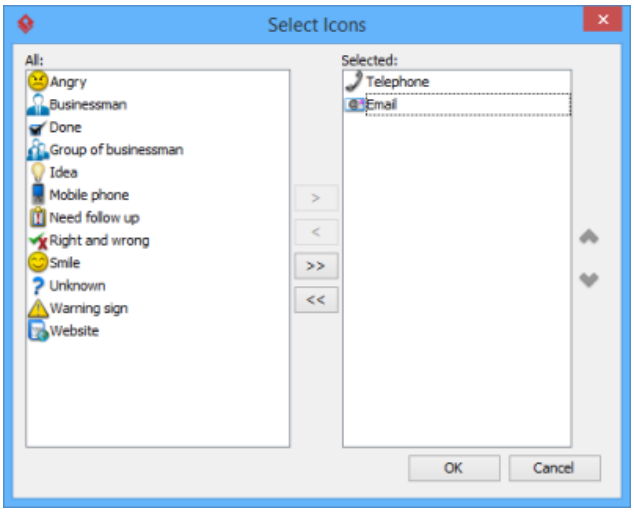
Changing icon of node

1. Select the node(s) that you want to set icon. Multiple node selection can be made by a range of selection or by pressing the Ctrl key and select the nodes subsequently.
2. Move the mouse pointer to a node within the selection. Click on the **Format** resource icon, then select **Icons...** from the popup menu.



To edit icons for a node

3. In the **Select Icons** dialog box, select the icon to set and click **>** to assign it to the node(s). Click **OK** to confirm.



To select icons for chosen node(s)

A node with icons

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

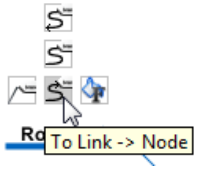
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Linking nodes

Other than the traditional branch connector that represents a generation of idea, you can link related ideas and concepts by using a link connector. There is no exact definition about how two nodes can said to be related. It is up to the designer whether to link the nodes or not. As long as you want to represent that two nodes and related, the relationship is meaningful and you can add a link between them.

To link nodes:

1. Move the mouse cursor over the source node. Press on any of the resources: Link, To Link, From Link. To and From links are directed relationship, which shows an arrow to indicate the flow from source to target node.



To link to antoher node

2. Drag to the target node and release the mouse button.

> **NOTE:** Unlike traditional resource icons, the Link resources must be released on an existing node. Releasing on diagram will not result in creating a new node.

3. Optionally enter the name of link. Press **Enter** to confirm editing.

Link is created between nodes

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

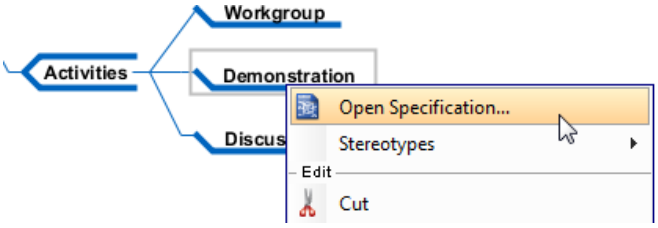
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reference to resources

You can add references to node, to reference to both internal and external resources such as a shape, a diagram, a file, a URL, etc. For example, to make a node *Prepare Agenda* link to a document of agenda template. This makes a mind map more informative by providing additional information from a mind map which might be casually developed.

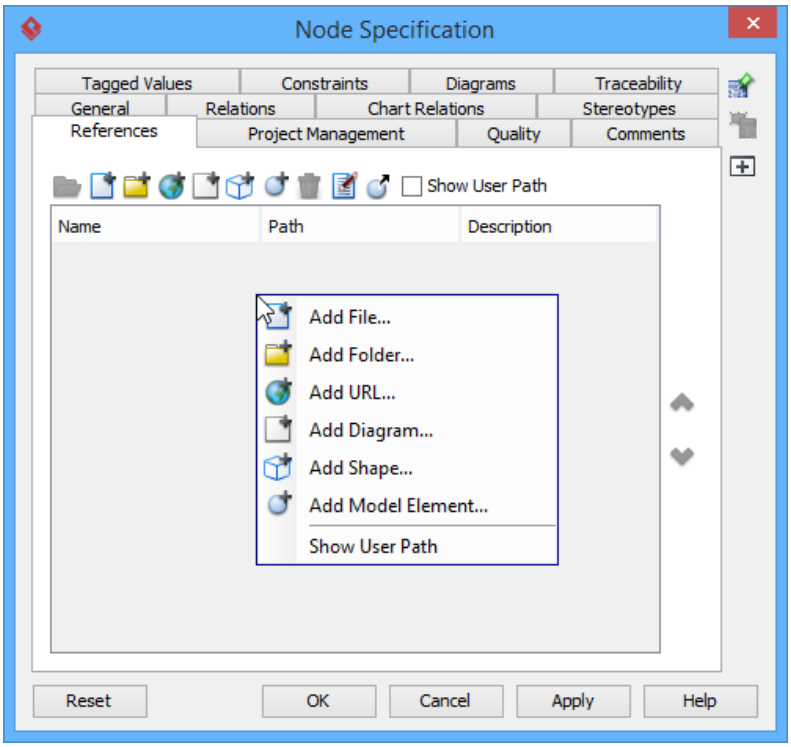
To add a reference:

1. Right click on the node you want to add reference and select Open Specification... from the popup menu.



Opening node specification

2. In the node specification, open the **References** tab. Right click on the center of pane and select the type of reference to add from the pop-up menu.



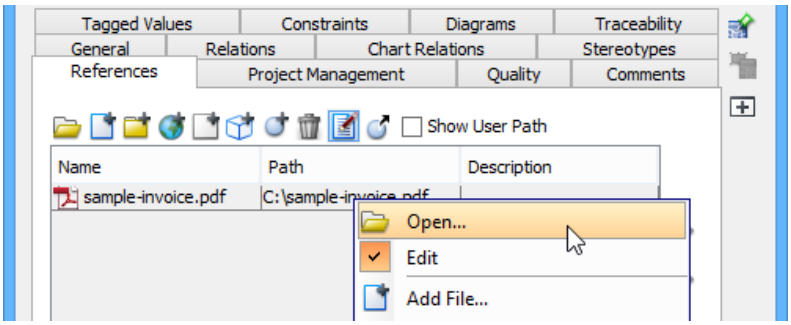
Add a reference

Type of reference	Description
File	An external file.
Folder	An external folder.
URL	A URL. For example, http://www.visual-paradigm.com
Diagram	A diagram in the opening project, such as a requirement diagram.
Shape	A shape in the opening project, such as a use case shape on a use case diagram.
Model element	A model element in the opening project, such as a use case.

Description of different kinds of reference

3. Supply the information of reference such as the file path of a file reference, a diagram for a diagram reference.
4. Click **OK** to confirm.

Once a reference has been added, you can open it from the **References** tab by right clicking on it and selecting **Open...** from the popup menu.



Open a referenced resource

Related Resources

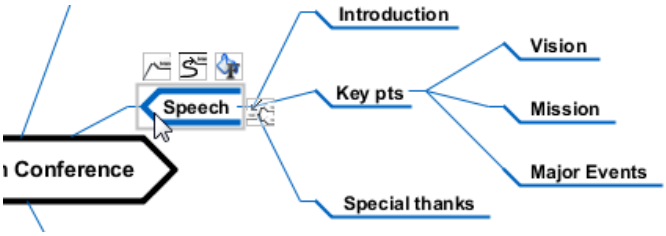
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Relocating a branch

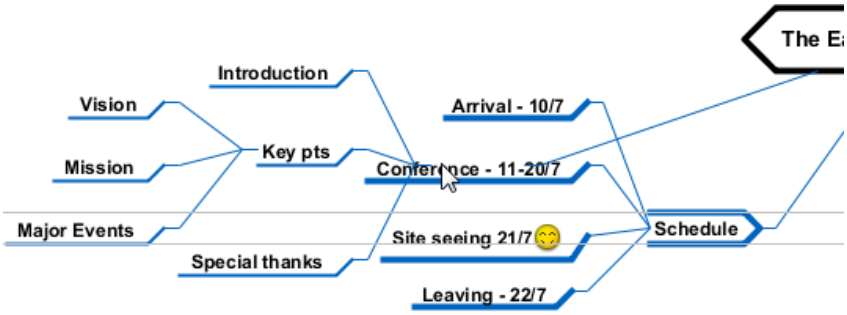
In case a branch of nodes is mis-positioned, you can reposition it to under another node through drag and drop. Here are the steps:

1. Press on the pointer end of the first node of a branch that you want to reposition.



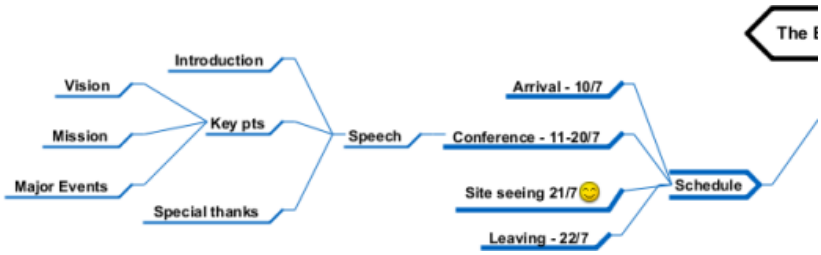
Pressing on the pointer end of a node

2. Drag to node that you want to move the branch to.



Dragging over the target node

3. Release the mouse button.



Mouse button released

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

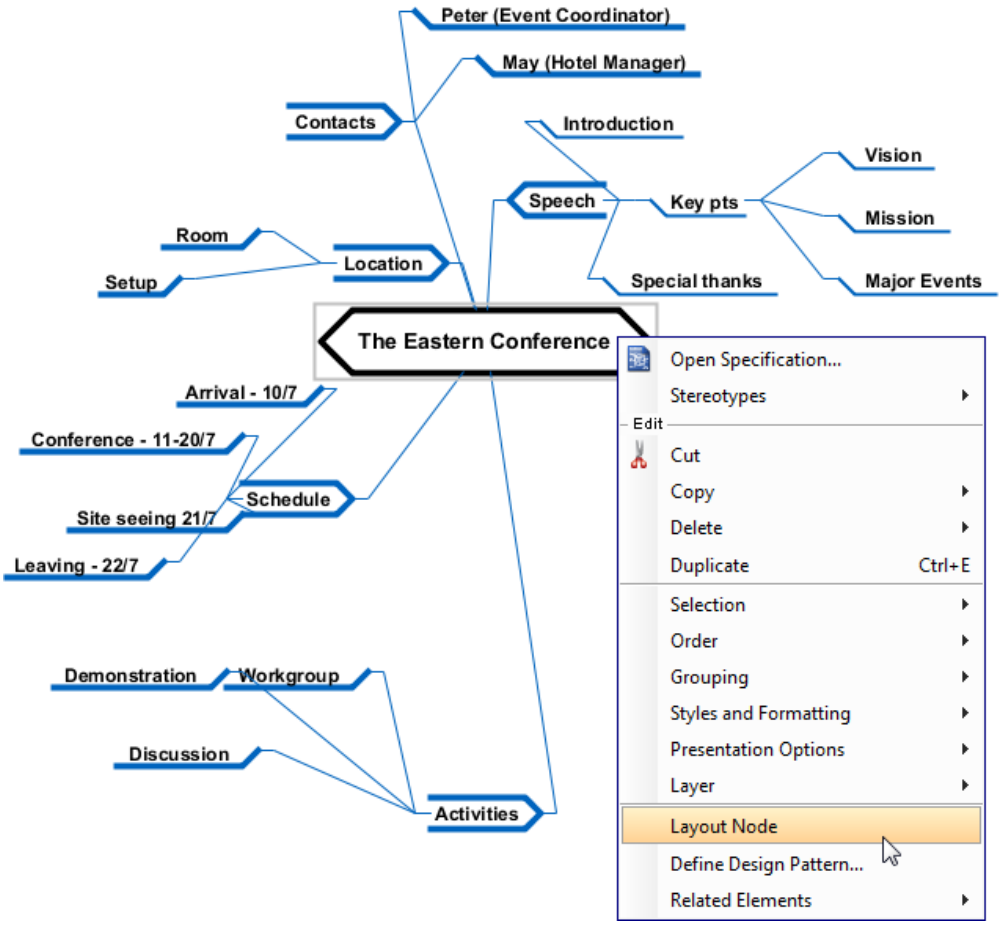
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Layout diagram

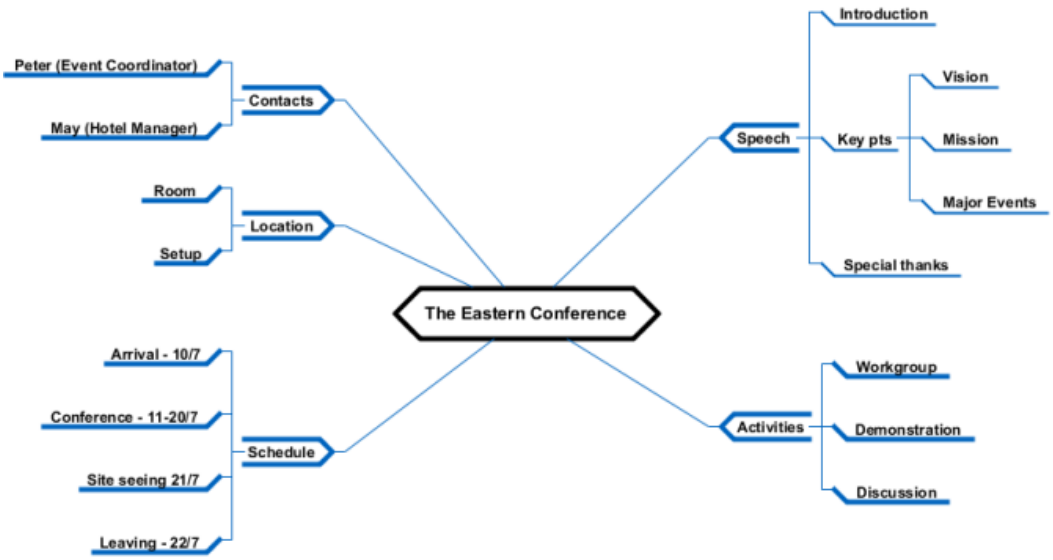
In a mind mapping diagram, ideas are stretching out across, which leads to difficulties in tracing nodes with different ideas due to the unorganized nodes. It will be time consuming to rearrange the idea nodes manually. This also affects our brainstorming procedure by caring the tidiness of diagram. By performing a layout, you can keep brainstorming and drawing the diagram without caring about the tidiness of the diagram. You can perform a layout once the diagram is drawn. Any nasty diagrams can be well organized in a breeze.

Diagram based

By performing a diagram based layout, all idea nodes in diagram are included in the range of layout. To perform a diagram based layout, right click on the central idea node and select **Layout Node** from the popup menu.



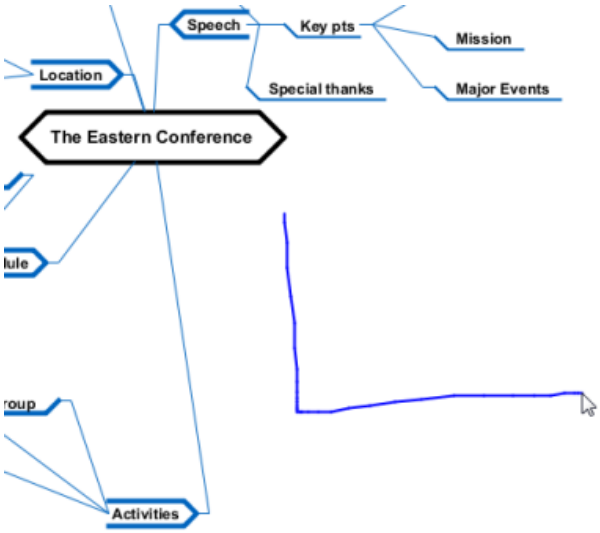
To layout nodes on a diagram



Result of diagram based layout - all nodes are layout-ed

Using resource-centric interface

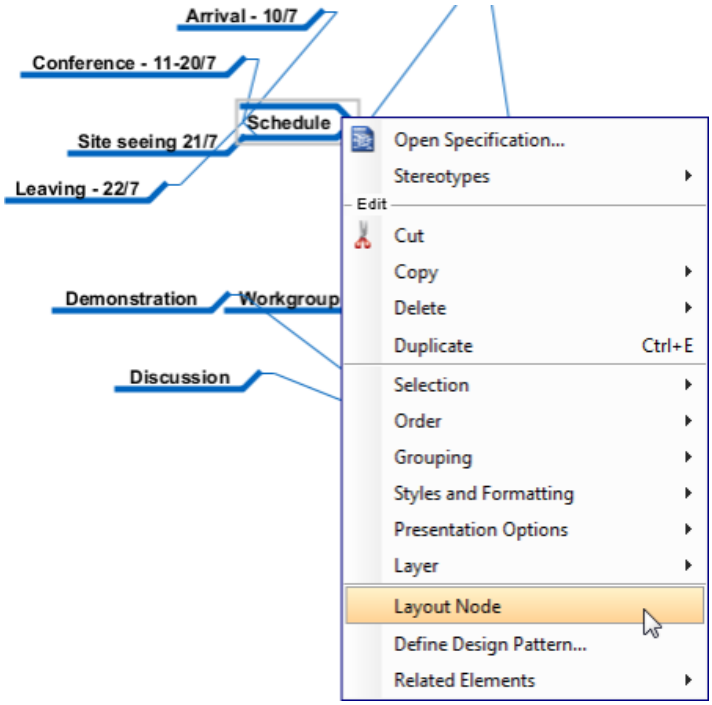
Mouse gesture enables you to layout shapes on a diagram through the movement of mouse. To perform a layout with mouse gesture, right press on the diagram background, sketch a "L" like gesture path and release the mouse button to execute layout.



Layout diagram with mouse gesture

Node based

By performing a node based layout, only the chosen node and its descendant nodes are included in the range of layout. To perform a node based layout, right click on the idea node and select **Layout Node** from the popup menu.



To perform a node based layout

Result of node based layout - only Schedule node is layout-ed

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Brainstorm

VP-UML enables you to record important ideas during a meeting through a note-taking feature called "Brainstorm". During the meeting, you create note (shapes) in a corkboard -like diagram. When the meeting ends, you may organize the notes, and derive a diagram from them.

Using Brainstorm

Introduce to Brainstorm and shows you how to create a Brainstorm diagram.

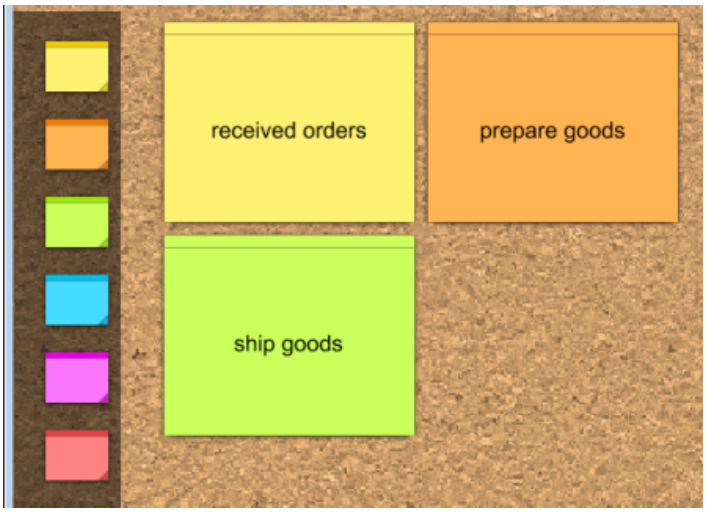
Realize Brainstorm Notes

Teaches you how to realize notes into model elements.

Using Brainstorm

Very often, you model a system not just by imagination but with facts, knowledges and customers' ideas. You may meet with users, understand how they work, identify their requirements and proceed to visualize their needs with models.

Visual Paradigm enables you to record important ideas during a meeting through a note-taking feature called "Brainstorm". During the meeting, you create note (shapes) in a corkboard -like diagram. When the meeting ends, you may organize the notes, and derive a diagram from them. This helps to ensure all important thoughts from users are well recorded and won't be lost when constructing a model.



A sample Brainstorm diagram

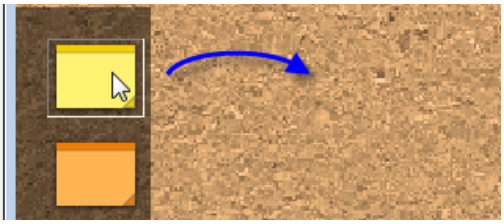
Creating Brainstorm diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Brainstorm**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating a note

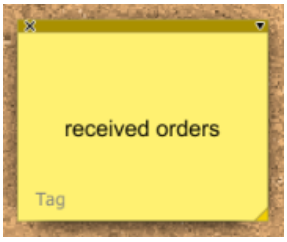
Brainstorm diagram applies a corkboard theme. You may add notes to it to record ideas and thoughts collected. To create a note:

1. Press and drag in diagram toolbar a note with the desired color.



Creating a note

2. Release the mouse button on the diagram to create a note.
3. Enter the note content. Press **Ctrl-Enter** to finish editing.



A note is created

Imagine when you are having a meeting with user and taking notes with Brainstorm, you may not be perfectly sure whether the notes you create are utlimately important or not. As long as you think that the information may help you model, it is worthwhile to note it down for now. The key idea is to add notes in a casual manner. Do not spend too much time on judging the importance of the content. Otherwise, you may miss out information that is really important during the process of strenuous sorting.

Deleting a note

If a note has been created by mistake or the note text is no longer correct/meaningful, you may want to delete the note. To delete a note, you may select it and press the **Delete** key. Alternatively, press on the cross button at top left of note shape to perform a deletion.

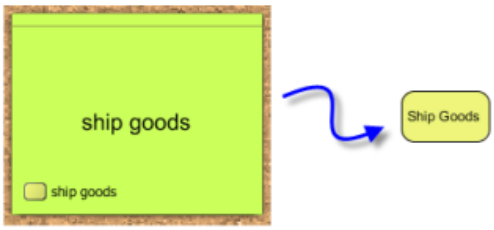
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Realizing Brainstorm notes

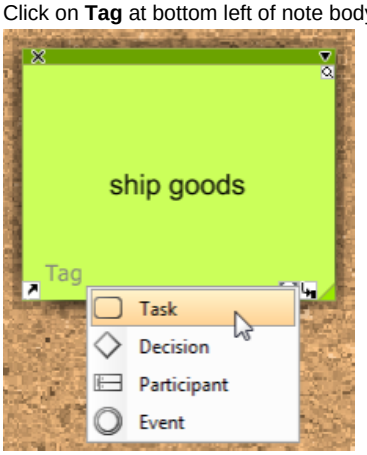
Visual Paradigm enables you to record important ideas during a meeting through a note-taking feature called "Brainstorm". During the meeting, you pay attention to what the participants say and create note (shapes) in a corkboard -like a diagram to record the key points. After the meeting, you may make use of the notes collected to help you construct the model. To optimize the process, Visual Paradigm supports a realize function to transform notes into model elements.



Producing a task from a note

To realize a note:

1. Look for the note that can help you to construct a model element. For example, when you see a note with text "ship goods", you may want to create a task *Ship Goods* from it.
2. Click on **Tag** at bottom left of note body. Select the type of element from the popup menu, such as Task.



Select note type

3. Click the tag at the bottom left and select **Realize...** from the popup menu.
4. In the **Transit Model Element** window, enter the properties of the model element you are going to produce. Click **OK** at bottom right to continue.
5. The **Visualize Model Element** window enables you to show the model element on a diagram. You may show it on a new diagram by selecting Create new diagram and entering the diagram name. Or, show it in an existing diagram by selecting Show in existing diagram and selecting the diagram to show. Or, not to show it on any diagram by selecting Do note visualize. Click **Create/ Show/ Close** at bottom right.

Changing note tag

If you have selected a tag for a note and you want to change it, click on any tag at bottom left of note shape and select **Type > [New Tag]** from the popup menu. This will clear the previously selected tag and apply the newly selected one. Note that once a tag has been realized, you cannot change it to another type anymore.

Adding note tag

You may add multiple tags to a note and realize multiple model elements from different tags (one model element per tag). To add another tag, click on any previously added tag at bottom left of note shape and select **Add > [New Type]** from the popup menu.

Deleting note tag

If you think that a note tag is no longer suitable for the note, you may delete it. To delete a tag, click on the tag you want delete from note and select **Delete** from the popup menu. Note that if the tag has been realized, deleting the tag would not delete the realized model element. And if you add the tag again, the realization relationship will be maintained.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating HTML/PDF/Word report

Report generation is the process of producing a report for sharing your design work and model specification with teammates and clients. You can generate reports in different formats such as HTML, PDF or MS Word for reading or publishing in different environments. In this chapter, you will see how to generate a report, and how to configure the generation process and output.

Generating report

Shows you the basic steps in generating a HTML, PDF or MS Word report.

Configuring report

Gives you a description of all the report generation options.

Generating document

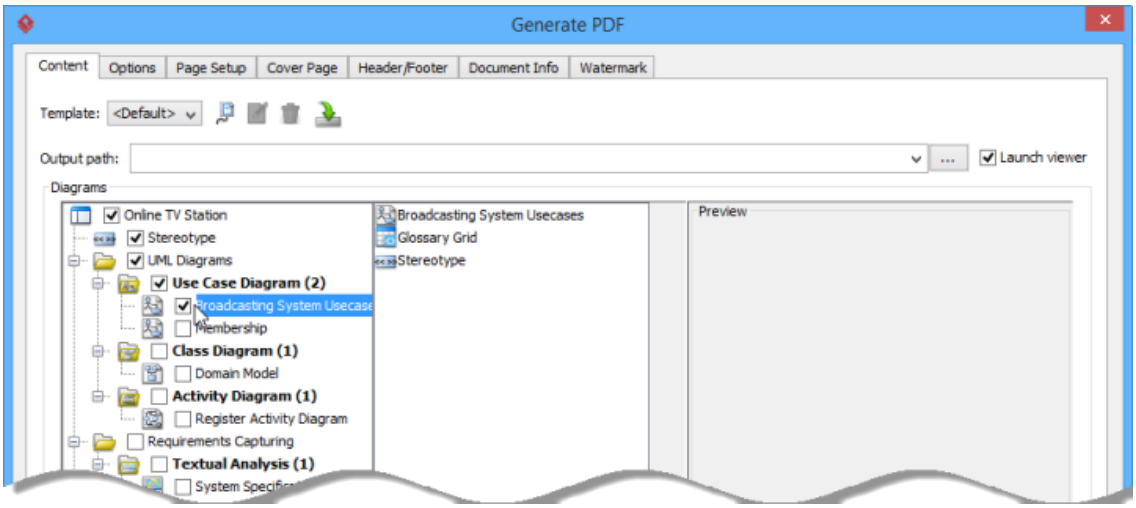
Document generation is the process of producing a document for sharing your design work and model specification with teammates and clients. You can generate documents in different formats such as HTML, PDF or MS Word for reading or publishing in different environments. They differ in file format but have the same layout. In this chapter, we will go through the core steps in document generation.

To generate a document:

1. Select **Tools > Doc** from the toolbar. Then, select **Generate HTML Doc..., Generate PDF Doc...** or **Generate Word Doc...** depending on the type of document you want to generate.
2. In the **Generate HTML/PDF/Word** window, fill in the output path where the document should be generated to.

> **NOTE:** For HTML document, specify the folder of the HTML files to be generated.
> For PDF document, specify the file path of PDF file (*.pdf) to be generated.
> For MS Word document, specify the file path of the document file (*.docx) to be generated.

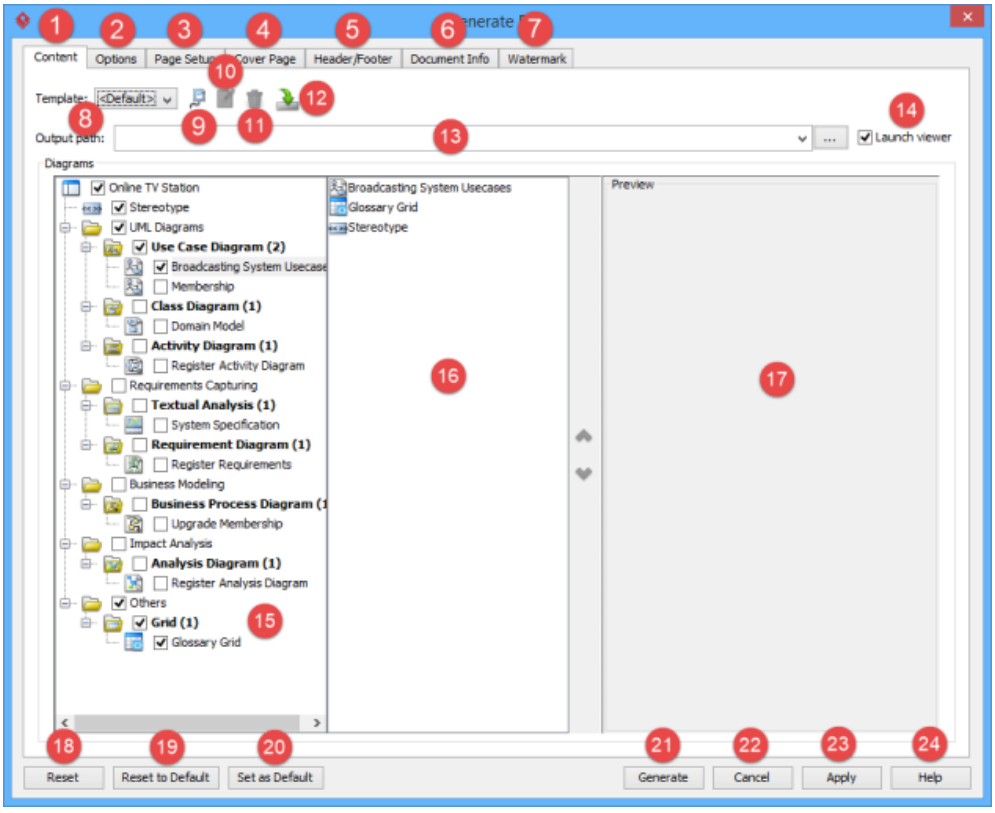
3. Select the grid(s) and/or diagram(s) to be included in document.



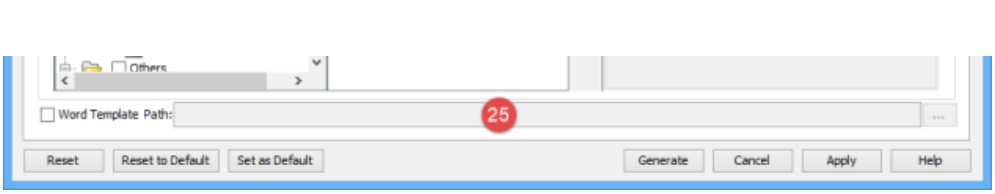
Select diagram to include in document

4. Make any necessary configuration such as the page layout, cover page, etc. For details about configuration, refer to the next chapter.
5. Click **Generate** to proceed with generation.

Overview of document generation window



Overview of document generation window



The bottom part of report generaition window for MS Word report

No.	Name	Description
1	Content	The main page of document generation that allows you to select the diagram(s) to generate document.
2	Options	Configurable options for detailed document configuration.
3	Page Setup	The setup of layout of the document.
4	Cover page	Cover page of the document.
5	Header/Footer	The content of header and footer.
6	Document info	To define document info.
7	Watermark	To add watermark to diagram images.
8	Template	You can define a template for document generation by clicking on the drop down menu next to **Template** and selecting **<New>**. Once a template is defined, you can select it from the same drop down menu and proceed to generation with the template. For details about document customization, please read the next chapter.
9	Use external template	Click to link to an external template file.
10	Edit template	Click to edit the template selected in the drop down menu of **Template**.
11	Delete template	Click to delete the template selected in the drop down menu of **Template**.
12	Import template	Click to import a template file into the current project.
13	Output path	The output path of document to be generated.
14	Launch viewer	Check to open the document automatically after generation.
15	Available diagram list	The list of diagrams in opening project.
16	Selected diagram list	The list of diagrams selected to generate document.
17	Preview	Preview of diagram which have been selected in the list of selected diagram
18	Reset	Reset changes made in this window
19	Reset to default	Reset changes made in this window to default settings.
20	Set as default	Set the settings in this window to default.
21	Generate	Click to generate document.
22	Cancel	Click to close the document window.
23	Apply	Click to apply the changes made in document, causing the reopening of this window to restore the applied settings.
24	Help	Click to read the help contents.
25	Word template path	Available only to MS Word document generation, this option enables you to specify the path of MS Word document file that you want the generator to use as template. Document generator will append the template file content in front of generated document. In other words, you can prepare a file for cover page. Apart from this, style will also follow the definition in template file. For details, please read the section *Generating MS Word document with template (MS Word document only)*.

Description of document generation window

Generating MS Word document with template (MS Word document only)

At the bottom of the MS Word document generation window, there is an option **Word template path** with a text box next to it for filling in the path of template file. A Word template file provides the start up contents and defines the style of document. During document generation, the generator will make a copy of the template file, treat the copied file as base, append the generated content to the copied file and save the file to the destination path. By using a word template, you can define your own headers/footers, cover page, start up content, styles for your generated document.

Details

User

Name	Value
Visibility	public
Abstract	false
Leaf	false
Root	false
Author	Peter
Create Date Time	Feb 9, 2010 8:30:18 AM
Last Modified	Feb 9, 2010 8:38:14 AM
Business Model	false
Actor ID	1

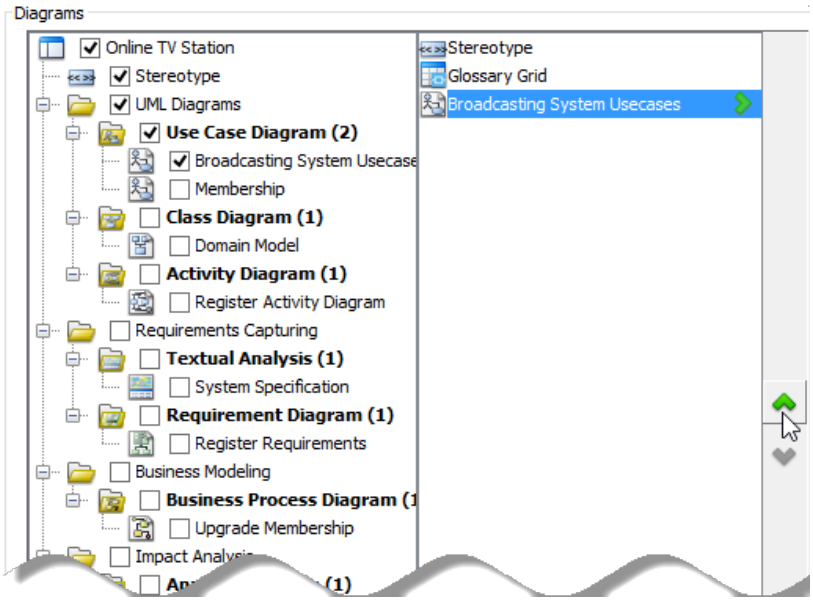
Relationships

Unamed Association		
To	Name	Value
	End Model Element	Compose Mail
	Author	Peter
	Create Date Time	Feb 9, 2010 8:30:27 AM
	Last Modified	Feb 9, 2010 8:38:14 AM

A generated document with style defined in template applied

Sorting diagrams in document

By default, diagrams show in document follows the order defined in diagram tree in the **Generate PDF** window. We may, however, sort the diagrams to make them appear in desired sequence. To sort diagram(s), select the diagram(s) to be ordered on the list at the center of window, and click or to sort.

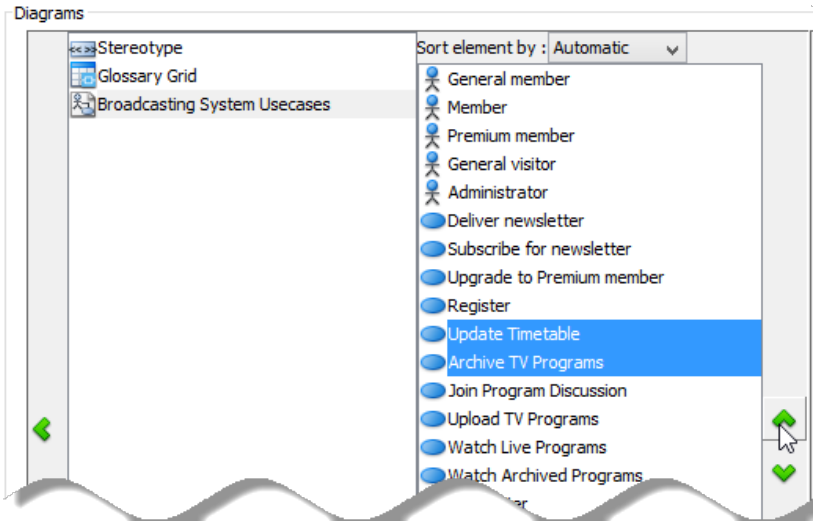


Reorder diagram

Sorting shapes in diagram

Custom sorting

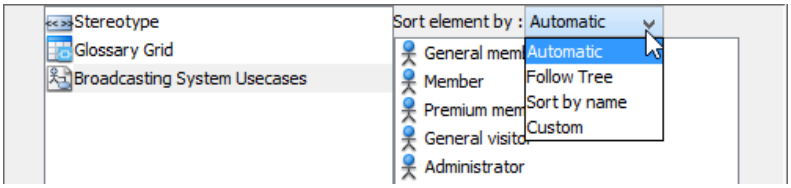
1. Select the diagram to sort and click to expand it.
2. Select the shape(s) to sort.
3. Click or to sort.



Sort shapes

Automatic sorting

1. Select the diagram to sort and click to expand it.
2. Select from the drop down menu **Sort element** a way to sort.



Select the way to sort shape

Way of sorting	Description
Automatic	The way of sorting elements is automatically managed. The order is often based on the flow and/or position of elements, which is the most logical order, following most users' understanding of that kind of diagram.
Sort by Tree	Sort diagram elements by following the order defined in Diagram Navigator.
Sort by Name	Sort diagram elements alphabetically base on their name, in ascending order.
Custom	The way of sorting elements is controlled by user, through selecting elements and pressing or .

Different ways of sorting

Related Resources

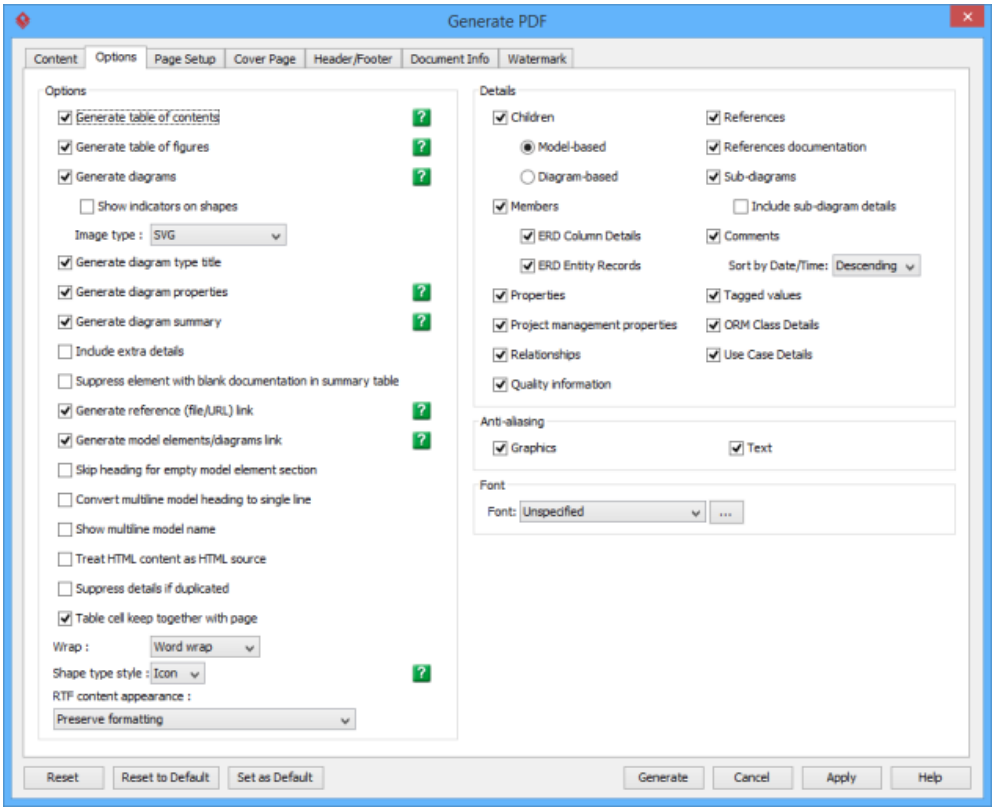
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Configuring document

Document generation can be configured to make the output closer to your expectation. Common configuration options include whether to generate table of contents/figure or not, whether to generate shape/diagram type as icon or text and whether to generate a particular type of detail such as children. Besides configuration options, you can also adjust the page setup, design the cover page and define header/footer. In this chapter, we will go through all the options one by one.

Options



Options

Option	Description
Generate table of contents	If this option is selected, table of content for this document will be generated to the document.
Generate table of figures	If this option is selected, table of figures for this document will be generated to the document.
Generate diagrams	If this option is selected, the image of the selected diagrams will be generated to the document. For PDF document, you can select the type of diagram. Here are the possible selections: • PNG - Images will look exactly the same as the diagrams in your project but not scalable against zooming. • SVG - Due to its scalable nature, image content will remain clear regardless of the level of zooming. However, image may look a bit different from the original diagram as there is a conversion re-construction from raster graphic data to SVG image. • SVG (text as shape) - Base on SVG, this option makes any text on diagram become text object, making it possible to select them in generation content.
Show indicators on shapes	If this option is sleected, resource indicators like references, sub-diagram will be shown in shape bodies.
Generate diagram type title	If this option is selected, the type of diagram (e.g. class diagram, use case diagram, etc) will be generated to the document, above the diagram name.
Generate diagram properties	If this option is selected, the properties of the selected diagrams will be generated to the document.

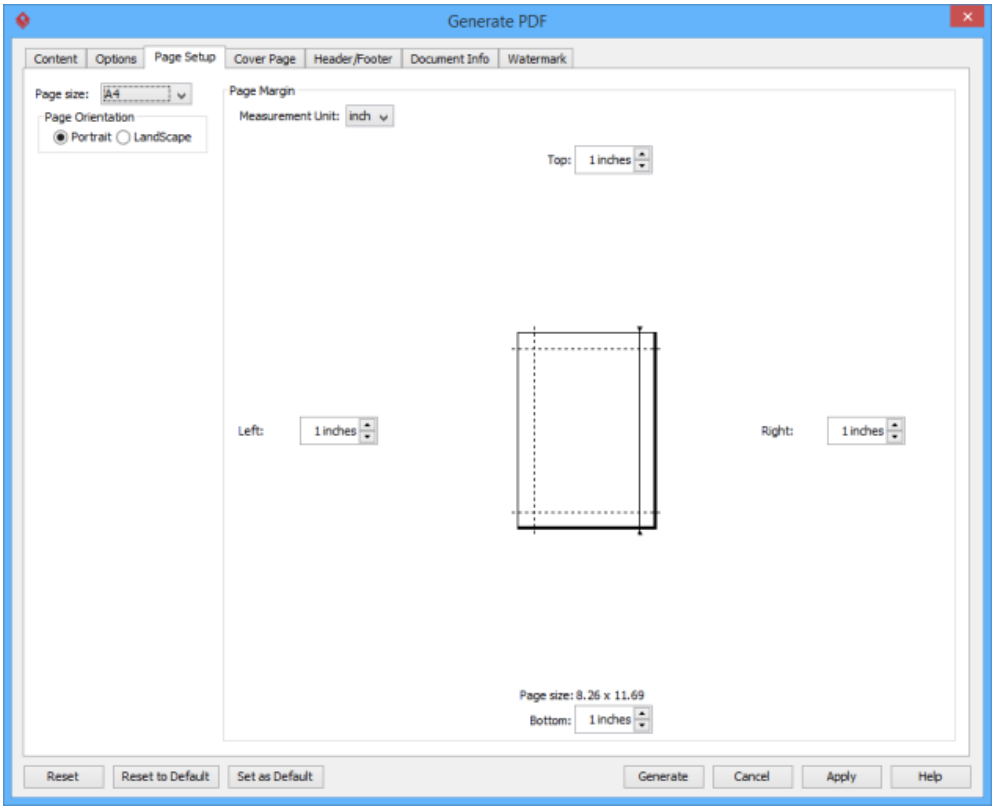
Generate diagram summary	If the option is selected, the summary of the selected diagrams will be generated to the document.
Include extra details	If the option is selected, information like ID and stereotypes will be generated to the summary table of document.
Suppress element with blank description in summary table	If the option is selected, diagram elements without description defined will not be generated to summary table.
Generate reference (file/URL) link	Select to generate links for referenced files/URLs defined in models.
Generate model elements/diagrams link	Select to generate links for navigating to related models and diagrams.
Skip heading for empty model element section	If this option is selected, heading for empty model element section will be skipped.
Convert multiline model heading to single line	If this option is selected, multiline model heading will be converted to single line.
Show multiline model name	If this option is selected, non heading multiline model name will remain in multiline, instead of being converted to single line.
Treat HTML content as HTML source	If this option is selected, HTML content will be treate as HTML source.
Suppress details if duplicated	If this option is selected, duplicated details will be suppressed.
Table cell keep together with page	If this option is selected, table cell will try to show on a page completely instead of breaking into separate pages. This option is only available to PDF document.
Wrap	Word wrap - Keep a complete word at the end of a line. Character - Split a word when reaching the end of a line, when needed.
Shape type style	Icon - using Icon to represent the type of shape and diagram elements Text - using text to represent the type of shape and diagram elements
RTF content appearance	Preserve formatting - using original formatting for RTF content Make font size consistent with the rest of the document - use the same font size for the RTF content in the whole document. Display in plain text - use plain text for RTF content
Copy reference files	If this option is selected, referenced files will be copied to output folder of document. With this option, you can copy the whole document folder to another machine and read there, without the need to break file linkage for references.
Details	Select a kind of content to generate it. Children - Everything a shape is containing. When selected, you can further select Model-based or Diagram-based for controlling the scope of children. Model-based consider all children the model of view contained. Diagram-based only consider the view in generating diagram. Let say if you have a package containing several classes. By selecting Model-based, all classes will be considered. By selecting Diagram-based, only the classes that are contained by the package in the generating diagram will be considered. Members - Attributes and operations are example of members. Properties - Name, description, abstract, leaf are example of properties. Project management properties - Author, create date, version are examples of project management properties. Relationships - Association, dependency are example of relationships Quality information - The quality of model elements - Bad, Fair, Good, etc. References - File, diagram, folder, URL, shape are possible kinds of reference References description - Determine whether to generate the referenced shape/diagram's description in reference table Sub-diagrams - Sub-diagrams of a shape Comments - Comments of shape

	Tagged values - Tagged values of shape ORM Class Details - ORM class details specialized for ORM Persistable class Use Case Details - Use case details of use case
Anti-aliasing	Determine the quality of document content. Graphics - To enable/disable the graphic anti-aliasing of the diagram images. Text - To enable/disable the text anti-aliasing of the diagram images.
Font	Determine the font family of document content. This option is only available to PDF document.
Encoding	Determine the encoding of HTML file to be generated. This option is only available to HTML document.

A description of general options

Page Setup

Page setup controls the layout of document. You can adjust a document size, page orientation and margin.



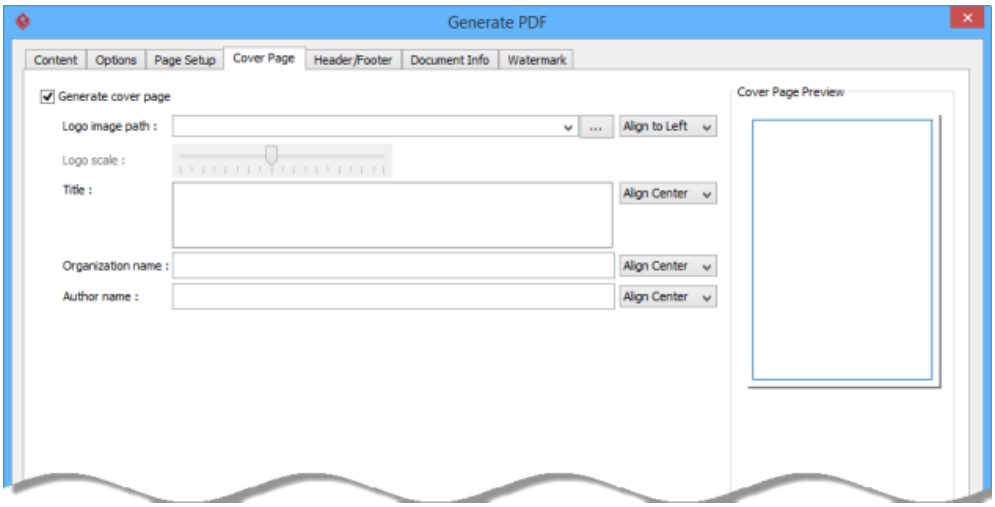
Page setup

Option	Description
Page size	To select the paper size of the generated document.
Page orientation	This option is used to select the orientation of the document (portrait/landscape). This option is only available to PDF and MS Word document.
Page margin	To specify the page margins of the document. This option is only available to PDF and MS Word document.

A description of options of page setup

Cover Page (Front Page for HTML document generation)

Cover page is the first page of document. You can add your company logo there and enter the document title, organization name and author name. Notice that in HTML document generation, the tab **Cover Page** is named as **Front Page**.



Cover page

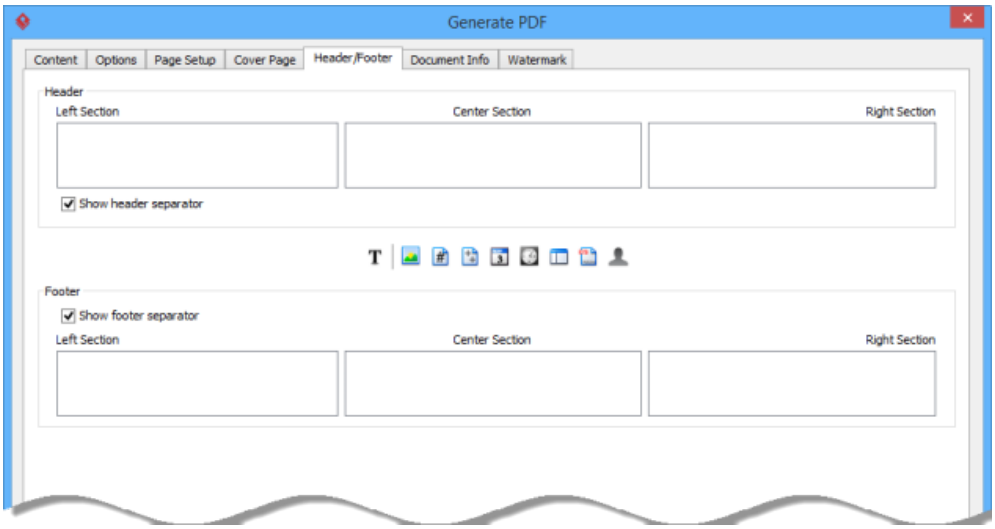
Option	Description
Generate cover page (PDF and MS Word) Generate front page (HTML)	If this option is selected, there will be a cover page generated to the d9ocument.
Logo image path	An image that appear at the document. You are expected to supply the file path of the image file. The drop down menu at the right hand side is for controlling the position of image.
Logo scale	Control the scale of logo image. This option is only available to PDF and MS Word document.
Title	The title text
Organization name	The organization name text
Author name	The author name text

A description of options of cover page

Header/Footer

Header and footer refers to the content that appears in the top and bottom of every page in document. For MS Word document, there are two text boxes for you to edit the header and footer. For PDF document, there are six boxes, three for each of the header and footer. Each of the text box represents a region in header/footer, such as the top left text box refers to the left region of header, while the bottom right text box refers to the right region of footer.

Instead of typing in the content of header/footer, there are a set of variables for you to apply with. The following table provides you with description with each of the variable.



Header/Footer

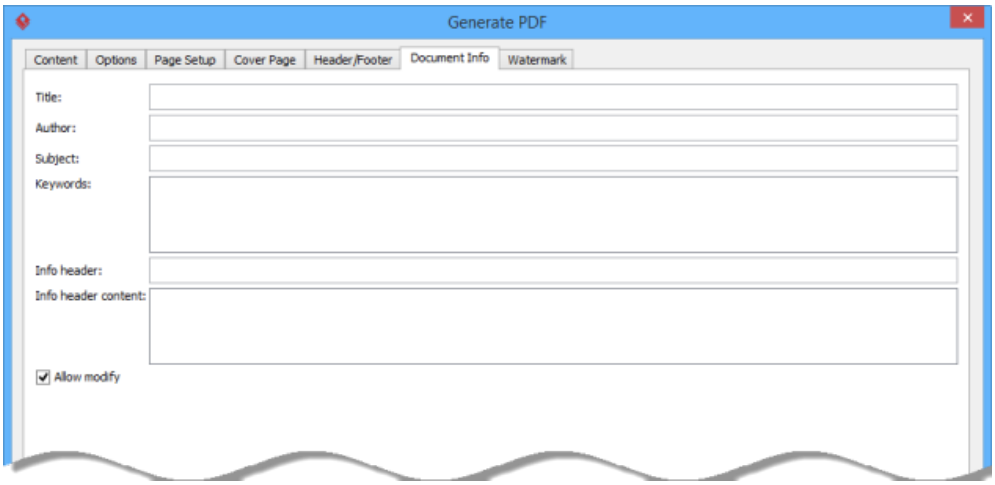
Variable	Name	Description
T	Selection font	Font settings of selected content

	Align to left	Align content to the left of header/footer. This option is available only to MS Word document.
	Align to center	Align content to the center of header/footer. This option is available only to MS Word document.
	Align to right	Align content to the right of header/footer. This option is available only to MS Word document.
	Add image	Insert an image to the position where the text cursor is placing.
	Insert page number	Insert page number to the position where the text cursor is placing.
	Insert page count	Insert page count to the position where the text cursor is placing.
	Insert date	Insert the date of when the document is generated to the position where the text cursor is placing.
	Insert time	Insert the time of when the document is generated to the position where the text cursor is placing.
	Insert project name	Insert the project name to the position where the text cursor is placing.
	Insert document file name	Insert the name of document file to the position where the text cursor is placing.
	Insert user name	Insert the name of the user logging into the system to the position where the text cursor is placing.

A description of variables that can be used in header and footer

Document Info

For HTML document, document info refers to the meta information of HTML document. For PDF and MS Word document, document info refers to the possible document properties that can be defined.



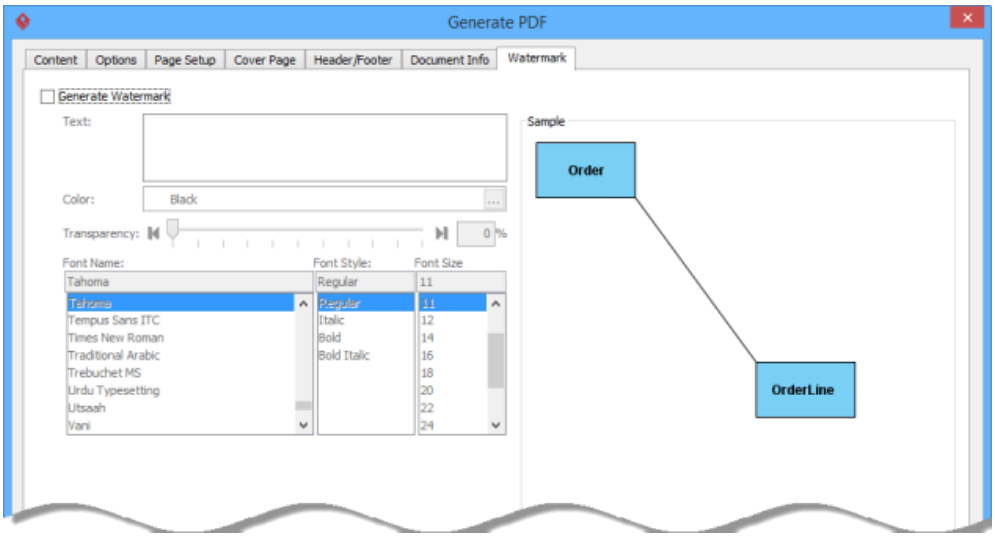
Document info

Option	Description
Title	The title of document.
Author	The author of the document. This option is only available to PDF document.
Subject	The subject of the document.
Keywords	The keywords of the document.
Info header	The info header of the document. This option is only available to PDF document.
Info header content	The info header content of the document. This option is only available to PDF document.
Allow modify	Select to allow modification on the document. This option is only available to PDF document.

A description of document info

Watermark

Print watermark to images in document. This is particularly useful when you want to specify the state of design. E.g. "Reviewed"



Watermark

Option	Description
Generate Watermark	Check to generate watermark to images in document.
Text	The text to appear as watermark.
Color	The color of text.
Transparency	The level of transparency of watermark (text) in document.
Font name	The font of watermark text.
Font style	The font style of watermark text.
Font size	The font size of watermark text.

A description of watermark

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Customizing report

Instead of generating a report in a way that VP-UML defined for you, you can develop report templates to customize the report content, to make output match your needs. In this chapter, you will see how to make use of the template editor to customize a report template.

Customizing report

Shows you how to launch the template editor to perform basic editing. It also describes the editor window in brief.

Export/import report template

Shows you how to export a template to a template file and import it into another instance.

Diagram loop

Define diagram loop and show you how to use a diagram loop in practice.

Diagram summary

Define diagram summary and show you how to use a diagram summary in practice.

Diagram paragraph

Define diagram paragraph and show you how to use a diagram paragraph in practice.

Element loop

Define element loop and show you how to use a element loop in practice.

Element summary

Define element summary and show you how to use a element summary in practice.

Element paragraph

Define element paragraph and show you how to use a element paragraph in practice.

Custom content

Define custom content and show you how to make use of it to enter formatted free text.

Diagram image

Define diagram image and show you how to make use of it to add an image of diagrams.

Property value

Define property value and show you how to make use it to extract a property from a parent model element.

Page break

Shows you how to insert a page break.

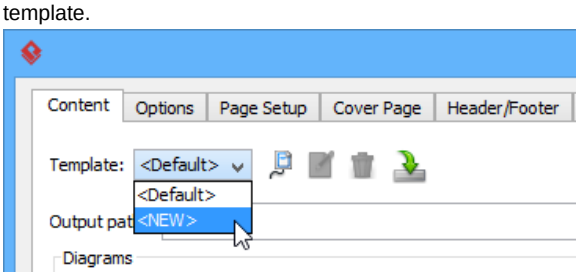
Customizing document

Instead of generating a document in a way that Visual Paradigm defined for you, you can develop document templates to customize the document content to make the output matches with your needs. With document customization, you can select model elements and properties to generate to document. You also can add custom text content. To customize document:

1. Select **Tools > Doc** from the toolbar. Then select **Generate HTML Doc...**, **Generate PDF Doc...** or **Generate Word Doc...** depending on the type of document you want to generate.

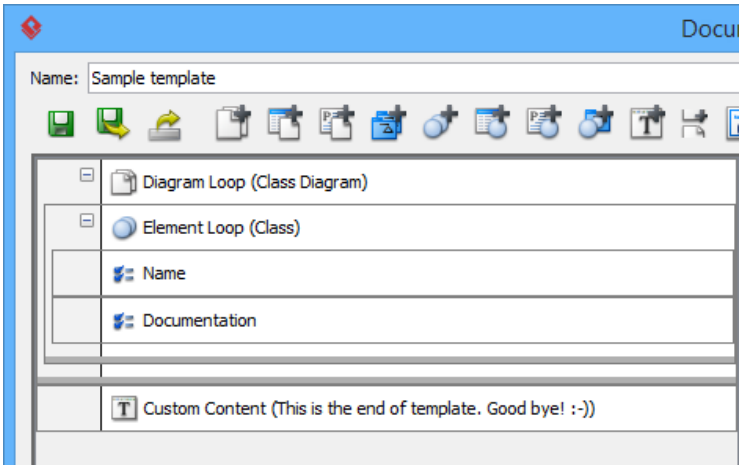
> **NOTE:** document templates are shared among HTML, PDF and Word documents. If you just want to design template but have no preference on the type of document to generate yet, just select either HTML, PDF or Word.

2. Press on the **Template** drop down menu and select **<New>** from the popup menu to open the document template editor and start editing template.



To create a template

3. In the **document Template** window, enter the template name at the top of window.
4. Construct the document template. A document template is formed by different kinds of components that put together in a hierarchical structure. A common way of building a template is to start with a diagram loop, which can be created from the editor toolbar. Then, select the diagram loop and create children components through its toolbar, such as to create an image component or a element loop for accessing diagram elements on diagrams being looped. To learn how to use the tools in detail, refer to the coming chapters.



A sample template showing the use of diagram loop, element loop, property and custom text components

5. Click **Save** to save your work.
6. Click **Close**.

An overview of tools in template editor

Tool	Name	Description
	Save	Save the opening template.
	Save as	Save the opening template as a new one.
	Export	To export the opening template to a document template file (.vpr). You can import the file to other machines for reusing it.
	Add diagram loop	To add a component to template editor, indicating the need of looping specific type(s) of diagram.
	Add diagram summary	To add a component to template editor, indicating the need of looping specific type(s) of diagram for constructing a tabular diagram summary.
	Add diagram paragraph	To add a component to template editor, indicating the need of looping specific type(s) of diagram for printing its properties in paragraph form.

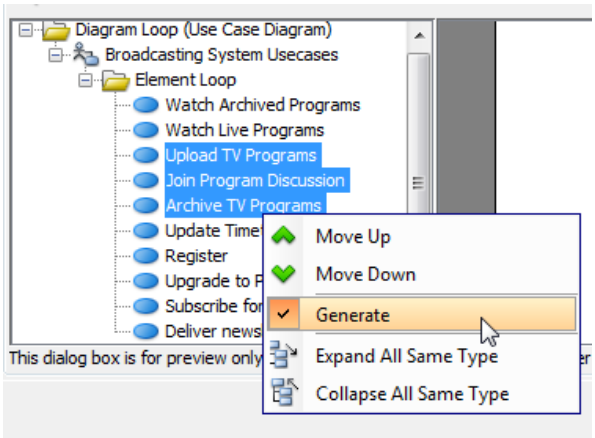
Add element loop (Model)	To add a component to template editor, indicating the need of looping Model.
Add root level element loop	To add a component to template editor, indicating the need of looping specific type(s) of model element that are at project root (i.e. not being contained).
Add element summary	To add a component to template editor, indicating the need of looping specific type(s) of model element for constructing a tabular element summary.
Add element paragraph	To add a component to template editor, indicating the need of looping specific type(s) of model element for printing its properties in paragraph form.
Add all level element loop	To add a component to template editor, indicating the need of looping model elements of specific type(s) in whole project.
Add custom content	To add a component to template editor, indicating the placement of text written by user.
Preview template content	Preview the template against the project content for document content. Rendering of document is costly especially when the project is large. If your project is large, be patient when waiting for outcome.
Preview template structure	Preview the template against the project content for document structure.
Options	Configure document options.

Description of different tools in template editor

Ignoring specific diagrams/shapes

A document template is independent of any project file. But if you try to apply a template on a project, you can ignore generating specific diagrams or shapes in that project. To ignore specific diagrams/shapes:

1. Open the template in template editor.
2. Preview the document by pressing or from the toolbar.
3. In the document structure tree, select the diagrams or shapes that you want to ignore in generated document.
4. Right click on your selection and de-select **Generate** from the popup menu.



To not generate model elements

More about Preview

When you try to preview a template, it tries to apply the template on the opening project to form the document structure and to render the preview. Due to the connection between template and project, there are several actions that you can perform with the preview.

Ignoring specific diagrams/shapes

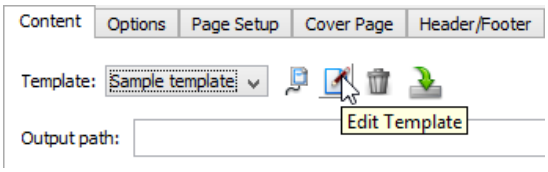
In the document structure tree, right click on the diagram(s) or shape(s) that you want to ignore when generating document and de-select **Generate** from the popup menu.

Reordering diagrams/shapes

In the document structure tree, right click on the diagram(s) or shape(s) that you want to re-order and select **Move Up** or **Move Down** to reposition them.

To edit a template

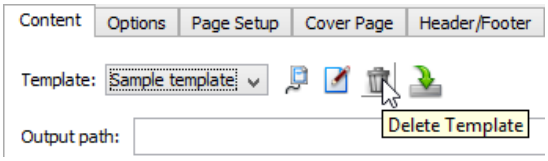
To edit an existing document template, select it in the generate window such as **Generate PDF**, then click on the edit button.



To edit a template

To delete a template

To delete an existing document template, select it in the generate window such as **Generate PDF**, then click on the delete button.



To delete a template

Related Resources

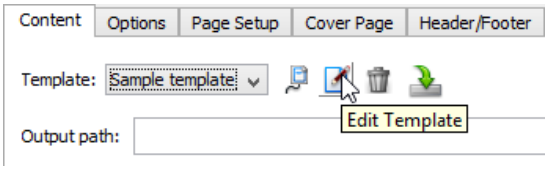
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export/import document template

You can apply a document template on another project by exporting it as a document template file and importing it to the target project. To export and import template:

1. Open the document template in template editor.



To open a template in template editor

2. Click on the  button at the top of template editor to export template.
3. In the **Export** dialog box, enter the file name and click **Save**.
4. Open the project that you want the template to import to. Open the document dialog box by selecting **Tools > Doc > Generate HTML Doc../ Generate PDF Doc.../ Generate Word Doc...** from the toolbar.
5. Click on the  button.
6. In the **Import** dialog box, select the document template file and click **Open**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

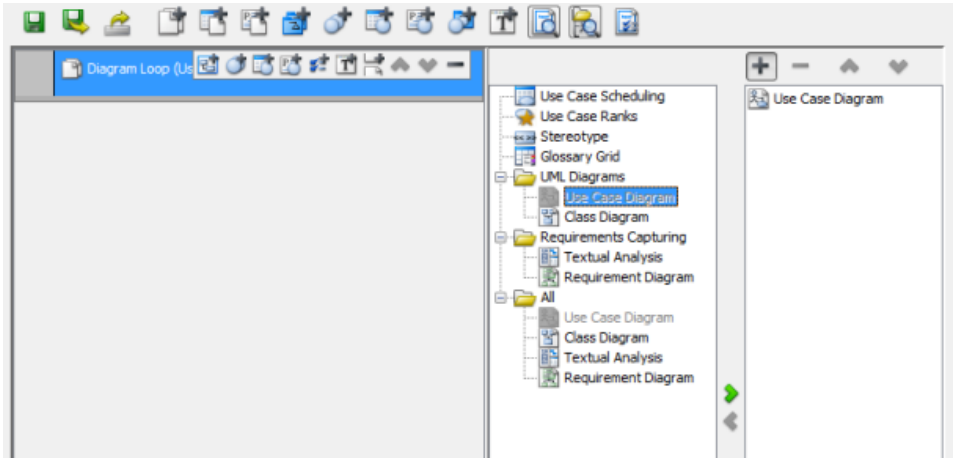
Diagram loop

A diagram loop is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram. For example, to loop all use case diagrams and class diagrams in a project. Diagram loop means nothing more than just to loop diagram. The content to print for each diagram being looped is determined by the children components of loop.

Looping diagrams in project

To loop specific type(s) of diagrams in project, create diagram loop at template root. To create diagram loop at template root:

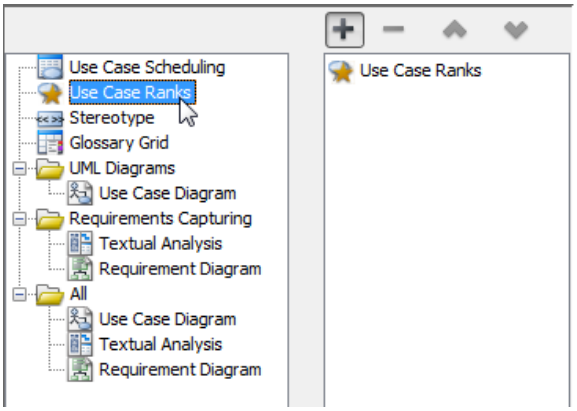
1. In the template editor, click on the  on toolbar.
2. On the right hand side of the template editor, select the type of diagram(s) to be looped and click  to confirm the selection.



Select to loop use case diagram

Including use case scheduling,ranks, stereotypes and various grid in document

Use case scheduling, use case ranks, stereotypes and various grid fall into the category of diagram. If you want to print them to document, follow the steps as described in the previous section and select the appropriate content to include at the final step.

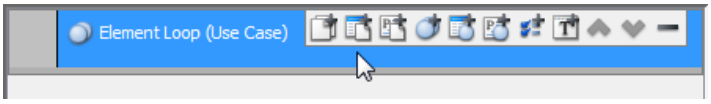


Choosing to include information of use case ranks in template

Looping sub-diagrams of specific element

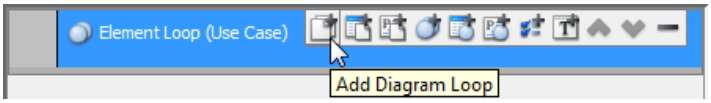
Instead of looping diagrams in a project, you can also place diagram loop under an element loop to loop for sub-diagrams of specific type(s) of model element. To create diagram loop under an element loop:

1. Select the element loop that you want to loop for its sub-diagrams.



Selecting an element loop

2. Click on the **Add Diagram Loop** button from the toolbar of element loop.



Adding a diagram loop

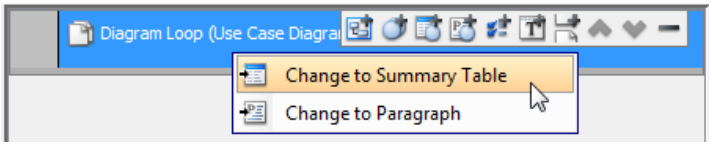
NOTE: Make sure you are clicking on the button from the toolbar of element loop. If you are clicking on the button at editor toolbar, you will create a loop at template root.

3. On the right hand side of the template editor, select the type of sub-diagram(s) to loop and click to confirm the selection.

Switching from diagram loop to diagram summary table/paragraph

A diagram summary indicates the need of looping specific type(s) of diagram for constructing a tabular diagram summary, while a diagram paragraph indicates the need of looping specific type(s) of diagram for constructing paragraphs of properties.

You can convert a diagram loop to diagram summary table or paragraph by right clicking on a diagram loop and selecting **Change to Summary Table/ Paragraph** from popup menu.



To convert a diagram loop to diagram summary table/paragraph

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram summary

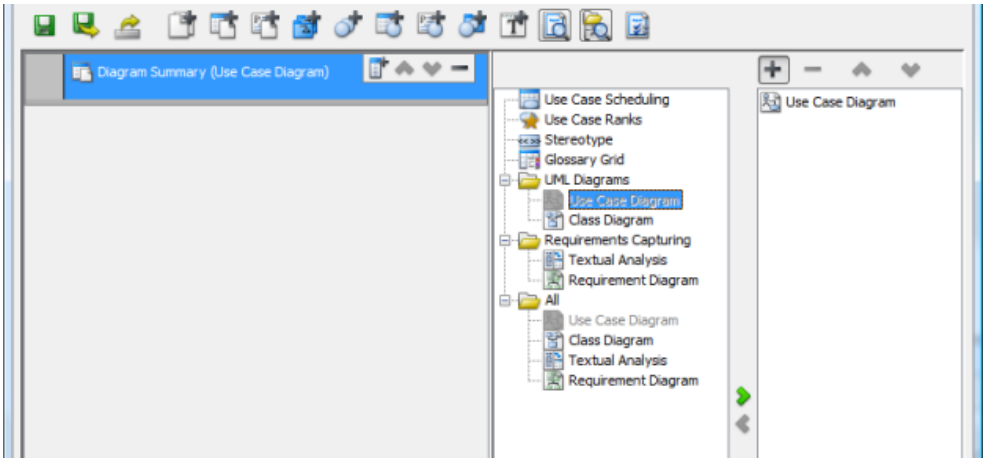
A diagram summary is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram, and presenting their properties in tabular form. For example, to loop all use case diagrams in a project and form a table consisting of diagram names and description. By default, a table of diagram names will be printed. You are expected to add property column(s) under a diagram summary to indicate the diagram properties to print in the table.

Showing a property table for diagrams in project

To loop specific type(s) of diagrams in project for listing their properties in a table, create diagram summary at template root. To create diagram summary at template root:

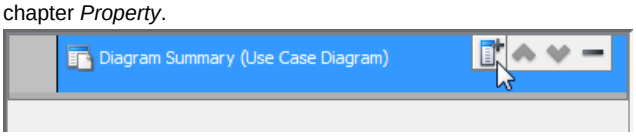
1. In the template editor, click on the  on toolbar.

2. On the right hand side of the template editor, select the type of diagram(s) to be looped and click  to confirm the selection.



Select to loop use case diagram

3. If you generate document with a template like this, you will obtain a table of diagram names, provided that the diagram(s) of the chosen type(s) exists. If you need to print specific diagram properties in table other than just name, click on the **Add Property Column** button in the toolbar of diagram summary, then select the property(ies) to be added into the table as column(s). For more details about the use of property, read the chapter *Property*.

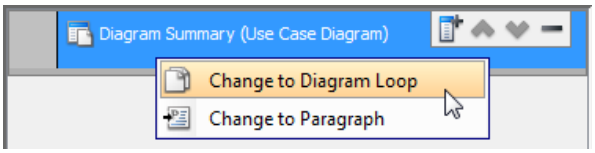


To add a property column in diagram summary

Switching from diagram summary to diagram loop/paragraph

A diagram loop indicates the need of looping specific type(s) of diagram, while a diagram paragraph indicates the need of looping specific type(s) of diagram for constructing paragraphs of properties.

You can convert a diagram summary to diagram loop or paragraph by right clicking on a diagram summary and selecting **Change to Diagram Loop/ Paragraph** from pop-up menu.



To convert a diagram summary to diagram loop/paragraph

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

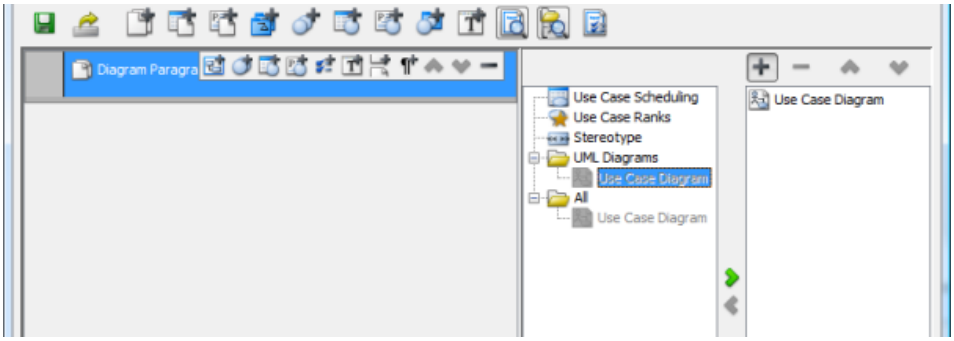
Diagram paragraph

A diagram paragraph is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram and presenting their properties in paragraph form. For example, to loop all use case diagrams in a project and print out their description and last modified data in a paragraph, one paragraph per diagram. You are expected to add property value(s) under a diagram paragraph to indicate the diagram properties to print out.

Showing a property paragraph for diagrams in project

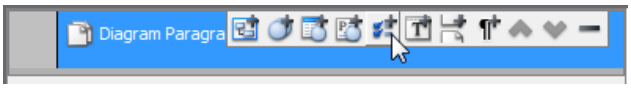
To loop specific type(s) of diagrams in project for listing their properties in paragraph form, create diagram paragraph at template root. To create diagram paragraph at template root:

1. In the template editor, click on the  on toolbar.
2. On the right hand side of the template editor, select the type of diagram(s) to be looped and click  to confirm the selection.



Select to loop use case diagram

3. If you generate document with a template like this, you will obtain a list of diagram types and names, provided that there exists diagram(s) of the chosen type(s). If you need to print specific diagram properties other than just name, click on the **Add Property Value** button in the toolbar of diagram paragraph, then select the property(ies) to show. For more details about the use of property, read the chapter *Property*.

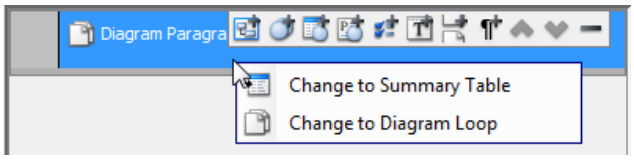


To add a property value in diagram paragraph

Switching from diagram paragraph to diagram summary table/loop

A diagram summary indicates the need of looping specific type(s) of diagram for constructing a tabular diagram summary, while a diagram loop indicates the need of looping specific type(s) of diagram.

You can convert a diagram paragraph to diagram summary table or loop by right clicking on a diagram paragraph and selecting **Change to Summary Table/Diagram Loop** from popup menu.



To convert a diagram paragraph to diagram summary table/loop

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Element loop

An element loop is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram/model elements. For example, to loop all use case and class in a project. Element loop means nothing more than just to loop diagram/model element. The content to print for each diagram/model element being looped is to be determined by the children components of loop.

Looping model elements in project

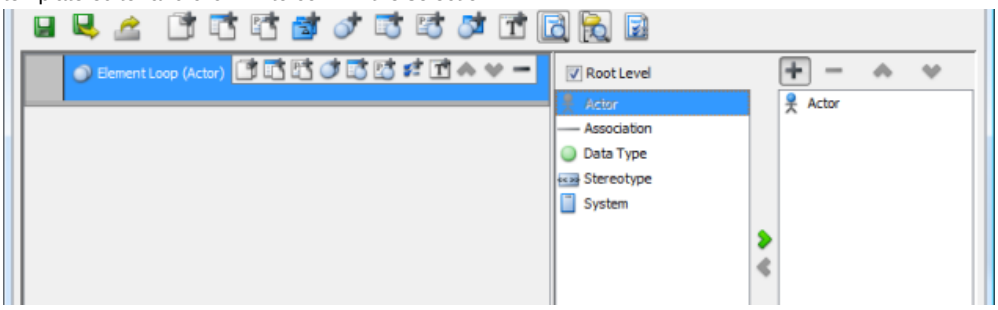
To loop specific type(s) of model elements in project, create element loop at template root. To create element loop at template root:

1. In the template editor, click on any of the buttons in the toolbar depending on your need.

Button	Name	Description
	Add Element Loop (Model)	To loop through all models in the project root. In other words, model being contained by other model element will not be accessed. This is a shortcut for creating a root level element loop whose chosen element type is model.
	Add Element Loop (Package)	To loop through all packages in the project root. In other words, package being contained by other model element will not be accessed. This is a shortcut for creating a root level element loop whose chosen element type is package.
	Add Root Level Element Loop	To loop through any kind of model element in project root. By selecting this option, you can choose the type of model element to be looped.
	Add All Level Element Loop	To loop through any kind of model element within the project, regardless of their leveling. By selecting this option, you can choose the type of model element to be looped.

Description of available type of element loop

2. If you have chosen to add a model or a package loop, you do not need to perform any actions in further.
If you have chosen to add either root level or all level element loop, select the type of element(s) to be looped on the right hand side of the template editor and click to confirm the selection.



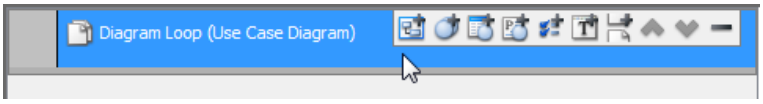
Select to loop actor

> **NOTE:** You can switch between a root level and a all-level loop by changing the option **Root Level** above the element list. Notice that changing the value of **Root Level** is not about changing available element selection, but also the end result, whether to access only root level elements or not.

Looping diagram elements in a diagram

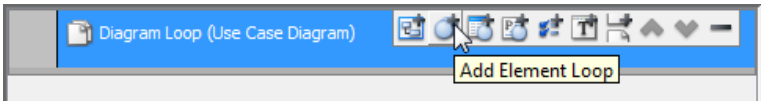
Instead of looping model elements in a project, you can also place element loop under a diagram loop to loop for diagram elements in specific type(s) of diagram. To create element loop under a diagram loop:

1. Select the element loop that you want to loop for its sub-diagrams.



Selecting a diagram loop

2. Click on the **Add Element Loop** button from the toolbar of element loop.



Adding an element loop

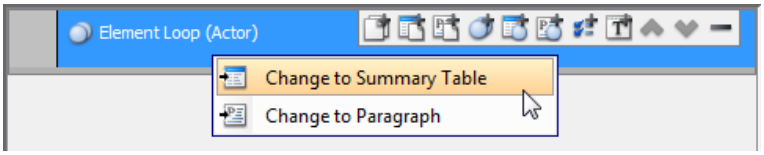
> **NOTE:** Make sure you are clicking on the button from the toolbar of diagram loop. If you are clicking on the button at editor toolbar, you will create a loop at template root.

3. On the right hand side of the template editor, select the type of diagram element to be looped and click to confirm the selection. If you want to skip those auxiliary views on the diagram, check Master View Only on top of the list of diagram elements.

Switching from element loop to element summary table/paragraph

An element summary indicates the need of looping (i.e. walking through) specific type(s) of diagram/model element and present their properties in tabular form, while an element paragraph indicates the need of looping specific type(s) of diagram/model element for constructing paragraphs of properties.

You can convert an element loop to element summary table or paragraph by right clicking on a element loop and selecting **Change to Summary Table/Paragraph** from popup menu.



To convert an element loop to element summary table/paragraph

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Element summary

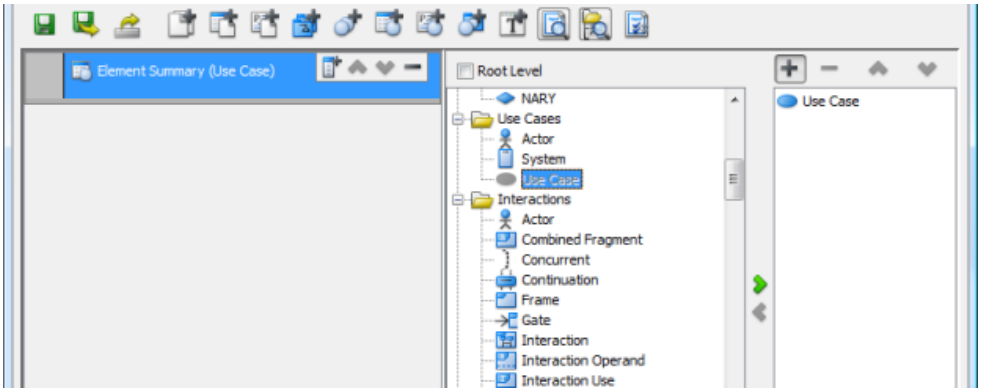
An element summary is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram/model element and presenting their properties in tabular form. For example, to loop all use cases in a project and form a table consisting of use case IDs and documentation. By default, a table of element names will be printed. You are expected to add property column(s) under an element summary to indicate the element properties to print in the table.

Showing a property table for elements in project

To loop specific type(s) of model elements in project for listing their properties in a table, create an element summary at template root. To create an element summary at template root:

1. In the template editor, click on the  on toolbar.

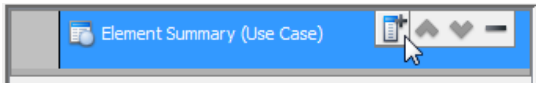
2. On the right hand side of the template editor, select the type of model element(s) to loop and click  to confirm the selection.



Select to loop use case

> **NOTE:** By default, element summary added to project root enables the looping of root level elements.
> If you want to change to access elements in all levels, change the option **Root Level** above the element list.
> Notice that changing the value of **Root Level** is not about changing available element selection but also the end result, whether to access only root level elements or not.

3. If you generate document with a template like this, you will obtain a table of element names, provided that the element(s) of the chosen type(s) exit(s). If you need to print specific element properties in table other than just name, click on the **Add Property Column** button in the toolbar of element summary, then select the property(ies) to be added into the table as column(s). For more details about the use of property, read the chapter *Property*.

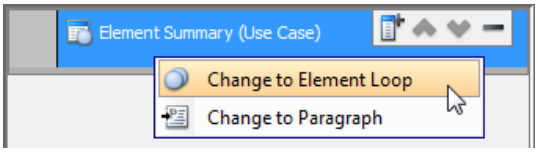


To add a property column in element summary

Switching from element summary to element loop/paragraph

An element loop indicates the need of looping specific type(s) of model/diagram element, while an element paragraph indicates the need of looping specific type(s) of diagram/model element for constructing paragraphs of properties.

You can convert an element summary to element loop/paragraph by right clicking on an element summary and selecting **Change to Element Loop/ Paragraph** from popup menu.



To convert a element summary to element loop/paragraph

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Element paragraph

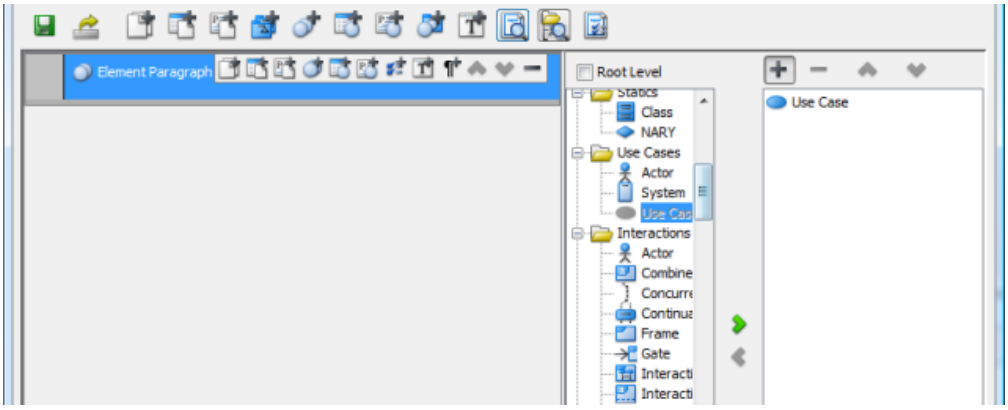
An element paragraph is a component in a document template, indicating the need of looping (i.e. walking through) specific type(s) of diagram/model element and presenting their properties in paragraph form. For example, to loop all use cases in a project and form paragraphs consisting of use case IDs and description. By default, a list of element names will be printed. You are expected to add property value(s) under an element paragraph to indicate the element properties to print out.

Showing a property paragraph for elements in project

To loop specific type(s) of model elements in project for listing their properties in paragraph form, create element paragraph at template root. To create element paragraph at template root:

1. In the template editor, click on the  on toolbar.

2. On the right hand side of the template editor, select the type of model element(s) to loop and click  to confirm the selection.



Select to loop use case

> **NOTE:** By default, element paragraph added to project root enables the looping of root level elements.
> If you want to change to access elements in all levels, change the option **Root Level** above the element list.
> Notice that changing the value of **Root Level** is not about changing available element selection but also the end result, whether to access only root level elements or not.

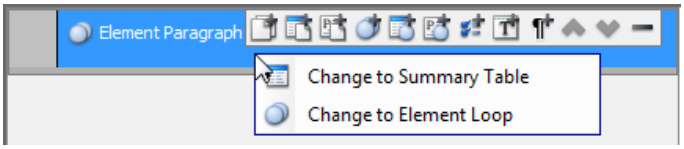
3. If you generate document with a template like this, you will obtain a list of element names, provided that there exists element(s) of the chosen type(s). If you need to print specific element properties other than just name, click on the **Add Property Column** button in the toolbar of element paragraph, then select the property(ies) to show. For more details about the use of property, read the chapter *Property*.

To add a property value in element paragraph

Switching from element paragraph to element summary table/loop

An element summary indicates the need of looping specific type(s) of diagram/model element for constructing a tabular element summary, while an element loop indicates the need of looping specific type(s) of diagram/model element.

You can convert an element paragraph to element summary table or loop by right clicking on a element paragraph and selecting **Change to Summary Table/Element Loop** from popup menu.



To convert a element paragraph to element summary table/loop

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

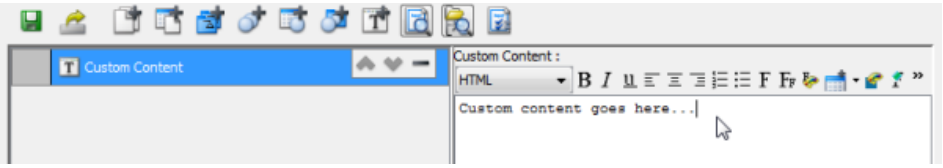
Custom content

Custom content is a component in a document templat which acts as a placeholder of user written text. For example, to add a section of acknowledgment at the beginning of document with custom text. You can write custom content in rich text and you can add custom content to template root or under a diagram/element loop.

Adding custom content at template root

To add custom content at template root:

1. In the template editor, click on  on toolbar.
2. On the right hand side of the template editor, enter the custom content. You can format your content through the buttons in the content pane. For details, read the section below about editing custom content.



Entering custom content

Adding custom content into a loop

To add custom content into a diagram/element loop:

1. Select the loop that you want to add custom content under it.

Selecting an element loop

2. Click on the **Add Custom Content** button from the toolbar of loop.

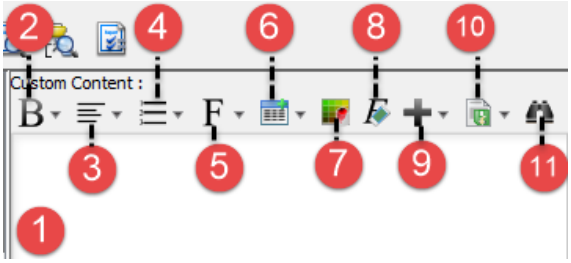
Adding custom content

> **NOTE:** Make sure you are clicking on the button from the toolbar of element loop. If you are clicking on the button at editor toolbar, you will create a loop at template root.

3. On the right hand side of the template editor, enter the custom content. You can format your content through the buttons in the content pane. For details, please read the section below about editing custom content.

Editing custom content

You can write plain text in custom content as well as to add formatted text, images and tables through the help of the tools in the editor.



Custom content editor

No.	Name	Description
1	Editor pane	The editor where you can enter and edit custom content.
2	Bold	Set the highlighted text to bold.
	Italic	Set the highlighted text to italic.
	Underline	Underline the highlighted text
3	Alignments	Set the alignment of highlighted text to right, center or left.
4	Ordered list	Add a numbered list.

	Un-ordered list	Add a list with bullet points.
5	Font	Select the font family of highlighted text.
	Font size	Select the size of highlighted text.
	Font color	Select the color of highlighted text.
6	Table	Add a table.
7	Background color	Select the background color of highlighted text.
8	Clear formats	Clear formats of whole editor to convert the content to plain text.
9	Add Link	Add a hyperlink.
	Add Image	Add an image.
	Add Model Element...	Add an model element
	Add Diagram...	Add a Diagram
10	Save as template...	
	Manage template...	
11	Find	

A description of custom content editor

Related Resources

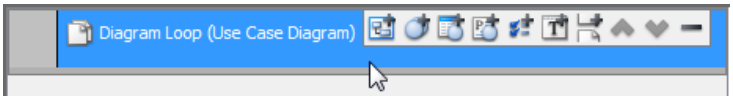
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagram image

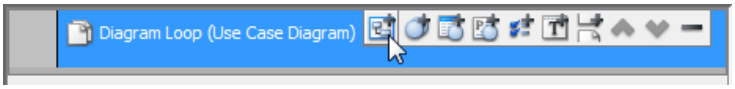
Diagram image is a component in a document template which represents a placeholder of the image of a diagram under a diagram loop. You must place a diagram image under diagram loop. To add a diagram image:

1. Select the diagram loop that you want the images of diagrams to be printed.



Selecting a diagram loop

2. Click on the **Diagram Image** button from the toolbar of loop.



Adding a diagram image

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

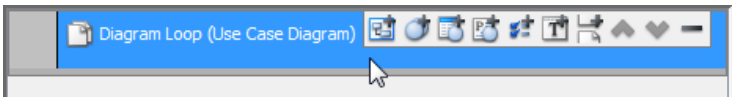
Property value

Property value is an element component in a document template, which represents the access of certain property of a diagram, model element or diagram element. For example, to print out the ID (property) of a use case. You can add property value into a diagram loop, a diagram summary, an element loop, an element summary. Property value added to a summary component will become a property column in summary table.

Adding property value into a loop or summary

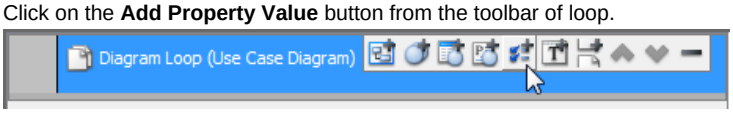
To add property value into a loop or summary:

1. Select the loop or summary that you want to add property value under it.



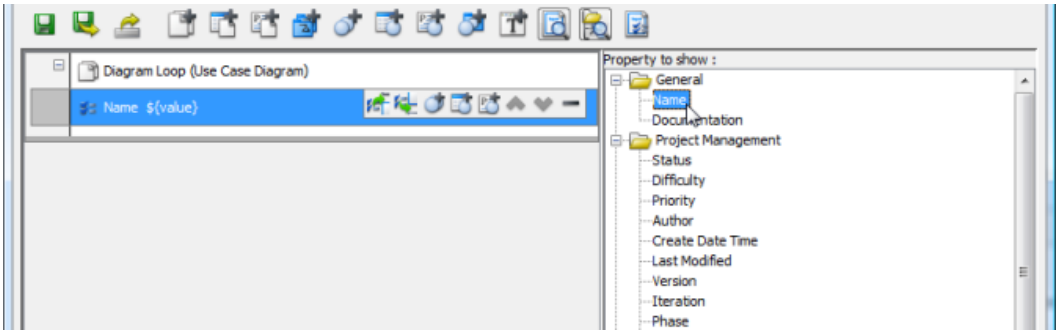
Selecting a diagram loop

2. Click on the **Add Property Value** button from the toolbar of loop.



Adding property value

3. On the right hand side of the template editor, select the property to access.

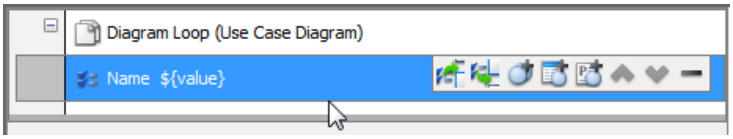


Selecting a property to access

Adding a property below another property

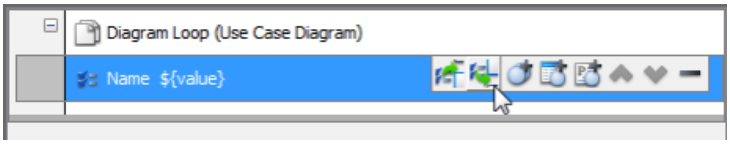
To add a property below another property:

1. Select an existing property value.



Selecting a property value

2. Click on the **Add Property Value Below** or **Add Property Value Above** button.



To add a property value below an existing one

3. On the right hand side of the template editor, select the property to access.

Related Resources

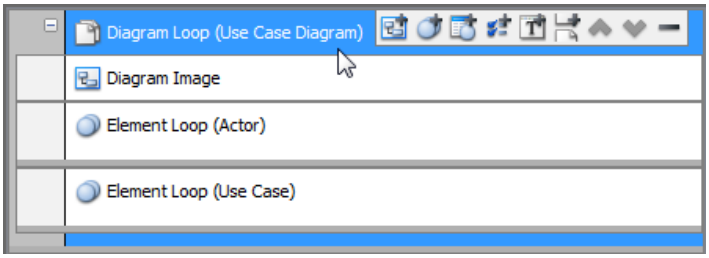
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Page break

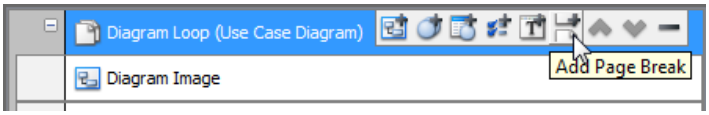
Page Break is where the document should start the contents which follows in a new page. It can be placed within a loop, either a diagram or element loop. To create a Page Break:

1. Select the loop that you want to insert a page break.



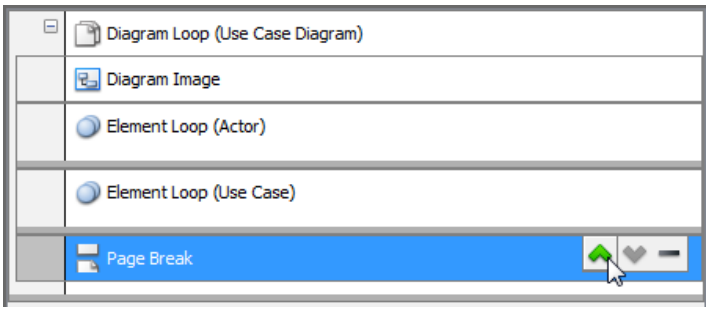
Selecting the loop for adding a page break

2. Click on the **Add Page Break** button from the toolbar of loop.



Adding a page break

3. Reposition the page break by pressing **Move Up** or **Move Down** from the toolbar of page break.



Repositioning a page break

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Publishing project to web site

The Project Publisher is a tool that exports the project, including detailed information in diagrams and models, into interactive and well-organized web pages. This chapter shows you how to publish a project.

Publish project using project publisher

Gives a brief description of publisher dialog box and guides you through the steps of publishing.

About publisher output

Describes the interactive features supported in published content.

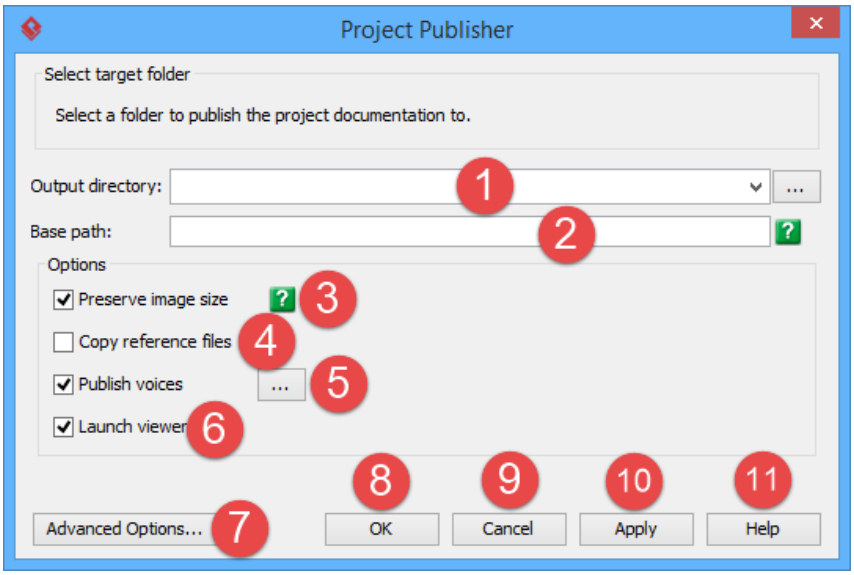
Publish project using Project Publisher

The Project Publisher is a tool that exports the project, including detailed information in diagrams and models into interactive and well-organized web pages. The generated web pages can be read in any web browser with no additional plug-in required, so collaborative partners may see the published product even if they do not have Visual Paradigm products installed.

To launch Project Publisher:

1. Select **Tools > Publish Project...** from toolbar.
2. In the **Project Publisher** window, specify the output directory where you want to save the published content.
3. You can optionally configure the publisher by adjusting the options or options. For details, refer to the sections below.
4. Click **OK** button to start publishing.

An overview of project publisher



Overview of project publisher

No.	Name	Description
1	Output directory	The folder where you want to publish the content to.
2	Base path	If a base path is specified and then published files are moved to the base path, diagram and model element can be accessed directly with a dedicated link.
3	Preserve image size	When checked, published content will show images in exact width and height.
4	Copy reference files	You can add file references to model elements. When this option is checked, referenced files will be copied to the output directory, so that you can access for any referenced file when browsing the published content in other machine easily.
5	Publish voices	When checked, audio clips added to the description will be published and can be opened when browsing the published outcome.
6	Launch viewer	When checked, the system will launch the web browser and open the published Web contents.
7	Advanced options	Click to configure advanced publisher options. For details about the options, read the next section.
8	OK	Click to publish.
9	Cancel	Click to close the dialog box without publishing.
10	Apply	Click to save the changes of settings.
11	Help	Click to open Help contents.

Description of project publisher

Advanced options

On the **Project Publisher** dialog box, you can configure some of the common options. You can also configure the advanced options for more detailed settings by clicking the **Advanced Options...** button.

Option	Description
Generate model element list in diagram page	Check to generate a list of model element in a diagram page.

Generate only description in model element page	Check to generate only description in model element page and exclude other contents.
Generate only when description is defined	Generate element page only when that element has description. When this option is off, the shape will have no linkage from image in diagram page due to the absent of element page.
Generate page header	Check to generate the pre-defined header.
Generate page footer	Check to generate the pre-defined footer.
Show description when hover over a shape	Check to show element's description when moving the mouse pointer over a shape in an image of diagram. **Show procedure for BP task and sub-process** - Popup the working procedures when you move the mouse pointer over BPMN tasks and sub-processes in an image. **Show test plan for test case** - Popup the test plan when you move the mouse pointer over a test case in an image.
Generate diagram type	Check to generate the diagram type in addition to diagram name.
Remove paragraph's top and bottom margin in RTF description	By default, top and bottom margins are added above and below RTF description text, due to the default style applied to the RTF description. If you want to remove the margins, you can override the setting by unchecking this option.
Generate referenced project diagrams	Check to include contents for diagrams in referenced project.
Generate quality information	Quality of model is assessed during the modeling. If you want to see the comment for each of your model elements, check this option to include in document.
Overwrite document style sheets	Project publishing is an "overwrite" action. If you publish to the same folder twice, files produced the first time would be overwritten. This option enables you to keep the style sheets file (.css) without being overwritten. This enables you to edit the styles defined and re-use it in subsequent publishing.
Generate grid configuration	Check to include configuration for grids, such as the type of elements to list and the scope (e.g. project/model/diagram, etc)
Generate menu	By default a diagram and model menus are shown on the left hand side of the published outcome. You can decide whether or not to generate the menus or produce two index files, one with menu and another one without.
Always show indicators	Indicators refer to the small icons that show over shape(s) in image(s). They appear to reflect different status of the view or model element - Is description entered? Is it a master or auxiliary view? Is sub-process/reference added? Is it a referenced element?
Drill down effect for general models	Choose the action when pressing on model elements on a diagram.
Drill down effect for business sub-process	Choose the action when pressing on sub-processes on a diagram.
Drill down effect for process (Process Map)	Choose the action when pressing on processes on a process map diagram.
Drill down effect for events (BPMN)	Choose the action when pressing on (BPMN) events on a diagram.
Drill down effect for diagram overview	Choose the action when pressing on diagram overviews on a diagram.
Drill down effect for action	Choose the action when pressing on (activity diagram) action on a diagram.
Drill down effect for UI elements	Choose the action when pressing on UI elements (e.g. Frame, Panel, Button) on a diagram.
Publisher engine	Choose the engine for publishing. You are advised to use the new engine.

Default diagram	Choose the diagram that first appear when the published content is opened. If unspecified, a default page with project information will be presented.
Sort elements in type groups by	Choose the way of sorting elements to be shown in summary or drop down menu in diagram page
Filtered content	Choose the content for not to show in published content

Description of project publisher advanced options

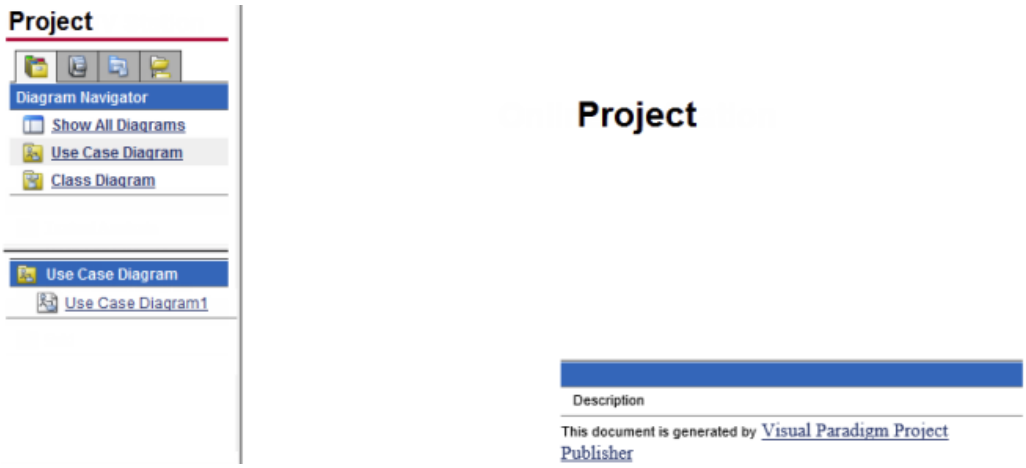
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

About publisher output

Go to the output directory of the published project and open the file 'index.html' with a web browser. The web page is organized in frames, namely the Navigator Pane, Menu Pane and Content Pane.



Published project

No.	Name	Description
1	Navigator pane	It comprises of the Diagram Navigator, Model Navigator and Class Navigator.
2	Menu pane	It shows the sub-menus of the Navigator pane. The contents shown in this pane varies with the link you clicked in the Navigator Pane.
3	Content pane	It shows the details of the item (diagram, model or package/class) you clicked in the Menu Pane or Content Pane.

Description of the interface of published Web content

Navigator pane

There are four tabs within the navigator pane - **Diagram Navigator**, **Model Explorer**, **Class Navigator** and **Logical View**. They are responsible for reading the project from different angles.

Diagram Navigator

Diagram Navigator shows the categories of diagrams in the project. You can click on a category to view its diagrams in the Menu Pane, or click Show All Diagrams to view all diagrams.

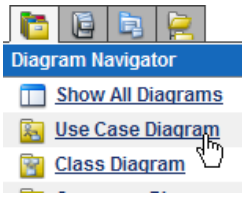
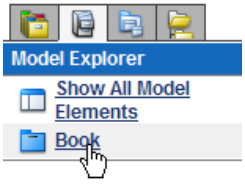


Diagram navigator

Model Explorer

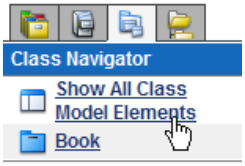
Shows the package models in the project. You can click on a package to view its child models in the **Menu Pane**, or click **Show All Models** to view all model elements.



Model Navigator

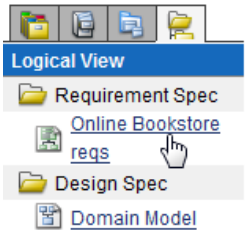
Class Navigator

Shows the Package models in the project. You can click on a package to view its child packages/classes in the **Menu Pane**, or click **Show All Models** to view all packages/classes.



Class navigator

Logical view
Echos the logical view defined in project. You can click on a diagram to open it.



Logical view

To view the details of an item (diagram, model or package/class), click on its link in the **Menu Pane** and its details will be shown in the **Content Pane**.

Menu navigator

Diagram Content

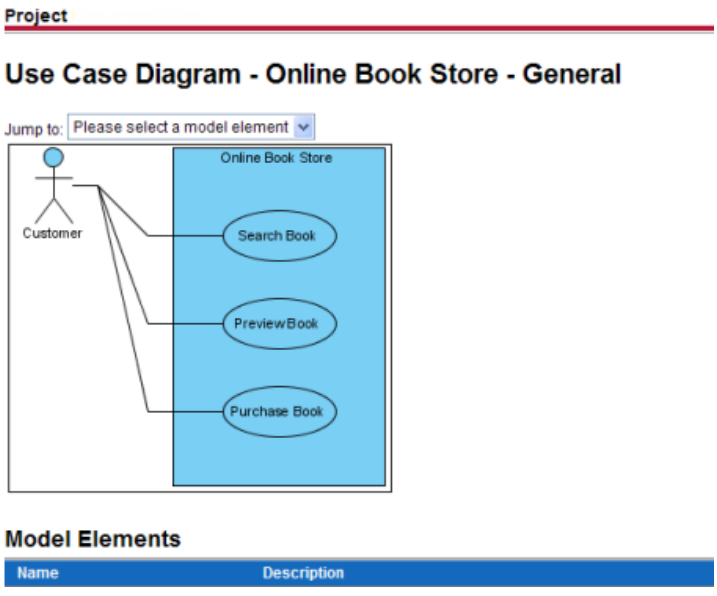
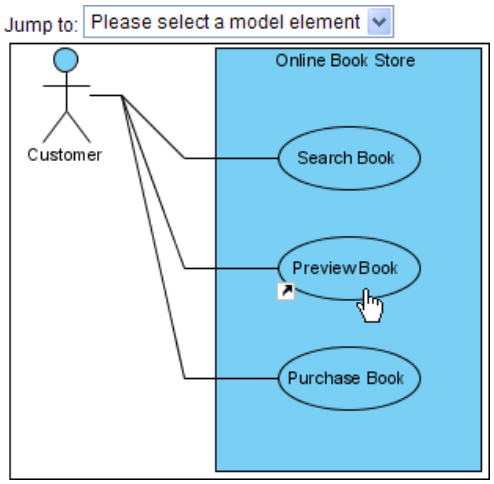


Diagram content

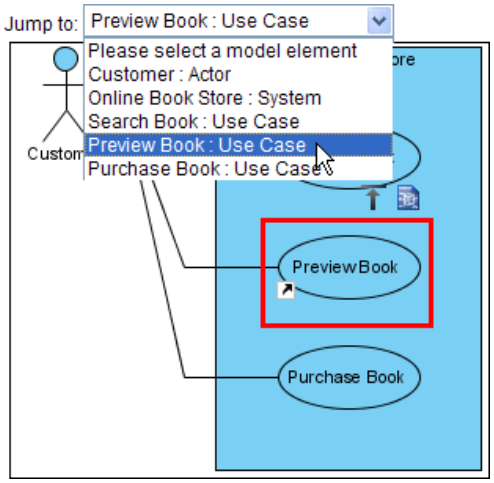
The diagram type, name, description, together with a full size image of the diagram are shown in the Content Pane. The image is mapped to different clickable regions for each shape, so, you can click on a shape in the image to view its details.



Shape link to descriptions

Using jump to

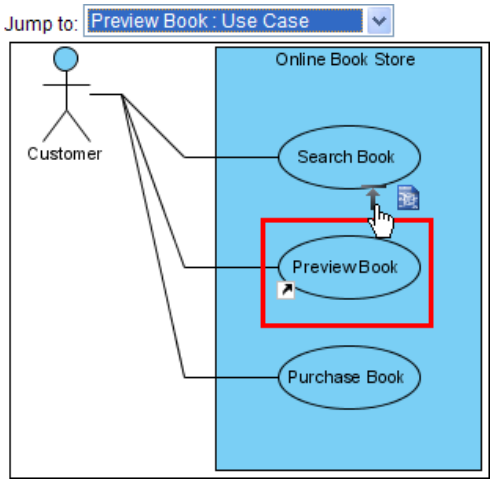
The **Jump to** drop down menu in the diagram content page lists all shapes in the diagram. You can select a shape to jump to. The content page will scroll to the selected shape and the shape will be highlighted by a red border.



Jump to

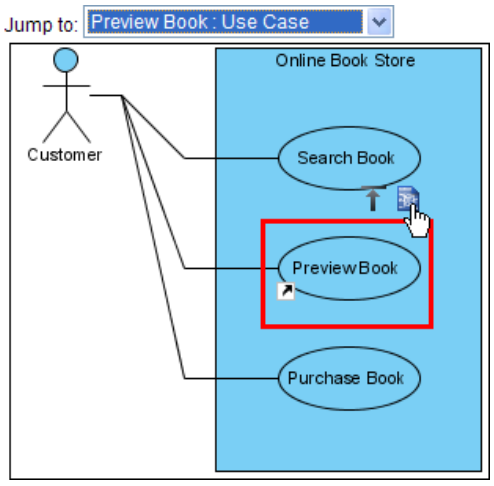
Besides, there will be two shortcut buttons above the selected shape.

The Back to top button brings you to the top of the page.



Back to top

The **Open specification** button brings you to the details page of the shape.



Open specification

Model elements

The Model Elements section of the diagram content page shows the name, type and description of the models of all shapes in the diagram. You can click on the link of a model to view its details.

Model Elements

Name	Description
Customer : Actor	
Online Book Store : System	
Search Book : Use Case	
Preview Book : Use Case	
Purchase Book : Use Case	

Model elements

Model element content

The type, name and general model properties of a model are shown in the content page.

Project

Online Book Store : System

Use Case - Preview Book

Properties

Name	Value
Abstract	false
Leaf	false
Root	false
Rank	Unspecified
Business Model	false

Relationships Summary

Name	Begin	End
: Association	Customer : Actor	Preview Book : Use Case

References

File Name	Description
C:\Demo\OrderForm.png	

Model element content

Parent hierarchy

The parent hierarchy is shown as a list of models on top of the page. You can click on a parent in the hierarchy to view its details.

Project

Online Book Store : System

Use Case - Preview Book

Parent hierarchy

Relationships

The summary of the relationships of the model is shown in the Relationships Summary section. Click on a relationship and it will take you to the Relationships Detail section.

Relationships Summary

Name	Begin	End
: Association	Customer : Actor	Preview Book : Use Case

Relationships summary

Relationships detail

Relationships Detail

Name	Value	
Type	Association	
From	Name	Value
	Role	
	Element	Customer : Actor
	Multiplicity	Unspecified
	Navigable	true
To	Name	Value
	Role	
	Element	Preview Book : Use Case
	Multiplicity	Unspecified
	Navigable	true
Abstract	false	
Leaf	false	
Visibility	Unspecified	
Derived	false	

Relationships detail

Other model details

Certain types of model have their own properties, for example, attributes and operations of class or columns of ERD table. They are also included in the content page as custom sections. For instance, the Operations Overview and the Operations Detail sections show the overview and details of the operations of a class respectively.

Operations Overview

Visibility	Return Type	Name
public	ORM_Shipment	loadShipmentByDate

Operations Detail

Name	Value
Name	loadShipmentByDate
Type Modifier	[]
Visible	true
Return Type	ORM_Shipment
Visibility	public
Scope	instance
Query	false
Abstract	false

Other model detail

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Doc. Composer - Introduction

Introducing Doc. Composer

Introducing Doc. Composer, a document builder that provides the tools you need to write your project documentation.

Introducing Doc. Composer

Doc. Composer is a document builder in Visual Paradigm. It provides the necessary tools development teams need to write their own project documentation with design specification embedded.

What is Doc. Composer?

Documentation is important in any software project. We write software requirement specification for describing requirements, database specification for detailing database structure, process specification for visualizing business activities, etc. Well written documentation can ensure quality software to be developed, and can make a good impression on customers and stakeholders. However, we understand that time is limited and writing good documentation is not something we relish doing. Therefore, we introduced Doc. Composer, a document builder that saves your previous time in writing documentation by providing a smooth integration between your documentation and your software design. As long as you need to have your design or design specification appear in your documentation, Doc. Composer can help.

Modes of Doc. Composer

When you open Doc. Composer, you will be prompted to select either to build a document from scratch, or to produce a document with a "Fill-in Doc". Here is a description of the two modes of Doc. Composer.

Build from Scratch

To build a document with the Build from Scratch mode is to being from a blank document, and then make use of the tools and element templates to write and complete the document.

Fill-in Doc

Typically, a project documentation or report is a combination of background information like project goal, scope and constraints, and design details like use case details, database design, process design, etc. The Fill-in Doc mode of Doc. Composer is designed to help you "fill-in" the design details of your documentation.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Doc. Composer - Build from Scratch

Introduction to Doc. Composer's Build from Scratch Mode

Begin from a blank document, and then make use of the tools and element templates available in Doc. Composer to write and complete the document.

Creating a Document

Create a document in Doc. Composer, under Build from Scratch mode.

Overview of Doc. Composer

Quick look of the various part of Doc. Composer in Visual Paradigm.

Developing a Document Using Element Template

Learn how to easily build a document in Doc. Composer, with the Build from Scratch mode.

Working with Content Block

Learn how to edit content blocks in Doc. Composer, and how to move a block up and down.

Using Loop

Query children element of element selected in Diagram Navigator / Model Explorer with the help of the Loop tool.

Adding Custom Text

Include custom text content in a document with the help of custom text.

Adding Image

Insert pictures, logo images or whatever images into Doc. Composer. You will learn how to add image in this article.

Add Table

Structure data with table. In this article you will learn how to insert a table into Doc. Composer.

Adding Page Break

You will learn how to insert page breaks into Doc. Composer, under Build from Scratch mode.

Using Section

You will learn how to add and configure sections in Doc. Composer in this article.

Adding Table of Contents

A table of contents is a list of key parts of a document. In this article you will learn how to add table of contents into Doc. Composer.

Adding Revision Log

Record the version of document, the date/time when the change took place, the person who made the change and other necessarily remarks regarding the changes with Revision Log.

Adding Cover Page

A cover page is a page that can be added at the beginning of a document. In this article you will learn how to add cover page into Doc. Composer.

Various Page Display Options

Doc. Composer supports 4 display options: single page, single page continuous, two-up and two-up continuous. We will cover each of them in this article.

Keeping Your Document Updated

Since document maintains the linkage between project data and document content, you can refresh the document upon project changes. You will learn how it works in this article.

Writing Your Template

Instead of creating a document with built-in element templates, you can write your own. You will learn how to do this in this article.

Exporting a Document

In Visual Paradigm, there are three types of document available for exporting: HTML, PDF and Word. This article will demonstrate how to export a document.

Introduction to Doc. Composer's Build from Scratch Mode

To build a document with the **Build from Scratch** mode is to begin from a blank document, and then make use of the tools and element templates available in Doc. Composer to write and complete the document. As an overview, "Build from Scratch" works in this way:

1. You create a document in Doc. Composer.
2. Form the content by dragging and dropping element templates from the **Templates** pane onto the document.
3. Touch-up the document by adding TOC, defining headers & footers, configuring styles, etc.
4. Export the document to a document file in HTML/PDF/MS Word format.

Understanding Element Template

An element template defines what and how content gets output in a document. For example, a Data Dictionary template is capable in producing a data dictionary in a document, with dedicated type of project data presented in the data dictionary (table). Each type of project data has its own set of element templates. You have Data Dictionary template for Entity Relationship Diagram and Sub Diagrams template for Use Case, etc.

Doc. Composer comes with a set of built-in element templates, but you can also create and edit custom templates. For example, you can create a custom template to output a table of actors' details, and then a list of use cases, and then a list of sub-diagrams.

Element templates are written using an XML-based language called Doc. Composer Template Language. If you want to create your own templates, you can learn this language by clicking here. In this chapter our focus is primarily on the use of built-in templates in creating a document.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

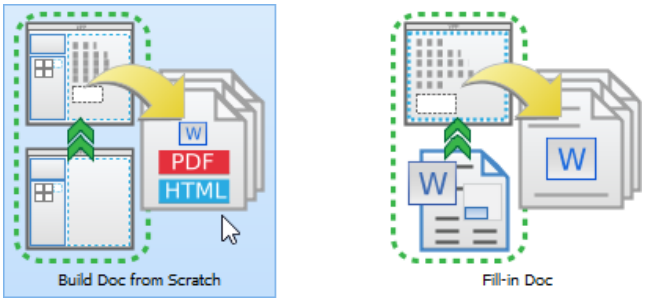
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Creating a Document

To create a document:

1. Under the **Tools** tab, click on the **Doc** button and then select **Doc. Composer** from the drop down menu.
2. Click **Build Doc from Scratch**.

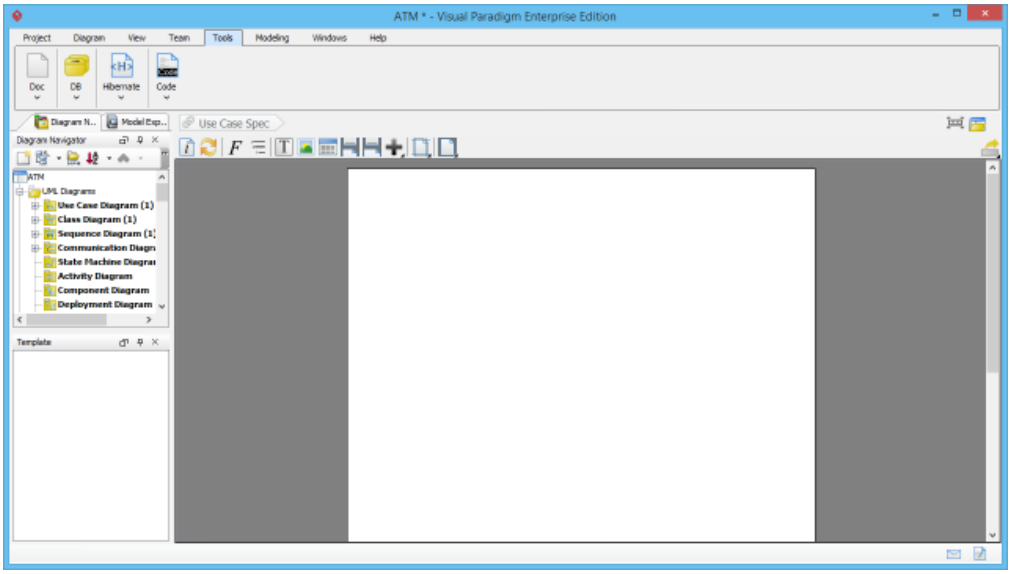


Selecting the Build Doc from Scratch mode of Doc. Composer

3. Name the document by double clicking on *Document1* in the breadcrumb and then typing in a new name.

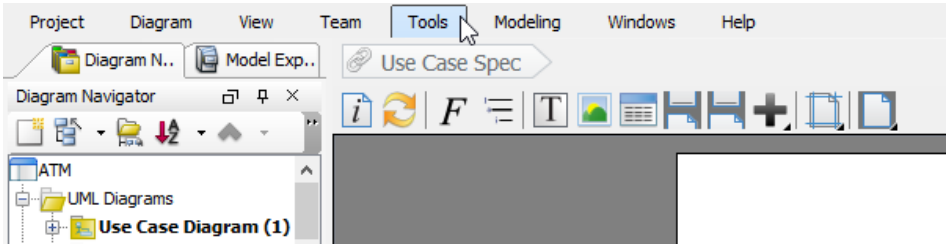
Entering document name

4. Press the **Enter** key to confirm the naming. Your screen should look like this:



Doc. Composer (Fill-in Doc)

5. To have more editing space, we recommend you to collapse the toolbar temporarily by double clicking on the **Tools** tab.



Collapsed toolbar

NOTE: If you don't see the **Doc. Composer** button, make sure you are running the Standard Edition (or higher) of Visual Paradigm, and have **Sleek** chosen as the UI style.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

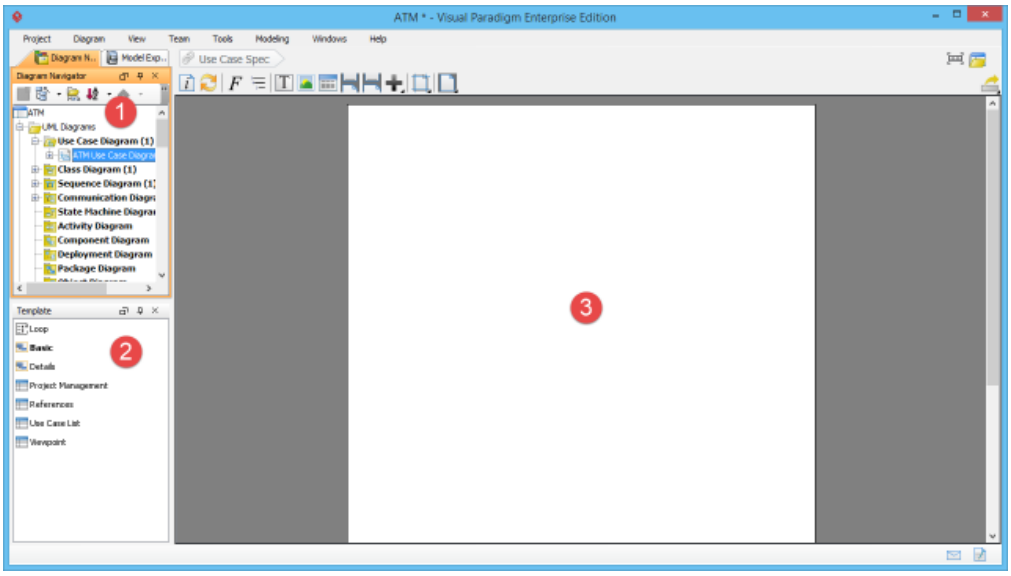
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Overview of Doc. Composer



Doc. Composer overview

The "Build from Scratch" mode of Doc. Composer consists of three main parts:

Part	Name	Description
1	Diagram Navigator/ Model Explorer	The **Diagram Navigator** and **Model Explorer** (behind **Diagram Navigator**) provides you with access to different parts of your project– the project, diagram, diagram elements and model elements. If you want to output content for say a use case diagram named "My Diagram", select "My Diagram" in **Diagram Navigator** (or **Model Explorer**) first.
2	Element Template List	A list of element templates available for the project data selected in **Diagram Navigator** or **Model Explorer**. As said earlier, each type of project data has its own set of element templates. Take the image above as an example, by selecting a use case diagram in Diagram Navigator, the templates **Basic**, **Details**, **Project Management**, etc. are listed. If you select other project data in **Diagram Navigator** or **Model Explorer**, a different list of element templates will be presented. Document creation is the process of dragging and dropping a suitable template from the Element Template List onto the document.
3	Documnet	Content of document. What you can see here will be what you will get when you export the document as HTML/PDF/Word. On top of the document there is a toolbar that provides you will access to tools like format configuration, report configuration, page break insertion, etc. We will talk about the tools later on in this chapter.

Description of Doc. Composer

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

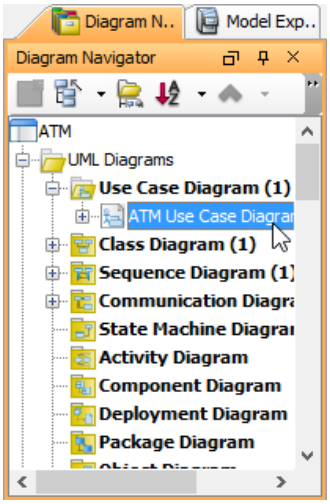
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Developing a Document Using Element Template

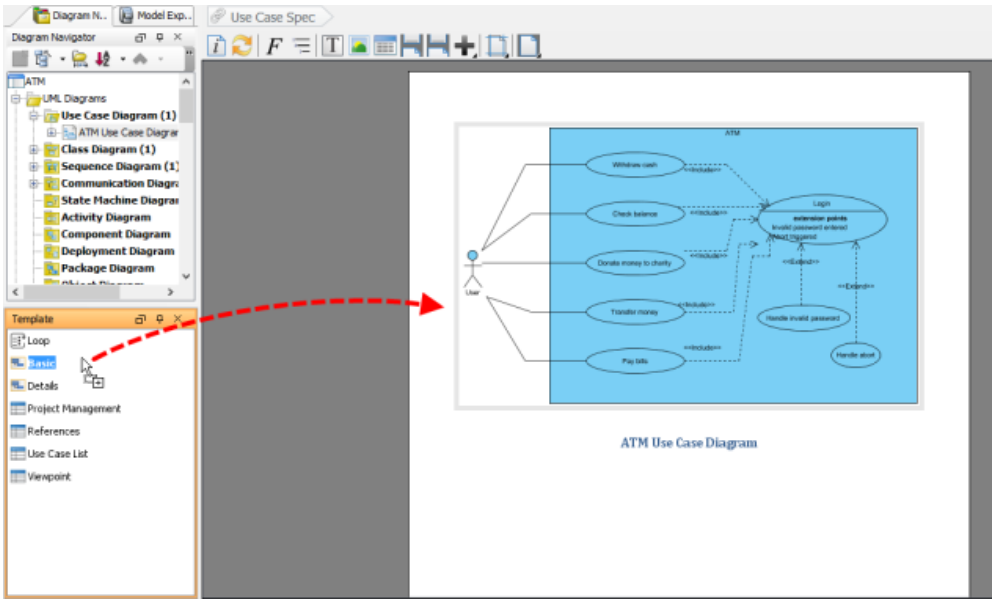
Doc. Composer provides a flexible way for you to create project documentation. All you need to do is to select your target model element/diagram, drag the elements template(s) from the Element Template Pane onto the document.

1. Select project / diagram / diagram element / model element on **Diagram Navigator** / **Model Explorer**.



Selecting a diagram

2. Select the desired element template from the **Element Template Pane** and drop it/ them on the document.



To drag an element template onto document

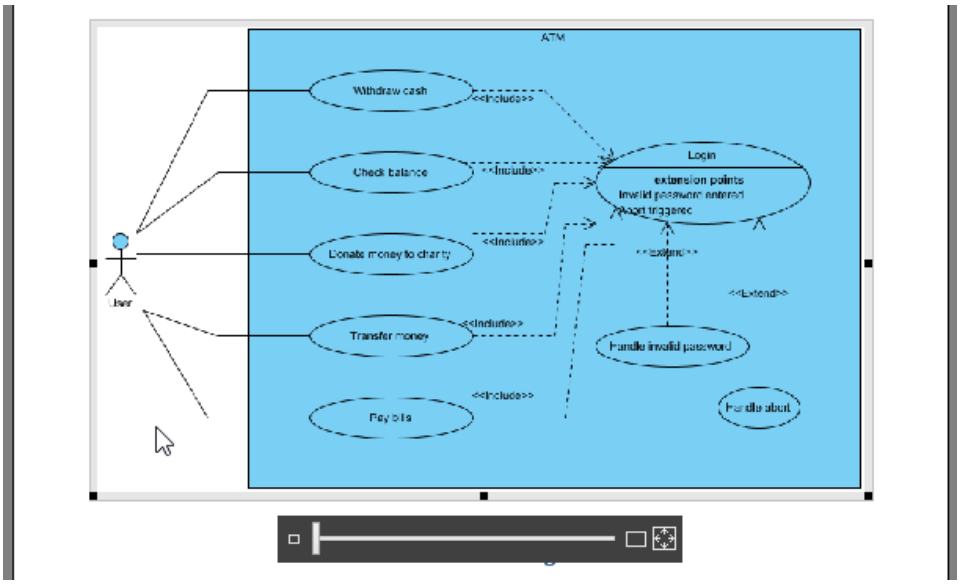
3. Repeat the previous step to build the document.

> **NOTE:** Instead of adding content element by element, you can select multiple elements at the same time to speed up the document creation process.

Editing Image

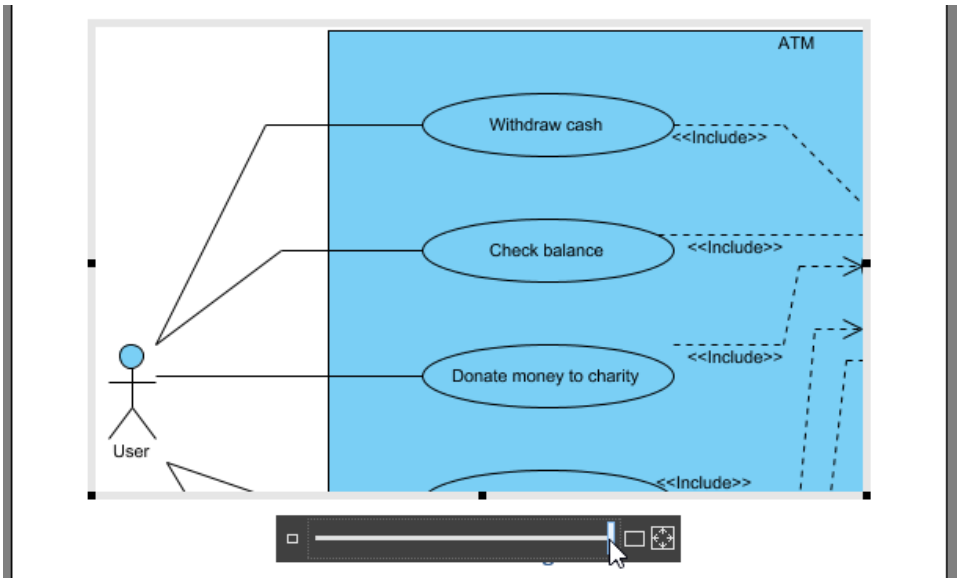
You can add diagram images into a document with the use of element template. For diagrams presented on a document, instead of showing the entire diagram you may want to have it focused on a specific part of a diagram. In order to achieve this, edit the image by taking the steps below.

1. Click each image so that a bar will appear at the bottom of the image. You can edit the image through the bar. Initially, the whole diagram is displayed to fit the placeholder.



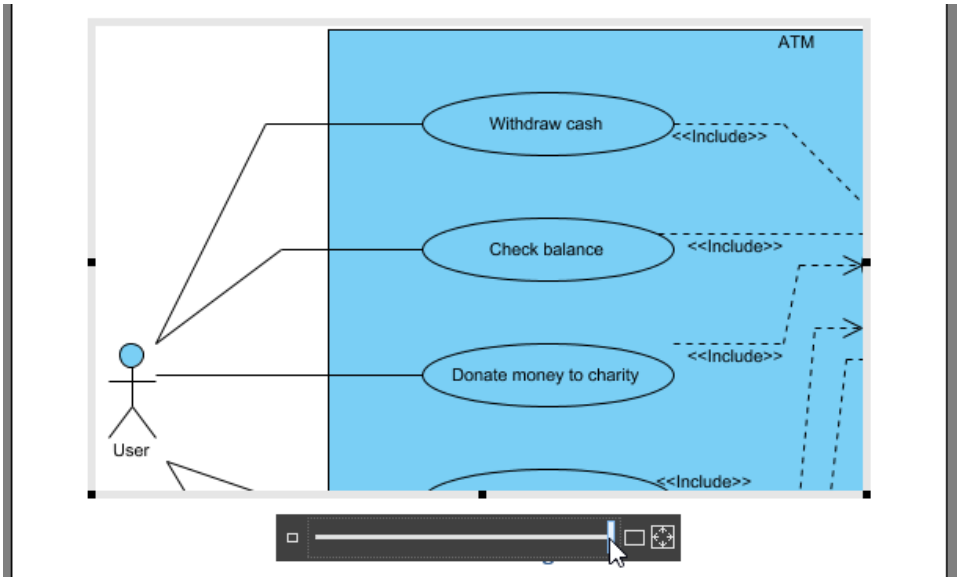
Selected an image

To zoom in the particular part of diagram, drag the slider to Zoom 100% (Actual Size).



Zooming into an image

2. To resize the image, drag the border of placeholder.



Resizing an image

NOTE: You can only edit the images produced by templates you dropped onto the document but not inserted images.

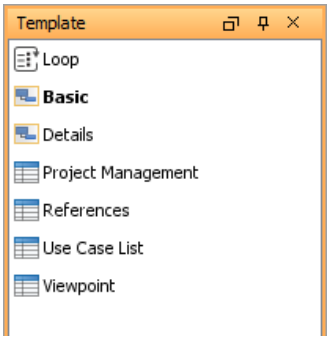
NOTE: The default setting of image on document is **Disable Auto Fit Placeholder**. Nevertheless, once you zoom or resize, it will turn to be **Enable Auto Fit Placeholder** automatically.

Understanding the default template

Each type of project data has its own set of element templates and, among these templates, one of them is the default template.

Normally, we add content into a document by first selecting an element in **Diagram Navigator** or **Model Explorer,** and then dragging a template from the **Element Template List** onto a diagram. This works well but wouldn't it be a bit faster if we can skip the template selection part? Default template was designed to serve this purpose. Besides dragging a template onto a document, you can also select a piece of project data in **Diagram Navigator** or **Model Explorer**, and then drag it directly onto the document to add content. If you do this, we will add content based on the default template of the selected element.

Default template is shown with its name bolded under the **Element Template Pane**, like the **Basic** template in the following image:



Basic template - the default template

By default, the **Basic** template is chosen to be the default template. You can, however, set a user created template to be the default. Note that you can select either the **Basic** template, or any of the user created templates as default. You cannot select a built-in template as the default, except the **Basic** template.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

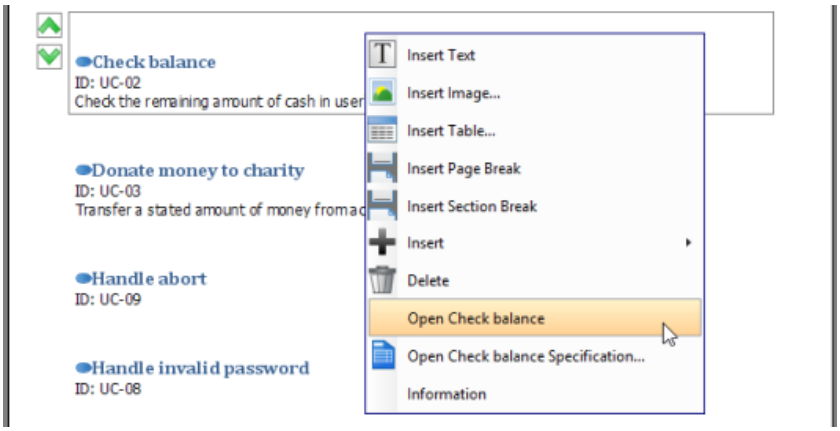
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Working with Content Block

Opening the Underlying Element

If you want to open the source element from which a content was created, right click on the content block and select **Open %NAME%** from the popup menu where **%NAME%** is the name of the element. Very often you need this when you spot a design flaw when reading a document, and you want to correct it.



Opening a use case from Doc. Composer

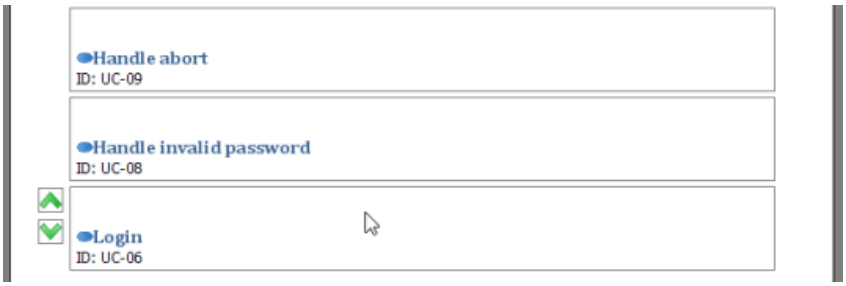
Opening the Specification of the Underlying Element

If you want to review or edit properties of an element from which a content was created, right click on the content block and select **Open %NAME% Specification** from the popup menu where **%NAME%** is the name of the element.

Repositioning a Content Block

You can always re-order content blocks in a document by performing these steps:

1. Press to select the blocks to move up or down. You can perform a multiple selection by pressing the **Shift** or **Ctrl** key.



Selecting multiple content blocks

2. Click on **Move Up** or **Move Down** to re-order the selected blocks.

Moving a content block

Deleting a Content Block

If you want to remove a content block from a document, simply select the blocks and press Delete.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

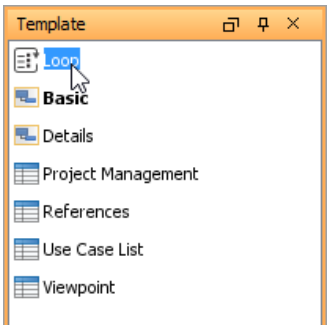
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

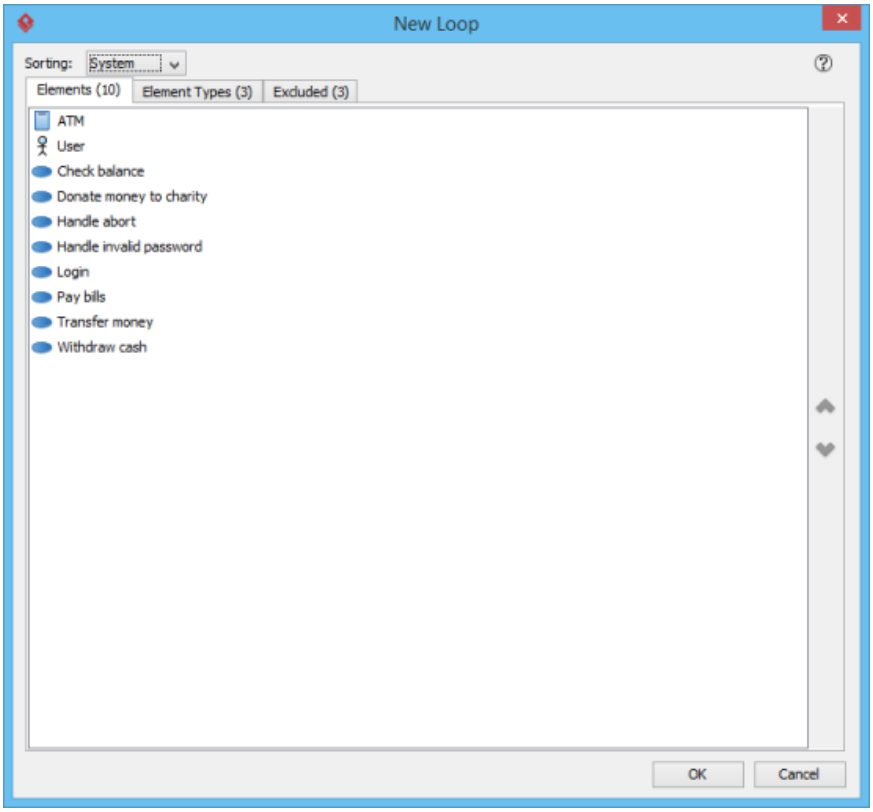
Using Loop

Besides element template, there is another tool to query project data and place it onto a document, called the Loop tool. You can access the Loop tool under the **Element Template Pane**.



The Loop tool under the Element Template Pane

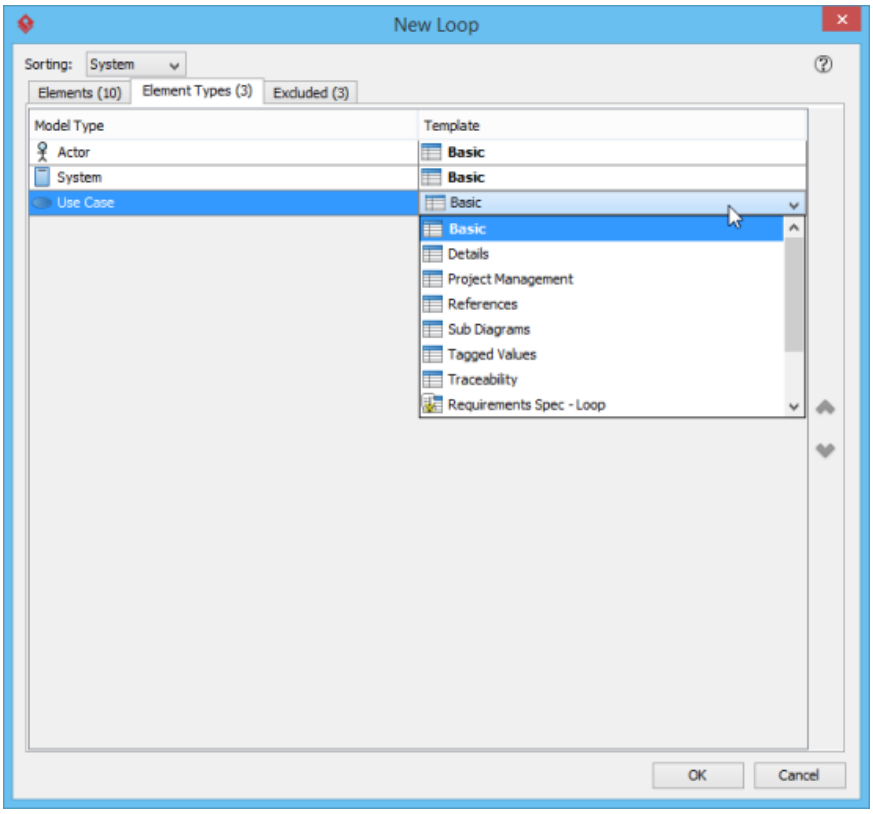
The Loop tool is capable in querying children element of element selected in **Diagram Navigator** / **Model Explorer**. When you select and drag the Loop tool onto a document, you will be prompted to configure the Loop, like this:



Configure loop

Items listed under the **Elements** tab are children elements of the element selected in **Diagram Navigator** / **Model Explorer**. For example, if you have selected a diagram, you can expect the diagram elements being listed here.

Content will be added to document for each of the elements listed. The content to produce is determined by an element template. You can select the template under the **Element Types** tab.



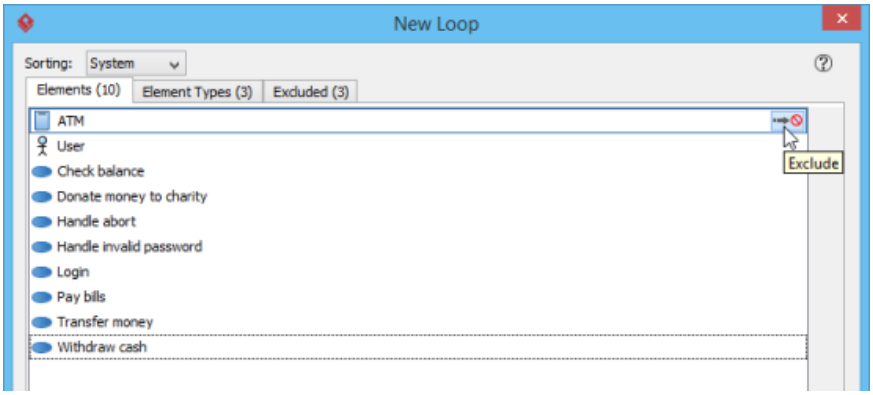
Choosing an element template for an element type

The left hand side of the **Element Types** tab lists the types of elements found by the loop while the right hand side lists the element template to be chosen for content creation. By default the **Basic** template is chosen for all element types. This means that for each of the elements under the **Elements** tab, content will be produced and added to the document based on the **Basic** template. You can select another template by click on **Basic** and make a selection.

Excluding Elements and Element Types

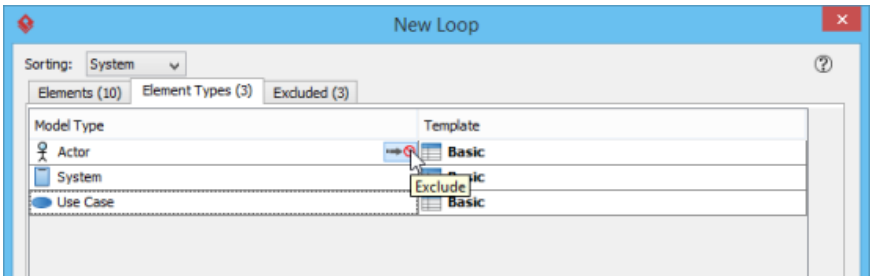
Sometimes, you may not want to produce content for all elements returned by a loop, but only some of the elements. For example, when you create a use case specification, you may not want to process the actors when looping under a Use Case Diagram. For such cases, you can exclude the elements or types of element that are not needed in content creation.

You can exclude an element, or a type of element. To exclude an element, move your mouse pointer over that element under the **Elements** tab and the click on the **Exclude** button on the right hand side of the hovering row.



Excluding an element from loop

To exclude an element type, open the **Element Types** tab and move your mouse pointer over the type to exclude, then click on the **Exclude** button.

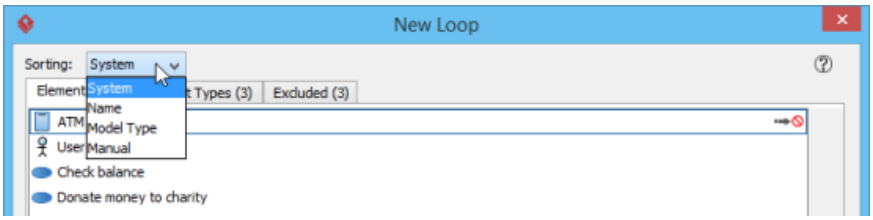


Excluding an element type from loop

If you want to remove an element or element type from the exclude list, open the **Excluded** tab, move your mouse pointer over the item to be removed and then click on the **Include** button on the right hand side of the hovering row.

Sorting

To re-order elements, click on the **Sorting** drop down menu at the top of the **New Loop** window and select the way to sort.



Changing sort option

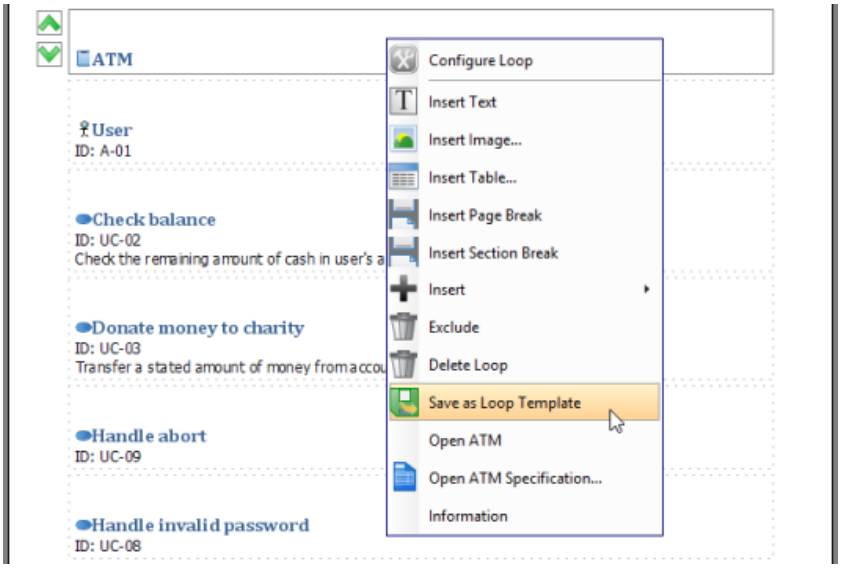
Here are a description of the various types of sorting method:

Type	Description
System	Elements are sorted according to the system's default setting.
Name	Sort by elements' name in alphabetical order.
Model Type	Sort by element type. You can customize the sort order in the Element Types tab.
Manual	Order the elements yourself.

Description of sort options

Saving a Loop Template

Although Loop itself is tool instead of a template, you can produce a template from it. Doing so allows you to customize the template further by editing the loop in XML form. To produce a loop template, right click on any content block produced with the Loop tool and select **Save as Loop Template** from the popup menu.



Saving a loop into loop template

You will be prompted for a template name. Enter the name and confirm. When finished you will find a new template listed in the Element Templates Pane. You can customize it and re-use it in creating content. For details about writing a template, please read Writing Your Template.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

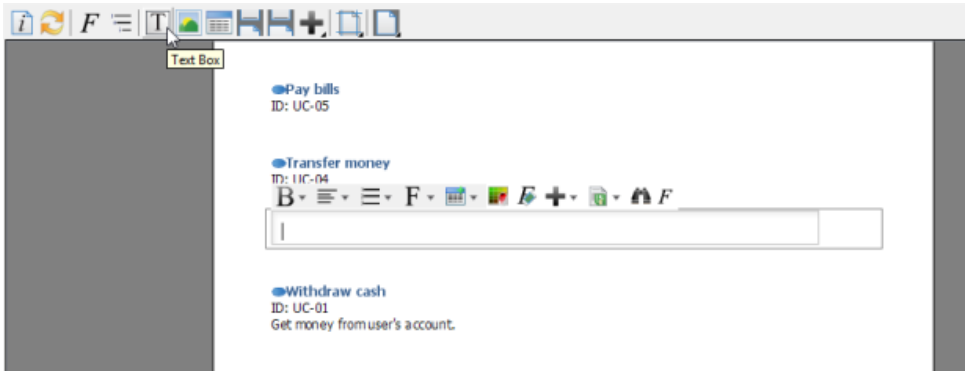
- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Custom Text

Although you cannot type in a document directly, you can add text through creating text boxes.

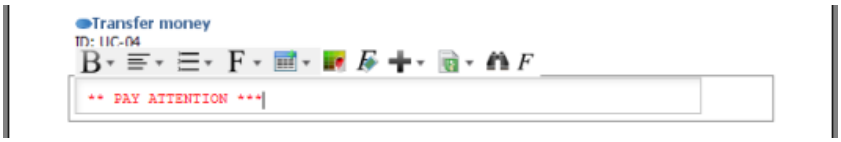
The text box is used for editing data on document. The significant characteristic is you can display many different types of data by applying RTF within the text boxes.

1. Select the space where you want to insert text beforehand.
2. Click Text box button on the document's toolbar.



Adding a text box

3. Enter text in the text box. You can use the pop-up formatting toolbar to convert your plain text into RTF when you want to emphasize some terms/ phrases.



Entering text in text box

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

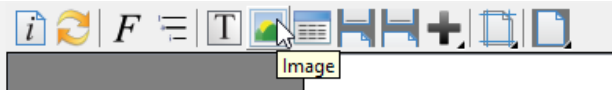
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Image

Document supports inserting images. An image can be a logo or picture that is placed on the document. Not only you can place pictures on the empty space of document but also fit them inside table cells. In this sense, you can insert your company logo into any preferred place within the document when you are doing a company document. The advantage is you can spare no effort in arranging a series of images in document and then resize them. To add an image:

1. Select the content block where you want to insert an image beforehand.
2. Click **Image** button on the document's toolbar.



To add an image

3. Select the directory of your target image and then click **Open** button in **Choose image(s)** dialog box. As a result, the selected image is inserted.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

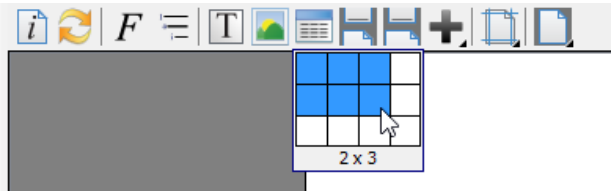
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Table

Table is one of the popular elements in structuring data. To create a table:

1. Select the content block where you want to insert a table beforehand.
2. Click on the  button on the document's toolbar.
3. Select the number of rows and columns to be created.



Selecting the number of rows and columns

4. Complete the table.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Page Break

To end the current page and continue the document to the next page, add a page break. To add a page break:

1. Select the content block where you want to insert a page break beforehand.
2. Click on the  button on the document's toolbar.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Using Section

A section is a number of continued pages that apply the same set of page properties. These properties include page size, page orientation, page margin, visibility of header/footer, content of header/footer, etc.

Because section allows you to define different layouts for different pages, you can make pages that consist of wide tables show in landscape, with the other pages remain in portrait. You can also add content-specific header and footer.

By inserting a section break, pages that appear after the break will apply the same set of page properties as defined in the break. To insert a section break:

1. Select the content block where you want to insert a section break beforehand.
2. Click on the  button on the document's toolbar. A section page is inserted which moves the chosen content to a new page.

The layout of pages within a section are controlled by the setting configured in the section break. To configure a section break:

1. Right click on the section break.
2. Select **Edit...** from the popup menu. This shows the **Section Properties** window.
3. Edit the settings and click **OK** to confirm the change.

Here is a description of different parts of the **Section Properties** window.

Part	Name	Description
1	Follow previous section	When checked, section properties will follow that defined in previous section. For the first section, it follows the properties set to the whole document.
2	Page size	Determine the dimension of page.
3	Page Orientation	Orientation of page, either in portrait or landscape.
4	Measurement unit	Unit of margin.
5	Top margin	Determine the empty space at the top of a page. You can edit the margins size by entering the sizes into the text fields. Alternatively, click the spinner buttons to increase/ decrease the margin sizes.
6	Right margin	Determine the empty space on the right hand side of a page. You can edit the margins size by entering the sizes into the text fields. Alternatively, click the spinner buttons to increase/ decrease the margin sizes.
7	Bottom margin	Determine the empty space at the bottom of a page. You can edit the margins size by entering the sizes into the text fields. Alternatively, click the spinner buttons to increase/ decrease the margin sizes.
8	Left margin	Determine the empty space on the left hand side of a page. You can edit the margins size by entering the sizes into the text fields. Alternatively, click the spinner buttons to increase/ decrease the margin sizes.
9	Previous of page	Preview the effect of adjusting margin.
10	Header/Footer	**Show** - Show the header/footer in document **Hide** - Hide the header/footer in document **Separator** - When checked, a line will be shown between header/footer and the main content. **Continue with previous section** - Following the previous section means to have the visibility of header, footer and separator follow that defined in the previous section. If you have added page number to header/footer, the numbering will continue with the previous section, too. If not to follow the previous section, the numbering will reset to 1.

Overview of the Section Properties window

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Table of Contents

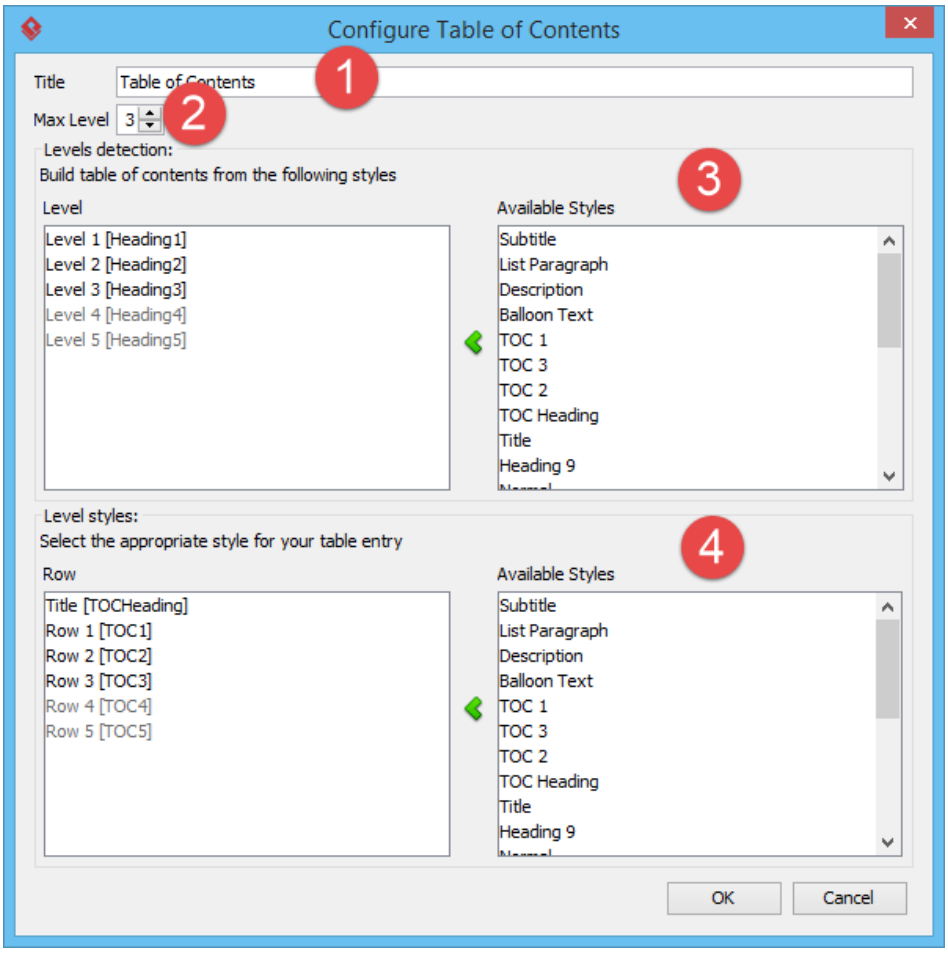
A table of contents is a list of key parts of a document. It is often constructed by headers or key titles in a document, to present readers with and outline of the whole document.

Doc. Composer allows you to insert a table of contents into a document. A table of contents can be formed not only from traditional headings styles like Heading 1 and Heading 2 but from any kind of style, even from user-defined styles. To insert a table of contents:

1. Select the content block where you want to insert a table of contents beforehand.
2. Click on the button on the document's toolbar and then select **Table of Contents** from the drop down menu.

To change the title, maximum number of level, level detection or styles of a table of contents, to configure it. To configure a table of contents, right click on the table of contents and select **Configure Table of Contents...** from the popup menu.

Here is a description of different parts of the **Configure Table of Contents** window.



Configure table of contents

Part	Description
1	The title of the table of contents. This is the text that appear above the table of contents in document.
2	Determine the depth of the table of contents.
3	Specify the style to check for each level. If you want level 1 shows all content with Heading 1 as style, select Level 1 on the left hand side, Heading 1 on right hand side, and click < to match them up.
4	Specify the appearance of text in table of contents. You can apply different styles for different rows (levels).

Overview of Configure Table of Contents window

To update a table of contents to make it reflect the structure of the latest document content, right click on the table of contents and select **Update Table of Contents** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

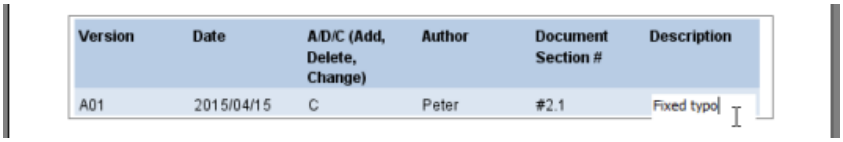
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Revision Log

When your team attempt to or has made significant changes on a document, you may want to record the version of document, the date/time when the change took place, the person who made the change and other necessarily remarks regarding the changes. Revision log is a piece of content you can add to a document to record all these information. With a revision log, you fill in the revision detail as well as to add/remove columns to suit the requirement of your team. To insert a table of contents:

1. Select the content block where you want to insert a revision log beforehand.
2. Click on the  button on the document's toolbar and then select **Revision Log** from the drop down menu.
3. To enter a revision, double click on the cells and enter the values one by one.



Version	Date	A/D/C (Add, Delete, Change)	Author	Document Section #	Description
A01	2015/04/15	C	Peter	#2.1	Fixed typo

Editing revision log

4. If you want to insert more rows or columns, right click on the revision log and select **Insert Row** or **Insert Column** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Adding Cover Page

A cover page is a page that can be added at the beginning of a document. Whether or not to add such page is up to the writer. There are two kinds of cover page you can add into a document. The first one is to print the cover page in a program defined way. This approach requires you fill in some of the background information like the document title, organization name and author name, etc. The second kind of cover page is fully designed by you, the writer. It is called a free style cover page.

Built-in Cover Page

1. Click on the  button on the document's toolbar.
2. Open the **Cover Page** tab.
3. Configure the cover page by specifying the file path of logo image, title, organization name, author name. You can preview the page at the right hand side of the **Document Properties** window.
4. Click **OK**.

> **NOTE:** Built-in cover page is only visible in exported document, not in Doc. Composer

Free Style Cover Page

Free style cover page provides you with a page that appears at the beginning of a document for you to design the page. You can add any text and image freely on the cover page and position them in any position you like within the cover page. To insert a free style cover page:

1. Click on the  button on the document's toolbar and then select **Cover Page** from the drop down menu.
2. When you insert a free style page the first time, you are prompted to override the generate cover page option. By default, the built-in cover page would be chosen as cover page. When you try to insert a free style cover page, the built-in cover page would be ignored. This option is to ask for your confirmation for ignoring the built-in cover page. Click **OK** to confirm.
3. You will see an empty cover page added to the beginning of the document. Note that the page MUST be added to the beginning of document and you cannot control its location. If you want to add a page of custom content in the middle of the document, insert a **Free Style Page** instead.
4. Start editing the page by inserting text and image. To insert text or image into the page, right click on the background of cover page and select **Insert Free Style Text** or **Insert Free Style Image** from the popup menu.
5. Fill in the text or select the image file to insert to the page. Repeat step 3 and 4 to complete the page.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Various Page Display Options

Page display is especially useful when you view the overview of document layout. Doc. Composer supports 4 display options: single page, single page continuous, two-up and two-up continuous.

Option	Description
Single Page	Display only one page at a time.
Single Page Continuous	Display pages in a consecutive and vertical column.
Two-Up	Display two pages side by side simultaneously.
Two-Up Continuous	Display pages side by side in two consecutive vertical columns.

Page display options

Click the **Page Display Option** button to select a page display option from the drop-down menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Keeping Your Document Updated

Since document maintains the linkage between project data and document content, you can refresh the document upon project changes. As a result, it saves your time on repeating the steps for creating document.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

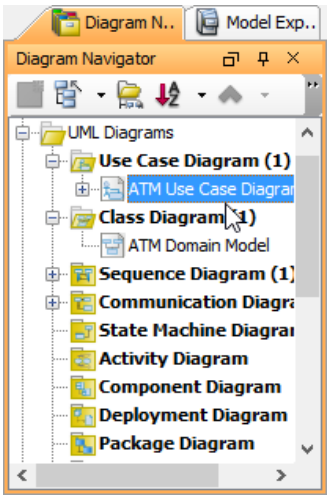
- Standard Edition
- Professional Edition
- Enterprise Edition

Writing Your Template

Although Doc. Composer comes with a complete set of built-in element templates, you may still want to customize or even to write your own templates for maximizing the efficiency of document design. You can accomplish this by writing and programming your own element templates.

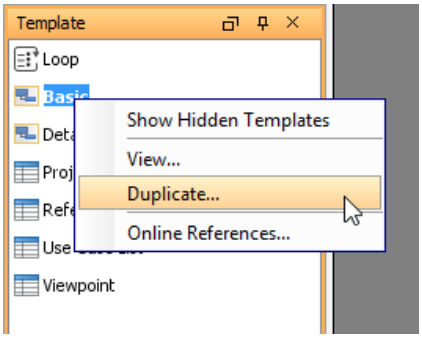
To create a template:

1. Open Doc. Composer.
2. Select the type of element to create template. For example, select ANY use case diagram in **Diagram Navigator** if you want to create a template to list specific shapes in use case diagram. You can select project / diagram / model element / diagram element in **Diagram Navigator** and **Model Explorer**.



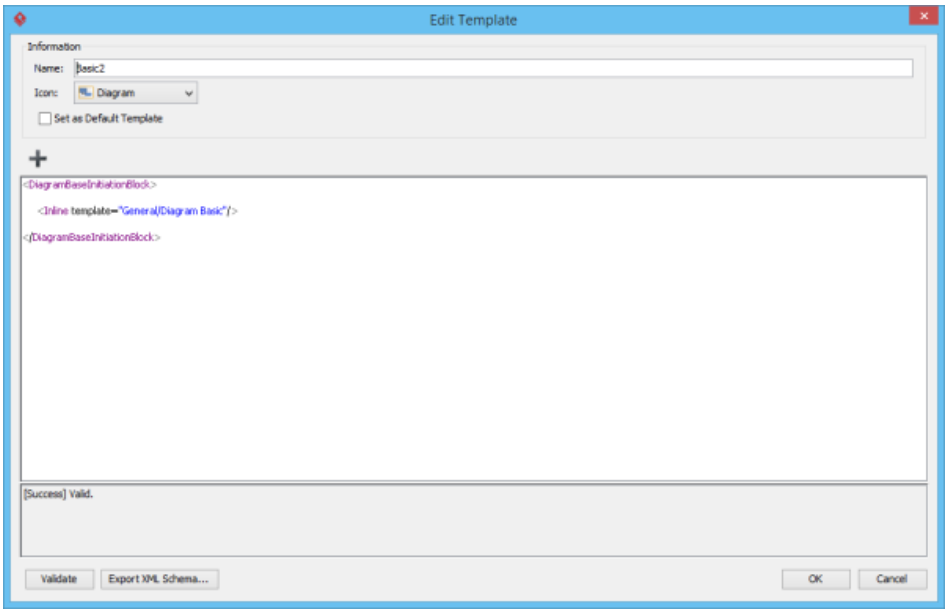
Selecting a diagram

3. The **Element Template Pane** lists the templates available for the selected element type. If your Template pane is hidden, select **View > Panes > Property** in the toolbar (or press **Ctrl+Shift+P**) to show it.
4. To simplify the programming of template, you are suggested to duplicate an existing template and start editing it, rather than do everything from scratch. Target on a template that gives the closest outcome to what you want to show in document. If you want to start from an empty document, select any templates. Right click on your selection and select **Duplicate...** from the popup menu.



Duplicate an element template

5. In the **Edit Template** window, specify the following information and click **OK** to continue.
Name: The name of the element template, which is the name that shown under the Element Template Pane.
Icon: An icon that best represent the layout of content that will be produced by using this template.
Set as Default Template: Check this option if you want Doc. Composer to apply this template automatically when dragging elements directly from Diagram Navigator / Model Explorer onto document.
Template content: Editor for programming the template.



Editing an element template

6. Customize your template and click **OK** to apply the changes. To learn how to write a template, click here.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

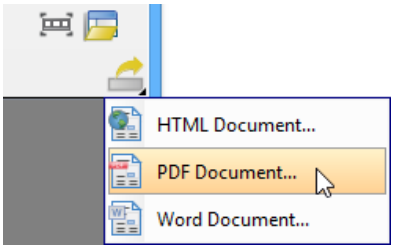
Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Exporting a Document

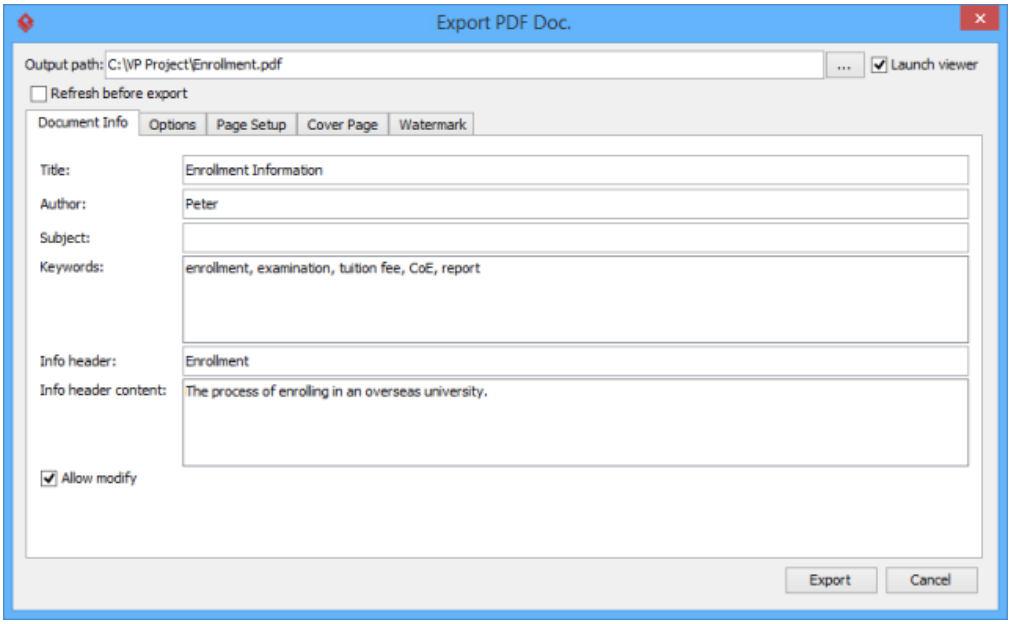
After you have customized your document template on document, you can export it into document. There are three types of document available for exporting: HTML, PDF and Word.

In document, click the **Export** button at the top right corner and select a type of document for exporting.



*Click **Export** button*

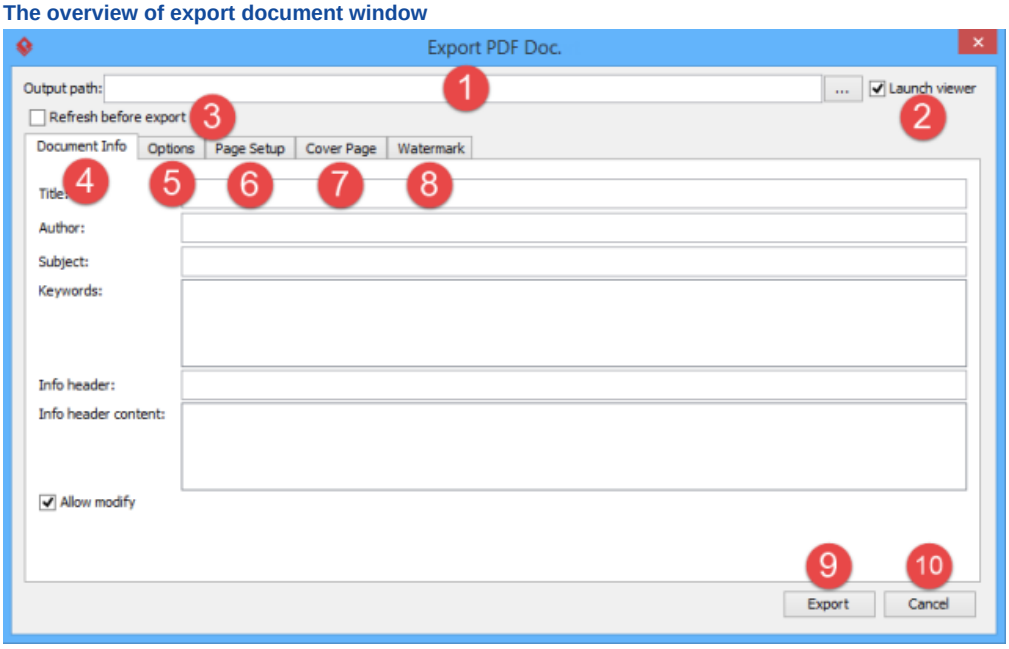
In the pop-up **Export [document type] document** window, specify output path and document info, and customize page setup, cover page and watermark.



Specify output path

At last, click **Export** button.

The overview of export document window



The overview of Export PDF document dialog box

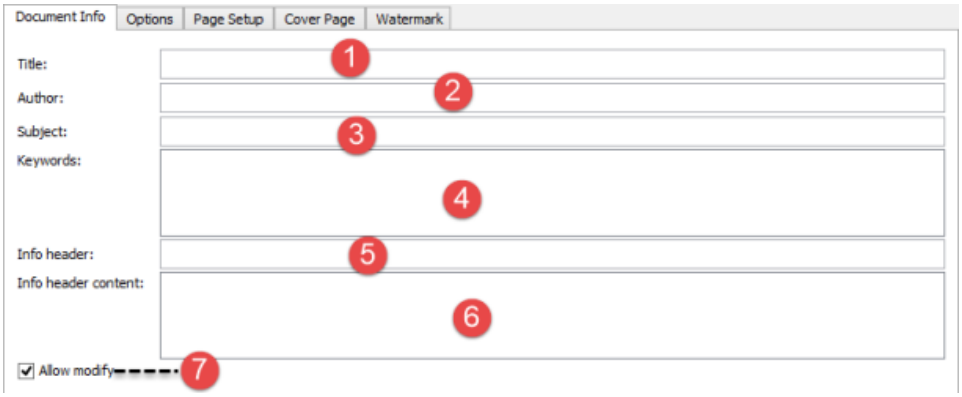
No.	Name	Description

1	Output path	The output path of document to be generated.
2	Launch viewer	Check to open the document automatically after generation.
3	Refresh before export	Before proceed exporting, refresh the document content.
4	Document info	To define document information.
5	Options	To determine how data is to be printed in document by setting some of the configurable options.
6	Page Setup	To customize the layout of document.
7	Cover Page	To customize the first page of document.
8	Watermark	To customize the watermark on document.
9	Export	Confirm and export the document.
10	Cancel	Close the export document dialog box without exporting.

Description of export document dialog box

NOTE: An additional **Content** tab is attached to **Export Word document** dialog box.

The overview of Document Info

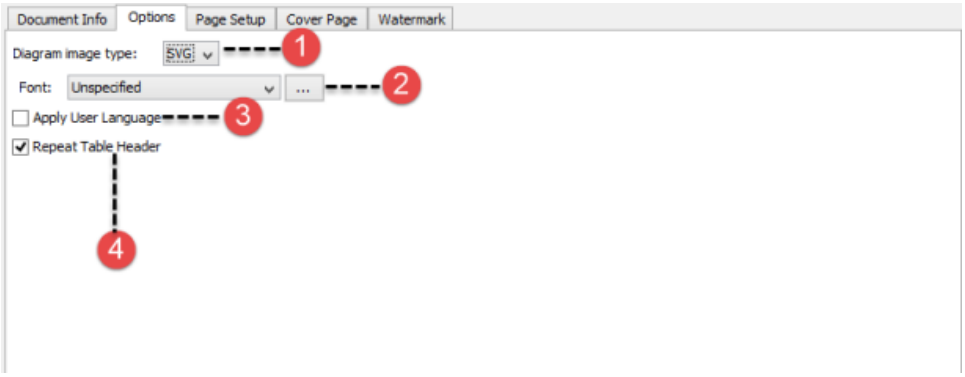


The overview of Document Info tab

No.	Option	Description
1	Title	The title of document. This option is only available for exporting PDF document.
2	Author	The author of the document.
3	Subject	The subject of the document. This option is only available for exporting PDF and Word document.
4	Keywords	The keywords of the document.
5	Info header	The info header of the document. This option is only available for exporting PDF document.
6	Info header content	The info header content of the document. This option is only available for exporting PDF document.
7	Allow modify	Select to allow modification on the document. This option is only available for exporting PDF document.

Description of Document Info tab

The overview of Options Setup

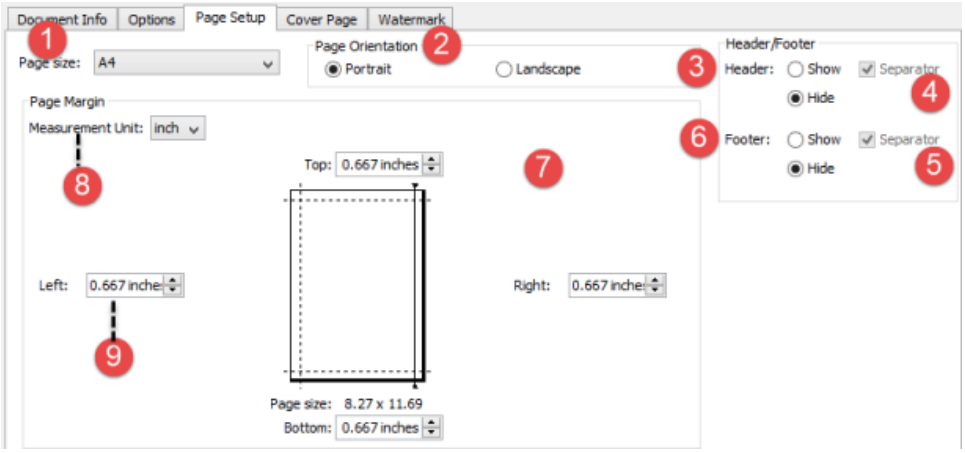


The overview of Page Setup tab

No.	Option	Description
1	Diagram image type	Select the type of image format for image that appear in the exported document.
2	Font	Control the font of document text.
3	Apply User Language	By default, document content will be printed in English. By checking this option, it will follow the language setting chosen in global options.
4	Repeat Table Header	By checking this option, table header would be repeatedly printed when the table span multiple pages.

Description of Page Setup tab

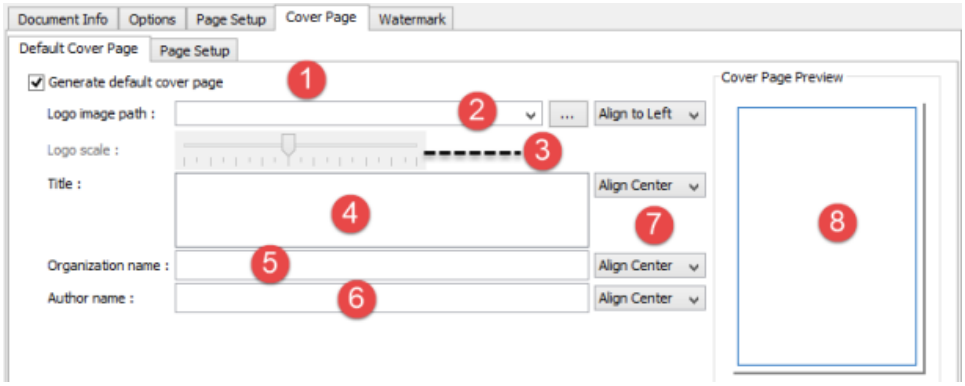
The overview of Page Setup



The overview of Page Setup tab

No.	Option	Description
1	Page size	To select the paper size of the exported document.
2	Page Orientation	This option is used to select the orientation of the document (portrait/ landscape). This option is only available to PDF and Word document.
3	Header	Check this option to insert header to the exported document. This option is only available to PDF and Word document.
4	Header Separator	Check this option to insert header separator to the exported document. This option is only available to PDF and Word document.
5	Footer Separator	Check this option to insert footer separator to the exported document. This option is only available to PDF and Word document.
6	Footer	Check this option to insert footer to the exported document. This option is only available to PDF and Word document.
7	Page Margin	To specify the page margins of the document: top, left, right and bottom. This option is only available to PDF and Word document.
8	Measurement Unit	To choose the measurement unit of page margin of the document: inch and cm. This option is only available to PDF and Word document.
9	Margin (Left/ Top/ Right/ Bottom)	Specify the width of spaces between the content and the page border.

The overview of Cover Page

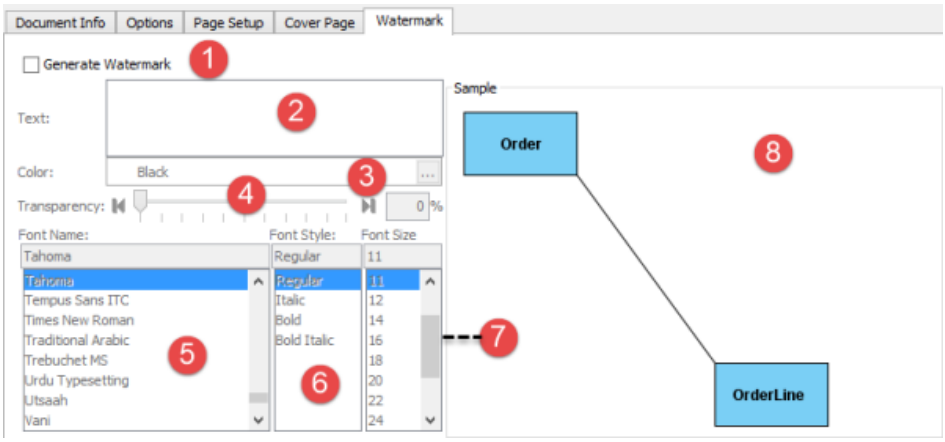


The overview of Cover Page tab

No.	Option	Description
1	Generate cover page	Check this option to generate a cover page to the document.
2	Logo image path	Insert an image to the cover page. You can specify the image's directory or select the directory by clicking the ... button.
3	Logo scale	Resize the inserted image by adjusting the slider.
4	Title	Specify the title of your document on cover page.
5	Organization name	Specify the organization name of your document on cover page.
6	Author name	Specify the author name on cover page.
7	Alignment	Control the position of content, whether to appear on the left, center or right hand side of the page.
8	Cover Page Preview	You can preview your cover page here.

Description of Cover Page tab

The overview of Watermark



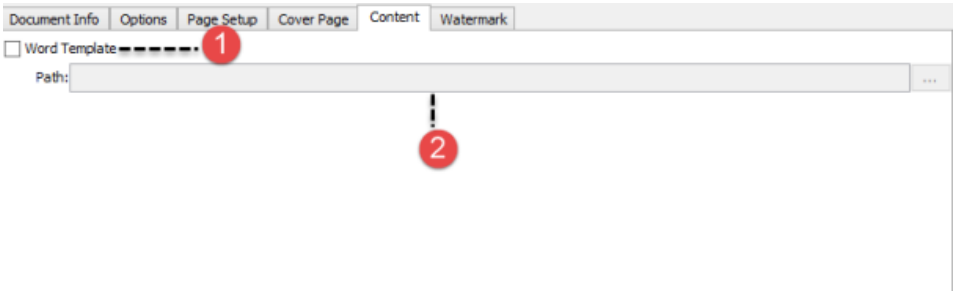
The overview of Watermark tab

No.	Option	Description
1	Generate Watermark	Check this option to generate watermark on all diagrams of document.
2	Text	Specify the text will be used for watermark.
3	Color	Specify the color of text will be used for watermark by clicking the **...** button.
4	Transparency	To change the background transparency for watermark, move the **Transparency** slider or specify the percentage of transparency directly.
5	Font Name	Select the font name for watermark.
6	Font Style	Select the font style for watermark.

7	Font Size	Select the font size for watermark.
8	Sample	Preview watermark here.

Description of Watermark tab

The overview of Content (Only for Word document)



The overview of Content tab

No.	Option	Description
1	Word Template	Check this option to select a Word template for Word document.
2	Path	Specify the directory of word template by clicking the **...** button.

Description of Content tab

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Managing Element Templates in Team Environment

If your team is using VPository.com or Teamwork Server as collaborative modeling solution, you can share element templates among team members with the built-in management and synchronization features. Doing so allows the entire team to compose document based on a common set of element templates. Besides, this ensures the completeness of document when being viewed in any member's environment because all members have access to the same and most updated templates needed by the documents.

In server, element templates are stored in repository based. This means that all of your projects managed under the same repository have access to the same set of element templates. In this page, you will learn how to manage those element templates and share them among team members.

Managing element templates

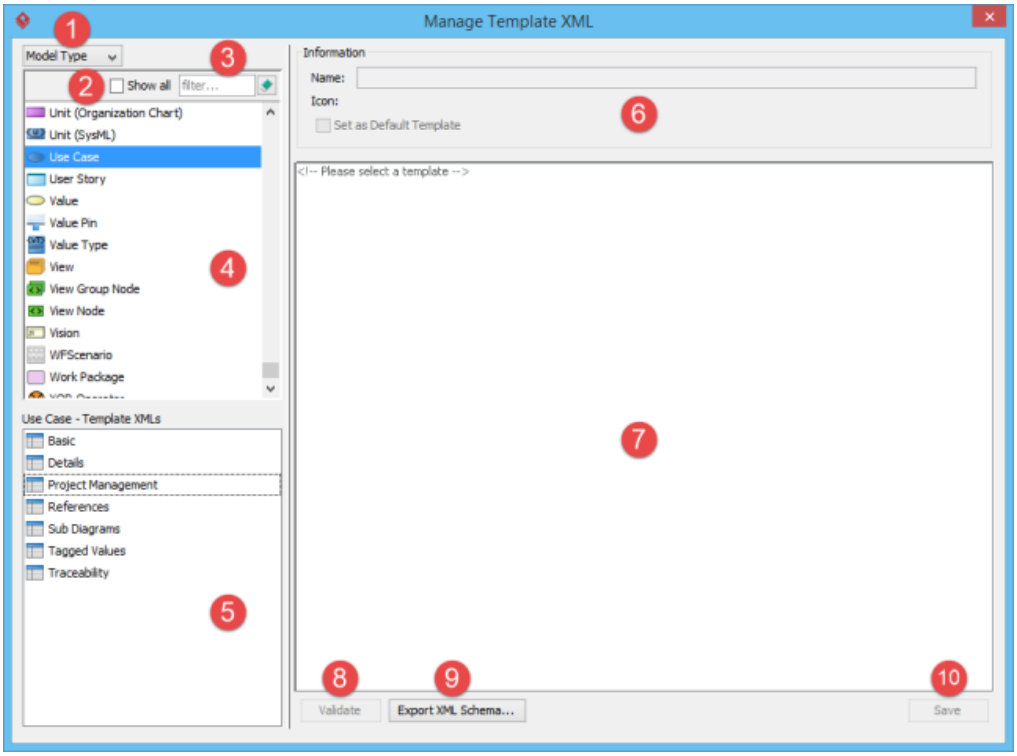
Manage element templates is the process to create, edit or delete element templates stored in repository. Once you have made the desired changes in Visual Paradigm locally, you can synchronize the changes to server. Teammates can get the updated templates by synchronizing changes to server as well.

As said earlier, element templates are stored in repository based. Therefore, no matter which project you have opened, you are managing the same set of element templates.

To manage element templates:

1. In Visual Paradigm, select **Tools > Doc. Composer > Manage Template XMLs...** from the toolbar. In order to access the management function, make sure you are opening a team project managed under either VPository.com or Teamwork Server. Besides, make sure you are a team member and have been granted the right to **Change document template** in server. You may need to contact your server administrator to confirm the permission settings made in server.
2. Now, you can manage element templates in the **Manage Template XML** window. Read the next section for details about what you can do in the **Manage Template XML** window.

Overview of Manage Template XML window



Overview of Manage Template XML window

No.	Name	Description
1	Type of project data	Different templates are available for different project data. The drop-down menu there divide project data into four main types: Project - The entire project. You will see element templates mainly for querying details of diagrams in project. Diagram Type - The available types of diagram. (e.g. Use Case Diagram, Business Process Diagram) Model Type - The available types of model elements. (e.g Use Case, Class) General - Mainly for listing legacy or obsolete element templates. Normally you don't need to deal with templates under General section.
2	Show all	Visual Paradigm supports a large volume of model elements, but not all of them are well known or meaningful. By default, we hide away those elements that aren't popular. If you need to edit their templates you can check **Show all** to reveal them.
3	Filter	Filter the items to be displayed in element list.
4	List of elements	The elements that support template editing. If you have chosen to list **Diagram Type** (in **Type of project data**), you will see a list of diagram here. If you have chosen to list **Model Type**, you will see a list of model elements here.

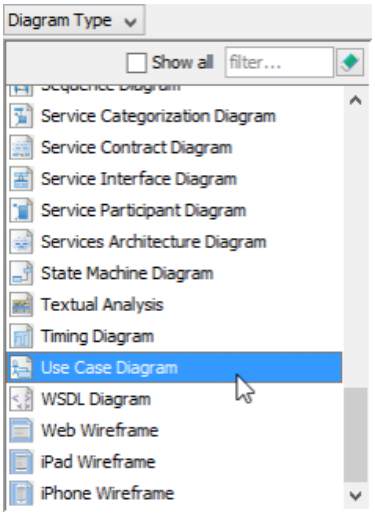
5	List of templates	A list of element templates available for the element selected in the list of elements. Each type of project data has its own set of element templates. Take Use Case Diagram as example, you have templates like Basic, Details, Project Management, etc. With a different selection of element, a different list of element templates will be presented.
6	Background information	Background information of an element template. Note that you can only edit background information of a user-defined template, but not any built-in template. Here is a description of properties you can set: **Name**: The name of the element template, which is the name that shown under the Element Template Pane. **Icon**: An icon that best represent the layout of content that will be produced by using this template. **Set as Default Template**: Check this option if you want Doc. Composer to apply this template automatically when dragging elements directly from Diagram Navigator / Model Explorer onto document. **Template content**: Editor for programming the template.
7	XML Editor	Customize your template in the XML editor. Again, you can only edit a user-defined template, but not any built-in template.
8	Validate	Validate the XML against the built-in XML schema.
9	Export XML Schema	Export the XML schema (*.xsd) for validating the XML template content.
10	Save	Save the modifications made in XML editor.

Description of Manage Template XML window

Creating a template

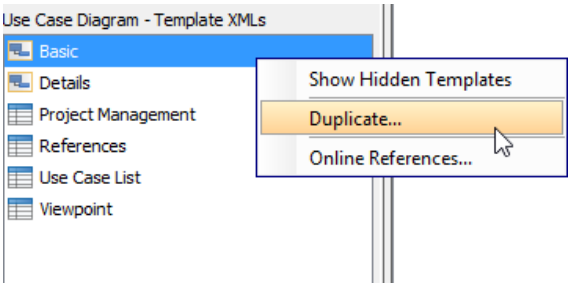
To simplify the programming of template, you are suggested to duplicate an existing template and start editing it, rather than do everything from scratch. Target on a template that gives the closest outcome to what you want to show in document. If you want to start from an empty document, select any templates.

1. Select the type of element to create template. For example, select **Use Case Diagram** if you want to create a template to list specific shapes in use case diagram. You can select project / diagram type / model element type.



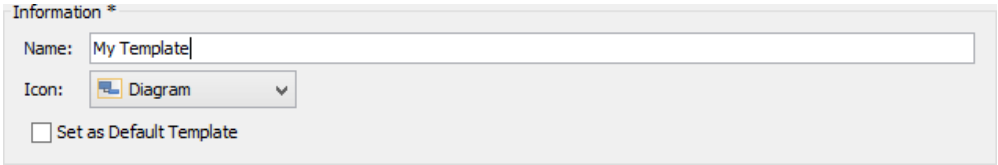
Selecting a use case diagram

2. Right click on an element template and select **Duplicate...** from the popup menu.



Duplicate a template

3. Enter the name of the new template.



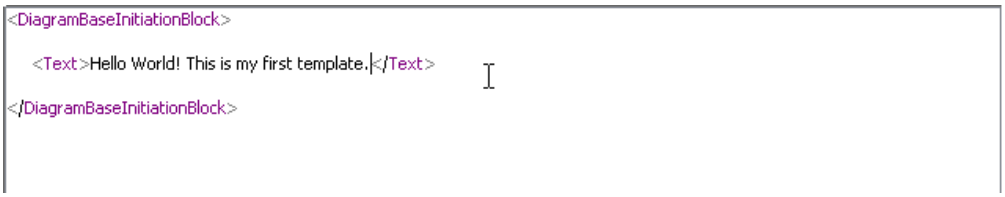
Naming a template

4. Choose an icon that represents the presentation of content output with your template.

Choosing an icon for a template

5. Check **Set as Default Template** if you want Doc. Composer to apply this template automatically when dragging elements directly from Diagram Navigator / Model Explorer onto document.

6. Compose the template in XML editor. If part of your template references content written in another element template, you can click + to add a reference to that template. Click here for details about template referencing.

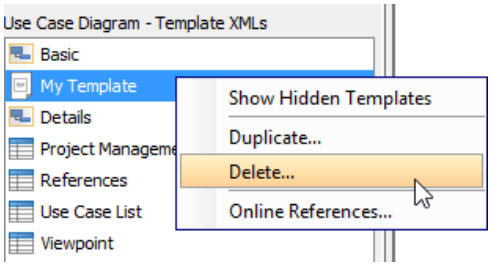


Editing a template in XML editor

7. Click **Save** when finished editing. Now, you can use the new template in your document. You can also share it with teammates.

Deleting a template

Right click on an element template and select **Delete...** from the popup menu to remove it. Note that this action cannot be undone. Moreover, documents that used a deleted template in content will have the missing parts be replaced by <empty> tag(s). This may severely affect the completeness of your document so think twice before you delete a template. Make sure the template is not currently in-used by any document, or the documents that use the templates are not important anymore.



To delete a template

NOTE: You can only delete a user-defined template.

Modifying a template

Click on the template in the list of template list and then modify it in XML editor. Click **Save** when finished editing. When finished, you can refresh your document to apply the changes.

Synchronizing element templates

Once you have finished editing element templates, you can synchronize the changes to server. Teammates can get the updated templates by synchronizing changes to server as well.

To synchronize changes to server manually, select **Tools > Doc. Composer > Sync. to VPository/VP Teamwork Server** from the toolbar.

Note that your changes will be synchronized automatically when you perform commit.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm

- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Managing Styles in Team Environment

If your team is using VPository.com or Teamwork Server as collaborative modeling solution, you can share styles configuration among team members with the built-in management and synchronization features. Doing so allows the entire team to compose document based on a common set of styles configuration. Besides, this ensures that documents are always up-to-date when being viewed in any member's environment because all members have access to the most updated styles configuration .

In server, styles configuration are stored in repository based. This means that all of your projects managed under the same repository have access to the same set of styles configuration. In this page, you will learn how to manage those styles and share them among team members.

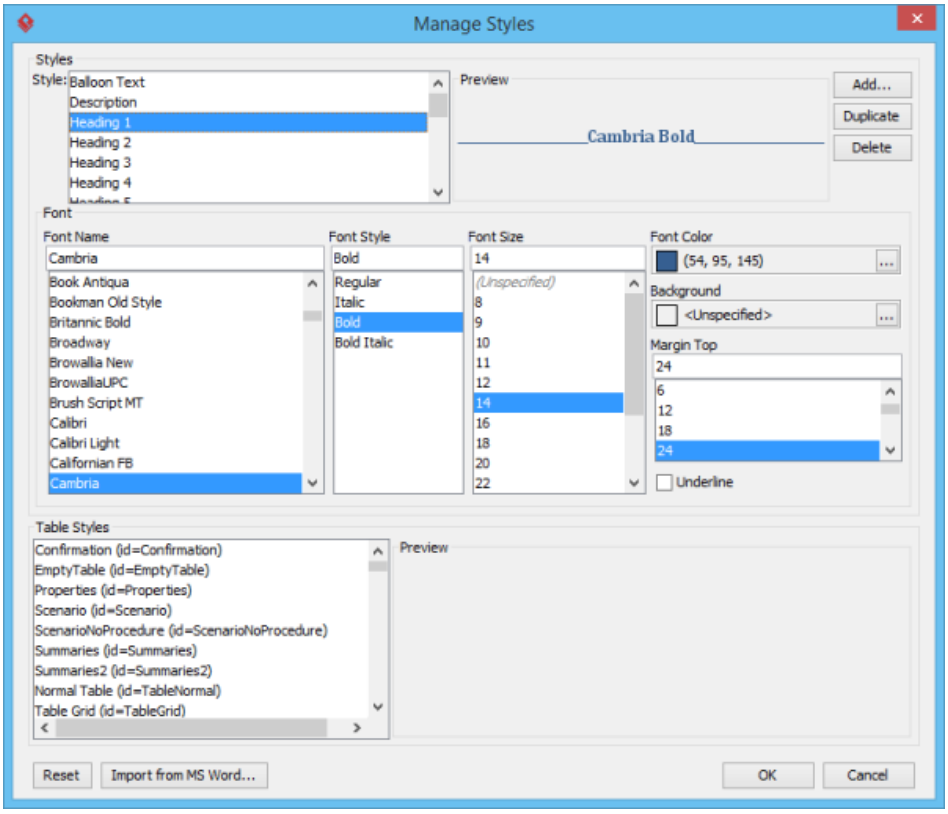
Managing Styles

Manage styles is the process to create, edit or delete styles configuration stored in repository. Once you have made the desired changes in Visual Paradigm locally, you can synchronize the changes to server. Teammates can get the updated styles configuration by synchronizing changes to server as well.

As said earlier, styles configuration are stored in repository based. Therefore, no matter which project you have opened, you are managing the same set of styles.

To manage styles:

1. In Visual Paradigm, select **Tools > Doc. Composer > Manage Styles...** from the toolbar. In order to access the management function, make sure you are opening a team project managed under either VPository.com or Teamwork Server. Besides, make sure you are a team member and have been granted the right to **Change document template** in server. You may need to contact your server administrator to confirm the permission settings made in server.
2. Now, you can manage styles in the **Manage Styles** window.



Manage Styles window

Synchronizing styles configuration

Once you have finished editing styles configuration, you can synchronize the changes to server. Teammates can get the updated configuration by synchronizing changes to server as well.

To synchronize changes to server manually, select **Tools > Doc. Composer > Sync. to VPository/VP Teamwork Server** from the toolbar.

Note that your changes will be synchronized automatically when you perform commit.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition

- Professional Edition
- Enterprise Edition

Doc. Composer - Fill-in Doc

Introduction to Doc. Composer's Fill-in Doc Mode

The Fill-in Doc mode of Doc. Composer is designed to help you "fill-in" the design details of your documentation. This article will show you how it works.

Understanding Doc Base

Let Doc. Composer fill-in your project document with the use of Doc Base. You will learn what Doc Base is in this article.

Understanding Doc Field

Add special fields into a Doc Base. Let Doc. Composer replaces the content for you with real design specification.

Creating a Fill-in Doc

In this page you will learn how to create a fill-in doc in Doc. Composer.

Touching-Up a Document

Open a Doc Base, and then touch-up the document in Doc. Composer before exporting. You will learn how it works in this article.

Previewing a Document

Take a quick look at the document file that will be generated with the applied Doc Base.

Generating a Document

Generate your project documentation from Doc. Composer in Visual Paradigm.

Doc Fields in Detail

Check out a list of available Doc. Fields. Learn their syntax and know how to write Doc Fields in your document.

Querying Diagrams

Read this section to learn how to retrieve a diagram or diagrams, and to output content like the diagram's image, name or a list of containing diagram elements, etc.

Querying Model Elements

Learn how to retrieve a model element or elements, and to output content like the element's name, description, or a list of member elements (e.g. attributes of class, columns of entity), etc.

Querying Diagram Elements

Learn the ways to retrieve a shape or shapes, and to output content like the shape's name, description, or a list of member elements (e.g. attributes of class, columns of entity), etc.

Using Custom Text

${TEXT} field is a placeholder of content that can only be provided when generating a document, such as project name or author name. You will learn the use of such a Doc Field in this article.

Working with Table

In this section we will introduce the various kinds of table you can create in a Doc Base, and explain how to create such tables by writing Doc Fields.

Introduction to Doc. Composer's Fill-in Doc Mode

Typically, a project documentation or report is a combination of background information like project goal, scope and constraints, and design details like use case details, database design, process design, etc. The Fill-in Doc mode of Doc. Composer is designed to help you "fill-in" the design details of your documentation. As an overview, Fill-in Doc works in this way:

1. You write your project documentation in Microsoft Word.
2. Place some special fields to the parts that requires the insertion of design details.
3. Load the document into Doc. Composer
4. Export a document. Doc. Composer analyze the special fields in your document and replace them with actual project content.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Understanding Doc Base

A Doc Base is a semi-completed version of your project documentation or report. It contains only background information, possibly filled by you or your colleague. The design details are leave empty and be filled by Doc. Composer. As its name suggested, Doc Base provides a base for documentation production. You provide such a base to Doc. Composer and then Doc. Composer fill the empty parts for you by embedded model data extracted from your Visual Paradigm project into your documentation.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Understanding Doc Field

A Doc Field is a special piece of text within a Doc Base. Doc Fields will be replaced by your actual project content when being read by Doc. Composer during document exporting. Here is an example of Doc Field:

```
${DIAGRAM, "List of use case diagrams", "UseCaseDiagram", LoopInProject, PROPERTY=name}
```

A Doc Field is written in this format: ${list_of_parameters}. We will talk about the parameters in detail in coming sections.

Simply speaking, if you create a Word document, type the above text into the document, save the document as a Word file, import the file into Doc. Composer as a Doc Base and then export a document from Doc. Composer, the exported document will look like this:

```
Use Case Diagram1, Use Case Diagram2, Use Case Diagram3
```

Here we assume that your project contains three use case diagrams, namely Use Case Diagram1, Use Case Diagram2, Use Case Diagram3.

This is how Doc Field basically works – You type the field text into your document at the places where you need to embed your project content, save the document, import it into Doc. Composer and let it produce a new document by replacing those fields with project content.

There are six kinds of fields. The following gives you the basic ideas of each fields, along with their required formats and capabilities. The detailed usage of these fields will be covered in next big sections.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

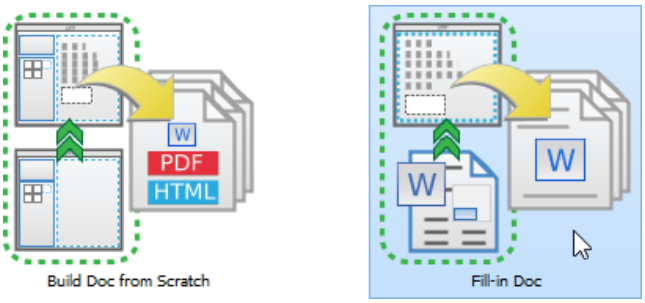
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

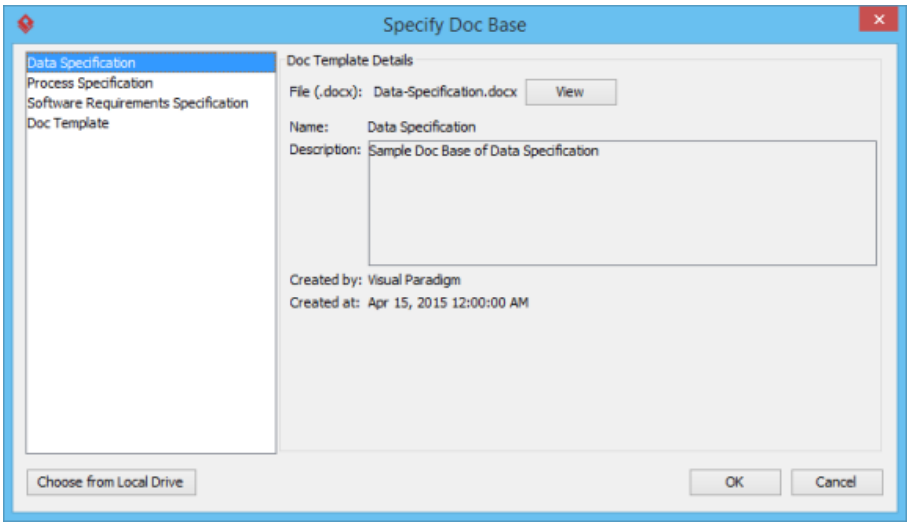
Creating a Fill-in Doc

1. Select **Tools** > **Doc. Composer** from the toolbar.
2. Click **Fill-in Doc**. This shows the **Specify Doc Base** window.



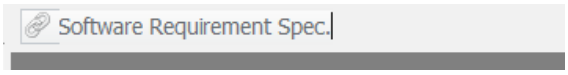
Creating a Fill-in Doc

3. Specify the Doc Base to use in document production. Doc base is a Word document file that contains both manually written content (e.g. Introduction, project scope, etc) and Doc Fields. If you are unclear about Doc Base and Doc Field, read the previous sections. There are two approaches from which Doc Base can be created from. One is to create from an external document file. If you take this approach, click **Choose from Local Drive** and then select the document file (*.docx). Another approach is to duplicate from an existing Doc Template. If you take this approach, select the Doc Template from the template list and click **OK**. You will then be prompted to save a copy of the Doc Template to your computer as Doc Base. Visual Paradigm provides three default templates for you to choose from. If your team uses Teamwork Server or VPository.com, you can create your own set of templates and share them among the team.



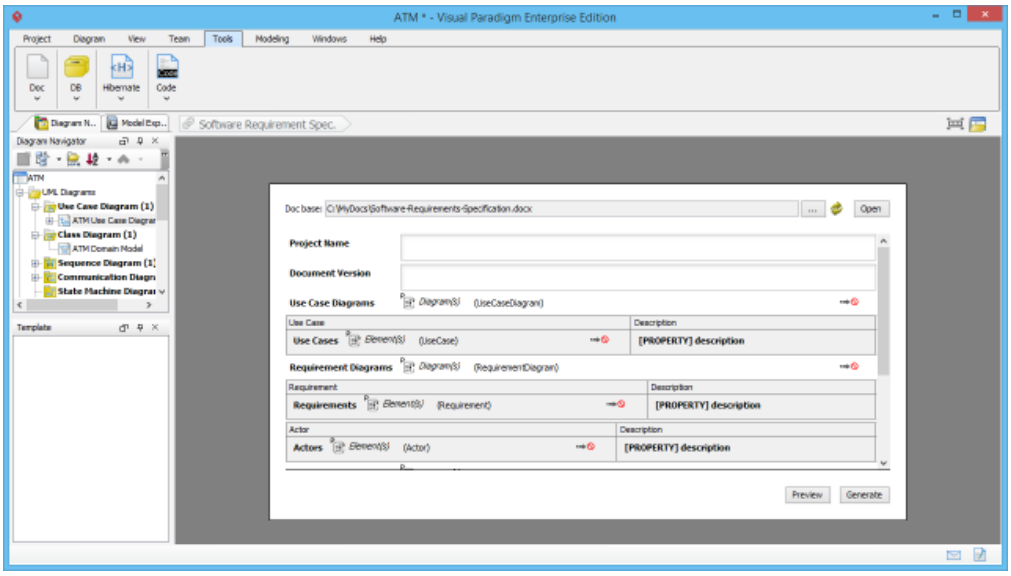
Specify Doc Base

4. If necessary, rename the document by double clicking on its name in breadcrumb and then typing in a new name.



Entering document name

5. Press the **Enter** key to confirm the naming.
6. Doc. Composer analyzes your Doc Base and presents the Doc Fields that exist in your document. Your screen should look like this:



Doc Base opened

To have more editing space, we recommend you to collapse the toolbar temporarily by double clicking on the **Tools** tab.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Touching-Up a Document

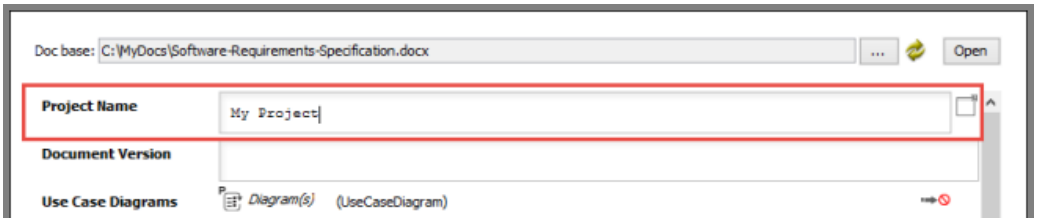
If your Doc Base contains any of the following kinds of Doc Fields, you have to touch-up the document in order to export a document file from Doc. Composer.

- Doc Field with Any as source
- Doc Field with One as source
- Doc Field with LoopInElement as source
- Doc Field with LoopInDiagram as source
- ${TEXT} field

To "touch-up" a document means to select diagram(s) or model element(s), or to enter the content required by Doc Fields in a document. For example, if in a Doc Base there exist a ${TEXT} like this:

```
<${TEXT, "Project Name"}>
```

You'll have to provide the project name in Doc. Composer. Here is what you will see in Doc. Composer:



Entering a {TEXT} in Doc. Composer

In the example above, we have entered My Project as the project name.

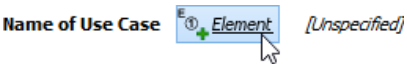
Another example would be the use of ${ELEMENT} field, with One as source:

```
${ELEMENT, "Name of Use Case", "UseCase", One, PROPERTY=name}
```

This example means to output the name of a use case to the document and such a use case shall be specified in Doc. Composer. Here is how the Doc. Composer will look like when applying a Doc Base that contains such an ${ELEMENT} field.

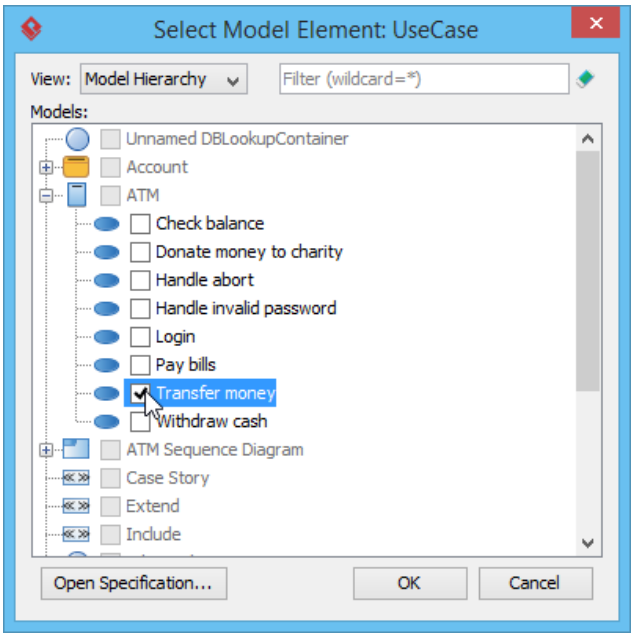
Appearance of an ${ELEMENT} field in Doc. Comoposer

What you need to do is to click on the Element link. Note that the title of this link varies depending on the type of field and source specified.



To select a model element

Then, select the desired element(s) and click OK to confirm.



Selecting a use case

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Previewing a Document

If you want to take a quick look at the document file that will be generated with the applied Doc Base, click Preview at the bottom right corner of the document preview. A temporary document file will be opened for you to preview the outcome.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Generating a Document

When you are ready for producing the final document, click Generate at the bottom right corner of the document preview. Enter the filename of the document and confirm. A complete document file will then be generated.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Doc Fields in Detail

${DIAGRAM}

The **${DIAGRAM}** field is used to query diagram(s) from a project (or a specific place in a project), and to output content from the diagrams querying.

Here is an example of **${DIAGRAM}**:

```
${DIAGRAM, "List of Use Case Diagrams", "UseCaseDiagram", LoopInProject, PROPERTY=name}
```

Here is the sample output:

```
Use Case Diagram1, Use Case Diagram2, Use Case Diagram3
```

This is the syntax of a **${DIAGRAM}** field:

```
${DIAGRAM,
  field_name ,
 [" diagram_type {, diagram_type ...}", ]
 One | Any | LoopInProject | LoopInElement,
  template_name | PROPERTY= property_name | ICON | IMAGE
}
```

This is a description of the various parts of a **${DIAGRAM}** field:

- DIAGRAM is to indicate that this is a **${DIAGRAM}** field.
- ***field_name*** is a short description of the field (e.g. "List of Use Case Diagrams"). In Doc. Composer, you can see the fields placed in an imported Doc Base. The fields are represented by the ***field_name*** typed here. ***field_name*** must be unique within a Doc Base. If there are two or more fields having the same field name, content may be produced wrongly.
- ***diagram_type*** indicates the type(s) of diagram that you want to query (e.g. "UseCaseDiagram"). If you want to query all types of diagram in a project, skip this parameter. If you want to query multiple types of diagram, enter their types respectively, separated by comma (e.g. "ClassDiagram,UseCaseDiagram"). Click here for the proper diagram types to use.
- One | Any | LoopInProject | LoopInElement indicates the source from which to query diagrams.
  - One – Query a **specific** diagram in project. This option is often used when your document, or part of the document is written around a specific diagram. If you choose One here, you will have to select in Doc. Composer the diagram to query.
  - Any – Query a number of diagrams in project. If you choose Any here, you will have to select in Doc. Composer the diagram to query.
  - LoopInProject – Query all the diagrams in project.
  - LoopInElement – Query the sub-diagram(s) of a specific model element. If you choose LoopInElement here, you will have to select in Doc. Composer the model element from which to query sub-diagrams.
- **template_name** | PROPERTY= ***property_name*** | ICON | IMAGE – The type of content to be extracted from the querying diagrams and printed on the document.
  - ***template_name*** – Output content from each of the querying diagrams, based on the template ***template_name*** written for the type of the querying diagram. For example, if you have a template *AllUseCases* written for Use Case Diagram, by specifying *AllUseCases* as ***template_name*** , Doc. Composer will output content for each of the Use Case Diagrams by following *AllUseCases*.
  - PROPERTY= ***property_name*** – Output a **specific** property (e.g. description) for each of the querying diagrams. If you want to output multiple properties, try write a template and make reference to it by providing its name here.
  - ICON – Output the icon for each of the querying diagrams.
  - IMAGE – Output the diagram image for each of the querying diagrams.

${ELEMENT}

The **${ELEMENT}** field is used to query model element(s) or diagram element(s) from a project (or a specific place in a project), and to output content from the elements querying.

Here is an example of **${ELEMENT}**:

```
${ELEMENT, "List of Use Cases", UseCase, LoopInProject, PROPERTY=name}
```

Here is the sample output:

```
Use Case1, Use Case2
```

This is the syntax of a **${ELEMENT}** field:

```
${ELEMENT,
  field_name ,
 [" element_type {, element_type ...}", ]
 One | Any | LoopInProject | LoopInElement | LoopInDiagram,
  template_name | PROPERTY= property_name | ICON
}
```

This is a description of the various parts of a **${ELEMENT}** field:

- ELEMENT is to indicate that this is a **${ELEMENT}** field.

- ***field_name*** is a short description of the field (e.g. "List of Use Cases"). In Doc. Composer, you can see the fields placed in an imported Doc Base. The fields are represented by the ***field_name*** typed here. ***field_name*** must be unique within a Doc Base. If there are two or more fields having the same field name, content may be produced wrongly.
- ***element_type*** indicates the type(s) of model/diagram element that you want to query (e.g. "UseCase"). If you want to query all types of model element in a project, skip this parameter. If you want to query multiple types of model element, enter their types respectively, separated by comma (e.g. "Actor,UseCase").
- One | Any | LoopInProject | LoopInElement | LoopInDiagram indicates the source from which to query elements.
  - One – Query a **specific** model element in project. This option is often used when your document, or part of the document is written around a specific model element. If you choose One here, you will have to select in Doc. Composer the model element to query.
  - Any – Query a number of model elements in project. If you choose Any here, you will have to select in Doc. Composer the model element to query.
  - LoopInProject – Query all the model elements in project.
  - LoopInElement – Query the child elements of a specific model element. If you choose LoopInElement here, you will have to select in Doc. Composer the model element from which to query child elements.
  - LoopInDiagram – Query the diagram elements from a specific diagram. If you choose LoopInDiagram here, you will have to select in Doc. Composer the diagram from which to query diagram elements.
- ***template_name*** | PROPERTY= ***property_name*** | ICON | IMAGE – The type of content to be extracted from the querying elements and printed on the document.
  - ***template_name*** – Output content from each of the querying elements, based on the template ***template_name*** written for the type of the querying element. For example, if you have a template *UseCaseInfo* written for Use Case, by specifying *UseCaseInfo* as ***template_name*** , Doc. Composer will output content for each of the Use Cases by following *UseCaseInfo*.
  - PROPERTY= ***property_name*** – Output a **specific** property (e.g. description) for each of the querying elements. If you want to output multiple properties, try write a template and make reference to it by providing its name here.
  - ICON – Output the icon for each of the querying elements.

${ICON}

When you are working with a table, you can place the **${ICON}** field in a table cell to let Doc. Composer replace it with the icon image of the querying diagram or element.

Note that **${ICON}** can only be used in a table cell.

Here is an example of **${ICON}**:

```
${ICON}
```

Here is the sample output:

This is the syntax of a **${ICON}** field:

```
${ICON}
```

${IMAGE}

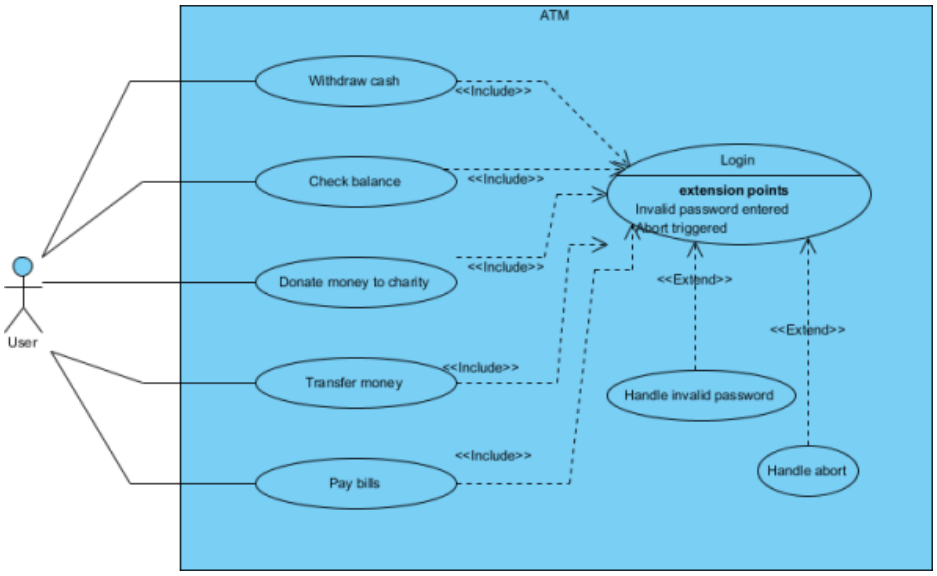
When you are working with a table, you can place the **${IMAGE}** field in a table cell to let Doc. Composer replace it with the diagram image of the querying diagram.

Note that **${IMAGE}** can only be used in a table cell.

Here is an example of **${IMAGE}**:

```
${IMAGE}
```

Here is the sample output:



Use Case Diagram

This is the syntax of an **${IMAGE}** field:

```
${IMAGE}
```

${PROPERTY}

When you are working with a table, you can place the **${PROPERTY}** field in a table cell to let Doc. Composer replace it with the property value of the querying diagram or element.

Note that **${PROPERTY}** can only be used in a table cell.

Here is an example of **${PROPERTY}**:

```
${PROPERTY}
```

Here is the sample output:

```
This is the description of use case.
```

This is the syntax of a **${ICON}** field:

```
${PROPERTY}
```

${TEXT}

The **${TEXT}** field is used when you need to include information that should be or can only be provided when generating document. A typical usage of **${TEXT}** is to request for project name.

In Doc. Composer, **${TEXT}** are represented as text fields. User can enter the value required by the **${TEXT}** field. When generating document, those **${TEXT}** will be replaced by the text entered.

Here is an example of **${TEXT}**:

```
${TEXT, "Project name"}
```

Here is the sample output:

```
Online Banking
```

This is the syntax of a **${TEXT}** field:

```
${TEXT,
  field_name
}
```

This is a description of the various parts of a **${TEXT}** field:

- TEXT is to indicate that this is a **${TEXT}** field.
- ***field_name*** is a short description of the field (e.g. "Project name"). It can also be used as a reminder to provide certain kind of information (e.g. "Please enter the project name."). ***field_name*** must be unique within a Doc Base. If there are two or more fields having the same field name, content may be produced wrongly.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube

- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Querying Diagrams

If you want to retrieve a diagram or diagrams, and to output content like the diagram's image, name or a list of containing diagram elements, etc., read this section to learn the ways to retrieve diagrams.

Querying Diagrams in Project

If you want to output the image or any detail of **all the diagrams in project**, write a **${DIAGRAM}** field in your Word document with **LoopInProject** specified as diagram source. Here are several examples of such a **${DIAGRAM}** field:

```
${DIAGRAM, "Name of ALL Use Case Diagrams", "UseCaseDiagram", LoopInProject, PROPERTY=name}
${DIAGRAM, "Description of ALL diagrams", , LoopInProject, PROPERTY=description}
${DIAGRAM, "Details of ALL Use Case Diagram and class diagram", "UseCaseDiagram,ClassDiagram", LoopInProject, MyTemplate}
```

In the first example, the name of all Use Case Diagrams in the project will be output. Note that "Name of ALL Use Case Diagrams" is the field name, which is a required and unique value for identifying this field.

In the second example, the description of all the diagrams in the project will be output.

In the third example, content will be output for each of the Use Case Diagrams and Class Diagrams in the project, based on the template *MyTemplate*.

Querying Selected Diagrams in Project

The diagrams in your project may be created for different contexts or about different problem domains. When you write a documentation, you may want to focus on a specific context at a time, which requires the insertion of design specification for that specific context. In that case, you will want to query a selected set of diagrams in your project, instead of querying all diagrams.

If you want to output the image or any detail of **selected diagrams in project**, write a **${DIAGRAM}** field in your Word document with **Any** specified as diagram source. Here are several examples of such a **${DIAGRAM}** field:

```
${DIAGRAM, "Use Case Diagrams (Admin)", "UseCaseDiagram", Any, PROPERTY=name}
${DIAGRAM, "Diagram Images", , Any, IMAGE}
${DIAGRAM, "Diagram Images", , Any, MyTemplate}
```

In the first example, the name of selected Use Case Diagrams in the project will be output. Note that "Use Case Diagrams (Admin)" is the field name, which is a required and unique value for identifying this field.

In the second example, the image of selected diagrams in the project will be output.

In the third example, content will be output for each of the selected diagrams in the project, based on the template *MyTemplate*.

When you pick-up a Doc Base with such a **${DIAGRAM}** field in it, you can select the diagrams to query in Doc. Composer.

Querying Specific Diagram in Project

Let's say you have created multiple Use Case Diagrams for multiple sub-systems. When you write a documentation for a specific sub-system, you may want to insert the design specification related to that specific sub-system. In that case, you will want to query a specific Use Case Diagram in your project.

If you want to output the image or any detail of a **specific diagram in project**, write a **${DIAGRAM}** field in your Word document with One specified as diagram source. Here are several examples of such a **${DIAGRAM}** field:

```
${DIAGRAM, "ATM Overview", "UseCaseDiagram", One, PROPERTY=description}
${DIAGRAM, "Online Photo Album", , One, IMAGE}
${DIAGRAM, "CS System &ndash; Use Cases", , One, Children}
```

In the first example, the description of selected Use Case Diagram in the project will be output. Note that "ATM Overview" is the field name, which is a required and unique value for identifying this field.

In the second example, the image of selected diagram in the project will be output.

In the third example, content will be output for the selected diagram in the project, based on the template *Children*.

When you pick-up a Doc Base with such a **${DIAGRAM}** field in it, you can select the diagram to query in Doc. Composer.

Querying Sub-Diagrams from Specific Model Element

Let's say you have created several use cases and, for each use case, there are multiple sub-Business Process Diagrams that describe the business workflow in which the use case might happen. When you write a use case report, you may want to present the details of the Business Process Diagrams of a chosen use case. In that case, you will want to query the sub-diagrams of a selected use case.

If you want to output the image or any detail of **sub-diagrams from a specific model element**, write a **${DIAGRAM}** field in your Word document with **LoopInElement** specified as diagram source. Here are several examples of such a **${DIAGRAM}** field:

```
${DIAGRAM, "Related Business Workflow", "BusinessProcessDIagram", LoopInElement, PROPERTY=name}
${DIAGRAM, "Images of Sub-Diagrams", , LoopInElement, IMAGE}
${DIAGRAM, "Sub Diagram Details", , LoopInElement, MySubDiagrams}
```

In the first example, the name of sub-Business Process Diagrams of the selected model element will be output. Note that "Related Business Workflow" is the field name, which is a required and unique value for identifying this field.

In the second example, the image of sub-diagrams of a selected model element will be output.

In the third example, content will be output for each of the sub-diagrams of a selected model element, based on the template *MySubDiagrams*.

When you pick-up a Doc Base with such a **${DIAGRAM}** field in it, you can select the model element to query in Doc. Composer.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Querying Model Elements

If you want to retrieve a model element or elements, and to output content like the element's name, description, or a list of member elements (e.g. attributes of class, columns of entity), etc., read this section to learn the ways to retrieve model elements.

Querying Model Elements in Project

If you want to output any detail of **all the model elements in project**, write an **${ELEMENT}** field in your Word document with **LoopInProject** specified as element source. Here are several examples of such an **${ELEMENT}** field:

```
${ELEMENT, "Name of ALL Use Cases", "UseCase", LoopInProject, PROPERTY=name}
${ELEMENT, "Description of ALL model elements", , LoopInProject, PROPERTY=description}
${ELEMENT, "Details of ALL Use Cases and Business Processes", "UseCase,BPTask,BPSubProcess", LoopInProject, MyTemplate}
```

In the first example, the name of all use cases in the project will be output. Note that "Name of ALL Use Cases" is the field name, which is a required and unique value for identifying this field.

In the second example, the description of all the model elements in the project will be output.

In the third example, content will be output for each of the use cases, BPMN tasks and sub-processes in the project, based on the template *MyTemplate*.

Querying Selected Model Elements in Project

The model elements in your project may be created for different contexts or purposes. When you write a documentation, you may want to focus on some of them at a time. In that case, you will want to query a selected set of model elements in your project, instead of querying all elements.

If you want to output any detail of **selected model elements in project**, write an **${ELEMENT}** field in your Word document with **Any** specified as element source. Here are several examples of such an **${ELEMENT}** field:

```
${ELEMENT, "Use Cases", "UseCase", Any, PROPERTY=name}
${ELEMENT, "Elements' Detail", , Any, MyTemplate}
```

In the first example, the name of selected use cases will be output. Note that "Use Cases" is the field name, which is a required and unique value for identifying this field.

In the second example, content will be output for each of the selected model elements, based on the template *MyTemplate*.

When you pick-up a Doc Base with such an **${ELEMENT}** field in it, you can select the model elements to query in Doc. Composer.

Querying Specific Model Element in Project

If you need to write a document for a specific model element, like a use case report, you may need to insert the details of a specific use case into your document. In that case, you will want to query a specific model element in your project, instead of querying all model elements.

If you want to output any detail of a **specific model element in project**, write an **${ELEMENT}** field in your Word document with **One** specified as element source. Here are several examples of such an **${ELEMENT}** field:

```
${ELEMENT, "Desc of Use Case", "UseCase", One, PROPERTY=description}
${ELEMENT, "List of Class Members", "Class", One, Children}
```

In the first example, the description of selected use case will be output. Note that "Desc of Use Case" is the field name, which is a required and unique value for identifying this field.

In the second example, content will be output for the selected class, based on the template Children.

When you pick-up a Doc Base with such a **${DIAGRAM}** field in it, you can select the model element to query in Doc. Composer.

Querying Model Elements from Specific Model Element

If you need to write a document for a specific model element by detailing its children elements, like a business responsibility report that details the tasks contained by specific pool, or a use case report that details the use cases contained by a system, you will want to query the children elements of a selected model element.

If you want to output any detail of **model elements from a specific model element**, write an **${ELEMENT}** field in your Word document with **LoopInElement** specified as element source. Here are several examples of such an **${ELEMENT}** field:

```
${ELEMENT, "List of System Use Cases", "UseCase", LoopInElement, PROPERTY=name}
${ELEMENT, "List of Elements (Any type)", , LoopInElement, PROPERTY=name}
```

In the first example, the name of the children use cases of selected element will be output. Note that "List of System Use Cases" is the field name, which is a required and unique value for identifying this field.

In the second example, the name of all children elements of selected element will be output.

When you pick-up a Doc Base with such an **${ELEMENT}** field in it, you can select the model element to query in Doc. Composer.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Querying Diagram Elements

If you want to retrieve a shape or shapes, and to output content like the shape's name, description, or a list of member elements (e.g. attributes of class, columns of entity), etc., read this section to learn the ways to retrieve shapes (i.e. diagram elements).

Querying Diagram Elements from Specific Diagram

If you need to write a document for a specific diagram, like an ERD report, you may need to insert the details of the containing shapes into your document. In that case, you will want to query diagram elements in a diagram.

If you want to output any detail of a **diagram elements in a diagram**, write an **${ELEMENT}** field in your Word document with **LoopInDiagram** specified as element source. Here are several examples of such an **${ELEMENT}** field:

```
${ELEMENT, "Tables in ERD", "DBTable", LoopInDiagarm, PROPERTY=name}
${ELEMENT, "List of Classes", "Class", LoopInDiagram, Details}
```

In the first example, the name of entities on a selected ERD will be output. Note that "Tables in ERD" is the field name, which is a required and unique value for identifying this field.

In the second example, content will be output for classes in the selected diagram, based on the template *Children*.

When you pick-up a Doc Base with such an **${ELEMENT}** field in it, you can select the diagram to query in Doc. Composer.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Using Custom Text

If you want to request the user of Doc Base to fill-in certain piece of content himself/herself, write a **${TEXT}** field. A **${TEXT}** field is a placeholder of content that can only be provided when generating a document, such as project name or author name. Here is an example of a **${TEXT}** field:

```
${TEXT, "Project Name"}
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Working with Table

You can present project data neatly with the use of table. In this section we will introduce the various kinds of table you can create in a Doc Base, and explain how to create such tables by writing Doc Fields. We assume that you have the basic knowledge of Doc Field. If you don't, please read the previous sections.

Let's begin by studying the following example, which consists of a table in a Doc Base, with an ${ELEMENT} Doc Field placed in the second row of the table.

Use Cases in Project
${ELEMENT, "List of Use Cases", UseCase, LoopInProject, PROPERTY=name}

Suppose the Doc Base is applied on a project that contains the design specification of an ATM. Here is the sample outcome:

Use Cases in Project
Withdraw Cash
Transfer Cash
Donate Money
Pay Bills

Based on the outcome, you can see:

- The row that contains the Doc Field replicate itself to list out all the elements queried.
- In each row, the name of use case is output, which is the result of using **PROPERTY=name** in the Doc Field.

While this example output the name of use case, you can output complex content with the use of an element template. You just need to replace **PROPERTY=name** with the name of that template.

The example above is perhaps a bit simple. Let's extend it to make it a bit more complicated and closer to practical usage. Let's study this table:

Use Cases in Project	ID	Description
${ELEMENT, "List of Use Cases", UseCase, LoopInProject, PROPERTY=name}	**${PROPERTY, "userID"}**	${PROPERTY, "description"} ***(to be confirmed)***

We have added two more columns into the table, one for displaying the ID of use cases and another for displaying the description of use cases. Again, if we apply the Doc Base on an ATM project, here is the sample outcome:

Use Cases in Project	ID	Description
Withdraw Cash	**UC01**	Get cash from the ATM. ***(to be confirmed)***
Transfer Cash	**UC02**	Transfer cash from one account to another. ***(to be confirmed)***
Donate Money	**UC03**	Donate money to a chosen charity ***(to be confirmed)***
Pay Bills	**UC04**	Settle bills ***(to be confirmed)***

Based on the outcome, you can see:

- In order to output multiple properties of an element, add extra columns into the table and use ${PROPERTY} to output those properties.
- You can format table content, like the green text you see above.
- You can add your own text into table cells.

You may also want to present the icons of querying element. The following example gives you some ideas how to achieve it.

Model Elements in a Class Diagram	Or if you want a column of icons
${ELEMENT, "List of Model Elements", , LoopInDiagram, ICON} ${PROPERTY, "name"}	${ICON}

Here is the sample output when applying the above Doc Base on an ATM project, with a class diagram chosen to be the source of elements to query.

Model Elements in a Class Diagram	Or if you want a column of icons
Account	
atm	
Transaction	
User	

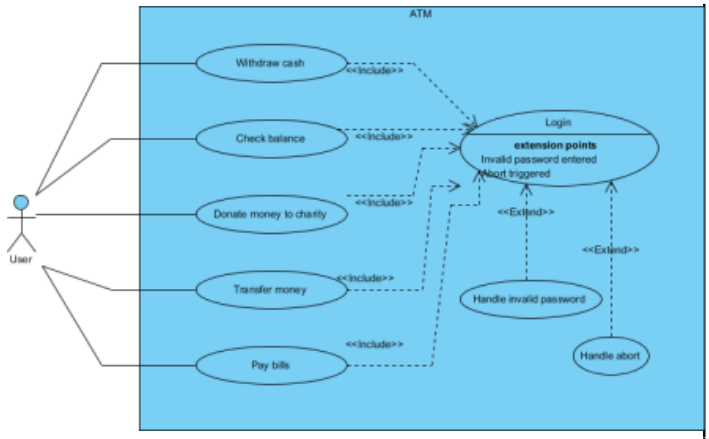
You may also output diagram images using ${IMAGE}. Here is an example:

All Diagrams in Project	Diagram Image
${DIAGRAM, "All Diagrams", , LoopInProject, PROPERTY=name}	${IMAGE}

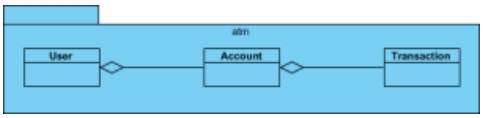
Here is the sample output when applying the above Doc Base on an ATM project.

All Diagrams in Project	Diagram Image

ATM Use Case Model



Domain Class Model



NOTE: ${PROPERTY}, ${ICON} and ${IMAGE} can only be used inside a table cell.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Managing Doc Templates in Team Environment

If your team is using VPository.com or Teamwork Server as collaborative modeling solution, you can share Doc Templates among team members with the built-in management and synchronization features. Doing so allows the entire team to compose document based on a common set of Doc Templates. Besides, this ensures that documents are always up-to-date when being viewed in any member's environment because all members have access to the most updated templates.

In server, Doc Templates are stored in repository based. This means that all of your projects managed under the same repository have access to the same set of Doc Templates. In this page, you will learn how to manage those Doc Templates and share them among team members.

Managing Doc Templates

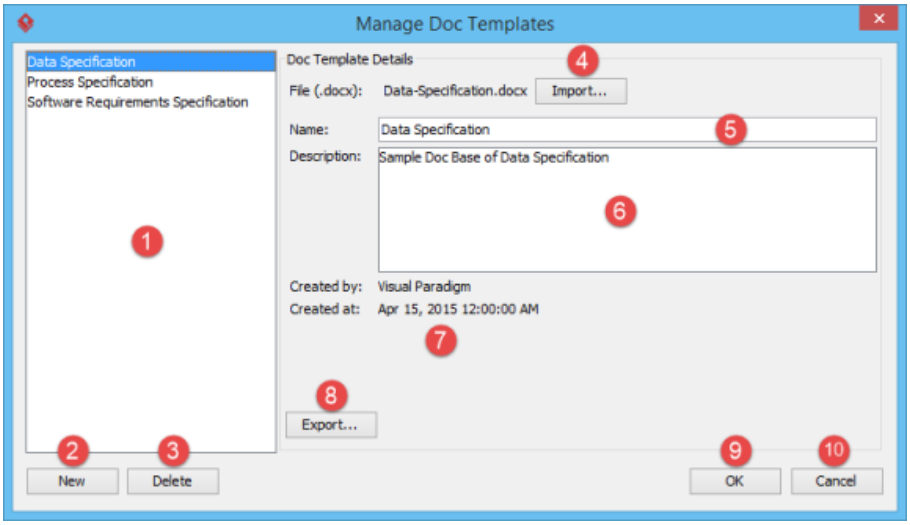
Manage Doc Templates is the process to create, edit or delete Doc Templates stored in repository. Once you have made the desired changes in Visual Paradigm locally, you can synchronize the changes to server. Teammates can get the updated templates by synchronizing changes to server as well.

As said earlier, Doc Templates are stored in repository based. Therefore, no matter which project you have opened, you are managing the same set of Doc Templates.

To manage Doc Templates:

1. In Visual Paradigm, select **Tools > Doc. Composer > Manage Doc Templates...** from the toolbar. In order to access the management function, make sure you are opening a team project managed under either VPository.com or Teamwork Server. Besides, make sure you are a team member and have been granted the right to **Change document template** in server. You may need to contact your server administrator to confirm the permission settings made in server.
2. Now, you can manage Doc Templates in the **Manage Doc Templates** window. Read the next section for details about what you can do in the **Manage Doc Templates** window.

Overview of Manage Doc Template window



Overview of Manage Doc Templates window

No.	Name	Description
1	List of Doc Templates	List of Doc Template available for use as Doc Base in document generation.
2	New	Create a Doc Template.
3	Delete	Delete the Doc Template.
4	Import	Import a Word document (.docx) file for this Doc Template. Note that the second time you import a .docx file into a Doc Template will have the original one be replaced by the importing one.
5	Name	Name of Doc Template.
6	Description	Description of Doc Template.
7	Create and modified date	The date of creation and modification of this Doc Template.
8	Export	Export the current or any earlier revisions of Doc Templates as .docx file.
9	OK	Confirm the changes made and return to Doc. Composer.
10	Cancel	Discard the changes and return to Doc. Composer.

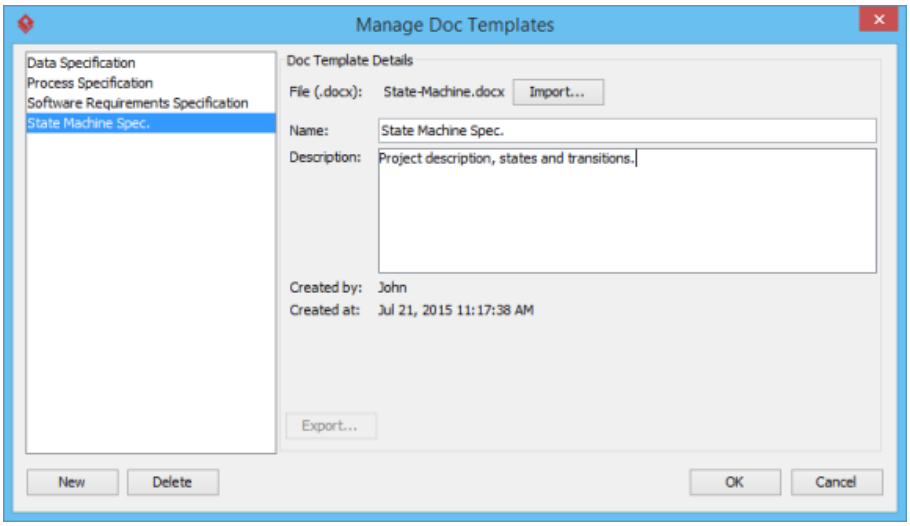
Description of Manage Doc Templates window

Creating a Doc Template

When you create a document under the Fill-in Doc mode of Doc. Composer, you will be asked to choose a Doc Base for document generation. There are two approaches of choosing Doc Base. One is to create from an external document file. Another one is to duplicate from an existing Doc Template. In this section, you will learn how to create a Doc Template. By creating a Doc Template, you can re-use it again and again in creating different documents.

By default, Visual Paradigm provides three default Doc Templates. You can also create your own set of Doc Templates in the **Manage Doc Templates** window by taking the steps below:

1. In the **Manage Doc Templates** window, click **New** at bottom left corner.
2. Choose your .docx file in the file chooser. The file you provide here should contain both manually written content (e.g. Introduction, project scope, etc) and Doc Fields.
3. Enter a meaningful name for the Doc Template.
4. You may enter the description of template as well.



Creating a Doc Template

5. Click **OK** to confirm the changes. Now, you can use the new template when you create a document with Fill-in Doc. You can also share it with teammates.

Deleting a Doc Template

In the **Manage Doc Templates** window, select the Doc Template to delete from the template list and click **Delete...** at bottom left corner to remove it permanently. Note that the deletion will NOT affect any of the document that used the deleted template. The only effect of deletion is that no one will be able to create documents with the Doc Template deleted.

Editing a Doc Template

If you have modified your project documentation, say, for updating its content or layout, you will need to replace the document file previously imported to a Doc Template with the new version of document. You can do this by editing the Doc Template.

In the **Manage Doc Templates** window, select the Doc Template to edit, and then modify its details on the right hand side. You can rename it, update its description or replace it with another document file by clicking **Import...**. Note that by importing a document file, the original one will be overwritten. However, you can always retrieve a previously imported file by clicking **Export...**. Click **OK** when finished editing. When finished, you can use the modified template when you create a document with Fill-in Doc. You can also share it with teammates.

Synchronizing Doc Templates

Once you have finished editing Doc Templates, you can synchronize the changes to server. Teammates can get the updated templates by synchronizing changes to server as well.

To synchronize changes to server manually, select **Tools > Doc. Composer > Sync. to VPository/VP Teamwork Server** from the toolbar.

Note that your changes will be synchronized automatically when you perform commit.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Doc. Composer - Writing Element Templates

What is Doc. Composer Template Language

DCTL (Doc. Composer Template Language) is an XML-based language that enables the transformation of design specification into document content. You will learn DCTL in this article.

Template Root

Learn how to start off an element template by using an appropriate initiation block.

Text and Property

When you want to output some text, you either use a or a . You will learn how it works in this article.

Looping (Non Connector)

When you want to retrieve the children elements from a querying model element / diagram, write a loop element. You will learn how it works in this article.

Looping (Connector)

When you want to retrieve connectors from a querying model element / diagram, write a loop element. You will learn how it works in this article.

Sorting in Loop

Add <Sortings> under a loop element (e.g. <IterationBlock>, <ForEach>) to sort retrieved elements. In this article you will learn how it works.

Conditional Expression

Write conditional expression to control the content to be presented. You will learn how to write conditional expression in this article.

Working with Table

Structure data with table. You will learn how to create table block, table row and table cell in this article.

Working with Image

Add diagram image or icon image into document. You will learn how it works in this article.

Working with Break

Add paragraph break or page break into a document. You will learn how to create breaks in this article.

DCTL Examples

Learn how to write Doc. Composer Template Language by reading a series of examples.

Supported Diagram Types

This page contains all the diagram types available in Visual Paradigm. Both the display and real diagram type are listed.

What is Doc. Composer Template Language?

DCTL (Doc. Composer Template Language) is an XML-based language that enables the transformation of design specification into document content. DCTL comes with a well-defined structure and syntactic rules for writers to define what and how project data should be extracted from a Visual Paradigm project, and how these data should be presented in a document.

The following shows a basic template:

```
<DiagramBaseInitiationBlock>
   <TableBlock tableStyle="Summaries">
      <TableRow>
         <TableCell>
            <Text>Name</Text>
         </TableCell
      </TableRow>

      <IterationBlock modelType="UseCase">
         <TableRow>
            <TableCell>
               <Property property="name"/>
            </TableCell>

         </TableRow>
      </IterationBlock>
   </TableBlock>
</DiagramBaseInitiationBlock>
```

Doc. Composer provides you with a dynamic and efficient document editing experience by letting you produce document through simple drag-and-drop. You just need to select a piece of model data, like a use case or a sequence diagram, then drag out a build-in template and drop it onto the Doc. Composer to create content.

All the build-in templates are opened for editing. To make the document more adoptable to your company's needs, you can edit a template or to design your own templates and re-use it document-by-document. For example, design a class specification by printing out the description of classes that contains "Controller" in their names.

This section is divided into two main parts. The first part is going to talk about the steps for creating a template in Doc. Composer. The second part comes with a detailed specification of template constructs you can use in constructing a template.

As the template shows, it tries to create a table, and add table rows for showing the names of use cases within a diagram. This example is a fairly simple one yet it outlines two main thing that every element template tries to achieve:

- Data retrieval - To query the use cases from a diagram, and then get the name of each use case.
- Layout of content - Table construction.

That's what you can do with a DCTL - You compose an element template with DCTL, drag the template into your document in Doc. Composer, let Doc. Composer interpret your template and output content accordingly.

In the coming sections you will see how to retrieve project data as well as to layout the content with the use of DCTL.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Beginning of template

Every element template mush has <ElementBaseInitiationBlock>, <DiagramBaseInitiationBlock> or <ProjectBaseInitiationBlock> as root element.

The use of <ElementBaseInitiationBlock> is to tell the template engine that the template will be applied to a model element. If you are writing a template for a model element (e.g. use case, package...), use <ElementBaseInitiationBlock> as template root.

The use of <DiagramBaseInitiationBlock> is to tell the template engine that the template will be applied to a diagram. If you are writing a template for a diagram (e.g. Class Diagram...), use <DiagramBaseInitiationBlock> as template root.

The use of <ProjectBaseInitiationBlock> is to tell the template engine that the template will be applied to a project. If you are writing a template for a project, use <ProjectBaseInitiationBlock> as template root.
If you check back the example used in the previous section, you will find that the template was written to query the use cases from a diagram. The template will be applied on a diagram so <DiagramBaseInitiationBlock> was used as root element.

The following examples show the use of <ElementBaseInitiationBlock> and <ProjectBaseInitiationBlock> in templates.

```
<ElementBaseInitiationBlock>
   <!&mdash;Output the name of selected class-->
   <Property property="name"/>
</ElementBaseInitiationBlock>
```

```
<ProjectBaseInitiationBlock>
   <!&mdash;Output the name of all use cases in the project-->
   <IterationBlock allLevel="true" modelType="UseCase">
      <Property property="name"/>
   </IterationBlock>
</ProjectBaseInitiationBlock>
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Text and Property

When you want to output some text, you either use a <Text> or a <Property>.

You use <Text> when you want to output specific words, sentences or paragraphs, such as "Here is a list of business activities:". The following example show the use of <Text> in a template.

```
<Text>Here is a list of elements in my project:</Text>
<ParagraphBreak/>
<IterationBlock allLevel="true" >
   <Property property="name"/>
   <Text> : </Text>
   <Property property="modelType"/>
   <ParagraphBreak/>
</IterationBlock>
```

The first <Text> in the example above outputs a sentence "Here is a list of elements in my project: ".

The second <Text> outputs a colon between the name and type of elements. Note that the space before and after the colon will get output into the document.

Here is the outcome of the example above.

```
Here is a list of elements in my project:
Place Order : UseCase
PaymentController : Class
Cancel Order : BPTask
```

The following table lists the available attributes of <Text>.

Name	Description	Required?
isBold : boolean	Set the text to bold.	Optiona
isItalic : boolean	Set the text italic.	Optiona
isUnderline : boolean	Underline text.	Optiona
fontFamily : string	Specify the name of font to apply to the text.	Optiona
fontSize : integer	Set the font size.	Optiona
foreColor : color	Set the color of text.	Optiona
alignment : string {left | center | right}	Set the alignment of text.	Optiona
style : string	Set the name of style. You can add and edit style in Doc. Composer.	Optiona
numberingLevel : short	Determine the Numbering Level if the text is showing as a number or bullet list.	Optiona
indentation : double	Determine the left margin of text.	Optiona
hyperlink : boolean	Specify whether the text is a hyperlink or not. If true, the text will be linkable.	Optiona
keepWithNextInWord : boolean	Make sure the text will be shown in same page with next item. (Used for WORD document only)	Optiona
keepWithNextInPreview : boolean		Optiona

Attributes of <Text>

You may have noticed the use of <Property> in the example above. <Property> is another way to output text. You use <Property> when you want to output text by extracting the data from a property of querying diagram or element. The following example show the use of <Property> in a template.

```
<IterationBlock allLevel="true" >
   <Text>Name: </Text>
   <Property property="name"/>
   <ParagraphBreak/>
   <Property property="description"/>
   <ParagraphBreak/>
   <ParagraphBreak/>
</IterationBlock>
```

The first <Property> outputs the name of the querying element, while the second <Property> outputs the description.

Here is the outcome of the example above.

```
Name: Place Order
The process to check out a shopping cart and finish the payment.
Name: PaymentController
A controller class that handles the payment logic.
Name: Cancel Order
```

The process to delete an order made within the last 7 days.

The following table lists the available attributes of <Property>.

Name	Description	Required?
property : string	The property to query.	Optiona
isIgnoreHTMLFontSize : boolean	Ignore the font size on the HTML text.	Optiona
isIgnoreHTMLFontFamiliy : boolean	Ignore the font selection of the HTML text	Optiona
forcePlainText : boolean	Force HTML text to show as plain text by removing formatting, if any.	Optiona
default : string	The text to show when the value of property has not ever been specified.	Optiona
isBold : boolean	Set the text to bold.	Optiona
isItalic : boolean	Set the text italic.	Optiona
isUnderline : boolean	Underline text.	Optiona
fontFamily : string	Specify the name of font to apply to the text.	Optiona
fontSize : integer	Set the font size.	Optiona
foreColor : color	Set the color of text.	Optiona
alignment : string {left | center | right}	Set the alignment of text.	Optiona
style : string	Set the name of style. You can add and edit style in Doc. Composer.	Optiona
numberingLevel : short	Determine the Numbering Level if the text is showing as a number or bullet list.	Optiona
indentation : double	Determine the left margin of text.	Optiona
hyperlink : boolean	Specify whether the text is a hyperlink or not. If true, the text will be linkable.	Optiona
keepWithNextInWord : boolean	Make sure the text will be shown in same page with next item. (Used for WORD document only)	Optiona
keepWithNextInPreview : boolean		Optiona

Attributes of <Property>

Understanding Dynamic Heading Style

When you want to set a text produced by a <Text> or a <Property> to be a heading, add and specify the style attribute in the <Text> or <Property>.

There are two ways of specifying a heading – Static and Dynamic. The following example shows the static way of specifying heading:

```
<Text style="Heading 1">Text in heading 1</Text>
<Property property="name" style="Heading 2"/>
```

The <Text> in the example above outputs a sentence "Text in heading 1", with Heading 1 as style. The <Property> outputs the name of a model element, with Heading 2 as style.

The static way of specifying heading requires you to provide the style name in the template.

In contrast to the static way, here is an example that shows the dynamic way of specifying heading:

```
<Text style="@heading+">Text in heading 1</Text>
<Property property="name" style="@heading+"/>
```

In the example above, we do not provide the name of the heading style. Instead, we use **@heading** to indicate the need to assign heading style, **@heading+** to indicate an increase of heading style level. By using **@heading**, the style Heading 1...N will be used in the output document.

Here is the outcome of the example above.

Text in heading 1
Place Order Use Case

The sentence *Text in heading 1* has Heading 1 applied, while the name *Place Order Use Case* has Heading 2 applied.

More about Heading Increment

By appending **+** to **@heading**, the leveling of heading style will be increased. For example, if the previous heading is a Heading 1, the use of **@heading+** will output a heading in Heading 2.

The use of **+** in **@heading** is optional though. If you want to add a heading that has the same heading level as the previous heading, skip + to reuse the style used by the previous heading.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Looping (Non Connector)

When you want to retrieve the children elements from a querying model element / diagram, write a loop element.

<IterationBlock>

Retrieve elements from project / model element / diagram. By iterating over project and model element, a list of model element will be returned. By iterating over diagram, a list of diagram element will be returned. The following example show the use of <IterationBlock> in a template.

```
<IterationBlock modelType="class">
   <Property property="name"/>
   <ParagraphBreak/>
</IterationBlock>
```

Here is the outcome of the example above.

```
User
Account
AccountManager
Transaction
AccountController
```

The following table lists the available attributes of <IterationBlock>.

Name	Description	Required?
modelType : string	Filter the children by specified model element type (e.g. package).	Optional
modelTypes : string	Filter the children by a number of model element types. (e.g. actor, usecase)	Optional
stereotypes : string	Filter the children by a number of stereotypes.	Optional
name : string	Filter the children by their name.	Optional
filterhidden : boolean	Filter hidden children diagram element. This is for retrieving from diagram/ diagram element only.	Optional
includeConnectors : boolean	Determines whether to retrieve shape or shape+connectors from diagram. This is for retrieving from diagram only.	Optional
allLevel : boolean	Determines whether to retrieve all model elements from project. When false, only the root level elements will be retrieved. This attribute is only useful when retrieving elements from project.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop. Break can be <ParagraphBreak> or <StaticText @content= "\n" or ",">.	Optional
breakString : string	Insert a string between model elements of current for-each.	Optional
identifier : string		Optional

Attributes of <IterationBlock>

<ForEach>

Retrieve model elements from a model element's property. The following example show the use of <ForEach> in a template.

```
<ForEach property = "stereotypes">
   <Property property="name"/>
   <ParagraphBreak/>
</ForEach>
```

Here is the outcome of the example above.

```
Control
ORM Persistable
```

The following table lists the available attributes of <ForEach>.

Name	Description	Required?
property : string	The property from which model elements can be retrieved.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop.	Optional
breakString : string	Insert a string between model elements of current for-each.	Optional

Attributes of <ForEach>

<ForEachSubDiagram>

Retrieve sub-diagram(s) from a model element. For example, retrieve sub-sequence-diagrams from a controller class. Note that you can only use <ForEachSubDiagram> to retrieve sub-diagram(s) of model element. If you want to retrieve diagrams from project, use <ForEachDiagram> instead.

The following table lists the available attributes of <ForEachSubDiagram>.

Name	Description	Required?
diagramType : string	The type of diagram to retrieve.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop.	Optional

Attributes of <ForEachSubDiagram>

<ForEachDiagram>

Retrieve diagram(s) from project. Like other for-each elements, you can specify the type of diagram to retrieve. For example, retrieve all class diagrams from project. Note that you can only use <ForEachDiagram> to retrieve diagram from project. If you want to retrieve sub-diagrams from model element, use <ForEachSubDiagram> instead.

The following table lists the available attributes of <ForEachDiagram>.

Name	Description	Required?
diagramType : string	The type of diagram to retrieve.	Optional
property : string	The property from which diagrams can be retrieved.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop.	Optional

Attributes of <ForEachDiagram>

<ForEachOwnerDiagram>

Retrieve the diagram(s) that owns a specific model element. For example, class diagram "Domain Diagram" and "Security" both contain class "Login" (same model element), by applying <ForEachOwnerDiagram> on the "Login" class, diagram "Domain Diagram" and "Security" will be returned.

The following table lists the available attributes of <ForEachOwnerDiagram>.

Name	Description	Required?
diagramType : string	The type of diagram to retrieve.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop.	Optional

Attributes of <ForEachOwnerDiagram>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Looping (Connector)

<ForEachSimpleRelationship>

Retrieve SimpleRelationship elements from model element or connector from diagram element.

The following table lists the available attributes of <ForEachSimpleRelationship>.

Name	Description/Usage	Required?
modelType : string	Filter relationship by specified model element type (e.g. Generalization). Note that not all kind of relationship belongs to simple relationship. Here are the possible types of simple relationship: Abstraction, ActivityObjectFlow, AnalysisComposition, AnalysisDiagramTransitor, AnalysisParentChild, AnalysisReference, AnalysisRelationship, AnalysisSubDiagram, AnalysisTransitor, AnalysisUsed, AnalysisView, Anchor, ArchiMateAccess, ArchiMateAggregation, ArchiMateAssignment, ArchiMateAssociation, ArchiMateCommunicationPath, ArchiMateFlow, ArchiMateNetwork, ArchiMateProvide, ArchiMateRealization, ArchiMateRequire, ArchiMateSpecialization, ArchiMateTriggering, ArchiMateUsedBy, AssociationClass, BPAssociation, BPDataAssociation, BPMessageFlow, BPSequenceFlow, BindingDependency, BusinessRuleAssociation, Constraint, ControlFlow, ConversationLink, DBForeignKey, DFDataFlow, Dependency, Deployment, EPCControlFlow, EPCInformationFlow, EPCOrganizationUnitAssignment, ExceptionHandler, Extend, Generalization, GenericConnector, GlossaryFactTypeAssociation, Include, InteractionDiagramDurationConstraint, Link, MindConnector, MindLink, OCLine, ObjectFlow, PMProcessLink, Permission, RQRefine, RQTrace, Realization, RequirementDerive, Satisfy, Transition, Transition2, Usage, Verify	Optional
modelTypes : string	Filter relationships by modelTypes. If @modelType is defined, @modelTypes will be ignored.	Optional
direction {all | from | to}	Filter relationship by direction.	Optional
ignoreLastSeparator : boolean = false	Ignore the break for the last element of current for-each loop. Break can be <ParagraphBreak> or <StaticText @content= "\n" or ",">.	Optional
breakString : string	Insert a string between model elements of current for-each.	Optional

Attributes of <ForEachSimpleRelationship>

<ForEachRelationshipEnd>

Retrieve the from or to end of an association.

The following table lists the available attributes of <ForEachRelationshipEnd>.

Name	Description/Usage	Required?
modelType : string	Filter relationship end by specific model element type.	Optional
modelTypes : string	Filter relationships by modelTypes. If @modelType is defined, @modelTypes will be ignored.	Optional
endPointer {all | from | to | both | self | other}	Filter relationship ends based on the way they are attached to the querying element. For example, if 'from' is specified, only relationships that take the querying element as the source (i.e. from end) will be chosen.	Optional
ignoreLastSeparator : boolean	Ignore the break for the last element of current for-each loop.	Optional
breakString : string	Insert a string between model elements of current for-each.	Optional

Attributes of <ForEachRelationshipEnd>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition

- Professional Edition
- Enterprise Edition

Sorting in Loop

Add <Sortings> under a loop element (e.g. <IterationBlock>, <ForEach>) to sort retrieved elements. <Sortings> contains one or more <Sorting>. Each <Sorting> defines a way to sort the elements retrieved. The following example show the use of <Sorting> in a template.

```
<IterationBlock modelType="class">
   <Sortings>
      <Sorting by="name"/>
   </Sortings>
   <Property property="name"/>
   <ParagraphBreak/>
</IterationBlock>
```

Here is the outcome of the example above.

```
Account
AccountController
AccountManager
Transaction
User
```

The following table lists the available attributes of <Sorting>.

Name	Description/Usage	Required?
by : string {name | type | modelType | diagramType | property | followTree | level | businessProcessFlow}	Sort by any of the following options: - name : sort by name - type : sort by type (type of model element or diagram) - modelType : sort by model element's type - diagramType : sort by diagram type - property: sort by property, requires the definition of @property, @sortValues, @defaultPropertyValue - followTree: sort following how elements are being sorted in Model Navigator and Diagram Navigator - level: sort by parent-child - businessProcessFlow: sorting the BPD elements by the ordering in BPD (calculated by their ordering in sequence/message flow). ONLY AVIALABLE for sorting Diagram Elements in a BPD	Required
property : string	If @by="property", you have to specify @property to specify the property to be sorted. You can also sort elements by their tagged values by specifying this: ${taggedValues.children(TAG_NAME).value} Replace TAG_NAME with the name of the tag to be sorted	Optional
sortValues : string	If @by="property", you can specify @sortValues to define the ordering of values to be sorted. e.g. @by="property" @property="visibility" @sortValues="public, protected, private" means 'public' model elements will list before 'protected' model elements, 'protected' will list before 'private'	Optional
defaultPropertyValue : string	If @by="property", @defaultPropertyValue can be specified for the default value of the model elements that don't have this property value.	Optional
descending : boolean	Sorting in descending order?	Optional

Attributes of <Sorting>

Suppress the default way of sorting

Without using <Sortings> and <Sorting>, elements in loop will still be sorted alphabetically. If you want to suppress the default way of sorting, write <Sortings noSort="true"/>. Here is an example:

```
<IterationBlock modelType="class">
   <Sortings noSort="true"/>
   <Property property="name"/>
   <ParagraphBreak/>
</IterationBlock>
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Conditional Expression

<DefaultValueChecker>

The <DefaultValueChecker> element evaluates the querying element to check if the property stated by the @property attribute equals to its default value. If the result of evaluation matches with the result stated by the attribute @flag, the child elements of <DefaultValueChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <DefaultValueChecker> in a template.

```
<IterationBlock modelType="Class">
   <DefaultValueChecker property="root" flag="false">
      <Text>The root property has been modified.</Text>
   </DefaultValueChecker>
</IterationBlock>
```

The following table lists the available attributes of <DefaultValueChecker>.

Name	Description/Usage	Required?
property : string	The property to check.	Required
flag : boolean = false	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional

Attributes of <DefaultConditionChecker>

<ValueChecker>

The <ValueChecker> element evaluates the querying element to check if the value of the property stated by the @property attribute equals to the value stated by the @value attribute. If the result of evaluation is true, the child elements of <ValueChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <ValueChecker> in a template.

```
<IterationBlock modelType="Class">
   <ValueChecker property="name" value="ShapeCreator">
      <Text>ShapeCreator class found!</Text>
   </ValueChecker>
   <ValueChecker property="description" operator="not equals" value="">
      <Text>Description: </Text>
      <Property property="description"/>
   </ValueChecker>
</IterationBlock>
```

The following table lists the available attributes of <ValueChecker>.

Name	Description/Usage	Required?
property : string	The property to check.	Optional
operator : string {equals | not equals | less than | equals or less than | greater than | equals or greater than | like | not like | equal | not equal}	Specify the way to compare the property value of model against your expectation. equals - The value of property must be the same as the expected value not equals - The value of property must be different from the expected value less than - The value of property must be smaller than the expected value. equals or less than - The value of property must be the same or smaller than the expected value. greater than - The value of property must be larger than the expected value. equals or greater than - The value of property must be the same or larger than the expected value. like – The value of property must contain the expected value. not like - The value of property must not contain the expected value.	Required
value : string	The value expected for the property. If @regularExpression is set to true, you can make use of '?' and '*' in the value field for representing wildcard characters. For example, use "*UI" as @value to find out all model elements with names end with "UI".	Optional
length : int		Optional
caseSensitive : boolean	Determine whether the checking of string property need to take care of the use of upper and lower case.	Optional
regularExpression : boolean	When true, you can make use of '?' and '*' in the value field for representing wildcard characters. For example, use "*UI" as @value to find out all model elements with names end with "UI".	Optional
id : string		Optional

<HasChildElementChecker>

The <HasChildElementChecker> element evaluates the querying element to check if it contains any child element, or type(s) of child elements specified by the @modelType or @modelTypes attributes. If the result of evaluation is true, the child elements of <HasChildElementChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasChildElementChecker> in a template.

```
<IterationBlock modelType="Class">
   <HasChildElementChecker modelType="Attribute" flag="true">
      <IterationBlock modelType="Attribute">
         <Property property="name"/>
         <ParagraphBreak>
      </IterationBlock>
   </HasChildElementChecker>
   <HasChildElementChecker modelType="Attribute,Operation" flag="true">
      <IterationBlock modelTypes="Attribute,Operation">
         <Property property="name"/>
         <ParagraphBreak>
      </IterationBlock>
   </HasChildElementChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasChildElementChecker>.

Name	Description/Usage	Required?
flag : boolean = true	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
modelType : string	The type of model element you want the parent to contain or not contain.	Optional
modelTypes : string	The types of model element you want the parent to contain or not contain.	Optional
stereotypes : strings	Filter the children by a number of stereotypes.	Optional
includeConnectors : boolean	Determine whether to retrieve shape or shape+connectors from diagram. This is for retrieving from diagram only.	Optional
filterHidden : boolean		
allLevel : boolean	Determine whether to retrieve all model elements from project. When false, only the root level elements will be retrieved. This attribute is only useful when retrieving elements from project.	
valueConditionCheckId : string		

Attributes of <HasChildElementChecker>

<HasRelationshipChecker>

The <HasRelationshipChecker> element evaluates the querying element to check if it contains any relationship, or type(s) of relationships specified by the @modelType or @modelTypes attributes. If the result of evaluation is true, the child elements of <HasRelationshipChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasRelationshipChecker> in a template.

```
<IterationBlock modelType="Class">
   <HasRelationshipChecker modelType="Association" flag="true">
      <ForEachRelationshipEnd modelType="AssociationEnd" endPointer="self">
         <RelationshipEndEndRelationship>
            <FromEnd>
               <ModelElementProperty property="EndModelElement">
                  <Property property="name"/>
                  <Text>, </Text>
               </ModelElementProperty>
            </FromEnd>
            <ToEnd>
               <ModelElementProperty property="EndModelElement">
                  <Property property="name"/>
                  <Text>, </Text>
               </ModelElementProperty>
            </ToEnd>
      </ForEachRelationshipEnd>
   </HasRelationshipChecker>
   <HasRelationshipChecker modelType="Generalization" direction="to">
      <ForEachSimpleRelationship type="Generalization">
         <ModelElementProperty property="from">
```

```
            <Property property="name"/>
            <Text>, </Text>
        </ModelElementProperty>
    </ForEachSimpleRelationship>
  </HasRelationshipChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasRelationshipChecker>.

Name	Description/Usage	Required?
flag : boolean = true	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
modelType : string	The type of relationship you want the querying model element to contain.	Optional
modelTypes : string	The types of relationship you want the querying model element to contain. If @modelType is specified, @modelTypes will be ignored.	Optional
direction {all | from | to}	Check if the querying model element belongs to a specific end of a relationship. Possible values: all, from, to, self_begins, self_ends	Optional

Attributes of <HasRelationshipChecker>

<HasDiagramChecker>

The <HasDiagramChecker> element evaluates the querying "thing" (project or model element) to check if it has specified any diagram for a given property specified by the @property attribute. If the result of evaluation is true, the child elements of <HasDiagramChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasDiagramChecker> in a template.

```
<ProjectBaseInitiationBlock>
  <HasDiagramChecker diagramType="ClassDiagram" flag="true">
    <Text>This project contains at least one class diagram.</Text>
  </HasDiagramChecker>
</ProjectBaseInitiationBlock>
```

The following table lists the available attributes of <HasDiagramChecker>.

Name	Description/Usage	Required?
flag : boolean	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
property : string	The property from which diagrams can be retrieved.	Optional
diagram : string	The type of diagram you want the project to contain or not contain.	Optional

Attributes of <HasDiagramChecker>

<HasValueChecker>

The <HasValueChecker> element evaluates the querying "thing" (project or model element) to check if it has specified a given property, specified by the @property attribute. If the result of evaluation is true, the child elements of <HasDiagramChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasValueChecker> in a template.

```
<IterationBlock modelType="Class">
  <HasValueChecker property="taggedValues" flag="true">
    <Text>This class contains at least one tagged value.</Text>
  </HasValueChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasValueChecker>.

Name	Description/Usage	Required?
flag : boolean	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
property : string	The name of property to check.	Optional
modelType : string	The result of evaluation will return a true only if the querying model element contains the type of elements specified by @modelType.	Optional
name : string	The result of evaluation will return a true only if the querying model element contains the elements with same name as specified by @name.	Optional

stereotypes : string	The result of evaluation will return a true only if the querying model element contains the elements that are extended from the stereotypes specified by @stereotypes.	Optional

Attributes of <HasValueChecker>

<HasParentModelChecker>

The <HasParentModelChecker> element evaluates the querying model element to check if it is being contained by a parent model element. If the result of evaluation is true, the child elements of <HasParentModelChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasParentModelChecker> in a template.

```
<IterationBlock modelType="Class">
   <HasParentModelChecker modelType="Package" flag="true">
      <Text>This class contains is contained by package: </Text>
      <ParentModel>
         <Property property="name"/>
      </ParentModel>
   </HasParentModelChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasParentModelChecker>.

Name	Description/Usage	Required?
flag : boolean	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
modelType : string	The type of parent model element you want the model element to be/not to be contained by.	Optional

Attributes of <HasParentModelChecker>

<HasSubDiagramChecker>

The <HasSubDiagramChecker> element evaluates the querying model element to check if it contains any sub-diagram. If the result of evaluation is true, the child elements of <HasSubDiagramChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasSubDiagramChecker> in a template.

```
<IterationBlock modelType="Class">
   <HasSubDiagramChecker diagramType="StateMachineDiagram" flag="true">
      <ForEachSubDiagram>
         <Property property="name"/>
      </ForEachSubDiagram>
   </HasSubDiagramChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasSubDiagramChecker>.

Name	Description/Usage	Required?
flag : boolean	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional
diagramType: string	The type of sub-diagram you want the model element to contain.	Optional

Attributes of <HasSubDiagramChecker>

<HasOwnerDiagramsChecker>

The <HasOwnerDiagramsChecker> element evaluates the querying model element to check if it has been visualized in any diagram. If the result of evaluation is true, the child elements of <HasOwnerDiagramsChecker> will be processed. Otherwise, the child elements will be skipped.

The following example shows the use of <HasOwnerDiagramsChecker> in a template.

```
<IterationBlock modelType="BPTask">
   <HasOwnerDiagramsChecker diagramType="BusinessProcessDiagram" flag="true">
      <ForEachOwnerDiagram>
         <Property property="name"/>
      </ForEachOwnerDiagram>
   </HasOwnerDiagramsChecker>
</IterationBlock>
```

The following table lists the available attributes of <HasOwnerDiagramsChecker>.

Name	Description/Usage	Required?
flag : boolean	The expected result of checking. If the actual result matches the value specified by @flag, the child elements will be processed.	Optional

diagramType: string	The type of diagram you want the owner diagram to be.	Optional

Attributes of <HasOwnerDiagramsChecker>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Working with Table

<TableBlock>

Insert a table to document. It is typically used to present table of elements or element properties. You must combine the use of <TableRow> and <TableCell> in order to form a complete table. Most of the build-in templates are formed by tables and contains <TableBlock>. You can look for references easily.

Name	Description/Usage	Required?
tableStyle : string	Specify the table style by its ID. You can find the ID of table styles in the **Formats** window of Doc. Composer.	
tableWidth : integer	The width of table, in percentage. For example: 50% means to occupy 50% of page width.	Optional
colWidths : integers	Specify the widths of table columns in ratio, separate by comma. Note that the number of columns specify in @colWidths must match the number of <TableCell> to add under <TableRow>, under this table. For example, specify "1, 1, 2" for a table with 20000 as width will result in creating a table with three columns, and have widths 5000, 5000, 10000.	Optional
rowBackgroundColors colors	Background color of rows in table.	Optional
repeatTableHeader {true | false | followOption}	True to repeat the table header row at the top of the next page for table that span multiple pages. Note that this option only works in PDF and Word document. If "followOption", it will follow the setting set in Doc. Composer's document export window.	Optional

Attributes of <TableBlock>

<TableRow>

Enables you to add cells to a <TableRow>.

Name	Description/Usage	Required?
height : integer	How tall it is for the table row.	Optional
backgroundColor : color	Background color of row.	Optional

Attributes of <TableRow>

<TableCell>

Enables you to add cells to a <TableRow>.

Name	Description/Usage	Required?
topBorderEnable : boolean = true	True to draw the top border of cell.	Optional
bottomBorderEnable : boolean = true	True to draw the bottom border of cell.	Optional
leftBorderEnable : boolean = true	True to draw the left border of cell.	Optional
rightBorderEnable : boolean = true	True to draw the right border of cell.	Optional
verticalAlignment {top | center | bottom}	The vertical alignment of cell.	Optional
color : color	The background color of cell.	Optional
splitted : boolean = false	Determine this cell is splitted. If true, <TableRow> + <TableCell> cannot be added into this cell to make the cell splitted by more cells.	Optional
colspan : integer	Specify the number of cell this cell consumes horizontally. For example, a colspan of 2 means to consume this and the cell on the right. This is equivalent to HTML's colspan.	Optional

Attributes of <TableCell>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Working with Image

<Image>

Name	Description/Usage	Required?
alignment = left {left | center | right}	Set the alignment of image.	Optional
width : string	Set the width of image. It can be an absolute value (e.g. "15500") or a scale to the original image width (e.g. "80%")	Optional
height : string	Set the height of image. It can be an absolute value (e.g. "15500") or a scale to the original image height (e.g. "80%")	Optional
maxWidth : integer	Set the maximum width of image.	Optional
maxHeight : integer	Set the maximum height of image.	Optional
rotate : string {none | right | left}	Rotate the image to right (90 degree) or left (270 degree)	Optional
keepWithPreviousInPDF : boolean	Make sure the previous item will be shown in same page with this item. (Used for PDF document only)	Optional
keepWithNextInWord : boolean	Make sure the this item will be shown in same page with next item. (Used for WORD document only)	Optional
keepWithNextInPreview : boolean	Make sure this item will be shown in same page with next item. (Used for previewing in Doc. Composer only)	Optional

Attributes of <Image>

<Icon>

Icon of a model element type.

Name	Description/Usage	Required?
alignment = left {left | center | right}	Set the alignment of icon image.	Optional
rotate : string {none | right | left}	Rotate the image to right (90 degree) or left (270 degree)	Optional

Attributes of <Image>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Working with Break

<ParagraphBreak>
Enables you to add a break in document to separate text into paragraphs. <ParagraphBreak> does not carry any text.

<PageBreak>
Enables you to insert a new page at where <PageBreak> is processed.

Related Resources
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions
- Standard Edition
- Professional Edition
- Enterprise Edition

Working with other DCTL Constructs

<OwnerDiagram>

Retrieve the diagram in which a diagram element or the master view of a model element reside. For example, class diagram "Domain Diagram" and "Security" both contain class "Login" (same model element), while the master view is placed inside "Login", by applying <OwnerDiagram> on the "Login" class, diagram "Domain Diagram" will be returned.

If you want to retrieve all the diagrams that own a model element, use <ForEachOwnerDiagram> instead.

<ParentModel>

<ParentModel> serves two purposes. First, to retrieve the immediate parent element of a model element or diagram. Second, to look for a specific *type* of parent element along the hierarchy.

Here is an example of how <ParentModel> can help you find the immediate parent element. For example, class "Circle" is in package "Shape", by applying <ParentModel> on the "Circle" class, the package "Shape" will be returned.

Name	Description/Usage	Required?
modelType : string	Used in finding a specific type of parent along the hierarchy. For example, if class "Shape" is in package "Shape" and "Shape" is in model "Main", by applying <ParentModel modelType="Model"> on "Circle", the model "Main" will be returned.	Optional

Attributes of <ParentModel>

<ParentShape>

<ParentShape> serves two purposes. First, to retrieve the immediate parent shape of a shape. Second, to look for a specific *type* of parent shape along the hierarchy.

Here is an example of how <ParentDiagram> can help you find the immediate parent element. For example, class "Circle" is drawn in package "Shape", by applying <ParentParent> on the "Circle" class, the package "Shape" will be returned.

Name	Description/Usage	Required?
shapeType : string	Used in finding a specific type of parent along the hierarchy. For example, if class "Shape" is drawn in package "Shape" and "Shape" is drawn in model "Main", by applying <ParentShape modelType="Model"> on "Circle", the model "Main" will be returned.	Optional

Attributes of <ParentShape>

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Reusing Template with Inline or Reference

You may want to produce same content under different templates. If you duplicate the same template code in multiple templates, you need to spend extra time and effort in keeping them consistent with each other. In such a case, you can create one template, and reuse it in other templates. The reuse of template can be done by using <Reference> and <Inline>. The following example show the use of <Reference> and <Inline> in a template.

```
<Inline template="Children (General)"/>
<Reference template="Children (General)"/>
```

The example above means that when <Inline> is met, substitute that part with content written in the Children (General) template. Same for <Reference>.

Inline vs Reference

Both <Inline> and <Reference> support the reuse of templates. They work nearly identically except one important difference - The way they handle dynamic heading style.

Dynamic heading style, as its name suggest, supports the dynamic assignment of heading style to text content. If you use <Inline> and in the referencing template a **@heading+** is used, the leveling of active heading style will be increased by one. Once the referencing template has ended and the flow flows back to the source template, the leveling of heading style will remain as-is, which means, same as the leveling used by the last heading defined in the referencing template.

What makes <Reference> different is that when its flow ends and flows back to the source template, the leveling of heading style will be reset to that before entering the referencing template.

Let's explain with an example. Here a template *Bar*.

```
<AnyBaseInitiationBlock>
  <Text style="@heading+">Heading</Text>
  <Text style="@heading+">Heading</Text>
  <Text style="@heading+">Heading</Text>
</AnyBaseInitiationBlock>
```

Here is another template *Foo*. It references the Bar template above.

```
<DiagramBaseInitiationBlock>
  <Text style="@heading">Init</Text>
  <Inline template="Bar"/>
  <Text style="@heading">Result</Text>
</DiagramBaseInitiationBlock>
```

In *Bar*, several **@heading+** has been used, which trigger the increases of the leveling of heading style. You can expect that in the end the text Result printed by *Foo* will be in Heading 4 because it follows last style used by the template referencing inline.

If we change <Inline> to <Reference>, the text *Result* will be in Heading 1, following the style last used within *Foo*.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

DCTL Examples

Working with Use Case Flow of Events

```
<ForEach property="stepContainers">
  <Property property="name"/>
  <TableBlock>
    <FlowOfEventIterationBlock> <!-- Walk through each step (row) in a scenario -->
      <TableRow>
        <TableCell>
          <FlowOfEventIndent/> <!-- Apply proper indentation to the current step. You don't have to specify the level of indentation. It's automatically done for you -->

          <Property property="index" foreColor="#848284" style="Table Contents"/> <!-- The step number. We set its foreColor to a lighter one to make it looks like how it looks in Visual Paradigm -->
          <Text style="Table Contents"> </Text>
          <ValueChecker property="type" operator="not equals" value="">
            <ValueChecker property="type" operator="not equals" value="system">
              <Property property="type" foreColor="#00B200" style="Table Contents"/> <!-- @type here refers to 'control labels' like if, then, elseif -->
            </ValueChecker>
            <ValueChecker property="type" operator="equals" value="system">
              <Property property="type" foreColor="#CA6400" style="Table Contents"/> <!-- @type here refers to SYSTEM -->
            </ValueChecker>
          </ValueChecker>
          <Property property="name" style="Table Contents"/> <!-- The content of step -->
        </TableCell>
      </TableRow>
    </FlowOfEventIterationBlock>
  </TableBlock>
</ForEach>
```

Working with Sub-Diagrams

```
<HasSubDiagramChecker>
  <!-- Name of all sub-diagrams -->
  <ForEachSubDiagram>
    <Property property="name" />
    <ParagraphBreak/>
  </ForEachSubDiagram >
  <!-- Name of all sub-BPD -->
  <ForEachSubDiagram diagramType="BusinessProcessDiagram">
    <Property property="name" />
    <ParagraphBreak/>
  </ForEachSubDiagram >
</HasSubDiagramChecker>
```

Working with References

```
<!-- Output the name of all Use Case reference -->
<HasValueChecker property="references">
  <ForEach property="references">
    <ValueChecker property="type" operator="equals" value="Model Element">
      <ModelElementProperty property="url">
        <ValueChecker property="modelType" operator="equals" value="UseCase">
          <Property property="name" />
        </ValueChecker>
      </ModelElementProperty>
    </ValueChecker>
  </ForEach>
</HasValueChecker>
<!-- Output the name of all model elements that are referencing the querying element as 'Model Element Reference' -->
<HasValueChecker property="modelElementReferencedBys">
  <ForEach property="modelElementReferencedBys">
    <Property property="name"/>
  </ForEach>
</HasValueChecker>
<!-- Output the name of all model elements that are referencing the querying element as 'Shape Reference' -->
<HasValueChecker property="viewReferencedBys">
  <ForEach property="viewReferencedBys">
    <Property property="name"/>
  </ForEach>
</HasValueChecker>
```

Working with Stereotypes and Tagged Values

```
<!-- Output the name of all assigned stereotypes -->
<HasValueChecker property="stereotypes">
  <ForEach property="stereotypes" ignoreLastSeparator="true">
```

```
    <Text>&lt;&lt;</Text>
      <Property property="name" />
    <Text>&gt;&gt;</Text>
  </ForEach>
</HasValueChecker>
<ParagraphBreak/>
<!-- Output the name of tagged values added -->
<HasValueChecker property="taggedValues">
  <ModelElementProperty property="taggedValues">
    <HasChildElementChecker modelType="TaggedValue">
      <IterationBlock modelType="TaggedValue">
        <Property property="name" />
        <Text> - </Text>
        <Property property="value" />
      </IterationBlock>
    </HasChildElementChecker>
  </ModelElementProperty>
</HasValueChecker>
```

Working with Table Records of Entity

```
<!-- Check if an entity (i.e. DBTable) has record specified -->
<HasValueChecker property="records">
  <TableBlock>
    <!-- Create a header row that shows the columns' name in each cell -->
    <TableRow>
      <IterationBlock modelType = "DBColumn">
        <TableCell>
          <Property property="name" />
        </TableCell>
      </IterationBlock>
    </TableRow>
    <!-- Create one row for each record -->
    <ModelElementProperty property="records">
      <IterationBlock modelType = "EntityRecord">
        <TableRow>
          <IterationBlock modelType = "EntityRecordCell">
            <TableCell>
              <!-- Output the value directly if it's a general column -->
              <Property property="value" />
              <!-- Use the following method to output the value if it's a FK -->
              <HasValueChecker property="value" flag="true">
                <ModelElementProperty property="value">
                  <Property property="value" />
                </ModelElementProperty>
              </HasValueChecker>
            </TableCell>
          </IterationBlock>
        </TableRow>
      </IterationBlock>
    </ModelElementProperty>
  </TableBlock>
</HasValueChecker>
```

Working with Working Procedures of BPMN Task/Sub-Process

```
<!-- Check if a BPMN task/sub-process has working procedure entered -->
<HasValueChecker property="bpProcedures">
  <!-- Retrieve the procedure steps -->
  <ForEach property="bpProcedures">
    <Property property="name" style="@heading"/>
    <ParagraphBreak/>

    <!-- Retrieve the steps in the querying procedure set -->
    <ForEach property="bpProcedureSteps">
      <Property property="name"/>
      <ParagraphBreak/>
    </ForEach>
  </ForEach>
</HasValueChecker>
```

Working with Action and Action's Type

```
<HasChildElementChecker flag="true">
  <TableBlock colWidths="1,2" tableWidth="14500" alignment="right">
    <TableRow>
      <TableCell leftBorderEnable="false" rightBorderEnable="false" color="230, 230, 230">
        <Text style="Column header 1">Element</Text>
```

```
      </TableCell>
      <TableCell leftBorderEnable="false" rightBorderEnable="false" color="230, 230, 230">
        <Text style="Column header 1">Description</Text>
      </TableCell>
    </TableRow>

    <IterationBlock modelType="ActivityAction">
      <!-- Sort the activities by name. -->
      <Sortings>
        <Sorting by="name"></Sorting>
      </Sortings>
      <ValueChecker property="modelType" operator="not equals" value="">
        <TableRow>
          <TableCell leftBorderEnable="false" rightBorderEnable="false">
            <Property property="name" style="Table Contents"/>
          </TableCell>
          <TableCell leftBorderEnable="false" rightBorderEnable="false">
            <ValueChecker property = "documentation" operator = "NOT EQUAL" value = "">
              <Property property="documentation" style="Table Contents"/>
              <ParagraphBreak/>
              <ParagraphBreak/>
            </ValueChecker>
            <ModelElementProperty property="actionType">

              <!-- Call Behavior Action -->
              <ValueChecker property="modelType" operator="equals" value="CallBehaviorAction">
                <Text style="Table Contents">Call action:</Text>
                <HasValueChecker property="behavior" flag="true">
                  <ModelElementProperty property="behavior">
                    <Property property="name" style="TableContents" isBold="true"/>
                  </ModelElementProperty>
                </HasValueChecker>
                <HasValueChecker property="behavior" flag="false">
                  <Text style="Table Contents">&lt;Unspecified&gt;</Text>
                </HasValueChecker>
              </ValueChecker>

              <!-- Call Operation Action -->
              <ValueChecker property="modelType" operator="equals" value="CallOperationAction">
                <Text style="Table Contents">Call operation:</Text>
                <HasValueChecker property="operation" flag="true">

                  <ModelElementProperty property="operation">
                    <ParentModel>
                      <Property property="name" style="Table Contents"  isBold="true"/>
                    </ParentModel>
                    <Text style="Table Contents">.</Text>
                    <Property property="name" style="Table Contents"  isBold="true"/>
                  </ModelElementProperty>
                </HasValueChecker>
                <HasValueChecker property="operation" flag="false">
                  <Text style="Table Contents">&lt;Unspecified&gt;</Text>
                </HasValueChecker>
              </ValueChecker>
            </ModelElementProperty>

          </TableCell>
        </TableRow>
      </ValueChecker>
    </IterationBlock>
  </TableBlock>
</HasChildElementChecker>
```

Working with Chart Relations

```
<!-- Show the Chart Relations details -->
<HasValueChecker property="chartRelations">
  <Text style="@heading+">Chart Relations</Text>
  <ParagraphBreak/>
  <TableBlock tableStyle="Summaries" colWidths="20, 80">
    <TableRow>
      <TableCell>
        <Text>Code</Text>
        </TableCell>
        <TableCell>
        <Text>Begins</Text>
      </TableCell>
      <TableCell>
        <Text>Ends</Text>
```

```
      </TableCell>
    </TableRow>
    <ForEach property="chartRelations">
      <TableRow>
        <TableCell>
          <ModelElementProperty property="code">
            <Property property="code"/>
          </ModelElementProperty>
        </TableCell>
        <TableCell>
          <ModelElementProperty property="from">
            <Icon/>
            <Property property="name"/>
          </ModelElementProperty>
        </TableCell>
        <TableCell>
          <ModelElementProperty property="to">
            <Icon/>
            <Property property="name"/>
          </ModelElementProperty>
        </TableCell>
      </TableRow>
    </ForEach>
  </TableBlock>
</HasValueChecker>
```

Working with Model Transitor

```
<!-- Check if the querying element has transition, either from/to -->
<HasValueChecker property="Traceability">
  <Text>Traceability detected.</Text>
  <ParagraphBreak/>
</HasValueChecker>
<ParagraphBreak/>
<!-- List out the Transit From elements -->
<Text isBold="true">Transit From:</Text>
<ParagraphBreak/>
<ForEach property="transitFrom">
  <Property property="name"/>
  <ParagraphBreak/>
</ForEach>
<ParagraphBreak/>
<!-- List out the Transit To elements -->
<Text isBold="true">Transit To:</Text>
<ParagraphBreak/>
<ForEach property="transitTo">
  <Property property="name"/>
  <ParagraphBreak/>
</ForEach>
<ParagraphBreak/>
<!-- List out the Transit From diagrams -->
<Text isBold="true">Transit From (Diagram):</Text>
<ParagraphBreak/>
<ForEachDiagram property="transitFrom">
  <Property property="name"/>
  <ParagraphBreak/>
</ForEachDiagram>
<ParagraphBreak/>
<!-- List out the Transit To diagrams -->
<Text isBold="true">Transit To (Diagram):</Text>
<ParagraphBreak/>
<ForEachDiagram property="transitTo">
  <Property property="name"/>
  <ParagraphBreak/>
</ForEachDiagram>
<ParagraphBreak/>
```

Working with InstanceSpecification in an Object Diagram

```
<!-- Print the description of instanceSpecifications and their classifiers, in a given object diagram-->
<DiagramBaseInitiationBlock>
   <IterationBlock modelType="InstanceSpecification">
      <!-- Name of instance (in ${instance_name} : ${classifier} format)-->
      <Property property="name"/>
      <Text> : </Text>
      <HasValueChecker property="classifiers">
         <ForEach property="classifiers">
            <Property property="name" />
```

```
        </ForEach>
      </HasValueChecker>
      <!-- Description of class-->
      <ParagraphBreak/>
      <Text isBold="true">Description of classifier: </Text>
      <HasValueChecker property="classifiers">
        <ForEach property="classifiers">
          <Property property="description" />
        </ForEach>
      </HasValueChecker>
      <!-- Description of instance-->
      <ParagraphBreak/>
      <Text isBold="true">Description of instance: </Text>
      <Property property="description"/>
      <ParagraphBreak/>
      <ParagraphBreak/>
    </IterationBlock>
</DiagramBaseInitiationBlock>
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Supported Diagram Types

The following table contains all the diagram types available in Visual Paradigm 12.1. When you need to query a specific type of diagram with DCTL, make sure the value presented in the second column is used. Please be aware that the list is version specific, so you may not find all of them in previous versions. The diagram types are ordered following the Diagram Navigator, so that you can easily locate the diagram type you want.

Display Diagram Type	Diagram Type to Use in Fields/Templates
Use Case Diagram	UseCaseDiagram
Class Diagram	ClassDiagram
Sequence Diagram	InteractionDaigram
Communication Diagram	CommunicationDiagram
State Machine Diagram	StateDiagram
Activity Diagram	ActivityDiagram
Component Diagram	ComponentDiagram
Deployment Diagram	DeploymentDiagram
Package Diagram	PackageDiagram
Object Diagram	ObjectDiagram
Composite Structure Diagram	CompositeStructureDiagram
Timing Diagram	TimingDiagram
Interaction Overview Diagram	InteractionOverviewDiagram
Textual Analysis	TextualAnalysis
Requirement Diagram	RequirementDiagram
Basic Diagram	FreehandDiagram
CRC Card Diagram	CRCCardDiagram
Android Tablet Wireframe	WFAndroidTabletDiagram
Android Phone Wireframe	WFAndroidPhoneDiagram
Desktop Wireframe	WFDesktopDiagram
iPad Wireframe	WFIPadDiagram
iPhone Wireframe	WFIPhoneDiagram
Web Wireframe	WFWebDiagram
Entity Relationship Diagram	ERDiagram
ORM Diagram	ORMDiagram
Business Process Diagram	BusinessProcessDiagram
Conversation Diagram	ConversationDiagram
Data Flow Diagram	DataFlowDiagram
EPC Diagram	EPCDiagram
Process Map Diagram	ProcessMapDiagram
Organization Chart	OrganizationChart
Fact Diagarm	FactDiagram
Decision Table	DTBDecisionTableEditorDiagram
Block Definition Diagram	BlockDefinitionDiagram
Internal Block Diagram	InternalBlockDiagram
Parametric Diagram	ParametricDiagram
Zachman Framework	ZachmanDiagram

Business Motivation Model	BusinessMotivationModelDiagram
ArchiMate Diagram	ArchiMateDiagram
Service Interface Diagram	SoaMLServiceInterfaceDiagram
Service Participant Diagram	SoaMLServiceParticipantDiagram
Service Contract Diagram	SoaMLServiceContractDiagram
Services Architecture Diagram	SoaMLServicesArchitectureDiagram
Service Categorization Diagram	ServiceCategorizationDiagram
Matrix Diagram	MatrixDiagram
Analaysis Diagram	AnalysisDiagram
Chart Diagram	ChartDiagram
Overview Diagram	OverviewDiagram
Mind Mapping Diagram	MindMapDiagram
Grid	GridDiagram
Brainstorm	Brainstorm
Profile Diagram	ProfileDiagram

Diagram types and their real type names

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Standard Edition
- Professional Edition
- Enterprise Edition

Data flow diagram

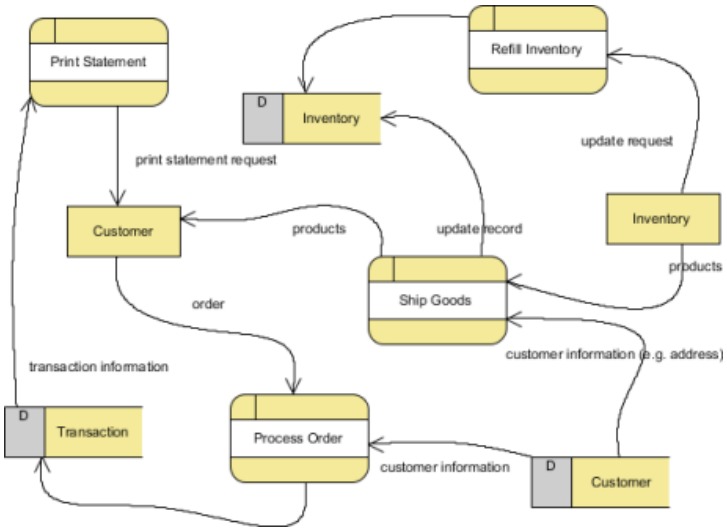
Data flow diagram is a well known approach to visualize the data processing in a business analysis field. This chapter teaches you how to create a data flow diagram.

Creating data flow diagram

There is a list of supported notations in a data flow diagram. You will also see how to decompose a process.

Creating data flow diagram

Data flow diagram is a well known approach to visualize the data processing in business analysis field. A data flow diagram is strong in illustrating the relationship of processes, data stores and external entities in business information system.

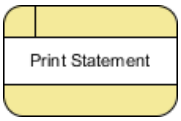
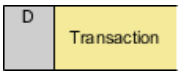
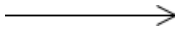


A sample data flow diagram

Creating data flow diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Data Flow Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

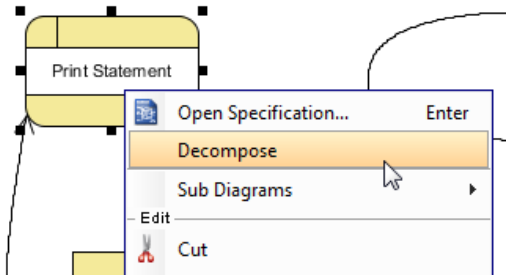
Notations

Name	Representation	Description
Process		A process takes data as input, execute some steps and produce data as output.
External Entity		Objects outside the system being modeled and interacted with processes in a system.
Data Store		Files or storage of data that store(s) data input and output from process.
Data Flow		The flow of data from process to process.
Bidirectional Data Flow		The flow of data that flow both from and to process.

A list of supported notations in data flow diagram

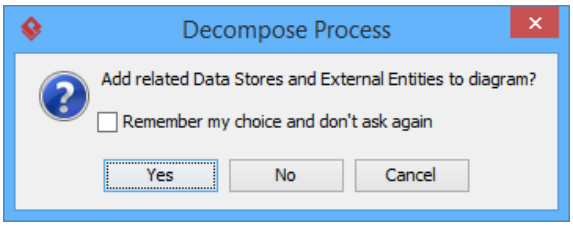
Decomposing a process

You can create multiple data flow diagrams for different levels of detail. A new level can be decomposed from a process in diagram. To decompose a process, right click on the process and select **Decompose** from the popup menu. In the sub-diagram you will see the in/out flows that allow you to connect the data flow from parent to sub-diagram. Click here you if you want to learn more about in/out flows.



Decompose process

The **Decompose Process** dialog box will be prompted to ask you whether to add related data stores and external entities to the new data flow diagram. If you choose **Yes**, those connected data stores and external entities will be copied to the new diagram.



Choose whether to add related model elements to diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Event-driven process chain diagram

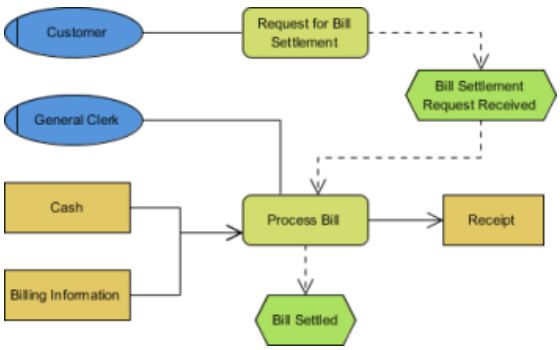
EPC diagram, abbreviation for event-driven process hain diagram is a flowchart based diagram that can be used for resource planning and identifying possible improvements of a business process. This chapter teaches you how to create a EPC diagram.

Creating event-driven process chain (EPC) diagram

This page teaches you how to create a EPC diagram through the diagram navigator. There is a list of supported notations in EPC diagram.

Creating event-driven process chain (EPC) diagram

EPC diagram , abbreviation for event-driven process chain diagram, is a flowchart based diagram that can be used for resource planning and identifying possible improvements of a business process.

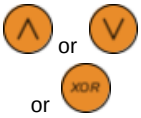
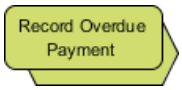
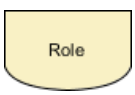
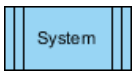


A sample EPC diagram

Creating EPC diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **EPC Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

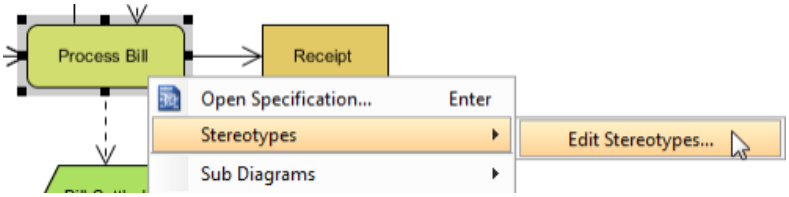
Notations

Name	Representation	Description
Event		An event describes what circumstances a function or a process works or which state a function or a process will be resulted in.
Function		A function describes the transformations from an initial state to a resulting state.
Operator		And - An and operation corresponds to activate all paths in the control flow concurrently. Or - An or operator corresponds to activate one or more paths among control flows. XOR - An XOR operator corresponds to make decision of which path to choose among several control flows.
Organization unit		An organization unit determines which person or organization within the structure of an enterprise is responsible for a specific function.
Control flow		A control flow connects events with function, process paths or operators that create chronological sequence and logical interdependencies between them.
Process path		A process path shows the connection from or to other processes.
Organization unit assignment		An organization unit assignment shows the connection between an organization unit and the function it is responsible for.
Information resource		An information resource portrays objects in the real world that can input data serving as the basic of a function or output data produced by a function.
Role		A role represents a unit, organization or a party that performs a function in a process.
System		A system is the provider of functions in a process.
Information flow		Information flow shows the connection between functions and input or output data, upon which the function reads changes or writes.

Applying stereotype to EPC elements

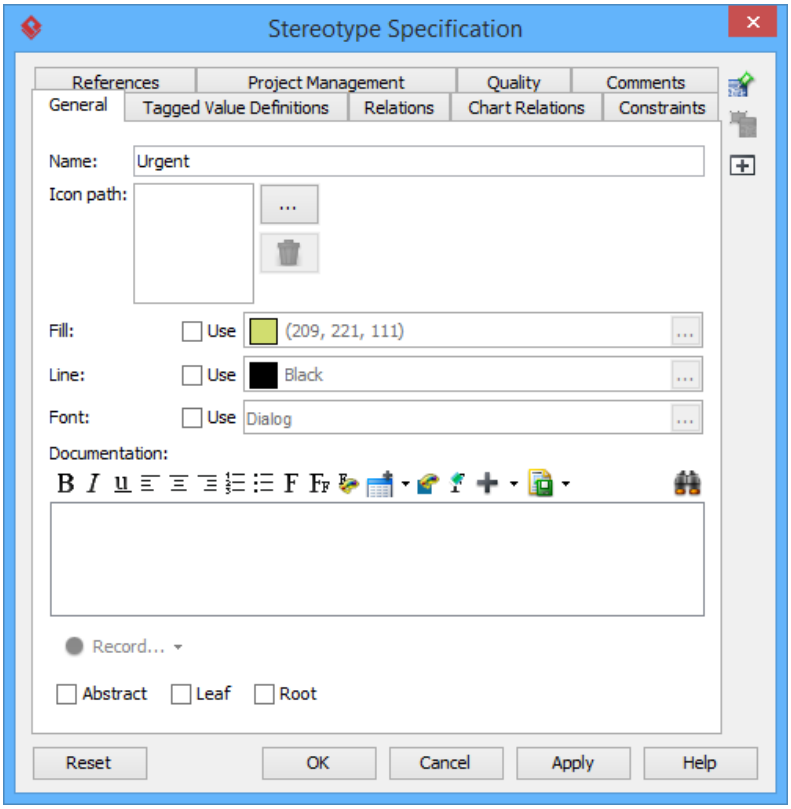
A stereotype defines how a model element may be extended. It also enables the use of platform or domain specific terminology or notation in place of or in addition to the one used for the extended metaclass. You can apply one or more stereotypes to model elements and decide whether or not to visualize the stereotype or tagged values in views. To apply stereotype to model element:

1. Right click on the model element or the view of the model element that you want to apply stereotype to. Select **Stereotypes > Stereotypes...** from the pop-up menu. As a side note for you, once you have ever applied a stereotype on the selected kind of element, you can re-select the same stereotype in this popup menu.



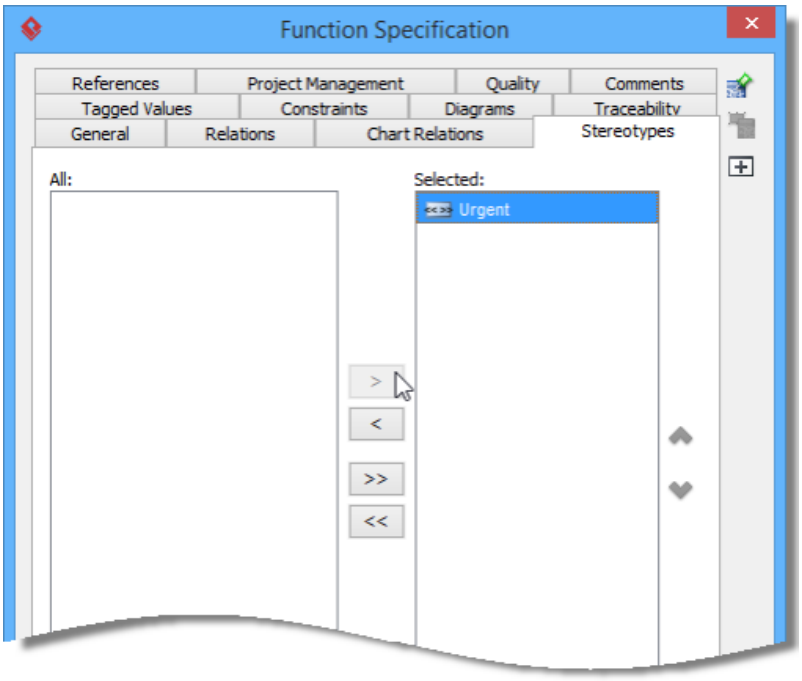
Open Stereotypes page

2. In the model element specification, open the **Stereotypes** tab and then click on **Edit Stereotypes...**.
3. In the **Configure Stereotypes** dialog box, click **Add**....
4. In the **Stereotype** specification dialog box, enter the name of stereotype and click **OK**.



Naming stereotype

5. This goes back to the specification dialog box. Select the stereotype you want to apply, then click **>** to assign it to the **Selected** list.



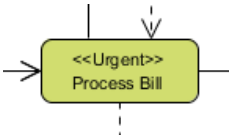
Selecting stereotype

NOTE: You can also double click on a stereotype to apply it.

NOTE: While clicking on **>** applies the selected stereotype to model element, clicking on **<** removes a stereotype selected in **Selected** list.

To apply all available stereotypes to model element, click on **>>** and likewise, clicking on **<<** to remove all the applied stereotypes.

6. Click **OK** to confirm. The stereotype will then be shown within a pair of guillemets above the name of the model element. If multiple stereotypes are applied, the names of the applied stereotypes are shown as a comma-separated list with a pair of guillemets.

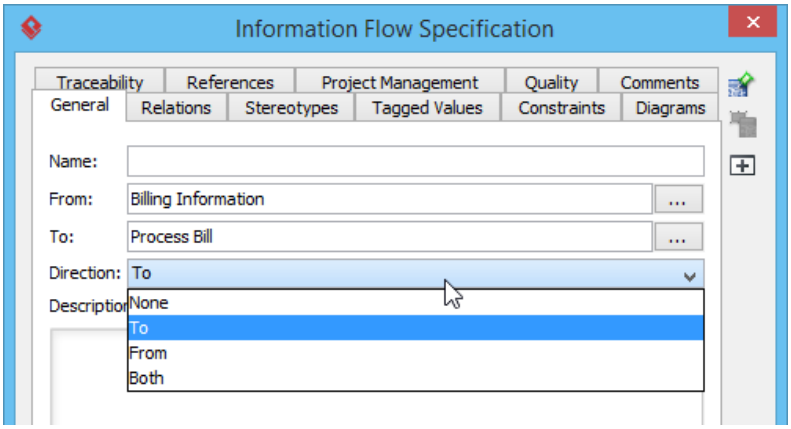


Stereotype is applied to element

Changing the navigability of an information flow

By default, arrow head is shown at the to-side of an information flow, representing a flow in "to" direction. If you want to represent that the flow has no direction, to flow back to the from end or flow in both sides, perform the steps below:

1. Right click on the information flow and select **Open Specification...** from the popup menu.
2. In the **Information Flow Specification** window, set **Direction** to **None/From/Both**.



Setting the direction of an information flow

3. Click **OK** to confirm the change. The diagram is updated.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Process map diagram

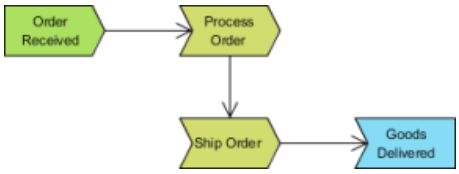
Process map diagram gives an overview to show the processes needed to approach to a business goal. This chapter teaches you how to create a process map diagram.

Creating process map diagram

This page teaches you how to create a process map diagram through the diagram navigator. There is a list of supported notations in process map diagram.

Creating process map diagram

Process map diagram gives an overview to show the processes needed to approach a business goal. It is rather in an upper level to analyze and understand a business process.



A sample process map diagram

Creating process map diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Process Map Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

Name	Representation	Description
Process	Process Order	A process is a part of process flow in achieving a goal.
Send	Order Received	An event that initiate the process chain.
Receive	Goods Delivered	The result of a process chain.
Process Link	⟶	The flow of process.

A list of supported notations in process map diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Organization chart

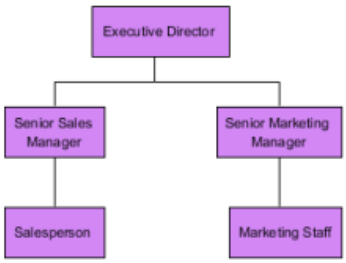
An organization chart is a diagram that visualizes the formal structure of an organization as well as the relationships and relative ranks of its positions. This chapter teaches you how to create an organization chart.

Creating organization chart

This page teaches you how to create an organization chart through the diagram navigator, as well as the steps to create subordinate, coworker and relocate branch.

Creating organization chart

An organization chart is a diagram that visualizes the formal structure of an organization as well as the relationships and relative ranks of its positions. It is usually drawn and read from top to the bottom. The default unit will pop out when a new organization chart is created.
In Visual Paradigm, organization chart is not only a diagram but also a reference used for other parts of your model. For example, you may use an organization chart to depict the company hierarchy involved in a business process model. Its prime function is to help a business analyst to visualize efficiently the company structure as well as the division of works when performing business analysis.
The organization chart sample is shown as below:



Organization chart sample

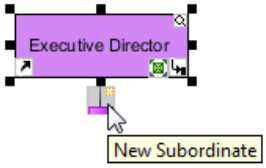
Creating an organization chart

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Organization Chart.**
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating a subordinate

Subordinate, which is subjected to the superior, is belonging to a lower rank. To create a subordinate under a superior unit:

1. Move mouse pointer on a unit and press its resource icon **New Subordinate**.



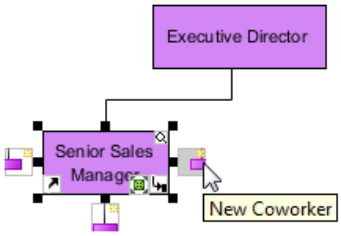
Create subordinate

2. Name the newly created subordinate unit and press **Enter** to confirm editing.

Creating a coworker

The coworker is a fellow worker of the same rank to the branch next to it. To create a coworker next to an existing unit:

1. Move the mouse pointer a unit and click its resource icon **New Coworker**, either on its left or right hand side. While clicking left resource icon will create coworker on the left of the unit, clicking right resource icon will create coworker on its right.



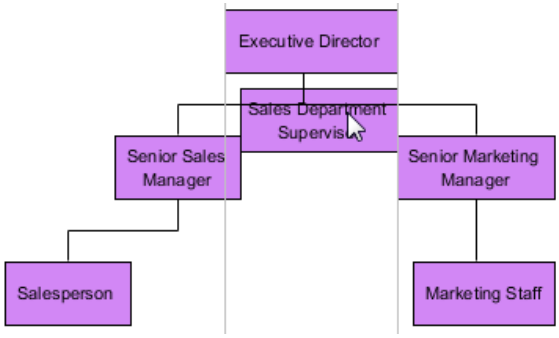
Create coworker

2. Name the newly created coworker unit and press **Enter** to confirm editing.

Relocating a branch

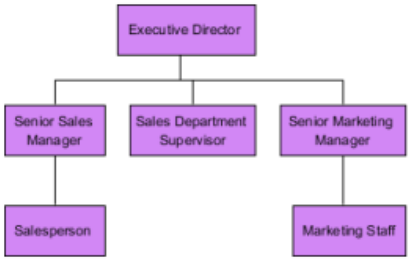
A unit can be relocated even when it has been placed under the subordination of another unit.

1. Press on a branch you want to relocate and drag it to the preferred branch.



Moving a unit

2. Release the mouse to confirm the position.



Completed relocating a branch

> **NOTE:** If you are not satisfied with the relocation, press **Esc** to cancel

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

RACI chart

A RACI chart is a matrix capable of showing how people or roles are related to business activities in a business process. You may form a RACI chart from a BPD to record the responsibility roles among swimlanes (i.e. Pool and lane) and activities (i.e. Task and sub-process).

Creating RACI chart

This page teaches you how to create a RACI chart through generating one from BPD.

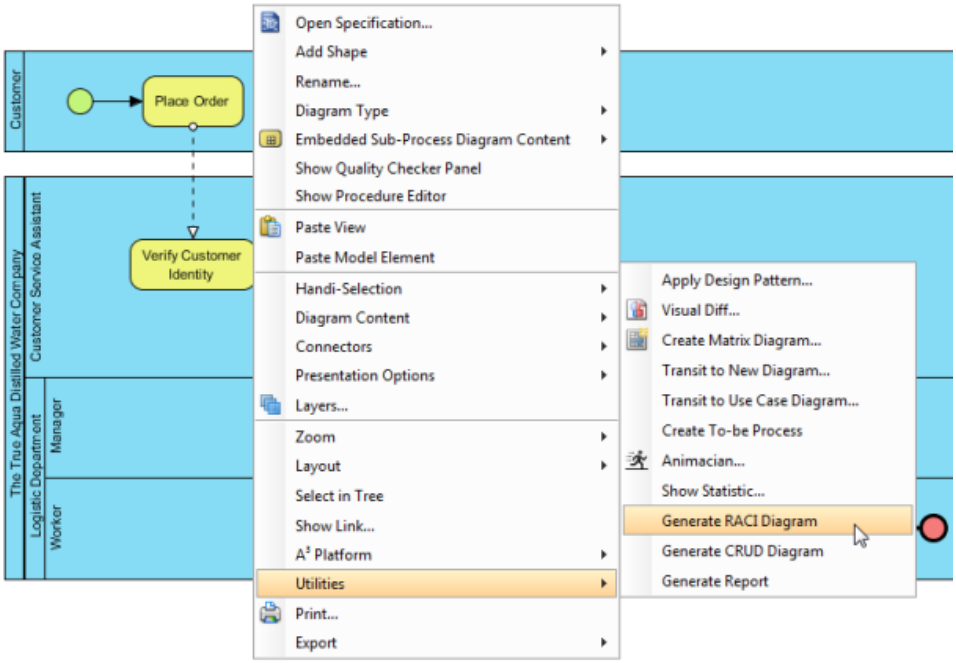
Creating RACI chart

A RACI chart is a matrix capable of showing how people or roles are related to business activities in a business process. You may form a RACI chart from a BPD to record the responsibility roles among swimlanes (i.e. Pool and lane) and activities (i.e. Task and sub-process. At any intersection of the participant and activity you can assign participant any of the four available responsible roles for a particular activity:

- Responsible - The participant who do the activity.
- Approver - The participant who approves or disapprove against an activity.
- Consulted - The participant who need to comment on the activity.
- Informed - The participant who need to be informed for any update about the activity.

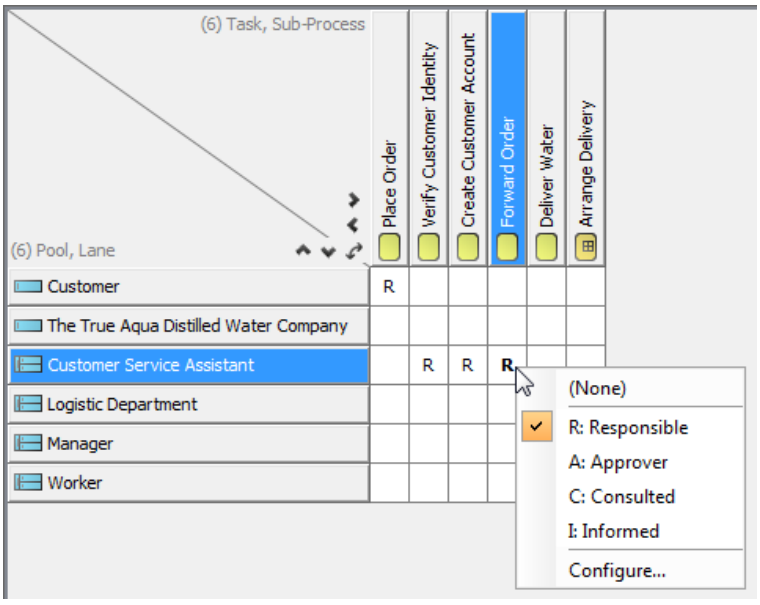
To create a RACI chart from BPD:

1. Right click on the background of a BPD and select **Utilities > Generate RACI Diagram** from the popup menu.



Generate RACI chart from BPD

2. The role R, abbreviation for responsible, is automatically assigned to the intersection of participant and activitiy whenever the activity is put inside an participant. You may add or remove roles by clicking on a cell in the chart and update the role selection.



Update the responsibility roles

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

- Contact us if you need any help or have any suggestion

UeXceler

What is UeXceler?

Learn what UeXceler is.

Why UeXceler?

Know the benefits of following UeXceler in requirements gathering.

Overview of UeXceler - The Five Phases

Familiar yourself with the 5 phases of UeXceler.

Responsibilities of UeXceler Roles

Check out the roles involved in UeXceler.

What is UeXceler?

One of the keys to a successful software project is to build the system according to users' needs, but unfortunately, many software companies fail to gather the right requirements from stakeholders, mostly due to stakeholders fail to express their needs and concerns clearly and thoroughly.

While it is easier for stakeholders to tell you what they need when they can experience a fully functional system or at least, prototypes, it is still time consuming in developing any product or semi-product. Consequently, most of the development teams end up with two extremes: gain real customer feedback at the end of project, or spent a lot of time and effort at the beginning of project until any real feedback is gathered. UeXceler strikes a balance between the two extremes by letting user to see and feel the final product early in requirements gathering phases, but without spending too much resources of the development team.

UeXceler is a guideline for software development teams to identify real system requirements at the very beginning of development projects. UeXceler emphasizes the involvement and contribution of stakeholders. Throughout the requirement gathering process, stakeholders are encouraged to express their concerns and needs, which are then converted into use cases, user scenarios and requirements of the solution. Wireframes are created for the illustration of the end product in quick and cost-effective manner. At the end of the requirements gathering process, not only stakeholders see a list of requirements but a set of business stories (scenarios) and a visualized system through wireframes. Both stakeholders and development teams can comfortably consent to the planned requirements specification.

UeXceler stresses the importance of collaboration with customer rather than any kind of negotiation. By adopting UeXceler in requirements gathering, stakeholders can easily preview what the system will look like without any functional features or prototype developed. Software development lifecycle is highly optimized by having clear direction, resulted by clear requirements. Solid partnership can be built between business stakeholders and technology team, which facilitates the accurate identification of system requirements and results in building software system that meets or even excels customer's expectation.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Why UeXceler?

As a contemporary way for requirement gathering, UeXceler addresses key challenges of historical ways for identifying system requirements. UeXceler focuses on identifying stakeholders and users' real needs and the result is both a better software product and a higher customer satisfaction. The following lists out the benefits that different roles of people, both in the stakeholders and people in project team can gain by adapting UeXceler.

To Stakeholder and User

- UeXceler values the expertise and opinions contributed by stakeholders. Instead of guessing what the stakeholders needs, stakeholders are encouraged to share their knowledge, concerns and suggestions, which are then converted into requirements of system. Because stakeholders do have say in the project, they feel empowered, respected and trusted. Customer satisfaction is, as a result, significantly increased.
- Because the solution is mostly made from stakeholders' input, it is more likely for them to buy into the final solution.
- Stakeholders are presented both the system interactions and wireframe early in the requirement gathering phases. They know how the end product will look like in early stage and feel more comfortable.
- With the early inspection of product design, modification and new ideas can be added into the project without significant cost.
- Because the project team develops feature according to the priority agreed with stakeholders, stakeholders can be more confident that the business values and business priorities are aligned with development activities. Critical features can therefore be done, tested and even used in early iterations.

To Project Team

- UeXceler values the participation of stakeholders in developing a good piece of system. The active involvement of stakeholders allows clear project direction throughout the product development, and avoids spending valuable time on re-work due to any late feedback.
- Development activities are determined based on the use cases identified. By evaluating the nature of use cases, parallel development can be planned and conducted. This saves a great deal of time on unnecessary waiting.
- The ability to prioritize use case and requirements allows for delivering highest priority features earlier, avoiding risk resulted by late change.
- The seamless connection between flow of events and wireframe provides an effective way for stakeholder to feel the user experience without any real product get done - not even any prototype.
- Save time by avoiding all sorts of non-productive work - no unnecessary documentation, no ineffective meetings, no complex screen design and no prototyping.
- Use case based stakeholders meeting makes the meeting fewer, shorter and more focused.
- Definition of success is very clear at the beginning of development – you have the flow of events, requirements, wireframes as targets.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Overview of UeXceler - The Five Phases

Identify

The purpose of the Identify phase is to outline the objectives that the client wants to achieve by using the end product and these objectives, in UeXceler, are called use cases. Use cases provide insight into the overall needs that must be addressed by the system from end-user perspective. By the end of this phase, you should have a prioritized list of use cases, a list of actors (i.e. users) who will interact with the system and a list of front-line stakeholders who have subject matter knowledge to contribute for each use case.

Discuss

The purpose of the discuss phase is to gain detailed understanding of use case-related ideas such as the preferred way of accomplishing the use case, related concerns and specific requirements. Through the discussions with stakeholders, basic system flows and user concerns are captured and are drafted as simple use case notes, which may be converted into use case flow of events and system requirements in the Elaborate phase.

Elaborate

The purpose of the Elaborate phase is to convert the project-related ideas gathered in the Discuss phase into description of product behaviors and a consolidated list of system requirements. Standard flow of events scenarios are created based on the use case notes gathered to describe the way how user can interact with the system to achieve the use cases. Concrete requirements are also created from some of the use case notes, which are likely to be the concerns raised by the stakeholders.

Design

The purpose of the Design phase is to produce wireframe of the system to be built, for visualizing user interface and representing screen flows. Wireframing is a technique to illustrate and modify the user interaction experience in a very quick manner without big investment. In UeXceler, wireframes are suggested to be created in accordance with use case flow of events to visualize the user experience at particular states throughout the system interactions.

Consent

The purpose of the Consent phase is to seek approval from stakeholders prior to initiate any actual development activity. Stakeholder is shown both the wireframe flow (like a screen flow), so that he/she can verify easily the behaviors of the system and to request modifications, if necessary.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Responsibilities of UeXceler Roles

Here are the most common roles that appear in the UeXceler gudeline. Notice that different teams and companies may have different titles for those roles (or no name at all). And sometimes, a team member has to play several roles due to the scale of project team, the type of project, the expertise and experience of that member, etc.

System analyst

A system analyst (sometimes simply referred as analyst) is responsible for identifying project stakeholders, identifying actors and their use cases, documenting use case flow of events, managing the gathering and prioritizing of system requirements and seeking the approval of system design from stakeholders. System analyst has to work hand-in-hand with the business stakeholders to complete these tasks.

In some development teams or projects, the role system analyst may not exist and the developers can take the role of system analyst.

Business stakeholder

A business stakeholder is someone who has the business vision and business authority in defining “what” the end product needs to accomplish, that is, the use cases. Business stakeholder works with system analyst in identifying use cases, noting business problems and concerns.

Business stakeholder may or may not be involved in day-to-day operations, but is knowledgeable enough to know the current state of the business and the problems that need to be addressed.

Front-line stakeholder

A front-line stakeholder is anyone who has a stake in the success of the project or have subject matter knowledge to contribute. He or she doesn't necessarily to be the user of the system to be develop, but must be capable of delivering project-related ideas that allows the development of system to achieve the end users' expectations.

A front-line stakeholder is responsible for defining preferred way to operate with the system. When use cases are identified, different groups of front-line stakeholders will be interviewed. Ideas are gathered from the front-line stakeholders as use case notes and those notes will become flow of events and concrete requirements of use cases.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Use Case Notes

Note down use case related ideas in Use Case Note

Learn how to take notes for use case related ideas.

Produce use case scenario from notes

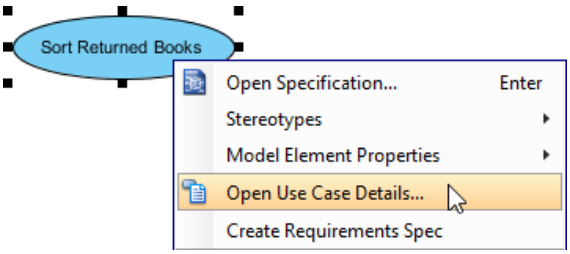
Learn how to instantly generate scenario from Use Case Notes.

Note down use case related ideas in Use Case Note

While meeting with users is an important part of requirements capturing, multiple meetings are essential for clarifying what user really wants, for clarifying usage of the system, and for obtaining users' consent to the requirements. Use Case Notes is designed for you to note down the discussion during requirements capturing meetings. You can outline the workflow, write down users' concerns and conclusion in a Use Case Note. You can also turn your note item into use case scenario or requirements by just one click. Please watch below video tutorial for details. All Use Case Notes will be kept under the use case. You don't need to spend extra effort to manage your meeting notes any more.

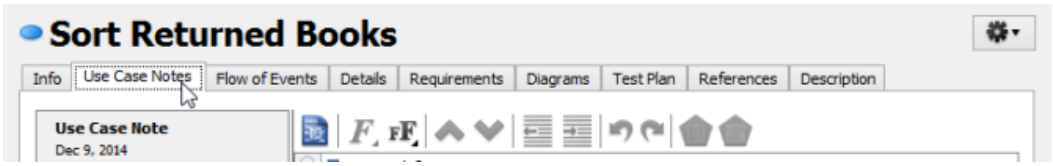
Accessing use case notes of a use case

The **Use Case Notes** (editor) is grouped as part of the **Use Case Details**. There are mainly two ways you can take to access the use case notes of a use case. When you are reading a use case diagram, you can access use case notes of a use case by right clicking on that use case and selecting **Open Use Case Details...** from the popup menu.



Open Use Case Details

Then, open the **Use Case Notes** tab in **Use Case Details**.

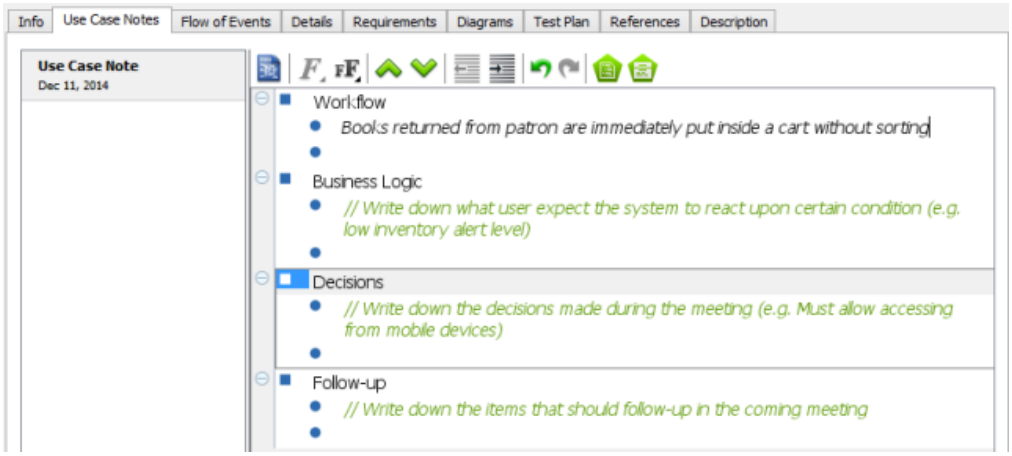


Open Use Case Notes

If you see the panes like **Diagram Navigator** and **Property Pane** minimized, with help contents showing in the middle of screen, this means that this is the first time you open the **Use Case Note**. The panes are minimized to maximize the viewable list area. Click **Keep Change** if you want to keep the panes minimized. If you want to roll back to the original screen layout, click **Revert**. And if you keep the change now, you can still customize the screen layout by applying a perspective later on (**View > Open Perspective**).

Entering use case notes

Once you have opened the **Use Case Notes** of a use case, you can start entering notes. There is a pre-defined template, with four points in it - **Workflow**, **Businses Logic**, **Decisions** and **Follow-up**. You can follow these points in note taking. All you need to do is to click on the green text and replace it by entering your note content.



Entering a note by following the template

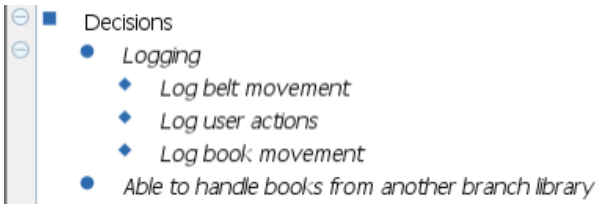
If you do not want to follow the points suggested by the templates, you can delete it by highlighting the rows and pressing the **Delete** key.

To enter a note, type the note content in the editor.

To create a new note, press **Enter**.

Working with nested notes

Different kinds of use case -related ideas can be recorded by creating multiple nested notes. You can press **Tab** to indent, and press **Shift-Tab** to reduce indentation.



Nested notes

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Take Use Case Notes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

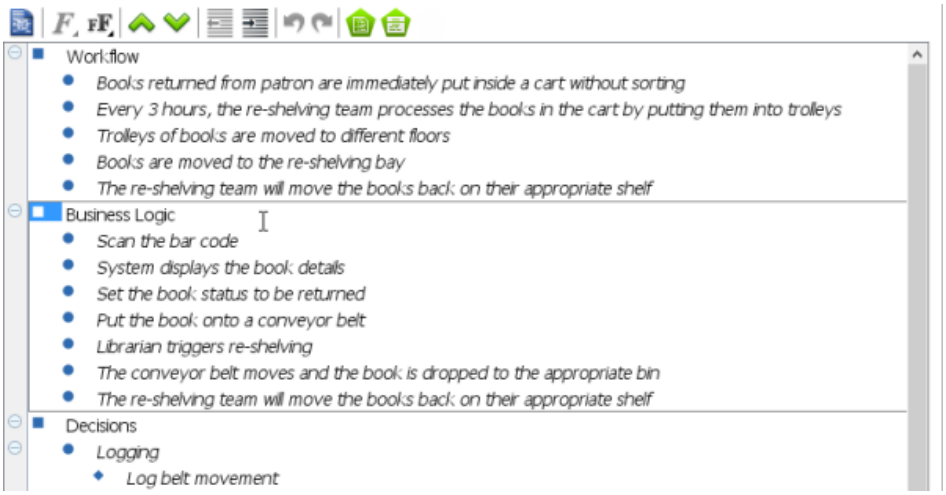
Produce use case scenario from notes

When you are meeting with the stakeholders, they will let you know their expectations regarding to the features of the system to be developed. Very often, it involves the system behaviors they preferred. By noting down the preferred system behaviors as use case notes, you can easily produce an initial use case scenario and to make further changes in it.

Producing a new use case scenario from use case notes

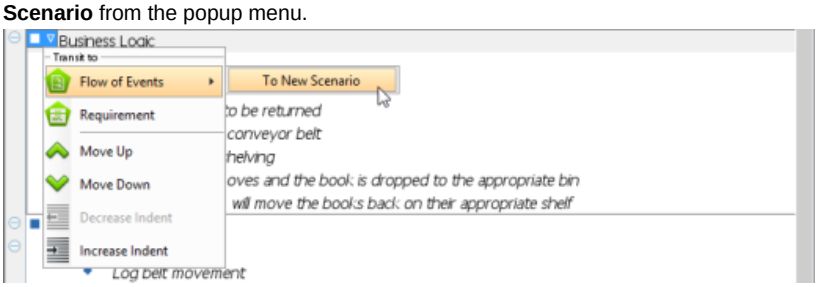
To produce a new use case scenario from use case notes:

1. Open the **Use Case Notes** of the desired use case.
2. Move the mouse pointer over the parent note item where the suggested system behaviors are recorded.



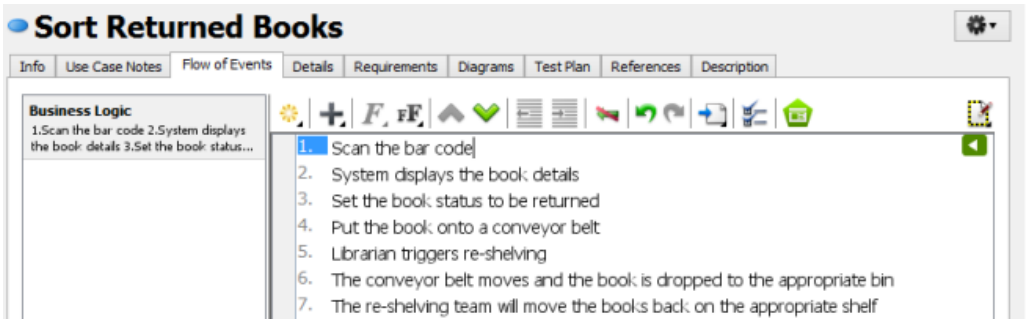
Moving mouse pointer over a note item

3. Move the mouse pointer to the beginning of the note item. Click on the down arrow next to the bullet point and select **Flow of Events > To New Scenario** from the popup menu.



Creating a new scenario

This produces a new scenario, with the text of the chosen note item becomes the name of scenario and the sub-note items becoming the steps of the scenario.



Scenario produced

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Take Use Case Notes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Use Case Flow of Events

Develop use case scenario in Flow of Events editor

Learn how to write use case scenario.

Working with multiple use case scenarios

Learn how to represent multiple stories by creating multiple use case scenarios

Perform scenario-based wireframing

Learn how to perform scenario-based wireframing.

Wireframe playback

Learn how to play your wireframes, which is important in a presentation.

Producing requirement specification

Learn how to generate requirement specification from use case flow of events.

Elaborating use case

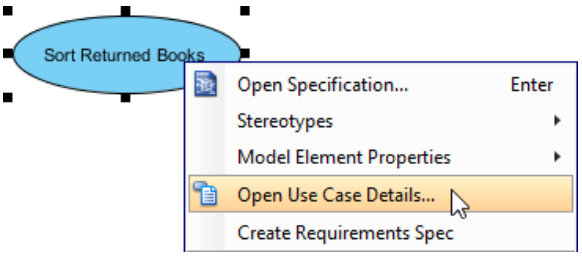
Learn how to elaborate use case with sequence diagrams.

Develop use case scenario in Flow of Events editor

A use case scenario refers to the steps required to go through and then achieve a user goal. When you want to specify the process of how to achieve a use case clearly as a guideline, you can define it step by step in the flow of events editor. This page will introduce the flow of events editor by describing its features, such as model element link presentation options and use case extension.

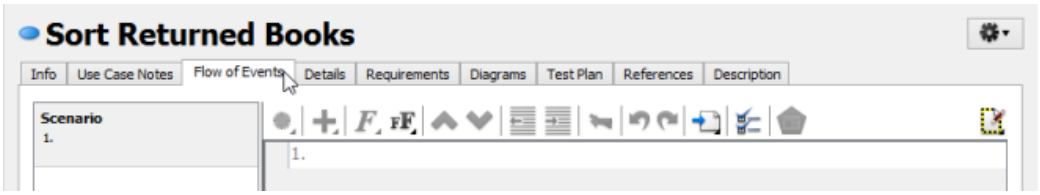
Accessing use case scenarios of a use case

The **Flow of Events** (editor) is grouped as part of the **Use Case Details**. There are mainly two ways you can take to access the scenarios of a use case. When you are reading a use case diagram, you can access use case notes of a use case by right clicking on that use case and selecting **Open Use Case Details...** from the popup menu.



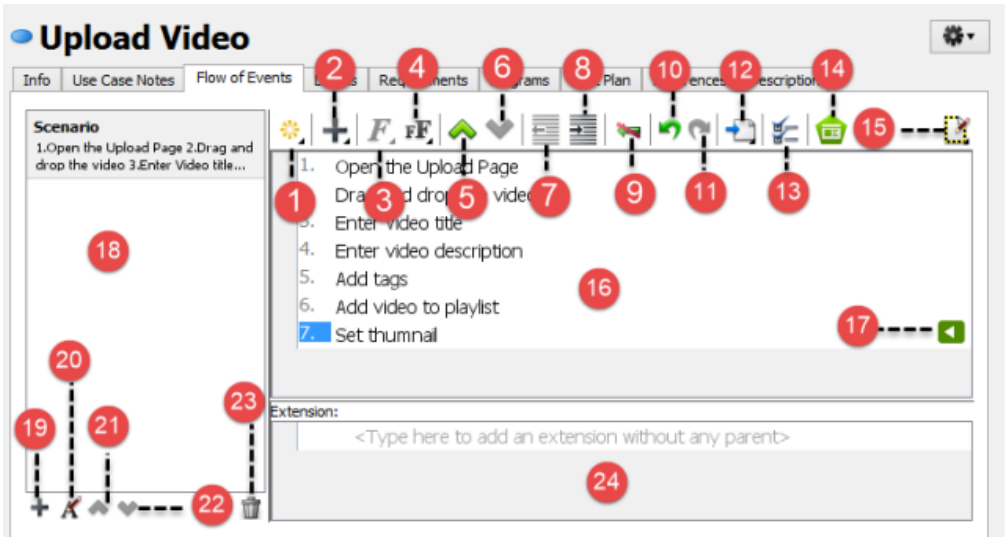
Open Use Case Details

Then, open the Flow of Events tab in **Use Case Details**.



Open Flow of Events

Overview of the Flow of Events editor



Flow of Events editor

No.	Name	Description
1	Add Step	You may select various functions in the menu: **Step (Enter)**: Enter next step. **Extension (Shift+Enter)**: Enter extension for the selected step. **If**: Enter conditional situation for the selected step. **Else if**: Insert another situation under If. **Else**: Insert it to control **If** and **Else if**. **Clear control**, **Go to previous condition** and **Go to end**. **While**: Perform some actions as long as the condition stated in the while clause is valid. **For each**: Perform some actions by walking through each item as stated by the for each clause. **Loop until**: Perform some actions until condition stated in loop remaining valid. **Exit**: Exit in the middle of loop. **Jump**: Insert **Jump** to manipulate the sub-step after the variable situation happened. You can go back to the previous step by clicking the small yellow arrow.
2	Select model element	Click to select an existing use case/actor/requirement/business rule/class/diagram in the current project and insert it into the selected step as referenced link.

3	Font	Select font effect for the highlighted text: **Bold**, **Italic** and **Font Color**.
4	Font size	Click to adjust the font size of text in **Requirement List**.
5	Move Up	Click to move the selected step (in the scenario) upwards.
6	Move Down	Click to move the selected step (in the scenario) downwards.
7	Decrease Indent	Click to decrease the indentation of the selected step by one level.
8	Increase Indent	Click to increase the indentation of the selected step by one level.
9	Delete Step	Click to delete the selected step(s).
10	Undo	Undo the last editing action made in the **Flow of Events** editor.
11	Redo	Redo the last editing action reverted by undo.
12	Synchronize to diagram	Synchronize the flow of events to activity diagram or sequence diagram. If no activity diagram has created previously, a new activity diagram will be generated.
13	Model Element Link Presentation Options	To select the presentation for use case link/actor link/requirement link/business rule link. **Name** : To display the name of model element only. **ID** : To display the ID of model element only. **ID: Name**: To display both the name and ID of model element.
14	Show Wireframe	Click to hide/show the wireframe pane that displays the wireframe(s) associated with the selected step in the scenario.
15	Testing Procedure	In order to ensure that a use case is able to achieve as your expectation, you can test each step of procedure defined in flow of events to view whether they will produce prospective result or not.
16	Scenario	Content of the scenario.
17	Define wireframe	Add or view the wireframe(s) associated with the selected step.
18	Scenario list	The scenarios of the use case is listed here.
19	Add Scenario	Click to create a new scenario. Mutliple scenarios can be created for describing the different ways of achieving the use case.
20	Edit Scenario	Click to rename the active scenario.
21	Move Up	Click to move the selected scenario upward.
22	Move Down	Click to move the selected scenario downward.
23	Delete Scenario	Click to delete the active scenario.
24	Extension	Use case extension pane that list the steps extended from the main flow.

Description of Flow of Events editor

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Writing effective use case
- YouTube Video - How to Manage Use Case Scenario with Flow of Events?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

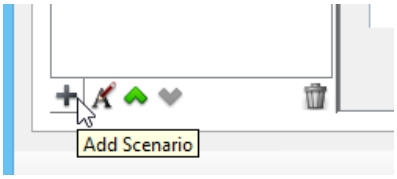
Working with multiple use case scenarios

Very often, a use case can be achieved by multiple ways. Take the cinema ticket selling system as an example. "Sell ticket" is one of its use cases. It can be achieved by two ways - selling tickets both online and by staff. Sometimes, different results may be produced under different conditions during the course of use case. Take ATM as an example. To withdraw cash from ATM may results in a success or a failure when there is an insufficient funds. All these are variations of a use case and can be described by creating multiple use case scenarios.

Creating multiple use case scenarios

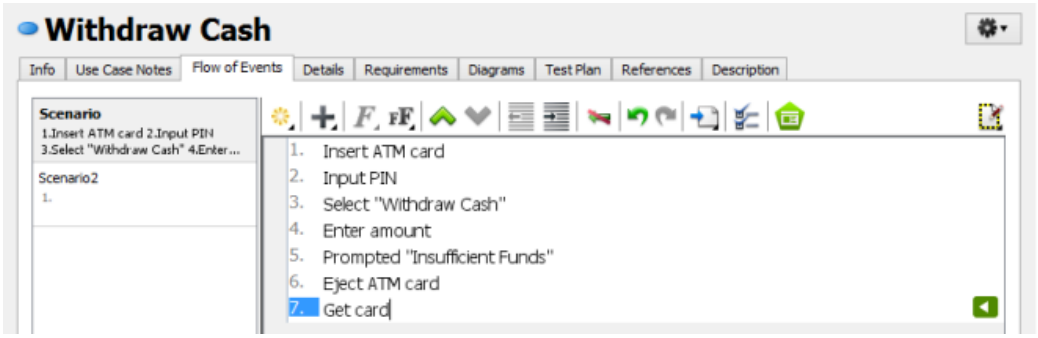
To create a use case scenario:

1. Open the Flow of Events of the desired use case.
2. Click **Add Scenario** at the bottom left corner.



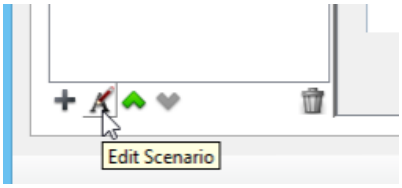
Add a scenario

3. Start entering the steps of the new scenario in the **Flow of Events** editor.



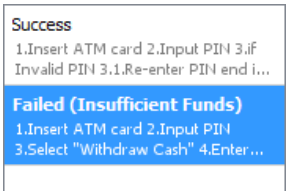
Enter the steps of the new scenario

4. Rename the scenarios. You can rename scenario by clicking **Edit Scenario** at the bottom left.



Edit Scenario

5. You are prompted for a new name. Enter the name and confirm editing.



Scenarios are renamed

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Writing effective use case
- YouTube Video - How to Manage Use Case Scenario with Flow of Events?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Perform scenario-based wireframing

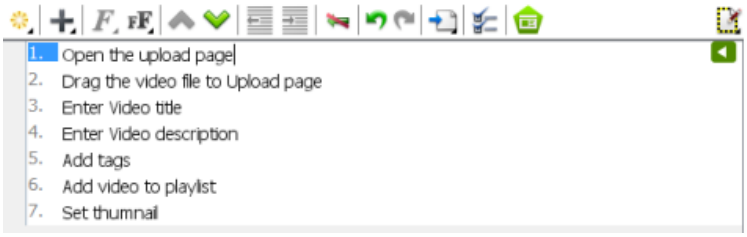
Showing a screen flow of the system to your customer guarantees your customer knows what will be delivered by the end of the project. This also saves us a lot of efforts in modifying the system in later stage of the development process because customer is involved and informed in early time. Instead of doing heavy system prototyping, you can "sketch" the user interface by wireframes. You can either sketch a new wireframe or reuse existing wireframes in each step of your scenario. The wireframe shows "just enough" information of the screen instead of the full details. The actual screen design will be produced at a later stage by referencing the wireframe.

By performing scenario-based wireframing, you can present your scenarios to your customer visually to obtain consent to the requirements easier.

Creating a wireframe for a step in scenario

To create a wireframe for a step in scenario:

1. Open the Flow of Events of the desired use case.
2. Click on the step that you want to create a wireframe.

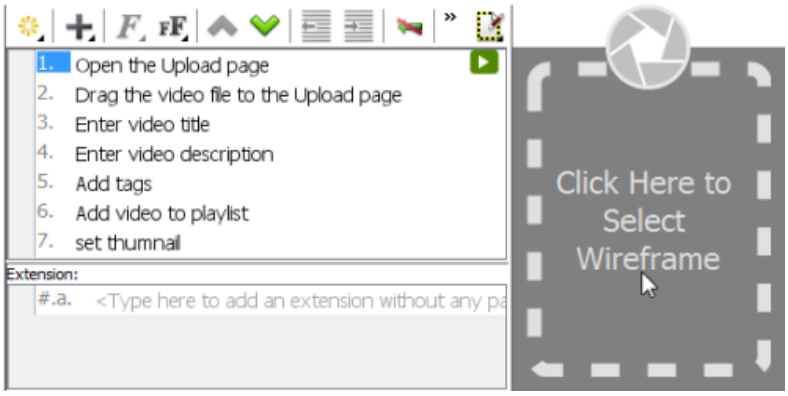


Selected a step in use case scenario

3. Move the mouse pointer over the **Define Wireframe** button (i.e. the green button) on the bottom-right of the step. Click on it.

Define a wireframe for a scenario step

4. This shows a gray pane on the right hand side. Click on it to select a kind of wireframe to create.

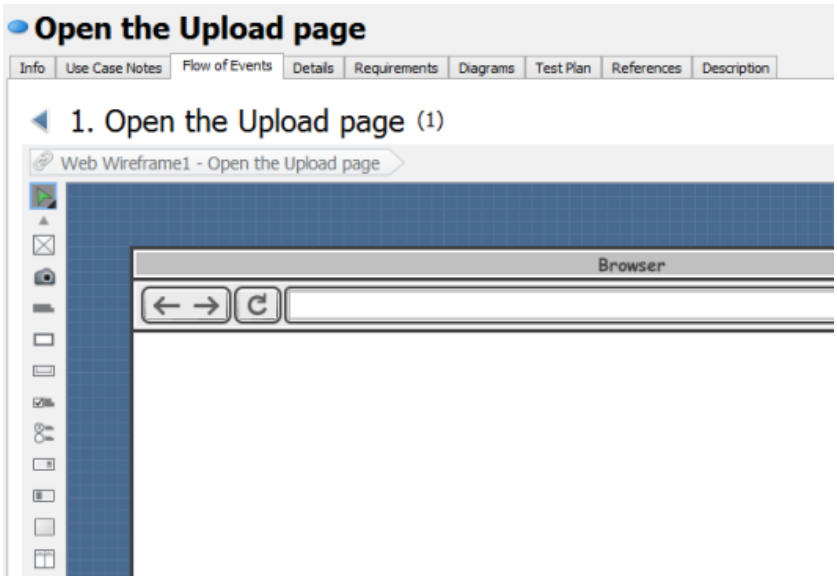


Select a wireframe

5. In the popup window, select the suitable type of device/platform for your application/system. If your system will run on multiple devices/platforms, please consider creating multiple scenarios.

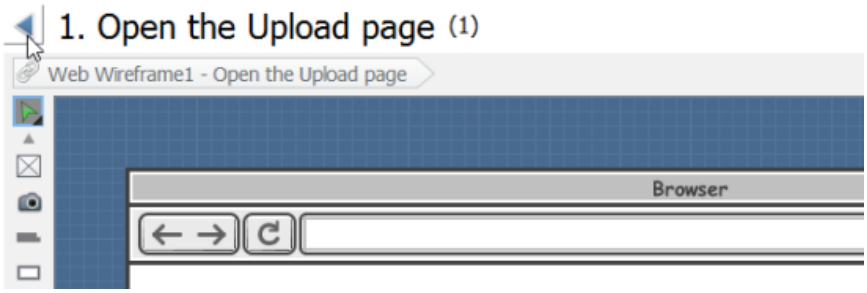
Select a type of wireframe to create

6. Click **New %TYPE% Wireframe** where **%TYPE% is** the type of device/platform you selected.
7. A blank, new wireframe appears and you can now begin editing.



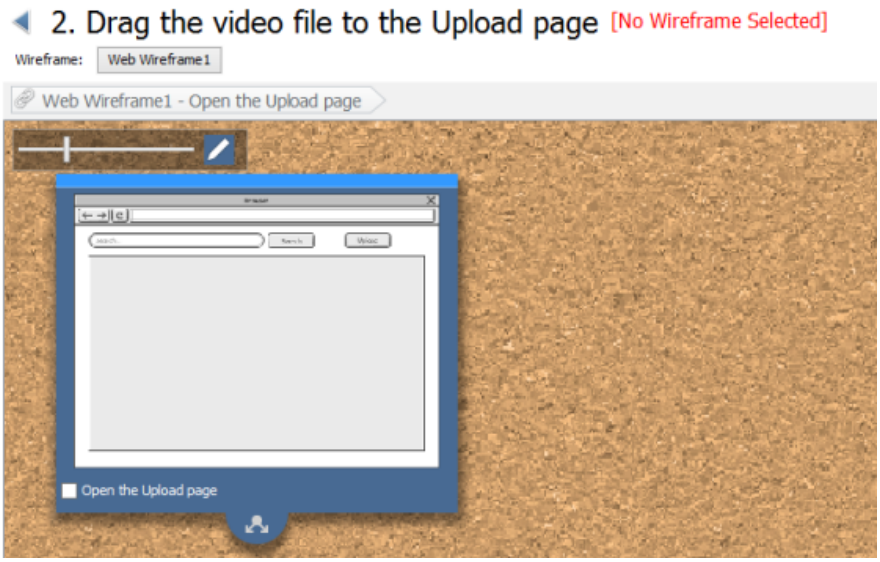
New wireframe created

8. When you have finished editing, you can go back to the scenario by clicking on the back button on top of the wireframe.



Go back to the flow of events

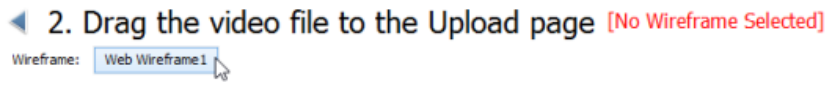
The above are the steps that involve in creating a wireframe from a scenario step when there is no wireframe in your project. Once you have created a wireframe, you will see something different after step 4, when you attempt to create a wireframe for another scenario step. Here is what you will see:



State overview

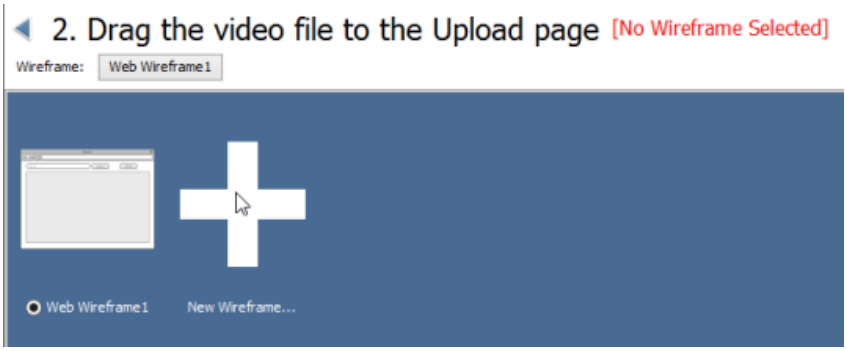
If you want to create an entirely new wirefram:

1. Click on the button next to **Wireframe:**.



Choose another wireframe

2. Click **New Wireframe...**.

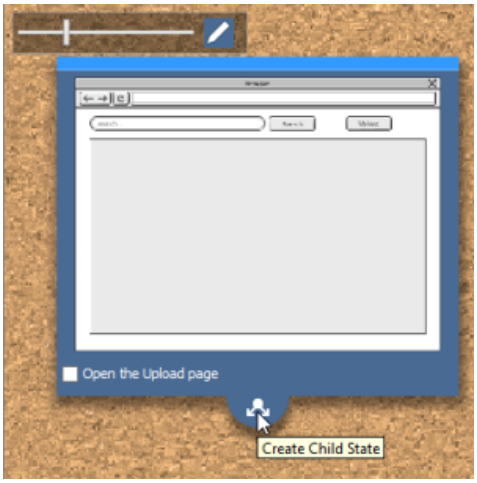


Create a new wireframe

3. The remaining steps are same as those mentioned above, starting from step 5.

If you want to re-use an existing wireframe but make a bit of change, you should create a child state instead:

1. Click **Create Child State** below the thumbnail of wireframe to create a child state under it.



Create a child state

2. Once clicked, a new wireframe state will be created. You can start editing it.

Selecting an existing wireframe for a step in scenario

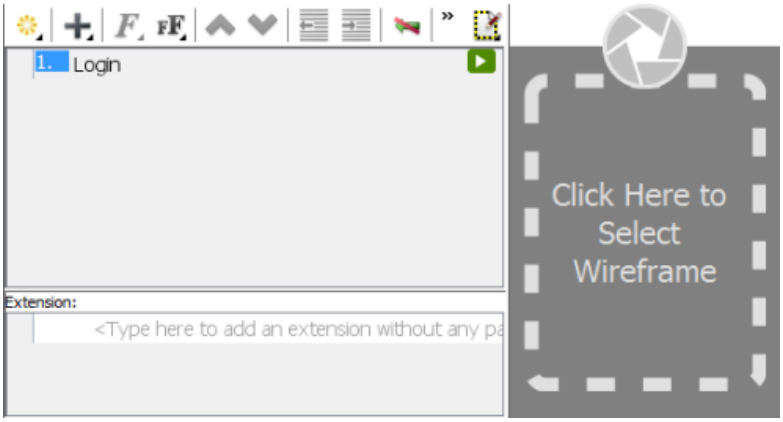
Sometimes, you may want to re-use a wireframe created earlier. For example, to reuse a wireframe about account login in scenarios that require user to login to do something.

To select an existing wireframe for a step in scenario:

1. Open the Flow of Events of the desired use case.
2. Click on the step that you want to associate a wireframe with it.

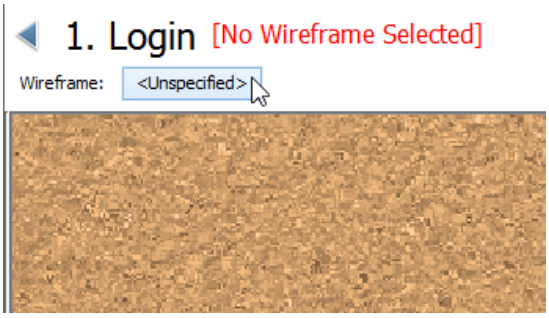
Selected a scenario step

3. Move the mouse pointer over the **Define Wireframe** button (i.e. the green button) on the right hand side of the step. Click on it.
4. This shows a gray pane on the right hand side. Click on it.



Select a wireframe for step

5. Click on the **<Unspecified>** button next to **Wireframe:**.



Clicking on Unspecified

6. Choose the wireframe for your step.

Choosing a wireframe

7. This shows the available states of the wireframe. Select the right one by checking the checkbox at bottom left corner. Make sure you did selected a state in this step. Without doing so, the wireframe won't be associated with the scenario step.

Selecting a wireframe state

8. Go back to the scenario by clicking on the back button on top of the state selection page.

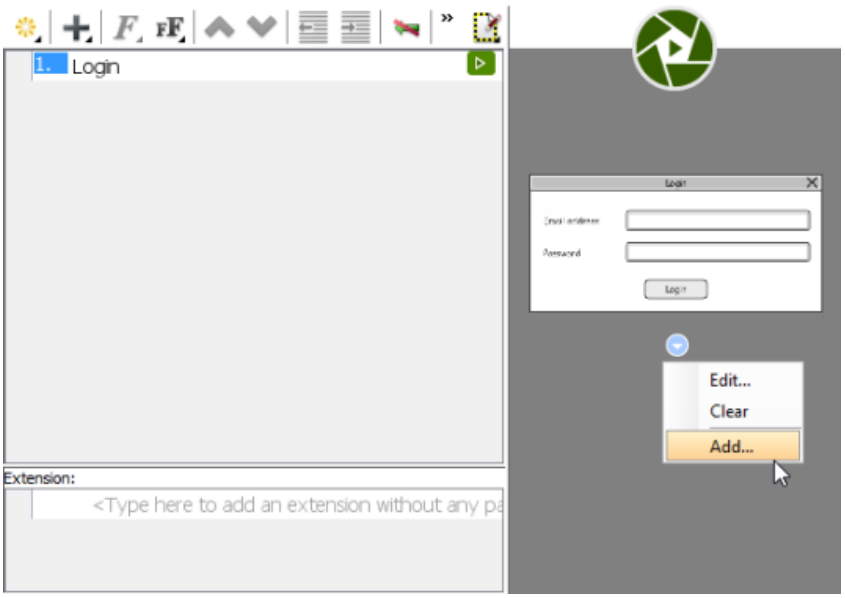
Go back to the flow of events

Adding extra wireframes to a step

If what you have written as a step of a scenario involves more than one screen change, you may need to add multiple wireframes to that step.

To add extra wireframes to a step:

1. Open the Flow of Events of the desired use case.
2. Click on the step that you want to add an extra wireframe to it.
3. Move the mouse pointer over the **Show Wireframe** button (i.e. the green button) on the right hand side of the step. Click on it.
4. There is a button under the thumbnail of the existing wireframe. Click on it and select **Add...** from the popup menu.

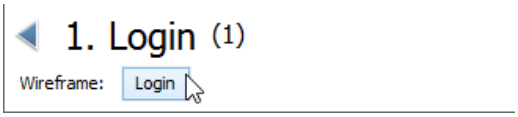


Add a wireframe to a scenario step

5. If you want to select another state of the selected wireframe, just check the checkbox of the wireframe state:

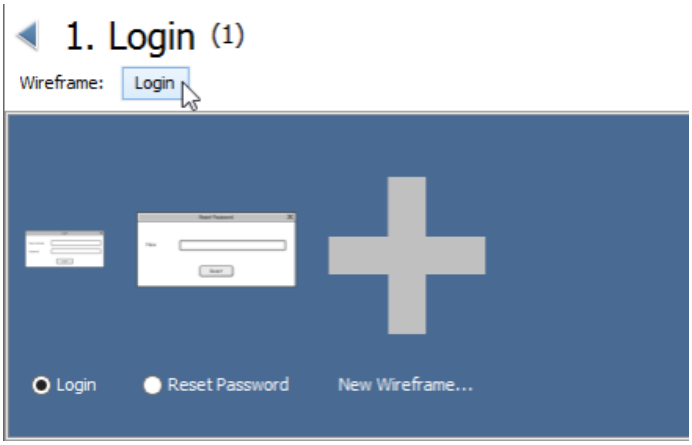
Selecting a wireframe state

If you want to select another wireframe, click on the button next to **Wireframe:**.



Select a wireframe

Then, select the wireframe or click **New Wireframe...** to create a new one.

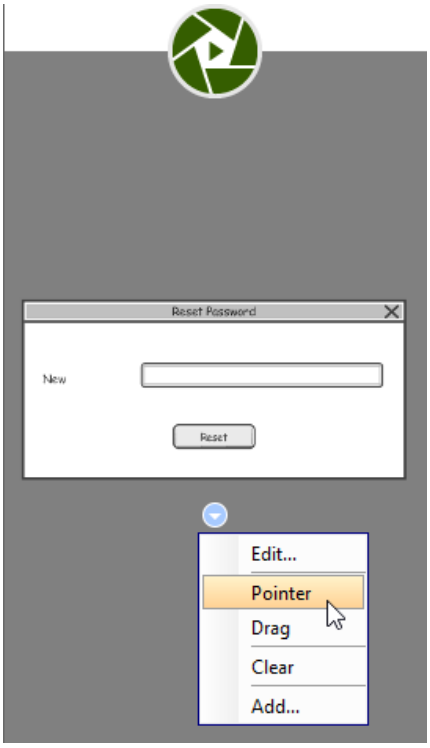


Select a wireframe

Using pointer / finger gesture

You can indicate in wireframe the position of mouse pointer, or the finger gesture required to execute an action. To do this:

1. Open the Flow of Events of the desired use case.
2. Click on the step that you want to edit its wireframe.
3. Move the mouse pointer over the **Show Wireframe** button (i.e. the green button) on the right hand side of the step. Click on it.
4. There is a button under the thumbnail of the existing wireframe. Click on it and select **Pointer/Drag/Finger Gesture** from the popup menu. Note that Pointer and Drag are available for Desktop and Web wireframe, while Finger Gesture is available for Android phone, Android tablet, iPad and iPhone wireframes.



Add a pointer to wireframe

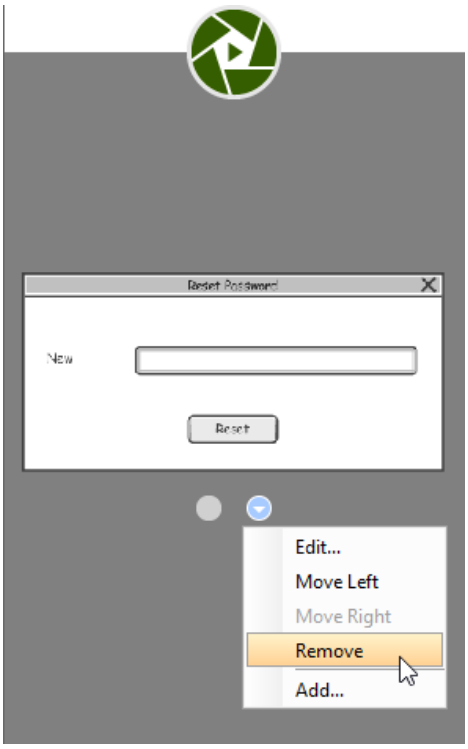
5. Drag on the pointer/drag/finger gesture symbol to reposition it.

Repositioning a pointer

Removing a wireframe from a step

To remove a wireframe from a step:

1. Open the Flow of Events of the desired use case.
2. Click on the step that you want to remove a wireframe from it.
3. Move the mouse pointer over the **Show Wireframe** button (i.e. the green button) on the right hand side of the step. Click on it.
4. There is a button under the thumbnail of the existing wireframe. Click on it and select **Remove** from the popup menu..



Remove a wireframe

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

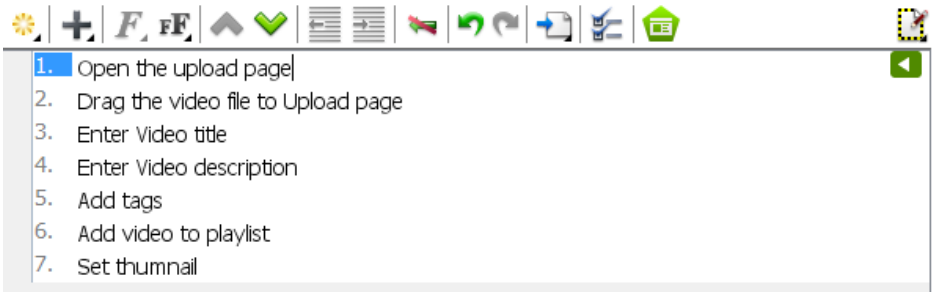
- Tutorial - Writing effective use case
- YouTube Video - How to Create Scenario-Based Wireframe?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Wireframe playback

Showing a screen flow of the system to your customer guarantees your customer to know what will be delivered by the end of the project. Visual Paradigm not only allows you to associate use case scenario with wireframes in illustrating system interations but also supports playing the wireframes associated with use case scenario. This can be very useful when you need to present the system design ideas to your customers and to look for their consent.

Playing wireframes

1. Open the Flow of Events of the desired use case.
2. Click on the first step of the scenario.



Use case scenario

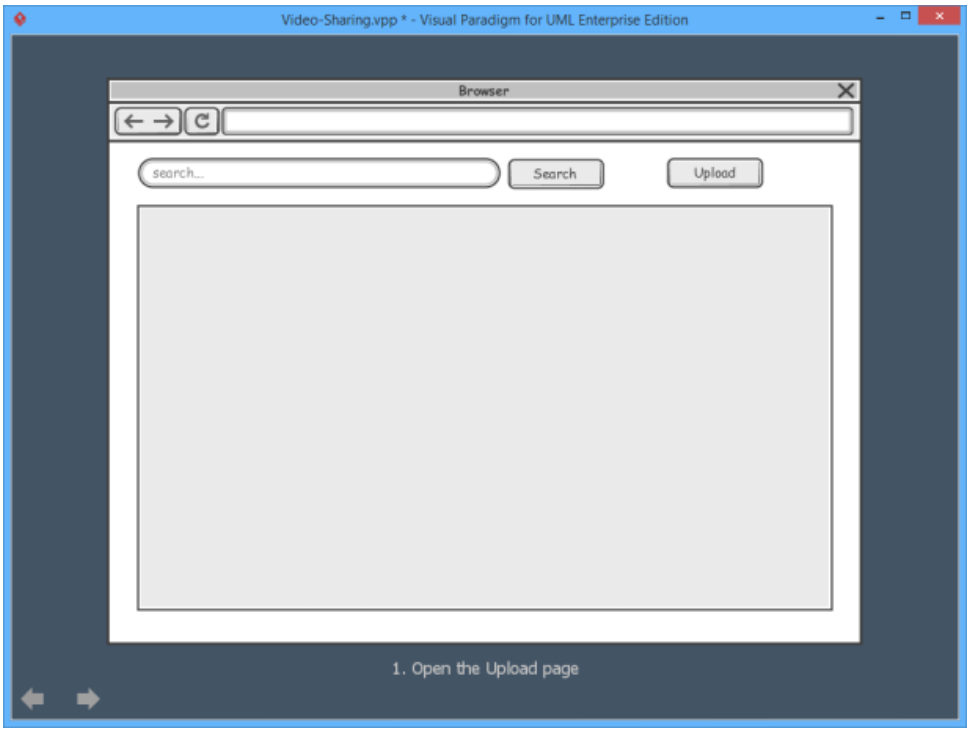
3. Move the mouse pointer over the **Show Wireframe** button (i.e. the green button) on the right hand side of the step. Click on it.

Show wireframe of a step in use case scenario

4. The preview of wireframe is now shown on the right hand side. On top of it there is a play button. Click on it.

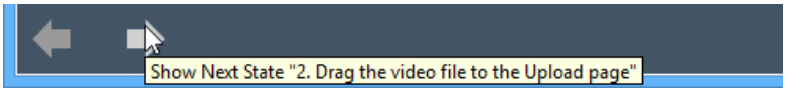
Play wireframes

5. This opens the wireframe player. The wireframe of the first step is shown in the player.



Playing a wireframe

You can move on to the next wireframe by pressing the **Right** key or clicking on the arrow button at the bottom left of the player. Similarly, you can press the **Left** key or click on the back button at the bottom left of the player to move to the previous wireframe. Note that when the flow comes to a condition (e.g. if-then-else) or looping (e.g. while) control, you have to decide the next step by selecting the step to take on the right hand side of the player.



Move to the next wireframe

Showing Annotations

To keep the wireframe clear and readable, annotations are hidden by default. You may click on the Show Annotations button t the bottom left of the player to have them visible. Let's show the annotations.

Show annotations

Ending the Show

You can end the show anytime by pressing the Esc key. When it arrives the final wireframe, you can also exit by clicking on the Exit button at the bottom left of the player.

Exit playback

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Writing effective use case
- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

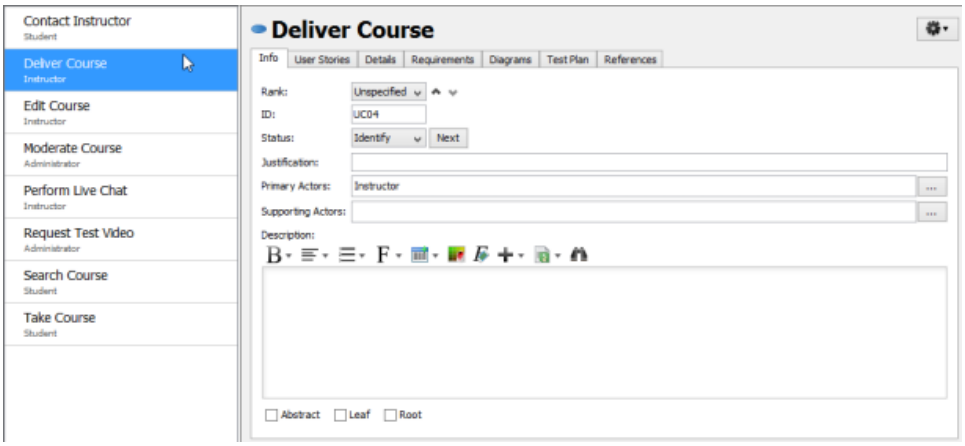
Producing requirement specification

Requirement specification is a document that contains all the details of use cases, requirements and wireframes. In Visual Paradigm you can produce such requirement specification from the **Use Case Details** of use cases.

Generating requirement specification from specific use case managed by UeXceler

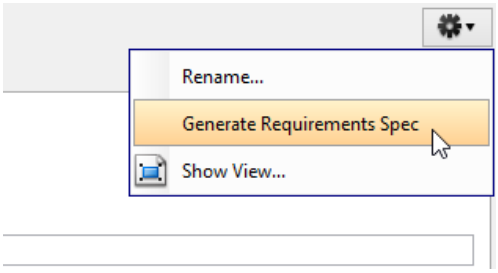
The following steps show you how to generate a requirement specification from a specific use case. Note that this method only works for use cases managed under UeXceler.

1. Select **Diagram > UeXceler** from the toolbar.
2. Open the **Use Case** tab.
3. On the left hand side, select the use case from which to produce a requirement specification.



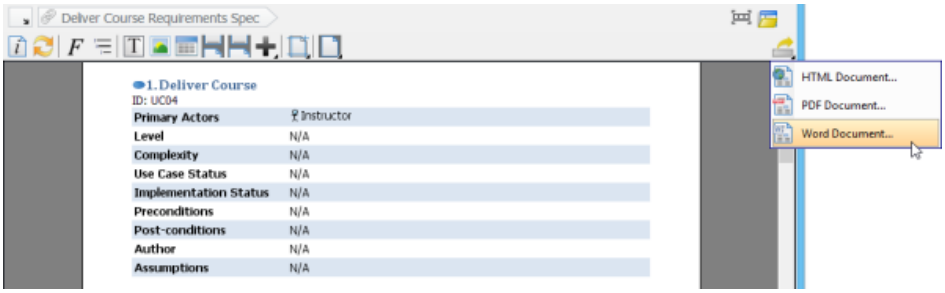
Selecting a use case

4. This opens the **Use Case Details**. On the right hand side of the screen, click on the gear button and then select **Generate Requirements Spec** from the drop down menu.



Generate Requirements Spec

5. This creates a requirement specification in the Doc. Composer. You can export the specification to Word/HTML/PDF by clicking on the **Export** button at the top right of the Doc. Composer. You may also edit the content of the report before exporting. To learn more about the use of Doc. Composer, click here.



Export requirement specification from Doc. Composer

Producing requirement specification from use cases on a use case diagram

The following steps show you how to generate requirement specification from use cases on a use case diagram.

1. Open the use case diagram.
2. Right-click on the background of diagram and select **Utilities > Generate Requirements Spec** from the popup menu.
3. This creates a requirement specification in the Doc. Composer. You can export the specification to Word/HTML/PDF by clicking on the **Export** button at the top right of the Doc. Composer. You may also edit the content of the report before exporting. To learn more about the use of Doc. Composer, click here.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Produce to Requirement Specification?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Elaborating use case

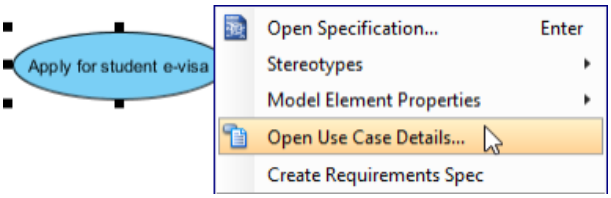
A use case can be elaborated by using sequence diagrams and activity diagrams to model its interactions and activity flows respectively.

Elaborating use case by sequence diagram

You can document the communication between user and system through the use of flow of events. You can then visualize the communication in sequence diagram through synchronizing flow of events to sequence diagram.

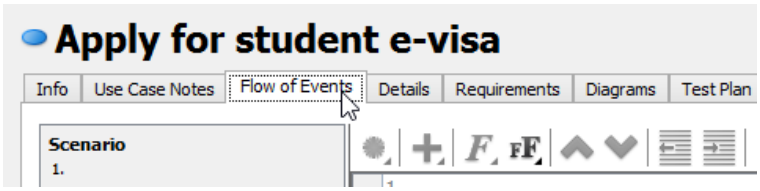
Synchronizing flow of events to sequence diagram

1. Right click on the target use case and select **Open Use Case Details...** from the pop-up menu.



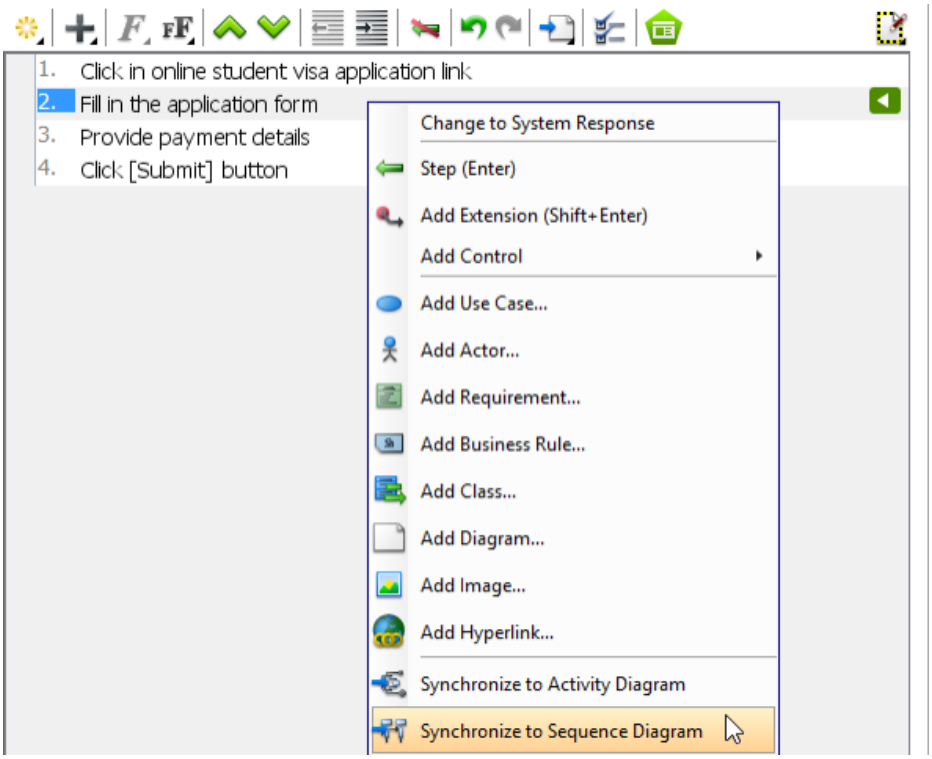
Open use case details

2. When the details dialog box pops out, select the **Flow of Events** tab.



Open flow of events editor

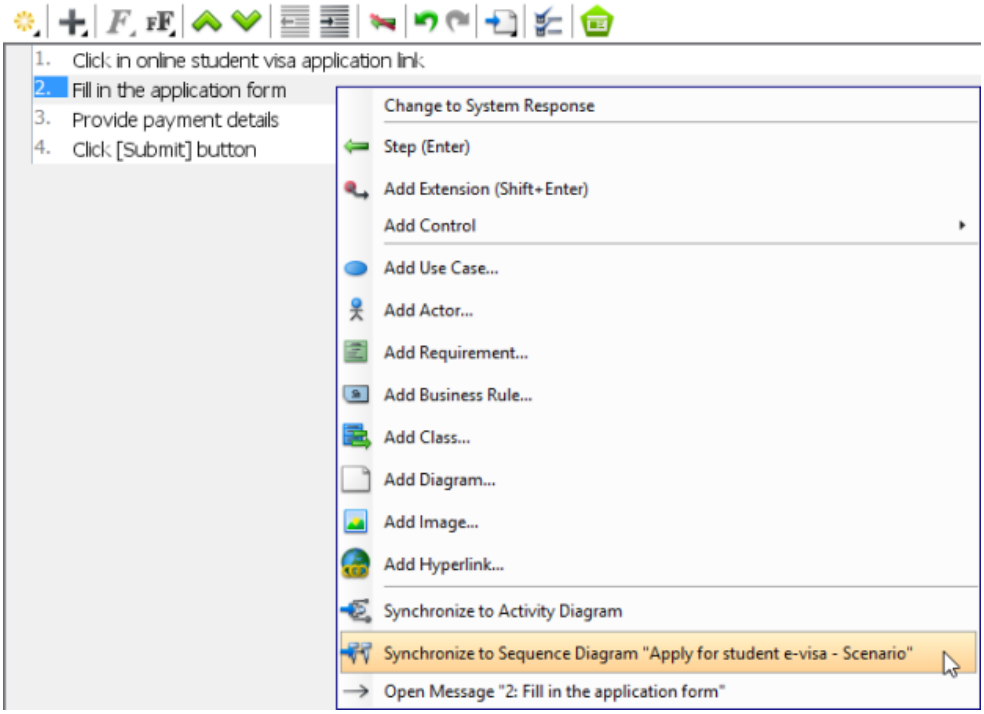
3. Edit the steps in the flow of events editor. When finish editing, right click on the editor and select **Synchronize to Sequence Diagram** from the pop-up menu.



Generate sequence diagram from flow of events

Updating sequence diagram from flow of events

When you have made some changes on flow of events, you can update the changes to sequence diagram accordingly. Right click on the flow of events editor and select synchronize to the name of sequence diagram from the pop-up menu.



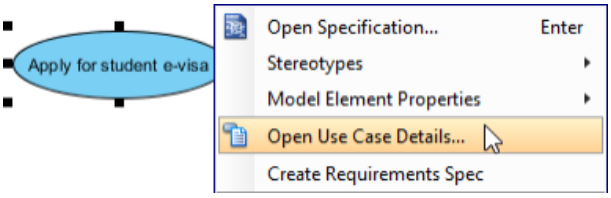
Update changes from flow of events to sequence diagram

Elaborating use case by activity diagram

You can document the events of a use case through the use of flow of events. You can then visualize the events in activity diagram through synchronizing flow of events to activity diagram.

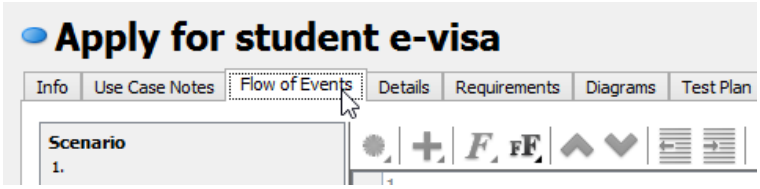
Synchronizing flow of events to activity diagram

1. Right click on the target use case and select **Open Use Case Details...** from the pop-up menu.



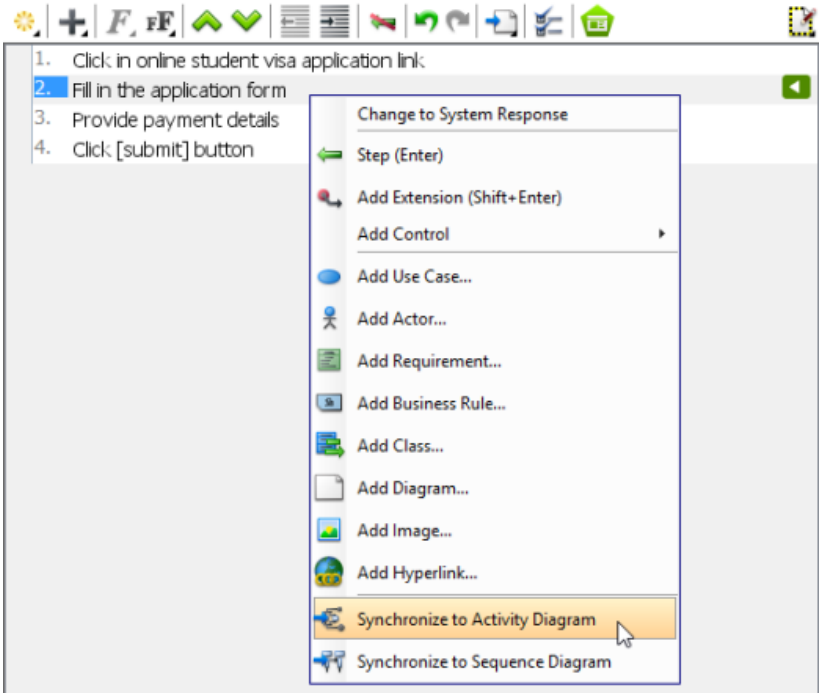
Open use case details

2. When the details dialog box pops out, select the **Flow of Events** tab.



Open flow of events editor

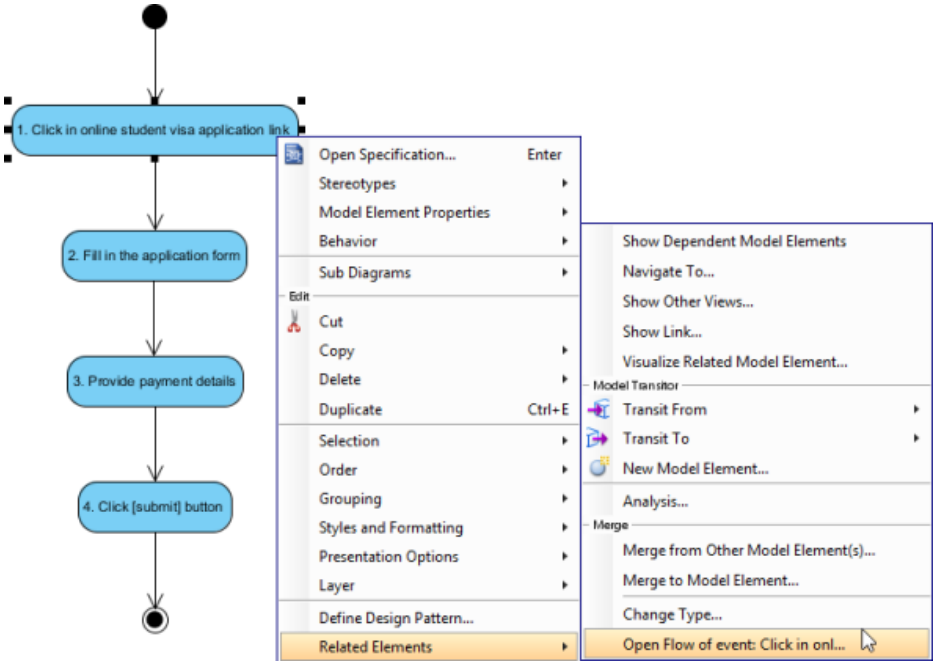
3. Edit the steps in the flow of events editor. When finish editing, right click on the editor and select **Synchronize to Activity Diagram** from the pop-up menu.



Generate activity diagram from flow of events

Navigating to flow of events

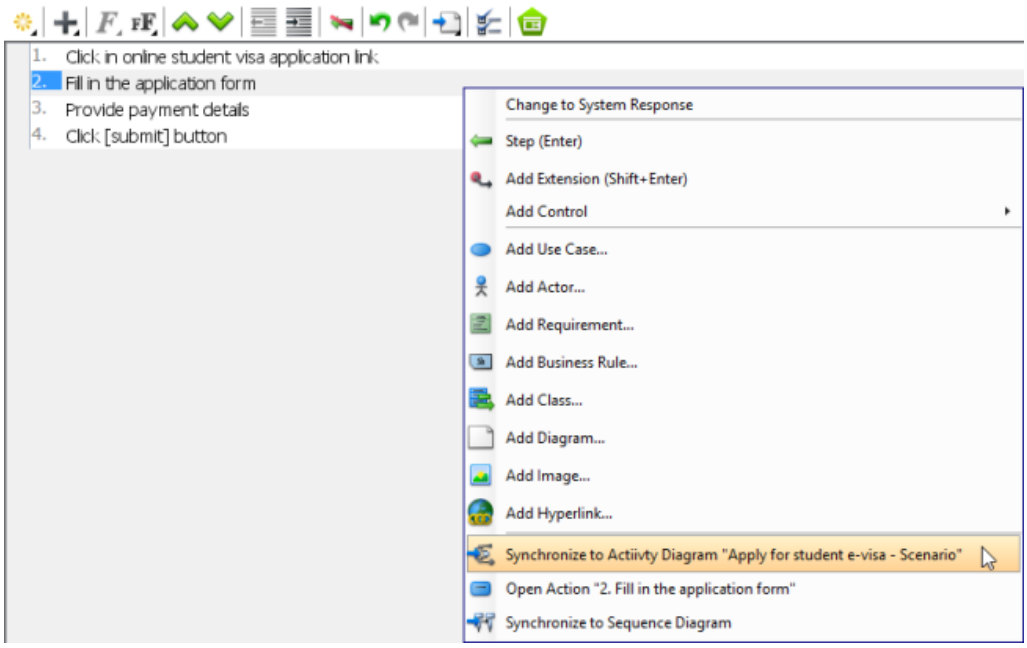
If you want to return to the flow of events, right click on a target action in activity diagram and select **Related Elements > Open Flow of event** from the popup menu.



Open flow of events from activity diagram

Updating activity diagram from flow of events

When you have made some changes on flow of events, you can update the changes to activity diagram accordingly. Right click on the flow of events editor and select synchronize to the name of activity diagram from the pop-up menu.



Update changes from flow of events to activity diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Generate sequence diagram from use case flow of events
- YouTube Video - How to Manage Use Case Scenario with Flow of Events?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Requirement List

Record and document software requirements in Requirement List

Learn how to manage requirements in Requirement List

Record and document software requirements in Requirement List

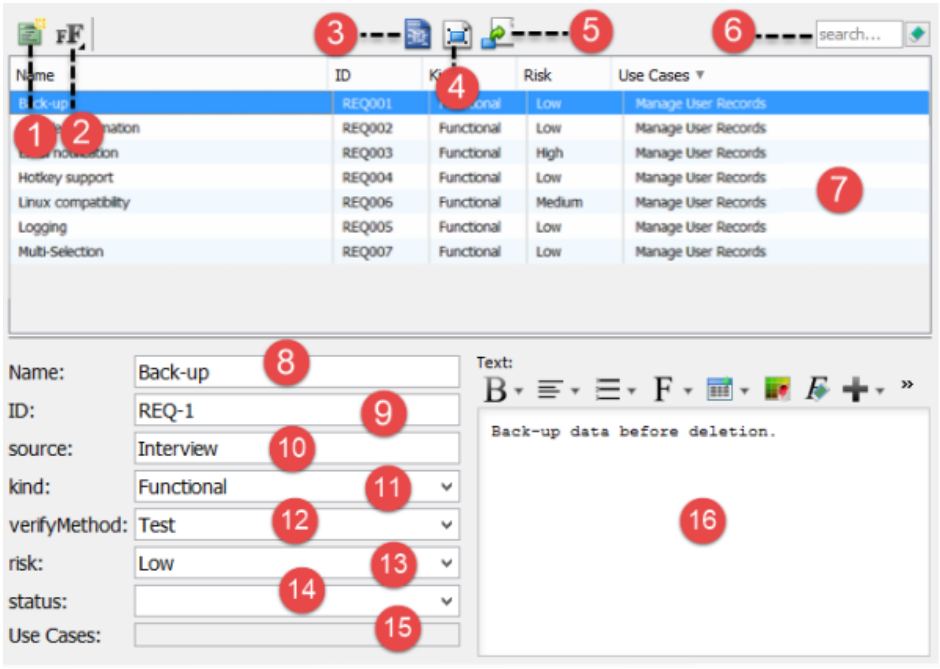
Every use case can be achieved by implementing a set of relevant requirements. Requirement states what the system needs to deliver. We will identify a set of requirements under use cases. While use case focuses on what user wants to do with our system, requirement focuses on what the system needs to deliver to fulfill the use cases.

The **Requirement List** is a place where you can store and manage requirements. You can also gain an overview of requirements involved in the entire system.

Opening the Requirement List

To open Requirement List, select **Modeling > Requirement List** from the toolbar.

Overview of Requirement List



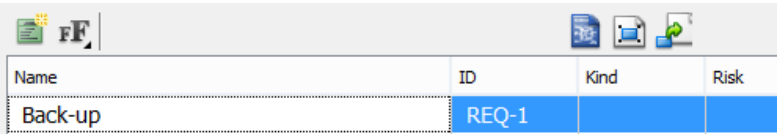
Requirement List

No	Name	Description
1	New Requirement	Click to create a requirement.
2	Font Size	Click to adjust the font size of text in **Requirement List**.
3	Open Specification...	Select a requirement in **Requirement List** and click this button to open its specification.
4	Show View...	Select a requirement in **Requirement List** and click this button to list the diagrams that contains the view of the selected requirement.
5	Visualize	Select a requirement in **Requirement List** and click this button to show it in a new or existing diagram.
6	Search	Find requirement(s) by entering search criteria.
7	List of requirements	Requirements are listed here.
8	Requirement name	Name of selected requirement.
9	Requirement ID	ID of selected requirement. ID are automatically generated when you create requirement. You may customize the pattern of ID in the **Project Options** window (**Windows > Project Options > Diagramming > Model Generation**).
10	Source	The way how the requirement was created.
11	Kind	The type of requirement.
12	Verify Method	The way how the requirement can be verified.
13	Risk	The level of risk in supporting the requirement.
14	Status	The current status of requirement.
15	Use Cases	Use cases can be achieved by implementing requirements. If the selected requirement was created from a use case, or added as a requirement of a use case, you can see the use cases here.
16	Requirement description editor	Description of selected requirement. The tools above the editor enables you to enter description in rich text format.

Creating requirement in Requirement List

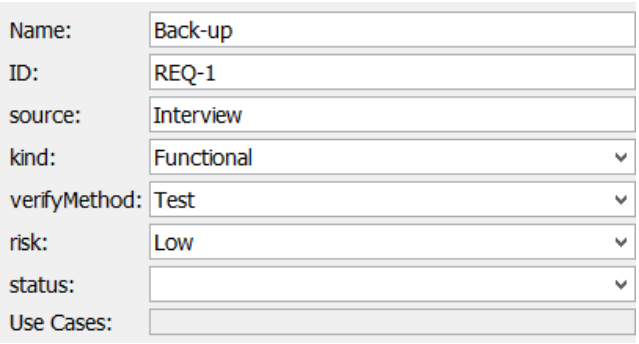
To create a requirement in **Requirement List**:

1. Click on **New Requirement** above the **Requirement List**.
2. Enter the name of requirement.



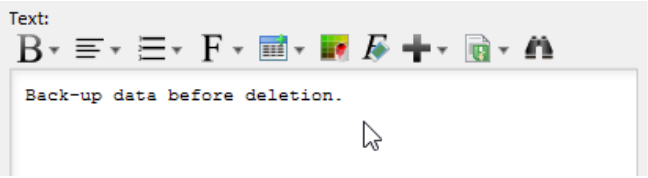
Creating requirement in Requirement List

3. Press **Enter** to confirm editing.
4. You can optionally edit the properties of the requirement.



Edit requirement properties

5. You can optionally edit the properties of the requirement.



Edit requirement description

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - How to Manage Requirements with Requirement List?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Use Case Statements

Writing use case statements

Learn how to identify use cases and actors by writing use case statements.

Creating use case from use case statement

Learn how to identify use cases and actors from use case statements.

Writing use case statements

Use case statements is a technique that helps to identify actors and use cases. The user voice form of use case expression begins with <role>, which provides a user-oriented approach to identifying the business objective (i.e. use case) and business value.

The form of a use case statement is as follows:

As a <role>, I want to <business objective> so that <business value>

where:

- <role> represents the person, system, subsystem or any entity else who will interact with the system to be implemented to achieve a goal. He/she/it will receive values by interacting with the system. By creating a use case from a use case statement, an actor will be created from <role>.
- <business objective> represents the business goal to be achieved throught the interaction with the system. By creating a use case from a use case statement, a use case will be created from <business objective>.
- <business value> represents the value behind the interaction with system.

Benefits of finding use cases with use case statements

Use case statements provides a user point of view in finding use cases. It keeps the development team focused on business values behind use cases instead of spending time on features that bring no value to users.

Writing a use case statement

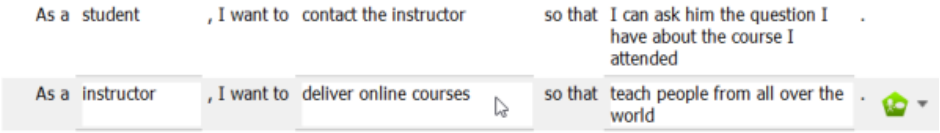
1. Select **Diagram > UeXceler** from the toolbar.
2. In the **Use Case Statement** tab, fill in the role, business objective and business value of the last statement to complete it.

Entering use case statement

Inserting a use case statement

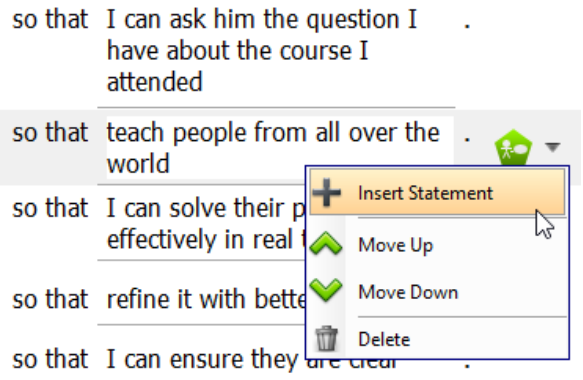
To insert a use case statement between two statements:

1. Hover your mouse pointer over the second statement.



Mouse pointer over a use case statement

2. Click on the down arrow next to the green button near the end of the statement.
3. Select **Insert Statement** from the popup menu.

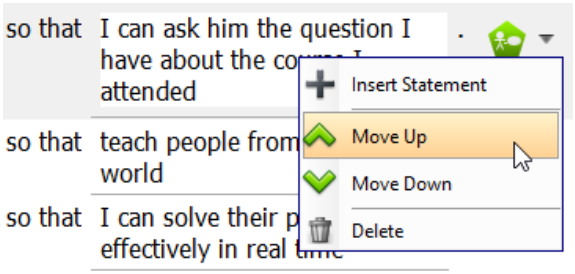


Insert Statement

Repositing a use case statement

To reposition a use case statement:

1. Hover your mouse pointer over the use case statement.
2. Click on the down arrow next to the green button near the end of the statement.
3. Select **Move Up** or **Move Down** from the popup menu.



Moving a use case statement

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

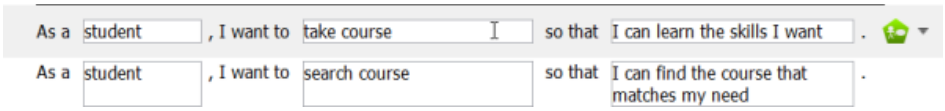
- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating use case from use case statement

You can identify and create use cases from use case statement. When you create a use case from a use case statement, the <role> will become an actor, while the <business objective> will become the use case. Both the actor and use case created will be visualized in the **Use Case Diagram** tab of UeXceler.

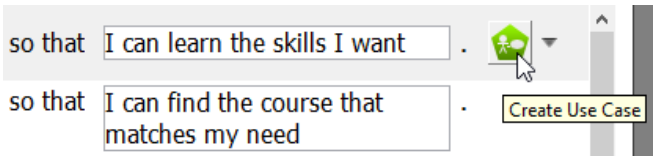
To create a use case from use case statement:

1. Select **Diagram > UeXceler** from the toolbar.
2. In the **Use Case Statement** tab, hover your mouse pointer over the desired use case statement.



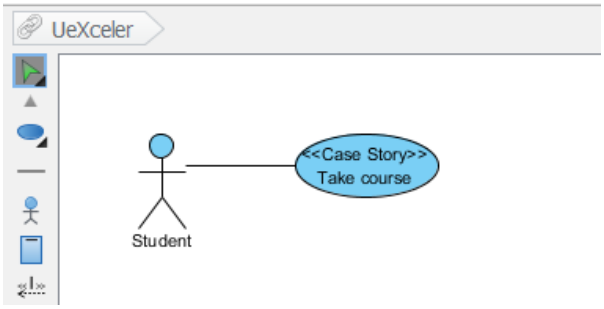
Mouse pointer over a use case statement

3. Click on the **Create Use Case** button near the end of the statement.



Create use case

This creates a use case with <business objective> as name and the use case will be visualized in the **Use Case Diagram** tab of UeXceler. An actor with <role> as name will also be created if it does not exist.



Use case created

4. Rename the use case to make it follow your naming convention.

Use case renamed

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

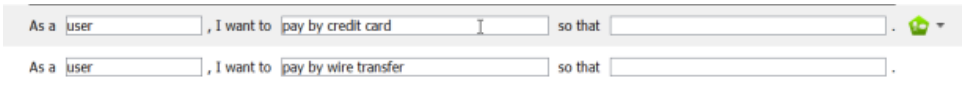
- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating user story from use case statement

You can identify and create user story from use case statement. When you create a user story from a use case statement, the <role> and <business objective> will become the description of user story.

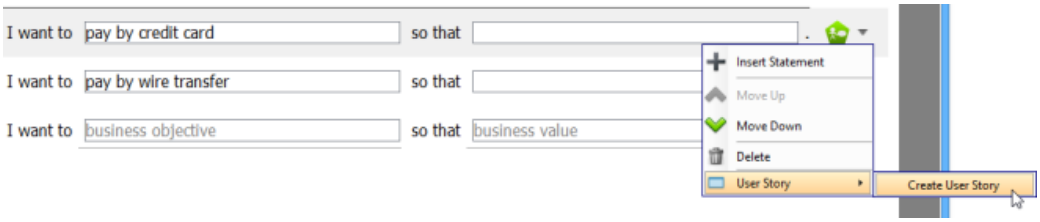
To create a user story from use case statement:

1. Select **Diagram > UeXceler** from the toolbar.
2. In the **Use Case Statement** tab, hover your mouse pointer over the desired use case statement.



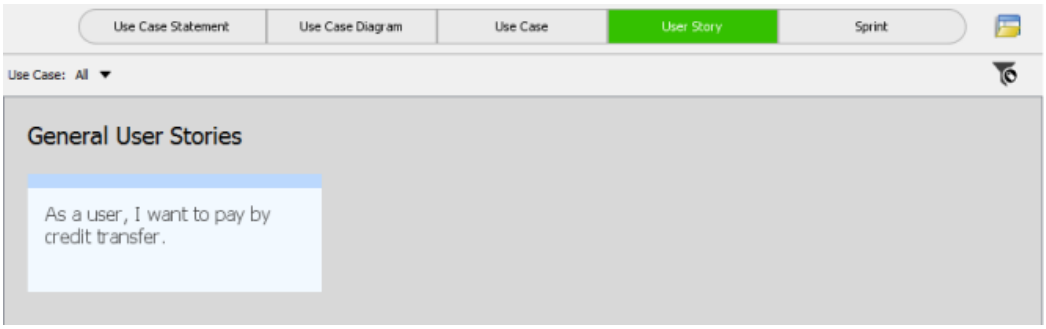
Mouse pointer over a use case statement

3. Click on the down arrow next to the green **Create Use Case** button, near the end of the statement.
4. Select **User Story > Create User Story** from the popup menu.



Create user story

This creates a user story with <role> and <business objective> as description. You can find the created user story in the **User Story** tab.



User story created

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

User Stories

Writing general user stories

Learn how to write user stories that do not belong to any use case.

Writing user story in use case basis

Learn how to write user stories based on the use cases identified with stakeholder.

Prioritize User Stories

Learn how to order user stories based on their business value.

Viewing user stories in UeXceler

Learn how to view user stories of particular use case.

Writing general user stories

In modern software development, requirements capturing is not something you can finish at the outset of a project. Instead, new requirements constantly emerge while previously suggested requirements can change throughout the project. To meet the ever changing expectations, we need a way that can facilitate the continuous capturing and management of user's requirements. User stories is one of the approaches that has been widely adopted by many agile projects.

Written by users (or customer team), user stories describe functionality that are needed by and valuable to users. As an integral part of many agile development processes, user stories offers a quick way in recording user's requirements without the need to write any detailed requirement documents or to have any prior consideration of system behaviors.

Visual Paradigm supports agile development teams in writing user stories for requirements capturing and project management. The user stories editor allows you to represent user stories as card, as well as to detail the conversation and confirmation items – The Three C.

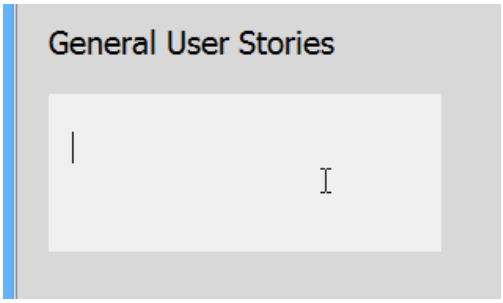
Writing general user story

You can create user stories as general stories, or create user stories under use cases. To create a general user story:

1. Select **Diagram > UeXceler** from the toolbar.
2. Open the **User Story** tab.

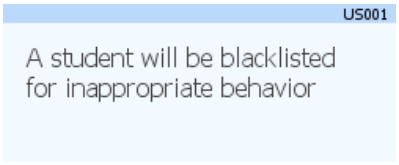
Open the User Story tab

3. Under the **General User Stories** section, double click on the block labeled **Add a feathre that can be finished in 10 days**.



To enter user story

4. Write a short description of user requirement and press **Enter** to confirm editing. Note that user story is supposed to be short and unique, so do not try to write a paragraph here.



User story created

To create another user story, double click on the right hand side of the last story created.

Entering a description for user story

The name of a user story, which is the text that appears on a user story card uses to be short and brief. If you want to describe a user story in more detail, you can enter its description by performing the steps below:

1. Select the user story.
2. Double click on the text **Write description**.
3. Enter the description.
4. Press **Enter** to confirm editing.

Modifying the ID of a user story

An ID will be automatically generated to each of the user story upon creation. If you want to modify the ID, perform the steps below:

1. Double click on a user story.
2. On the top right corner, double click on the pre-assigned ID.
3. Enter your own ID.
4. Press **Enter** to confirm.

NOTE: You can change the format of auto-generated ID by changing it in the **Project Options** window. Select **Windows > Project Options** from the toolbar. In the **Project Options** window, select **Diagramming > Model Generation**. You can then edit the prefix, number of digits and suffix of user story ID.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Writing user story in use case basis

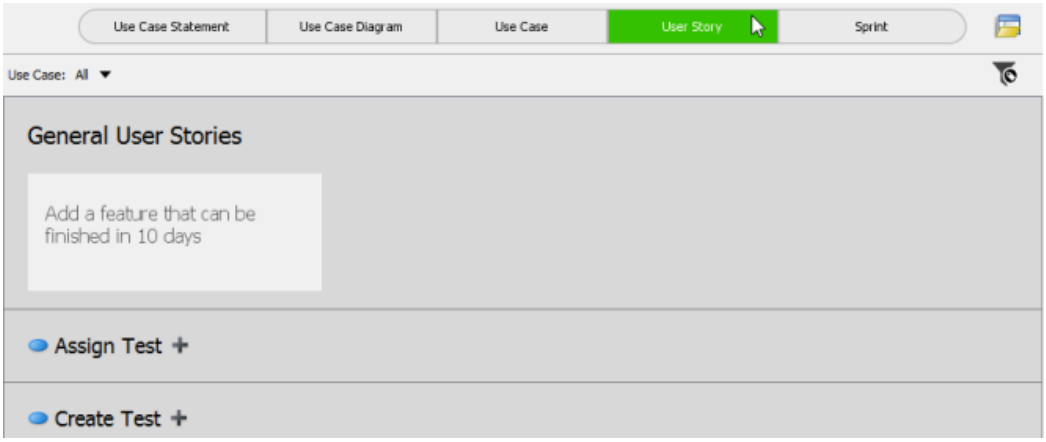
While Use Case is the business goal of an IT system to be developed, User story is the requirement captured by the analyst and customer. No doubt that, all captured requirements aim to fulfill the business goal of the IT system. In other words, all User Stories are based on Use Cases. Therefore, the best way to produce User Stories is to write in Use Case basis.

Benefit of writing User Stories based on Use cases

When you organize User Stories within Use Cases, this can ensure that your User Stories must stay within the system scope. In other words, after the captured requirements are developed, they must match with the goals of the system. As any User Stories which do not fit the business objectives have been eliminated before the development process, no time would be wasted on the development of unrelated features.

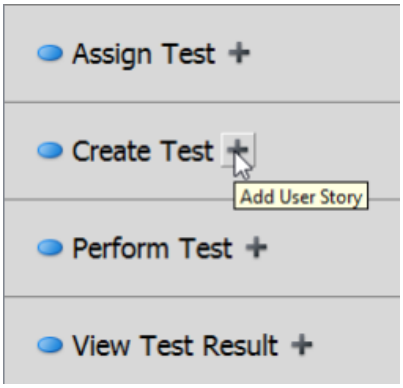
Writing User Stories around Use Cases

1. Open the **User Story** tab in UeXceler.



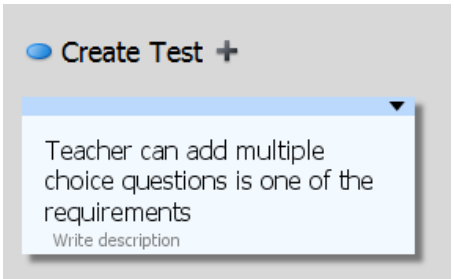
Open the User Story tab

2. Click the **Plus** button next to the Use Case that you want to add the User Story under.



Add user story under use case

3. A story card appears. Enter the User Story. Press **Enter** to confirm editing.



Confirm creating user story

4. Repeat step 2 and 3 to create other User Stories according to their Use Cases.

Merging of use cases

When you find two use cases similar in nature, you may want to merge them into one single use case. By merging use cases, the containing user stories will be brought to the target use case. To merge a use case to another use case, right click on the 'from' use case and select **Related Elements > Merge to Model Element...** from the popup menu. In the **Merge to Model Element** window, select the target use case and click **Merge**.

Entering a description for user story

The name of a user story, which is the text that appears on a user story card uses to be short and brief. If you want to describe a user story in more detail, you can enter its description by performing the steps below:

1. Select the user story.
2. Double click on the text **Write description**.
3. Enter the description.
4. Press **Enter** to confirm editing.

Modifying the ID of a user story

An ID will be automatically generated to each of the user story upon creation. If you want to modify the ID, perform the steps below:

1. Double click on a user story.
2. On the top right corner, double click on the pre-assigned ID.
3. Enter your own ID.
4. Press **Enter** to confirm.

NOTE: You can change the format of auto-generated ID by changing it in the **Project Options** window. Select **Windows > Project Options** from the toolbar. In the **Project Options** window, select **Diagramming > Model Generation**. You can then edit the prefix, number of digits and suffix of user story ID.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Categorizing user stories by tag

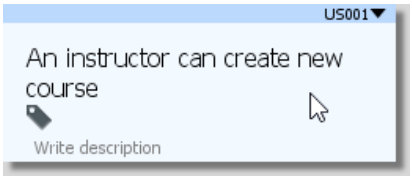
Besides writing description and documenting the confirmation items for user stories, you may want to categorize them in further. For example, you may want to indicate which user stories are urgent and require instant resolution. You may also want to indicate user stories that are planned to be available in the coming release. In Visual Paradigm, categorization of user stories can be made by tag.

Creating a tag

You can create tag directly from a user story. By doing so, you can immediately tag the user story with the tag just created. Besides, you can also create a tag globally.

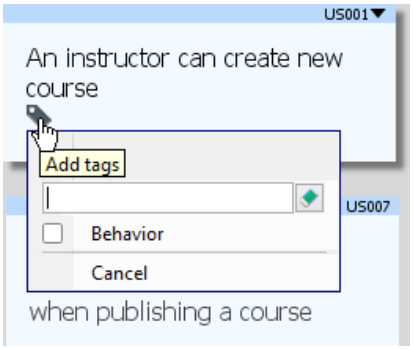
Creating a tag from a user story

1. In the **User Story** tab of UeXceler, click to select the desired user story.



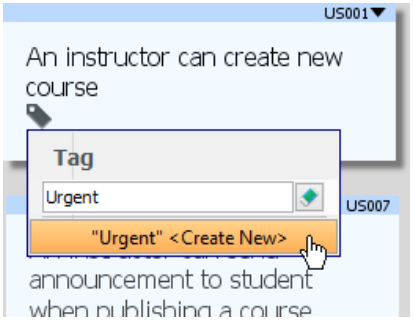
User story selected

2. Click on the **Add tags** button.



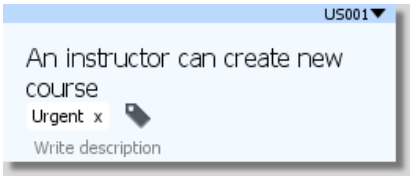
Try to add tags

3. Enter the name of the new tag in the text field, and then select <Create New>.



Create a tag

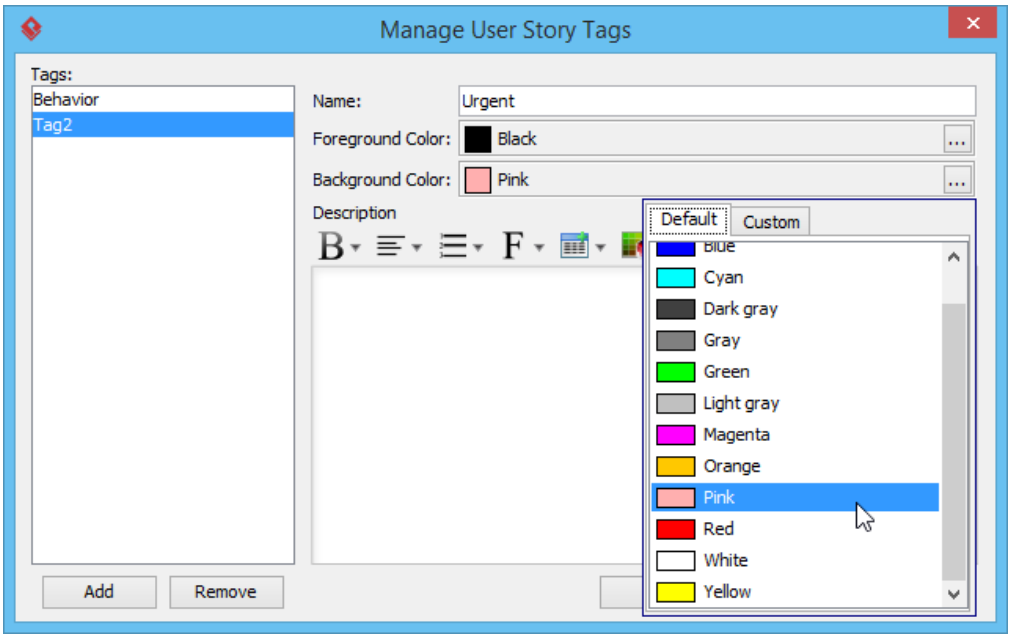
The tag is created and is added to the user story.



Tag is created

Creating a tag globally

1. Select **Windows > Configuration > Manage User Story Tags...** in the toolbar.
2. In the **Manage User Story Tags** window, click **Add** to create a tag.
3. Enter the name of the tag and then specify its foreground and background color. The color setting will affect the appearance of tag when show in UeXceler.

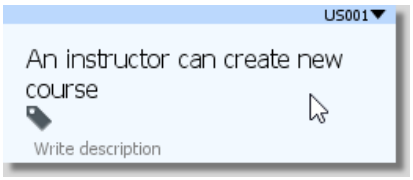


Adding a tag in Manage Tags of User Story window

4. Click **OK** to confirm.

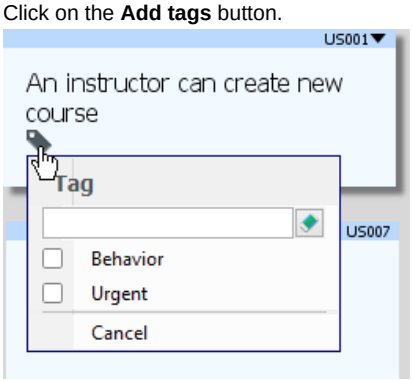
Adding a tag to user story

1. In the **User Story** tab of UeXceler, click to select the desired user story.



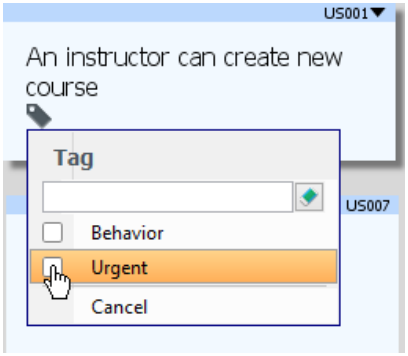
User story selected

2. Click on the **Add tags** button.



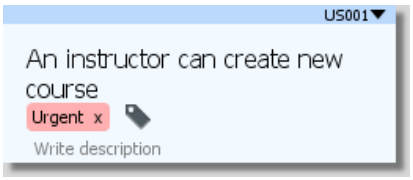
Try to add tags

3. Select the tag(s) from the popup menu.



Select tags to add

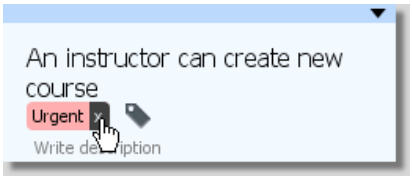
4. Click **Apply**.



Tag added

Removing a tag from user story

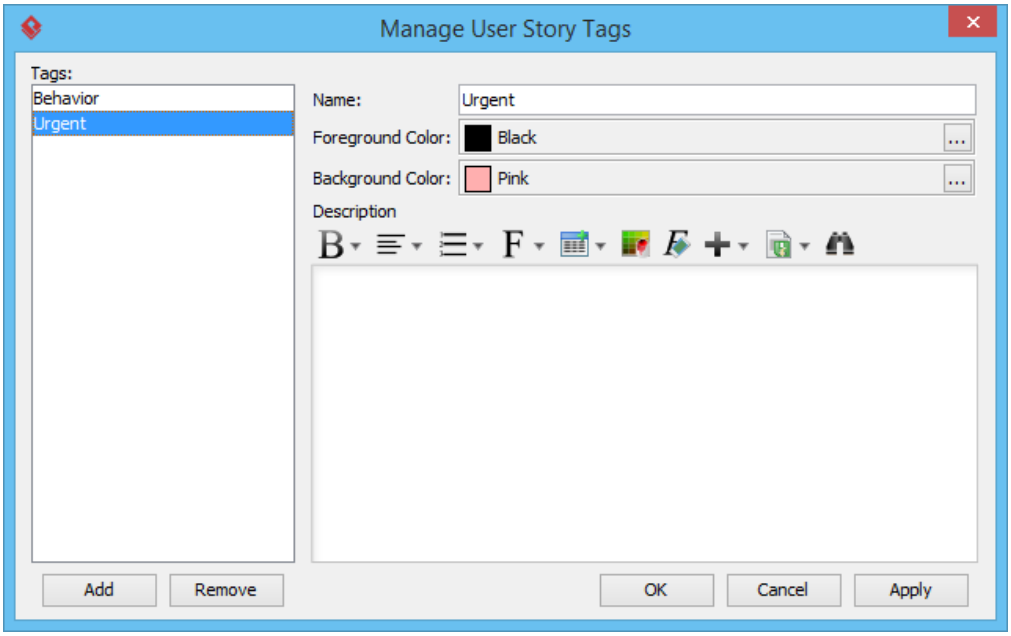
To remove a tag from user story, move your mouse pointer to the target user story and then click on the tiny cross icon next to the tag you want to remove.



To remove a tag from a user story

Editing or Deleting a tag

Both the editing and deletion of tag can be done in the **Manage User Story Tags** window. To open the Manage Tags of User Story window, select **Windows > Configuration > Manage User Story Tags** in the toolbar. Then, select the tag to delete/edit. To delete a tag, click **Remove**. To edit a tag, edit its properties like name, color and description and then click **OK** to confirm.



The Manage Tags of User Story window

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Prioritize User Stories

Within General User Stories or one Use Case, there might be countless number of User Stories. However, among all these User Stories, it is possible that some are more important (i.e. in terms of urgency or business value) while some are not during the development process. In order to classify them base on their importance, Visual Paradigm allows you to prioritize them with simple method.
To prioritize Use Cases:

1. Drag the User Story.

Drag user story

2. Drop it to a desired location.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Viewing user stories in UeXceler

In a software development process, there might be uncountable number of Use Cases. No doubt that, you might not need all Use Cases at all time and showing all of them at once will make you very hard to read and look for particular one. In order to help you to only focus on those Use Cases and their corresponding User Stories that you are interested in, Visual Paradigm allows you to filter off non-interested Use Cases.

To filter off non-interested Use Cases:

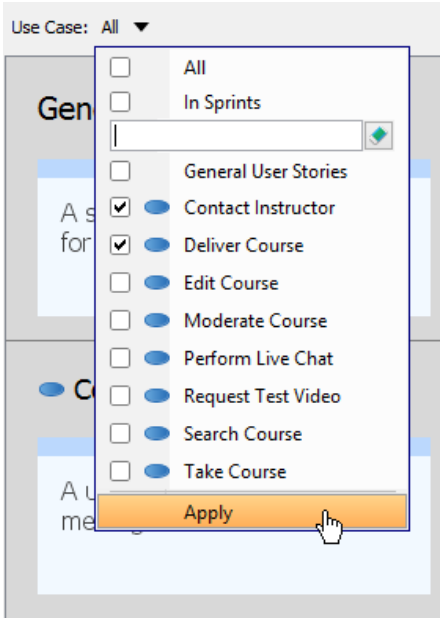
1. Select **Diagram > UeXceler** from the toolbar.
2. Open the **User Story** tab in UeXceler.

User Story tab in UeXceler

3. Click the **All** button near the top left corner of UeXceler.

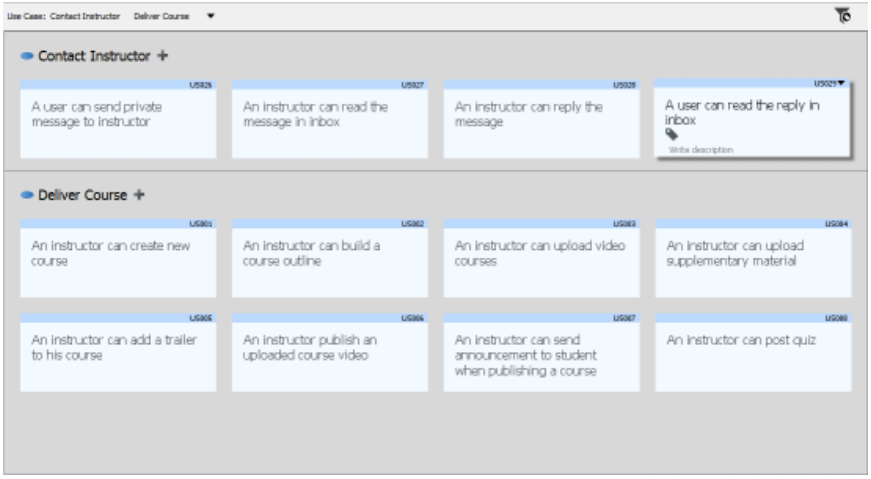
*Click **All** button*

4. Check the Use Case(s) that you are interested in. Select **Apply** from the drop down menu.



Apply selection

5. As a result, only the checked Use Cases and their corresponding User Stories are shown.

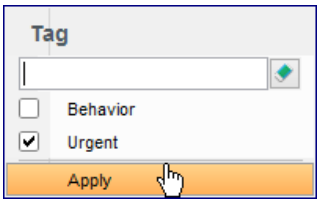


Checked Use Cases are shown

In some occasions, you might want to fiter off User Stories by Tag(s) instead of Use Cases, Visual Paradigm also supports this for your convinence.

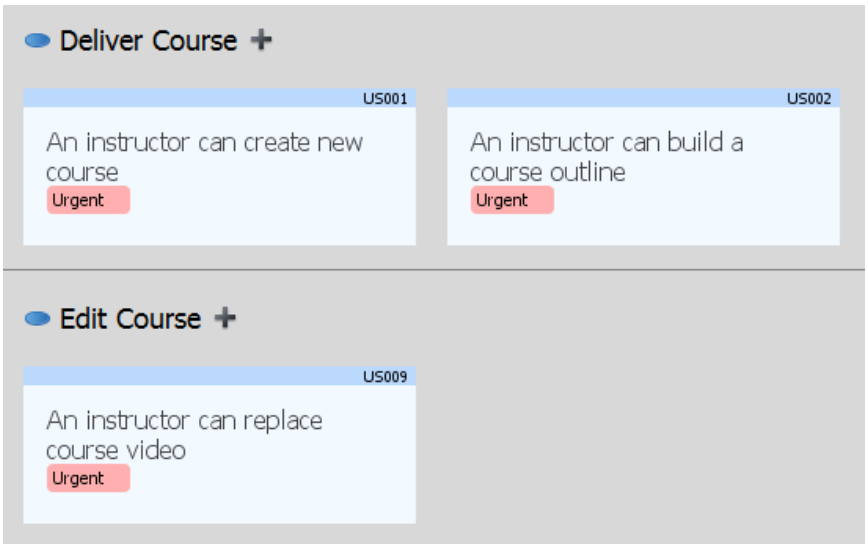
To filter off User Stories base on Tag(s):

1. Click on the **Filter by Tags** button near the top right corner of UeXceler.
2. Check the Tag(s) that you are interested in. Select **Apply** from the drop down menu.



Apply Selection

3. As a result, only those User Stoies with checked Tag(s) are shown.



User Stories with checked Tag are shown

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

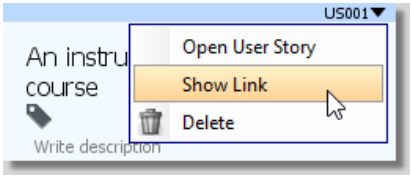
Sharing user stories by sharing links

Sometimes, you may want to ask your teammates to look into a user story for accessing its details like the conversation notes, confirmation items, scenarios, etc. Instead of asking your teammates to check user story "blah... blah... blah...", you can simply share the link of that user story with them.

A link is a unique address of an element in a project. It is a general element accessing mechanism available in Visual Paradigm. You can share a link with teammate, so to let him/her to locate an element quickly and accurately. By using the link feature to locate user story, you can provide teammates with accurate link of a specific user story. Teammates can open the user story instantly without the need of searching.

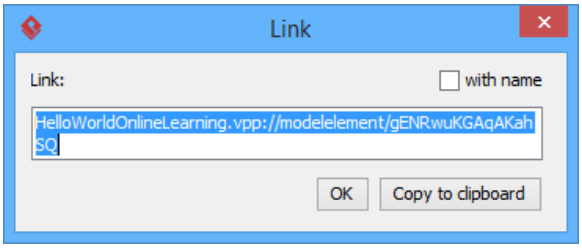
Showing the link of a user story

1. In the **User Story** tab of UeXceler, click on the desired user story.
2. Click on the down arrow at the top right of the user story.
3. Select **Show Link** from the popup menu.



To show the link of a user story

4. You can now copy the link by clicking **Copy to clipboard** or by pressing **Ctrl-C**. If you want to include the name of user story, check **with name**. The URL will work perfectly both with and without name.



Copying the link of a user story

Open a user story with link

To open a user story with a link:

1. Select **Diagram > Link** from the toolbar.
2. Paste the link into the Link field and press **Enter** to visit it. Note that the link has to be a non-modified string copied with the Show Link feature. Any modification made to the link may result in a broken link.

Opening a user story with link

This will open the user story in target project.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

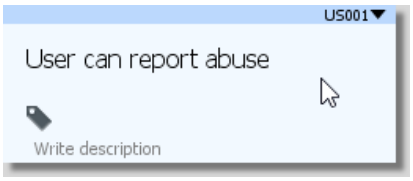
Adding storyboard to user stories

Storyboard is a useful technique in system analysis and design in that it represents the screens and screen flow under different scenarios and situations. Say, if you are designing an ATM system, you can design a storyboard to show the screens that involve in a cash withdraw process, starting from inserting card until cash dispensing and card ejection.

In Visual Paradigm, you can design storyboard for each of the user stories. This helps in explaining system design ideas and aids the confirmation of requirements when in requirements gathering phase.

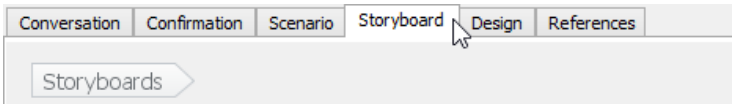
To create a wireframe storyboard:

1. In the **User Story** tab of UeXceler, double click on the desired user story to open it.



Open a user story

2. Open the **Stotyboard** tab.

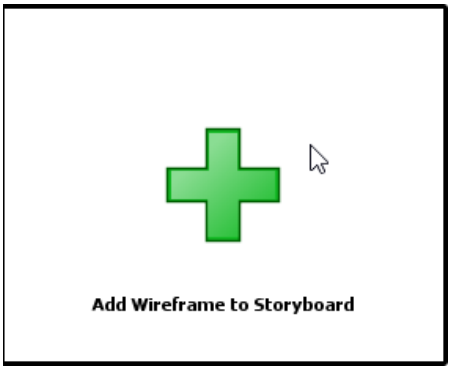


Open the Storyboard tab

3. Click **Add New Storyboard**.

To add a storyboard

4. Input the storyboard name. You can treat a storyboard as a story. So if you want to represent the screen flow of the 'Report abuse' function, you can name it *Report Abuse*.
5. Double click on the added storyboard to open it.
6. Click **Add Wireframe to Storyboard**.



To add a wireframe to storyboard

7. If your project has no wireframe in it, you will be asked to create one. If the project contains a wireframe, you can now add an existing wireframe to the storyboard or just create a new wireframe. More about wireframe selection will be covered in the following sections.

The New Wireframe window

8. Create/select a wireframe.
9. Go back to the storyboard by clicking on the **Done** button on top of the wireframe.

Go back to the storyboard

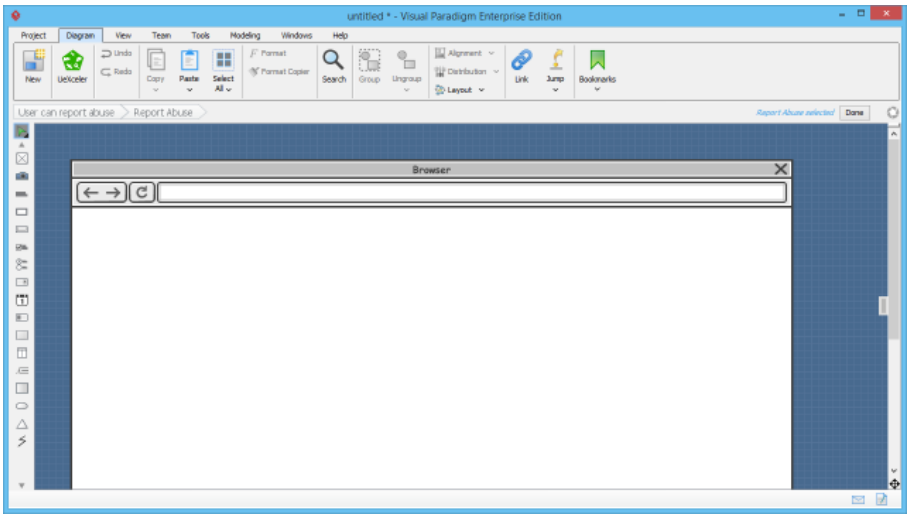
Creating a wireframe in a storyboard

To create a new wireframe in a storyboard:

1. Open **Storyboard** and select the desired storyboard.
2. Click **Add Wireframe to Storyboard...**.
3. If your project has no wireframe in it, you will be asked to create one. Select the suitable type of device/platform for your application/system. If your system will run on multiple devices/platforms, please consider creating multiple storyboards.

Select a type of wireframe to create

4. Click **New %TYPE% Wireframe** where **%TYPE% is** the type of device/platform you selected.
5. A blank, new wireframe appear and you can now begin editing.

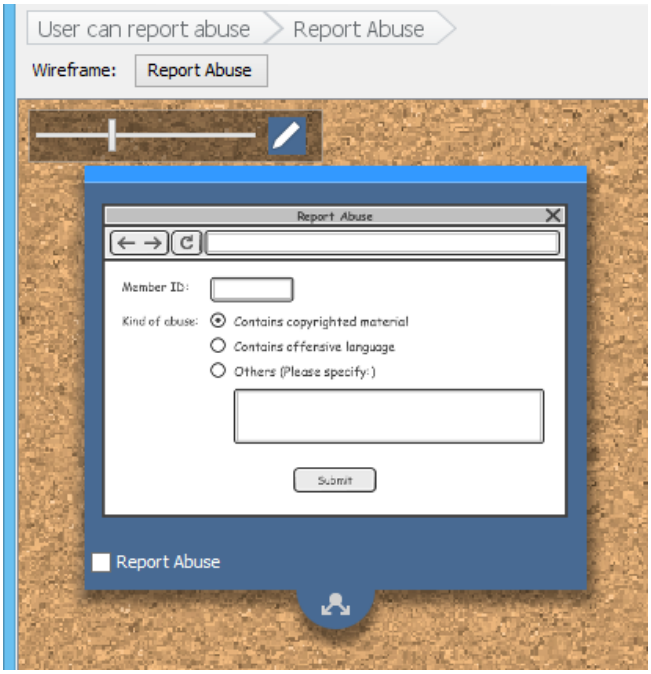


New wireframe created

6. When you finish editing, you can go back to the storyboard by clicking on the **Done** button on top of the wireframe.

Go back to the storyboard

The above are the steps that involve in creating a wireframe from a storyboard when there is no wireframe in your project. Once you have created a wireframe, you will see something different after step 3. Here is what you will see:



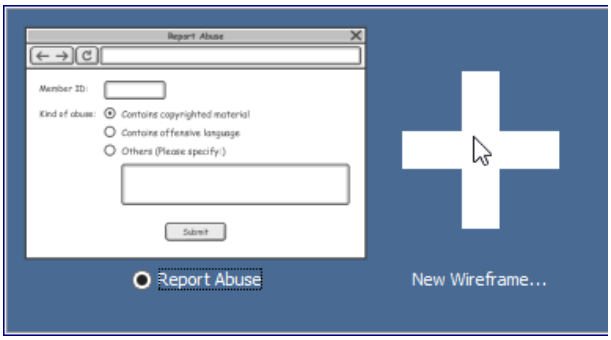
State overview

If you want to create an entirely new wirefram:

1. Click on the button next to **Wireframe:**.

Choose another wireframe

2. Click **New Wireframe...**.

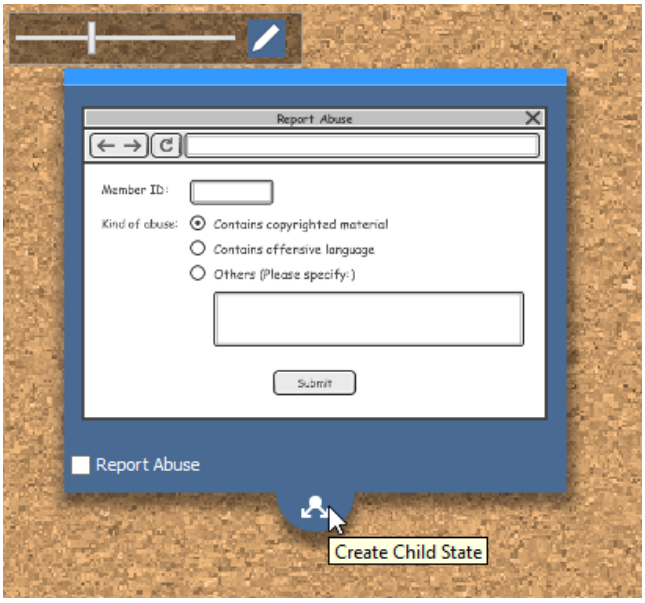


Create a new wireframe

3. The remaining steps are same as those mentioned above, starting from step 3.

If you want to re-use an existing wireframe but make a bit of change, you should create a child state instead:

1. Click **Create Child State** below the thumbnail of wireframe to create a child state under it.



Create a child state

2. Once clicked, a new wireframe state will be created. You can start editing it.

Adding an existing wireframe to a storyboard

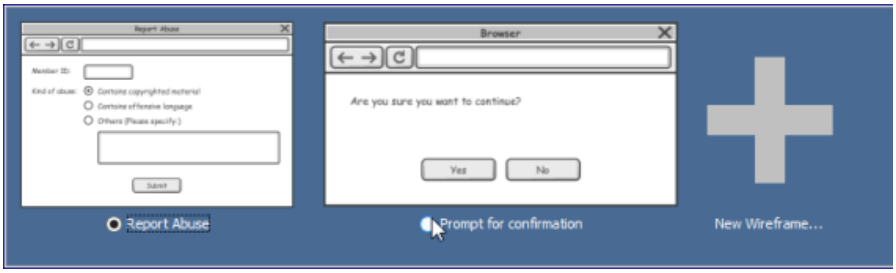
Sometimes, you may want to re-use a wireframe created earlier. For example, to reuse a wireframe about account login in storyboards that require user to login to do something.

To add an existing wireframe into a storyboard:

1. Open **Storyboard** and select the desired storyboard.
2. Click **Add Wireframe to Storyboard**.
3. Click on the button next to **Wireframe:**.

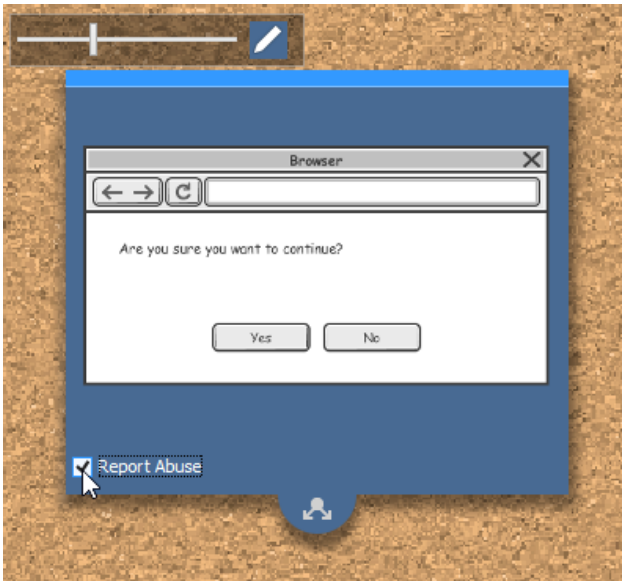
To select a wireframe

4. Choose the wireframe for your storyboard.



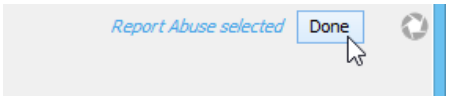
Choosing a wireframe

5. This shows the available states of the wireframe. Select the right one by checking the checkbox at bottom left corner. Make sure you do selected a state in this step. Without doing so, the wireframe won't be associated with the storyboard.



Selecting a wireframe state

6. Go back to the storyboard by clicking on the **Done** button on top of the state selection page.

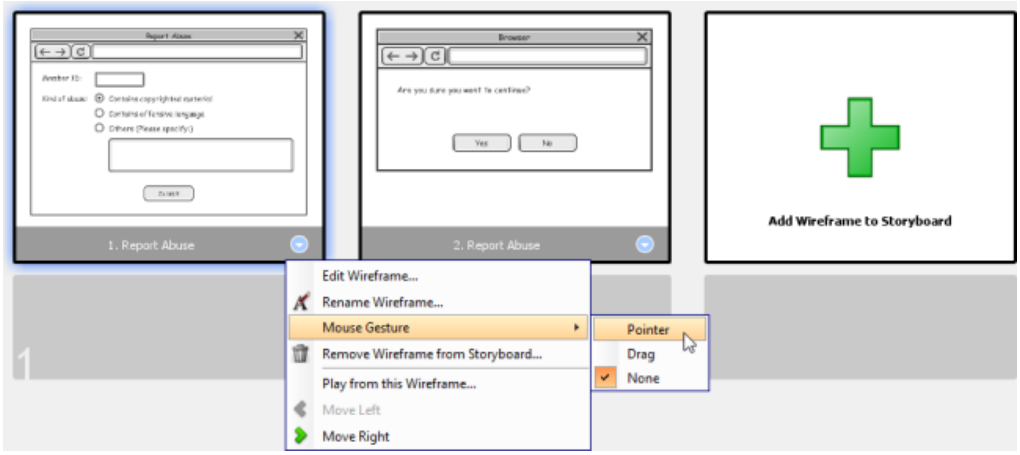


Go back to the storyboard

Using pointer / finger gesture

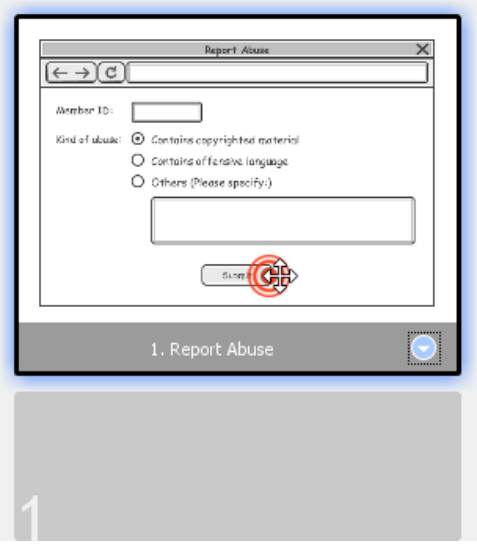
You can indicate in wireframe the position of mouse pointer, or the finger gesture required to execute an action. To do this:

1. Open **Storyboard** and select the desired storyboard.
2. There is a button at the bottom right of a wireframe. Click on it and select **Mouse Gesture > Pointer/Drag/Finger Gesture** from the popup menu. Note that Pointer and Drag are available for Desktop and Web wireframe, while Finger Gesture is available for Android phone, Android tablet, iPad and iPhone wireframes.



Add a pointer to wireframe

3. Drag on the pointer/drag/finger gesture symbol to reposition it.



Repositioning a pointer

Renaming a storyboard

If you want to change the name of a wireframe storyboard, rename it by performing these steps:

1. In the **User Stories** tab of UeXceler, double click on the desired user story to open it.
2. Open the **Storyboard** tab.
3. Right click on the storyboard to rename and select **Rename Storyboard...** from the popup menu.
4. Click **OK** to confirm the new name.

Wireframe playback

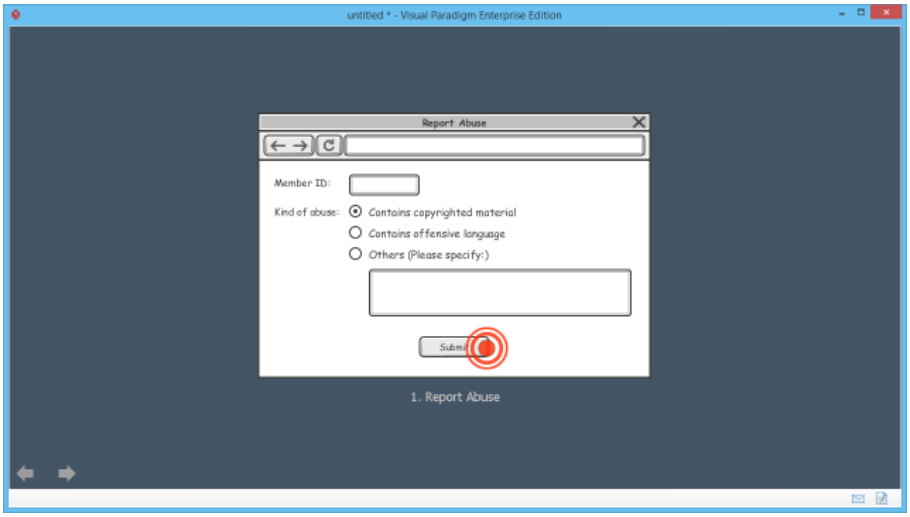
Showing a screen flow of the system to your customer guarantees your customer knows what will be delivered by the end of the project. Visual Paradigm not only allow you to associate use case scenario with wireframes in illustrating system interations, but also supports playing the wireframes associated with wireframe storyboard. This can be very useful when you need to present the system design ideas to your customers, and to look for their consent.

Playing wireframes

1. In the **User Story** tab of UeXceler, double click on the desired user story to open it.
2. Open the **Storyboard** tab.
3. Click on the **Play Storyboard** button on the right of the scren.

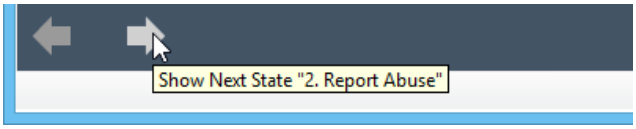
Play storyboard

4. This opens the wireframe player. The wireframe of the first step is shown in the player.



Playing a wireframe

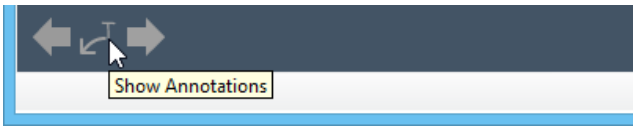
You can move on to the next wireframe by pressing the **Right** key, or clicking on the arrow button at the bottom left of the player. Similarly, you can press the **Left** key or click on the back button at the bottom left of the player to move to the previous wireframe.



Move to the next wireframe

Showing Annotations

To keep the wireframe clear and readable, annotations are hidden by default. You may click on the Show Annotations button t the bottom left of the player to have them visible. Let's show the annotations.



Show annotations

Ending the Show

You can end the show anytime by pressing the **Esc** key. When it arrives the final wireframe, you can also exit by clicking on the Exit button at the bottom left of the player.

Exit playback

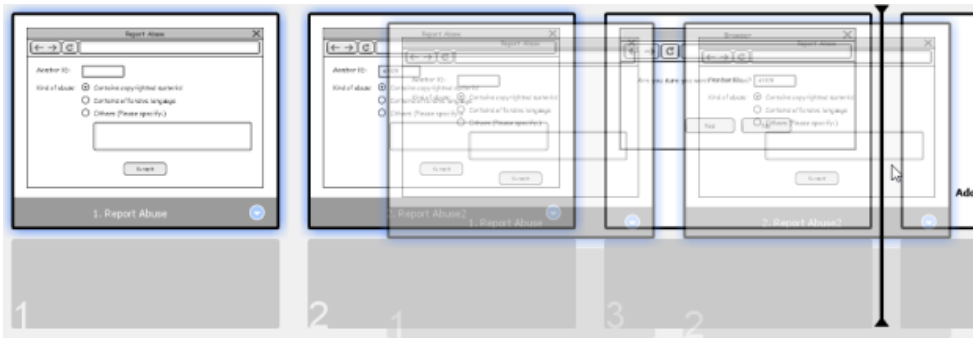
Reordering wireframes in a storyboard

You can always reorder the wireframes added to a storyboard by performing these steps:

1. Select **Modeling > Wireframe Storyboard** from the toolbar.
2. Double click on the desired storyboard to open it.
3. In the center of the editor, select the wireframe(s) to reorder. You can perform multiple selection by first pressing **Ctrl** on a wireframe, and then select the other wireframes one by one.

Select the wireframes to reorder

4. Drag your selection to the target place.



Reordering wireframes

5. Release the mouse button.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

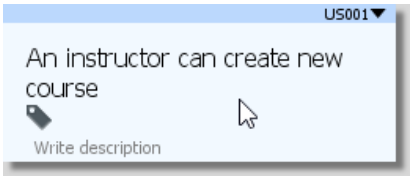
- YouTube Video - Requirements Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Adding sub-diagrams and references to user stories

Once the requirements are gathered, development team can proceed with designing and implementing the user stories, following the planning of sprint. Throughout the process, you can add sub-diagrams or reference material to user stories to help designing the system. A sample usage would be to draw a sub-activity diagram to visualize the desired behavior of a user story.

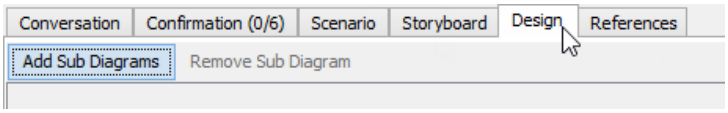
Adding sub-diagrams to a user story

1. In the **User Story** tab of UeXceler, double click on the desired user story to open it.



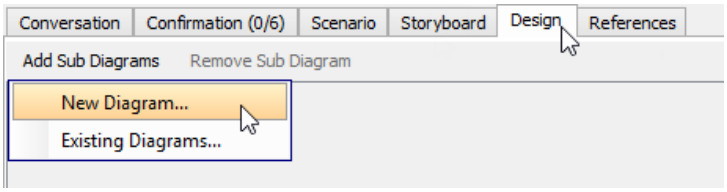
Open a user story

2. Open the **Design** tab.



Open the Design tab

3. Click on **Add Sub Diagrams**. You can add a new diagram as sub-diagram by selecting **New Diagram...** from the popup menu. Then, choose the type of diagram to create in the **New Diagram** window and click **OK** to confirm. Alternatively, you can add an existing diagram as a sub-diagram by selecting **Existing Diagrams...** from the popup menu.



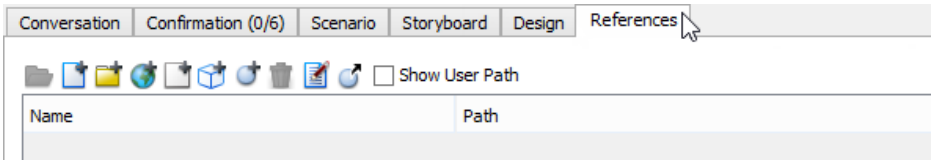
Add a new diagram as sub-diagram

Sub-diagrams added are all listed in the **Design** tab. You can double click on a thumbnail to open the sub-diagram.

Adding references to a user story

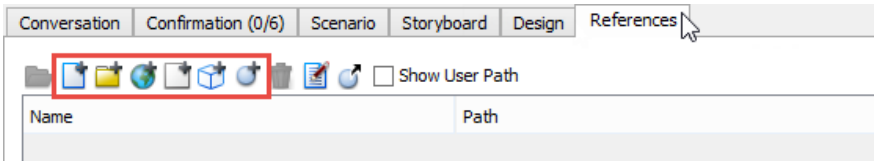
Supplementary resources can be attached to a user story by adding references, such as inserting a model element, a diagram, a file, a folder and a URL.

1. In the **User Story** tab of UeXceler, double click on the desired user story to open it.
2. Open the **References** tab.



Open the Reference tab

3. On the toolbar, click on the type of reference to add.



Add reference

4. Select/configure the reference to add.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - Requirement tos Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Search bar

You might have created a lot of user stories, which makes it difficult to locate the one you want. The search bar is a convenient way of locating the user story(ies) you want without having to scroll through the page. To search for a user story, press **Ctrl-F** or select **Diagram > Search** from the toolbar to toggle the search bar. As you begin typing your search string in the search bar, the first result will be highlighted in the active page.

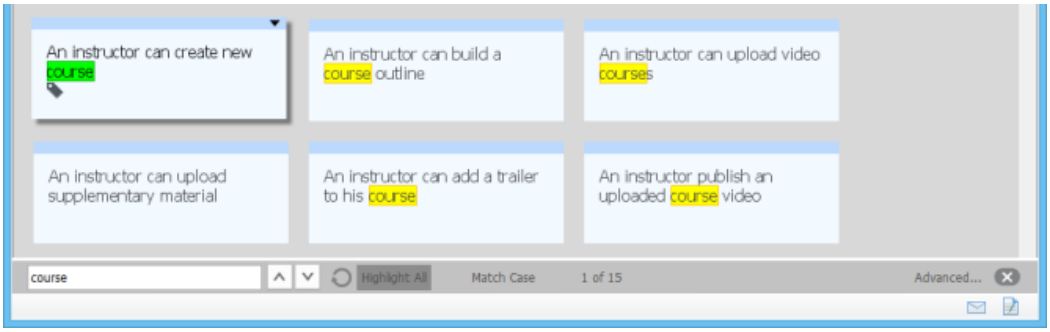
Searching with search bar

Moving to next and previous result

There may be multiple user stories that match your search string. If you would like to see all the results for your search string, press **Enter** to proceed to the next occurrence after you've typed the search string. Alternatively, press the ^ and v buttons in the search bar to move between search results.

Highlighting all results

You can also highlight all the search results by clicking the **Highlight All** button in the search bar.



All results highlighted

Match case

Searching is a case in-sensitive process by default. If you want it to be case sensitive, click **Match Case** in the search bar.

Closing the search bar

To close the search bar, click on the ⊗ button on the right of the search bar.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - Requirement tos Gathering with UeXceler
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Wireframe

What is wireframe?

Learn what wireframe is.

What is a wireframe state?

Learn how to re-use existing wireframe and make small modification by creating wireframe state.

Android phone wireframing skills

Learn how to draw wireframe for Android phones.

Android tablet wireframing skills

Learn how to draw wireframe for Android tablet.

Desktop wireframing skills

Learn how to draw wireframe for desktop applications.

iPad wireframing skills

Learn how to draw wireframe for iPad apps.

iPhone wireframing skills

Learn how to draw wireframe for iPhone apps.

Web wireframing skills

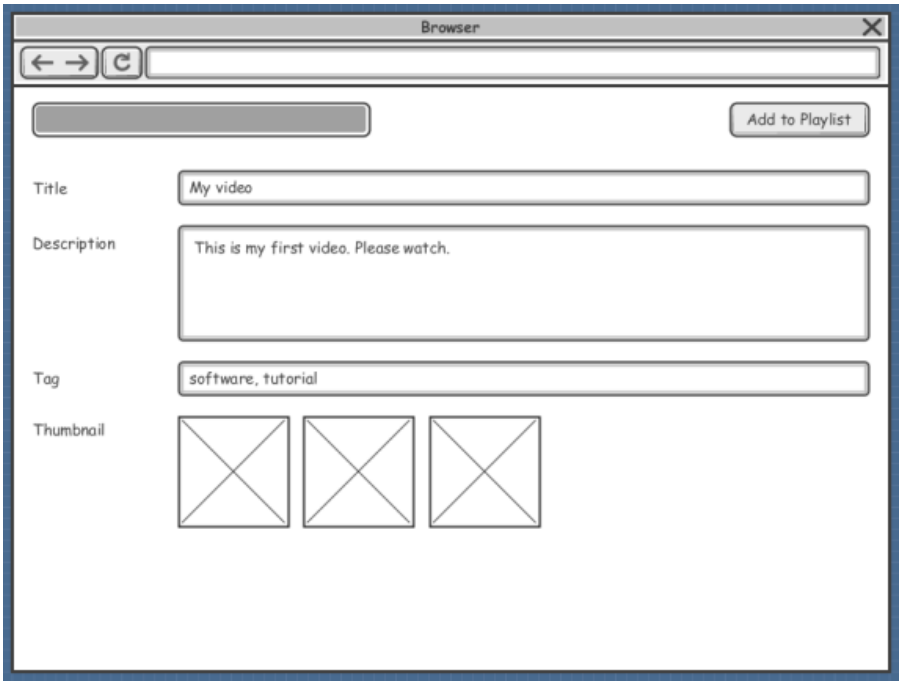
Learn how to draw wireframe for web site.

Duplicate wireframe

The duplicate function allows you to re-use an existing wireframe ito create a new one. This page describes the steps required to duplicate a wireframe.

What is wireframe?

A wireframe is a sketch of the system to be built. It's simple, clear and allows everyone to read and understand easily. Wireframe shows "just enough" information of the screen instead of the full details. The actual screen design will be produced at a later stage by referencing the wireframe. You can show the scenario to your customer visually to obtain consent about the requirements.



Sample wireframe

Benefits of wireframing

Comparing to prototyping or any kind of detailed screen designs, wireframe features the following advantages:

- Easy to draw: Wireframe has a simple and clean layout. It is formed by simple screen elements without any detailed styling and formattings.
- Easy to understand: Wireframe is welcomed by both the development team and business people. It is so simple that everyone can understand without learning.
- Easy to modify: You don't need any programming to visualize new design ideas.
- No coding required: No heavy prototyping and no coding. You just need to draw the wireframe as if you are using a drawing tool.
- In-line annotations: Annotate design ideas in-place with the help of annotation shapes. These annotations can be shown in requirement specification as well.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

What is a wireframe state?

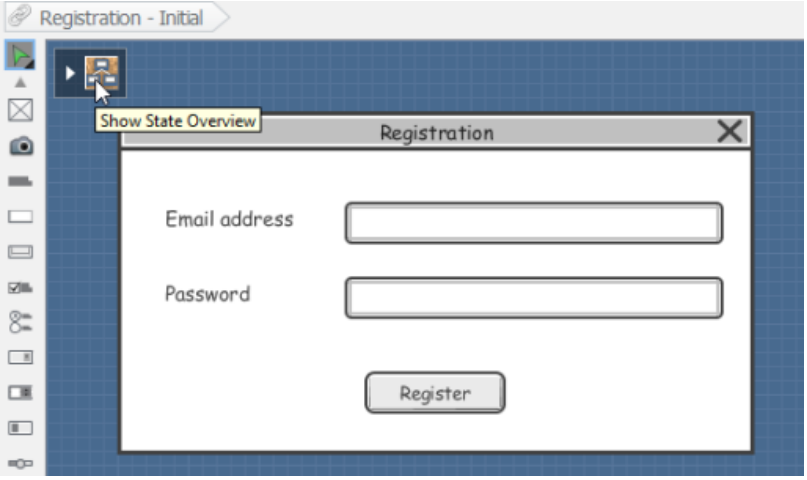
A wireframe state is a snapshot of wireframe throughout an interaction. Generally speaking, a state changes when an event occurs. For example, an entry of text in text field, a click of a button, etc.

Visual Paradigm supports wireframing. You can create wireframe and represent the modifications of screen over time by defining states.

Adding a child wireframe state

To create a wireframe state:

1. Open the wireframe where you want to add a state.
2. Click on **Show State Overview** at the top-left of wireframe.



Show State Overview

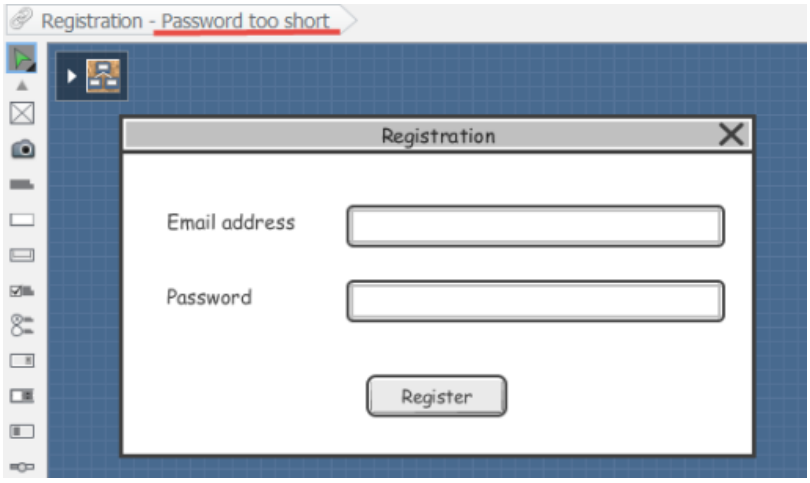
3. This opens the state overview with states placed on a corkboard-like background. Click **Create Child State** below the Initial state to create a child state under it.

Create child state

4. Name the wireframe state meaningfully to reflect the change of the wireframe at this particular state.

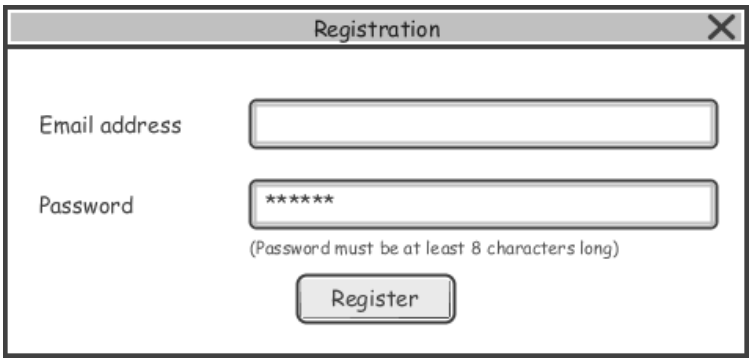
Naming state

5. Press **Enter**. This opens the wireframe state for editing. You can check the editing state by referring to the diagram title.



Check the name of editing state

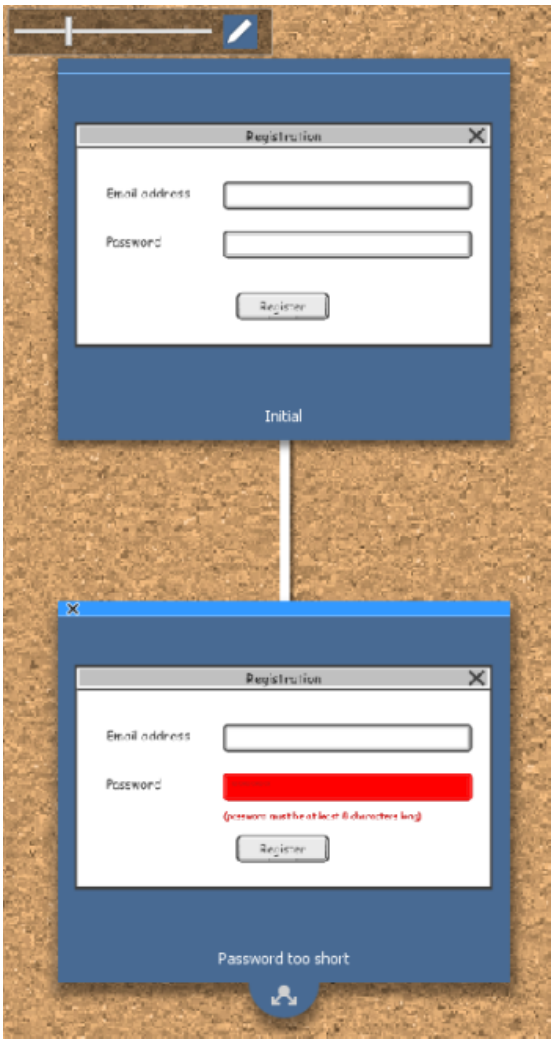
6. You can now start editing.



Wireframe edited

Checking the differences between states

The state overview shows you the thumbnails of wireframe states plus the differences between parent and child state. The modified regions are paint in red. You can check the red regions to identify the changes made in a particular wireframe state. If you are failed to read the thumbnail clearly, you may double click on a thumbnail to open it or adjust the zoom ratio by dragging on the slider at the top of the overview.



Differences are highlighted

Synchronization of change between states

Generally speaking, when you make a change in a parent state, like adding a new wireframe element, modifying an element (e.g. enter a text in text box) or removing element, the change applies to all its child state, provided that the modified element has not been modified in the child states. Here are some cases that can help in explaining the idea:

Case 1
If I add a new button in parent state, the same button will appear in its child state.

Case 2
If I removed a button in parent state but that button has not been modified in its child state, the button will still be removed.

Case 3
If I renamed the caption of a button in parent state but the same button has not been renamed in child state, the caption of the button in child state will be renamed. Note that if the button in child state was resized, the renaming will still take place.

Case 4
If I renamed the caption of a button in parent state and the same button has been renamed in child state, the caption of the button in child state will remain unchanged.

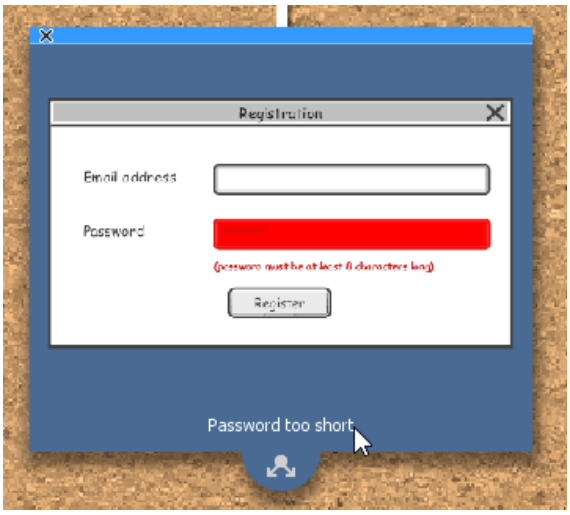
Editing a child wireframe state

To edit a child wireframe state:

1. Open the wireframe where you want to edit a state.
2. Click on **Show State Overview** at the top left of wireframe.
3. Double click on the thumbnail of the state that you want to edit.
4. The wireframe state is opened and you can start editing.

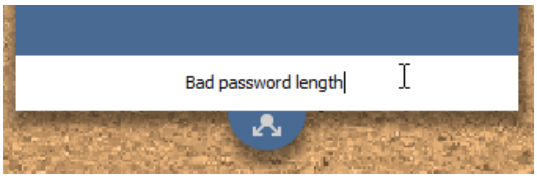
Renameing a wireframe state
To rename a child wireframe state:

1. Open the wireframe where you want to rename a state.
2. Click on **Show State Overview** on the top-left of wireframe.
3. Double click on the name under the thumbnail of the state that you want to rename.



Double click on the name of a wireframe state

4. Enter a new name and press **Enter** to confirm editing.

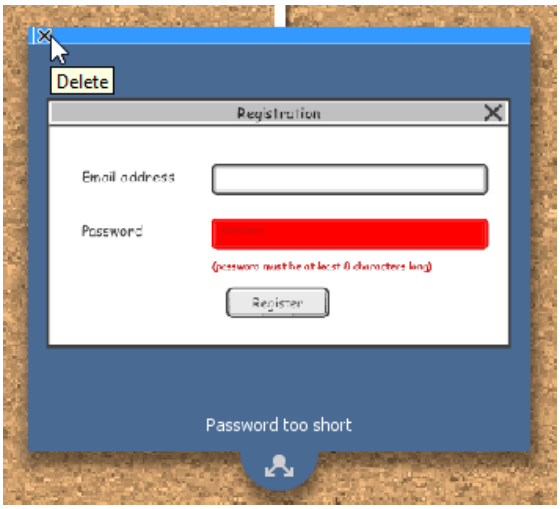


Renaming a wireframe state

Deleting a child wireframe state

By deleting a child wireframe state, you are not only deleting the chosen state but also all of its children states. The deletion is not undo-able, so think twice before you delete a wireframe state. To delete a child wireframe state

1. Open the wireframe where you want to delete a state.
2. Click on **Show State Overview** on the top-left of wireframe.
3. Click on the thumbnail of the state that you want to delete.
4. Click on the tiny cross button on the top-left corner.



Delete a wireframe state

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

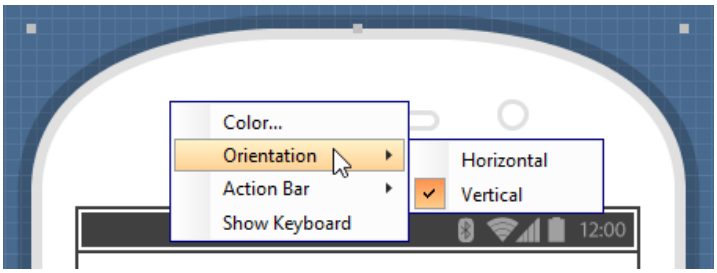
- Professional Edition

- Enterprise Edition

Android phone wireframing skills

Changing the orientation of Android phone

Initially, the Android phone is shown vertically in the wireframe. If your apps works under a horizontal layout, you can change its orientation. To adjust the orientation, right click on the phone border and select **Orientation > Horizontal/Vertical** from the popup menu.



Changing orientation of Android phone

NOTE: You can only change orientation when there is no wireframe element created inside the phone body

Show/Hide action bar

To hide the action bar and toolbar, right click on the phone border and select **Action Bar > Show** or **Action Bar > Show with Drawer** from the popup menu.

Show action bar

Show/Hide keyboard

To show the keyboard, right click on the phone border and select **Show Keyboard** from the popup menu.

Show keyboard

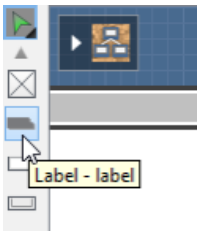
This shows the keybaord at the bottom of the phone:

Keyboard shown

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).



Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

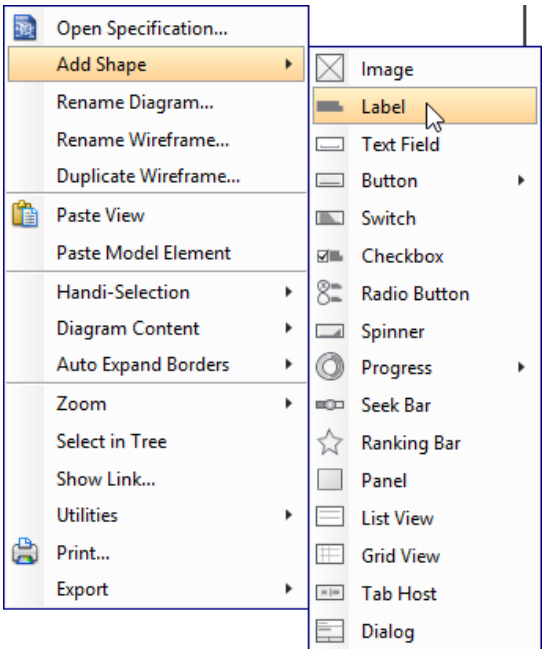
1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

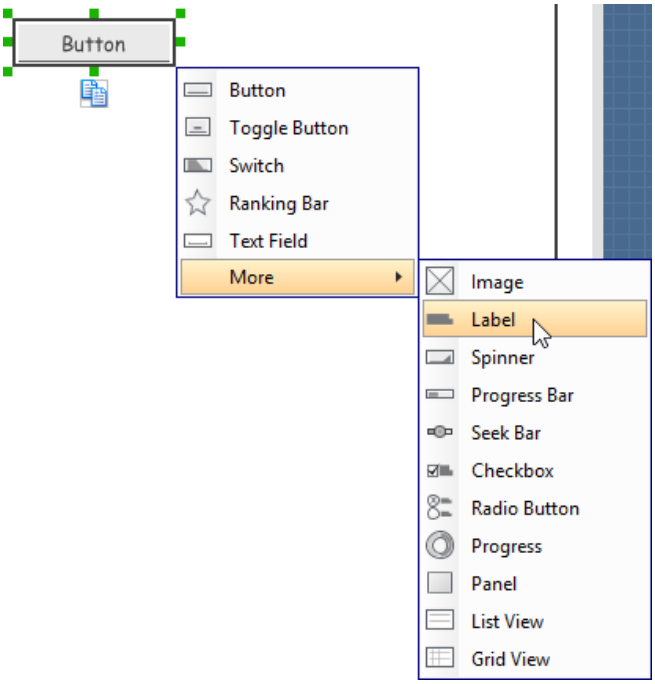
1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appears, known as the **Smart Create** resource.

Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.
3. Drag to outline the size of the wireframe element to be created.

Creating a label in specific size

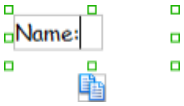
4. Release the mouse button. In the popup menu, choose the type of wireframe element to be created.



Choosing the wireframe element to be created

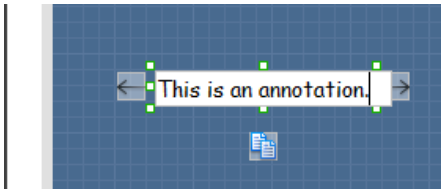
Method 5 - Double-clicking (Label and annotation only)

To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the phone border and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

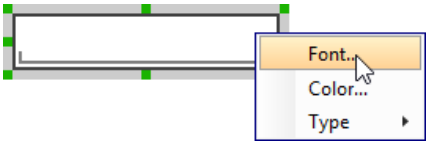
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you to align elements perfectly with others. Simply select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjust the positioning of the selection with the help of the guide.

Using the alignment guide

Adjusting font property of wireframe elements

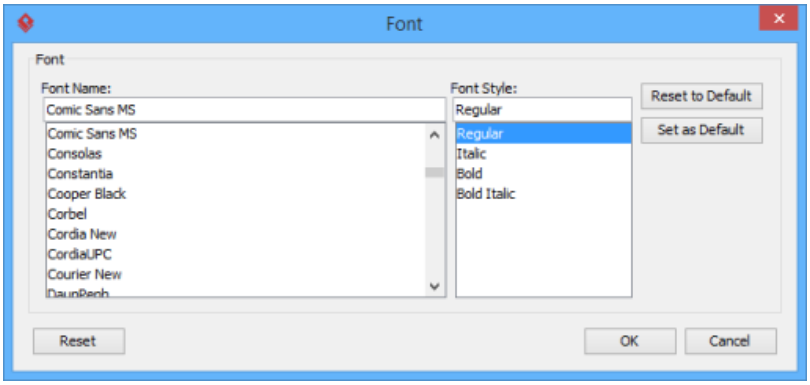
For wireframe elements that can display text, you can adjust their font properties like the font familiy and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

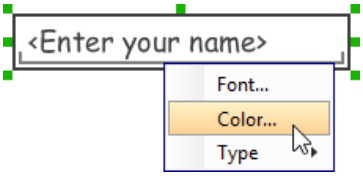


The Font window

Setting color for wireframe elements

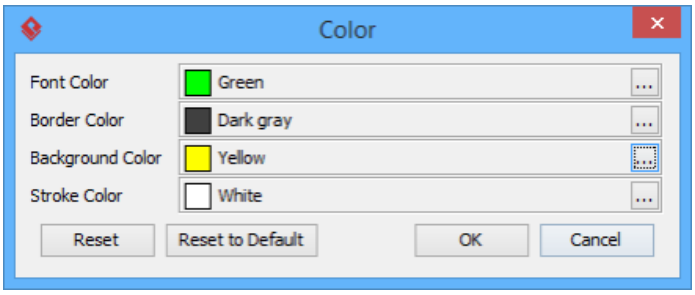
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, list view doesn't.



Editing color properties

You will then see the new color applied.

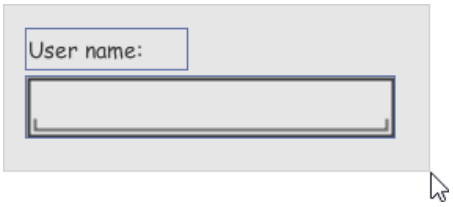
Color applied

Duplicating wireframe elements

Duplicating wireframe elements enables you to create new elements based on the existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

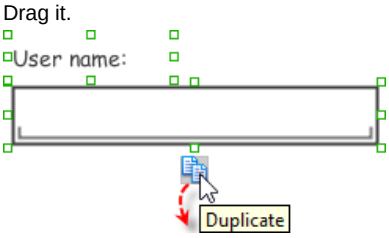
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to be duplicated.



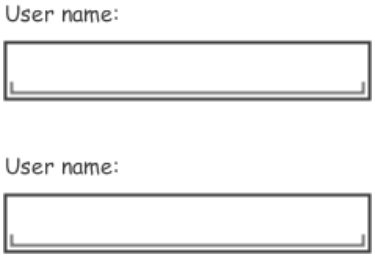
Selecting wireframe elements to duplicate

2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



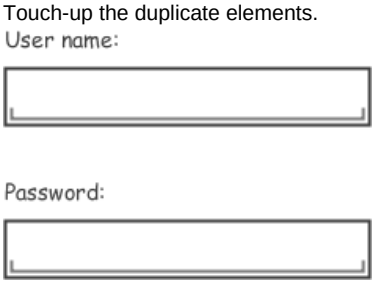
To duplicate wireframe elements

4. Release the mouse button at the position where you want the wireframe element to be created.



Wireframe elements duplicate

5. Touch-up the duplicate elements.



Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may also click on it..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

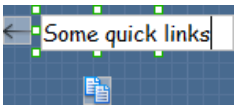
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the phone border. In other words, you cannot create or move an annotation inside the phone body.

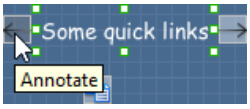
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



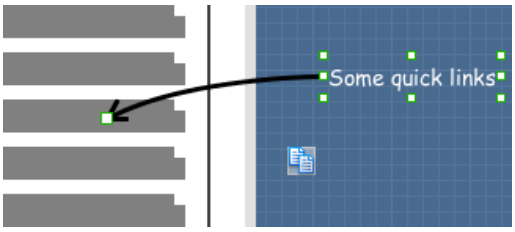
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

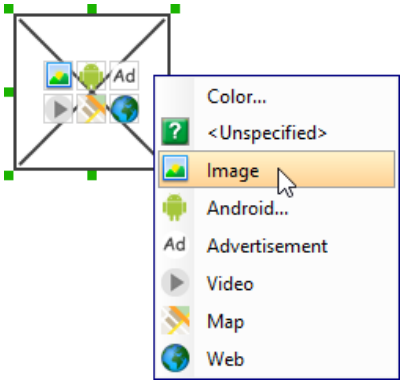
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video, map or web component. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to be embeded into the image component.



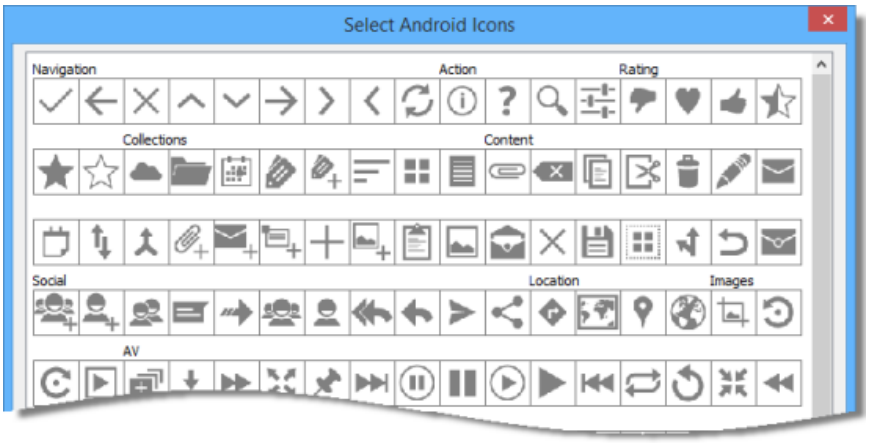
To embed image into image component

To represent an advertisement, video, map or web component, right click on the image component and select **Advertisement**, **Video**, **Map** and **Web** respectively.

Image component showed as video

Showing Android icon

You can also use an image component to show an Android icon for a tab bar. To show an Android icon, right click on the image component and select **Android...** from the popup menu. In the popup window, choose the icon to show and click **OK** to confirm.

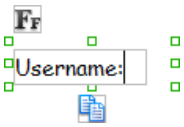


Choosing an Android icon

Wireframing tips - Label

Specifying the content of label

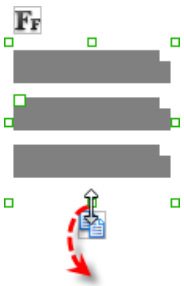
To specify the content of a label, double click on the label and enter the content. You can press **Enter** to create a new line or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

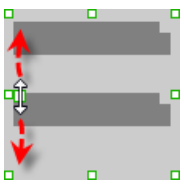
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.

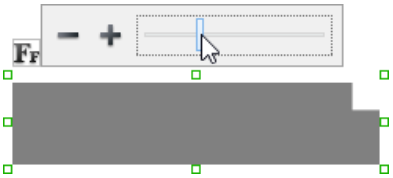


Adjusting the spacing between labels

NOTE: When a label has specified content, you cannot show multiple labels in it.

Adjusting label or font size

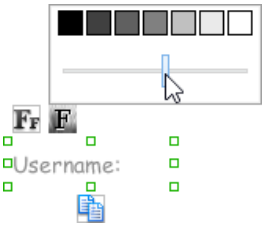
When the content is filled, the size of label(s) or text in label can be changed. To adjust the font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.



Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.

Specifying the content of text field

Showing the text field as a search field

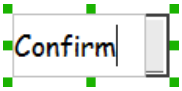
Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Type > Search** or **Type > Search with Icon** from the popup menu.

A search field

Wireframing tips - Button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.



Entering button caption

Wireframing tips - Toggle Button

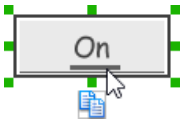
Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.

Entering button caption

Altering the state of button

To alter the state of a switch, select the switch first. Then, click the line at the bottom of the button to change its state.

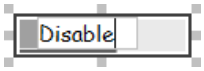


Altering the state of toggle button

Wireframing tips - Switch

Editing switch caption

To edit the caption of a switch, double click on the text in the switch and enter the caption. You may need to resize the switch afterwards in order to see the caption entered.



Entering switch caption

Altering the state of switch

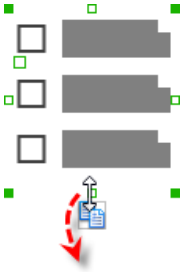
To alter the state of a switch, select the switch first. Then, click on the inactive end of the switch to switch to that end.

Altering switch state

Wireframing tips - Checkbox

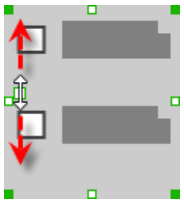
Creating the checkboxes

The checkbox component is in fact a placeholder of checkboxes. You can show multiple checkboxes in it by increasing the height of the checkbox component.



Creating more checkboxes

To adjust the spacing between checkboxes in a checkbox component, select the checkbox and drag the handler between the first and the second checkbox. Space will be added by dragging downwards.



Adjusting the spacing between checkboxes

Specifying the value of checkbox

To specify the value of a checkbox, double click on the label attached with the checkbox and enter the value. You may need to resize the checkbox afterwards in order to see the content entered.

Entering the value of a checkbox

Checking a checkbox

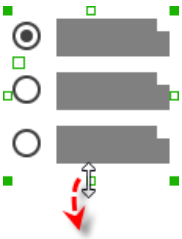
To check a checkbox, simply click on the checkbox. You can uncheck it by clicking again.

Checking checkboxes

Wireframing tips - Radio Button

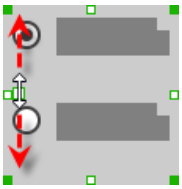
Creating the radio buttons

The radio button component is in fact a radio button group. You can show multiple radio buttons in it by increasing the height of the radio button component.



Creating more radio buttons

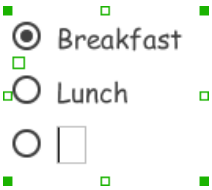
To adjust the spacing between radio buttons in a radio button component, select the radio button and drag the handler between the first and the second radio button. Space will be added by dragging downwards.



Adjusting the spacing between radio buttons

Specifying the value of radio button

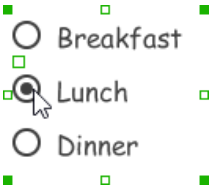
To specify the value of a radio button, double click on the label attached with the radio button and enter the value. You may need to resize the radio button afterwards in order to see the content entered.



Specifying the value of radio button

Selecting a radio button

To select a radio button, simply click on the radio button. You can uncheck it by clicking again.

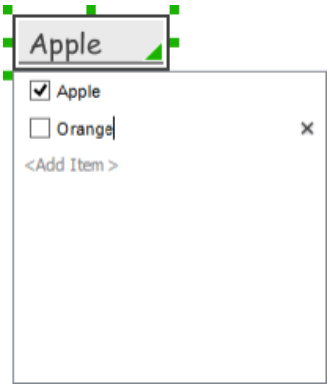


Selecting a radio button

Wireframing tips - Spinner

Specifying the items in a spinner

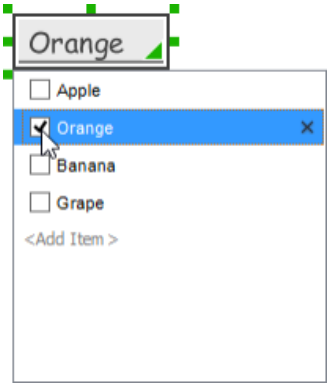
By default, a spinner has no items specified. You can add a list of items into a spinner by clicking on the down arrow on the right of the spinner and then click on **<Add Item>** and start entering the item.



Adding an item to spinner

Changing the selected item in a spinner

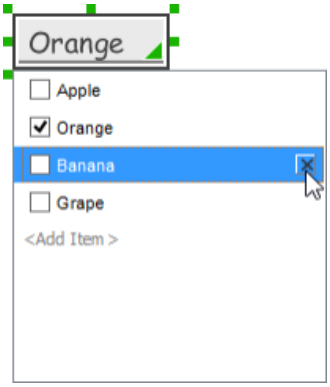
To change the selected item of a spinner, click on the down arrow on the right of the spinner, then check the item to select it.



Selecting an item in spinner

Removing an items from a spinner

To remove an item from a spinner, click on the down arrow on the right of the spinner, then select the item to be removed and click on the cross button on the right to remove it.

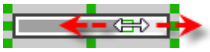


Removing an item from spinner

Wireframing tips - Progress Bar

Adjusting the progress

To adjust progress, select the progress bar first. Then drag the handler in the middle towards left or right to control the progress.

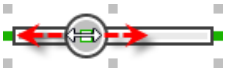


Adjusting the progress of progress bar

Wireframing tips - Seek Bar

Adjusting the slider position

To adjust silder position, select the seek bar first. then drag the handler in the middle towards left or right to control the position.



Adjusting the slider position

Wireframing tips - Ranking Bar

Adding more stars

To add more stars to a ranking bar, select the ranking bar first. Then, extend the ranking bar to let more stars appear.

Adding more stars to a ranking bar

Adjusting the number of filled stars

To adjust the number of filled stars, select the ranking bar first. Then, fill the stars by clicking on a star in the ranking bar.

Adjusting the number of filled stars

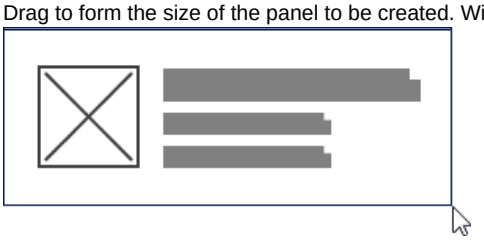
Wireframing tips - Panel

Containing existing components with panel

Panel is a useful wireframe component that helps you to visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

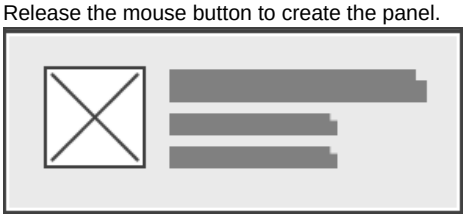
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

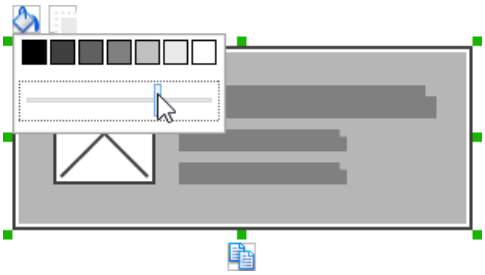
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

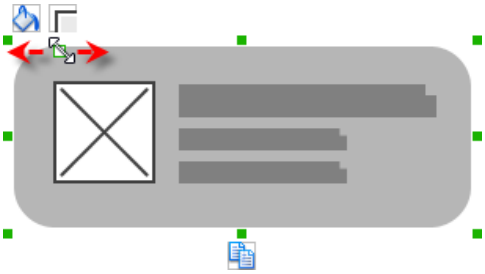
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler on the top-left corner to adjust the size of the rounded corner. The four corners will be updated accordingly.

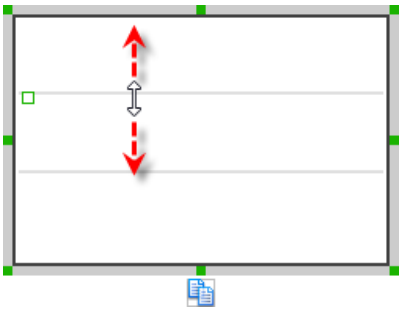


Making the corner of panel rounded

Wireframing tips - List View

Adjusting row height (for single row)

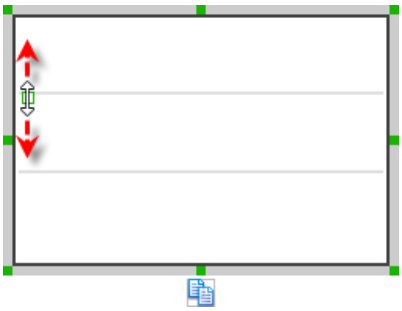
To adjust the height of a row, drag directly on the row separator under the row. By doing so, the row will be expanded or contracted.



Adjusting row height of a list view

Adjusting rows height (for all rows)

To adjust the height of all rows in a list view, drag on the handler attached to the row separator between the first and the second row to resize all rows at the same time.

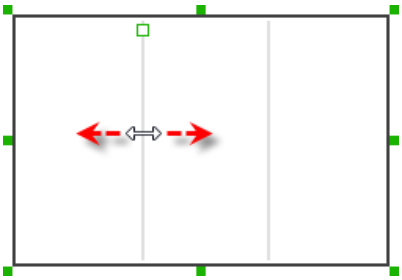


Adjusting rows' height of a list view

Wireframing tips - Grid View

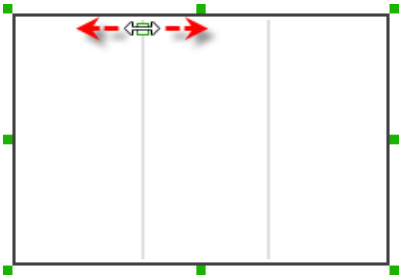
Adjusting column width

To adjust column width, drag directly on the column separator nearby. By doing so, the adjacent columns will be updated in their width.



Adjusting column width of a table

Adding more columns
To add more columns to a table, select the table first. Then, drag on the handler attached to the column separator between the first and the second column to create more columns.



Adding more columns to a table

Adding more rows
When you put a wireframe component (e.g. a label component) into a table component, a new row will be created automatically.

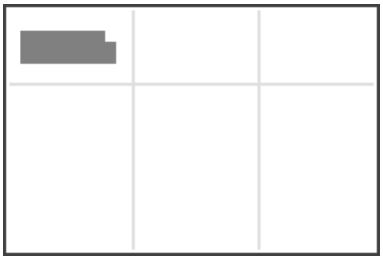
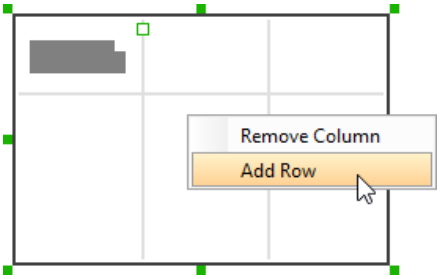


Table with one row

You may also add a row manually by right clicking on the table and selecting **Add Row** from the popup menu.



Adding row to a table

Wireframing tips - Tab Host

Editing the caption of a tab
To edit the caption of a tab, double click on the tab and enter the title.

Editing the tab caption

Adding more tabs
To add more tabs to a tab host, select the tab host first. Then, drag on the handler attached to the separator between tabs to create more segments.

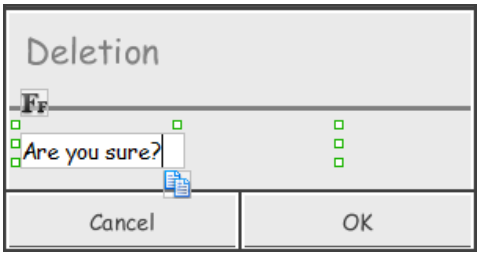
Adding more tabs to a tab host

Changing the active tab
To change the active tab, select the tab host first. Then, click on the tab directly in the tab host.

Changing the selected tab in a tab host

Wireframing tips - Dialog

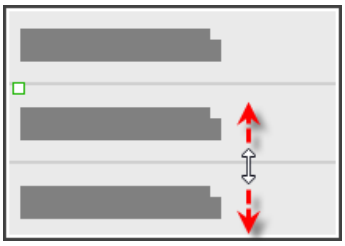
Editing the content of labels and buttons in a dialog
To edit the content of labels and buttons in a dialog, double click on the label or button and enter the content. You may need to resize the label or the dialog afterwards in order to see the content entered.



Editing the message in a dialog

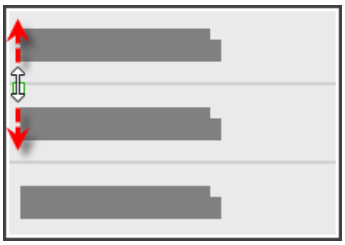
Wireframing tips - Menu

Adjusting row height (for single row)
To adjust the height of a row, drag directly on the row separator under the row. By doing so, the row will be expanded or contracted.



Adjusting row height of a menu

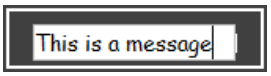
Adjusting rows height (for all rows)
To adjust the height of all rows in a menu, drag on the handler attached to the row separator between the first and the second row to resize all rows at the same time.



Adjusting rows' height of a menu

Wireframing tips - Toasts

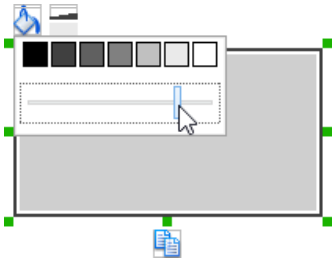
Editing the message
To edit the message of a toasts, double click on the toasts and enter the message.



Editing the toasts message

Wireframing tips - Rectangle

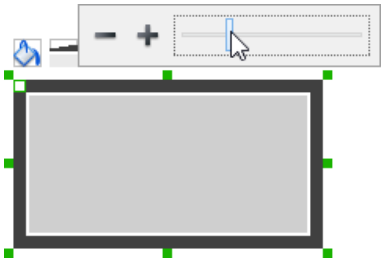
Adjusting fill color
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

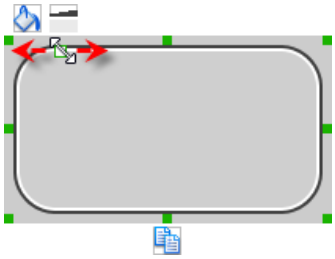
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler on top-left to adjust the size of the rounded corner. The four corners will be updated accordingly.

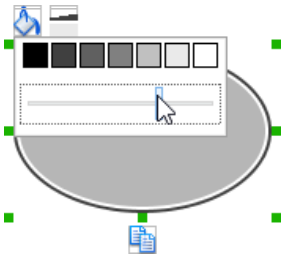


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

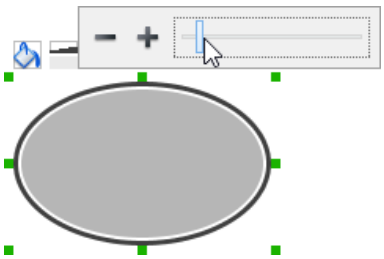
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

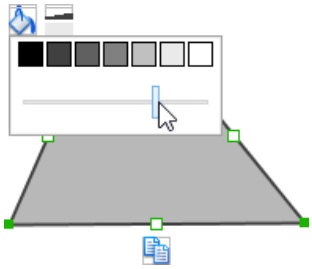
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

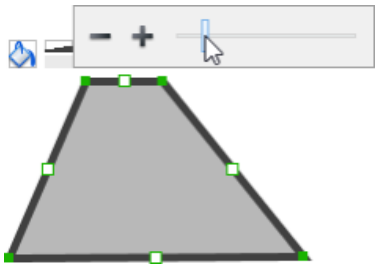
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

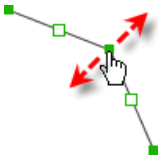


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



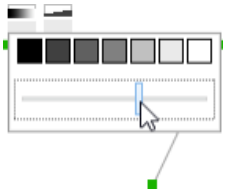
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

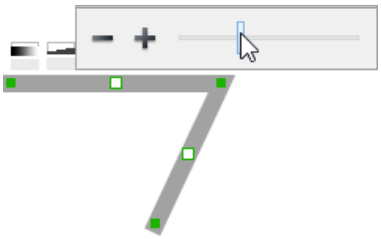
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Trademark Disclaimer

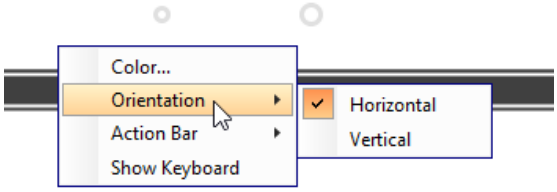
Supported Editions

- Professional Edition
- Enterprise Edition

Android tablet wireframing skills

Changing the orientation of Android tablet

Initially, the Android tablet is shown horizontally in the wireframe. If you apps works under a vertical layout, you can change its orientation. To adjust the orientation, right click on the tablet border and select **Orientation > Horizontal/Vertical** from the popup menu.

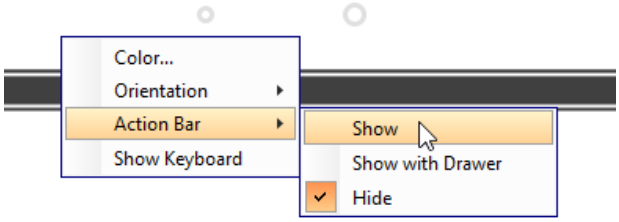


Changing orientation of Android tablet

NOTE: You can only change orientation when there is no wireframe element created inside the tablet

Show/Hide action bar

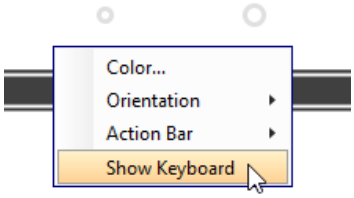
To show the action bar and toolbar, right click on the tablet border and select **Action Bar > Show** or **Action Bar > Show with Drawer** from the popup menu.



Show action bar

Show/Hide keyboard

To show the keyboard, right click on the tablet border and select **Show Keyboard** from the popup menu.



Show keyboard

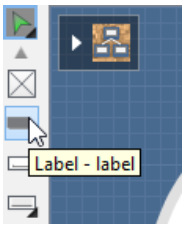
This shows the keybaord at the bottom of the phone:

Keyboard shown

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).

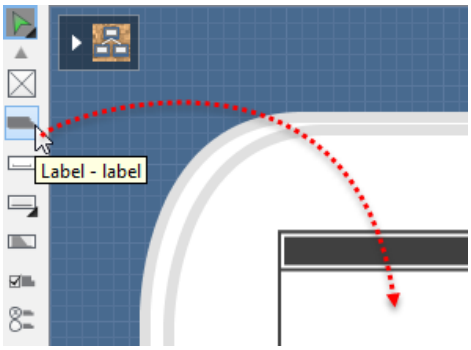


Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

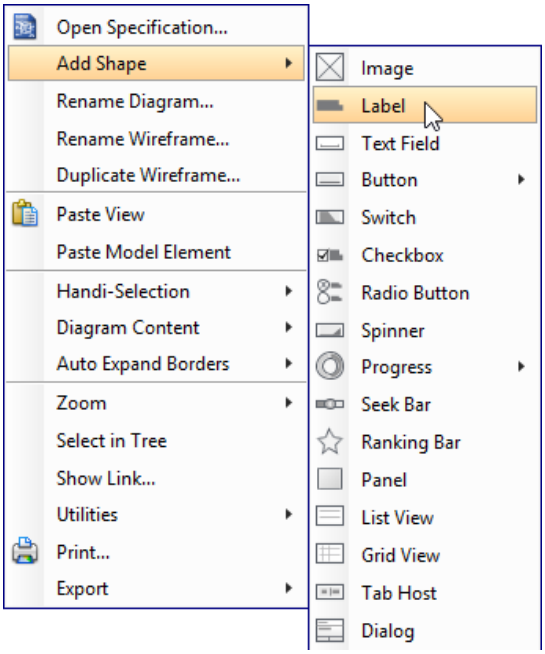


Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appear, known as the **Smart Create** resource.

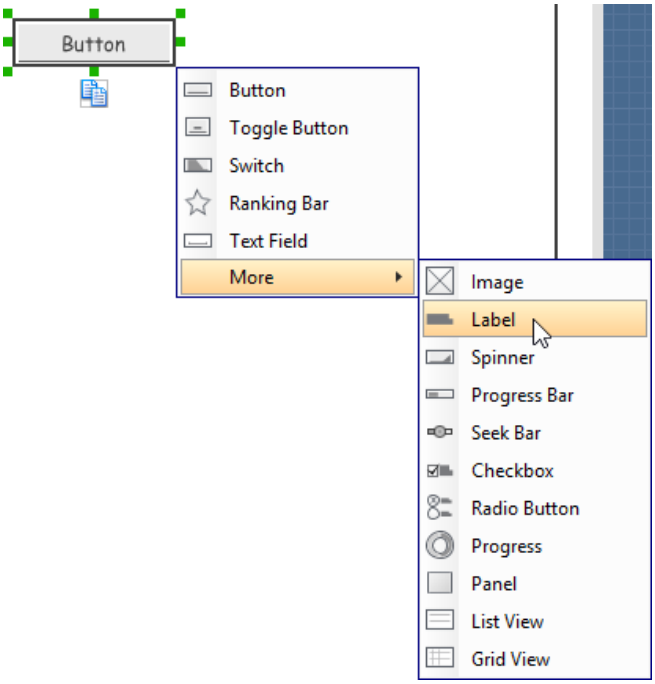
Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.

3. Drag to outline the size of the wireframe element to be created.

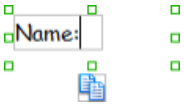
Creating a label in specific size

4. Release the mouse button. In the popup menu, choose the type of wireframe element to be created.



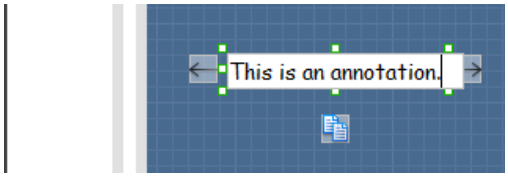
Choosing the wireframe element to be created

Method 5 - Double-clicking (Label and annotation only)
To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the tablet and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

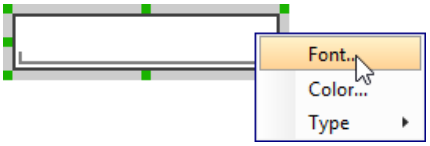
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you to align elements perfectly with others. Simple select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjust the positioning of selection with the help of the guide.

Using the alignment guide

Adjusting font property for wireframe elements

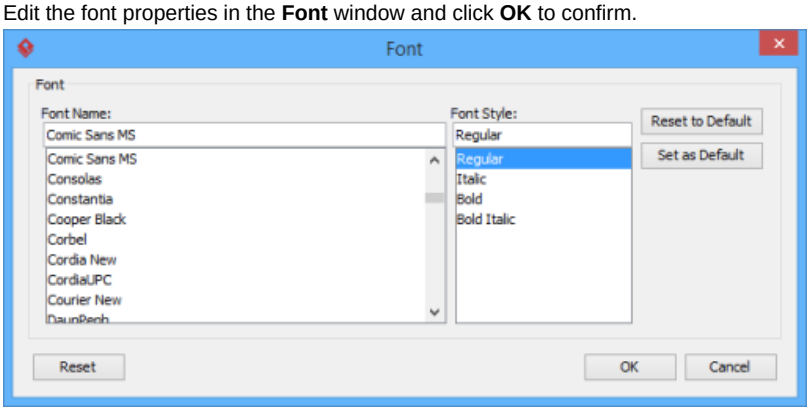
For wireframe elements that can display text, you can adjust their font properties like the font familiy to use and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

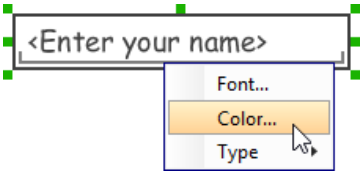


The Font window

Setting color for wireframe elements

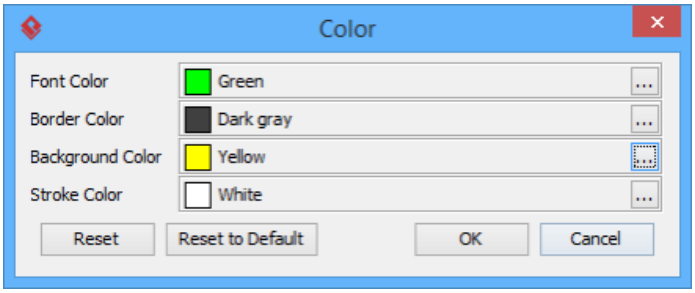
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, list view doesn't.



Editing color properties

You will then see the new color applied.

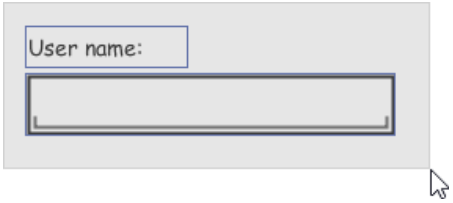
Color applied

Duplicating wireframe elements

Duplicating wireframe elements enables you to create new elements based on existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

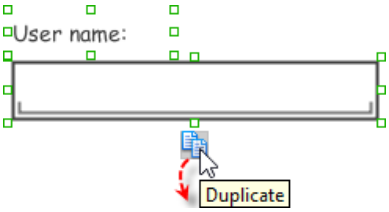
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to be duplicated.



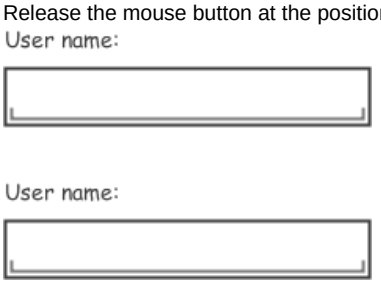
Selecting wireframe elements to duplicate

2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



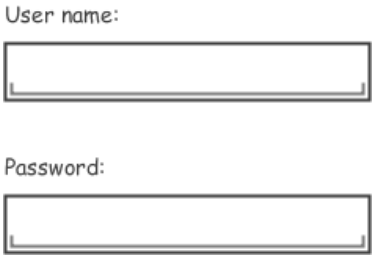
To duplicate wireframe elements

4. Release the mouse button at the position where you want the wireframe element to be created.



Wireframe elements duplicate

5. Touch-up the duplicate elements.



Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may also click on it..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

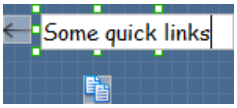
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the tablet. In other words, you cannot create or move an annotation inside the tablet.

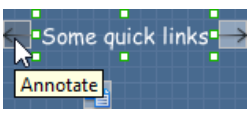
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



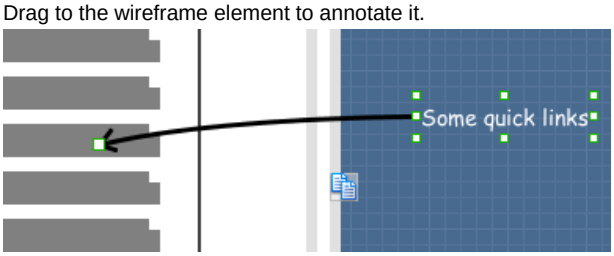
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

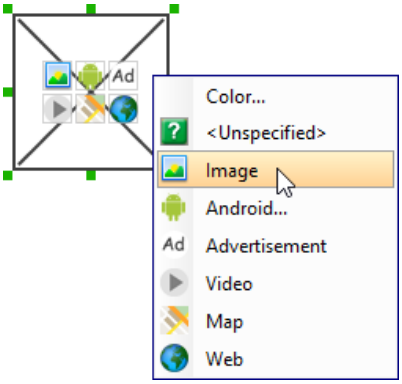
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video, map or web component. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to be embeded into the image component.



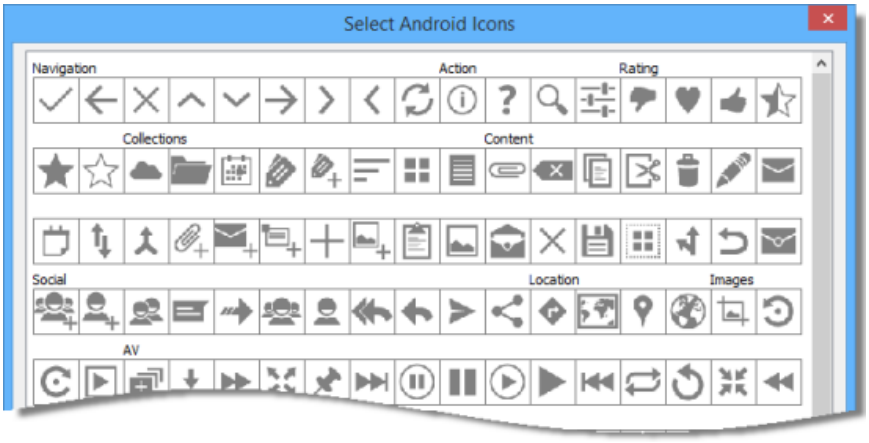
To embed image into image component

To represent an advertisement, video, map or web component, right click on the image component and select **Advertisement**, **Video**, **Map** and **Web** respectively.

Image component showed as video

Showing Android icon

You can also use an image component to show an Android icon for a tab bar. To show an Android icon, right click on the image component and select **Android...** from the popup menu. In the popup window, choose the icon to show and click **OK** to confirm.

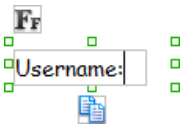


Choosing an Android icon

Wireframing tips - Label

Specifying the content of label

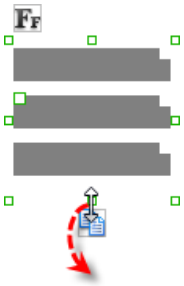
To specify the content of a label, double click on the label and enter the content. You can press **Enter** to create a new line, or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

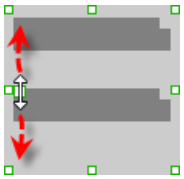
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.



Adjusting the spacing between labels

NOTE: When a label has content specified, you cannot show multiple labels in it.

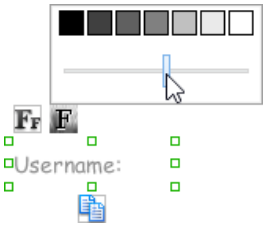
Adjusting label or font size

When the content is filled, the size of label(s) or text in label can be changed. To adjust font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.

Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.

Specifying the content of text field

Showing the text field as a search field

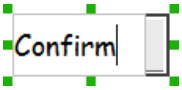
Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Type > Search** or **Type > Search with Icon** from the popup menu.

A search field

Wireframing tips - Button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.



Entering button caption

Wireframing tips - Toggle Button

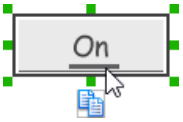
Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.

Entering button caption

Altering the state of button

To alter the state of a switch, select the switch first. Then, click the line at the bottom of the button to change its state.

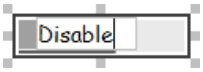


Altering the state of toggle button

Wireframing tips - Switch

Editing switch caption

To edit the caption of a switch, double click on the text in the switch and enter the caption. You may need to resize the switch afterwards in order to see the caption entered.



Entering switch caption

Altering the state of switch

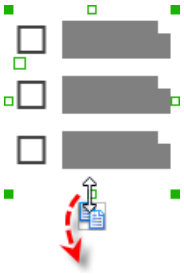
To alter the state of a switch, select the switch first. Then, click on the inactive end of the switch to switch to that end.

Altering switch state

Wireframing tips - Checkbox

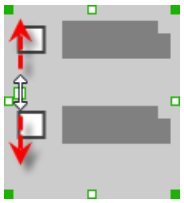
Creating the checkboxes

The checkbox component is in fact a placeholder of checkboxes. You can show multiple checkboxes in it by increasing the height of the checkbox component.



Creating more checkboxes

To adjust the spacing between checkboxes in a checkbox component, select the checkbox and drag the handler between the first and the second checkbox. Space will be added by dragging downwards.



Adjusting the spacing between checkboxes

Specifying the value of checkbox

To specify the value of a checkbox, double click on the label attached with the checkbox and enter the value. You may need to resize the checkbox afterwards in order to see the content entered.

Entering the value of a checkbox

Checking a checkbox

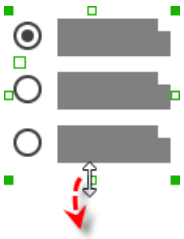
To check a checkbox, simply click on the checkbox. You can uncheck it by clicking again.

Checking checkboxes

Wireframing tips - Radio Button

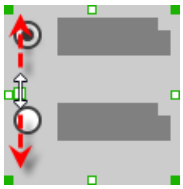
Creating the radio buttons

The radio button component is in fact a radio button group. You can show multiple radio buttons in it by increasing the height of the radio button component.



Creating more radio buttons

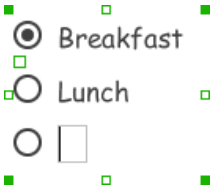
To adjust the spacing between radio buttons in a radio button component, select the radio button and drag the handler between the first and the second radio button. Space will be added by dragging downwards.



Adjusting the spacing between radio buttons

Specifying the value of radio button

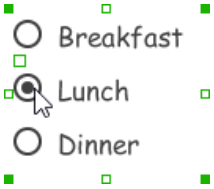
To specify the value of a radio button, double click on the label attached with the radio button and enter the value. You may need to resize the radio button afterwards in order to see the content entered.



Specifying the value of radio button

Selecting a radio button

To select a radio button, simply click on the radio button. You can uncheck it by clicking again.

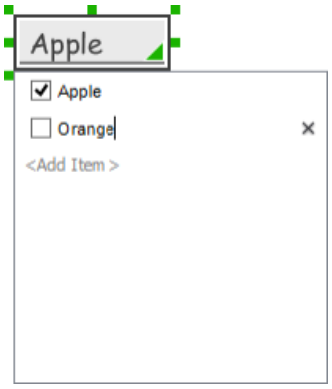


Selecting a radio button

Wireframing tips - Spinner

Specifying the items in a spinner

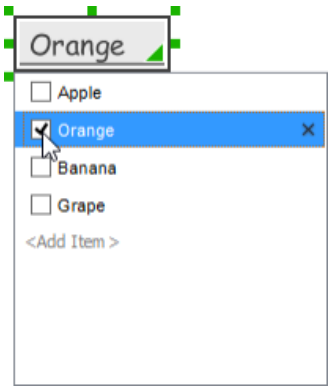
By default, a spinner has no items specified. You can add a list of items into a spinner by clicking on the down arrow on the right of the spinner and then click on **<Add Item>** and start entering the item.



Adding an item to spinner

Selecting the selected item in a spinner

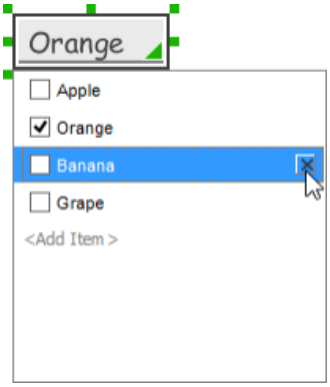
To change the selected item of a spinner, click on the down arrow on the right of the spinner, then check the item to select it.



Selecting an item in spinner

Removing an items from a spinner

To remove an item from a spinner, click on the down arrow on the right of the spinner, then select the item to remove and click on the cross button on the right to remove it.

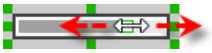


Removing an item from spinner

Wireframing tips - Progress Bar

Adjusting the progress

To adjust progress, select the progress bar first. Then drag the handler in the middle towards left or right to control the progress.

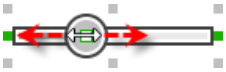


Adjusting the progress of progress bar

Wireframing tips - Seek Bar

Adjusting the slider position

To adjust silder position, select the seek bar first. Then drag the handler in the middle towards left or right to control the position.



Adjusting the slider position

Wireframing tips - Ranking Bar

Adding more stars

To add more stars tp a ranking bar, select the ranking bar first. Then, extend the ranking bar to let more stars appear.

Adding more stars to a ranking bar

Adjusting the number of filled stars

To adjust the number of filled stars, select the ranking bar first. Then, fill the stars by clicking on a star in the ranking bar.

Adjusting the number of filled stars

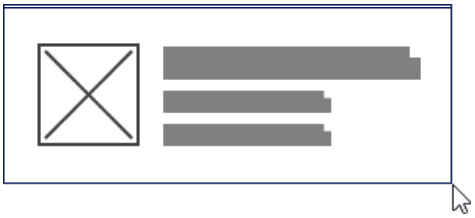
Wireframing tips - Panel

Containing existing components with panel

Panel is a useful wireframe component that helps you visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

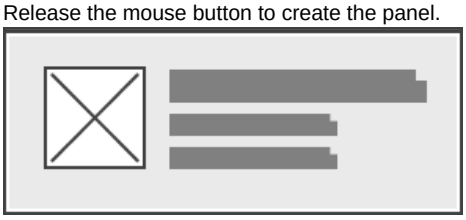
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

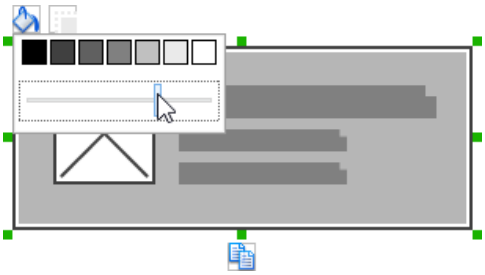
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

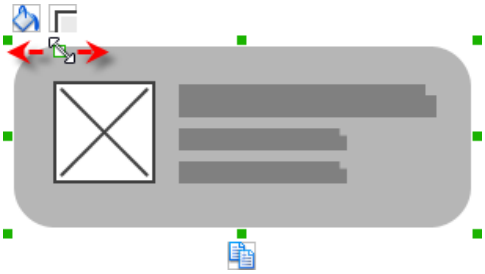
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler on the top-left corner to adjust the size of the rounded corner. The four corners will be updated accordingly.

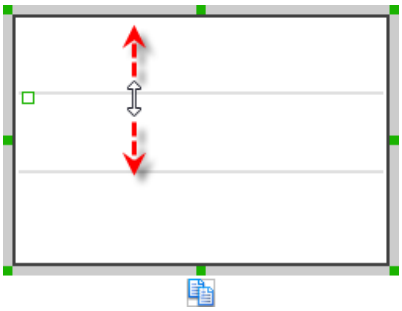


Making the corner of panel rounded

Wireframing tips - List View

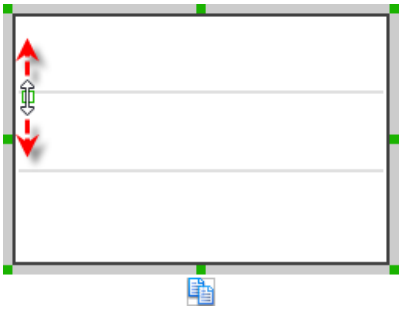
Adjusting row height (for single row)

To adjust the height of a row, drag directly on the row separator under the row. By doing so, the row will be expanded or contracted.



Adjusting row height of a list view

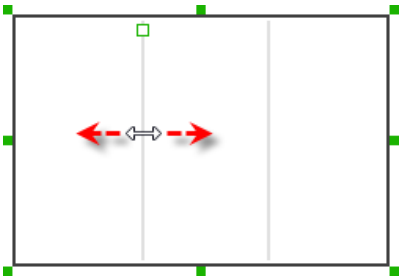
Adjusting rows height (for all rows)
To adjust the height of all rows in a list view, drag on the handler attached to the row separator between the first and the second row to resize all rows at the same time.



Adjusting rows' height of a list view

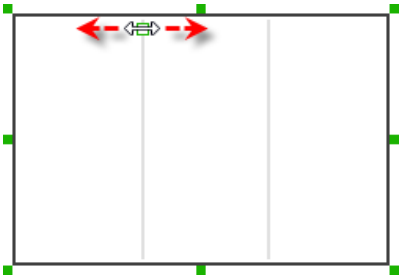
Wireframing tips - Grid View

Adjusting column width
To adjust column width, drag directly on the column separator nearby. By doing so, the adjacent columns will be updated in their width.



Adjusting column width of a table

Adding more columns
To add more columns to a table, select the table first. Then, drag on the handler attached to the column separator between the first and the second column to create more columns.



Adding more columns to a table

Adding more rows
When you put a wireframe component (e.g. a label component) into a table component, a new row will be created automatically.

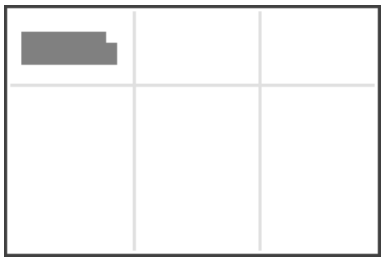
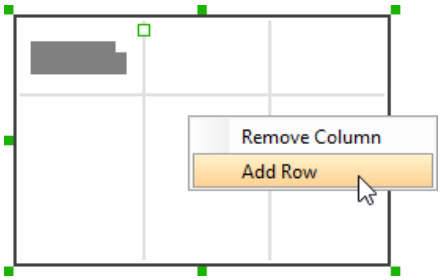


Table with one row

You may also add a row manually by right clicking on the table and selecting **Add Row** from the popup menu.



Adding row to a table

Wireframing tips - Tab Host

Editing the caption of a tab
To edit the caption of a tab, double click on the tab and enter the title.

Editing the tab caption

Adding more tabs
To add more tabs to a tab host, select the tab host first. Then, drag on the handler attached to the separator between tabs to create more segments.

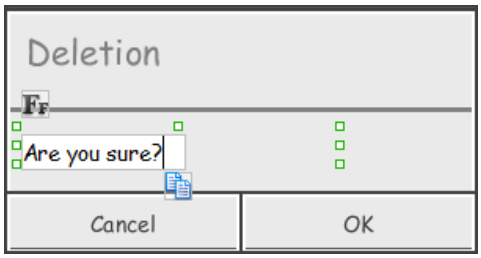
Adding more tabs to a tab host

Changing the active tab
To change the active tab, select the tab host first. Then, click on the tab directly in the tab host.

Changing the selected tab in a tab host

Wireframing tips - Dialog

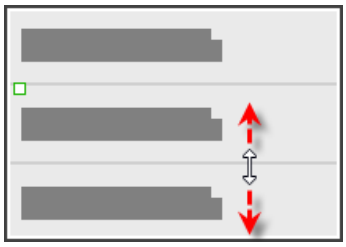
Editing the content of labels and buttons in a dialog
To edit the content of labels and buttons in a dialog, double click on the label or button and enter the content. You may need to resize the label or the dialog afterwards in order to see the content entered.



Editing the message in a dialog

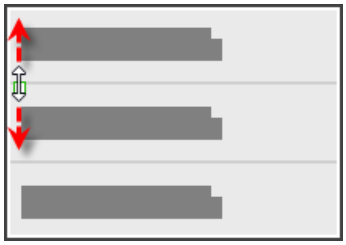
Wireframing tips - Menu

Adjusting row height (for single row)
To adjust the height of a row, drag directly on the row separator under the row. By doing so the row will be expanded or contracted.



Adjusting row height of a menu

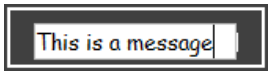
Adjusting rows height (for all rows)
To adjust the height of all rows in a menu, drag on the handler attached to the row separator between the first and the second row to resize all rows at the same time.



Adjusting rows' height of a menu

Wireframing tips - Toasts

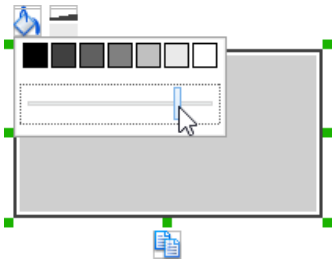
Editing the message
To edit the message of a toasts, double click on the toasts and enter the message.



Editing the toasts message

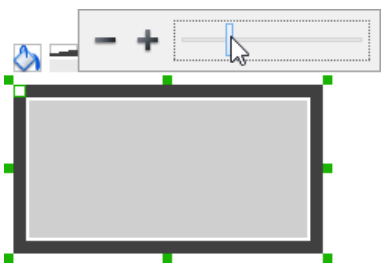
Wireframing tips - Rectangle

Adjusting fill color
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

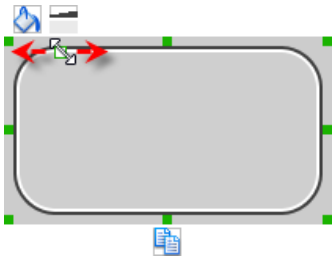
Adjusting the thickness of border
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.

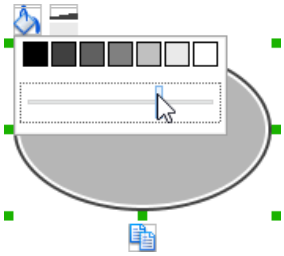


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

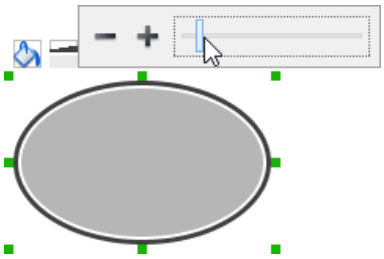
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

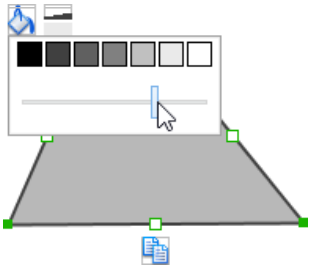
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

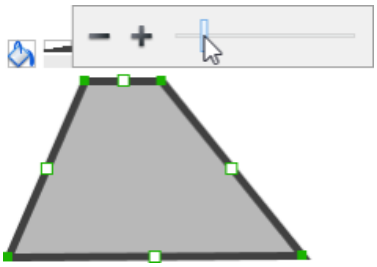
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

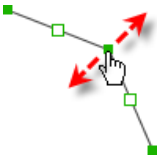


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



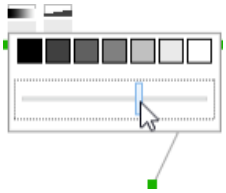
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

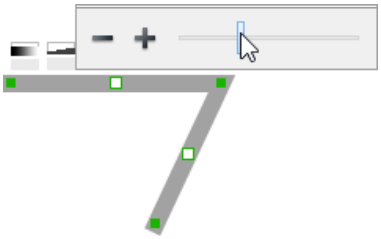
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Trademark Disclaimer

Android is a trademark of Google Inc.

Supported Editions

- Professional Edition
- Enterprise Edition

Desktop wireframing skills

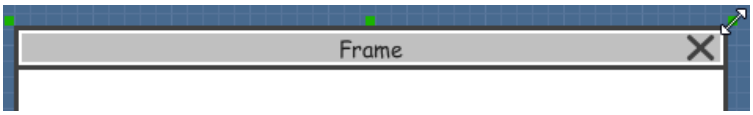
Adjust the size of frame

If you find the frame too large or too small, you can resize it to your preferred size. To adjust the size of frame:

1. Click on the frame's title bar.

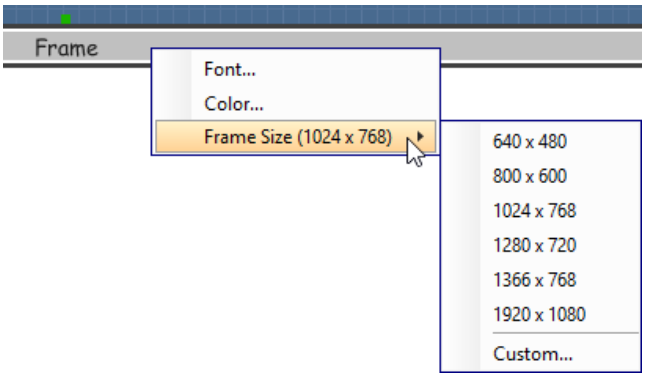
Clicking on frame's title bar

2. This shows the resize handlers at the corners and edges of the frame. You can drag on them to resize the frame.



Resizing frame

Alternatively, you can right click on the frame's title bar and select **Browser Size > %PREFERRED_SIZE%** from the popup menu.

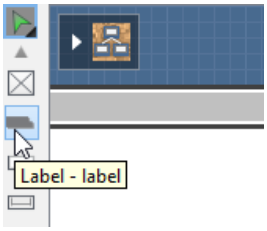


Adjusting browser window size

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).

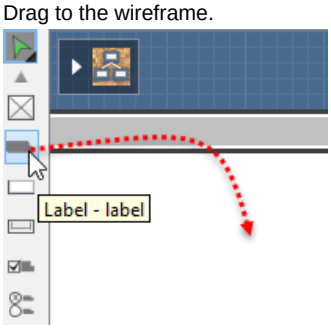


Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

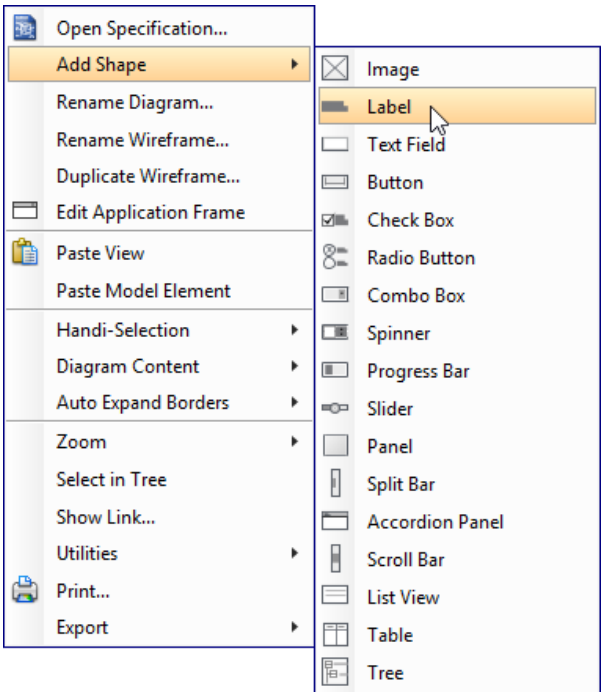


Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

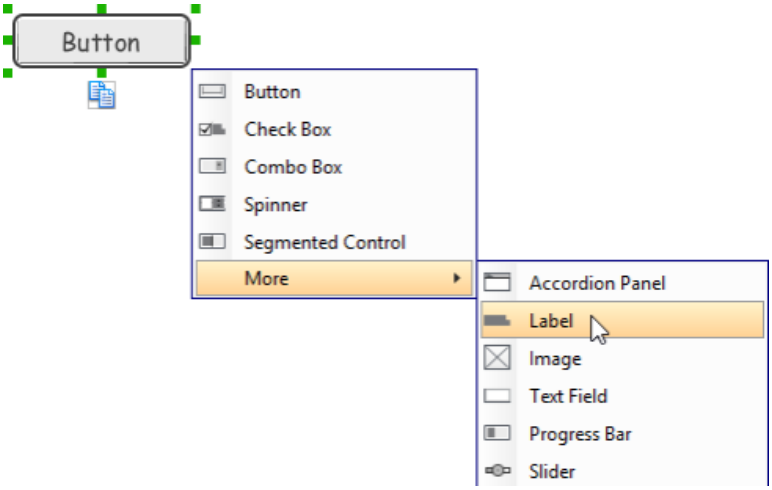
1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appear, known as the **Smart Create** resource.

Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.
3. Drag to outline the size of the wireframe element to be created.

Creating a label in specific size

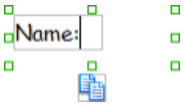
4. Release the mouse button. In the popup menu, choose the type of wireframe element to be created.



Choosing the wireframe element to be created

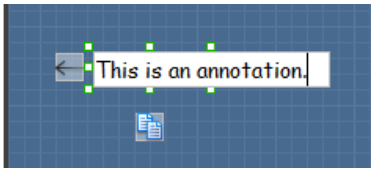
Method 5 - Double-clicking (Label and annotation only)

To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the frame and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

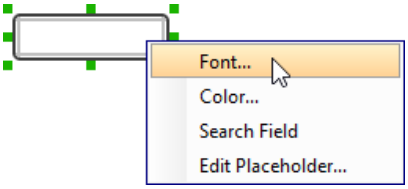
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you to align elements perfectly with others. Simple select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjusting the positioning of selection with the help of the guide.

Using the alignment guide

Adjusting font property for wireframe elements

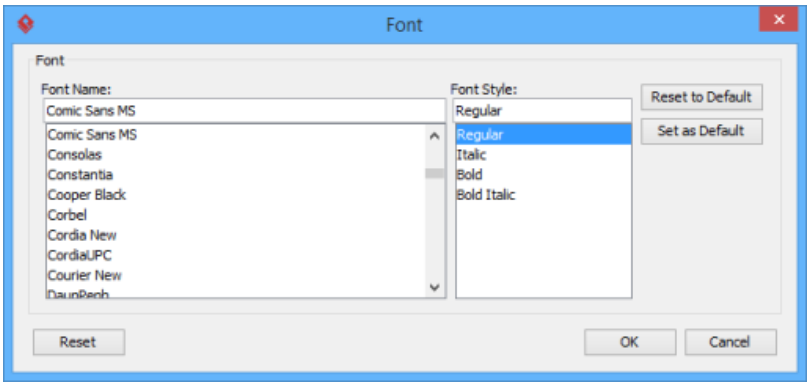
For wireframe elements that can display text, you can adjust their font properties like the font familiy and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

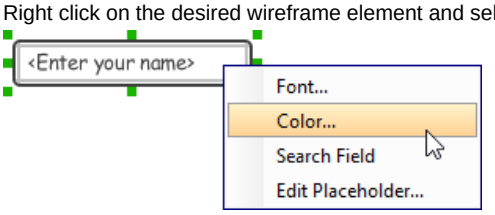


The Font window

Setting color for wireframe elements

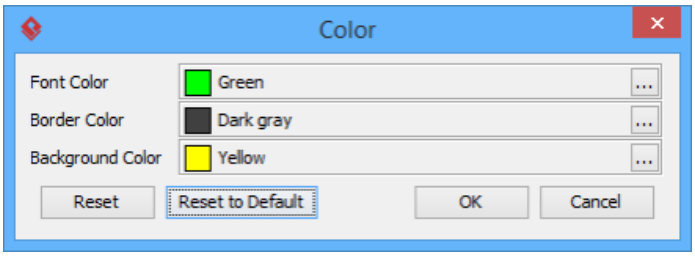
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, list view doesn't.



Editing color properties

You will then see the new color applied.

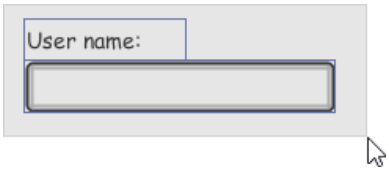
Color applied

Duplicating wireframe elements

Duplicate wireframe elements enables you to create new elements based on existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

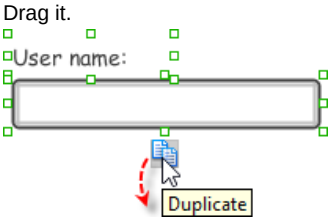
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to duplicate.



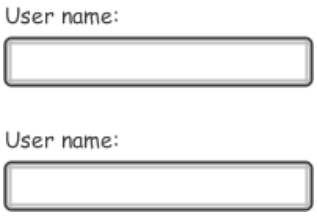
Selecting wireframe elements to duplicate

2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



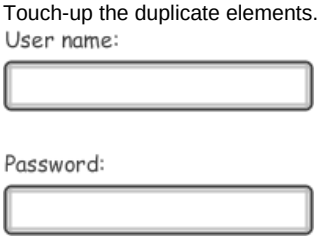
To duplicate wireframe elements

4. Release the mouse button at the position where you want the wireframe element to be created.



Wireframe elements duplicate

5. Touch-up the duplicate elements.



Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may also click on it..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

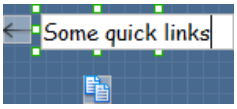
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the frame. In other words, you cannot create or move an annotation to inside the frame area.

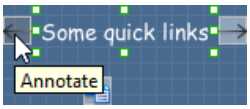
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



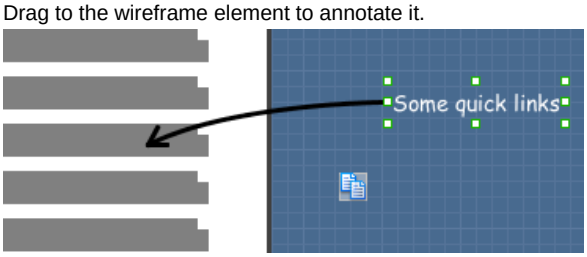
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

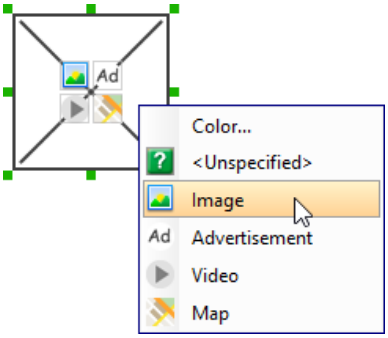
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video or map. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to embed into the image component.



To embed image into image component

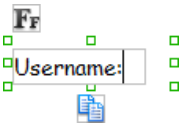
To represent an advertisement, video or map, right click on the image component and select **Advertisement**, **Video** and **Map** respectively.

Image component showed as video

Wireframing tips - Label

Specifying the content of label

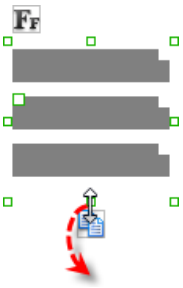
To specify the content of label, double click on the label and enter the content. You can press **Enter** to create a new line or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

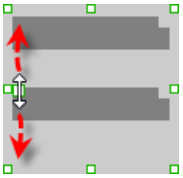
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.

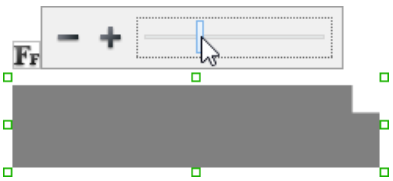


Adjusting the spacing between labels

NOTE: When a label has specified, you cannot show multiple labels in it.

Adjusting label or font size

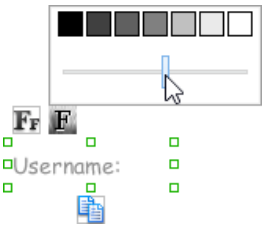
The size of label(s) or text in label, when content is filled, can be changed. To adjust font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.



Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

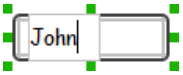


Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.



Specifying the content of text field

Showing the text field as a search field

Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Search Field** from the popup menu.

A search field

Editing the placeholder text

Placeholder text is the text that appears in the background of a text field. Very often, placeholder text is used to provide hints for user. For example, a text field of user name may have *<please enter your name here>* as placeholder text. Note that the placeholder text is only active when no content has been specified for the text box. To edit placeholder text of a text field, right click on the text field component and select **Edit Placeholder...** from the popup menu. Then, enter the placeholder text in the popup dialog box.

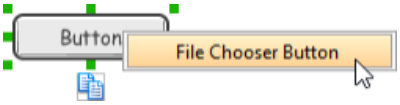
Text field with placeholder text entered

Wireframing tips - Button

Setting the type of button

There are two kinds of button - general button and file chooser button. General button is what you commonly see in any user interface. You click on a general button to do something as description by the button caption. File chooser button is a special kind of button that allows users to provide a file by clicking on the button. A file chooser button is followed by the text "No file chosen".

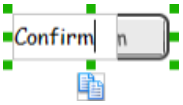
When you create a button, it is a general button by default. To change make it a file chooser button, right click on the button and select **File Chooser Button** from the popup menu.



Changing a button to a file chooser button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered. Note that the caption of a file chooser button is not editable.

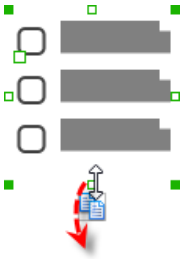


Entering button caption

Wireframing tips - Checkbox

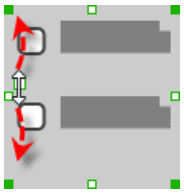
Creating the checkboxes

The checkbox component is in fact a placeholder of checkboxes. You can show multiple checkboxes in it by increasing the height of the checkbox component.



Creating more checkboxes

To adjust the spacing between checkboxes in a checkbox component, select the checkbox and drag the handler between the first and the second checkbox. Space will be added by dragging downwards.



Adjusting the spacing between checkboxes

Specifying the value of checkbox

To specify the value of a checkbox, double click on the label attached with the checkbox and enter the value. You may need to resize the checkbox afterwards in order to see the content entered.

Entering the value of a checkbox

Checking a checkbox

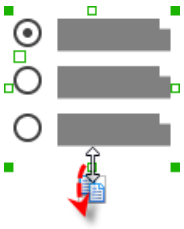
To check a checkbox, simply click on the checkbox. You can uncheck it by clicking again.

Checking checkboxes

Wireframing tips - Radio Button

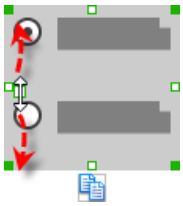
Creating the radio buttons

The radio button component is in fact a radio button group. You can show multiple radio buttons in it by increasing the height of the radio button component.



Creating more radio buttons

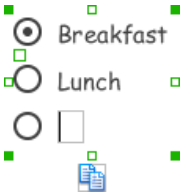
To adjust the spacing between radio buttons in a radio button component, select the radio button and drag the handler between the first and the second radio button. Space will be added by dragging downwards.



Adjusting the spacing between radio buttons

Specifying the value of radio button

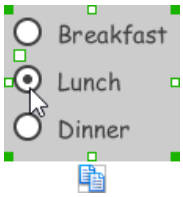
To specify the value of a radio button, double click on the label attached with the radio button and enter the value. You may need to resize the radio button afterwards in order to see the content entered.



Specifying the value of radio button

Selecting a radio button

To select a radio button, simply click on the radio button. You can uncheck it by clicking again.

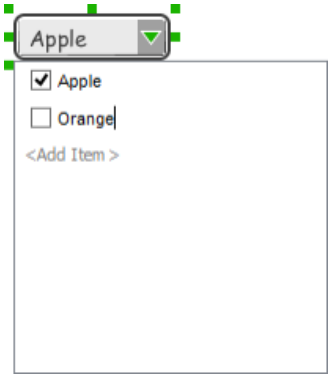


Selecting a radio button

Wireframing tips - Combo Box

Specifying the items in a combo box

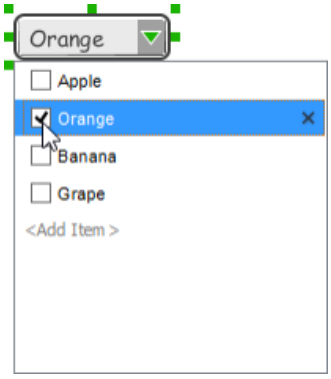
By default, a spinner has no items specified. You can add a list of items into a spinner by clicking on the down arrow on the right of the spinner and then click on **<Add Item>** and start entering the item.



Adding an item to combo box

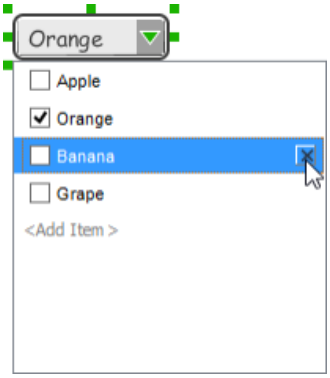
Changing the selected item in a combo box

To change the selected item of a combo box, click on the down arrow on the right of the combo box, then check the item to select it.



Selecting an item in combo box

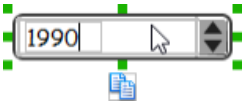
Removing an items from a combo box
To remove an item from a combo box, click on the down arrow on the right of the combo box, then select the item to remove and click on the cross button on the right to remove it.



Removing an item from combo box

Wireframing tips - Spinner

Specifying the value in spinner
To edit the value of a spinner, double click on the spinner and enter the value. You may need to resize the spinner afterwards in order to see the value entered.



Specifying the value of spinner

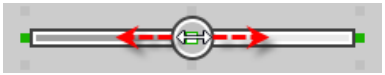
Wireframing tips - Progress Bar

Adjusting the progress
To adjust progress, select the progress bar first. Then drag the handler in the middle towards left or right to control the progress.

Adjusting the progress of progress bar

Wireframing tips - Slider

Adjusting the slider position
To adjust silder position, select the silder first. Then drag the handler in the middle towards left or right to control the position.



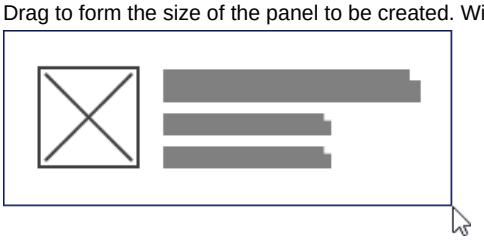
Adjusting the slider position

Wireframing tips - Panel

Containing existing components with panel
Panel is a useful wireframe component that helps you visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

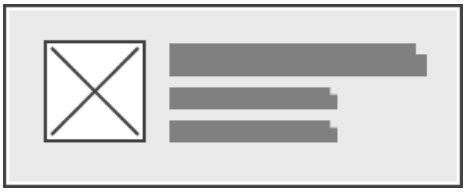
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

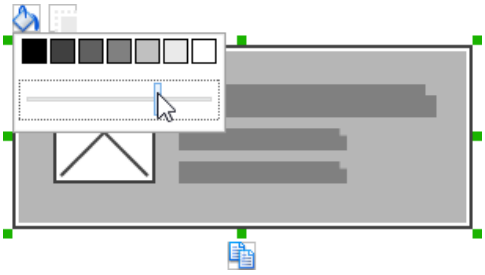
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

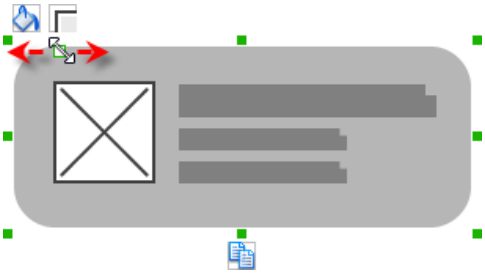
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.



Making the corner of panel rounded

Changing a panel to a titled pane or tabbed pane

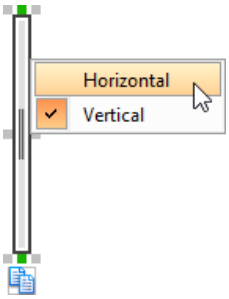
A panel can be presented as a titled pane or tabbed pane. To do this, right click on the panel and select **Type > Titled Pane** or **Type > Tabbed Pane** from the popup menu.

When presented as titled pane or tabbed pane, you can edit the caption of the pane or tabs by double clicking on captions.

Wireframing tips - Split Bar

Changing the orientation

To change the orientation of a split bar, right click on it and select either **Horizontal** or **Vertical** from the popup menu.

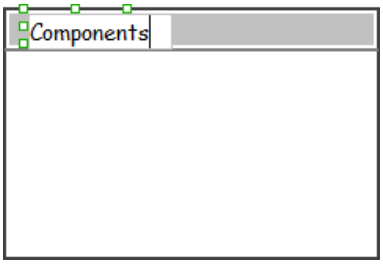


Changing the orientation of split bar

Wireframing tips - Accordion Panel

Editing the header

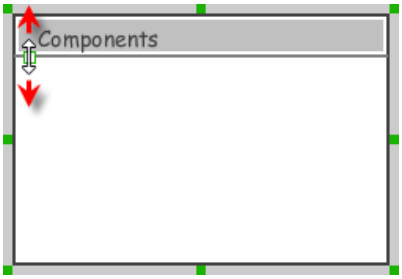
To edit table header, double click on the table header and enter the content. You may need to resize the header in order to see the text entered.



Editing the header

Adjusting header height

To adjust header height, drag on the handler below the header.

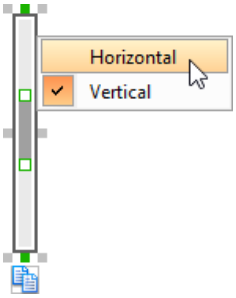


Adjusting the height of header

Wireframing tips - Scroll Bar

Changing the orientation

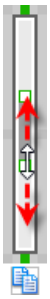
To change the orientation of a scroll bar, right click on it and select either **Horizontal** or **Vertical** from the popup menu.



Changing the orientation of scroll bar

Adjusting the scroll thumb (i.e. scrollbar slider)

To change the length of the thumb, select the scrollbar first. Then, drag on the handler attached to the ends of the thumb to resize it.



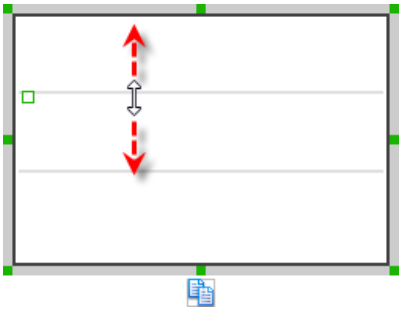
Changing the length of scroll thumb

To adjust the position of thumb, select the scrollbar first. Then, drag on the thumb along the track to reposition it.

Changing the position of scroll thumb

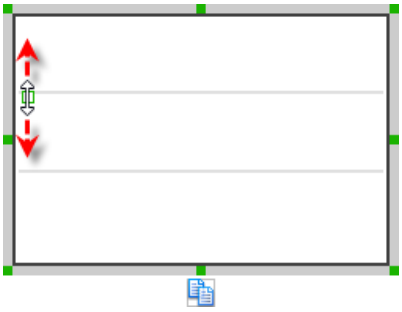
Wireframing tips - List View

Adjusting row height (for single row)
To adjust the height of a row, drag directly on the row separator under the row. By doing so, the row will be expanded or contracted.



Adjusting row height of a list view

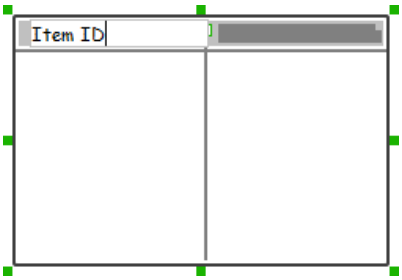
Adjusting rows height (for all rows)
To adjust the height of all rows in a list view, drag on the handler attached to the row separator between the first and the second row to resize all rows at the same time.



Adjusting rows' height of a list view

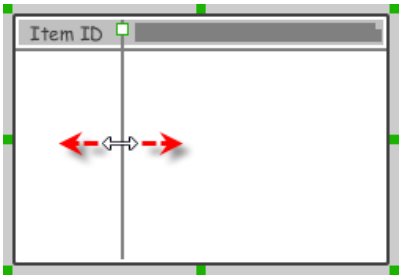
Wireframing tips - Table

Editing table header
To edit table header, double click on the table header and enter the content.



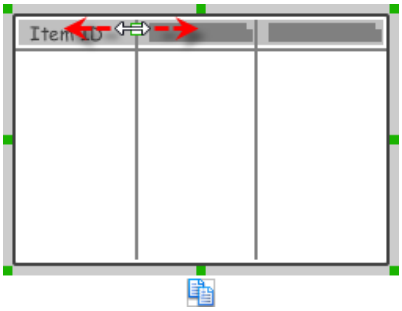
Editing table header

Adjusting column width
To adjust column width, drag directly on the column separator nearby. By doing so, the adjacent columns will be updated in their width.



Adjusting column width of a table

Adding more columns
To add more columns to a table, select the table first. Then, drag on the handler attached to the column separator between the first and the second column to create more columns.



Adding more columns to a table

Adding more rows
When you put a wireframe component (e.g. a label component) into a table component, a new row will be created automatically.

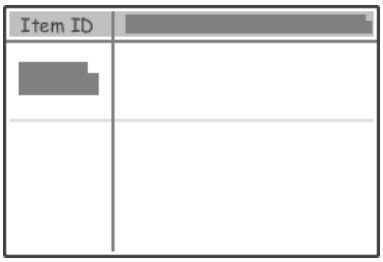
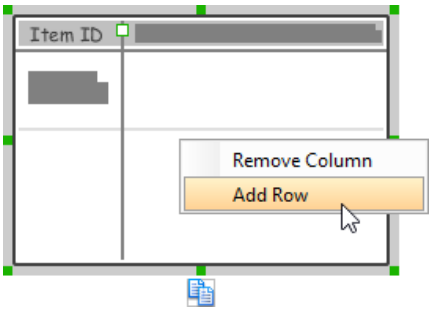


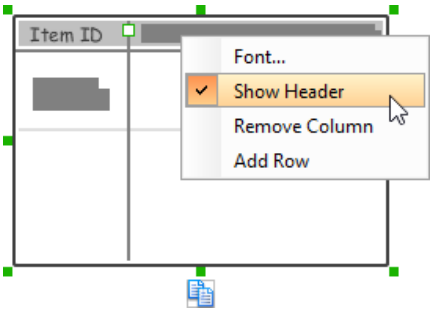
Table with one row

You may also add a row manually by right clicking on the table and selecting **Add Row** from the popup menu.



Adding row to a table

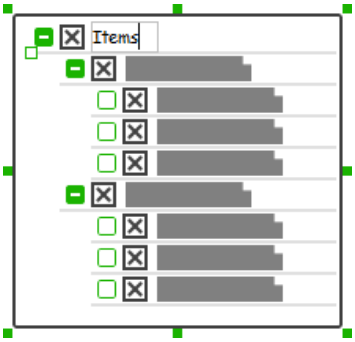
Hiding table header
To hide away the table header, right click on the header and uncheck Show Header from the popup menu.



Hiding table header

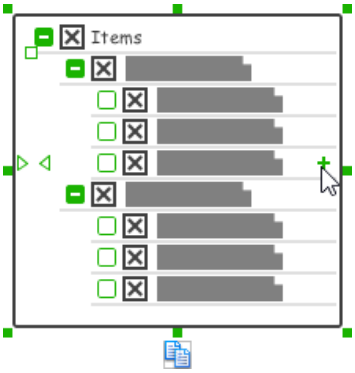
Wireframing tips - Tree

Editing the name of a tree node
To edit the name of a tree node, double click on the tree node and enter the name.



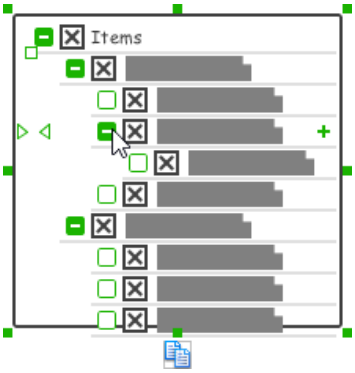
Editing the name

Adding an adjacent tree node
To add a tree node next to an existing one, select the tree first. Then, move the mouse pointer over the existing tree node where the new node will be created next to it. Click on the + button on the right of the tree node.



Creating a tree node

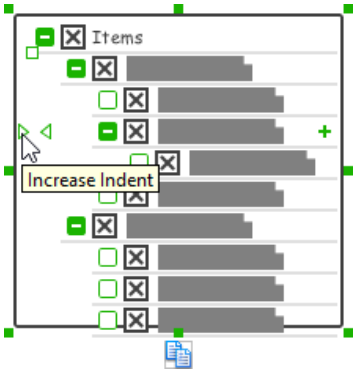
Adding a child tree node
To add a tree node under an existing one as child node, select the tree first. Then, move the mouse pointer over the existing tree node where the new node will be created under it. Click on the empty box on the right of the tree node. The box will then become a solid box with + in it. Click again, a child node will be created under it.



Creating a child tree node

Adjusting the level of indentation of tree node

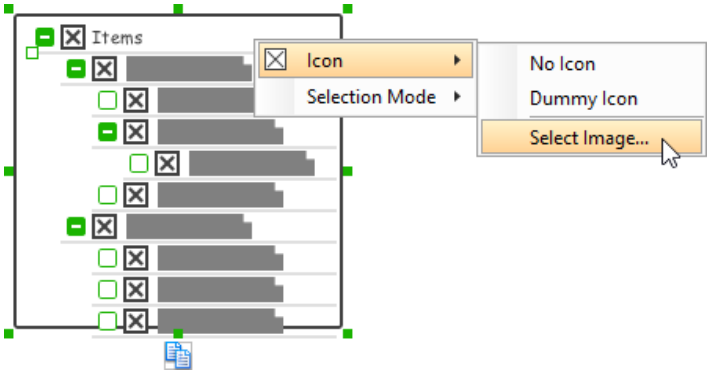
To adjust the level of indentation of a tree node, select the tree first. Then, move the mouse pointer over the existing tree node where you want to adjust its level of indentation. Click on the < or > on the left of the tree node to make the node one level backward or forward.



Adjusting the level of indentation of a node

Specifying icon for ALL tree nodes

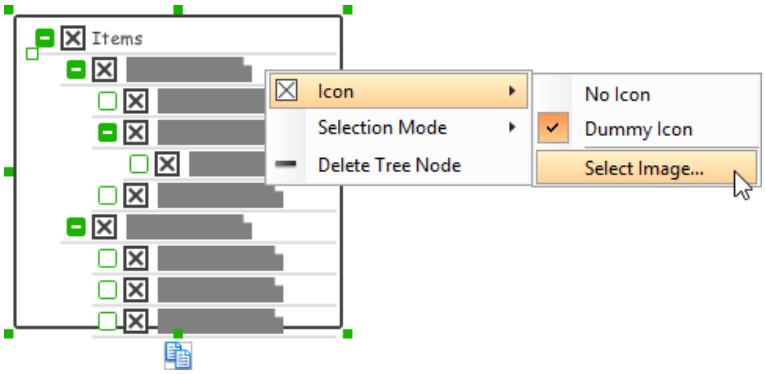
You can optionally specify an image icon for all of your tree nodes. To specify an icon, de-select the tree first. Then, right click on the tree component and select **Icon > Select Image...** from the popup menu. If you want to have no icons for all tree nodes, select **No Icon**. If you want an icon but do not want to specify its content, choose **Dummy Icon**. Make sure you are not clicking on a specific tree node or else only that node will be updated.



Specifying an icon for all tree nodes

Specifying icon for a specific tree node

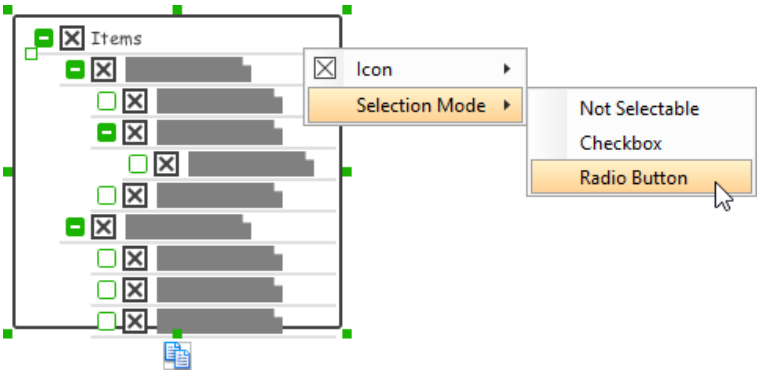
You can optionally specify an image icon for a specific tree node. To specify an icon, select the tree first. Then, right click on the tree node to specify icon and select **Icon > Select Image...** from the popup menu. If you want a node without icon, select **No Icon**. If you want an icon but do not want to specify its content, choose **Dummy Icon**.



Specifying an icon for tree node

Adjusting the selection mode for ALL tree nodes

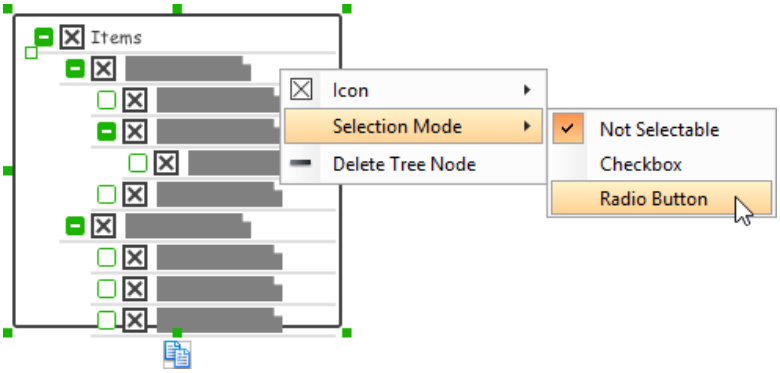
There are three selection modes for a menu item - Not Selectable, Chekcbox and Radio Button. To adjust the selection mode of all tree nodes, de-select the tree first. Then, right click on the tree component and select **Selection Mode > [MODE]** from the popup menu. Make sure you are not clicking on a specific tree node or else only that node will be updated.



Adjusting the selection mode for all tree node

Adjusting the selection mode for a specific tree node

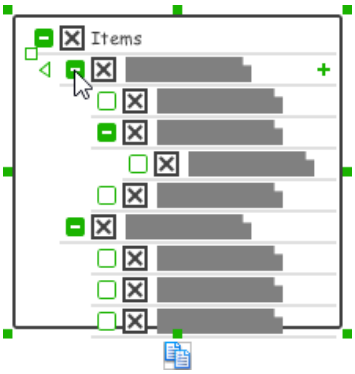
There are three selection modes for a menu item - Not Selectable, Chekcbox and Radio Button. To adjust the selection mode of a tree node, select the tree first. Then, right click on the tree node and select **Selection Mode > [MODE]** from the popup menu.



Adjusting the selection mode for a specific tree node

Expanding and collapsing a tree node

To expand or collapse a tree node, click on the + or - button on the left of a tree node.

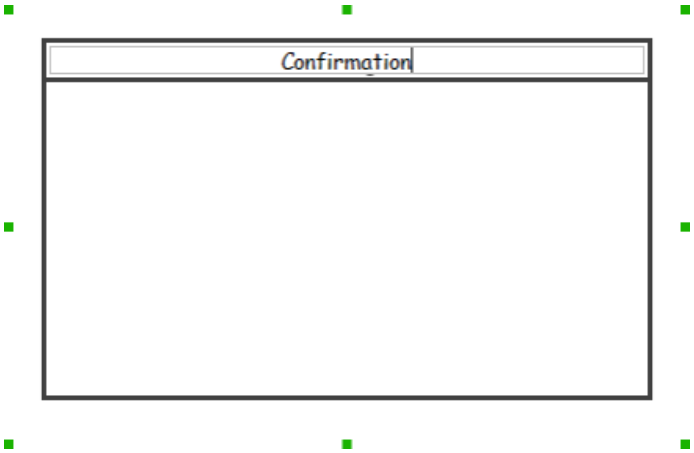


Tree node expanded

Wireframing tips - Dialog

Editing the title

To edit title of dialog, double click at the top of the dialog enter the content.



Editing the dialog title

Wireframing tips - Toolbar

Representing buttons in a toolbar

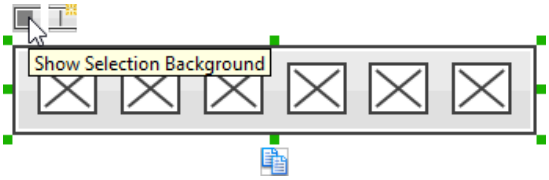
By default, a toolbar is just an empty horizontal bar. To represent the buttons in toolbar, you can either create buttons in it or represent with an image component, like this:

Toolbar with buttons

Representing a button being selected

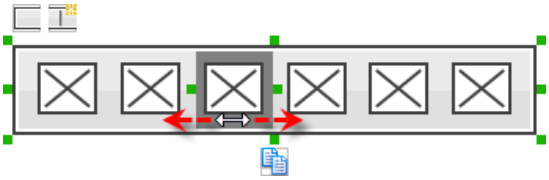
To represent a button being selected:

1. Click on the toolbar first.
2. Click on the resource **Show Selection Background**.



To show selection background

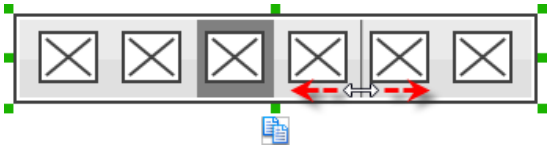
3. Drag on the selection background to the suitable place to indicate the active selection of button.



Reposition the selection background

Adding separator

To add a separator, click on the toolbar first. Then, click on the resource **Add Separator**. You can then drag the separator to reposition it.



Repositioning toolbar separator

Wireframing tips - Menu Bar

Editing the name of a menu

To edit the name of a menu in menu bar, double click on the menu and enter the name.

Editing the name

Adding more menus
To add more menus to a menu bar, select the menu bar first. Then, click on **click to add...** and enter the name of the new menu.

Adding more menus to a menu bar

Rearranging menus
To rearrange menus, select the menu bar first. Then, drag on a menu and move it along the menu bar to reposition it.

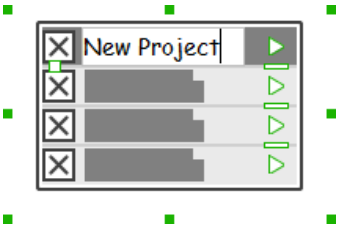
Rearranging menus in a menu bar

Specifying the selected menu
To represent that a menu has been selected, specify the selected menu by selecting the menu bar first. Then, click directly on the menu to make it appear selected.

Specifying the selected menu

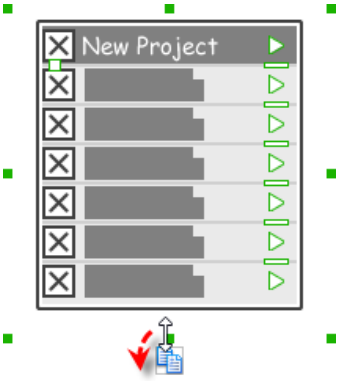
Wireframing tips - Menu

Editing the name of a menu item
To edit the name of a menu item in menu, double click on the menu item and enter the name.



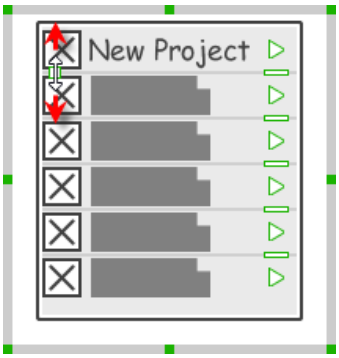
Editing the name

Adding more menu items
You can add more menu items to a menu by increasing the height of the menu component.



Creating more menu items

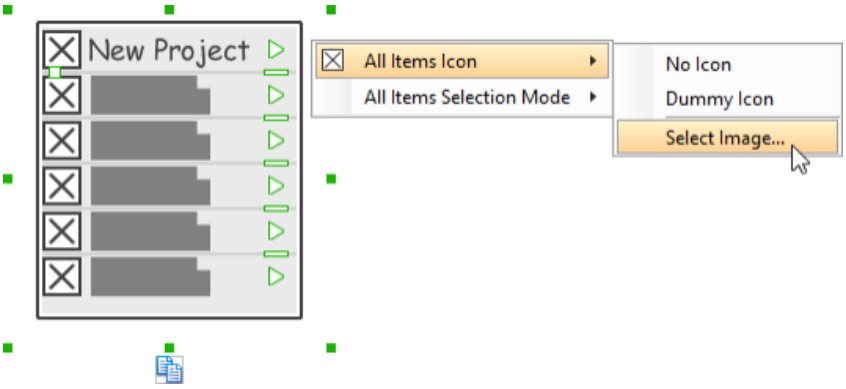
Adjusting label or font size
The size of label(s) or menu items, when name is specified, can be changed. To adjust font, drag the slider between the first and the second menu item to adjust the font size.



Adjusting font size of menu items

Specifying icon for ALL menu items

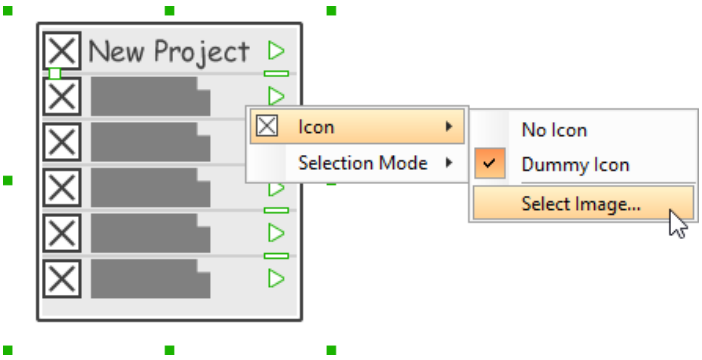
You can optionally specify an image icon for all of your menu items. To specify an icon, de-select the menu first. Then, right click on the menu component and select **Icon > Select Image...** from the popup menu. If you want all menu items to have no icon, select **No Icon**. If you want an icon but do not want to specify its content, choose **Dummy Icon**. Make sure you are not clicking on a specific menu item or else only that node will be updated.



Specifying an icon for all menu items

Specifying icon for a specific menu item

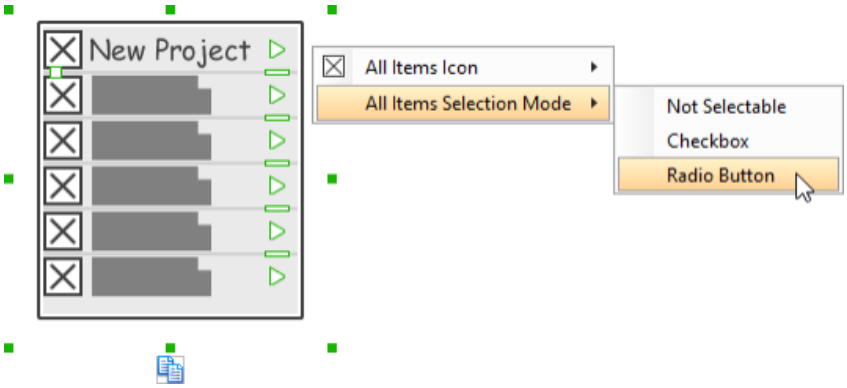
You can optionally specify an image icon for a specific menu item. To specify an icon, select the menu first. Then, right click on the menu item to specify icon and select **Icon > Select Image...** from the popup menu. If you want a menu item without icon, select **No Icon**. If you want an icon but do not want to specify its content, choose **Dummy Icon**.



Specifying an icon for menu item

Adjusting the selection mode for ALL menu items

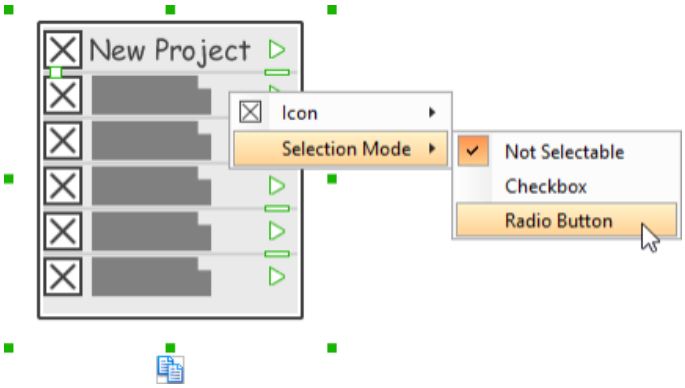
There are three selection modes for a menu item - Not Selectable, Chekcbox and Radio Button. To adjust the selection mode of all menu items, de-select the menu first. Then, right click on the menu component and select **Selection Mode > [MODE]** from the popup menu. Make sure you are not clicking on a specific menu item or else only that node will be updated.



Adjusting the selection mode for all menu items

Adjusting the selection mode for a specific menu item

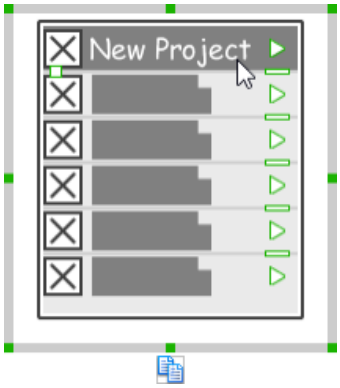
There are three selection modes for a menu item - Not Selectable, Chekcbox and Radio Button. To adjust the selection mode of a menu item, select the menu first. Then, right click on the menu item and select **Selection Mode > [MODE]** from the popup menu.



Adjusting the selection mode for a specific menu item

Specifying the selected menu item

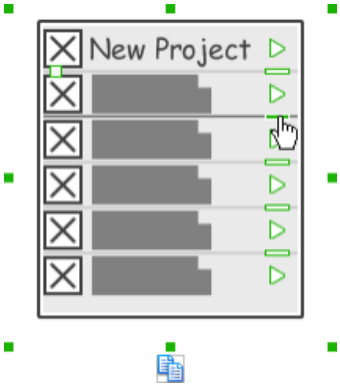
To represent that a menu item has been selected, specify the selected menu item by selecting the menu first. Then, click directly on the menu item to make it appear selected.



Specifying the selected menu item

Adding separator to menu

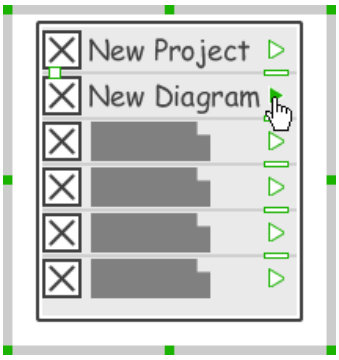
To add a separator, select the menu first. You will see a number of short horizontal line on the right of the menu. You can then click on them to show the separators.



Adding separator to menu

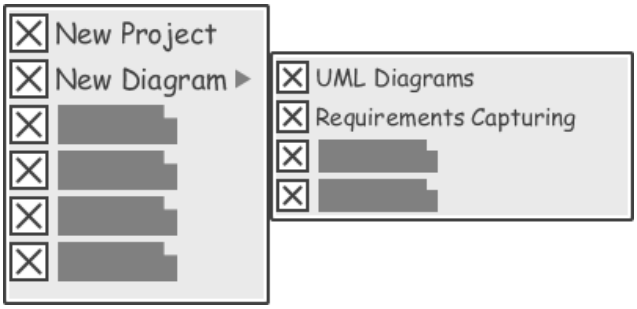
Representing nested menu structure
To represent a nested menu structure:

1. Click on the menu.
2. Click on the triangle button on the right of the menu item where the sub-menu pops out.



Representing the availability of sub-menu

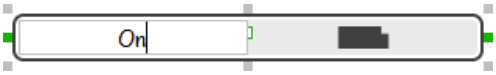
3. Create another menu next to it.



Menu and sub-menu

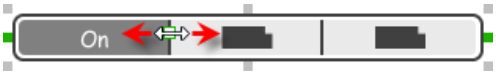
Wireframing tips - Segmented Control

Editing the title of a segment
To edit the title of a segment, double click on the segment and enter the title.



Editing the title

Adding more segments
To add more segments to a segmented control, select the segmented control first. Then, drag on the handler attached to the segment separator between segments to create more segments.



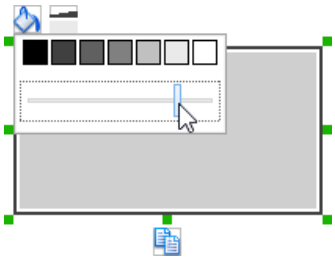
Adding more segments to a segmented control

Changing the selected segment
To change the selected segment, select the segmented control first. Then, click on the segment directly in the segmented control.

Changing the selected segment in a segmented control

Wireframing tips - Rectangle

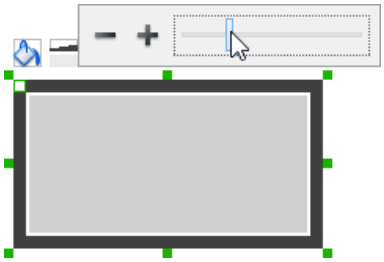
Adjusting fill color
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

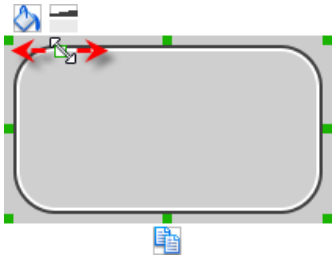
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.

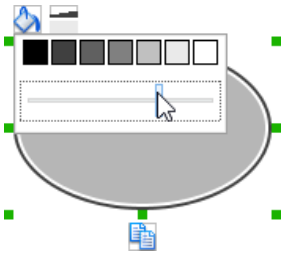


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

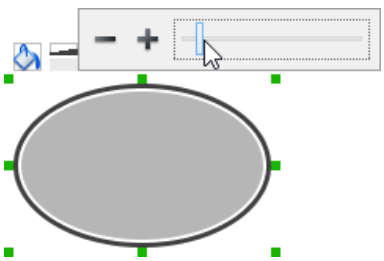
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

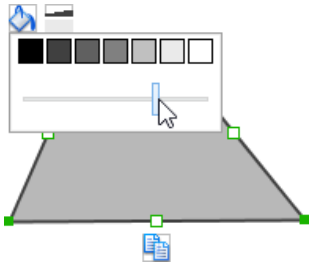
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

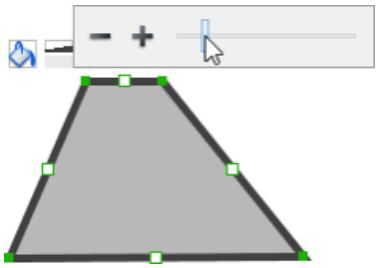
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

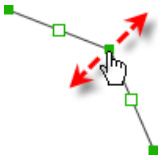


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



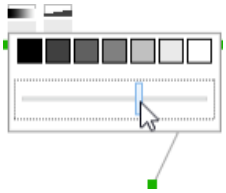
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

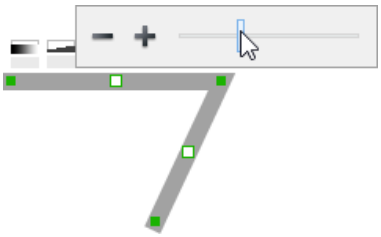
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

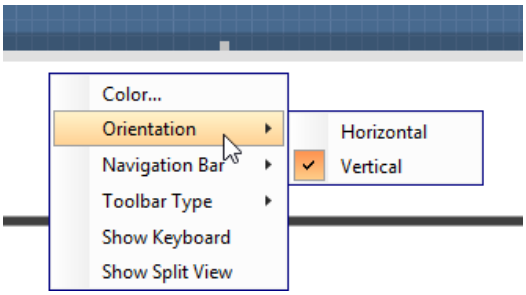
Supported Editions

- Professional Edition
- Enterprise Edition

iPad wireframing skills

Changing the orientation of iPad

Initially, iPad is shown horizontally in the wireframe. If your apps works under a vertical layout, you can change its orientation. To adjust the orientation, right click on the iPad border and select **Orientation > Horizontal/Vertical** from the popup menu.

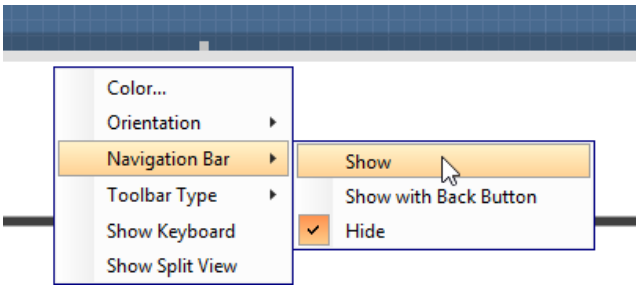


Changing orientation of iPad

NOTE: You can only change orientation when there is no wireframe element created inside iPad

Show/Hide navigation bar

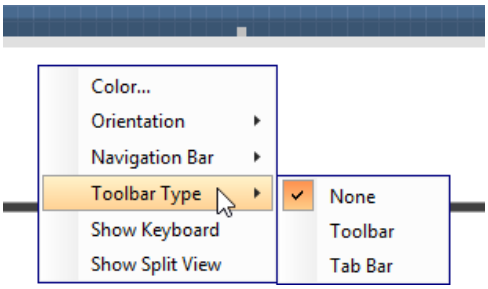
To hide the navigation bar, right click on the iPad border and select **Navigation Bar > Show** or **Navigation Bar > Show with Back Button** from the popup menu.



Show navigation bar

Show/Hide toolbar and tab bar

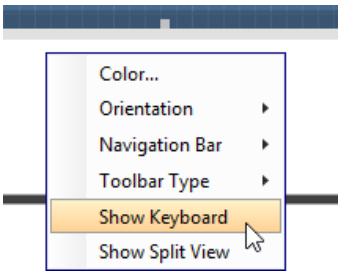
To hide the toolbar and tab toolbar, right click on the iPad border and select **Toolbar Type > Toolbar** or **Toolbar Type > Tab Bar** from the popup menu.



Show toolbar bar

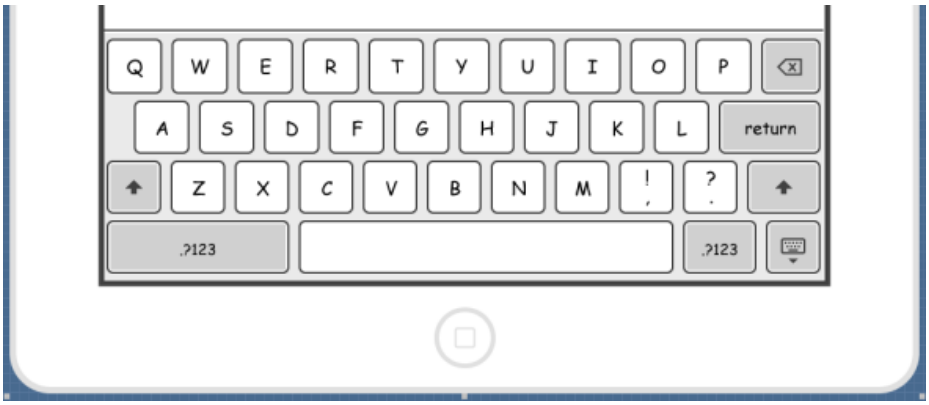
Show/Hide keyboard

To show the keyboard, right click on the iPad border and select **Show Keyboard** from the popup menu.



Show keyboard

This shows the keybaord at the bottom of iPad:

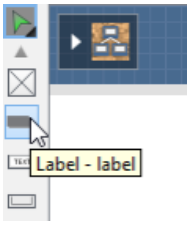


Keyboard shown

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).

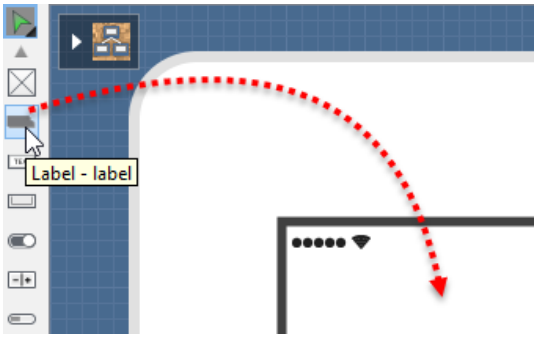


Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

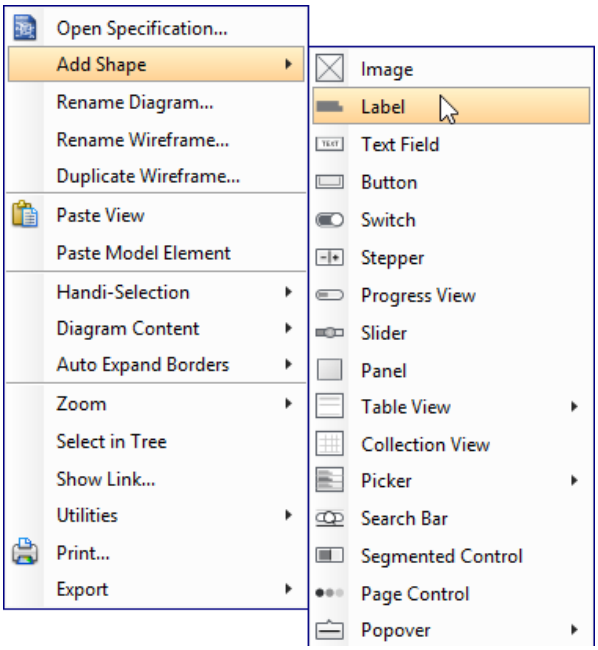


Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

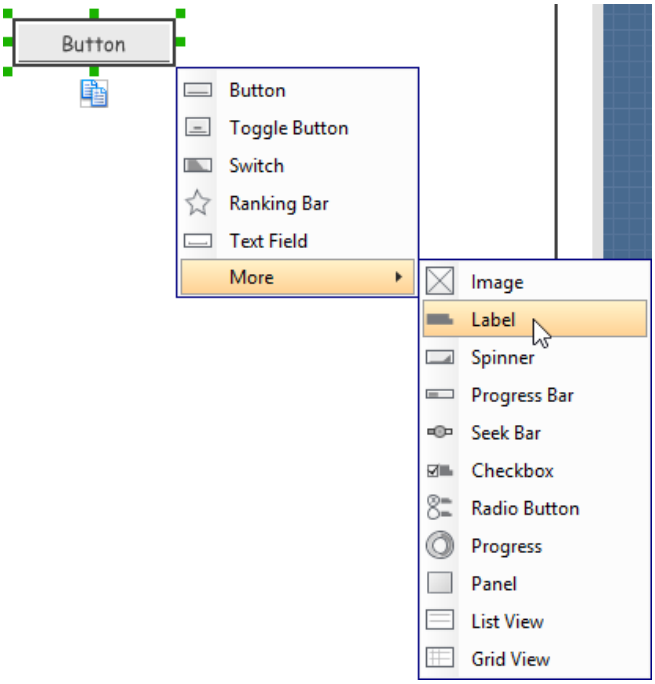
1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appear, known as the **Smart Create** resource.

Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.
3. Drag to outline the size of the wireframe element to be created.

Creating a label in specific size

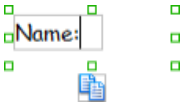
4. Release the mouse button. In the popup menu, choose the type of wireframe element to create.



Choosing the wireframe element to be created

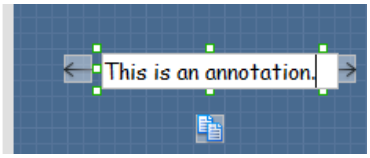
Method 5 - Double-clicking (Label and annotation only)

To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the iPad and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

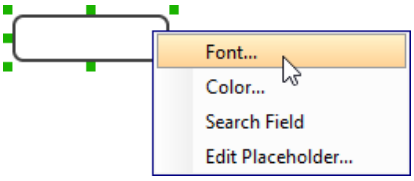
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you align elements perfectly with others. Simple select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjusting the positioning of selection with the help of the guide.

Using the alignment guide

Adjusting font property for wireframe elements

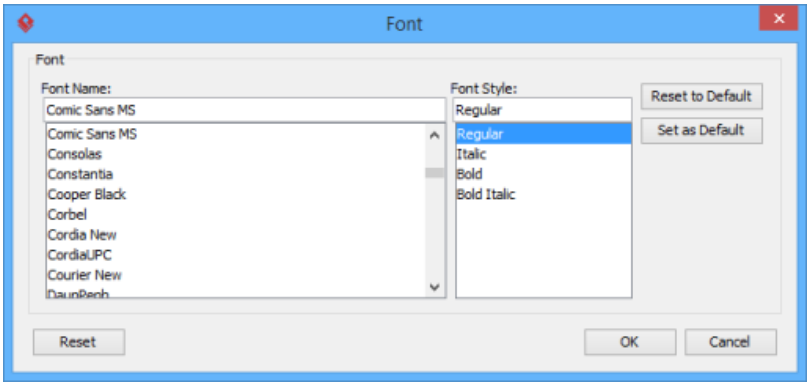
For wireframe elements that can display text, you can adjust their font properties like the font familiy to use and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

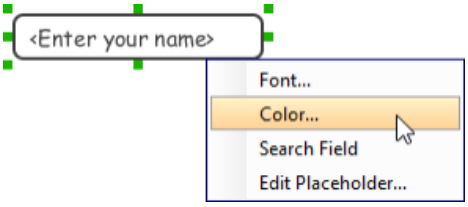


The Font window

Setting color for wireframe elements

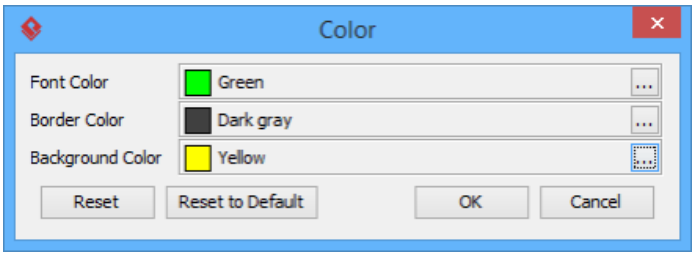
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, collection view doesn't.



Editing color properties

You will then see the new color applied.

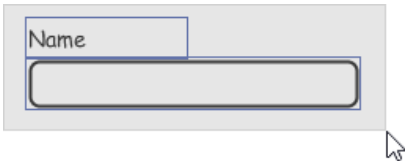
Color applied

Duplicating wireframe elements

Duplicate wireframe elements enables you to create new elements based on existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

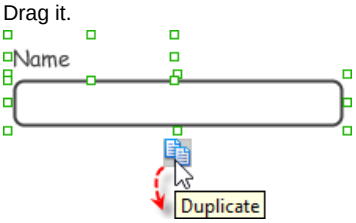
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to duplicated.



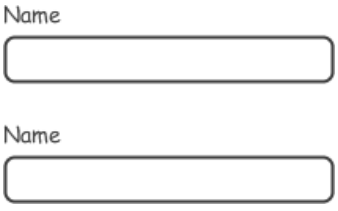
Selecting wireframe elements to duplicate

2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



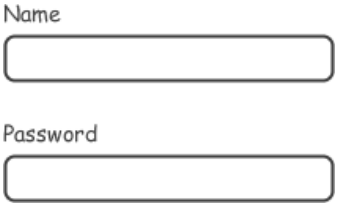
To duplicate wireframe elements

4. Release the mouse button at the position where you want the wireframe element to be created.



Wireframe elements duplicate

5. Touch-up the duplicate elements.



Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may click on it, too..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

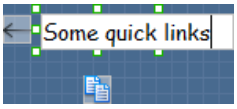
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the iPad. In other words, you cannot create or move an annotation to inside the iPad.

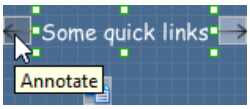
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



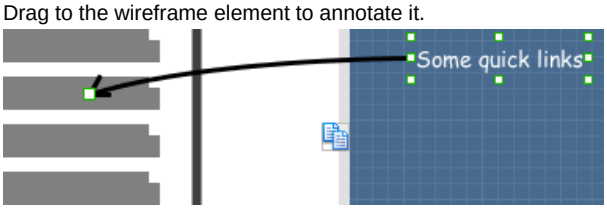
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

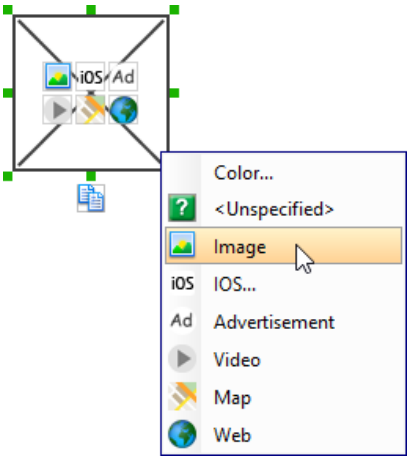
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video, map or web component. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to embed into the image component.



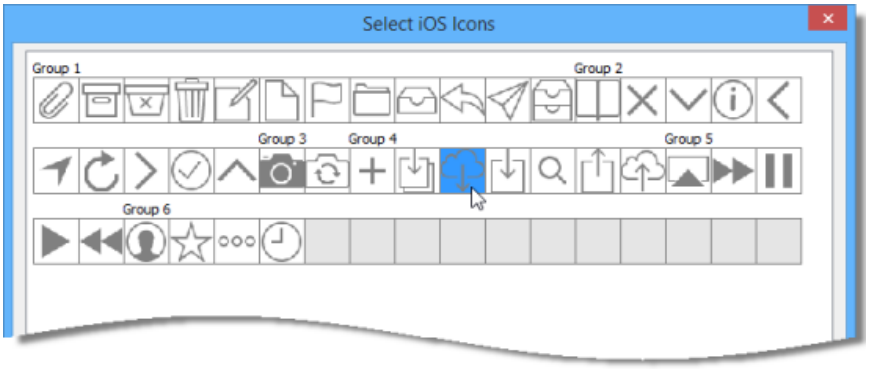
To embed image into image component

To represent an advertisement, video, map or web component, right click on the image component and select **Advertisement**, **Video**, **Map** and **Web** respectively.

Image component showed as video

Showing IOS icon

You can also use an image component to show an IOS icon for a tab bar. To show an IOS icon, right click on the image component and select **IOS...** from the popup menu. In the popup window, choose the icon to show and click **OK** to confirm.

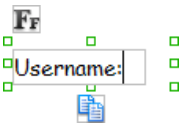


Choosing an IOS icon

Wireframing tips - Label

Specifying the content of label

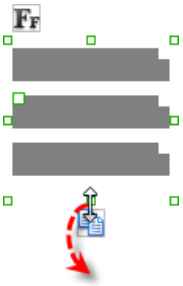
To specify the content of label, double click on the label and enter the content. You can press **Enter** to create a new line, or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

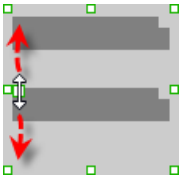
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.



Adjusting the spacing between labels

NOTE: When a label has content specified, you cannot show multiple labels in it.

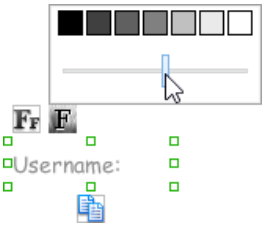
Adjusting label or font size

When the content is filled, the size of label(s) or text in label can be changed. To adjust the font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.

Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

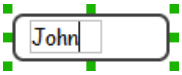


Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.



Specifying the content of text field

Showing the text field as a search field

Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Search Field** from the popup menu.

A search field

Editing the placeholder text

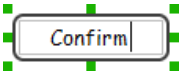
Placeholder text is the text that appears in the background of a text field. Very often, placeholder text is used to provide hints for user. For example, a text field of user name may have *<please enter your name here>* as placeholder text. Note that the placeholder text is only active when no content has been specified for the text box. To edit placeholder text of a text field, right click on the text field component and select **Edit Placeholder...** from the popup menu. Then, enter the placeholder text in the popup dialog box.

Text field with placeholder text entered

Wireframing tips - Button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.



Entering button caption

Wireframing tips - Switch

Altering the state of switch

To alter the state of a switch, select the switch first. Then, click on the inactive end of the switch to switch to that end.

Altering switch state

Wireframing tips - Progress View

Adjusting the progress

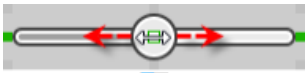
To adjust progress, select the progress view first. Then drag the handler in the middle towards left or right to control the progress.

Adjusting the progress of progress view

Wireframing tips - Slider

Adjusting the slider position

To adjust silder position, select the silder first. Then drag the handler in the middle towards left or right to control the position.



Adjusting the slider position

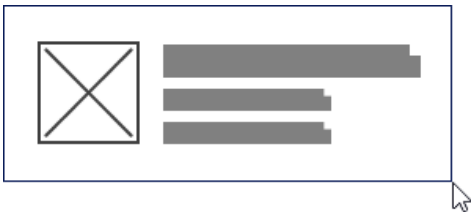
Wireframing tips - Panel

Containing existing components with panel

Panel is a useful wireframe component that helps you to visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

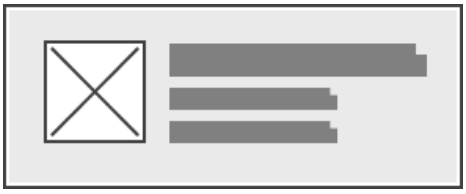
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

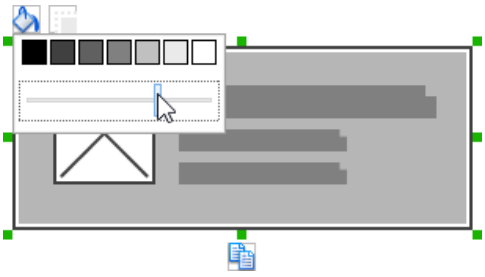
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

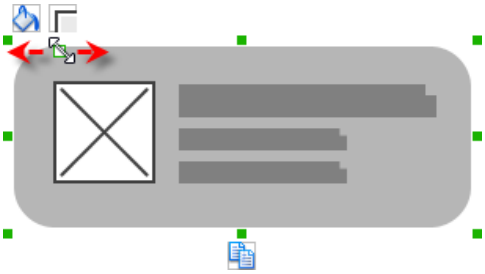
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.



Making the corner of panel rounded

Wireframing tips - Plain Table View

Editing the header

To edit the table header, double click on the table header and enter the content.

Editing the header

Adjusting row height

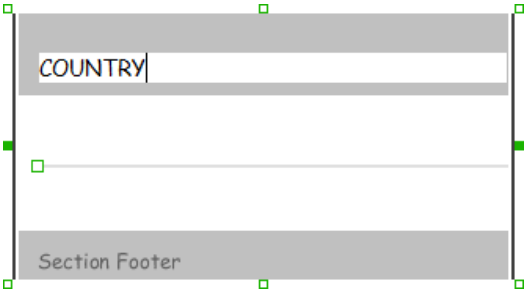
To adjust the row height, drag on the handler between the first and the second row.

Adjusting the row height

Wireframing tips - Grouped Table View

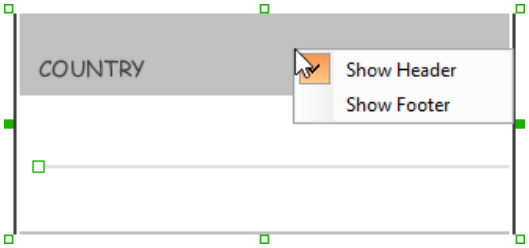
Editing the header/footer

To edit table header/footer, double click on the table header/footer and enter the content.



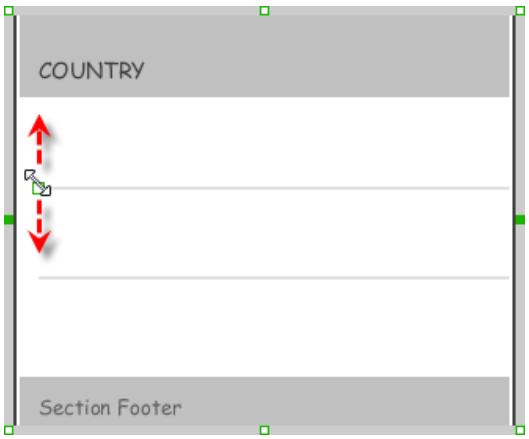
Editing the header

Showing/hideing the header/footer
By default, table header and footer are visible in a grouped table view. To hide header/footer, right click on the grouped table view and de-select **Show Header/Footer** from the popup menu.



Footer in grouped table view hidden

Adjusting row height
To adjust row height, drag on the handler between the first and the second row.



Adjusting the row height

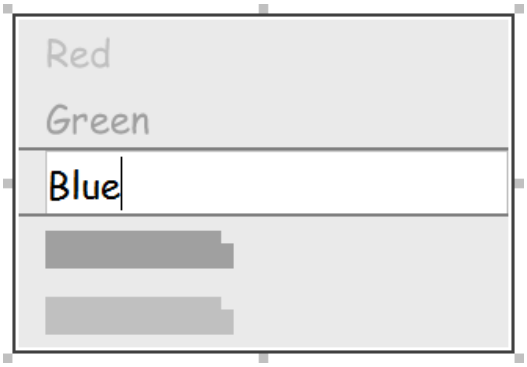
Wireframing tips - Collection View

Adjusting cell size
To adjust the size of cells, drag on the handler between the first and the second row.

Adjusting cell size

Wireframing tips - Picker View

Editing label in picker view
To edit the lable in a picker view, double click on the label and enter the content.

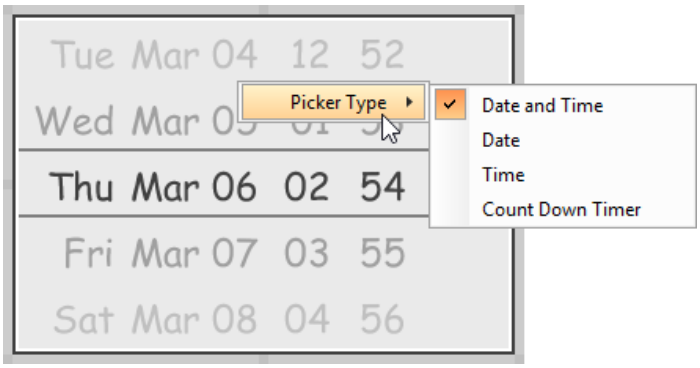


Editing a label in picker view

Wireframing tips - Date Picker View

Changing the picker type

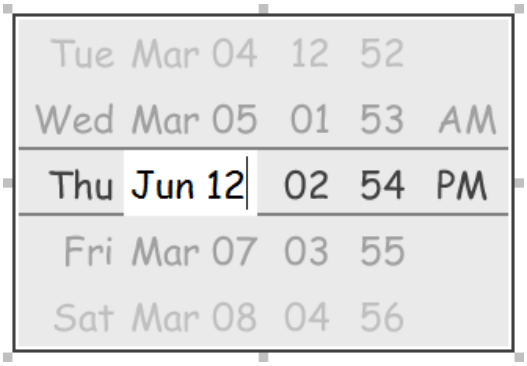
There are four kinds of date picker - Date-and-Time, Date, Time, Count Down Timer. To change the picker type, right click on the date picker and select **Picker Type > [TYPE]** from the popup menu.



Changing picker type

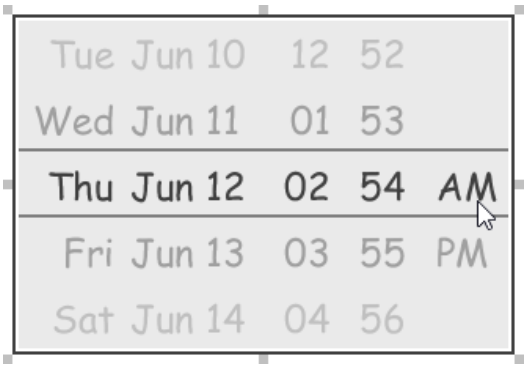
Editing the values of date/time

To edit the values in a date picker, like to edit the date in a Date-and-Time picker, double click on the field to edit and enter the new value.



Editing the date in a date picker

To switch between AM and PM, double click on AM/PM directly.



Changing PM to AM

Wireframing tips - Search Bar

Specifying the content of search field

To specify the content of the search field, double click on the text field and enter the content. You may need to resize the search bar afterwards in order to see the content entered.

Specifying the content of search field

Renaming the Cancel button

If you have specified the content of search field, the Cancel button will appear. To rename the Cancel button, double click on it and enter the new caption.

Renaming the Cancel button

Adjusting font color of Cancel button

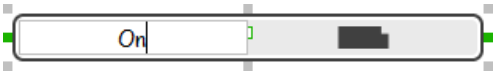
When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

Adjusting font color of the Cancel button

Wireframing tips - Segmented Control

Editing the title of a segment

To edit the title of a segment, double click on the segment and enter the title.



Editing the title

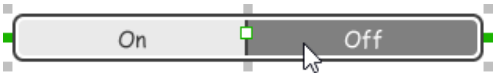
Adding more segments

To add more segments to a segmented control, select the segmented control first. Then, drag on the handler attached to the segment separator between segments to create more segments.

Adding more segments to a segmented control

Changing the selected segment

To change the selected segment, select the segmented control first. Then, click on the segment directly in a segmented control.



Changing the selected segment in a segmented control

Wireframing tips - Page Control

Adding more page indicators

To add more page indicators to a page control, select the page control first. Then, extend the page control to let more indicators appear.

Adding more page indicators to a page control

Setting the active page indicator

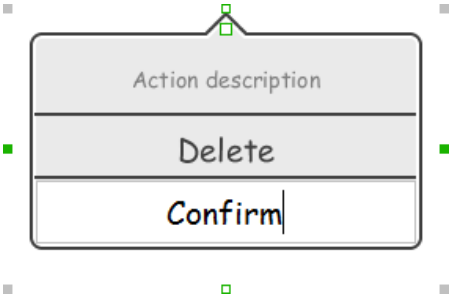
To set the active page indicator, select the page control first. Then, click on the page indicator directly in the page control.

Setting the active page indicator in a page control

Wireframing tips - Action Sheet

Editing the action description and action

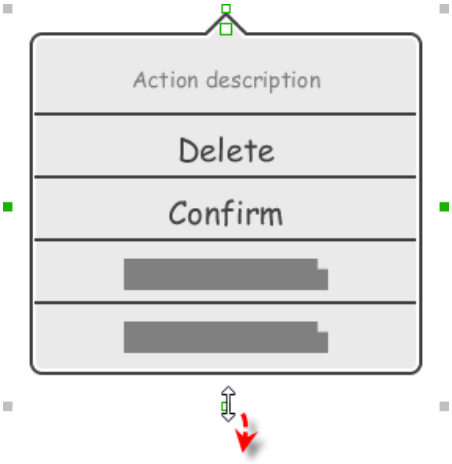
To edit action description or action, double click on the corresponding label and enter the content.



Editing the action

Showing more actions

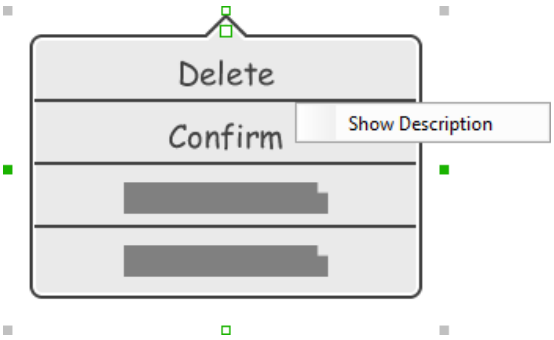
To show more actions, increase the height of the action sheet.



Resizing an action sheet to show more actions

Showing/hideing the action description

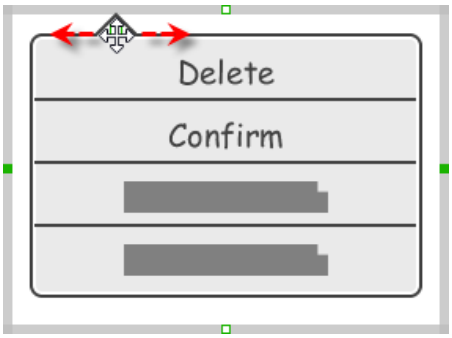
By default, the action description field is visible in an action sheet. To hide an action description, right click on the action sheet and de-select **Show Description** from the popup menu.



Action description in an action sheet is hidden

Adjusting the position of pointer

To adjust the position of pointer, select the action sheet first. Then, drag on the pointer to adjust its position. You can drag to all the four sides of the action sheet.



Adjusting the position of pointer in an action sheet

Wireframing tips - Popover

Adjusting the position of pointer

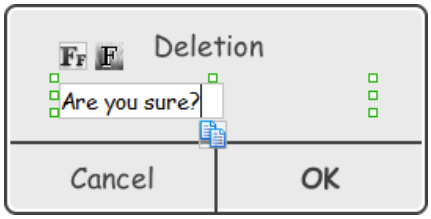
To adjust the position of pointer, select the popover first. Then, drag on the pointer to adjust its position. You can drag to all the four sides of the popover.

Adjusting the position of pointer in an popover

Wireframing tips - Alert View

Editing the content of labels and buttons in an alert view

To edit the content of labels and buttons in an alert view, double click on the label or button and enter the content. You may need to resize the label or the alert view afterwards in order to see the content entered.

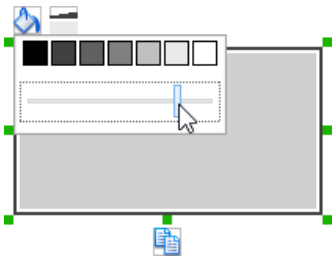


Editing the message in an alert view

Wireframing tips - Rectangle

Adjusting fill color

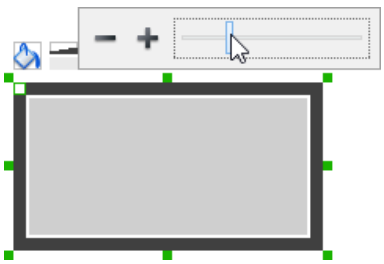
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

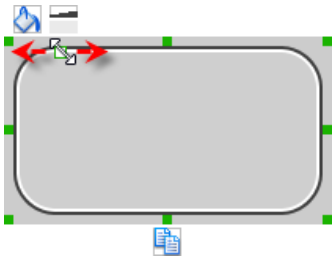
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler on top-left to adjust the size of the rounded corner. The four corners will be updated accordingly.

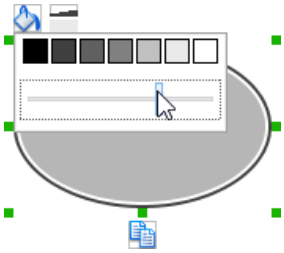


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

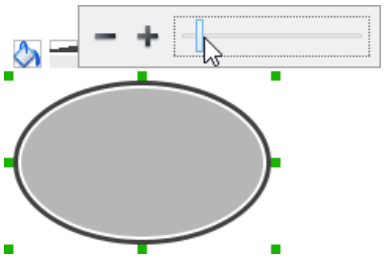
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

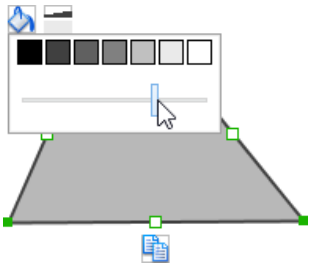
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

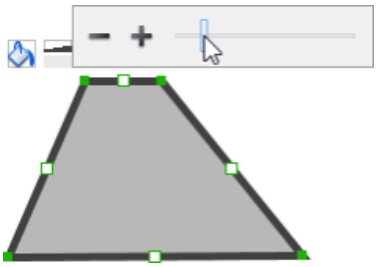
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

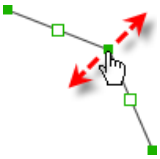


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



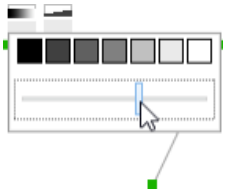
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

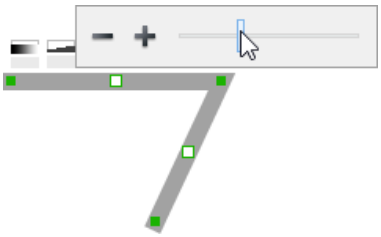
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Trademark Disclaimer

iPad is a trademark of Apple Inc.

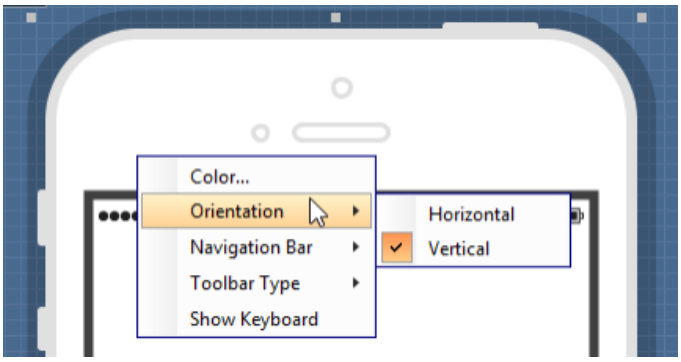
Supported Editions

- Professional Edition
- Enterprise Edition

iPhone wireframing skills

Changing the orientation of iPhone

Initially, iPhone is shown vertically in the wireframe. If your apps works under a horizontal layout, you can change its orientation. To adjust the orientation, right click on the phone border and select **Orientation > Horizontal/Vertical** from the popup menu.

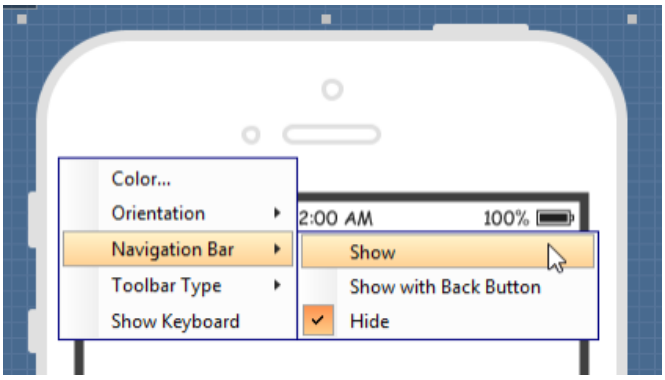


Changing orientation of iPhone

NOTE: You can only change orientation when there is no wireframe element created inside the phone body

Show/Hide navigation bar

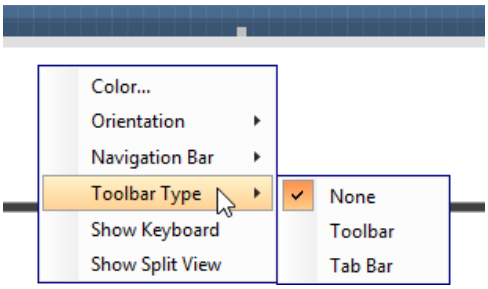
To hide the navigation bar, right click on the phone border and select **Navigation Bar > Show** or **Navigation Bar > Show with Back Button** from the popup menu.



Show navigation bar

Show/Hide toolbar and tab bar

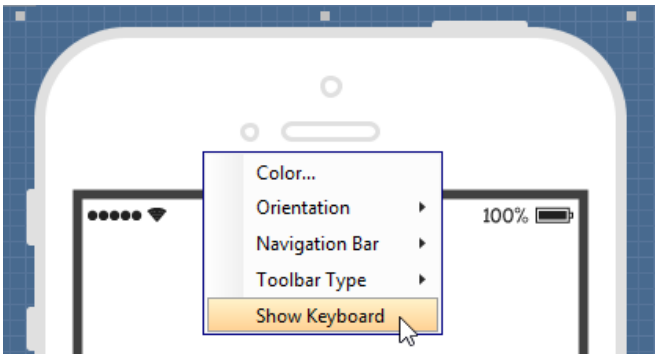
To hide the toolbar and tab toolbar, right click on the phone border and select **Toolbar Type > Toolbar** or **Toolbar Type > Tab Bar** from the popup menu.



Show toolbar bar

Show/Hide keyboard

To show the keyboard, right click on the phone border and select **Show Keyboard** from the popup menu.



Show keyboard

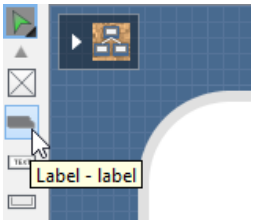
This shows the keybaord at the bottom of the phone:

Keyboard shown

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).

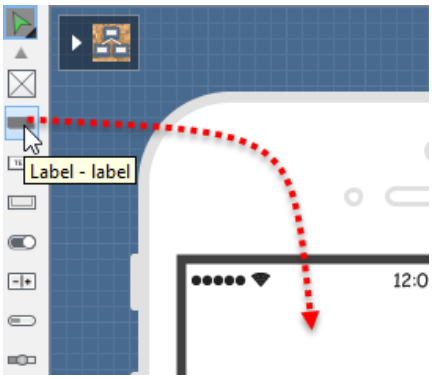


Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

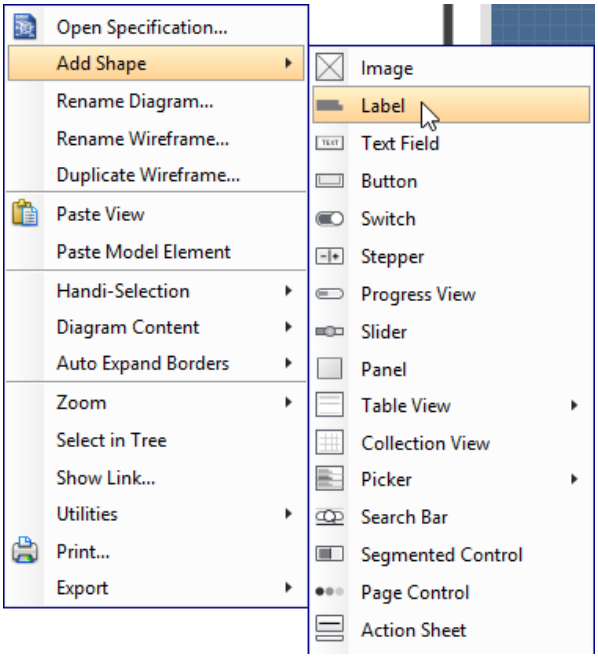


Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

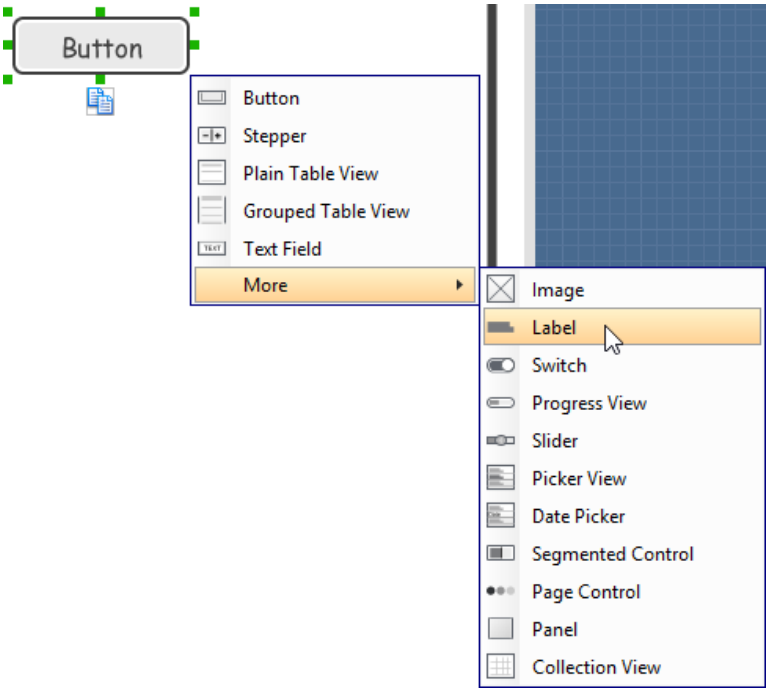
1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appears, known as the **Smart Create** resource.

Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.
3. Drag to outline the size of the wireframe element to be created.

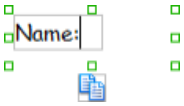
Creating a label in specific size

4. Release the mouse button. In the popup menu, choose the type of wireframe element to create.



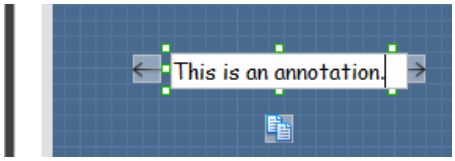
Choosing the wireframe element to be created

Method 5 - Double-clicking (Label and annotation only)
To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the phone border and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

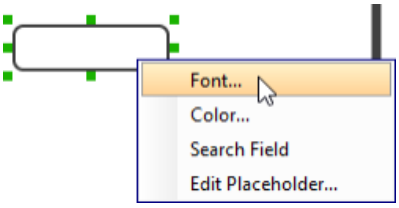
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you to align elements perfectly with others. Simply select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjust the positioning of the selection with the help of the guide.

Using the alignment guide

Adjusting font property for wireframe elements

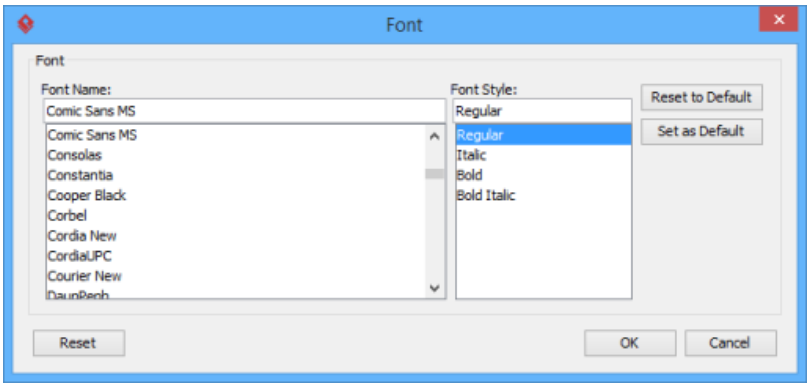
For wireframe elements that can display text, you can adjust their font properties like the font familiy and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

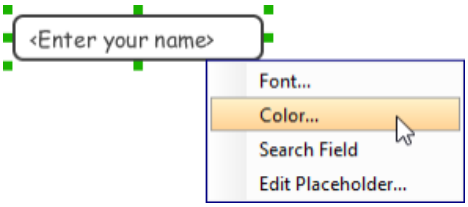


The Font window

Setting color for wireframe elements

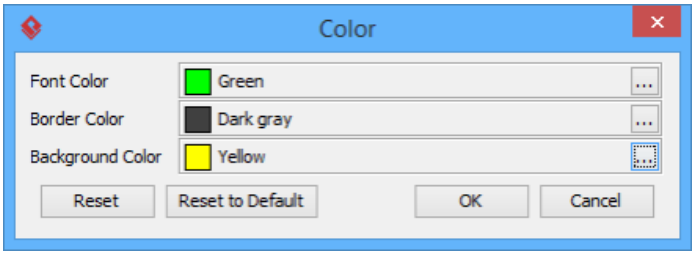
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, collection view doesn't.



Editing color properties

You will then see the new color applied.

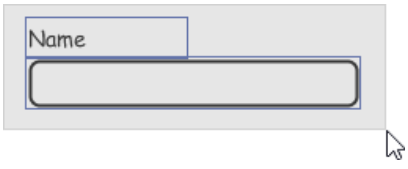
Color applied

Duplicating wireframe elements

Duplicate wireframe elements enables you to create new elements based on existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

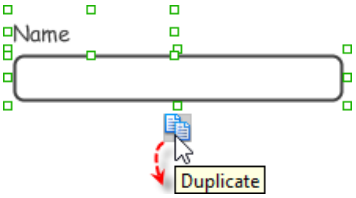
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to duplicate.



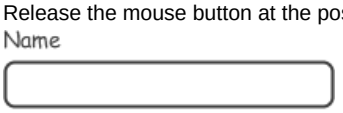
Selecting wireframe elements to duplicate

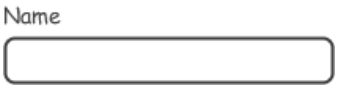
2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



To duplicate wireframe elements

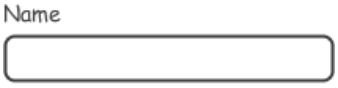
4. Release the mouse button at the position where you want the wireframe element to be created.

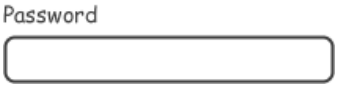




Wireframe elements duplicate

5. Touch-up the duplicate elements.





Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may click on it, too..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

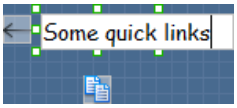
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the phone border. In other words, you cannot create or move an annotation to inside the phone body.

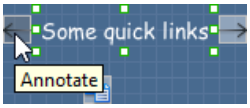
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



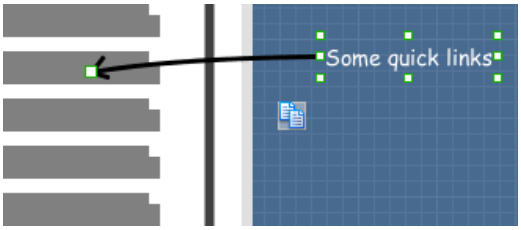
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

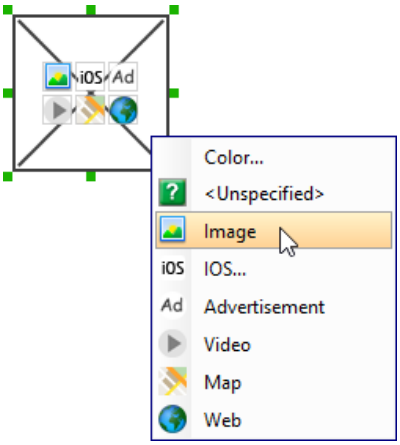
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video, map or web component. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to embed into the image component.



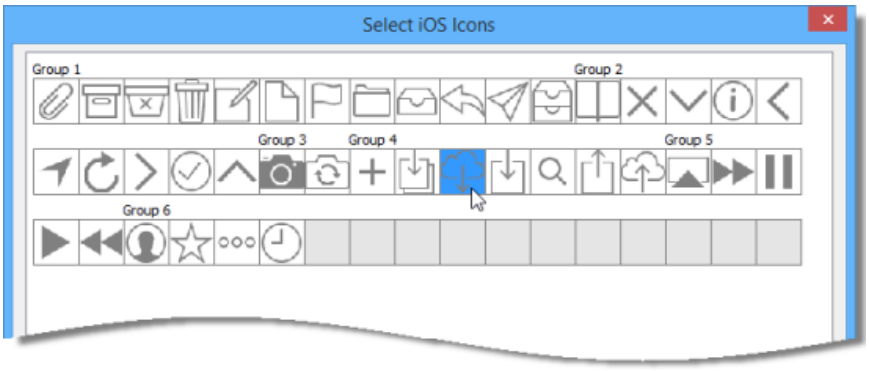
To embed image into image component

To represent an advertisement, video, map or web component, right click on the image component and select **Advertisement**, **Video**, **Map** and **Web** respectively.

Image component showed as video

Showing IOS icon

You can also use an image component to show an IOS icon for a tab bar. To show an IOS icon, right click on the image component and select **IOS...** from the popup menu. In the popup window, choose the icon to show and click **OK** to confirm.

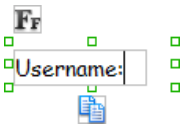


Choosing an IOS icon

Wireframing tips - Label

Specifying the content of label

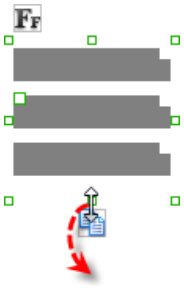
To specify the content of label, double click on the label and enter the content. You can press **Enter** to create a new line or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

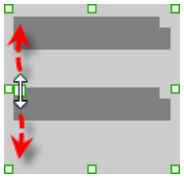
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.



Adjusting the spacing between labels

NOTE: When a label has specified content, you cannot show multiple labels in it.

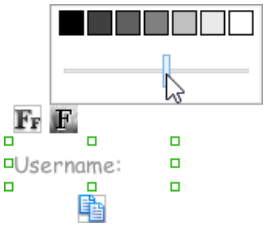
Adjusting label or font size

When the content is filled, the size of label(s) or text in label can be changed. To adjust the font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.

Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

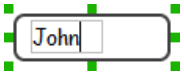


Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.



Specifying the content of text field

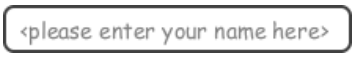
Showing the text field as a search field

Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Search Field** from the popup menu.

A search field

Editing the placeholder text

Placeholder text is the text that appears in the background of a text field. Very often, placeholder text is used to provide hints for user. For example, a text field of user name may have *<please enter your name here>* as placeholder text. Note that the placeholder text is only active when no content has been specified for the text box. To edit placeholder text of a text field, right click on the text field component and select **Edit Placeholder...** from the popup menu. Then, enter the placeholder text in the popup dialog box.

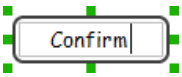


Text field with placeholder text entered

Wireframing tips - Button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered.



Entering button caption

Wireframing tips - Switch

Altering the state of switch

To alter the state of a switch, select the switch first. Then, click on the inactive end of the switch to switch to that end.

Altering switch state

Wireframing tips - Progress View

Adjusting the progress

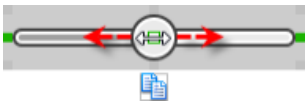
To adjust the progress, select the progress view first. Then drag the handler in the middle towards left or right to control the progress.

Adjusting the progress of progress view

Wireframing tips - Slider

Adjusting the slider position

To adjust the silder position, select the silder first. Then drag the handler in the middle towards left or right to control the position.



Adjusting the slider position

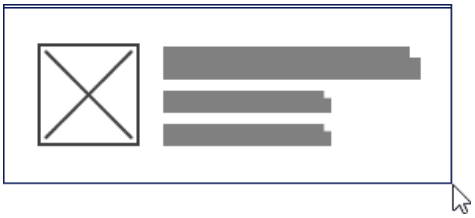
Wireframing tips - Panel

Containing existing components with panel

Panel is a useful wireframe component that helps you to visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

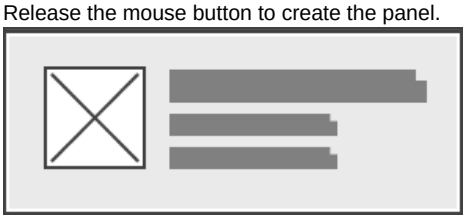
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

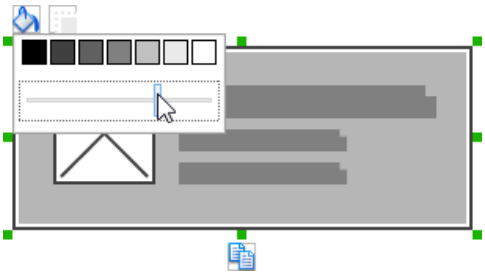
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

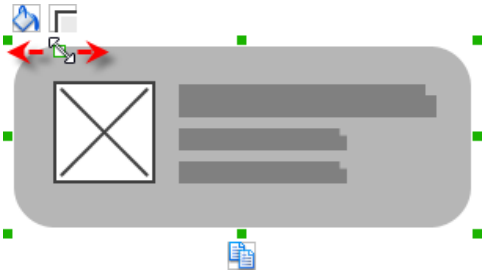
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.



Making the corner of panel rounded

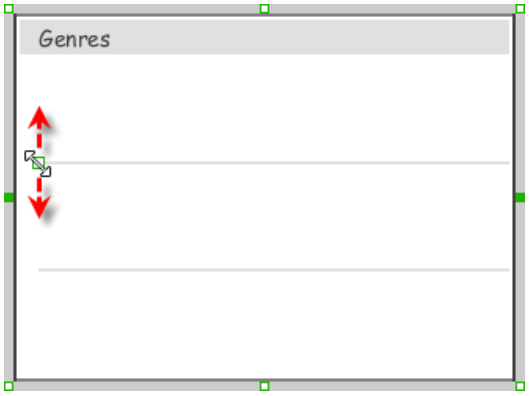
Wireframing tips - Plain Table View

Editing the header

To edit the table header, double click on the table header and enter the content.

Editing the header

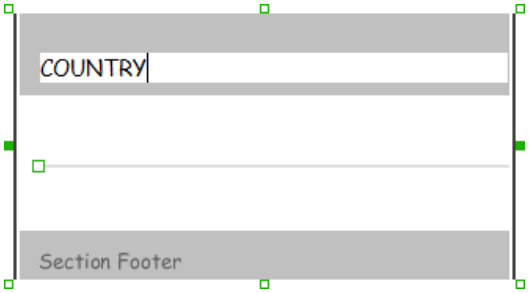
Adjusting row height
To adjust the row height, drag on the handler between the first and the second row.



Adjusting the row height

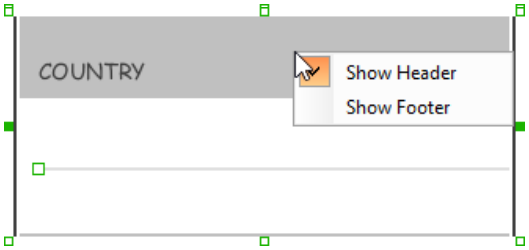
Wireframing tips - Grouped Table View

Editing the header/footer
To edit the table header/footer, double click on the table header/footer and enter the content.



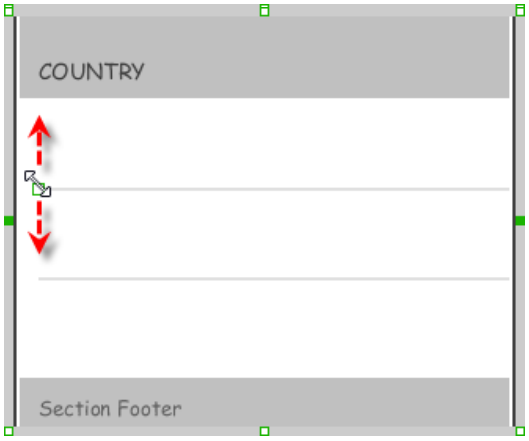
Editing the header

Showing/hideing the header/footer
By default, table header and footer are visible in a grouped table view. To hide header/footer, right click on the grouped table view and de-select **Show Header/Footer** from the popup menu.



Footer in grouped table view hidden

Adjusting row height
To adjust row height, drag on the handler between the first and the second row.



Adjusting the row height

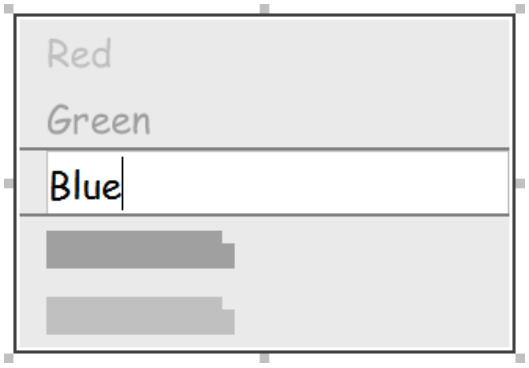
Wireframing tips - Collection View

Adjusting cell size
To adjust the size of cells, drag on the handler between the first and the second row.

Adjusting cell size

Wireframing tips - Picker View

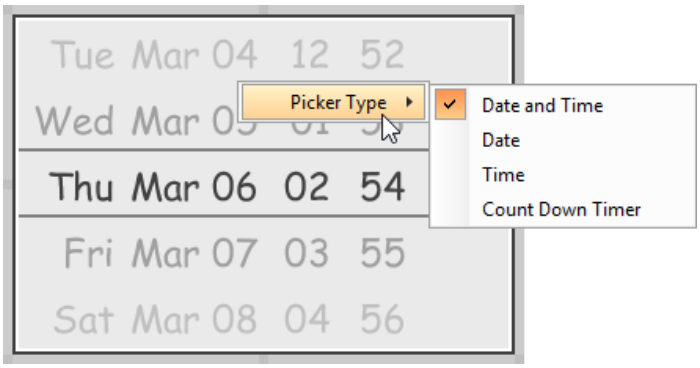
Editing label in picker view
To edit the lable in a picker view, double click on the label and enter the content.



Editing a label in picker view

Wireframing tips - Date Picker View

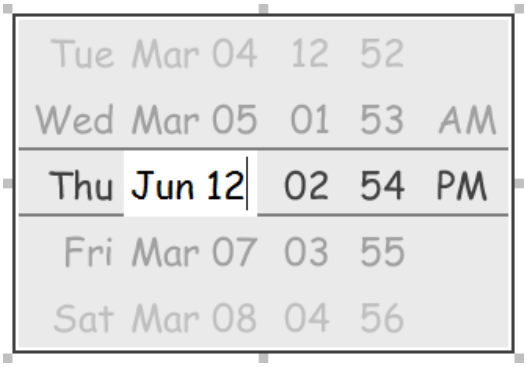
Changing the picker type
There are four kinds of date picker - Date-and-Time, Date, Time, Count Down Timer. To change the picker type, right click on the date picker and select **Picker Type > [TYPE]** from the popup menu.



Changing picker type

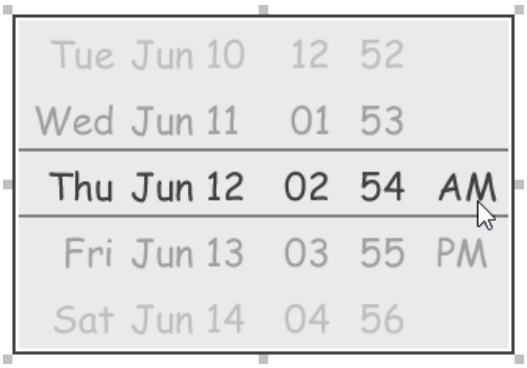
Editing the values of date/time

To edit the values in a date picker, like to edit the date in a Date-and-Time picker, double click on the field to edit and enter the new value.



Editing the date in a date picker

To switch between AM and PM, double click on AM/PM directly.



Changing PM to AM

Wireframing tips - Search Bar

Specifying the content of search field

To specify the content of the search field, double click on the text field and enter the content. You may need to resize the search bar afterwards in order to see the content entered.

Specifying the content of search field

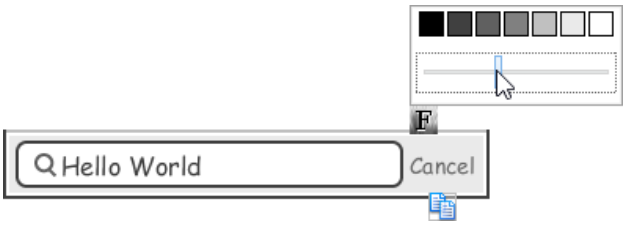
Renaming the Cancel button

If you have specified the content of search field, the Cancel button will appear. To rename the Cancel button, double click on the it and enter the new caption.

Renaming the Cancel button

Adjusting font color of Cancel button

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

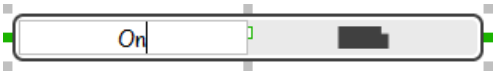


Adjusting font color of the Cancel button

Wireframing tips - Segmented Control

Editing the title of a segment

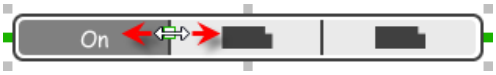
To edit the title of a segment, double click on the segment and enter the title.



Editing the title

Adding more segments

To add more segments to a segmented control, select the segmented control first. Then, drag on the handler attached to the segment separator between segments to create more segments.



Adding more segments to a segmented control

Changing the selected segment

To change the selected segment, select the segmented control first. Then, click on the segment directly in a segmented control.

Changing the selected segment in a segmented control

Wireframing tips - Page Control

Adding more page indicators

To add more page indicators to a page control, select the page control first. Then, extend the page control to let more indicators appear.

Adding more page indicators to a page control

Setting the active page indicator

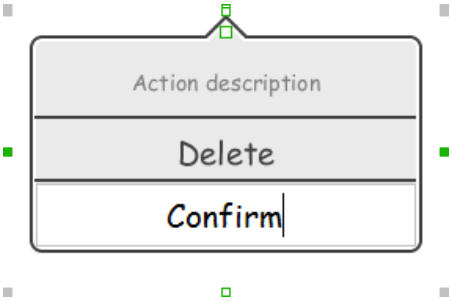
To set the active page indicator, select the page control first. Then, click on the page indicator directly in the page control.

Setting the active page indicator in a page control

Wireframing tips - Action Sheet

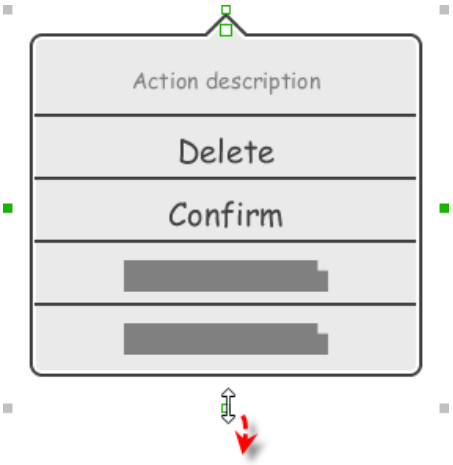
Editing the action description and action

To edit action description or action, double click on the corresponding label and enter the content.



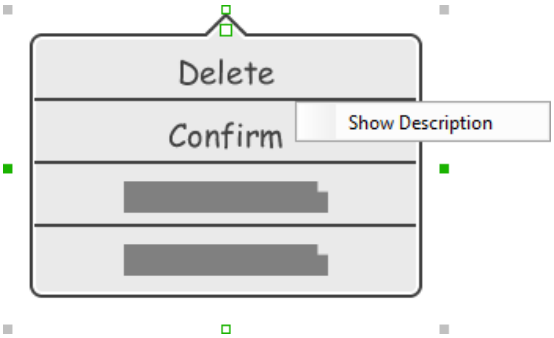
Editing the action

Showing more actions
To show more actions, increase the height of the action sheet.



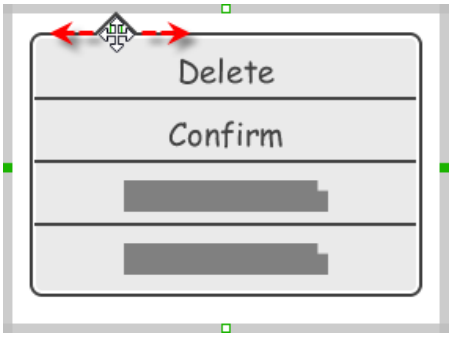
Resizing an action sheet to show more actions

Showing/hideing the action description
By default, the action description field is visible in an action sheet. To hide an action description, right click on the action sheet and de-select **Show Description** from the popup menu.



Action description in an action sheet is hidden

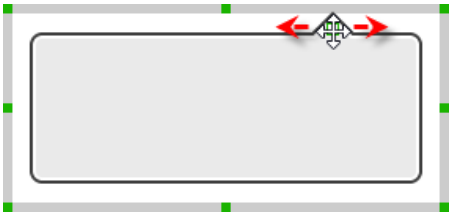
Adjusting the position of pointer
To adjust the position of pointer, select the action sheet first. Then, drag on the pointer to adjust its position. You can drag to all the four sides of the action sheet.



Adjusting the position of pointer in an action sheet

Wireframing tips - Popover

Adjusting the position of pointer
To adjust the position of pointer, select the popover first. Then, drag on the pointer to adjust its position. You can drag to all the four sides of the popover.

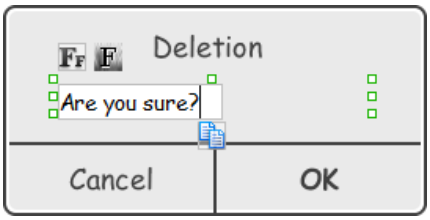


Adjusting the position of pointer in an popover

Wireframing tips - Alert View

Editing the content of labels and buttons in an alert view

To edit the content of labels and buttons in an alert view, double click on the label or button and enter the content. You may need to resize the label or the alert view afterwards in order to see the content entered.

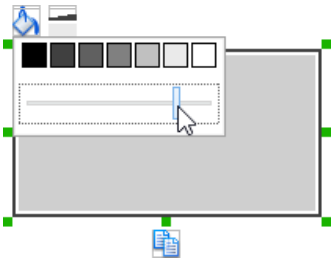


Editing the message in an alert view

Wireframing tips - Rectangle

Adjusting fill color

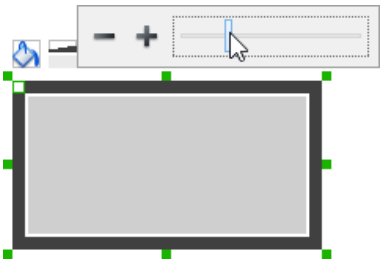
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

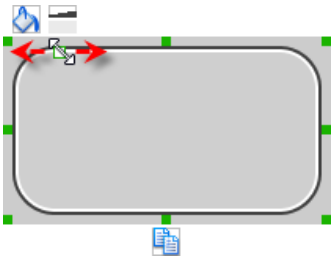
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.

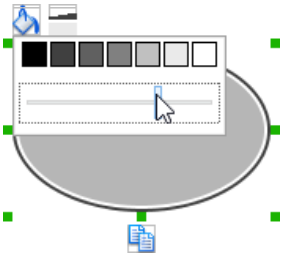


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

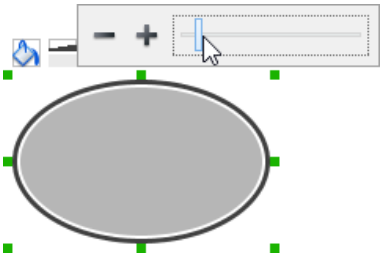
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

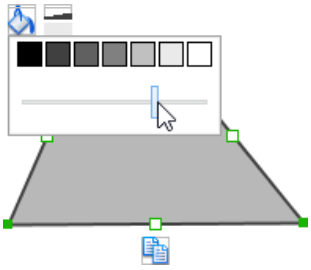
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

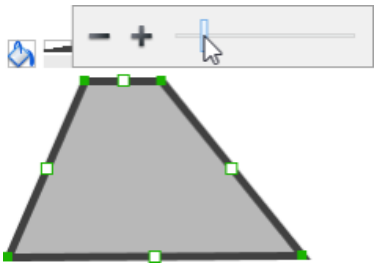
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

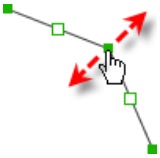


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



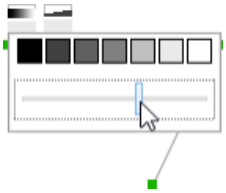
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

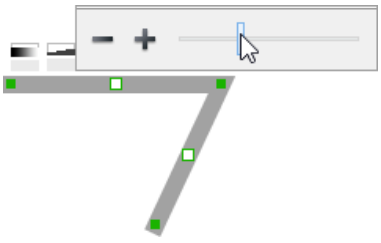
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Trademark Disclaimer

iPhone is a trademark of Apple Inc.

Supported Editions

- Professional Edition
- Enterprise Edition

Web wireframing skills

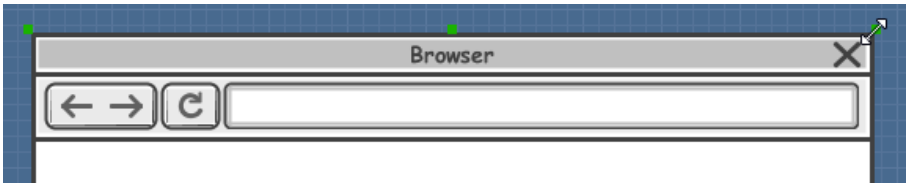
Adjust the size of browser window

If you find the browser window too large or too small, you can resize it to your preferred size. To adjust the size of browser window:

1. Click on the browser window's title bar.

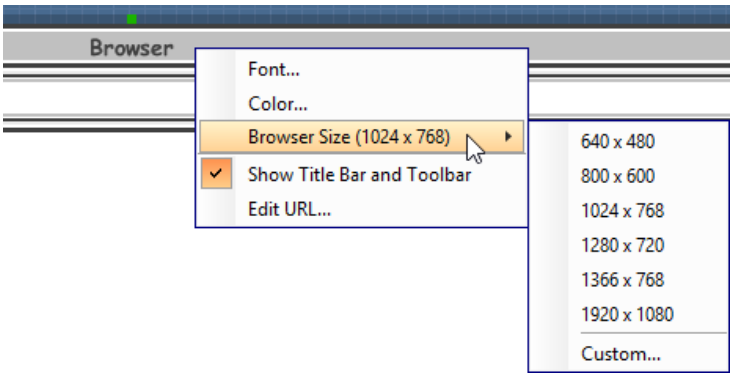
Clicking on browser window's title bar

2. This shows the resize handlers at the corners and edges of the browser window. You can drag on them to resize the browser window.



Resizing browser window

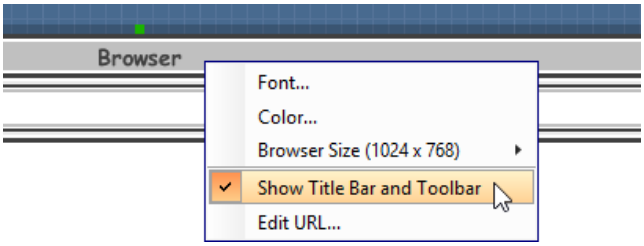
Alternatively, you can right click on the browser window's title bar and select **Browser Size > %PREFERRED_SIZE%** from the popup menu.



Adjusting browser window size

Show/Hide title bar and toolbar

To hide the title bar and toolbar, right click on the browser window's title bar and de-select **Show Title Bar and Toolbar** from the popup menu.



Hide Show Title Bar and Toolbar

To show it again, right click on the wireframe and select Edit Browser Frame. Move the mouse pointer near the top most of the browser window. Right click and select **Show Title Bar and Toolbar** from the popup menu.

Editing the displaying URL

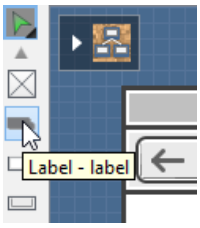
To edit the displaying URL, just double click on the URL field and enter the URL.

Editing the displaying URL

Creating a wireframe element

Method 1 - Diagram toolbar: Select-and-Click

1. Select the desired wireframe element from the diagram toolbar (e.g. Label).

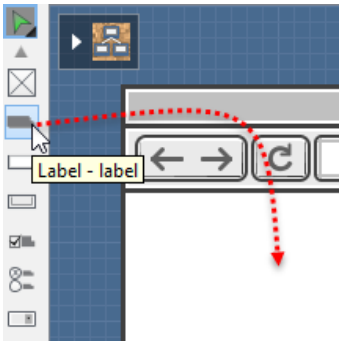


Create a label by selecting it from the diagram toolbar

2. Click on the wireframe, at the position where you want the wireframe element to be created.

Method 2 - Diagram toolbar: Drag and drop

1. Press on the desired wireframe element in the diagram toolbar
2. Hold the mouse button.
3. Drag to the wireframe.

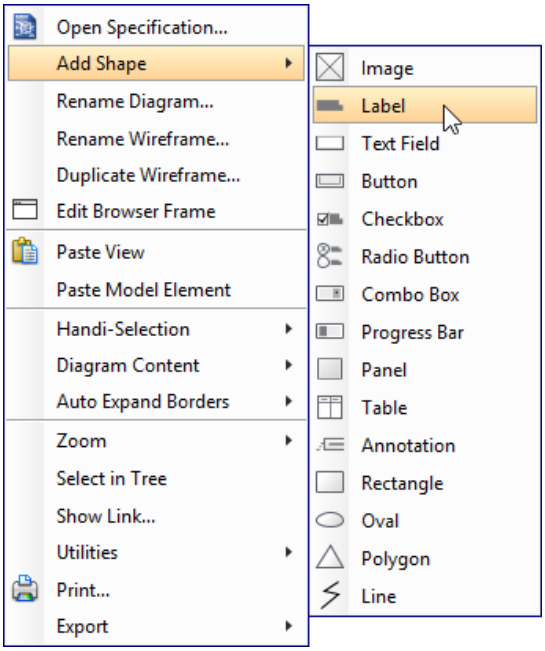


Create a Label with drag-and-drop

4. Release the mouse button in the wireframe, at the position where you want the wireframe element to be created.

Method 3 - Popup menu

1. Right click on the wireframe, at the position where you want the wireframe element to be created.
2. Select **Add Shape > %SHAPE_TYPE%** from the popup menu, where **%SHAPE_TYPE%** is the kind of wireframe element you want to create.



Creating a wireframe element via the popup menu

Method 4 - Through smart create resource

1. Click directly on the wireframe, at the position where you want the wireframe element to be created. You should see a green icon appears, known as the **Smart Create** resource.

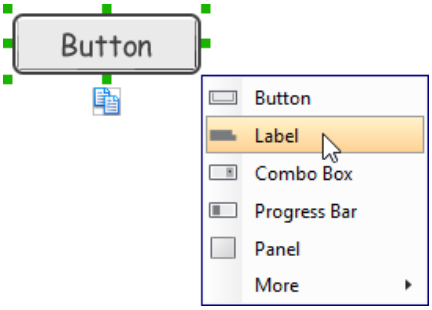
Creating a wireframe element using Smart Resource

2. Press on the **Smart Create** resource and hold the mouse button.

3. Drag to outline the size of the wireframe element to be created.

Creating a label in specific size

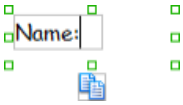
4. Release the mouse button. In the popup menu, choose the type of wireframe element to be created.



Choosing the wireframe element to be created

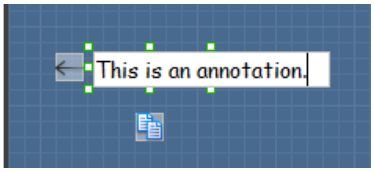
Method 5 - Double-clicking (Label and annotation only)

To create a label, double-click on the wireframe and enter the label caption.



Creating a label

To create an annotation, double-click outside the browser window and enter the annotation text.



Entering annotation text

Accurate positioning of wireframe element using the alignment guide

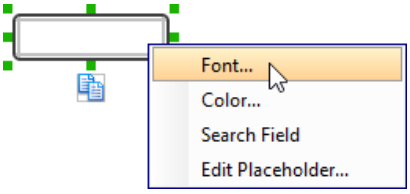
Alignment guide is a dotted line that appears when you move wireframe elements in a wireframe. It helps you to align elements perfectly with others. Simply select element(s) and drag it around. When the selection approaches another element in the wireframe, you can adjust the positioning of selection with the help of the guide.

Using the alignment guide

Adjusting font property for wireframe elements

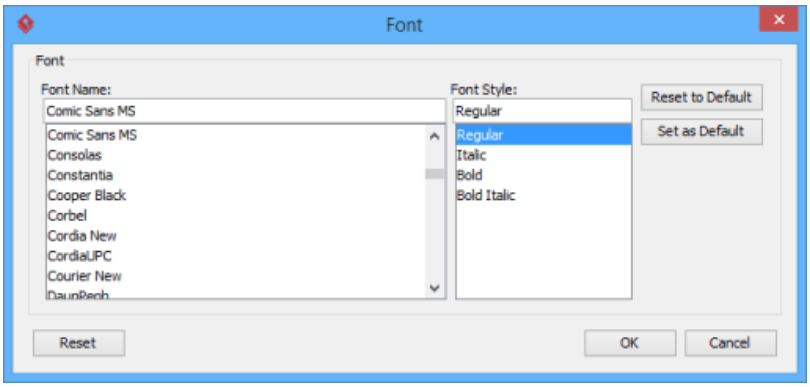
For wireframe elements that can display text, you can adjust their font properties like the font familiy to use and font style. To adjust font properties:

1. Right click on the desired wireframe element and select **Font...** from the popup menu.



Set font

2. Edit the font properties in the **Font** window and click **OK** to confirm.

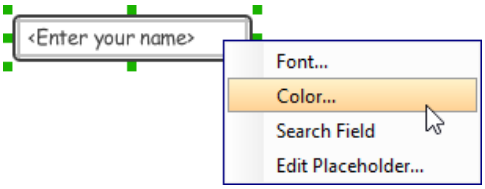


The Font window

Setting color for wireframe elements

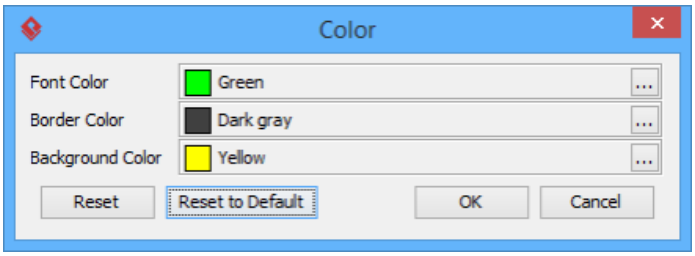
In case you want to colorize your wireframe, you can set the color of wireframe elements by taking the steps below.

1. Right click on the desired wireframe element and select **Color...** from the popup menu.



Set color

2. Edit the color properties in the **Color** window and click **OK** to confirm. Note that different wireframe elements support different settings. For example, while text field support editing font color, panel doesn't.



Editing color properties

You will then see the new color applied.

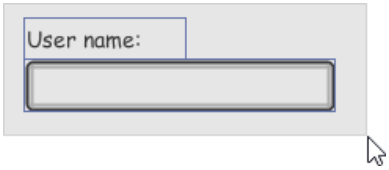
Color applied

Duplicating wireframe elements

Duplicate wireframe elements enables you to create new elements based on existing ones. This saves you a lot of time in creating elements with same/similar style, size and content to the existine ones.

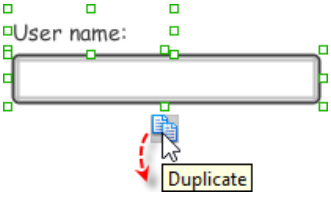
To duplicate wireframe elements:

1. Drag in the wireframe to select a range of elements to be duplicated.



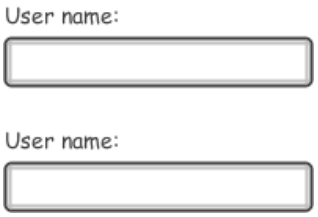
Selecting wireframe elements to duplicate

2. At the bottom of your selection, press on the **Duplicate** resource icon and hold the mouse button.
3. Drag it.



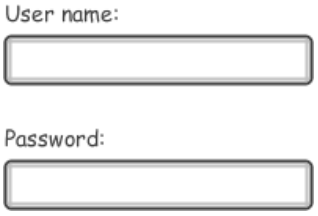
To duplicate wireframe elements

4. Release the mouse button at the position where you want the wireframe element to be created.



Wireframe elements duplicate

5. Touch-up the duplicate elements.



Renamed a label

NOTE: Instead of dragging the **Duplicate** resource, you may click on it, too..

Using stereotypes and tagged values in wireframe

Like most model elements, you can add stereotypes and tagged values to wireframe widgets. The stereotype mechanism allows you to create "typed" widgets. For example, you can create a stereotype "password field" for wireframe text field widget. Then, when you draw a wireframe, you can create a "password field" by assigning the "password field" stereotype to a text field.

Tagged values allows you to add custom properties to wireframe widgets. For example you can add a "screen ID" field to widgets for associating the widgets to components in an actual screen design.

You can add and edit stereotypes and tagged values both in the specification of wireframe widgets. To open specification, right click on a widget and select **Open Specification...** from the popup menu.

Note that the stereotypes and tagged values added to a wireframe widget is shared across all states, which means that you will see the same stereotypes and tagged values added to a widget in all wireframe states.

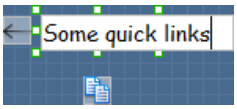
Annotating wireframe with Annotation shape

The use of annotations in wireframe allows you to detail the elements on the wireframe. With annotation, you can describe or explain the existence of certain wireframe element, as well as to describe the calls to action and the expected results.

In order to keep the wireframe content readable, annotations are forced to put outside the browser window. In other words, you cannot create or move an annotation to inside the browser window area.

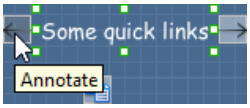
To create an annotation:

1. Double click on the background (i.e. the blue region) of the wireframe.
2. Enter the annotation text.



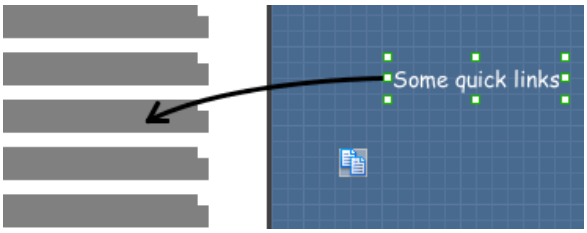
Entering annotation text

3. Press on the arrow resource and hold the mouse button.



To annotate a wireframe element

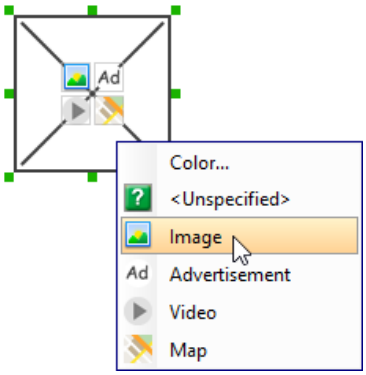
4. Drag to the wireframe element to annotate it.



Label annotated

Wireframing tips - Image

You can use an image component to represent a picture, a placeholder of advertisement, video or map. When you create an image in a wireframe, you see a box with a cross in it. This is how an image should be shown in a wireframe but if you want to specify the content of the image, right click on the image component and select **Image** from the popup menu. Then, choose the image file (*.jpg, *.jpeg, *.gif, *.png, *.bmp) to embed into the image component.



To embed image into image component

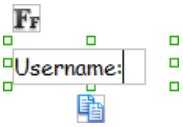
To represent an advertisement, video or map, right click on the image component and select **Advertisement**, **Video** and **Map** respectively.

Image component showed as video

Wireframing tips - Label

Specifying the content of label

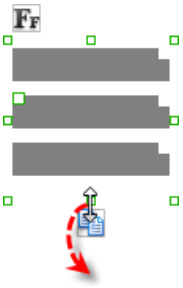
To specify the content of label, double click on the label and enter the content. You can press **Enter** to create a new line, or press **Ctrl-Enter** to confirm editing. You may need to resize the label afterwards in order to see the content entered.



Specifying the content of label

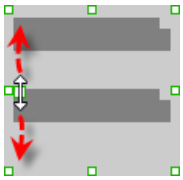
Showing multiple labels

The label component is in fact a placeholder of label. You can show multiple labels in it by increasing the height of the label component.



Creating more labels

To adjust the spacing between labels in a label component, select the label and drag the handler attached with the second label. Space will be added by dragging downwards.

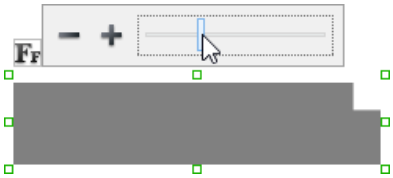


Adjusting the spacing between labels

NOTE: When a label has specified content, you cannot show multiple labels in it.

Adjusting label or font size

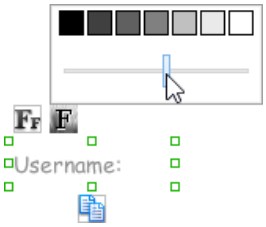
When the content is filled, the size of label(s) or text in label can be changed. To adjust the font, click on the label component. Then, click on the **Font Size** button. After that, drag the slider or press + or - to adjust the font size.



Adjusting label size

Adjusting font color

When a label component has text content filled, you can set its color by clicking on the label component first. Then, click on the **Font Color** button. After that, drag the slider or directly click on a suggested color to apply it.

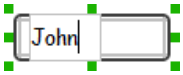


Adjusting font color of label

Wireframing tips - Text Field

Specifying the content of text field

To specify the content of text field, double click on the text field and enter the content. You may need to resize the text field afterwards in order to see the content entered.



Specifying the content of text field

Showing the text field as a search field

Search field is a kind of text field that allows user to specify a search string and trigger searching. To show a text field as a search field, right click on the text field component and select **Search Field** from the popup menu.

A search field

Editing the placeholder text

Placeholder text is the text that appears in the background of a text field. Very often, placeholder text is used to provide hints for user. For example, a text field of user name may have *<please enter your name here>* as placeholder text. Note that the placeholder text is only active when no content has been specified for the text box. To edit placeholder text of a text field, right click on the text field component and select **Edit Placeholder...** from the popup menu. Then, enter the placeholder text in the popup dialog box.

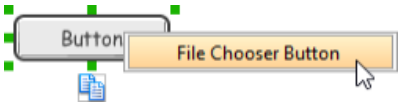
Text field with placeholder text entered

Wireframing tips - Button

Setting the type of button

There are two kinds of button - general button and file chooser button. General button is what you commonly see in any user interface. You click on a general button to do something as description by the button caption. File chooser button is a special kind of button that allows users to provide a file by clicking on the button. A file chooser button is followed by the text "No file chosen".

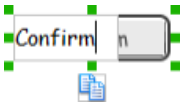
When you create a button, it is a general button by default. To change make it a file chooser button, right click on the button and select **File Chooser Button** from the popup menu.



Changing a button to a file chooser button

Editing button caption

To edit the caption of button, double click on the button and enter the caption. You may need to resize the button afterwards in order to see the caption entered. Note that the caption of a file chooser button is not editable.

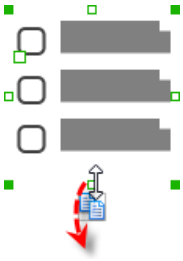


Entering button caption

Wireframing tips - Checkbox

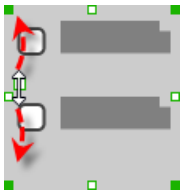
Creating the checkboxes

The checkbox component is in fact a placeholder of checkboxes. You can show multiple checkboxes in it by increasing the height of the checkbox component.



Creating more checkboxes

To adjust the spacing between checkboxes in a checkbox component, select the checkbox and drag the handler between the first and the second checkbox. Space will be added by dragging downwards.



Adjusting the spacing between checkboxes

Specifying the value of checkbox

To specify the value of a checkbox, double click on the label attached with the checkbox and enter the value. You may need to resize the checkbox afterwards in order to see the content entered.

Entering the value of a checkbox

Checking a checkbox

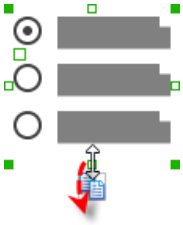
To check a checkbox, simply click on the checkbox. You can uncheck it by clicking again.

Checking checkboxes

Wireframing tips - Radio Button

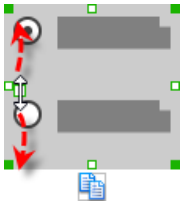
Creating the radio buttons

The radio button component is in fact a radio button group. You can show multiple radio buttons in it by increasing the height of the radio button component.



Creating more radio buttons

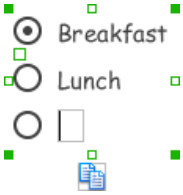
To adjust the spacing between radio buttons in a radio button component, select the radio button and drag the handler between the first and the second radio button. Space will be added by dragging downwards.



Adjusting the spacing between radio buttons

Specifying the value of radio button

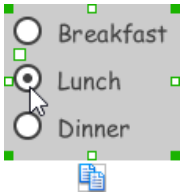
To specify the value of a radio button, double click on the label attached with the radio button and enter the value. You may need to resize the radio button afterwards in order to see the content entered.



Specifying the value of radio button

Selecting a radio button

To select a radio button, simply click on the radio button. You can uncheck it by clicking again.

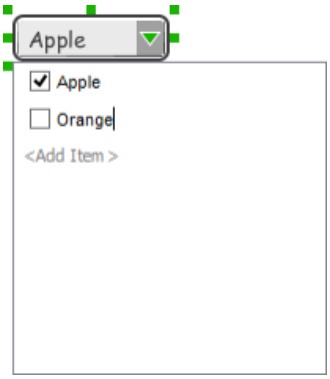


Selecting a radio button

Wireframing tips - Combo Box

Specifying the items in a combo box

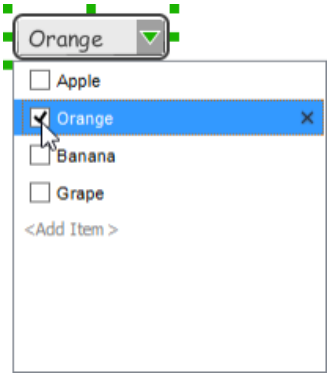
By default, a combo box has no items specified. You can add into a combo box a list of items by clicking on the down arrow on the right of the combo box, and then click on **<Add Item>** and start entering the item.



Adding an item to combo box

Selecting the selected item in a combo box

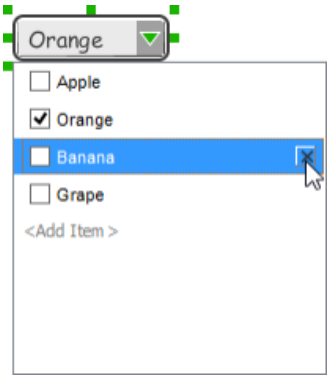
To change the selected item of a combo box, click on the down arrow on the right of the combo box, then check the item to select it.



Selecting an item in combo box

Removing an items from a combo box

To remove an item from a combo box, click on the down arrow on the right of the combo box, then select the item to remove and click on the cross button on the right to remove it.

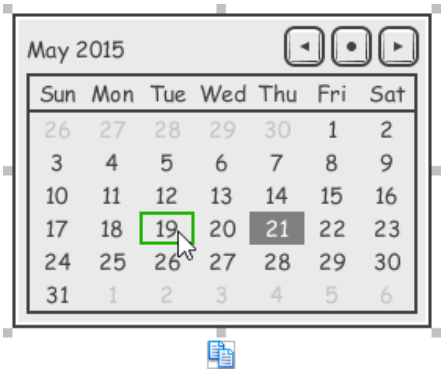


Removing an item from combo box

Wireframing tips - Date Picker

Specifying a date

By default, a date picker highlights the date on which the widget was created. If you want to 'pick' another date, simply click on the desired date on the date picker. If you want to change the active month, click on the forward and previous button at top right. If you want to change the active year, double click on the text of month and year at top left and edit it.



Choosing a date in Date Picker

Wireframing tips - Progress Bar

Adjusting the progress

To adjust progress, select the progress bar first. Then drag the handler in the middle towards left or right to control the progress.

Adjusting the progress of progress bar

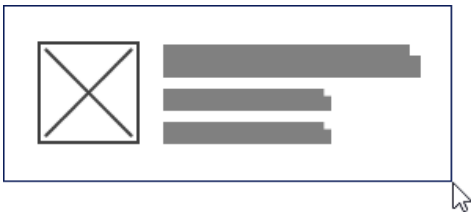
Wireframing tips - Panel

Containing existing components with panel

Panel is a useful wireframe component that helps you visualize the different areas of a screen design. You can put other wireframe components in a panel and move the panel around to reposition the wireframe components at the same time.

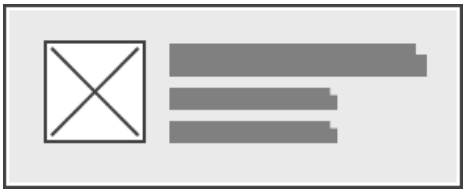
To create a panel and make it contains existing components:

1. Select **Panel** from the diagram toolbar.
2. Press on the wireframe and hold your mouse button.
3. Drag to form the size of the panel to be created. Wireframe components contained entirely in the drag range will be contained by the panel.



Creating a panel

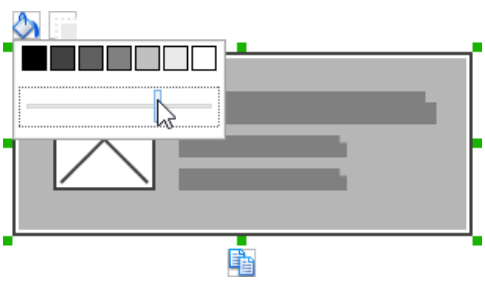
4. Release the mouse button to create the panel.



Panel created

Adjusting fill color

To adjust the fill color of a panel, click on the panel first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

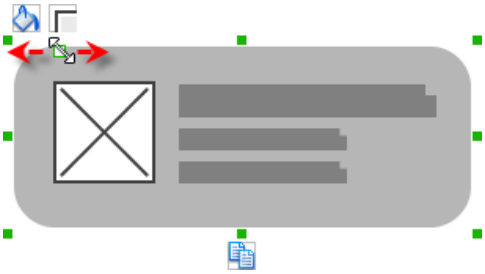
Hiding the border of panel

To hide the border of panel, click on the panel first. Then, click on the **Hide Border** button. You can click again to show the border again.

Panel with border hidden

Making the corner of panel rounded

To make the corner of a panel rounded, click on the panel first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.



Making the corner of panel rounded

Changing a panel to a titled pane or tabbed pane

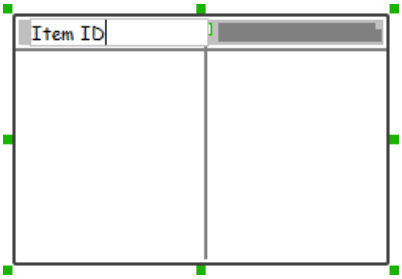
A panel can be presented as a titled pane or tabbed pane. To do this, right click on the panel and select **Type > Titled Pane** or **Type > Tabbed Pane** from the popup menu.

When presented as titled pane or tabbed pane, you can edit the caption of the pane or tabs by double clicking on captions.

Wireframing tips - Table

Editing table header

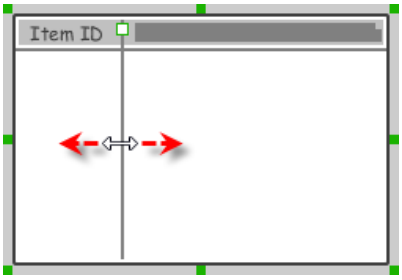
To edit table header, double click on the table header and enter the content.



Editing table header

Adjusting column width

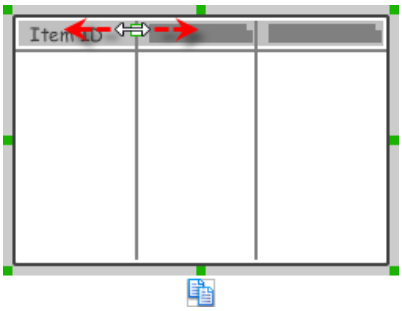
To adjust column width, drag directly on the column separator nearby. By doing so, the adjacent columns will be updated in their width.



Adjusting column width of a table

Adding more columns

To add more columns to a table, select the table first. Then, drag on the handler attached to the column separator between the first and the second column to create more columns.



Adding more columns to a table

Adding more rows

When you put a wireframe component (e.g. a label component) into a table component, a new row will be created automatically.

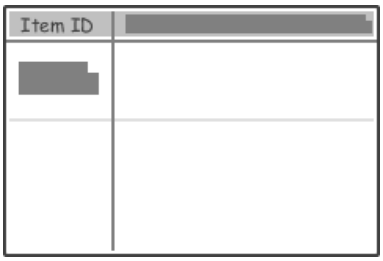
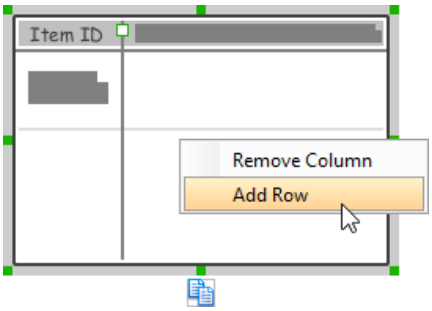


Table with one row

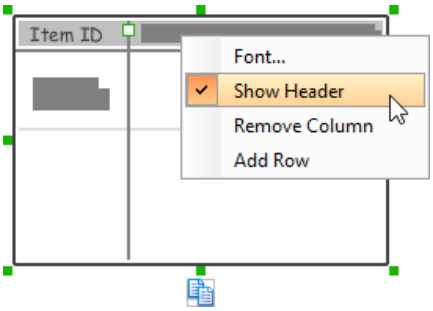
You may also add a row manually by right clicking on the table and selecting **Add Row** from the popup menu.



Adding row to a table

Hiding table header

To hide away the table header, right click on the header and uncheck Show Header from the popup menu.

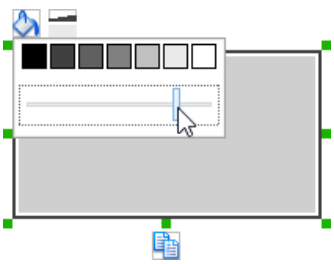


Hiding table header

Wireframing tips - Rectangle

Adjusting fill color

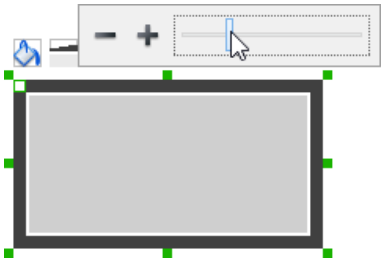
To adjust the fill color of a rectangle, click on the rectangle first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

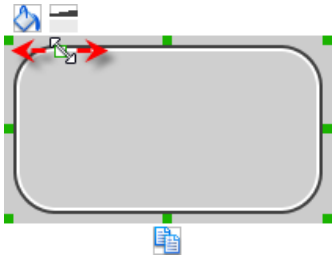
To adjust the thickness of the border of a rectangle, click on the rectangle first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Making the corner of rectangle rounded

To make the corner of a rectangle rounded, click on the rectangle first. Then, drag on the handler at top left to adjust the size of the rounded corner. The four corners will be updated accordingly.

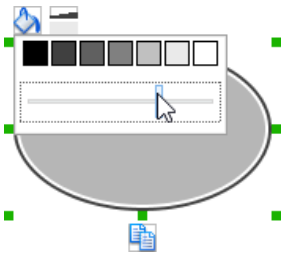


Making the corner of rectangle rounded

Wireframing tips - Oval

Adjusting fill color

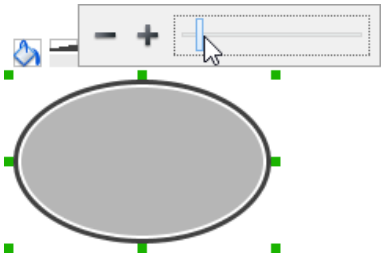
To adjust the fill color of an oval, click on the oval first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of an oval, click on the oval first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.



Adjusting the thickness of border

Wireframing tips - Polygon

Adding a side

To add a side, select the polygon first. Then, drag on the white handler attached to the border of the polygon to split a border into two.

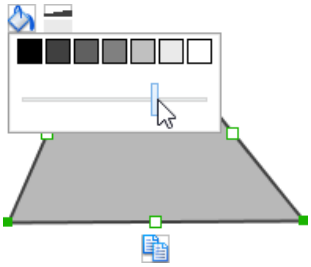
Adding a side

Then adjust the position of the new point.

Polygon edited

Adjusting fill color

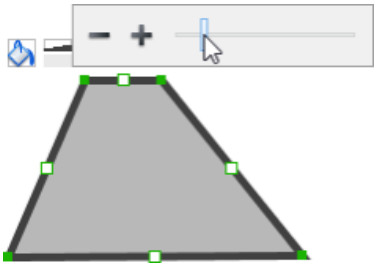
To adjust the fill color of a polygon, click on the polygon first. Then, click on the **Fill Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting fill color

Adjusting the thickness of border

To adjust the thickness of the border of a polygon, click on the polygon first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the thickness.

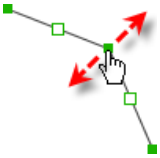


Adjusting the thickness of border

Wireframing tips - Line

Adding a point

To add a point, select the line first. Then, drag on the white handler on the line to create a new point.



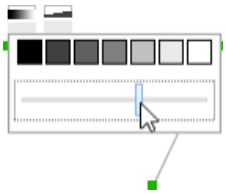
Adding a point

Then adjust the position of the point.

Line edited

Adjusting line color

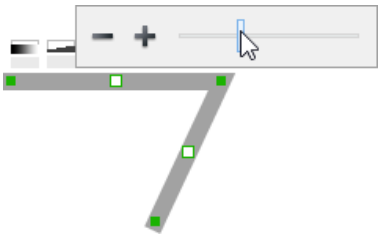
To adjust the line color of a line, click on the line first. Then, click on the **Line Color** button. After that, drag the slider or directly click on a suggested color to apply it.



Adjusting line color

Adjusting the line width

To adjust the width of the border of an oval, click on the line first. Then, click on the **Line Width** button. After that, drag the slider or press + or - to adjust the width.



Adjusting line width

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Duplicate wireframe

The duplicate function allows you to re-use an existing wireframe to create a new one. Note that the duplicate function does not duplicate any child state.

To duplicate a wireframe:

1. Open the wireframe to duplicate.
2. Right click on the background of wireframe or on the device and select **Duplicate Wireframe...** from the popup menu.
3. Enter the name of the new wireframe and click **Duplicate** to confirm.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Using the screen capture tool in wireframe

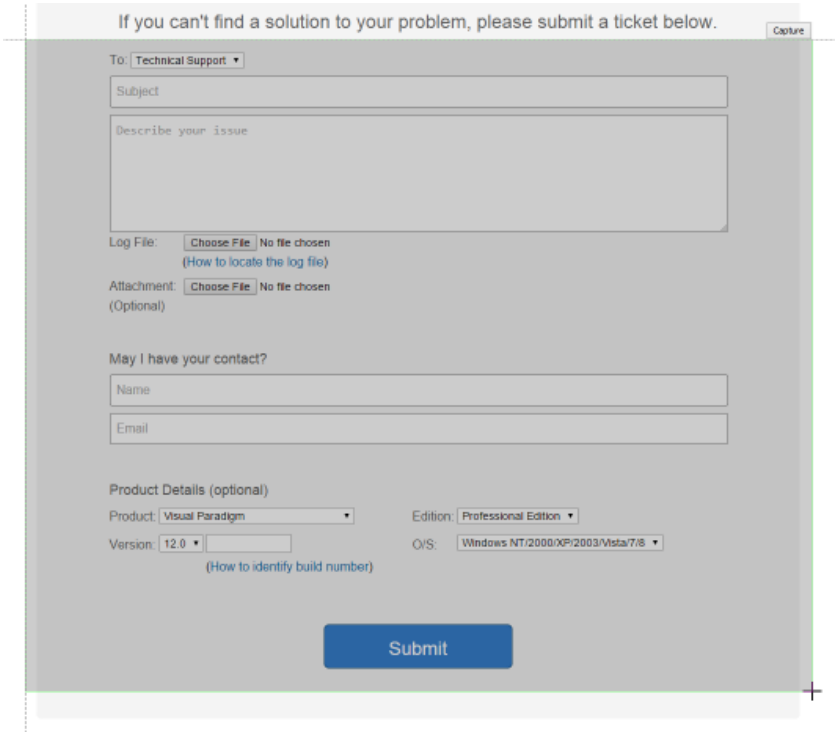
In Visual Paradigm, you can take screenshot of your computer and then embed the screenshot into a wireframe. This enables you to snap the screen of an existing application or system and have it included in a wireframe as reference. This is very useful if your team is going to re-build an existing IT system or to add new features to it, which need to consider the consistency between the new and the existing system when working out the requirements.

To use the screen capture tool in wireframe:

1. Select the **Screen Capture** tool in the diagram toolbar of wireframe.

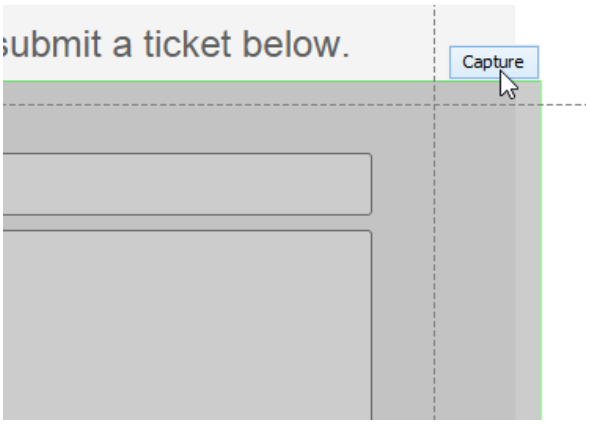
Select the Screen Capture tool

2. Once selected, you will immediately enter the screen capture mode. Drag directly on your screen to select the range to capture.



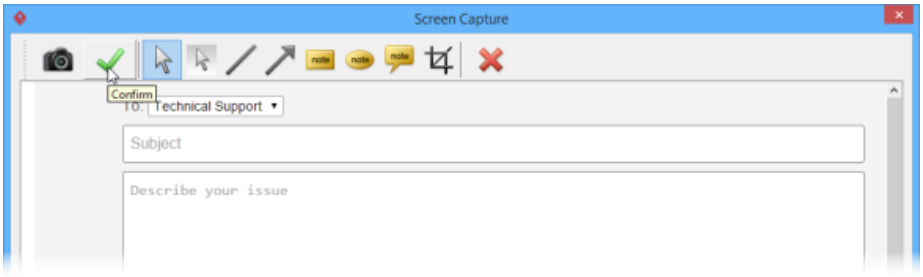
Select the range to capture

3. Release your mouse button to confirm the range. If you are not pleased with your selection, you can select again.
4. Click on the **Capture** button on top of the selected range.



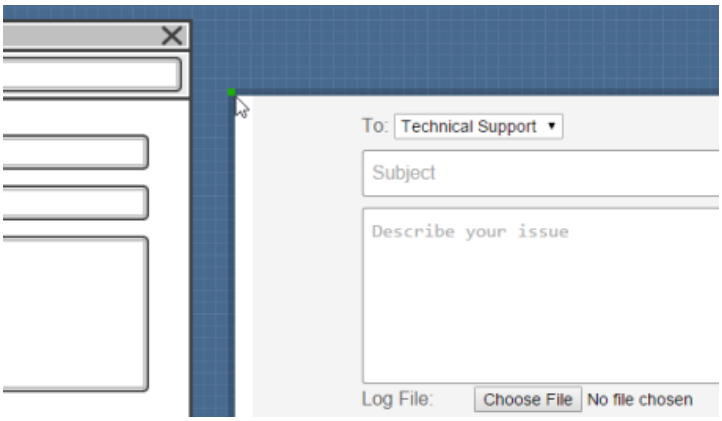
Capture the selected range

5. Screen is captured. You can optionally edit the captured screen in the **Screen Capture** window, like to add arrow and notes into the screenshot. Click **Confirm** to confirm the editing. Note that the screenshot is non-editable.



Confirm editing

6. Click in the wireframe to drop the captured screen into it.



–

Captured screen placed in wireframe

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Using Smart Edit in Wireframe

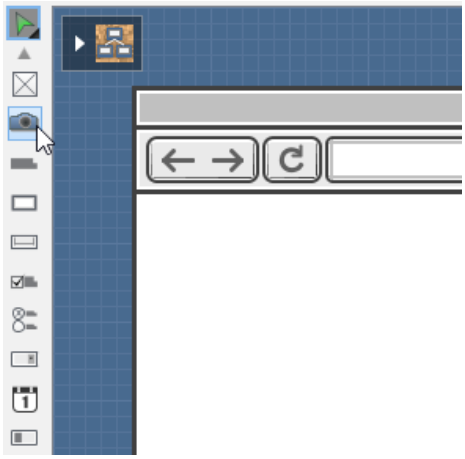
Software development is an ongoing process, with changes made based on new user requests. In order to satisfy users' needs, we need to understand their requests accurately. Unfortunately, user requests are often vague or lack details

Although changes are inevitable, having a way to accurately understand user requests can ease the process. Visual Paradigm suggests to use wireframe as a medium in confirming user requests. With the Smart Edit tool in wireframe, you can take screenshots from the current system screens, and then make ad-hoc changes like to reposition screen components, reset component type (e.g. combo box to textbox) and add new components. Users can confirm their needs based on the wireframe, which ensures the proper implementation of their requests.

Capturing a screenshot

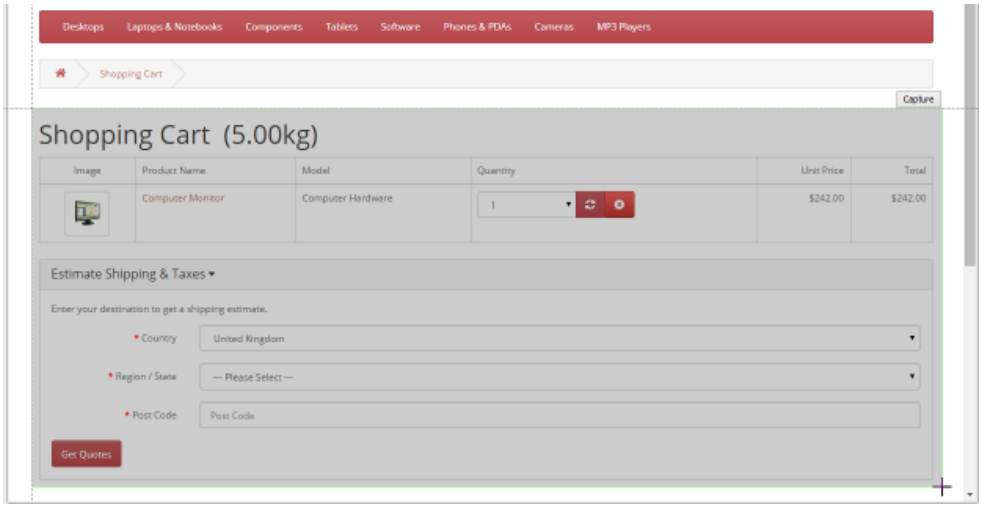
In Visual Paradigm, you can take screenshot of your computer and then embed the screenshot into a wireframe. This enables you to snap the screen of an existing application or system and have it included in a wireframe as reference.

1. Open the wireframe in which you want to insert a screenshot, or create one. Note that you can add screenshot to any type of wireframe.
2. Select the **Screen Capture** tool in the diagram toolbar of wireframe.



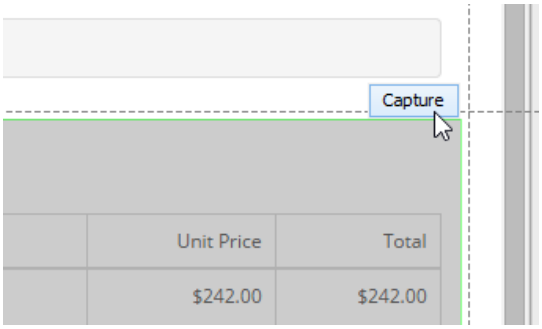
Select Screen Capture tool

3. Once selected, you will immediately enter the screen capture mode. Your computer screen is temporarily "frozen" for you to capture the region you want. Drag directly on your screen to select the range to capture.



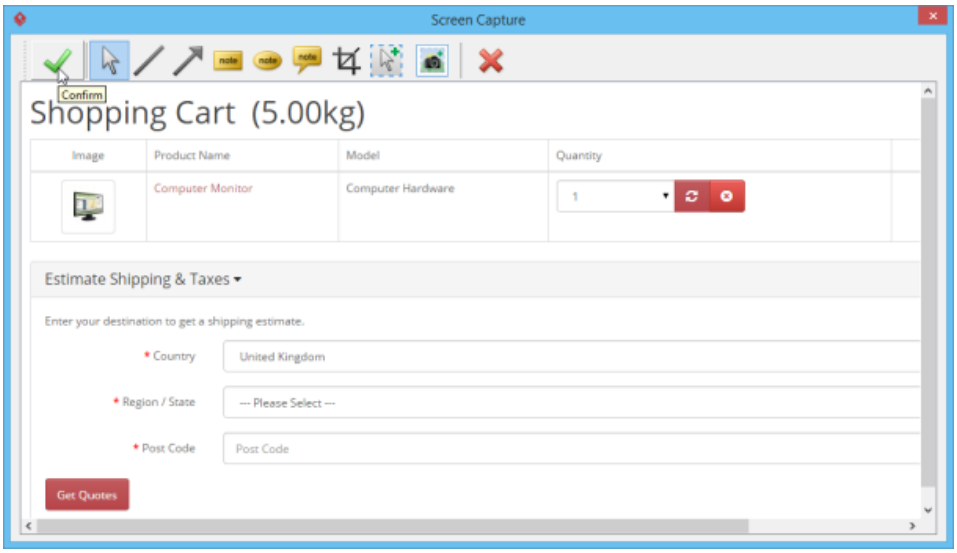
Capture screen

4. Release your mouse button to confirm the range. If you are not pleased with your selection, you can select again.
5. You can perform minor adjustment to the captured range. Press the **Up**/ **Down**/ **Right**/ **Left** key to adjust the position of the top left corner of the captured region, or press **Shift-Up**/ **Shift-Down**/ **Shift-Right**/ **Shift-Left** to adjust the position of the bottom right corner of the captured region.
6. When ready, click **Capture** on top of the selected range.



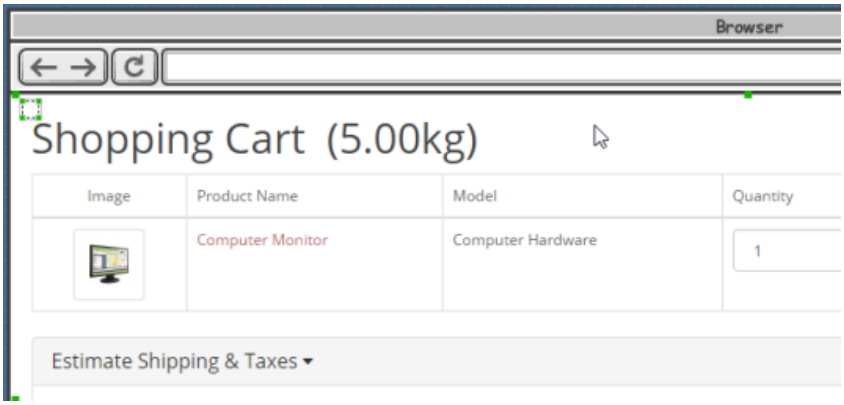
Confirm screen capture

7. Screen is captured. An image editor is opened for you to touch-up the captured content. You can optionally edit the captured screen in the **Screen Capture** window, like to add arrow and notes into the screenshot.
8. When ready, click **Confirm** to confirm the editing. Note that the screenshot is non-editable.



Confirm editing

9. The captured image is automatically placed inside the wireframe. You can drag it to move it to the place you want.

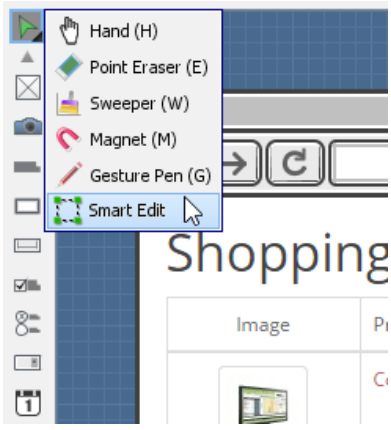


Repositioning screenshot in a wireframe

Using Smart Edit on captured screenshot

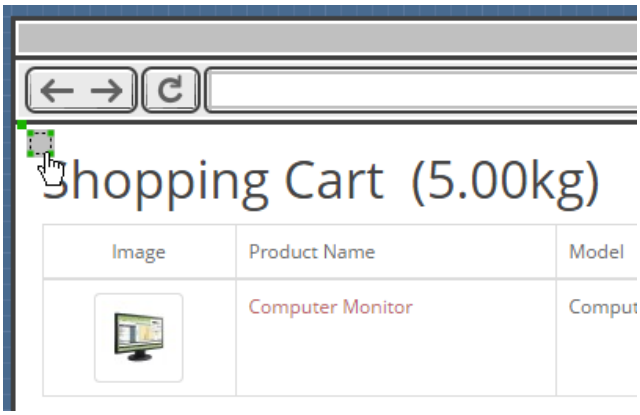
The **Smart Edit** tool lets you express screen design ideas by editing screenshots captured by the **Screen Capture** tool. There are two ways you can take to use the **Smart Edit** tool:

- Select **Smart Edit** from the diagram toolbar. The **Smart Edit** tool is grouped under the **Selector** tool. To use **Smart Edit** you need to press on the **Selector** tool to reveal the **Smart Edit** tool.



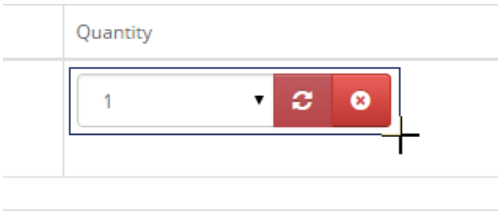
Selecting Smart Edit from diagram toolbar

- Click on the little **Smart Edit** resource icon at the top left of a screenshot.



The Smart Edit resource icon

Once you have selected the **Smart Edit** tool, drag on the screenshot to select the range to edit. For example, if you want to move a drop down menu in the screenshot, drag around that drop down menu.



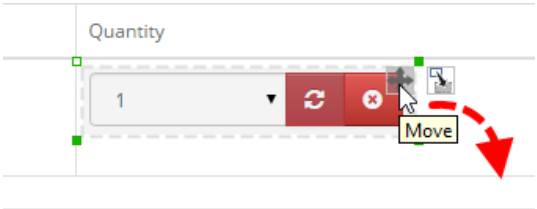
Selecting the range to edit

You can perform minor adjustment to the captured range. Press the **Up**/ **Down**/ **Right**/ **Left** key to adjust the position of the top left corner of the captured region. Then, press **Tab**, and then press **Up**/ **Down**/ **Right**/ **Left** to adjust the position of the bottom right corner of the captured region.

The following section describes the actions you can take to a selected range.

Reposition a screen component

You can move a component to somewhere else by dragging the **Move** resource at the top right of the editing region.

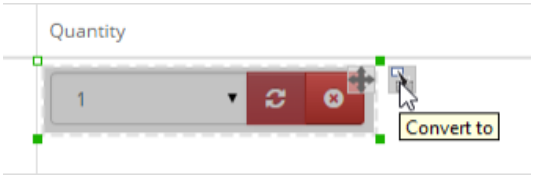


To move a selected range

Convert a screen component to another kind of component

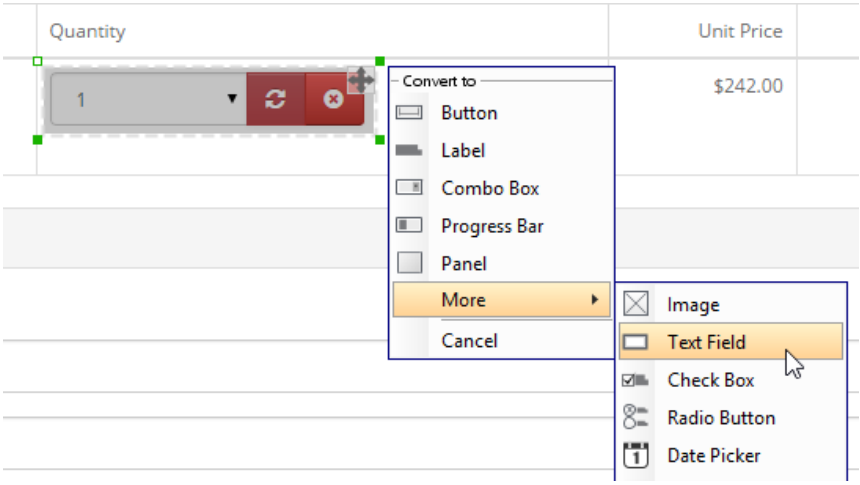
You can change a screen component to another type, say, to change a drop down menu to a text field by performing the steps below:

1. Click on the **Convert** resource at the top right of the editing region.



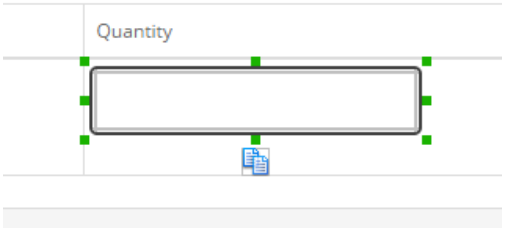
To convert a range into something

2. Select the type of widget to convert to from the popup menu.



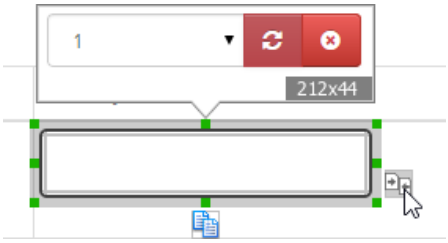
Converting a selected range to wireframe text field

Once converted you can edit the converted widgets like any other general wireframe widgets. For example, you can move it, resize it, change its color, enter its caption (for text field, button, etc).



Text field converted by Smart Edit

You may want to see the original look of a converted widget. To do this, move your mouse pointer over a converted widget and then click on the Compare resource to see the size and look of the original widget.



Tracing the source

Remove a component
If you want to remove a component, press delete on the editing region. The background will be filled by a colored that best matches the edge of the selected range.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Wireframe storyboard

Creating a wireframe storyboard

Visual Paradigm supports the organization and presentation of wireframes based on user-created storyboards. In this page you will learn how to create a wireframe storyboard as well as to add/create wireframes in a storyboard.

Renaming a wireframe storyboard

If you want to change the name of a wireframe storyboard, rename it by performing the steps described in this page.

Wireframe Playback in Wireframe Storyboard

In this page you will learn how to play the wireframes added to a wireframe storyboard as a slideshow.

Re-ordering wireframes in a wireframe storyboard

You can reorder the wireframes added to a storyboard by performing the steps described in this page.

Creating a wireframe storyboard

Wireframe is a sketch of user interface. It provides screen blueprints of the software system to be developed. By representing system screens with wireframes, customer can easily picture what will be developed even without any prototype or semi-product built. This facilitates the collection of early user suggestions in requirements gathering phase of software development, which in turns save cost and increase the quality of the end product.

Visual Paradigm supports the wireframing of website, desktop applications as well as mobile applications (Android and IOS). Not only do we support the sketching of wireframe but as a full-featured requirements tool, we also support the organization and presentation of wireframes as storyboard.

While wireframe itself represents a static screen blueprint,storyboard allows the dynamic representation of wireframes by grouping and ordering wireframes as storyboards. Take online banking as an example, one can add a storyboard called "Transfer Cash" and add the wireframes to the storyboard by representing the screen flow from account login to entering the target account ID, specifying the amount to transfer and finally executing the transferal command.

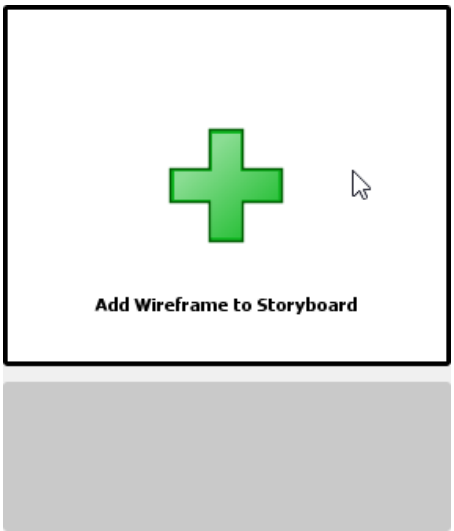
Same as use case based wireframing, wireframe storyboard also allows user to present the wireframes involved in form of a slideshow. This allows users to demonstrate screen flow to their customers easily.

To create a wireframe storyboard:

1. Select **Modeling > Storyboard** from the toolbar.
2. Click **Add New Storyboard**.

To add a storyboard

3. Input the storyboard name. You can treat a storyboard as a story. So if you want to represent the screen flow of the password retrieval process, you can name it *Password Retrieval*.
4. Click **Add Wireframe to Storyboard**.

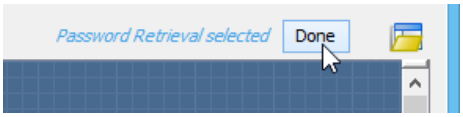


To add a wireframe to storyboard

5. If your project has no wireframe in it, you will be asked to create one. If the project contains a wireframe, you can now add an existing wireframe to the storyboard or just create a new wireframe. More about wireframe selection will be covered in the following sections.

The New Wireframe window

6. Create/select a wireframe.
7. Go back to the storyboard by clicking on the **Done** button on top of the wireframe.



Go back to the storyboard

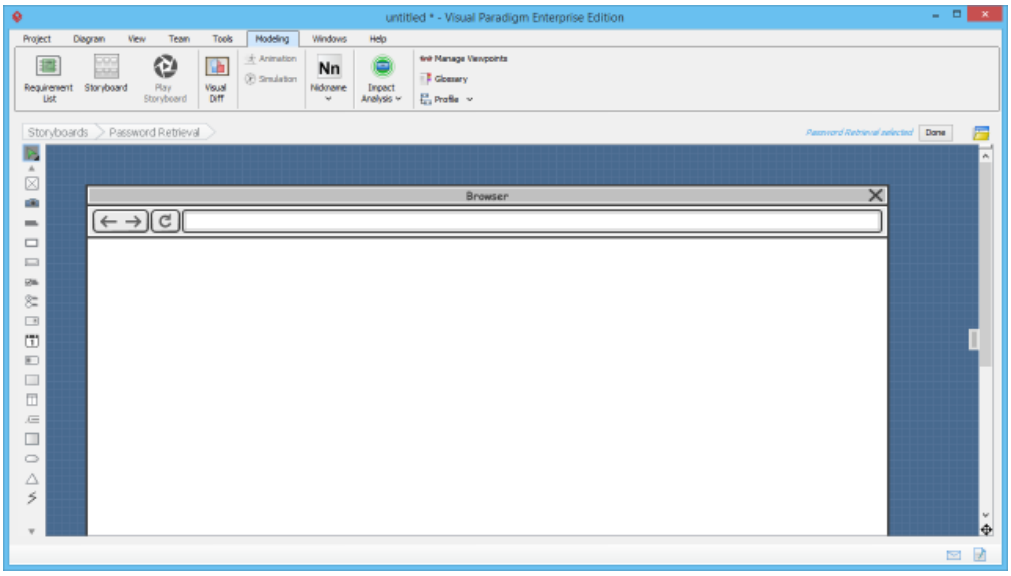
Creating a wireframe in a storyboard

To create a new wireframe in a storyboard:

1. Open **Storyboard** and select the desired storyboard.
2. Click **Add Wireframe to Storyboard...**.
3. If your project has no wireframe in it, you will be asked to create one. Select the suitable type of device/platform for your application/system. If your system will run on multiple devices/platforms, please consider creating multiple storyboards.

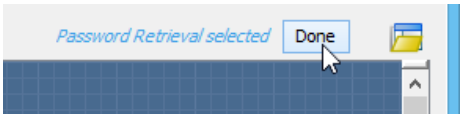
Select a type of wireframe to create

4. Click **New %TYPE% Wireframe** where **%TYPE% is** the type of device/platform you selected.
5. A blank, new wireframe appear and you can now begin editing.



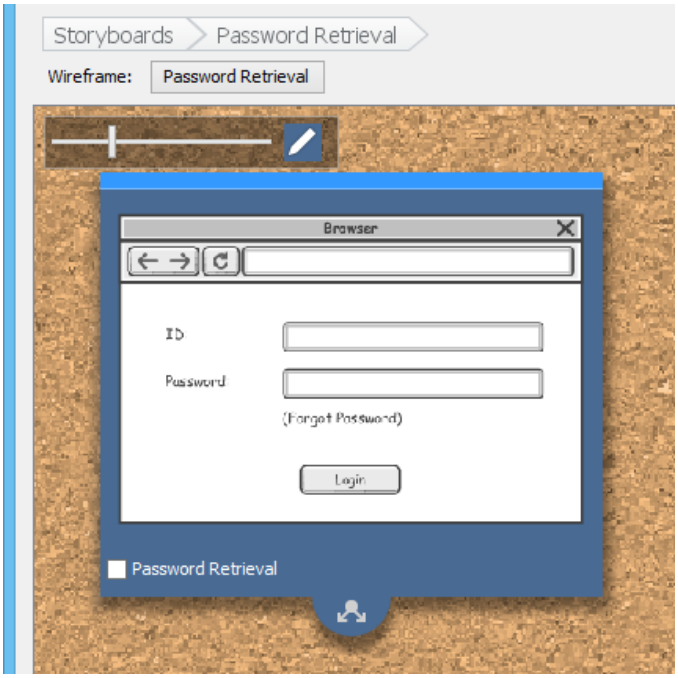
New wireframe created

6. When you finish editing, you can go back to the storyboard by clicking on the **Done** button on top of the wireframe.



Go back to the storyboard

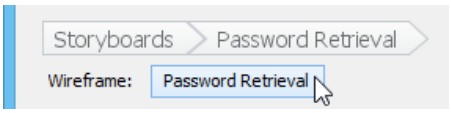
The above are the steps that involve in creating a wireframe from a storyboard when there is no wireframe in your project. Once you have created a wireframe, you will see something different after step 3. Here is what you will see:



State overview

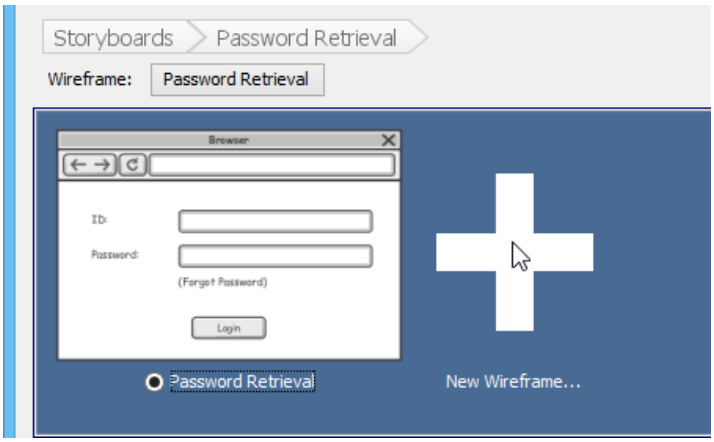
If you want to create an entirely new wirefram:

1. Click on the button next to **Wireframe:**.



Choose another wireframe

2. Click **New Wireframe...**.



Create a new wireframe

3. The remaining steps are same as those mentioned above, starting from step 3.

If you want to re-use an existing wireframe but make a bit of change, you should create a child state instead:

1. Click **Create Child State** below the thumbnail of wireframe to create a child state under it.

Create a child state

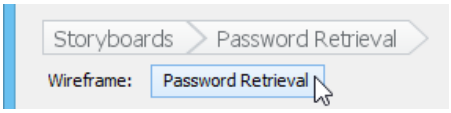
2. Once clicked, a new wireframe state will be created. You can start editing it.

Adding an existing wireframe to a storyboard

Sometimes, you may want to re-use a wireframe created earlier. For example, to reuse a wireframe about account login in storyboards that require user to login to do something.

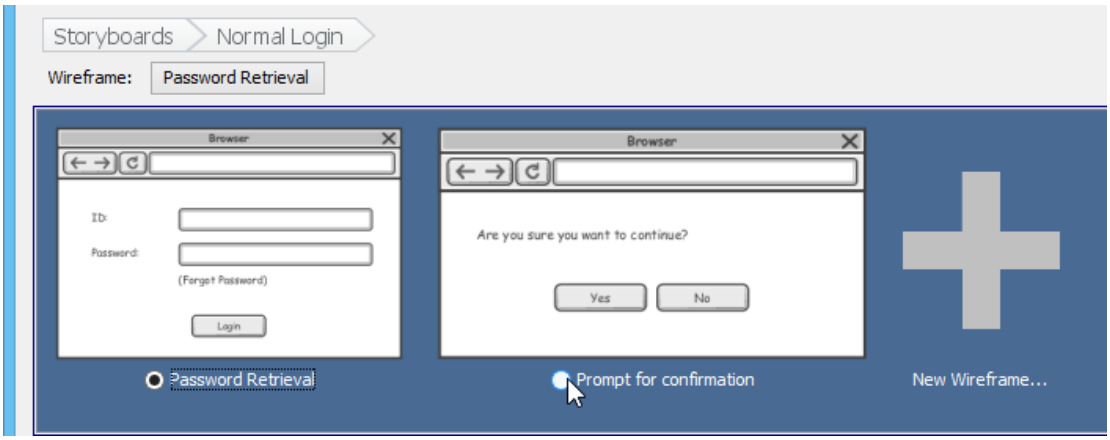
To add an existing wireframe into a storyboard:

1. Open **Storyboard** and select the desired storyboard.
2. Click **Add Wireframe to Storyboard**.
3. Click on the button next to **Wireframe:**.



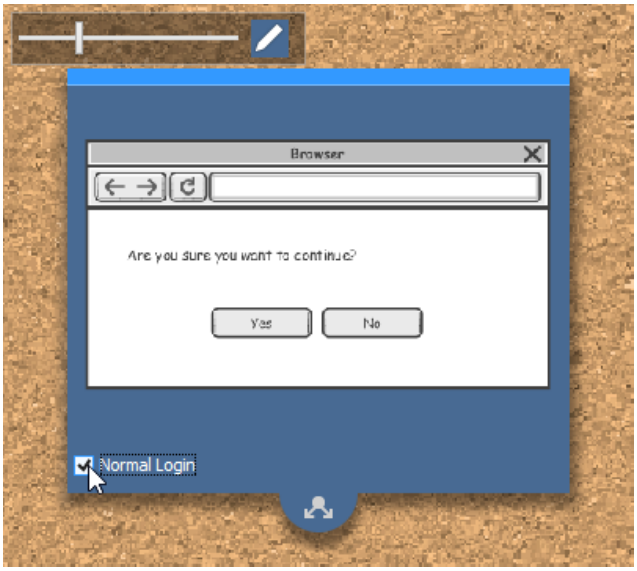
To select a wireframe

4. Choose the wireframe for your storyboard.



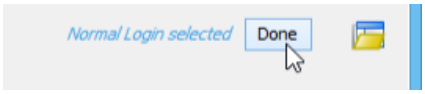
Choosing a wireframe

5. This shows the available states of the wireframe. Select the right one by checking the checkbox at bottom left corner. Make sure you do selected a state in this step. Without doing so, the wireframe won't be associated with the storyboard.



Selecting a wireframe state

6. Go back to the storyboard by clicking on the **Done** button on top of the state selection page.

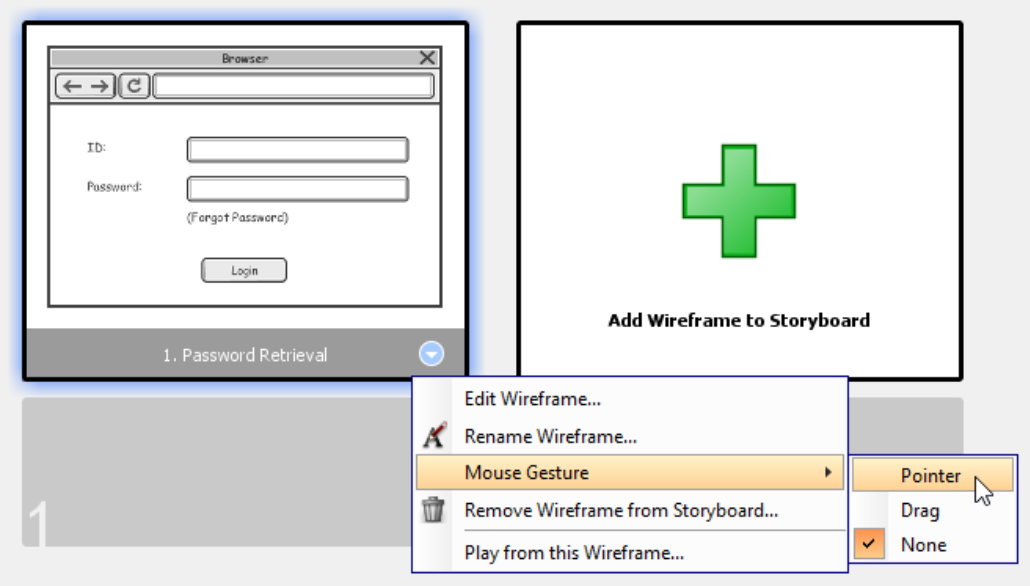


Go back to the storyboard

Using pointer / finger gesture

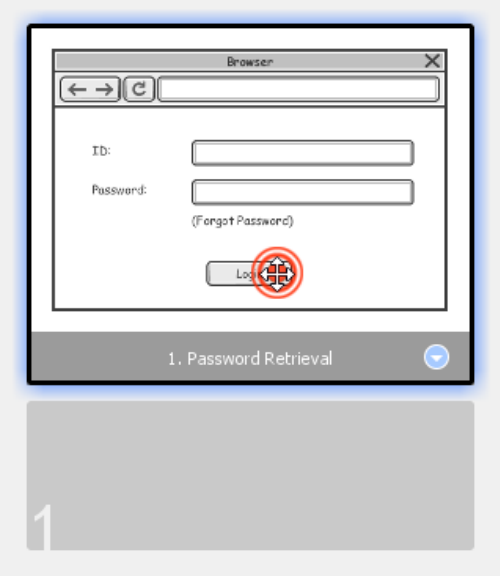
You can indicate in wireframe the position of mouse pointer, or the finger gesture required to execute an action. To do this:

1. Open **Storyboard** and select the desired storyboard.
2. There is a button at the bottom right of a wireframe. Click on it and select **Mouse Gesture > Pointer/Drag/Finger Gesture** from the popup menu. Note that Pointer and Drag are available for Desktop and Web wireframe, while Finger Gesture is available for Android phone, Android tablet, iPad and iPhone wireframes.



Add a pointer to wireframe

3. Drag on the pointer/drag/finger gesture symbol to reposition it.



Repositioning a pointer

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Renaming a wireframe storyboard

If you want to change the name of a wireframe storyboard, rename it by performing these steps:

1. Select **Modeling > Storyboard** from the toolbar.
2. Right click on the storyboard to rename and select **Rename Storyboard...** from the popup menu.
3. Click **OK** to confirm the new name.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

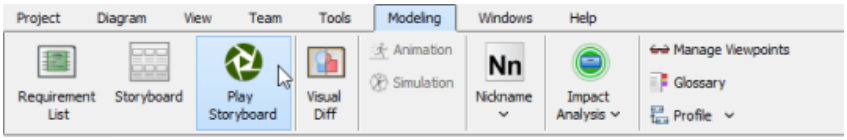
- Professional Edition
- Enterprise Edition

Wireframe playback in wireframe storyboard

Showing a screen flow of the system to your customer guarantees your customer knows what will be delivered by the end of the project. Visual Paradigm not only allow you to associate use case scenario with wireframes in illustrating system interations, but also supports playing the wireframes associated with wireframe storyboard. This can be very useful when you need to present the system design ideas to your customers, and to look for their consent.

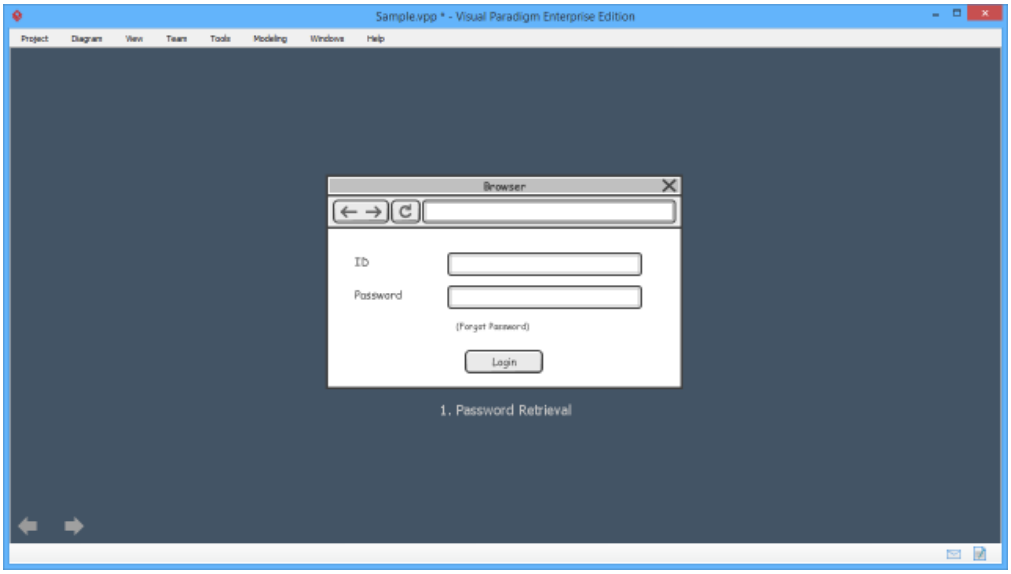
Playing wireframes

1. Select **Modeling > Storyboard** from the toolbar.
2. Select the storyboard to play.
3. Click on **Modeling > Play Storyboard** from the toolbar.



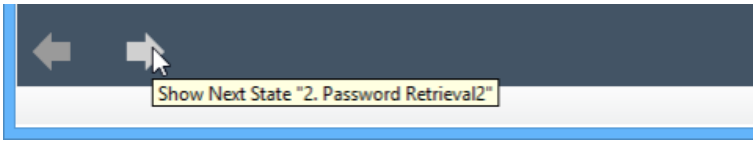
Play storyboard

4. This opens the wireframe player. The wireframe of the first step is shown in the player.



Playing a wireframe

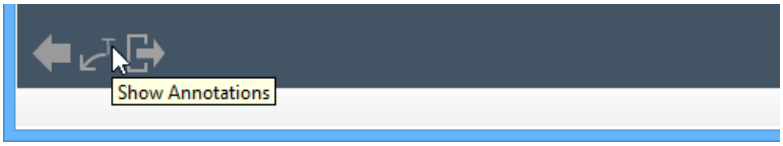
You can move on to the next wireframe by pressing the **Right** key, or clicking on the arrow button at the bottom left of the player. Similarly, you can press the **Left** key or click on the back button at the bottom left of the player to move to the previous wireframe.



Move to the next wireframe

Showing Annotations

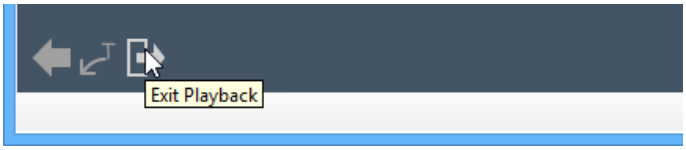
To keep the wireframe clear and readable, annotations are hidden by default. You may click on the Show Annotations button t the bottom left of the player to have them visible. Let's show the annotations.



Show annotations

Ending the Show

You can end the show anytime by pressing the **Esc** key. When it arrives the final wireframe, you can also exit by clicking on the Exit button at the bottom left of the player.



Exit playback

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- Tutorial - Writing effective use case
- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

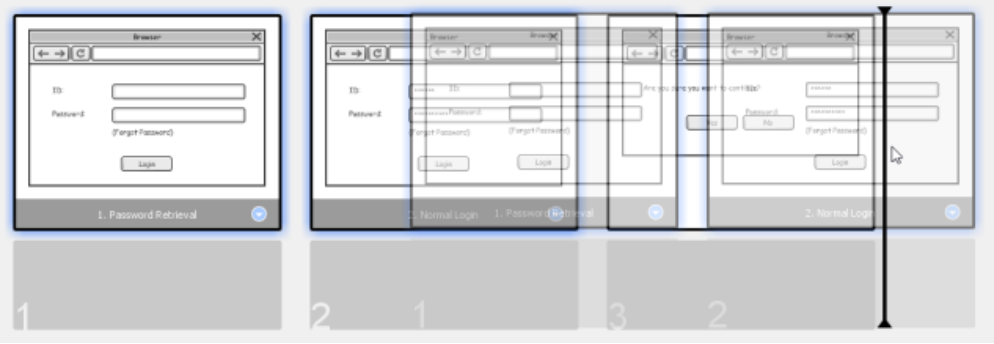
Re-ordering wireframes in a wireframe storyboard

You can always reorder the wireframes added to a storyboard by performing these steps:

1. Select **Modeling > Wireframe Storyboard** from the toolbar.
2. Double click on the desired storyboard to open it.
3. In the center of the editor, select the wireframe(s) to reorder. You can perform multiple selection by first pressing **Ctrl** on a wireframe, and then select the other wireframes one by one.

Select the wireframes to reorder

4. Drag your selection to the target place.



Reordering wireframes

5. Release the mouse button.

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- YouTube Video - How to Create Scenario-Based Wireframe?
- YouTube Video - How to Present Wireframes?
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Supported Editions

- Professional Edition
- Enterprise Edition

Introduction of impact analysis

Impact analysis is the technique to find out what the potential influences are when updating a design blueprint. This chapter provides you with general understanding about impact analysis.

Introduction of impact analysis

Describe the several ways of impact analysis supported by Agilian.

Introduction of impact analysis

What is impact analysis?

Impact analysis is the technique to find out the potential influences that may happen when updating a design blueprint. For example, when we want to update the use case model, we may also want to update the sequence diagram which model how to achieve the user's goal. There are two ways of performing impact analysis in Visual Paradigm, analysis diagram and matrix.

How does impact analysis improves your work?

Impact analysis helps to avoid unexpected consequences resulted by updating your design blueprint. Contrary to this, it allows you to find out everything you need to update along when a change is to be made.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Analysis diagram

By analyzing a model element, you can know its relationships with other elements. This chapter shows you how to analyze things by forming and reading an analysis diagram.

Analyzing a model element

Shows you how to analyze a shape.

Updating analyzed result

When the model keep growing, you may need to update the analysis diagram to reflect the latest project content.

Grouping of nodes

Instead of having many many nodes show on analysis diagram, you may want to group nodes of same type to make the diagram tidier.

Opening view from node

Shows you how to open a view from a view node.

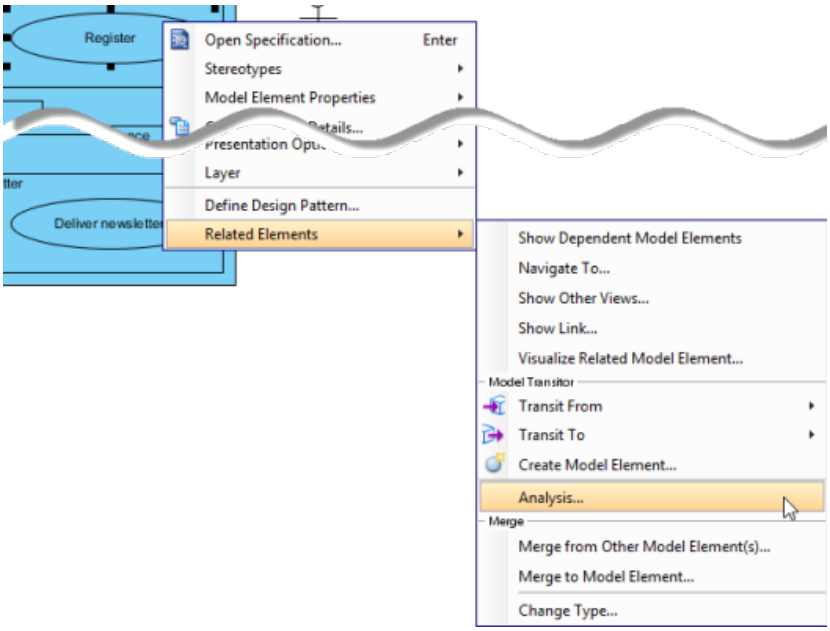
Analyzing a model element

We analyze a model element when we want to identify its related elements so as to foresee the potential impact that may cause on the model resulted by modifying the model element. The term "related" here represents any kinds of connection that can have between two elements, such as a general to-and-from relationship, a parent-child relationship, transitor, or even a sub-diagram relationship with a diagram.

To analyze a model element

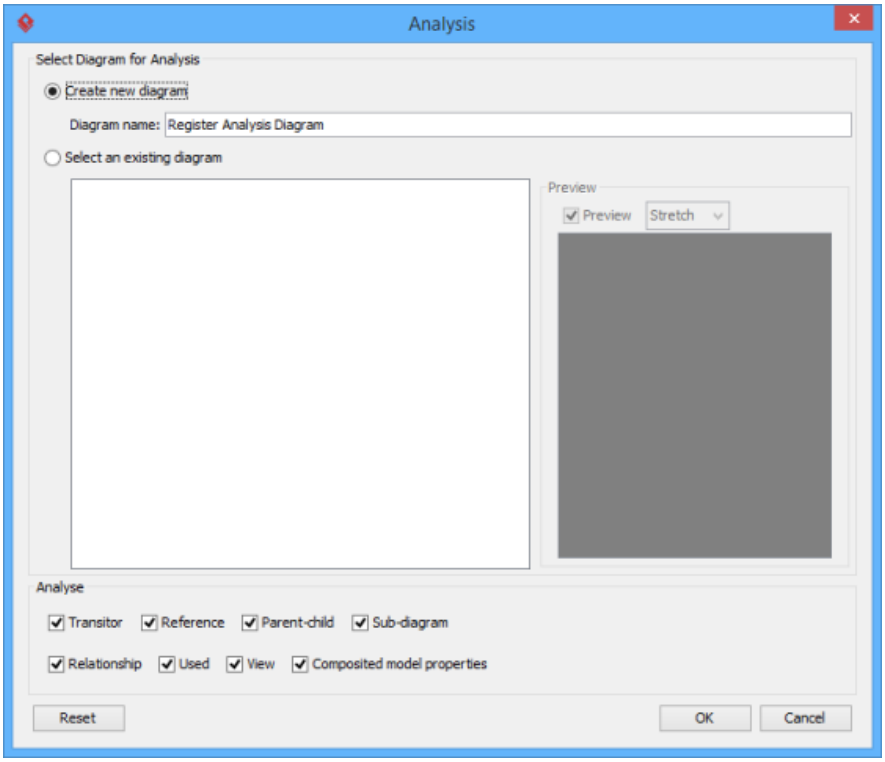
By analyzing a model element, you can know its relationships with other elements. To analyze:

1. Right click on the diagram element we want to analyze and select **Related Elements > Analysis...** from the pop-up menu.



Analyze a diagram element

2. The result of analysis will be presented in analysis diagram. In the **Analysis** window, either select **Create new diagram** to present the result in a new analysis diagram or select to present in an existing analysis diagram. The check boxes at the **Analyse** section governs the type(s) of relationship to be analyzed. Click **OK** when ready.



Create a diagram, or select an existing analysis diagram to present the result

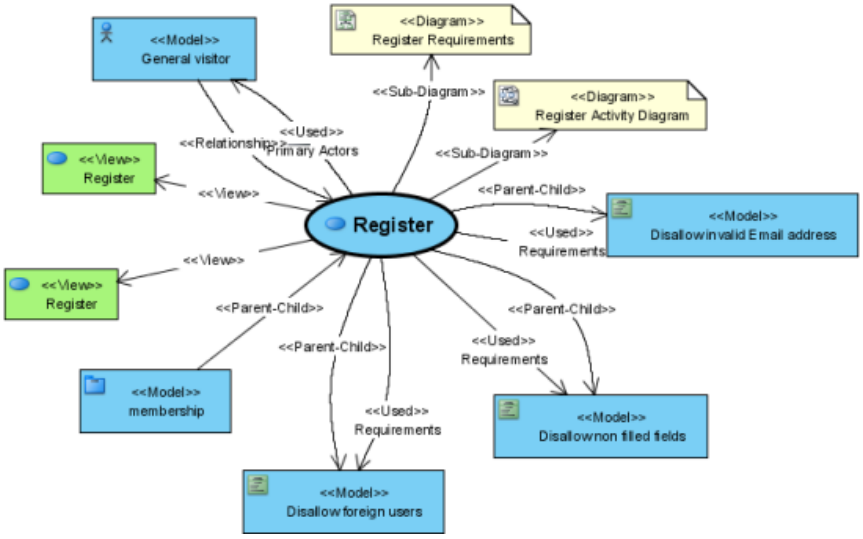
Type	Description
Transitor	The transited element of the chosen element or the element where the chosen element was transited from
Reference	The shape or diagram references of the chosen element

Parent-Child	The parent (e.g. package) or the child of the chosen element
Sub-diagram	The sub-diagram(s) of the chosen element
Relationship	The relationship(s) of the chosen element, such as association, dependencies, sequence flow, etc
Used	The connection with the chosen element, other than any other kinds of relationship types. For example, requirement owned by use case added through use case description is a kind of Used relationship.
View	The view(s) of a model element, which can be seen as the shapes of a model element

Kinds of relationships that can analyze

Reading analysis diagram

The result of analysis is shown in an analysis diagram.



An analysis diagram

The oval node at the center of the diagram represents the element you have chosen to analyze, the connectors branching out are the relationships with the analyzing element and the nodes at the opposite end of connectors are the related elements.

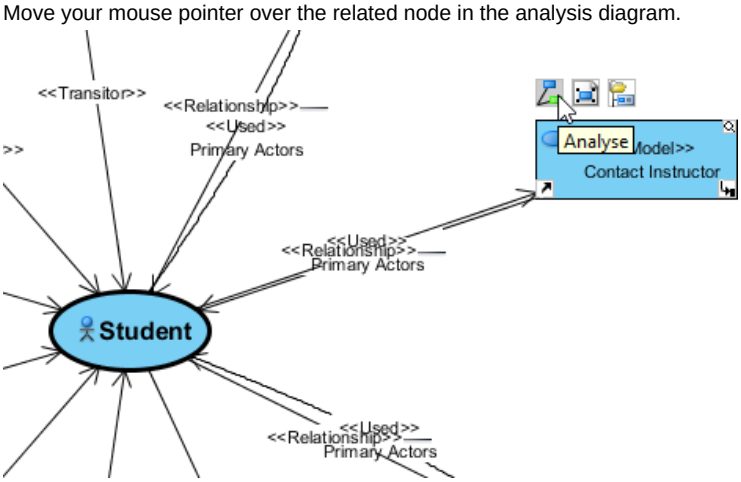
Inside a node of a related element, you can see a tag (e.g. <<View>>) which represents the type of that related element. At the bottom part of a node box is the name of the related element.

By reading the diagram, you can identify the relationship of a model element and determine the impact that may act upon the model when modifying the model element.

Analyze further

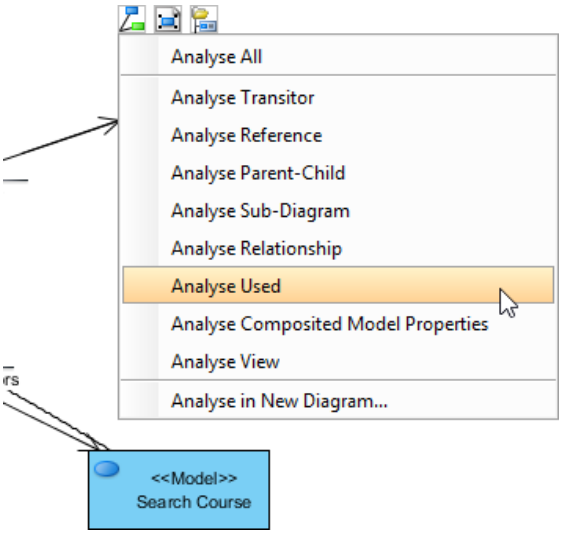
Analysis diagram allows you to visualize the relationships between a model element and its related elements. If you want to visualize the relationship between a related element with its related elements, you can analyze further by performing the steps below:

1. Move your mouse pointer over the related node in the analysis diagram.



To analyze a use case

2. Click on the **Analyze** resource.
3. Select the type of relationship to be analyzed.



Analyse Used

Note that by analyzing mutliple model elements on the same analysis diagram, the diagram may contain a lot of nodes and connectors, making it hard to read and understand. To solve this problem, you may consider showing the result in another analysis diagram by selecting **Analyse in New Diagram...** in the step above.

Related Resources

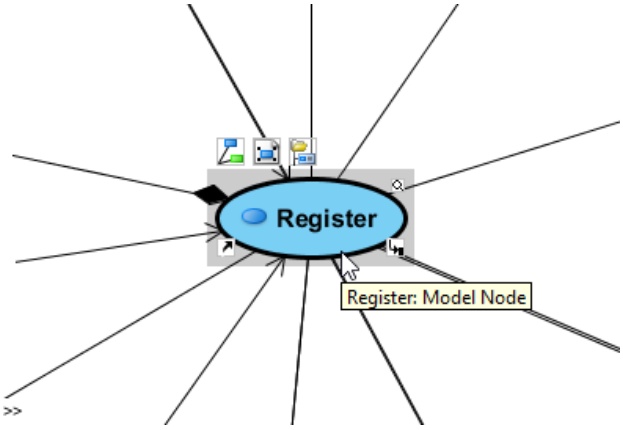
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Updating analyzed result

Once a model is refined, the previous analysis result may no longer reflects the latest changes. Therefore, you need to update the analysis result in order to perform impact analysis for the chosen element on the latest model.

1. Move the mouse pointer over the node that you want to update its relationships with others.



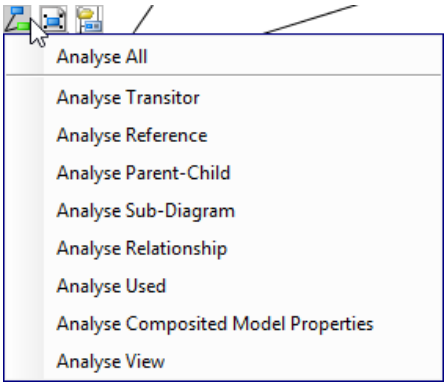
To analyze a diagram element

> **NOTE:** Besides the central node, you may update/show the relationships of other Model nodes on diagram, too.

2. Click on the **Analyze** resource icon.

To click on Analyse

3. Select a type of relationship to analyze.



To select a type of relationship to analyze

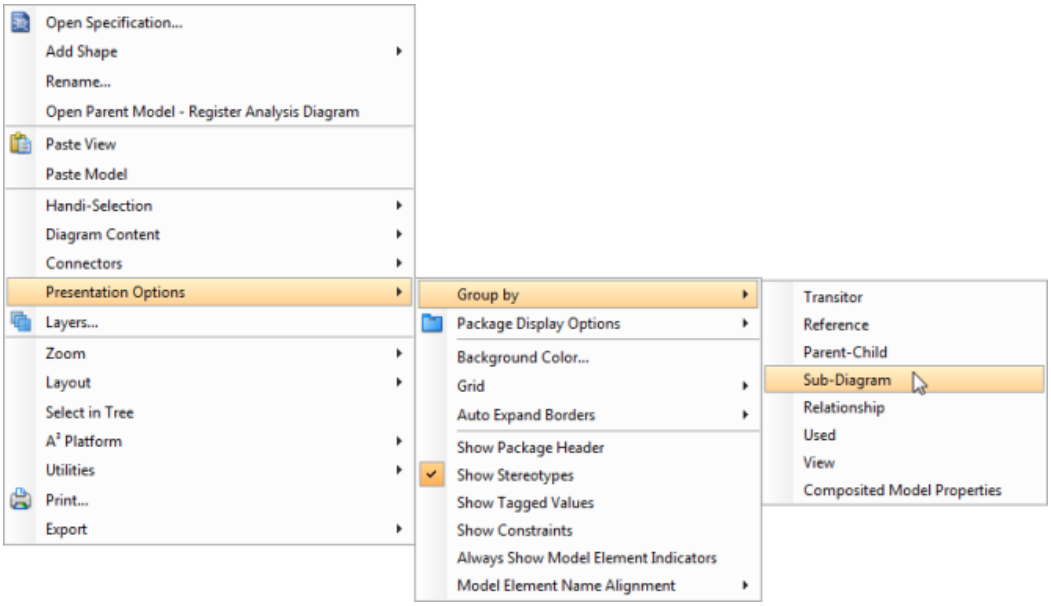
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

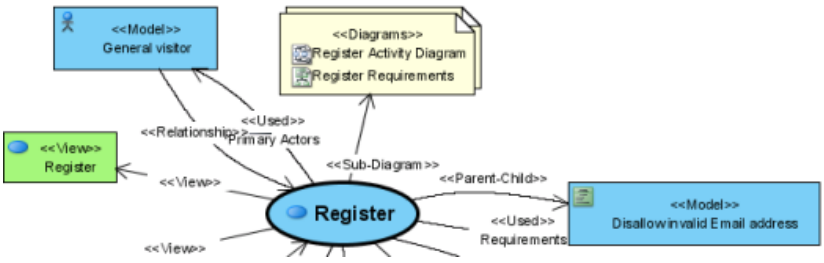
Grouping of nodes

To make it easier to identify the relationships of a chosen element, you can group all relationships of a particular kind into a single node, eliminating the connectors being shown on diagram. To group, right click on the background of analysis diagram and select **Presentation Options > Group By** and then the type of node from the popup menu.



Select a kind of relationship to group by

Nodes of same type are grouped.



Diagrams are grouped

To ungroup, simply deselect the node type by walking through the same popup menu path.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

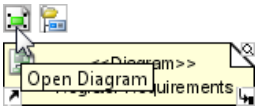
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening view from node

When reading an analysis diagram, you may find the existence of related model element or related diagrams, such as sub-diagram of the chosen model element. You can open the view of such related elements through the resource icons appear on top of the nodes.

Opening a diagram of diagram node

1. Move the mouse pointer over a **Diagram** node.
2. Click on the **Open Diagram** resource icon.

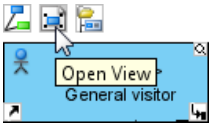


To open diagram of Diagram node

This opens the diagram.

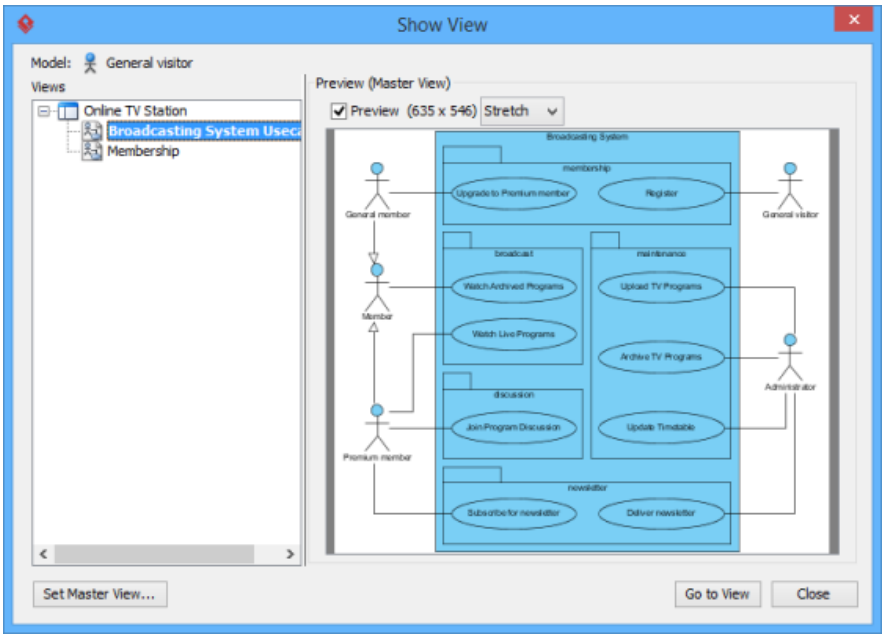
Opening a view of model node

1. Move the mouse pointer over a **Model** node.
2. Click on the **Open View** resource icon.



To open view of a Model node

3. If the target model has only one view, that view is opened. If there a multiple views, the **Show View** window is presented. Select a view to open and click **Go to View** at the bottom right of window.



*The **Show View** window*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Matrix diagram

A matrix is a table, which shows the relationships among model elements of particular types. By reading a matrix, you can tell easily whether two model elements are related or not, and what kind of relationship do they have. This chapter not only tells you how to create a matrix but also how to read it, to get the information you need.

Creating a matrix

Shows you how to create a matrix.

Reading a matrix

Explains each part of a matrix in detail.

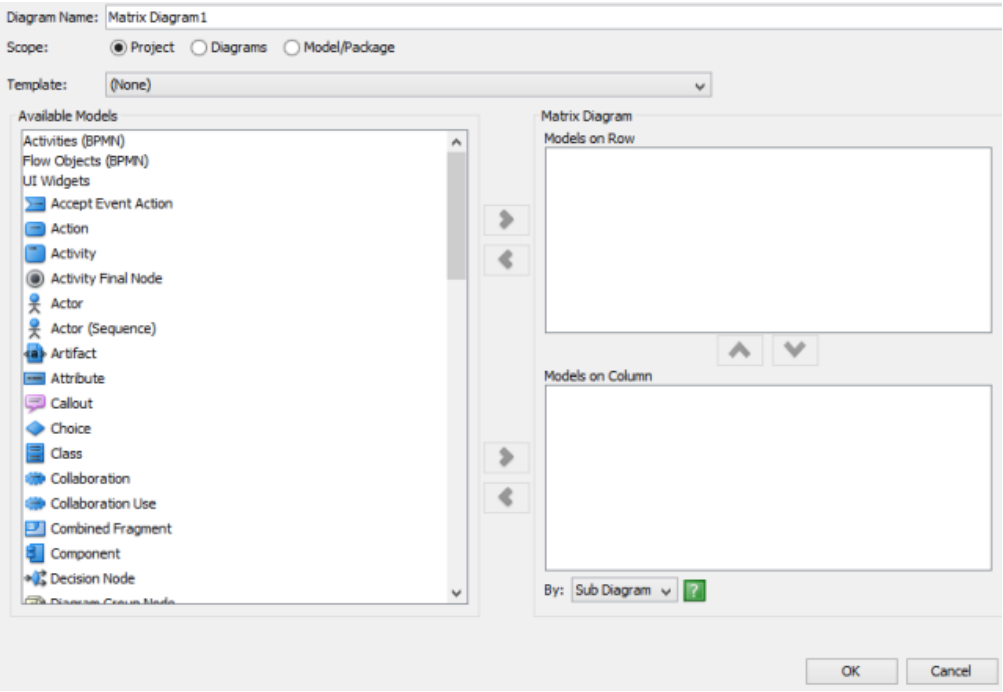
Showing the Use of Terms with Matrix

Shows you how to present the the use of terms by forming a matrix.

Creating a matrix

A matrix is a table, which shows the existence of relationships among model elements of particular types. By reading a matrix, you can tell easily whether two model elements are related or not and what kind of relationship they have.

1. Select **Modeling > Impact Analysis > Matrix** from the toolbar.
2. Configure the matrix.



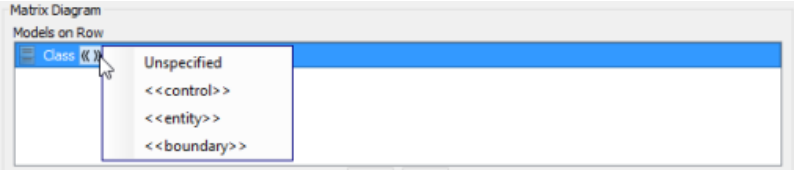
Configure matrix

Field	Description
Diagram Name	The name of diagram which is also the name of matrix.
Scope	The source of model elements to compare in matrix, in **Project** (all model elements), **Diagram** (only model elements in specific diagrams which are selected by users) or under **Model/Package**.
Template	Template offers a default setup to **Models on Column**, **Models on Row** and **By**. It is available according to the project content. For example, template "Use Case <-> Requirement" appears for selection when a project have use case and requirement.
Available Models	All available models in your selected scope are listed here. You can select a model to add it in the target Models on Row and/or Models on Column.
Models on Column	The type of model element to list at the column side of matrix. In order to list multiple types of model element, select it/them on **Available Models** and click  button to insert it/them in here.
Models on Row	The type of model element to list at the row side of matrix. In order to list multiple types of model element, select it/them on **Available Models** and click  button to insert it/them in here.
By	The way how matrix will match against rows and columns. **Sub Diagram** - The column/row model element is placed in a sub diagram of the matching model element. **Child** - The column/row model element is a child of the matching model element. **Relationship** - The column/row model element is related with the matching model element. You can specify the kind of relationship between model element, as well as the stereotype, if any, assigned to the relationship. Besides, if you choose to match by **Relationship**, you will see the button **Skip in Relationship...**. This button allows you to ignore specific kind(s) of model elements when considering if two elements are related or not. It's particularly useful when two elements are conceptually connected with each other but with an intermediate element in between. For instance, you may have two components connected with each other, but with an interface in between as the provided and required interface of the components. Without skipping the interface (class) in between, the two components will not be considered to be "related", but by skipping class, which is the real element type of an interface, classes will be ignored and thus, the two components will be considered as "related". **Reference** - The column/row model element is referencing the matching model element. **As Classifier** - The column/row model element takes the matching model element as classifier. **Transitor** - The column/row model element is transited from/to the matching model element. **Dependent** - The column/row model element has properties that depend on the matching model element. "Depend" means any of the following: (1) With any kind of relationship such as association, dependency, etc. (2) As a contained element (3) As a link in description content (4) As a selected value for any property. **Using Term** - The column/row model element is a term (element) and is indirectly used by the matching model element, either or both as part of its name or in its description.

3. Click **OK** button to form the matrix.

Filtering rows and columns by stereotype

Instead of listing all elements with same type in row or column, you can fitler them further by selecting only elements with stereotype assigned by listed. For example, you can form a matrix to visualize the relationships between <<control>> and <<entity>> classes. To fitler, move your mosue pointer over an assigned element type in the configuration screen. Click on the **<<>>** button and select the stereotype.



Using filter

Related Resources

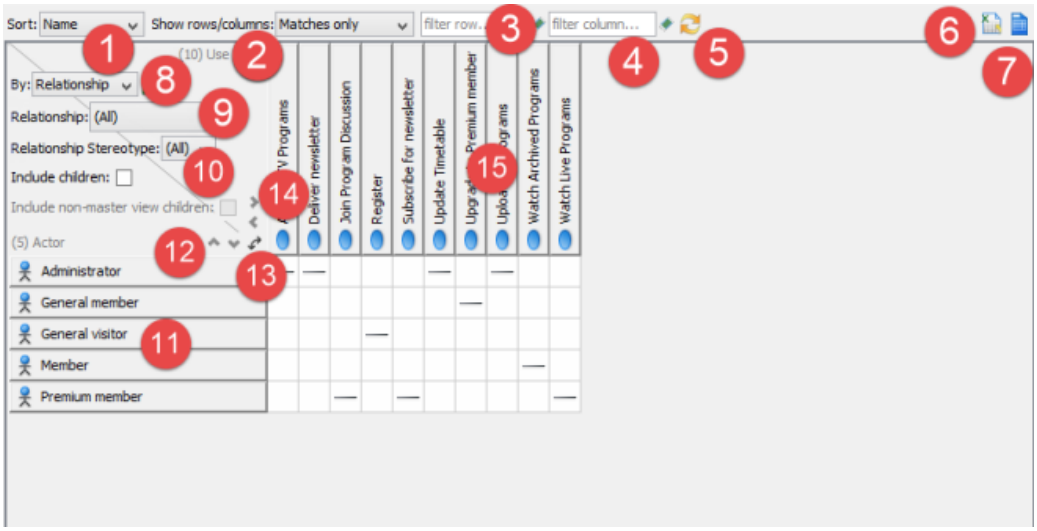
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reading a matrix

Knowing how to read a matrix helps you to understand your model better and to perform further modifications more comfortably. In this chapter, we will see how to read a matrix and how to refine the matrix content with the help of functionalities like hiding columns and rows and filter.

A matrix is a table with rows and columns, both represent sets of model elements of specific types. A cell in table is an intersection of a row and a column which reflects the relationship of the row and column. If the cell is filled either by a tick or by a kind of relationship (when a matrix was set to match things by Relationship), this means that the model elements of the row and column are related. The type of relationship can be checked by referring to the drop down menu at the top left of matrix.



Overview of a matrix diagram

No.	Name	Description
1	Sort	Adjust the way elements are ordered in rows and columns - by name, stereotype or ID
2	Show only rows/columns with matches	Matrix lists only rows and columns with matches by default. Uncheck it when you want to show entries without matches as well.
3	Filter row	Type the full name of a model element or part of it that you are looking for to narrow down the searching field in rows when too much data is displayed.
4	Filter column	Type the full name of a model element or part of it that you are looking for to narrow down the searching field in columns when too much data is displayed.
5	Refresh	You can update the content of matrix by clicking this button manually, for reflecting the changes you have made in models.
6	Export to Excel	Click this button to export the opening matrix to Excel.
7	Configure	Click this button to configure the content to display in matrix.
8	By	It shows how matrix will match against rows and columns.
9	Relationship	Select the kind of relationship to be considered.
10	Relationship Stereotype	Select the stereotype assignedto relationships.
11	Rows	The model elements have been chosen will be displayed in rows.
12	Move up and move down	Rows and columns are ordered alphabetically by default. They help to re-order rows and columns in moving vertically.
13	Swap rows and columns	It helps to change the presentation between rows and columns, but not switch the actual relationships between model elements being represented by them.
14	Move left and move right	Rows and columns are ordered alphabetically by default. They help to re-order rows and columns in moving horizontally.
15	Columns	The model elements have been chosen will be displayed in columns.

Description of matrix diagram

Exporting Excel

You can export Excel file from matrix, and analyze relationships between model elements in worksheet in Excel. To export Excel:

1. Click **Export to Excel** above the matrix, near the **Configure** button.

To Click Export to Excel

2. In the **Export Excel** dialog box, specify the output destination and click **Save**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

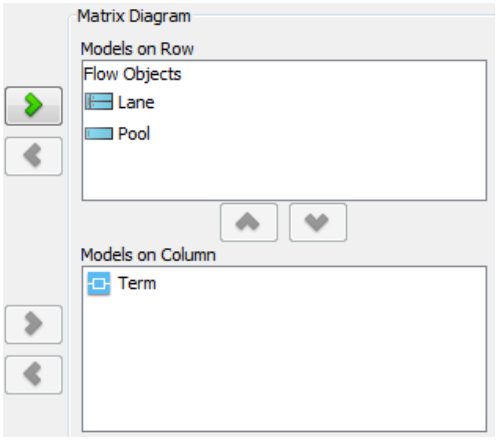
Showing the use of terms with matrix

Glossary is a function to allow you to identify important terms from name and description of model elements, so that you can build a glossary with terms and describe them in detail. For details about the use of glossary, please read the page Identify glossary term.

When you want to access the elements whose names or descriptions have terms included, you may form a matrix to illustrate the relationships between elements and terms. You can even see how frequent the terms are referred to in different model elements in a matrix view.

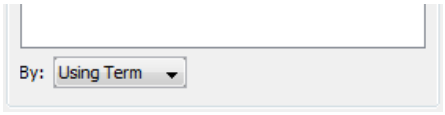
To create such matrix:

1. Select **Diagram > New** from the toolbar. In the **New Diagram** window, select **Impact Analysis > Matrix Diagram**. Click **OK** to confirm.
2. Configure the matrix. If you want to place the terms at the top of matrix, add **Term** to **Models on Column**. If you want to place the terms on the left hand side of matrix, add **Term** to **Models on Row**.



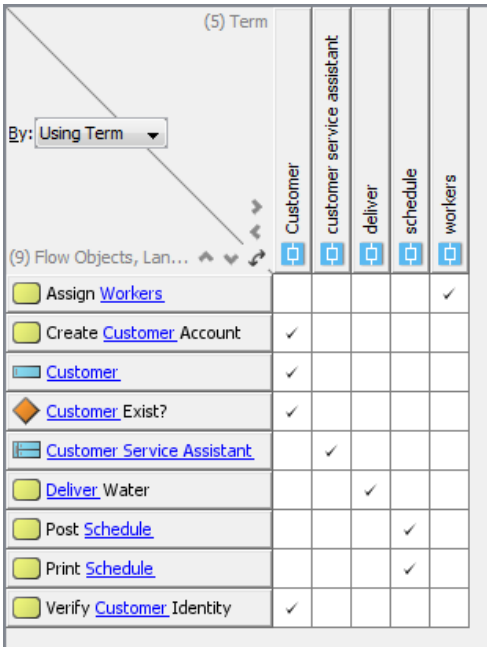
Elements are selected for row and column

3. Select **Using Term** under the drop down menu **By**.



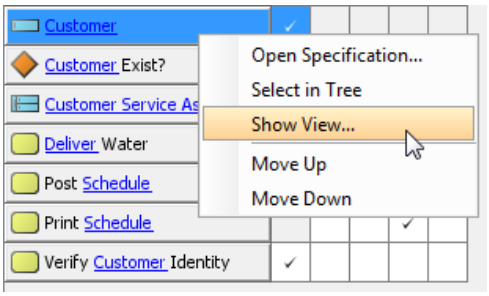
Select Using Term

4. Click **OK**. The ticks show in the matrix indicate the use of term for specific element.



Matrix formed

5. If you want to open a view of a model element, right click on the element and select **Show View...** from the popup menu. in the **Show View** window, select the diagram to open and click **Go to View**.



Show view of pool

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Chart diagram

Chart diagram enables you to specify and show the relationship between model elements by mean of chart. This chapter shows you how to create and configure chart.

Developing a chart

Shows you how to create and develop chart.

Configuring a chart

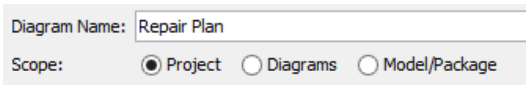
Shows you how to configure chart.

Chart diagram

Visual Paradigm enables you to build a chart. In addition to the built-in RACI chart available for general purposes, you can define your own type of chart for problem-specific purposes. In this page, you can learn how to develop a chart, define color for roles and define a new code type.

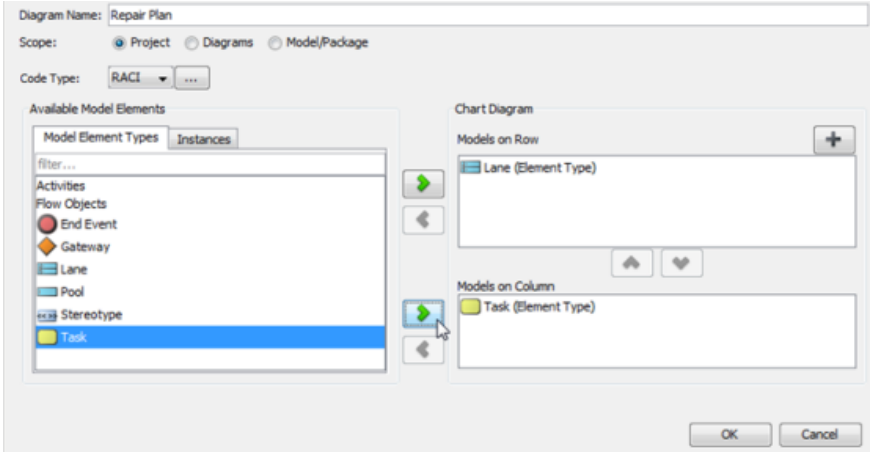
Developing a chart

1. Select **Modeling > Import Analysis > Chart** from the toolbar.
2. Name the chart diagram.



Name the chart diagram

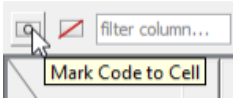
3. Select a model element type for row and column respectively to identify which participant is involved in an activity.



Select a model element type for row and column

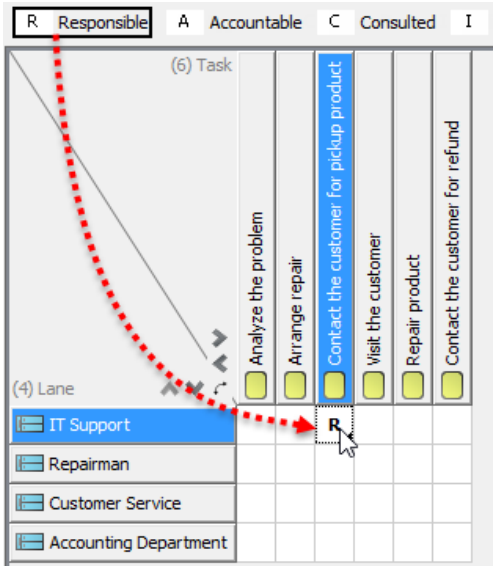
> **NOTE:** The choice of model element types will be in accordance with your exisiting model elements in the same project.

4. Click **OK** button.
5. You can then assign the role to participant(s) by clicking **Mark Code to Cell** button on the top of chart.



*Click **Mark Code to Cell** button*

6. When the roles reveal, click the target role and select the target cell to assignment.



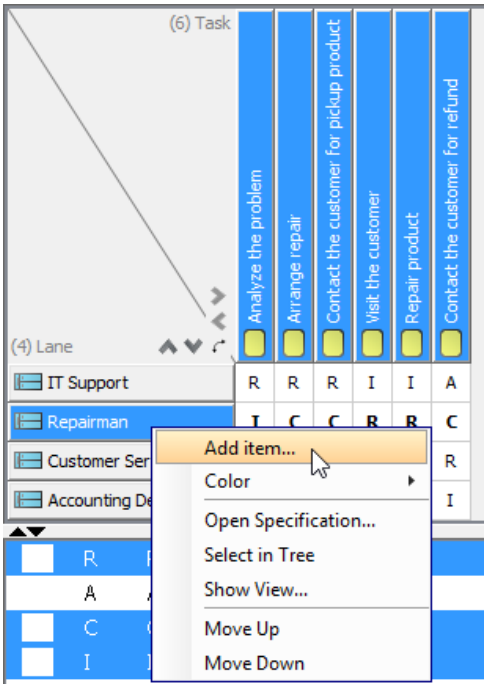
Assign role

Adding an item

Apart from selecting the existing model elements on chart, you can also create objects without model view. Those objects will have the similar properties with model elements that you can assign role to them.

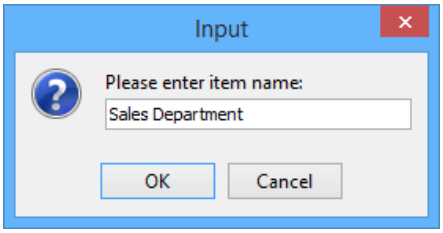
To create an item without model view:

1. Right click on a model element where you want to insert an item in font of that model element and select **Add item...** from the pop-up menu.



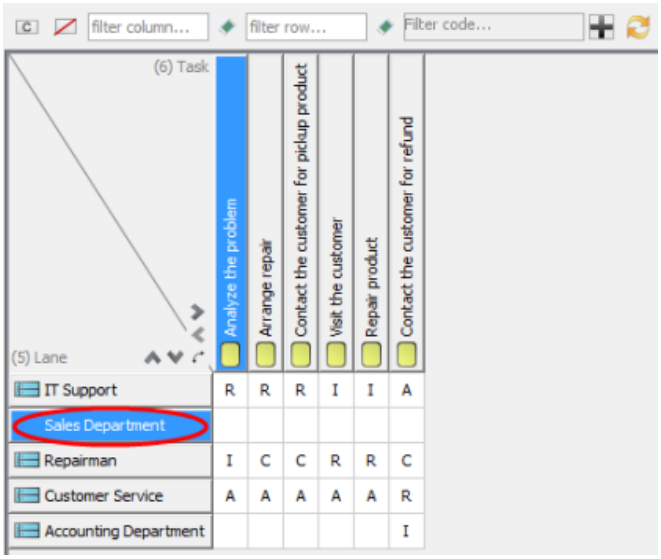
Add an item

2. Enter name for new item in the pop-up **Input** dialog box. Click **OK** button to confirm and close the dialog box.



Enter item's name

As a result, the newly created item is shown.



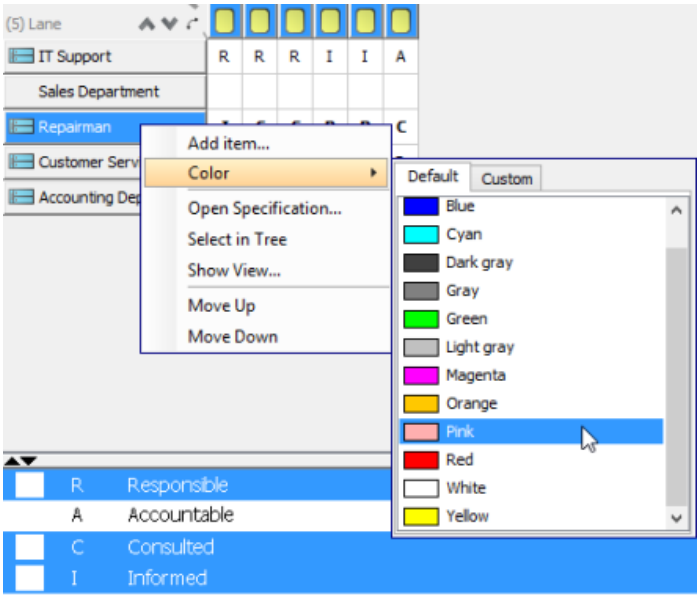
The newly created item is shown

Specifying background color for column/ row

The background color of columns and rows are white by default. You can specify background color for the entire columns/ rows by right clicking on the target row/ column and then selecting color for it.

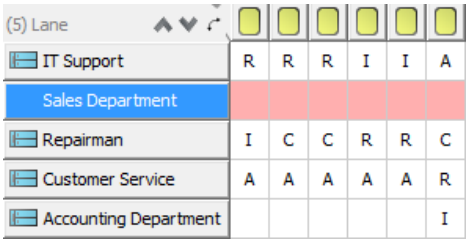
To set background color for column/ row:

Right click on the target row/ column and select **Color** > [target color] from the pop-up menu. You can select a color from either **Default** category or **Custom** category.



Select a background color

As a result, the background color for selected row/ column is set.



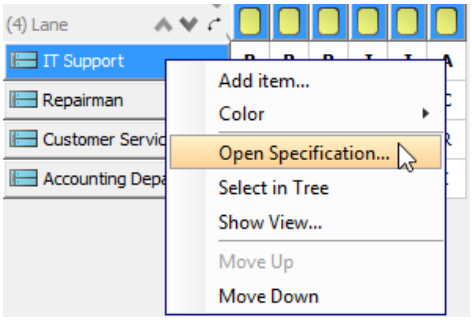
Background color is set

Opening model element's specification

You can view and specify the properties of model elements by opening their specification dialog box.

To open a model element's specification:

1. Right click on the target model element and select **Open Specification...** from the pop-up menu.



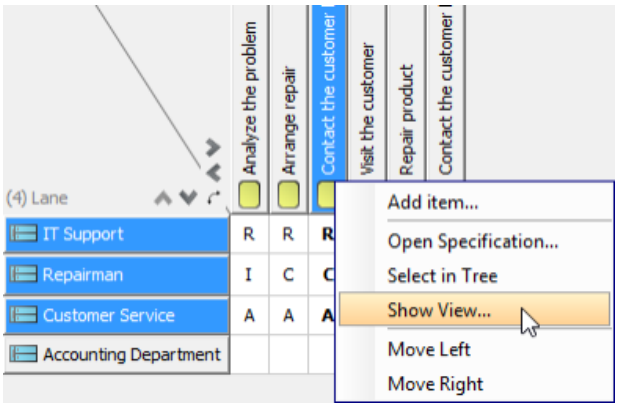
Open Lane's Specification dialog box

2. In specification dialog box, specify its properties, for example, description, references, tagged values and comments. Click **OK** button.

Showing the view of model element

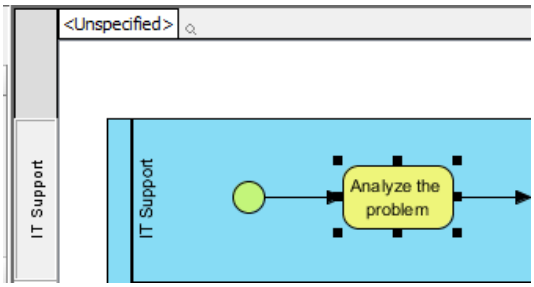
When reading the chart, you may want to view the model element on its source diagram.

You can view the model element by right clicking on the target model element and selecting **Show View...** from the pop-up menu.



Show the view of model element

As a result, the view of model element is selected on the diagram.

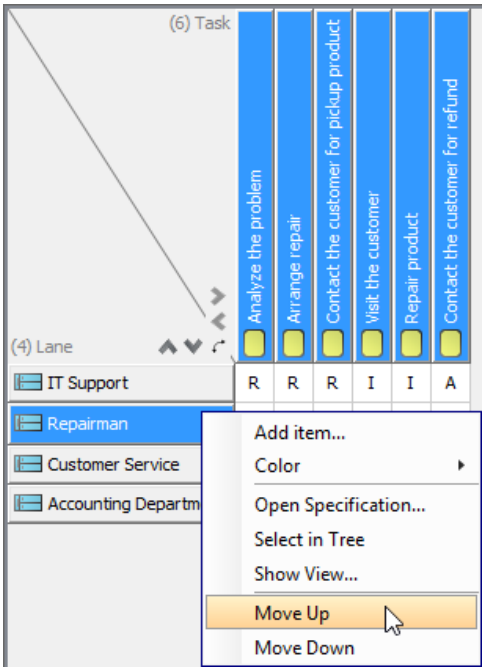


Model element is selected on its source diagram

Reordering row/ column

You can arrange the order of model elements with your preference.

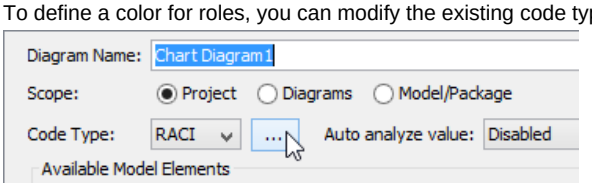
Right click on target row/ column and select **Move up** or **Move Down** from the pop-up menu.



Move the model element up

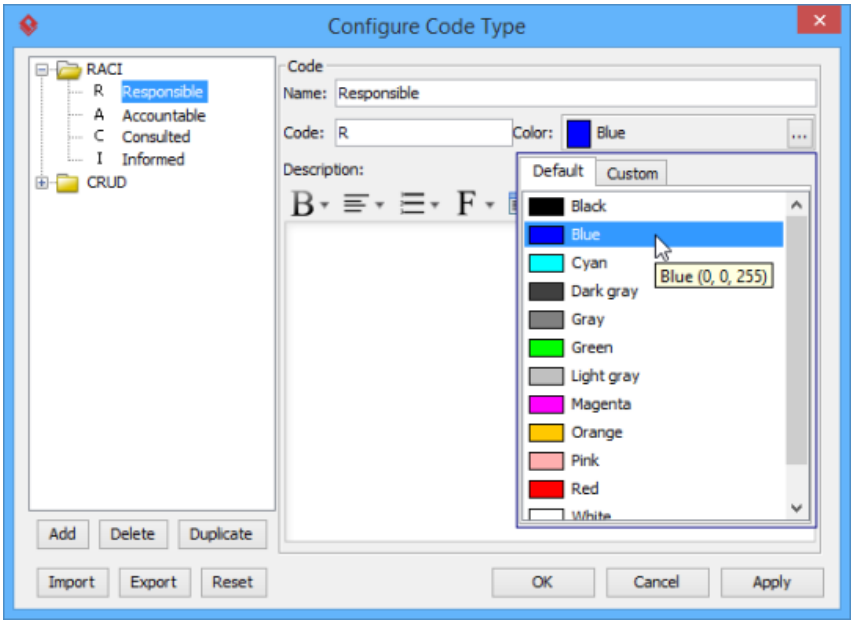
Defining color for roles

1. To define a color for roles, you can modify the existing code type by clicking the ... button of **Code Type** in **Chart Diagram**.



Define code type

2. To define a color for a role, click the target role you want to change its color on the left, click the **...** button next to **Color** and select a color from the pop-up menu.

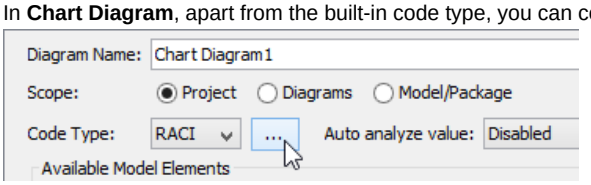


Select a color for target role

3. Click **OK** button.

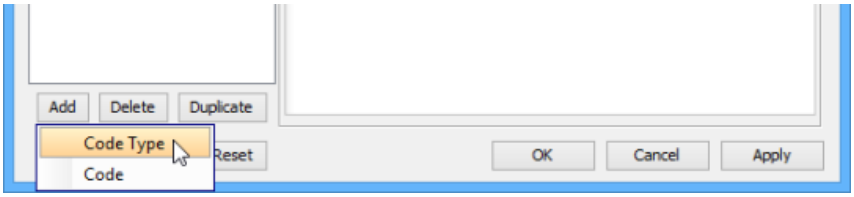
Defining a new code type

1. In **Chart Diagram**, apart from the built-in code type, you can configure a new code type by clicking the **...** button of **Code Type**.



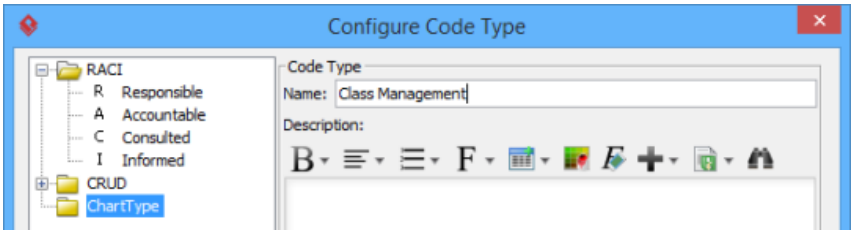
Define a new type of chart

2. In the pop-up **Configure Code Type** dialog box, select **Add > Code Type** from the pop-up menu at the bottom to add a code type.



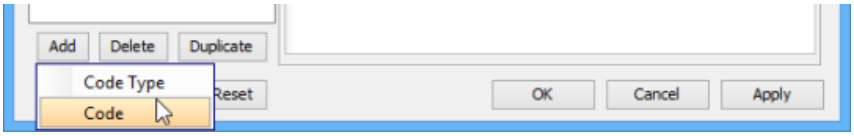
Add a code type

3. Rename the newly created code type in **Name**.



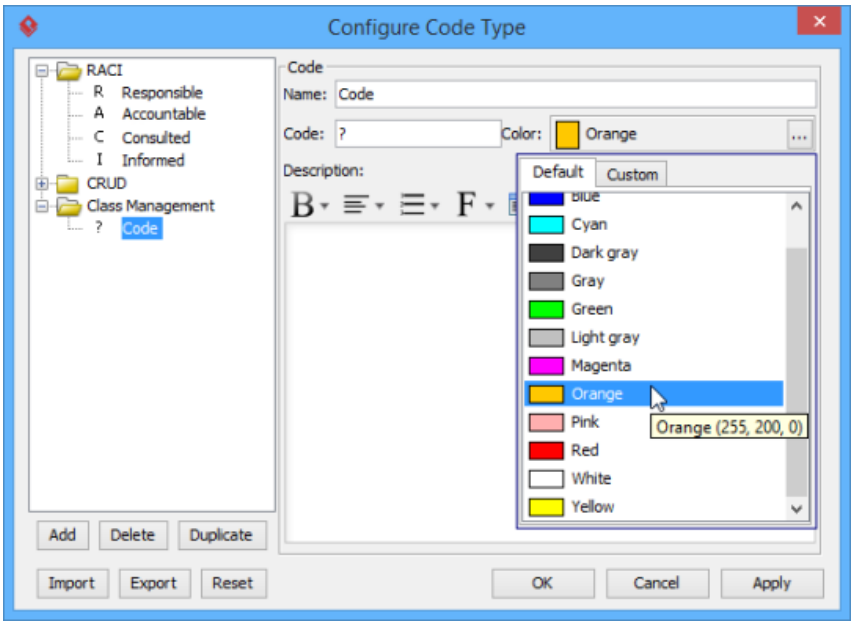
Rename the code type

4. To configure the roles under the new code type, select **Add > Code** from the pop-up menu.



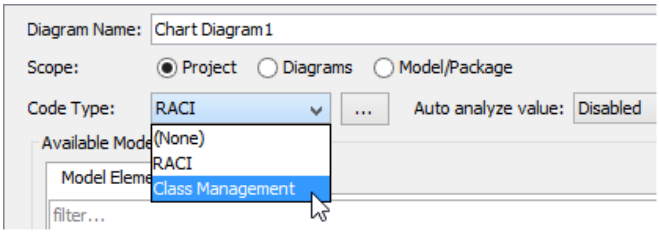
Add a new code

5. Enter the name of the code in **Name** and a letter which presents the code in **Code**. Click the **...** button next to **Color** and select a color for the code from the pop-up menu.



Enter the properties of code

6. Click **OK** button.
7. The chart type you created previously is currently available from the combo box of **Code Type**. Select it from the combo box of **Code Type**.



*Select a code type from the combo box of **Code Type***

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Service interface diagram

The interface-based approach of SoaML involves the use of simple interfaces and service interface. Simple interface focuses mainly on one-way service delivery that requires no protocol between parties.

Creating service interface diagram

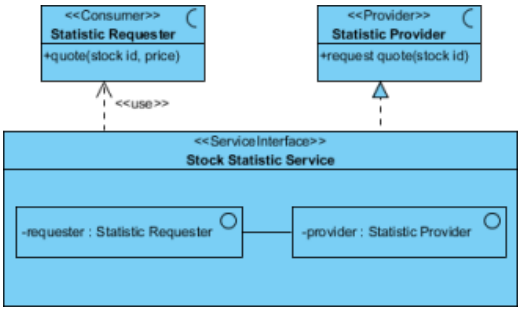
Learn how to create a service interface diagram. Have a look at the notations supported.

Creating service interface diagram

SoaML supports both a contract and an interface-based approach to SOA. They differ in the way services are specified.

The interface-based approach involves the use of simple interfaces and service interface. Simple interface focuses mainly on one-way service delivery that requires no protocol between parties. Service interface allows for bi-directional services. Provider and consumer work together to complete a service.

Service interface diagram is a type of SoaML diagram specialized for the definition and specification of both simple interface and service interface.



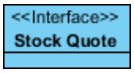
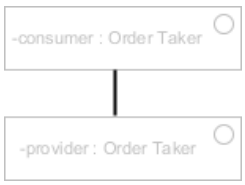
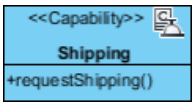
A sample service interface diagram

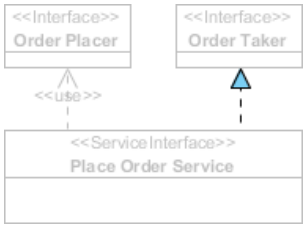
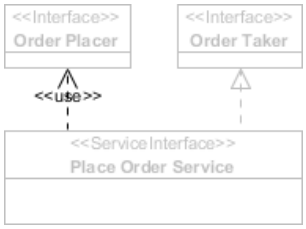
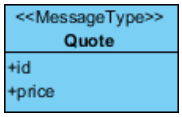
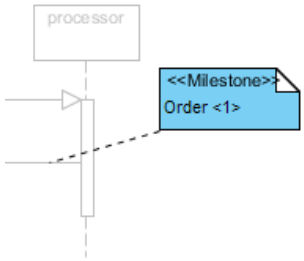
Creating service interface diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Service Interface Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	Description
Service Interface		A ServiceInterface defines the interface and responsibilities of a participant to provide or consume a service. It is used as the type of a Service or Request Port. A ServiceInterface is the means for specifying how a participant is to interact to provide or consume a Service. A ServiceInterface may include specific protocols, commands, and information exchange by which actions are initiated and the result of the real world effects are made available as specified through the functionality portion of a service. A ServiceInterface may address the concepts associated with ownership, ownership domains, actions communicated between legal peers, trust, business transactions, authority, delegation, etc.
Interface		Simple interfaces define one-way services that do not require a protocol. Such services may be defined with only a single UML interface and then provided on a "Service" port and consumed on a "Request" port.
Role		A ServiceInterface is a UML Class. It defines specific roles for each participant plays in the service interaction. These roles have a name and an interface type. The interface of the provider (which must be the type of one of the parts in the class) is realized (provided) by the ServiceInterface class. The interface of the consumer (if any) must be used by the class.
Connector		Connect roles in a service interface.
Capability		A Capability models the ability to act and produce an outcome that achieves a result that may provide a service specified by a ServiceContract or ServiceInterface irrespective of the Participant that might provide that service. A ServiceContract alone, has no dependencies or expectation of how the capability is realized – thereby separating the concerns of 'what" vs. "how." The Capability may specify dependencies or internal process to detail how that capability is provided including dependencies on other Capabilities. Capabilities are shown in context using a service dependencies diagram.

Expose		The Expose dependency provides the ability to indicate what Capabilities that are required by or are provided by a participant should be exposed through a Service Interface.
Dependency		The Provider may also have a uses dependency on the consumer interface, representing the fact that the provider may call the consumer as part of a bi-directional interaction. These are also known as "callbacks" in many technologies.
Realization		A ServiceInterface specifies the receptions and operations it receives through InterfaceRealizations. A ServiceInterface can realize any number of Interface. Some platform specific models may restrict the number of realized interfaces to at most one.
Usage		A ServiceInterface specifies its required needs through Usage dependences to Interfaces.
Message Type		A MessageType is a kind of value object that represents information exchanged between participant requests and services. This information consists of data passed into, and/or returned from, the invocation of an operation or event signal defined in a service interface. A MessageType is in the domain or service-specific content and does not include header or other implementation or protocol-specific information.
Milestone		A Milestone depicts progress by defining a signal that is sent to an abstract observer. The signal contains an integer value that intuitively represents the amount of progress that has been achieved when passing a point attached to this Milestone.

A list of supported notations in service interface diagram

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Service participant diagram

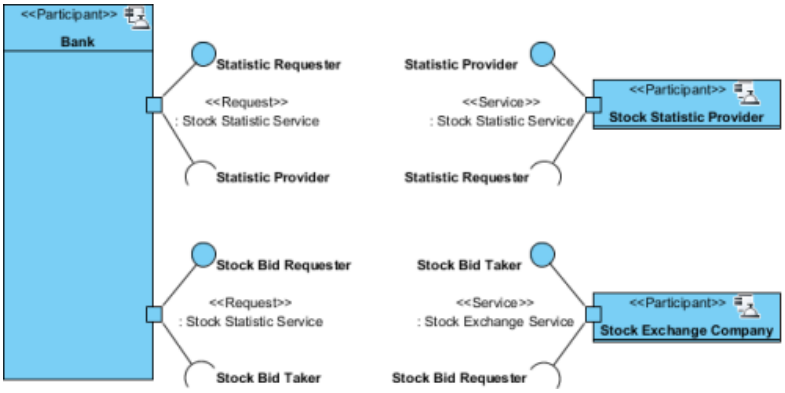
Modelers use SoaML service participant diagram to represent these participants as well as the interfaces they required or provided in accomplishing services.

Creating service participant diagram

Learn how to create a service participant diagram. Have a look at the notations supported.

Creating service participant diagram

SoaML service participant diagram focuses on the person, organization, system or anyone who take part in a services architecture. Modelers use service participant diagram to represent these participants as well as the interfaces they required or provided in accomplishing services. Note that the way how participants interact is not modeled in service participant diagram.



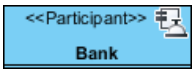
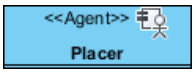
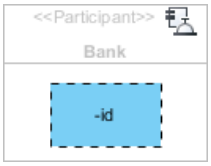
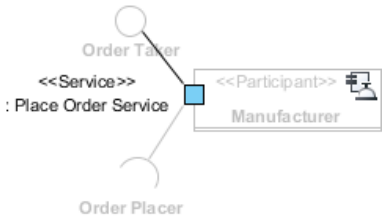
A sample service participant diagram

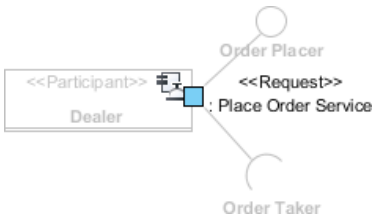
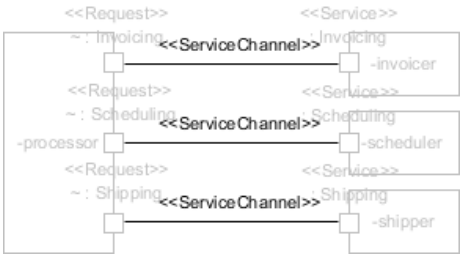
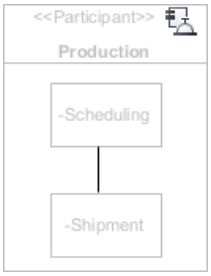
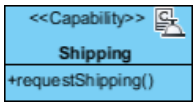
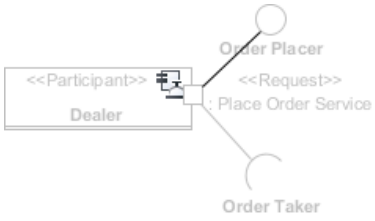
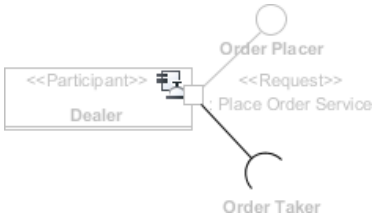
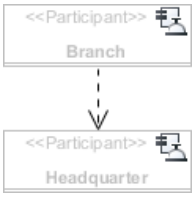
Creating service participant diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Service Participant Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	Description
Participant		A Participant represents some (possibly concrete) party or component that provides and/or consumes services (participants may represent people, organizations, or systems that provide and/or use services). A Participant is a service provider if it offers a service. A Participant is a service consumer if it uses a service. A participant may provide or consume any number of services. Service consumer and provider are roles Participants play: the role of providers in some services and consumers in others, depending on the capabilities they provide and the needs they have to carry out their capabilities. Since most consumers and providers have both services and requests, Participant is used to model both.
Agent		In general, agents can be software agents, hardware agents, firmware agents, robotic agents, human agents, and so on. While software developers naturally think of IT systems as being constructed of only software agents, a combination of agent mechanisms might in fact be used from shop-floor manufacturing to warfare systems.
Part		Used as a composite component for the participant.
Property		A property is a structural feature. It relates an instance of the class to a value or collection of values of the type of the feature. A property may be designated as an identifier property, a property that can be used to distinguish or identify instances of the containing classifier in distributed systems.
Service Port		A service port is the feature that represents the point of interaction on a Participant where a service is actually provided. On a service provider, this can be thought as the "offer" of the service (based on the service interface). In other words, the service port is the point of interaction for engaging participants in a service via its service interfaces.

Request Port		A consumer of a service specifies the serviceinterface they require by using a request port.
Port		Port is extended with a connectorRequired property to indicate whether a connector is required on this port or the containing classifier may be able to function without anything connected.
Service Channel		A ServiceChannel provides a communication path between consumer Requests and provider services.
Connector		Connect parts in a participant, if any.
Capability		A Capability models the ability to act and produce an outcome that achieves a result that may provide a service specified by a ServiceContract or ServiceInterface irrespective of the Participant that might provide that service. A ServiceContract alone, has no dependencies or expectation of how the capability is realized – thereby separating the concerns of "what" vs. "how." The Capability may specify dependencies or internal process to detail how that capability is provided including dependencies on other Capabilities. Capabilities are shown in context using a service dependencies diagram.
Realization		Connects service/request port and interface to represent the interface provided by the participant at specific port.
Usage		Connects service/request port and interface to represent the interface required by the participant, at specific port.
Dependency		Shows that a participant/interface/port depends on a participant/interface/port.

A list of supported notations in service participant diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Service contract diagram

Service contract diagram is designed for the specification of service contract. Modelers usually combines the use of sequence diagram and activity diagram in representing the choreography of a service contract.

Creating service contract diagram

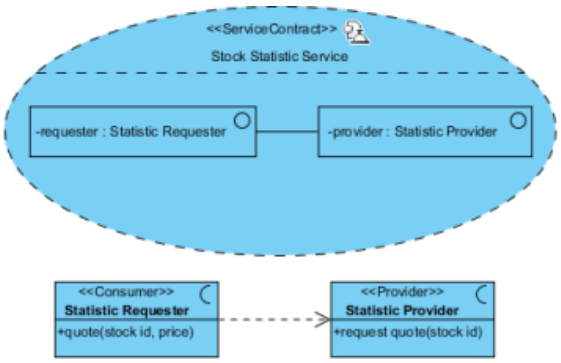
Learn how to create a service contract diagram. Have a look at the notations supported.

Creating service contract diagram

SoaML supports both a contract and an interface-based approach to SOA. They differ in the way services are specified.

The service contract approach defines the contract that specify how providers and consumers work together to achieve a goal, through the use of service. The service contract represents an agreement between parties for how the service is to be provided and consumed. Such agreement includes the interfaces, choreography and other terms and conditions.

Service contract diagram is designed for the specification of service contract. Modelers usually combines the use of sequence diagram and activity diagram in representing the choreography of a service contract.



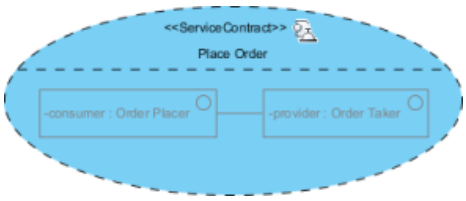
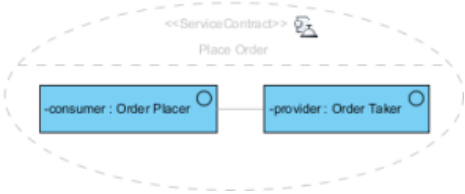
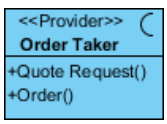
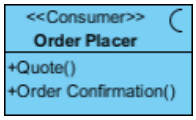
A sample service contract diagram

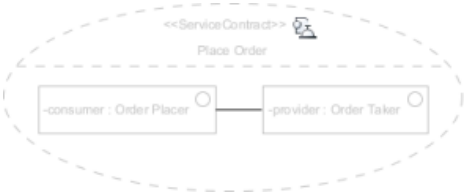
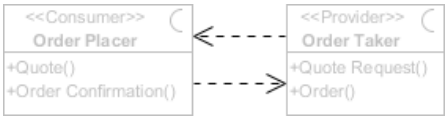
Creating service contract diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Service Contract Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	Description
Service Contract		A ServiceContract is the specification of the agreement between providers and consumers of a service as to know what information, products, assets, value, and obligations will flow between the providers and consumers of that service. It specifies the service without regard for realization, capabilities, or implementation. A ServiceContract does not require the specification of who, how, or why any party will fulfill their obligations under that ServiceContract, thus providing for the loose coupling of the SOA paradigm. In most cases, a ServiceContract will specify two roles (provider and consumer) but other service roles may be specified as well. The ServiceContract may also own a behavior that specifies the sequencing of the exchanges between the parties as well as the resulting state and delivery of the capability. The owned behavior is the choreography of the service and may use any of the standard UML behaviors such as an interaction, timing, state, or activity diagram.
Role		ServiceContract captures an agreement between the roles played by consumers and providers of the service, their capabilities and needs and the rules for how the consumers and providers must interact. The roles in a ServiceContract are typed by Interfaces that specify Operations and events which comprise the choreographed interactions of the services.
Provider		A "Provider" models the interface provided by the provider of a service. The provider of the service delivers the results of the service interaction. The provider will normally be the one that responds to the service interaction. Provider interfaces are used as the type of a "ServiceContract" and are bound by the terms and conditions of that service contract.
Consumer		A «Consumer» models the interface provided by the consumer of a service. The consumer of the service receives the results of the service interaction. The consumer will normally be the one that initiates the service interaction. Consumer interfaces are used as the type of a «ServiceContract» and are bound by the terms and conditions of that service contract.

Connector		Connect roles, if any, in a participant.
Dependency		The Provider may also have a uses dependency on the consumer interface, representing the fact that the provider may call the consumer as part of a bi-directional interaction. These are also known as "callbacks" in many technologies.

A list of supported notations in service participant diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Services architecture diagram

Services architecture diagram is a SoaML diagram that represents services architecture.

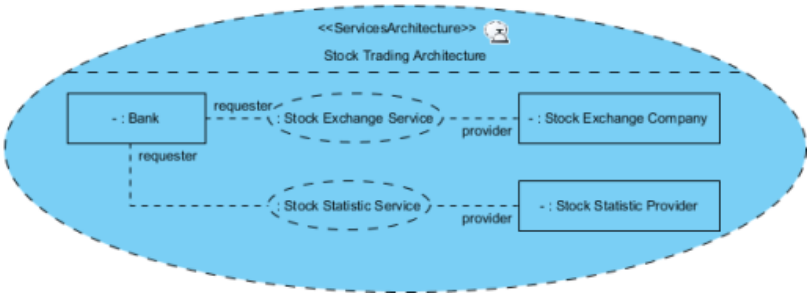
Creating services architecture diagram

Learn how to create a services architecture diagram. Have a look at the notations supported.

Creating services architecture diagram

Understanding how people, team and organizations work together for a goal enables them to work more cohesively using services without getting overly coupled. SoaML enables modelers to build a services architecture model for this purpose. The services architecture put together the service specification and participants, and shows how they work together to achieve a goal.

Services architecture diagram is a SoaML diagram that represents services architecture.



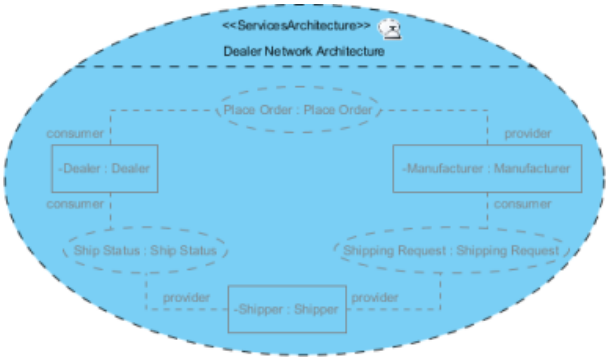
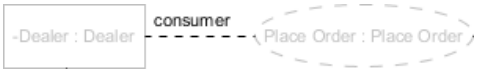
A sample services architecture diagram

Creating services architecture diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Services Architecture Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	
Services Architecture		A ServicesArchitecture (a SOA) describes how participants work together use of a service in a ServicesArchitecture is represented by the use of a S
Internal Participant		Participants who work together in an architecture.
External Participant		External participant uses the "shared aggregation" feature of UML to show
Service Contract Use (CollaborationUse)		A CollaborationUse explicitly indicates the ability of an owning Classifier to plays. If the CollaborationUse is strict, then the parts must be compatible w
Role Binding		A CollaborationUse contains roleBindings that binds each of the roles of it

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Service categorization diagram

The SoaML service categorization diagram allows categorizing SoaML elements with catalog, categories and category values.

Creating service categorization diagram

Learn how to create a service categorization diagram. Have a look at the notations supported.

Creating service categorization diagram

In order to allow model elements to be used for multiple purposes and to be viewed from different perspectives, we need a way to organize the content of model. Categorization is available for such purpose.

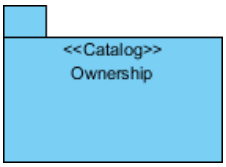
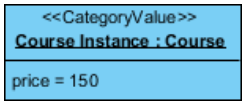
The SoaML service categorization diagram allows categorizing SoaML elements with catalog, categories and category values.

Creating service categorization diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Service Categorization Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	Description
Catalog		A named collection of related elements, including other catalogs characterized by a specific set of categories. Applying a Category to an Element using a Categorization places that Element in the Catalog.
Category		A Category is a piece of information about an element. A Category has a name indicating what the information is about and a set of attributes and constraints that characterize the Category. An Element may have many Categories and the same Category can be applied to many Elements. Categories may be organized into Catalogs hierarchies.
Category Value		A CategoryValue provides values for the attributes of a Category. It may also be used to categorize model elements providing detailed information for the category.

A list of supported notations in service categorization diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Animation

Animation helps you make possible paths in an active diagram by presenting the paths in animation form. This chapter will tell you how to animate your design and export the result to movie.

What is animation?

Describe animation in detail.

Animate business process diagram

Shows you how to animate a BPD and describe the various parts of Animation dialog box.

Animate sequence diagram

Shows you how to animate a sequence diagram and describe the various parts of Animation dialog box.

Animate activity diagram

Shows you how to animate an activity diagram and describe the various parts of Animation dialog box.

Exporting animation to Adobe Flash

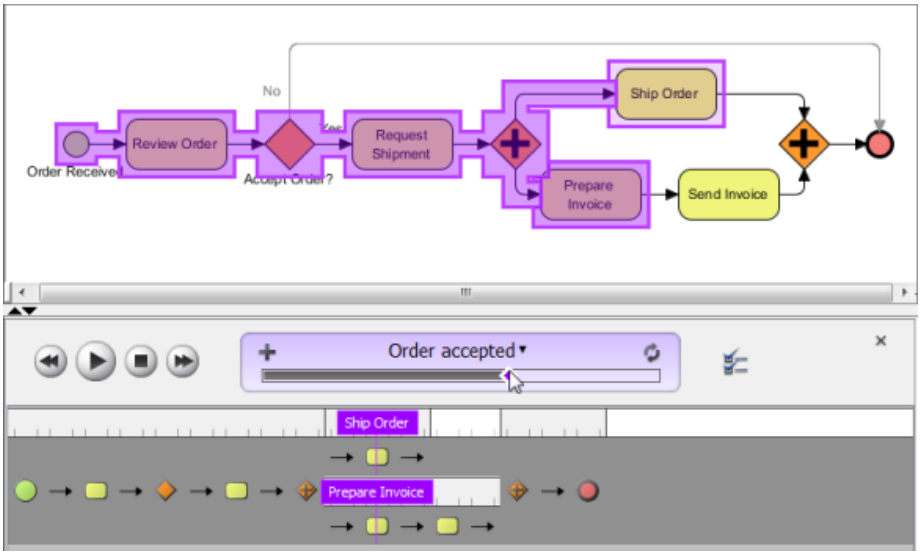
Shows you how to export the animation to Adobe Flash movie to play externally.

What is design animation?

Design animation helps you to make possible paths in an active diagram by presenting the paths in animation form. This can make your design more attractive by animating it. Besides, you can control the flow of animation yourself to help demonstrating your work to client with your annotation. It also calculates all possible paths of the interaction, making the design more accurate.

Animating Paths in Diagram

Animation can be played directly on diagram. When the animation begins, a tiny black dot will be attached to the begining of the path selected to animate. During the animation, the black dot will traverse through the path, shapes that lie on the path will be painted in purple one by one when being approached by the black dot until the black dot reached the end of the path.



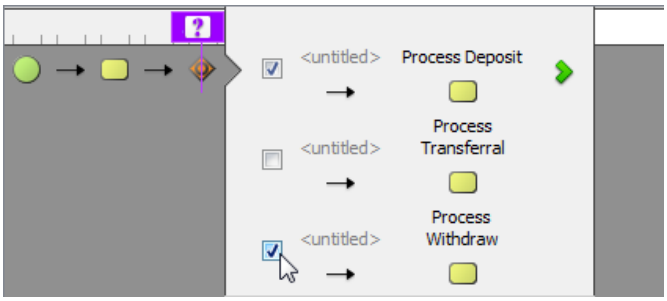
An animating path

Automatic Paths Identification

Interconnected shapes form a path. It is possible to have multiple paths on a diagram. The animation tool helps finding out all possible paths in a diagram. When opening the Animation dialog box, valid paths on the opening diagram will be identified and listed for selection. You can then select a path to animate. Unclosed paths or paths that does not obey the notation are classified as invalid, thus won't be available for playing animation.

Filtering Business Paths Base on Conditions

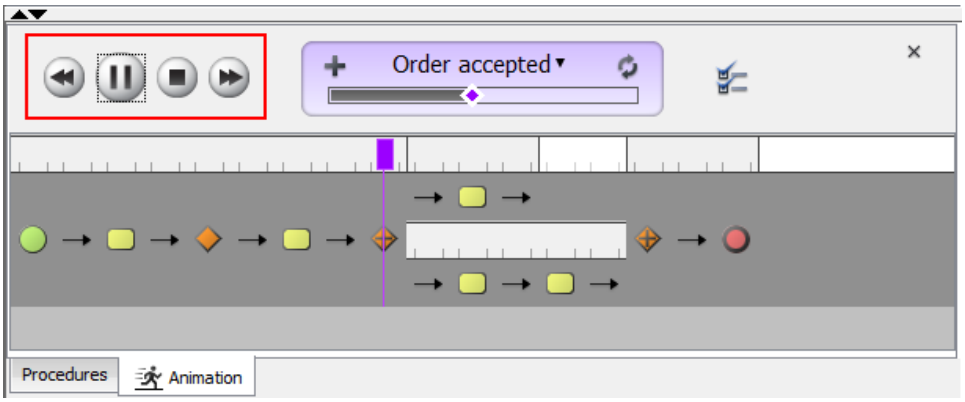
In BPD, BPMN gateway can be used to control how process flows. It can have multiple incoming and outgoing flows. There are several kinds of gateways that result in different flow behaviors. When you add an animation for process that involves gateway, you can select the outgoing flows for gateway(s).



Selecting outgoing flow for gateway

Walking through a Path Step-by-Step

Instead of letting the animation to run itself, you can control it by yourself. The horizontal bar that appears at the bottom of Visual Paradigm when animating lets you control the animation. Besides pausing, playing and stopping the animation, you can also move a shape backward or forward by pressing the  and  button. By making use of the forward and backward buttons, you can walk through a path shape by shape.



Walk though a path step by step

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

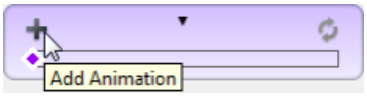
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Animating business process diagram

By animating a business process diagram, you can see the flow of tasks within a process, from the beginning until the end. This does not only help to understand a process but also trace the bottleneck and look for improvements.

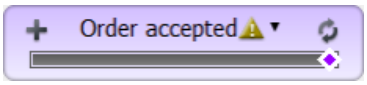
Adding an Animation

1. Select **Modeling > Animation...** from the toolbar.
2. Click **Add Animation** (graphically, a plus sign) at the top of the **Animation Panel**.



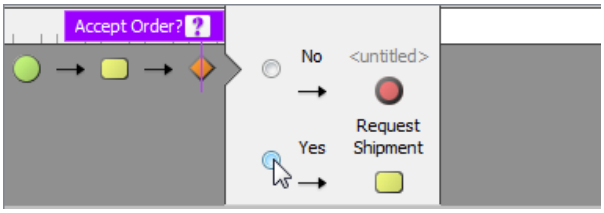
Add animation

3. Give a meaningful name to the animation based on the flow you want to animate. Press **Enter** to confirm editing.



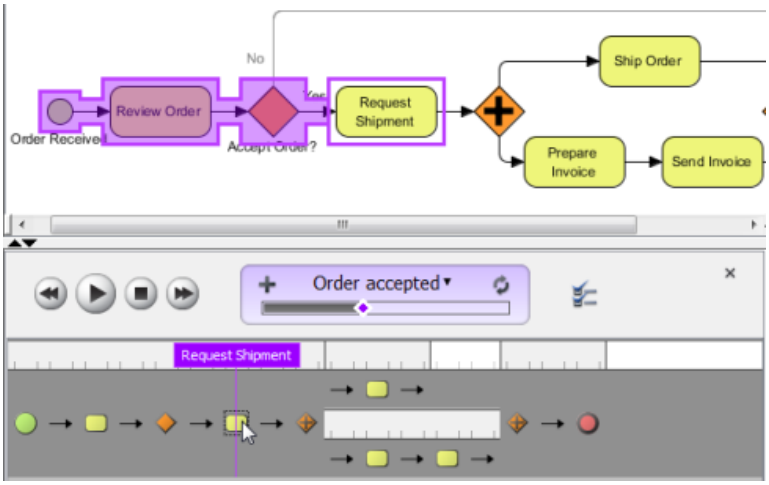
Named animation

4. When you add an animation for a process that involves gateway, you need to select the outgoing flows for gateway(s) in order to complete the path. To resolve an exclusive gateway requires the selection of the outgoing path. To resolve an inclusive gateway requires the selection of zero to multiple outgoint paths. Make your selection and click the green arrow button to confirm.



Select the outgoing path for an exclusive gateway

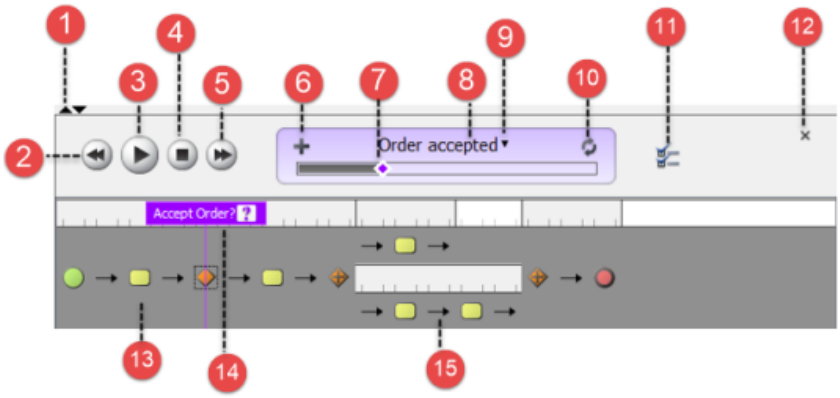
The path of the animation is determined automatically by evaluating the flow modeled in the diagram. Shapes that form the path are shown in the **Animation Panel** as icons. If you click on any of them, it will jump right to the corresponding model element in the diagram.



Select a shape in Animation Panel

Overview of Animation Panel

The **Business Process Diagram Animation** dialog box will pop out after clicking **Animation...**. This dialog box is where you can select an execution path to play an animation.

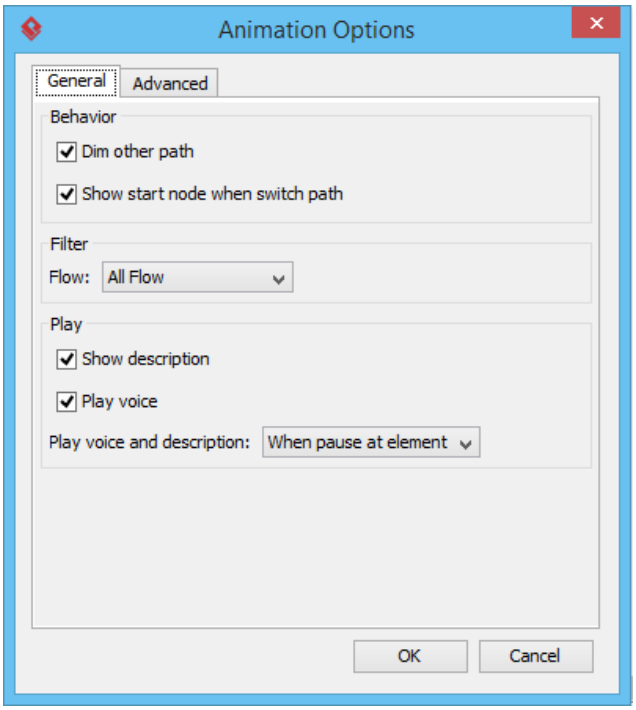


The Animation Panel

No.	Name	Description
1	Expand/Collapse	Expand or collapse Animation Panel
2	Backward	Move one shape backward in the flow.
3	Play	Play or continue to play the animation with Animation minimized.
4	Stop	Terminate the animation.
5	Forward	Advance to the next shape in the flow.
6	Add Animation	Create a new animation.
7	Slider	It is used for controlling the flow of animation.
8	Current Animation	The name of the animation.
9	Animation selection	Click on this button to select another path to animate.
10	Refresh	It is used for re-identifying the flow of animations base on the diagram content.
11	Options	Click to configure Animation.
12	Close	Click to close the **Animation Panel**.
13	Flow of shapes	It displays all shapes of the current animation. Pressing on a shape here will highlight the corresponding shape in diagram.
14	Shape name	The name of the selected or animating shape. A question mark indicates that the shape is a decision shape. You can click on the question mark to re-select its outgoing flow.
15	Parallel flow	Parallel flow is presented as a branch.

*Description of **Animation Panel***

Animation Options - General

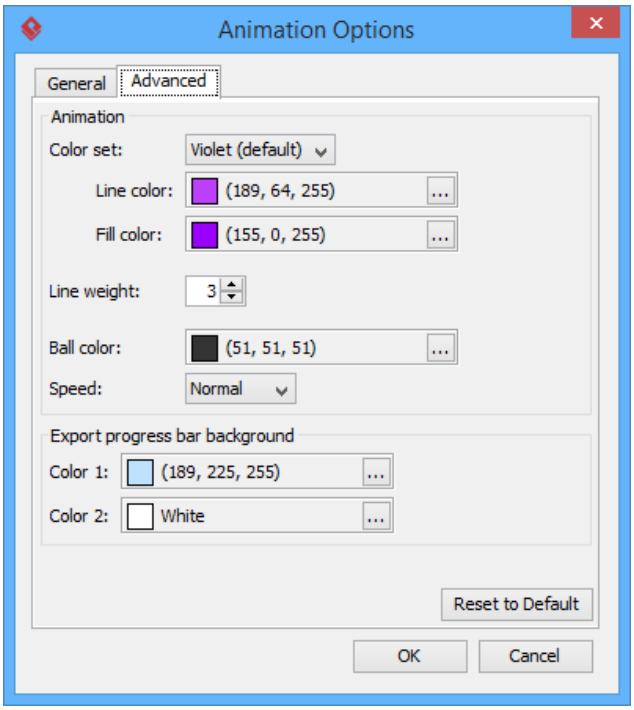


Animation Options - General

Name	Description
Dim other path	It dims the components that are not a part of the selected path.
Show start node when switch path	Jump to the first node of the selected path, or keep staying at the current viewing field.
Flow	All Flow: Re-evaluate added animations to accept all available paths. Sequence Flow: Re-evaluate added animations to accept only paths that are joined by sequence flows. Message Flow: Re-evaluate added animations to accept only paths that are joined by message flows.
Show description	It shows description of shape when playing the animation in exported HTML.
Play voice	Voice can be recorded as description of model element. Check this if you want to play recorded voice when playing the animation in exported HTML.

The description of Animation Options (General) window

Animation Options - Advanced



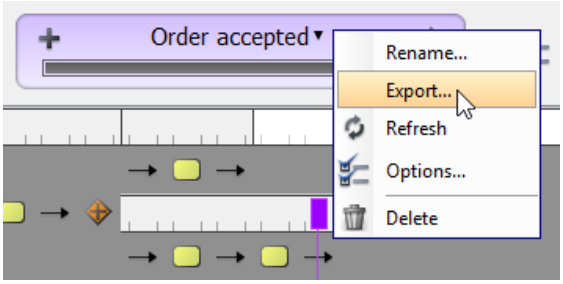
Animation Options - Advanced

Name	Description
Color set	Select a color set to controls the line and fill color of visited shape.
Line color	The line color of visited shapes.
Fill color	The fill color of visited shapes.
Line weight	The thickness of rectangle that surround the visited shapes'.
Ball color	The color of ball that goes through a path during animation for indicating the progress of flow.
Speed	The pace of animation.
Export progress bar background - Color 1	The background color for the top of progress bar in exported HTML.
Export progress bar background - Color 2	The background color for the bottom of progress bar in exported HTML.

The description of Animation Options (General) window

Exporting Animation

You can export the animation to Web contents so that you can play it externally in another computer just by playing in a Web browser. To export animation, right click beside the name of animation in **Animation Panel** and select **Export...** from the popup menu. Then, fill in the file path and click **OK** in the **Export window**. You can add paths to be exported by clicking the plus button.



Export animation

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

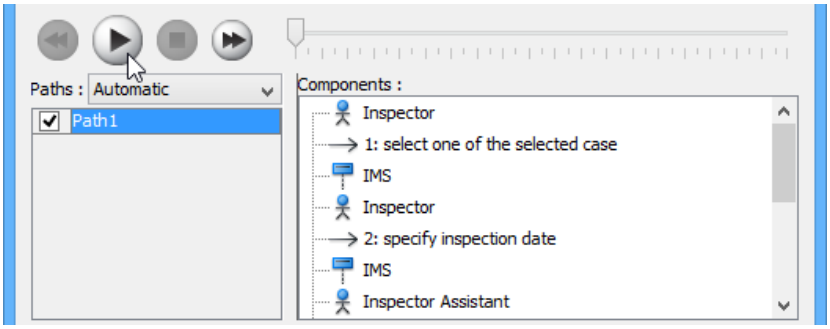
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Animating sequence diagram

By animating a sequence diagram, you can see interaction between lifelines and the flow of message calls in active.

Launching an Animation

1. Select **Modeling > Animation...** from the toolbar.
2. In **Sequence Diagram Animation** dialog box, select a path and then click **Play**.



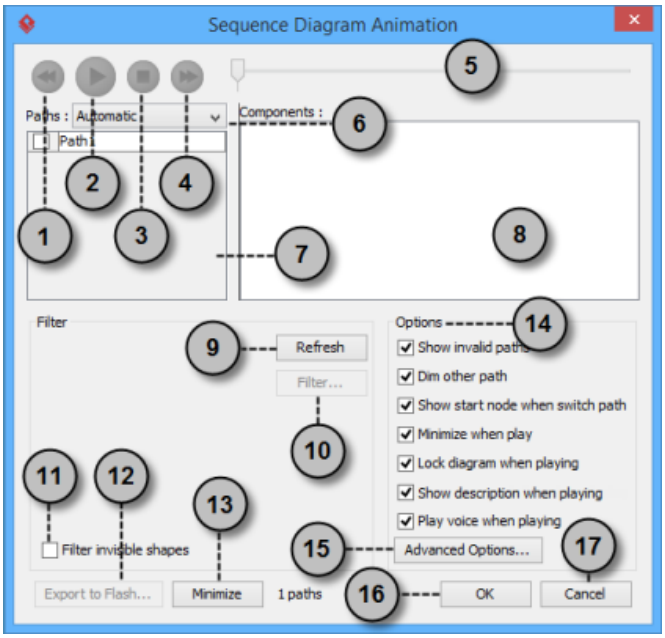
*Clicking **Play** in **Sequence Diagram Animation** dialog box*

NOTE: The animation tool can also be started by using any of the ways below:

- Right-click on the diagram background and select **Utilities > Animation...** from the popup menu.
- Click **Show Action Bar** on the right of the diagram pane, then select **Animation**.

Overview of Animation

The **Sequence Diagram Animation** dialog box will pop out after clicking **Animation...**. This dialog box is where you can select an execution path to play an animation.



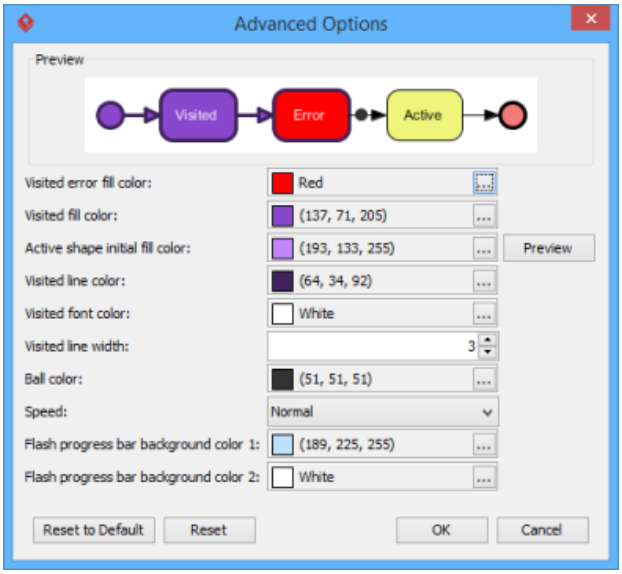
***Sequence Diagram Animation** dialog box*

No.	Name	Description
1	Backward	Move one shape backward in the flow.
2	Play	Play or continue to play the animation with Animation minimized.
3	Stop	Terminate the animation.
4	Forward	Advance to the next shape in the flow.
5	Slider	It is used for controlling the flow of animation.
6	Paths	It provides two ways of producing animation for the possible paths. **Automatic**: It is chosen by default. This helps you to detect all possible paths automatically. **Manual**: Choose when you want to select the possible path(s) manually.

7	Paths list	It lists all possible ways of executing a sequence. By default, paths are named as Path1, Path2, and so forth. You can rename them by double clicking on them and giving meaningful names.
8	Components list	It displays all components of the selected path. Pressing on a component will highlight the first shape of the chosen path until the chosen shape in the diagram.
9	Refresh	It is used for re-identifying the paths base on filter assignment and diagram content.
10	Filter...	It helps removing the non-selected paths by specifying the end result of fork nodes.
11	Filter invisible shapes	A shape can be set invisible on a diagram or become invisible due to belonging to an invisible layer. By checking this option, invisible shapes will be ignored when calculating paths. By unchecking, invisible path will be included when calculating paths. By unchecking, you will see a black ball flying on diagram without attaching to the invisible shape(s) when executing a path.
12	Export to Flash...	Select an output path for exporting this diagram's animation to Adobe Flash.
13	Minimize	Click to minimize this dialog box.
14	Options pane	The Options pane helps you to configure animation. **Show invalid paths**: It lists not only the valid and selected path but also the invalid and non-playable paths in the **Paths list**. **Dim other path**: It dims the components that are not a part of the selected path. **Show start node when switch path**: Jump to the first node of the selected path or keep staying at the current viewing field. **Minimize when play**: It minimizes this dialog box when playing an animation. **Lock diagram when playing**: It locks the diagram when playing the animation to prevent accidental editing. **Show description when playing**: It shows description of shape at the bottom right of diagram when playing the animation. **Play voice when playing**: Voice can be recorded as desription of model element. Check this if you want to play recorded voice when running animation.
15	Advanced Options...	It provides the color and speed options for animation.
16	OK	Click this button to confirm the settings and close Animation.
17	Cancel	Click this button to close Animation without saving the editing.

*Description of **Sequence Diagram Animation** dialog box*

Advanced Options



***Advanced Options** dialog box*

Name	Description
Visited error fill color	The background color of visited shape that cause an error. An error means the flow object that causes a path invalid.
Visited fill color	The background color of visited shapes.
Active shape initial fill color	When playing an animation, a tiny black ball will traverse the chosen path, from one shape to another. When it reaches a shape, the shape will render with a transition effect that means transiting from an initial color to visited fill color. This option manages the initial background color for visiting shape.

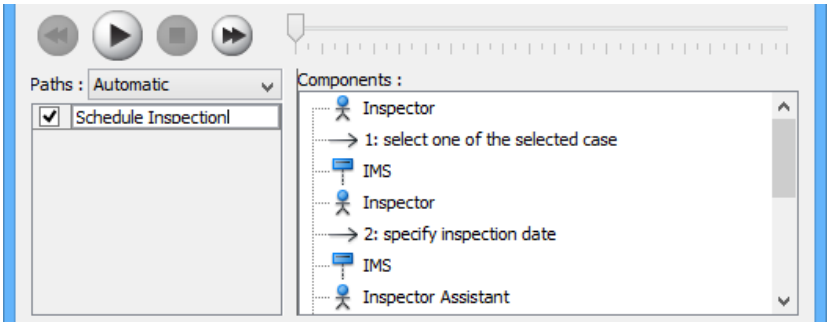
Visited line color	The line color of visited shapes.
Visited font color	The font color of visited shapes.
Visited line width	The thickness of visited shape's border.
Ball color	The color of ball that goes through a path during animation for indicating the progress of flow.
Speed	The pace of animation.
Flash progress bar background color 1	The background color for the top of progress bar in exported Flash movie.
Flash Progress Bar background color 2	The background color for the bottom of progress bar in exported Flash movie.

*The description of **Advanced Options** dialog box*

Naming a Path

The **Paths** list displays all possible animation paths of your diagram. Each path represents a possible way to go through the diagram. By default, paths are named as Path1, Path2, and so forth. It is recommended to name to the path(s) for better clarification.

1. To rename a path, move the mouse pointer on a path in the list and double click on it.
2. Enter the name of path.
3. Press **Enter** to confirm editing.

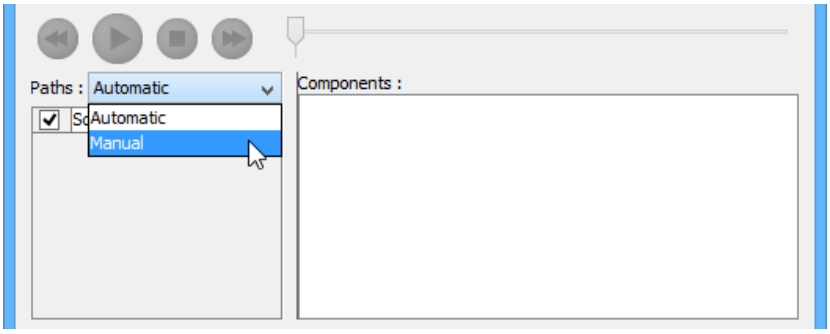


Naming the path

Creating a Manual Path

In **Sequence Diagram Animation** dialog box, all paths are listed in **Paths list** by default. However, you can manage the flow of animation with your own choice. To create a manual path:

1. Select **Manual** in **Paths**.

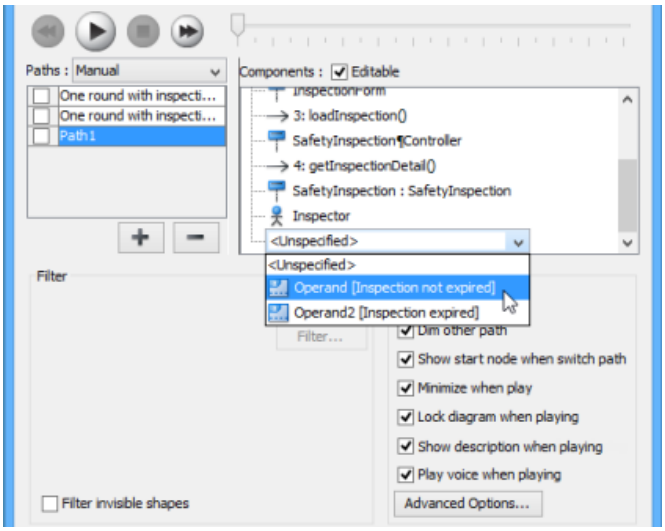


*Selecting **Manual** in **Paths***

2. Press **Add Path** to insert a new path.
3. Select the shapes that are shown on the **Components list** to direct the flow of animation.
4. Click **OK** to confirm editing.

Handling Decision

You should choose an operand when there is more than one option in the interaction. Different decisions will lead to different forks and make a different outcome for the flow of animation. Make either decision to view the outcome.



Making a decision for the flow of path

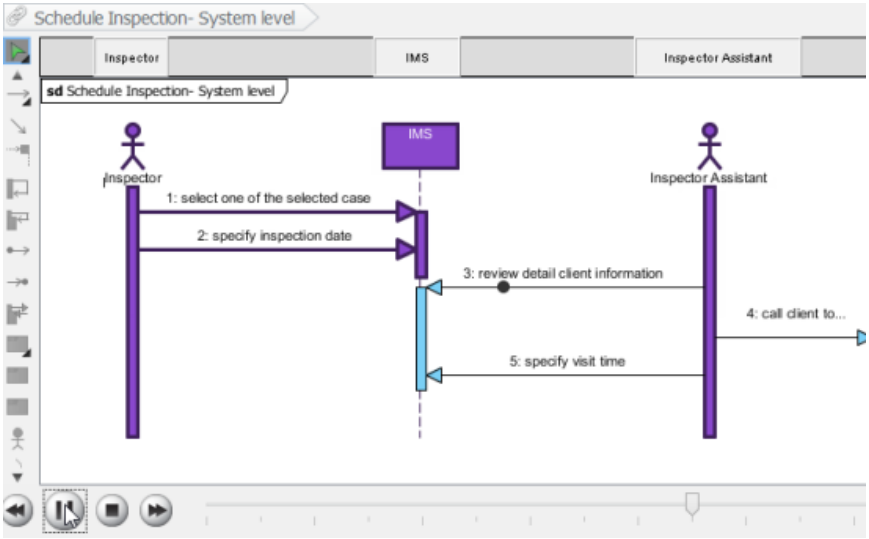
Reviewing an Animation

1. When everything is ready, click **Play** to start the animation of the selected path.
2. After click **Play**, **Sequence Diagram Animation** dialog box will be minimized to the bottom of your diagram, with several buttons and a slider revealing on it.

Button	Name	Description
	Backward	Move one shape backward in the flow.
	Pause	Temporary stop playing the movie. Press **Play** to continue to play.
	Play	Play or continue to play the animation.
	Forward	Advance to the next shape in the flow.
	Stop	Terminate the animation.
	Maximize	Maximize **Animation**.

Description of Animation bar

3. When the animation starts, a black ball will appear at beginning of path and traverse through the path until the end.
4. When the black ball reaches a shape, the shape will turn into purple.

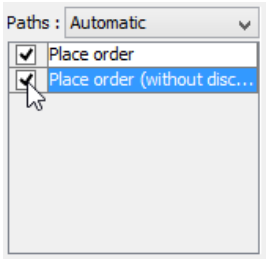


Reviewing the animation

Exporting an Animation

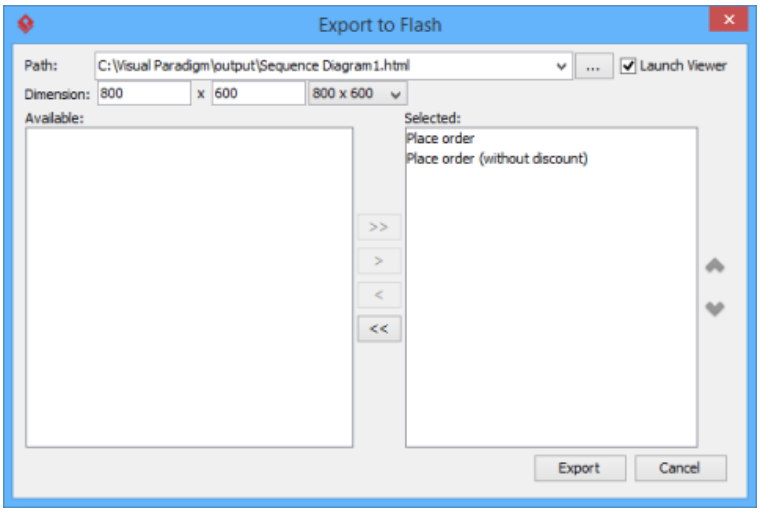
You can export the animation to Web contents so that you can play it externally in another computer just by playing in a Web browser.

1. From the **Paths** list in the **Animation** window, select the execution paths to export as Flash movie.



Path selection

2. Click the **Export to Flash...** button at bottom left. This shows the **Export to Flash** dialog box. Here is a description of the **Export to Flash** dialog box.



The Export to Flash window

Here is a description of the **Export to Flash** dialog box.

Part	Description
Path	The path of the exported HTML file. Flash movie file (.swf) will also be exported to the same folder as the HTML file.
Launch Viewer	When checked, default web browser will automatically start and play the exported Flash movie.
Dimension	The width and height of viewing region of Flash.
Available	Available paths that can be selected to export to Flash movie for animation.
Selected	Selected paths to export to Flash movie for animation.

Description of the Export to Flash dialog box

3. An HTML web page will be exported. Specify the path of the HTML file. Note that the Flash movie files (.swf) will be exported to the same folder as the HTML file.
4. Choose or enter the dimension of movie if necessary. Note that the dimension determins the size of viewable region instead of the size of diagram.
5. Click **Export**. Open the HTML file in the web browser to play the movie. If there are more then one path being selected, you can click on the drop down menu at top right corner and select another path to play with.

Related Resources

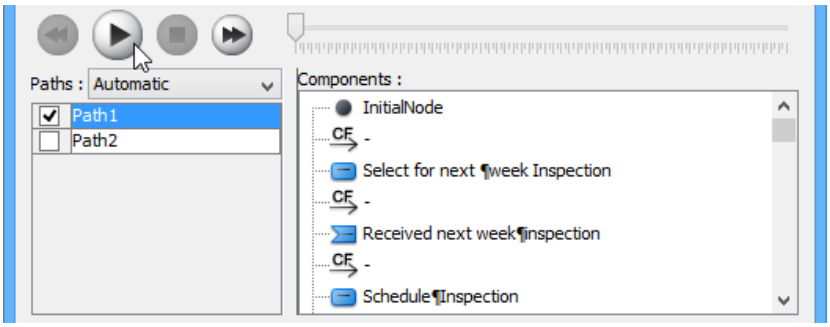
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Animating activity diagram

By animating an activity diagram, you can see the flow of actions in active.

1. Select **Modeling > Animation...** from the toolbar.
2. In **Activity Diagram Animation** dialog box, select a path and then click **Play**.



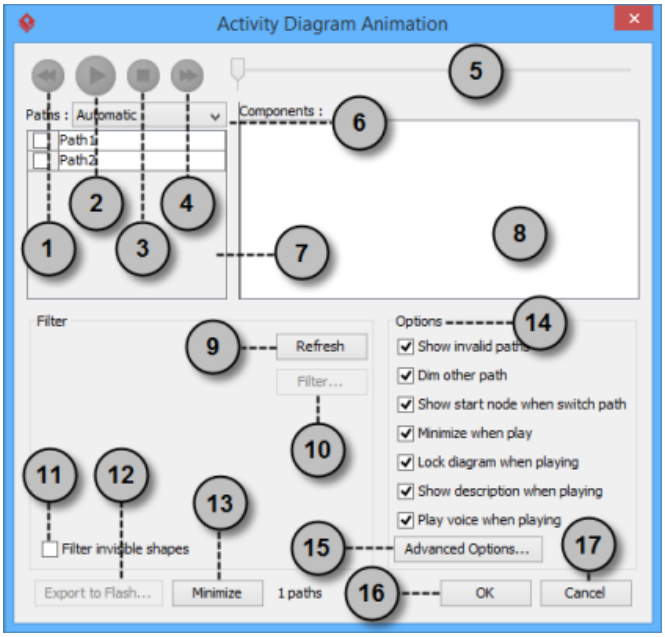
*Clicking **Play** in **Activity Diagram Animation** dialog box*

NOTE: Animation can also be started by using any of the ways below:

- Right-click on the diagram background and select **Utilities > Animation...** from the popup menu.
- Click the drop-down menu of **Modeling Tools** and select **Animation...** on the toolbar.

Overview of Animation

The **Activity Diagram Animation** dialog box will pop out after clicking **Animation...**. This dialog box is where you can select an execution path to play an animation.



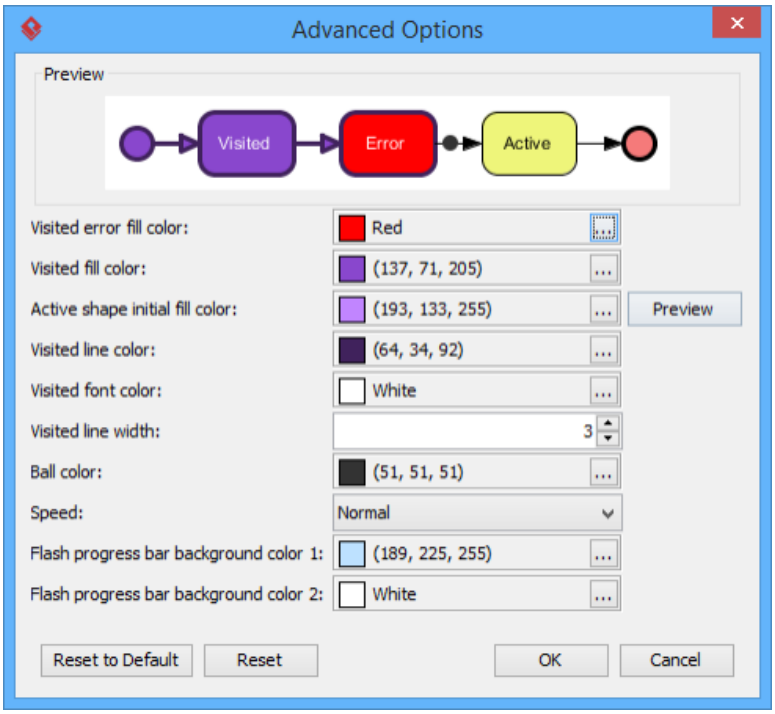
***Activity Diagram Animation** dialog box*

No.	Name	Description
1	Backward	Move one shape backward in the flow.
2	Play	Play or continue to play the animation with Animation minimized.
3	Stop	Terminate the animation.
4	Forward	Advance to the next shape in the flow.
5	Slider	It is used for controlling the flow of animation.
6	Paths	It provides two ways of producing animation for the possible paths. **Automatic**: It is chosen by default. This helps you to detect all possible paths automatically. **Manual**: Choose when you want to select the possible path(s) manually.
7	Paths list	It lists all possible ways of executing an Activity. By default, paths are named as Path1, Path2, and so forth. You can rename them by double clicking on them and giving meaningful names.

8	Components list	It displays all components of the selected path. Pressing on a component will highlight the first shape of the chosen path until the chosen shape in the diagram.
9	Refresh	It is used for re-identifying the paths base on filter assignment and diagram content.
10	Filter...	It helps removing the non-selected paths by specifying the end result of fork nodes.
11	Filter invisible shapes	A shape can be set invisible on a diagram, or become invisible due to belonging to an invisible layer. By checking this option, invisible shapes will be ignored when calculating paths. By unchecking, invisible path will be included when calculating paths. By unchecking, you will see a black ball fly on diagram without attaching to the invisible shape(s) when executing a path.
12	Export to Flash...	Select an output path for exporting this diagram's animation to Adobe Flash.
13	Minimize	Click to minimize this dialog box.
14	Options pane	The Options pane helps you to configure animation. **Show invalid paths**: It lists not only the valid and selected path, but also the invalid and non-playable paths in the **Paths list**. **Dim other path**: It dims the components that are not a part of the selected path. **Show start node when switch path**: Jump to the first node of the selected path or keep staying at the current viewing field. **Minimize when play**: It minimizes this dialog box when playing an animation. **Lock diagram when playing**: It locks the diagram when playing the animation to prevent accidental editing. **Show description when playing**: It shows description of shape at the bottom right of diagram when playing the animation. **Play voice when playing**: Voice can be recorded as descriprion of model element. Check this if you want to play recorded voice when running animation.
15	Advanced Options...	It provides the color and speed options for animation.
16	OK	Click this button to confirm the settings and close Animation.
17	Cancel	Click this button to close Animation without saving the editing.

*Description of **Activity Diagram Animation** dialog box*

Advanced Options



***Advanced Options** dialog box*

Name	Description
Visited error fill color	The background color of visited shape that cause an error. An error means the flow object that causes a path invalid.
Visited fill color	The background color of visited shapes.
Active shape initial fill color	When playing an animation, a tiny black ball will traverse the chosen path, from one shape to another. When it reaches a shape, the shape will render with a transition effect that means

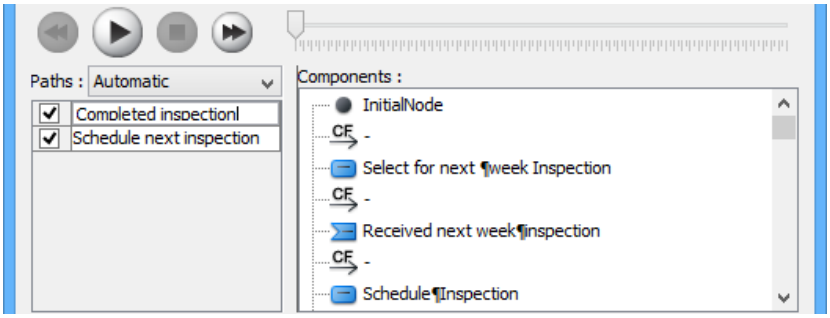
	transiting from an initial color to visited fill color. This option manages the initial background color for visiting shape.
Visited line color	The line color of visited shapes.
Visited font color	The font color of visited shapes.
Visited line width	The thickness of visited shape's border.
Ball color	The color of ball that goes through a path during animation for indicating the progress of flow.
Speed	The pace of animation.
Flash progress bar background color 1	The background color for the top of progress bar in exported Flash movie.
Flash progress bar background color 2	The background color for the bottom of progress bar in exported Flash movie.

*The description of **Advanced Options** dialog box*

Naming a Path

The **Paths** list displays all possible animation paths of your diagram. Each path represents a possible way to go through the diagram. By default, paths are named as Path1, Path2, and so forth. It is recommended to name to the path(s) for better clarification.

1. To rename a path, move the mouse pointer on a path in the list and double click on it.
2. Enter the name of path.
3. Press **Enter** to confirm editing.

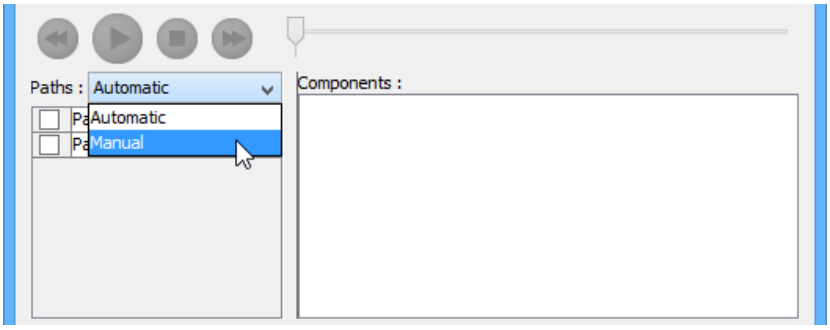


Naming the paths

Creating a Manual Path

In **Activity Diagram Animation** dialog box, all paths are listed in **Paths list** by default. However, you can manage the flow of animation with your own choice. To create a manual path:

1. Select **Manual** in **Paths**.

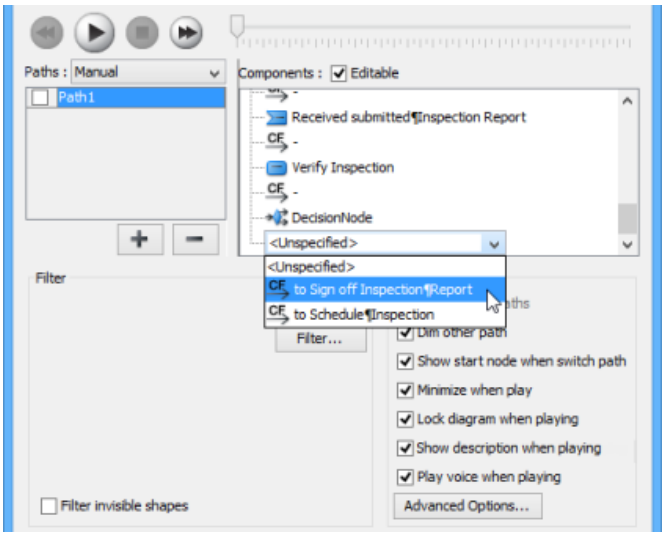


*Selecting **Manual** in **Paths***

2. Press **Add Path** to insert a new path.
3. Select the shapes that are shown on the **Components list** to direct the flow of animation.
4. Click **OK** to confirm editing.

Handling Decision

You should choose an outgoing flow when there is more than one option in the flow. Different decisions will lead to different forks and make a different outcome for the flow of animation. Make either decision to view the outcome.



Making a decision for the flow of path

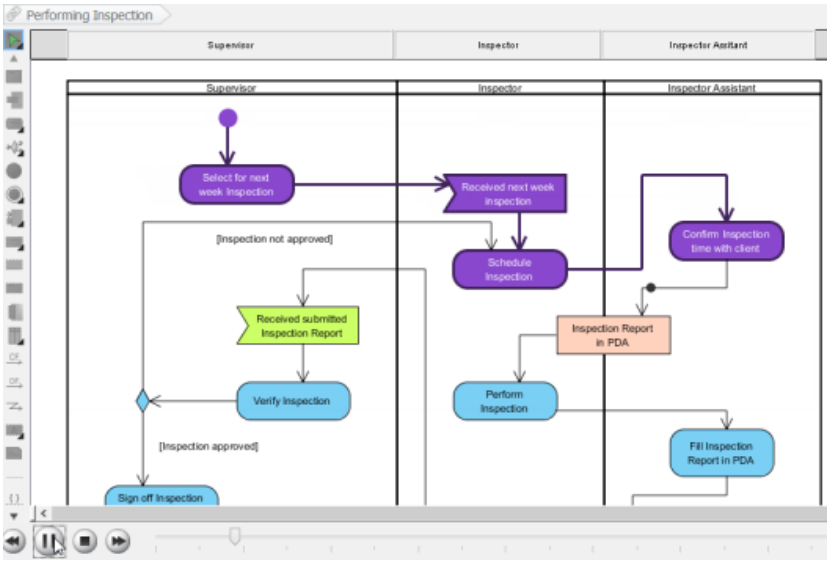
Reviewing an Animation

1. When everything is ready, click **Play** to start the animation of the selected path.
2. After click **Play**, **Activity Diagram Animation** dialog box will be minimized to the bottom of your diagram with several buttons and a slider revealing on it.

Button	Name	Description
	Backward	Move one shape backward in the flow.
	Pause	Temporary stop playing the movie. Press **Play** to continue to play.
	Play	Play or continue to play the animation.
	Forward	Advance to the next shape in the flow.
	Stop	Terminate the animation.
	Maximize	Maximize **Animation**.

Description of Animation bar

3. When the animation starts, a black ball will appear at beginning of path and traverse through the path until the end.
4. When the black ball reaches a shape, the shape will turn into purple.

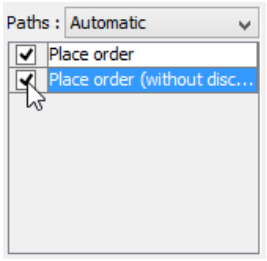


Reviewing the animation

Exporting an Animation

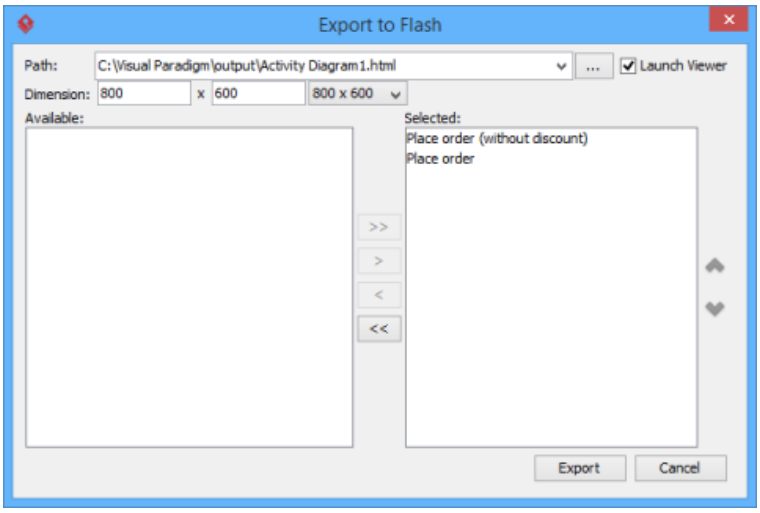
You can export the animation to Web contents so that you can play it externally in another computer just by playing in a Web browser.

1. From the **Paths** list in the **Animation** window, select the execution paths to export as Flash movie.



Path selection

2. Click the **Export to Flash...** button at bottom left. This shows the **Export to Flash** dialog box. Here is a description of the **Export to Flash** dialog box.



The Export to Flash window

Here is a description of the **Export to Flash** dialog box.

Part	Description
Path	The path of the exported HTML file. Flash movie file (.swf) will also be exported to the same folder as the HTML file.
Launch Viewer	When checked, default web browser will automatically start and play the exported Flash movie.
Dimension	The width and height of viewing region of Flash.
Available	Available paths that can be selected to export to Flash movie for animation.
Selected	Selected paths to export to Flash movie for animation.

Description of the Export to Flash dialog box

3. An HTML web page will be exported. Specify the path of the HTML file. Note that the Flash movie files (.swf) will be exported to the same folder as the HTML file.
4. Choose or enter the dimension of movie if necessary. Note that the dimension determins the size of viewable region instead of the size of diagram.
5. Click **Export**. Open the HTML file in the web browser to play the movie. If there are more then one path being selected, you can click on the drop down menu at top right corner and select another path to play with.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Zachman Framework

Zachman Framework provides structured and disciplined way of defining an enterprise.

Creating Zachman Framework

How to create Zachman Framework

Editing cell in Zachman Framework

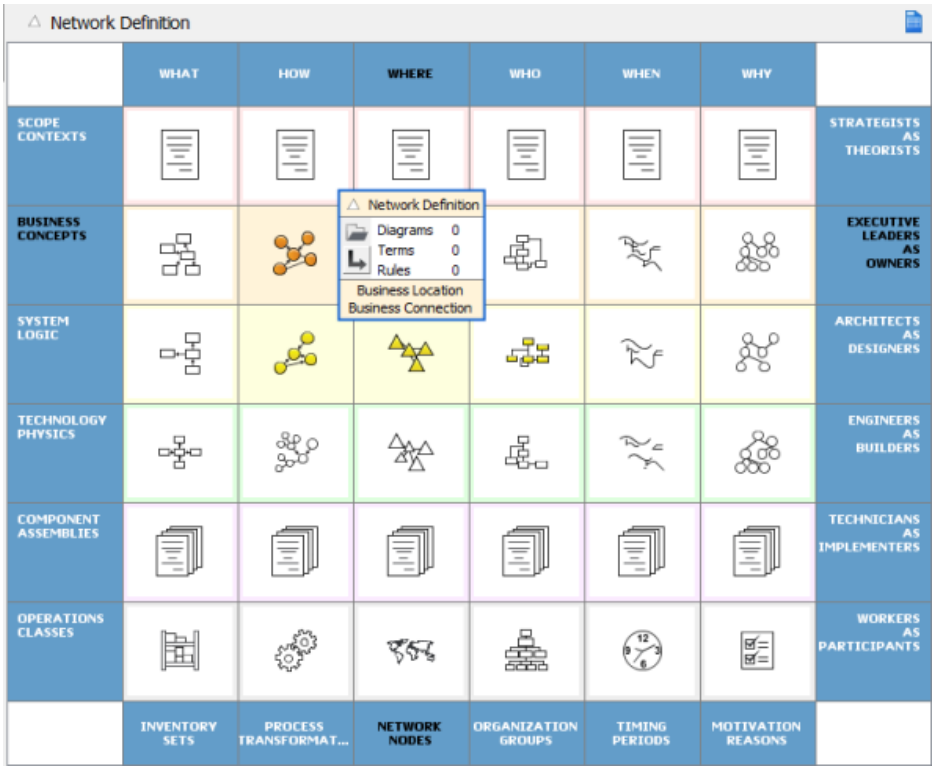
Diagrams, terms and business rules can be added to cells in Zachman Framework to describe enterprise from different perspectives.

Collapsing/Expanding rows or columns

Collapse the non-related rows and columns to make the remaining cell be focused.

Creating Zachman Framework

Zachman Framework provides structured and disciplined way of defining an enterprise. It has a matrix representation, with six rows (scope contexts, business concepts, system logic, technology physics, component assemblies, operations classes) and six columns (what, how, where, who, when, why). By adding proper diagrams, terms or business rules into cells, enterprise can be defined.



Zachman Framework

Creating Zachman Framework

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Zachman Framework**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

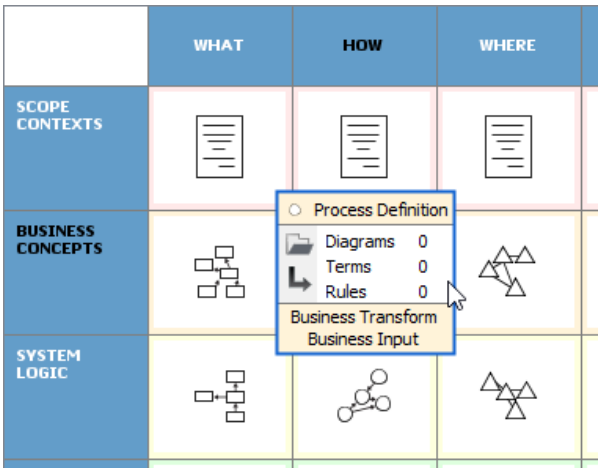
- Tutorial - View and define enterprise with Zachman Framework
- Full set of enterprise architecture tutorials
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Editing cell in Zachman Framework

Diagrams, terms and business rules can be added to cells in Zachman Framework to describe enterprise from different perspectives. To edit a cell, click on it. Then, click on appropriate link to add things to cell.

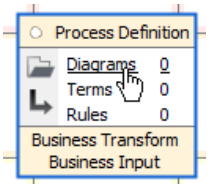
Add/Edit diagrams

1. Click on the cell you want to edit.



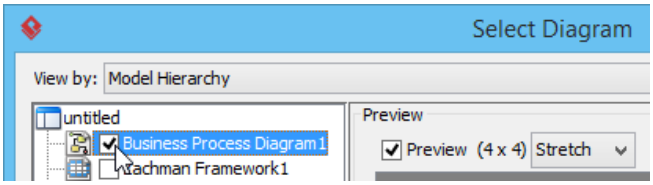
To edit the Process Definition cell

2. Click on the **Diagrams** link.



Click on the Diagrams link

3. In the window popped up, click **Add...** under the **Diagrams** tab.
4. In the **Select Diagram** window, select the diagram(s) to add to cell and click **OK**.



Select diagram(s) to add

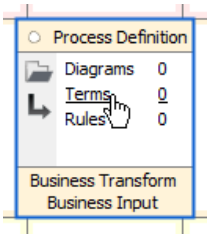
> **NOTE:** You can check **Add as sub diagram** to make the selected diagrams be added to the sub-diagrams of the cell element. When unchecked, the selected diagram will have their parent elements unchanged.

5. Click **OK** to return to diagram. You can see that the symbol in the edited cell is highlighted.

Add/Edit terms

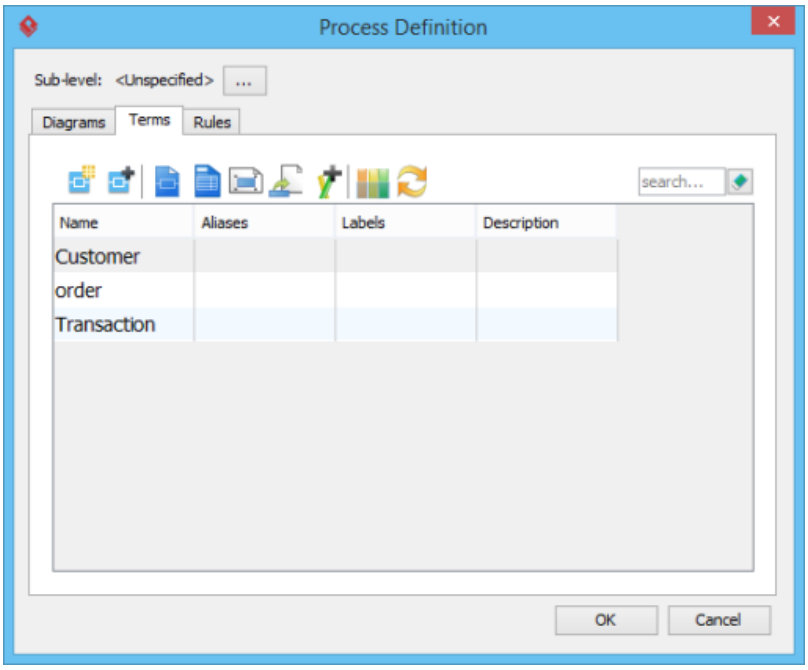
Here 'terms' refers to glossary terms. You can add terms to a cell.

1. Click on the cell you want to edit.
2. Click on the **Terms** link in the cell.



Click on the Terms link

3. This pops up a window with **Terms** tab selected. If you want to define a term here, click **New Term** in toolbar, which is the first button. Then, enter the name of the term. If you want to add reference to an existing term, click on the **Add Existing Terms** button, which is the second button. Then, select the terms to be added in the popup window and click **OK** to confirm.



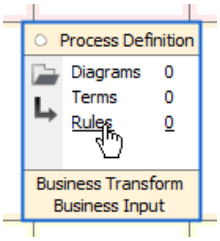
Terms added to cell

4. Click **OK** to return to diagram. You can see that the symbol in the edited cell is highlighted.

Add/Edit rules

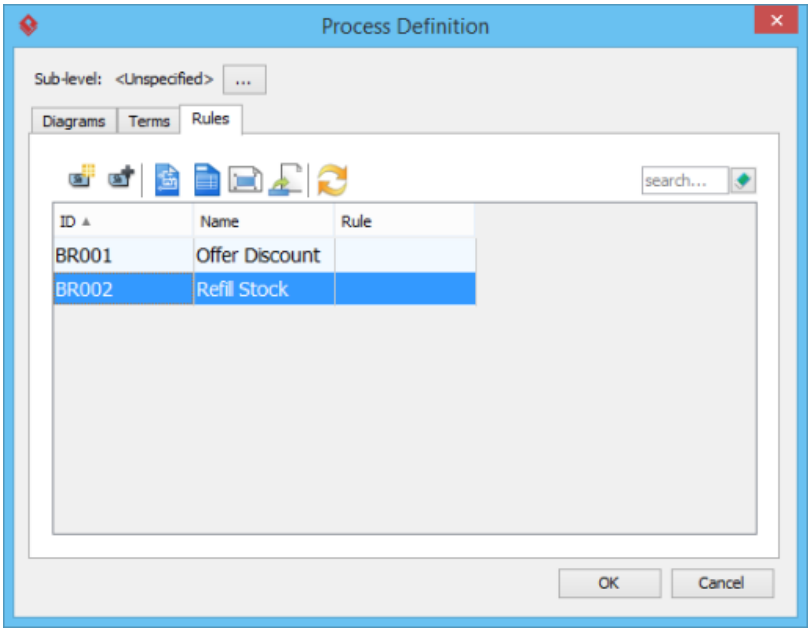
Business rules can also be added to cell.

1. Click on the cell you want to edit.
2. Click on the **Rules** link in the cell.



Click on the Rules link

3. This pops up a window with **Rules** tab selected. If you want to define a rule here, click **New Business Rule** in toolbar, which is the first button. Then, enter the name of therule . If you want to add reference to an existing rule, click on the **Add Existing Rules** button, which is the second button. Then, select the rules to add an in the popup window and click **OK** to confirm.

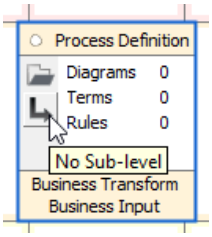


Rules added to cell

4. Click **OK** to return to diagram. You can see that the symbol in the edited cell is highlighted.

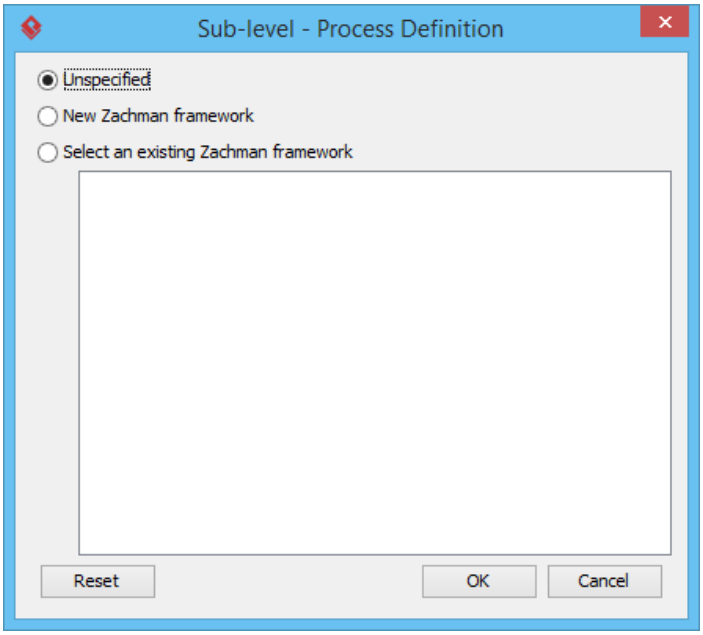
Forming sub-level

1. Click on the cell you want to edit.
2. Click on the button at the left hand side of the cell for adding sub-level.



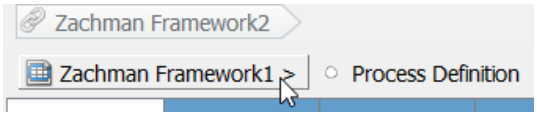
Form sub-level of cell

3. In the **Sub-level** window, select either of the following and click **OK**.
   - **Unspecified** - Nothing will happen (same as clicking **Cancel** directly)
   - **New Zachman framework** - Create a new Zachman Framework and add it as the sub-level of the editing cell.
   - **Select an existing Zachman framework** - Select a Zachman Framework created before to be the sub-level of the editing cell.



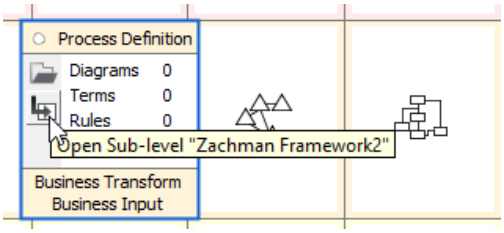
Select the Zachman Framework to add as sub-level

4. Click **OK**. Now, you are on the sub-level. To go back to the previous level, you may click on the link at the top of Zachman Framework. Cells with sub-level added will have their background painted.



To go back to the parent level

You can open the sub-level by clicking on the same button as clicked in step 2.



Open sub-level

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

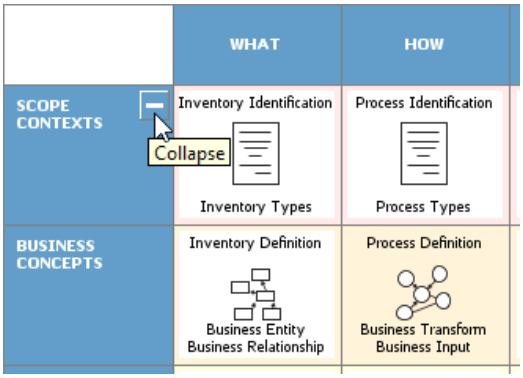
- Zachman Framework in multiple levels
- Full set of enterprise architecture tutorials

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Collapsing/Expanding rows or columns

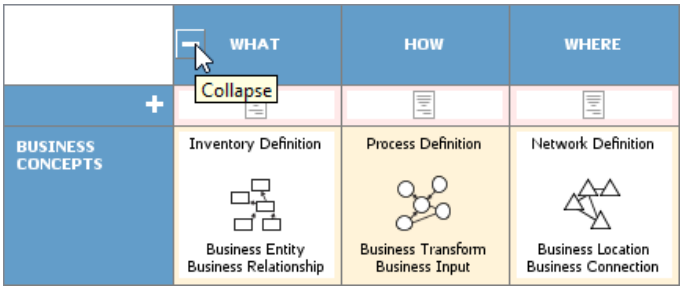
By default, you can see the names, symbols and descriptions in all cells in Zachman Framework. If you want to focus on specific cell, you can collapse the non-related rows and columns to make the remaining cell be focused. Collapsed cells show only the tiny symbol without showing any name and description.

1. To collapse a row, click on its **Collapse** button.



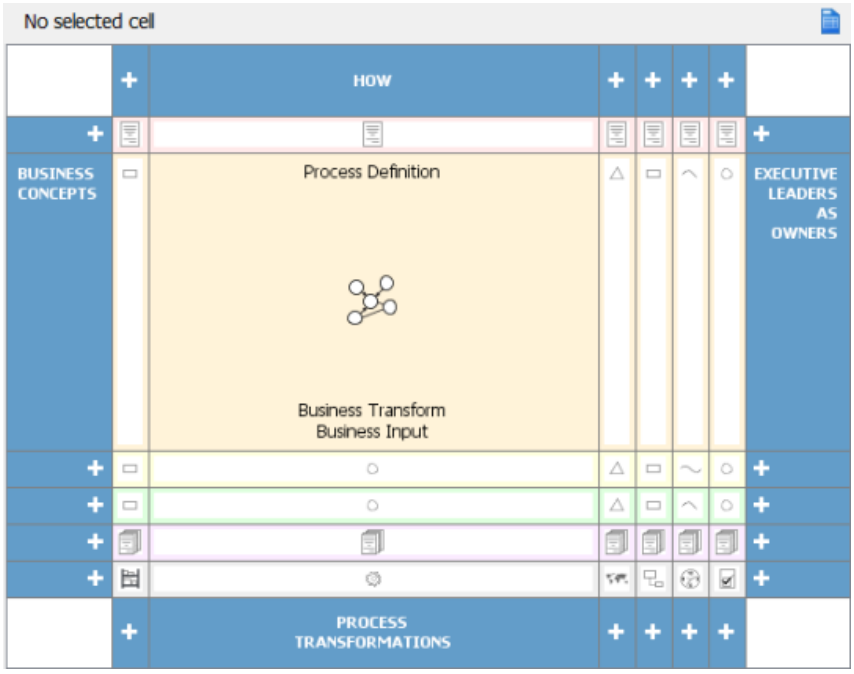
Collapse the Scope Contexts row

Similarly, you can click on the **Collapse** button of column to collapse it.



Collapse the What column

2. Collapse the non-interested rows and columns to make the interested cell remain expanded and dominate the matrix.



Cells collapsed

On the contrary, you can click on the **Expand** button (+) to expand rows/columns.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Full set of enterprise architecture tutorials
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

- Contact us if you need any help or have any suggestion

Business Motivation Model diagram

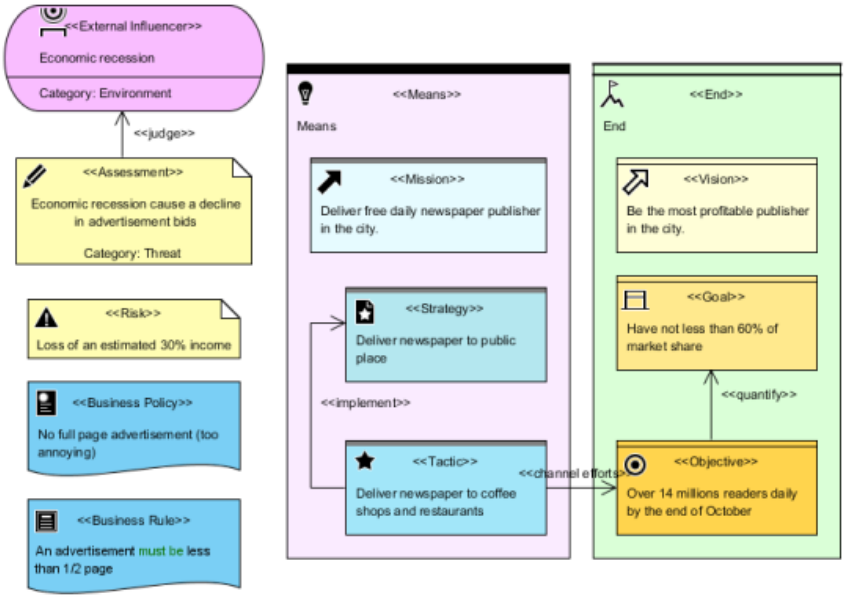
A Business Motivation Model provides business enterprises a set of notations for forming business plans.

Creating Business Motivation Model diagram

Learn how to create a BMM.

Creating business motivation model diagram

A Business Motivation Model provides business enterprises a set of notations for forming business plans. It models what the enterprise wishes to achieve, how to achieve, potential impacts, resources and etc.



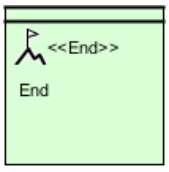
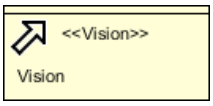
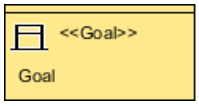
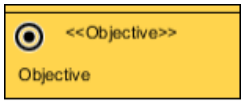
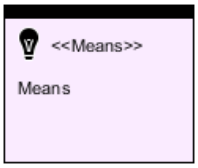
A sample Business Motivation Model diagram

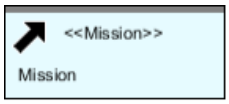
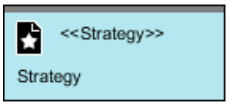
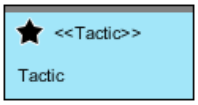
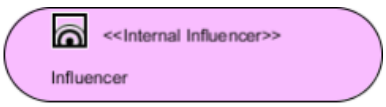
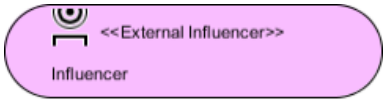
Creating Business Motivation Model (BMM) diagram

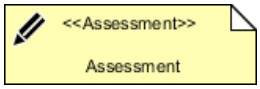
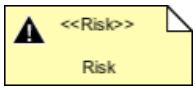
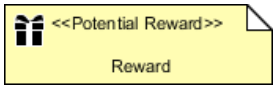
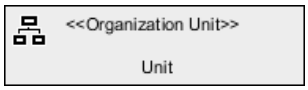
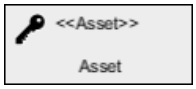
1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Business Motivation Model Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

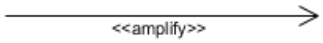
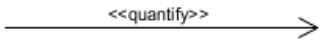
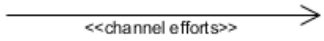
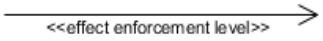
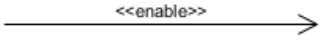
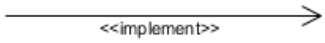
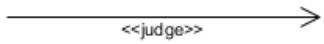
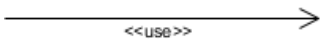
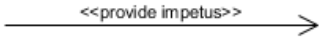
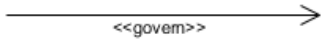
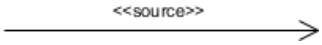
Notations

The description of notations is taken from OMG BMM Specification v1.1.

Name	Representation	Description
End		Ends are about what an enterprise wants to be. Ends can be about changing what the enterprise is (e.g., developing new lines of business, moving into new markets) orabout maintaining its current position relative its market and competition. The definition of an end does not say how it will be achieved.
Vision		A Vision describes the future state of the enterprise without regard to how it is to be achieved. A Vision is the ultimate, possibly unattainable, state the enterprise would like to achieve. A Vision is often compound, rather than focused toward one particular aspect of the business problem. A Goal, in contrast, should generally be attainable and should be more specifically oriented to a single aspect of the business problem.
Goal		A Goal is a statement about a state or condition of the enterprise to be brought about or sustained through appropriate Means. A Goal amplifies a Vision; that is, it indicates what must be satisfied on a continuing basis to effectively attain the Vision.
Objective		An Objective is a statement of an attainable, time-targeted, and measurable target that the enterprise seeks to meet in order to achieve its Goals.
Means		Means are about what an enterprise has decided to do in order to become what it wants to be. A Means is some "device, capability, regime, technique, restriction, agency, instrument or method that may be called upon, activated or enforced to achieve Ends." It does not include either the tasks (business processes and workflow) necessary to exploit it, nor responsibility for such tasks.

		In the Business Motivation Model, Means are organized into Mission, Courses of Action and Directives. A Mission indicates the ongoing operational activity of the enterprise. Its definition should be broad enough to cover all Strategies and the complete area of operations. An enterprise can use the Business Motivation Model without defining a Mission explicitly.
Mission		A Mission indicates the ongoing operational activity of the enterprise. The Mission describes what the business is or will be doing on a day-to-day basis. A Mission makes a Vision operative; that is, it indicates the ongoing activity that makes the Vision a reality. A Mission is planned by means of Strategies.
Strategy		A Strategy is one component of the plan for the Mission. A Strategy represents the essential Course of Action to achieve Ends (Goals in particular). A Strategy usually channels efforts towards those Goals. A Strategy is more than simply a resource, skill or competency that the enterprise can call upon; rather, a Strategy is accepted by the enterprise as the right approach to achieve its Goals, given the environmental constraints and risks.
Tactic		A Tactic is a Course of Action that represents part of the detailing of Strategies. A Tactic implements Strategies. For example, the Tactic "Call first-time customers personally" implements the Strategy "Increase repeat business."
Business Policy		A Business Policy is a Directive that is not directly enforceable whose purpose is to govern or guide the enterprise. Business Policies provide the basis for Business Rules. Business Policies also govern Business Processes. The formulation of a Business Policy, which is always under the enterprise's control is controled by some parties who are authorized to manage, control, or regulate the enterprise by selecting from a variety of alternatives in response to one or more Assessments. Business Policies that exist merely to enable a Strategy in a direct and trivial manner should be avoided. For example, suppose the enterprise has the Strategy "Encourage repeat business." A Business Policy that says "Repeat business should be encouraged" is trivial and does not need to be expressed. In general Business Policies exist to govern; that is, control, guide, and shape the Strategies and Tactics. For example, the Business Policy "We will not make on-site visits" governs the Strategy "Encourage repeat business," as well as the specific Tactics that might be selected to implement the Strategy. Specifically, no Tactic requiring on-site visits will be permitted to support the Strategy; even though on-site visits would probably be effective in that regard. On the other hand, a Tactic involving sending coupons by mail would be acceptable under the Business Policy since it involves no onsite visits.
Business Rule		A Business Rule is a Directive, intended to govern, guide, or influence business behavior, in support of Business Policy that has been formulated in response to an Opportunity, Threat, Strength, or Weakness. It is a single Directive that does not require additional interpretation to undertake Strategies or Tactics. Often, a Business Rule is derived from Business Policy. Business Rules guide Business Processes. Formally, a Business Rule is a rule that is under business jurisdiction. A rule always introduces an obligation or necessity.
Internal Influencer		An Influencer is something that can cause changes that affect the enterprise in its employment of its Means or achievement of its Ends. Alternatively, it might confirm that there are no changes where changes might have been expected. Influencers may be Internal (from within the enterprise) or External (from outside the enterprise boundary). If the enterprise being modeled is an Organization Unit within a larger organization, it may choose to treat the larger organization as an External Influencer. The Business Motivation Model provides an example set of categories of Influencer. In practice, enterprises define their own set.
External Influencer		An Influencer is something that can cause changes that affect the enterprise in its employment of its Means or achievement of its Ends. Alternatively, it might confirm that there are no changes where changes might have been expected. Influencers may be Internal (from within the enterprise) or External (from outside the enterprise boundary). If the enterprise being modeled is an Organization Unit within a larger organization, it may choose to treat the larger organization as an External Influencer.

		The Business Motivation Model provides an example set of categories of Influencer. In practice, enterprises define their own set.
Assessment		An influence (a change caused by an Influencer) is neutral. It is more or less simply just 'there' until the enterprise decides how to react to it. An Assessment is a judgment about the influence on the enterprise's ability to employ its Means or achieve its Ends. The decisions are reflected in changes to the Ends and/or Means. Different people might make different Assessments of a given influence on the same Ends and Means, perhaps even the same people at different points in time. The model supports a record of which people made which Assessments and when, providing an audit trail for future reference. The Business Motivation Model suggests SWOT (Strength, Weakness, Opportunity, Threat) as an example of an approach for making assessments. In practice, enterprises can substitute different approaches. The model also includes Potential Impacts that can be identified to support Assessments. Potential Impacts are categorized as Risk and Potential Reward. As well as more general associations between Assessment, Ends and Means, there is a direct association "Directive is motivated by Potential Impact." This is one of the minor enhancements in version 1.1 of the Business Motivation Model, based on experience of using the model in risk management.
Risk		An Assessment records judgments about the impact (or potential for impact) of some Influencer on Ends and/or Means in terms of Potential Impacts. In other words, an Assessment identifies some Potential Impact(s) that is/are significant to that Assessment. Each Potential Impact is an evaluation that quantifies or qualifies some aspect of an Assessment in specific terms, types or dimensions. A Potential Impact significant to an Assessment can provide the impetus for Directives that govern Courses of Action or support the achievement of Ends. An Influencer may lead to the creation of a Business Policy only through an Assessment having been made that identifies some Potential Impact. Potential Impacts are categorized as follows: Risk, Potential Reward. Typically, Risks are regarded to be negative impacts, whereas Rewards are considered positive.
Potential Reward		An Assessment records judgments about the impact (or potential for impact) of some Influencer on Ends and/or Means in terms of Potential Impacts. In other words, an Assessment identifies some Potential Impact(s) that is/are significant to that Assessment. Each Potential Impact is an evaluation that quantifies or qualifies some aspect of an Assessment in specific terms, types, or dimensions. A Potential Impact significant to an Assessment can provide the impetus for Directives that govern Courses of Action or support the achievement of Ends. An Influencer may lead to the creation of a Business Policy only through an Assessment having been made that identifies some Potential Impact. Potential Impacts are categorized as follows: Risk, Potential Reward. Typically, Risks are regarded to be negative impacts, whereas Rewards are considered positive.
Organization Unit		Organization Unit has two roles: 1. It is a concept in the Business Motivation Model, participating in the following associations: • defines Ends, • establishes Means, • makes Assessments, • recognizes Influencers, • may be defined by a Strategy, and • may be responsible for Business Processes. 2. It is usually the basis for defining the boundaries of the enterprise being modeled. The decomposition of Business Policies, Courses of Action and Desired Results and assignment of responsibilities within the enterprise is often guided by (or, at least, consistent with) the definition of units within the organization structure.
Asset		When Courses of Action are being defined, 'things' that are used in operating the enterprise often have to be considered. They are represented in the Model as Assets, of two kinds: • Fixed Assets - things that are kept long-term, maintained, reused, and perhaps eventually replaced. They can be tangible, such as equipment and buildings, or intangible, such as patents and licenses.

		• Resources - things that are consumed and replenished, such as raw materials, parts, finished goods, and cash.
Amplify		N/A
Quantify		N/A
Channel Efforts		N/A
Effect Enforcement Level		N/A
Enable		N/A
Implement		N/A
Regulate		N/A
Judge		N/A
Use		N/A
Provide Impetus		N/A
Govern		N/A
Responsible		N/A
Source		N/A
Composition		N/A

A list of supported notations in Business Motivation Model (BMM) diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Business planning with business motivation model (BMM) diagram
- Full set of enterprise architecture tutorials
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ArchiMate diagram

ArchiMate is a modeling technique for describing enterprise architectures. You will learn in this chapter how to create archimate diagram, and learn the notations supported by archimate diagram.

Creating ArchiMate diagram

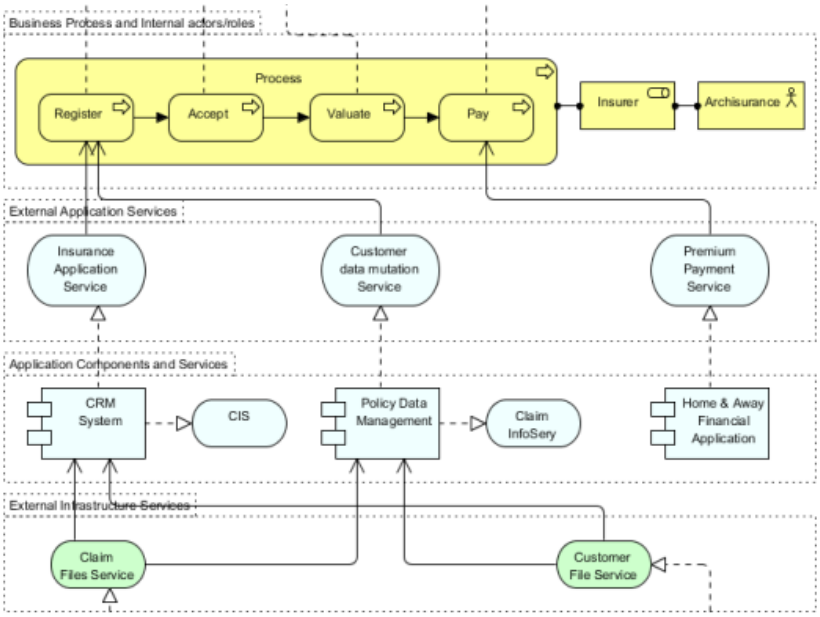
Teaches you how to create and draw an archimate diagram.

Working with Viewpoints

Teaches you how to create standard and user-defined viewpoints

Creating ArchiMate diagram

ArchiMate is a modeling technique for describing enterprise architectures. It divides architecture into three layers - business layer, application layer and technology layer. The business layer offers products and services to external customers. The application layer supports business layer and the technology layer offers infrastructural services for application layer.



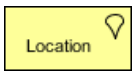
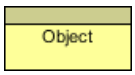
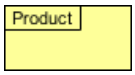
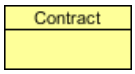
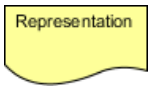
A part of a sample ArchiMate diagram

Creating ArchiMate diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **ArchiMate Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

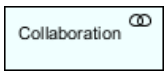
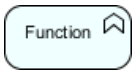
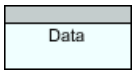
Notations - Business Layer

Name	Representation
Business actor	Actor
Business role	Role
Business collaboration	Collaboration
Business process	Process
Business function	Function
Business interaction	Interaction
Business event	Event
Business service	Service
Business interface	

Location	
Business object	
Product	
Contract	
Representation	
Meaning	
Value	

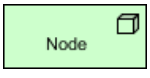
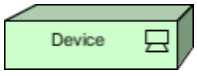
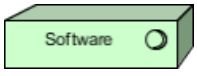
A list of supported notations in ArchiMate diagram, for business layer

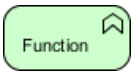
Notations - Application Layer

Name	Representation
Application collaboration	
Application component	
Application service	
Application function	
Application interaction	
Application interface	
Data object	

A list of supported notations in ArchiMate diagram, for application layer

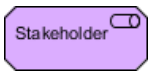
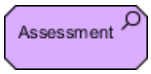
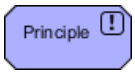
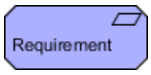
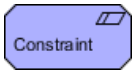
Notations - Technology Layer

Name	Representation
Node	
Device	
System software	

Infrastructure interface	
Infrastructure function	
Infrastructure service	
Artifact	
Communication path	
Network	

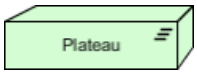
A list of supported notations in ArchiMate diagram, for technology layer

Notations - Motivation Layer

Name	Representation
Stakeholder	
Driver	
Assessment	
Goal	
Principle	
Requirement	
Constraint	
Influence	

A list of supported notations in ArchiMate diagram, for motivation layer

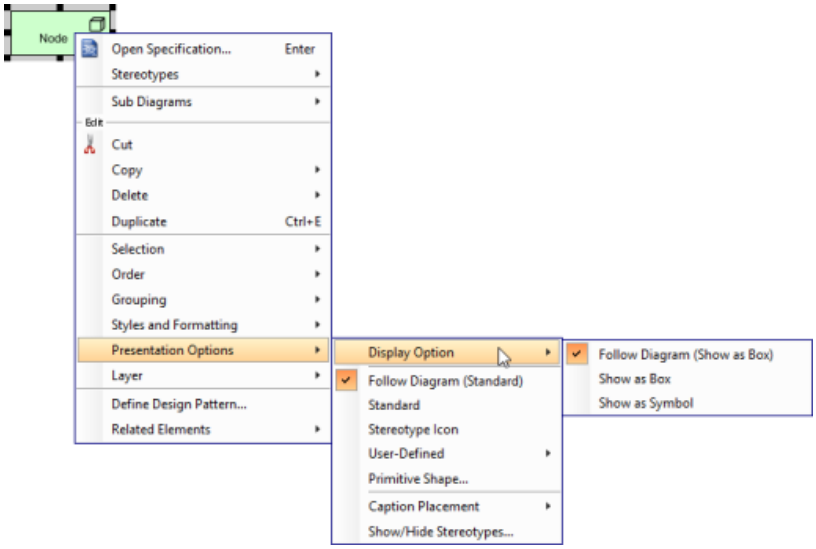
Notations - Implementation & Migration Layer

Name	Representation
Plateau	
Gap	
Deliverable	
Work Package	

A list of supported notations in ArchiMate diagram, for implementation & migration layer

Changing the appearance of some notations

Some of the ArchiMate notations support different ways of presentation. Take node in technology layer as example, to change to another presentation, right click on the node and select **Presentation Options > Display Option, Show as Box/Symbol** from the popup menu.



To change the presentation of a node

Node shown as box or symbol

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Full set of enterprise architecture tutorials
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Working with viewpoint

To develop and maintain an enterprise architecture requires the cooperation between different people, teams and even organizations. These stakeholders have different backgrounds, expertise and responsibilities and have different interests, goals and needs. If there is a means to allow stakeholders to focus on particular aspects of an enterprise architecture, conerns can be better addressed, and this is what ArchiMate's viewpoint aimed to to achieve.

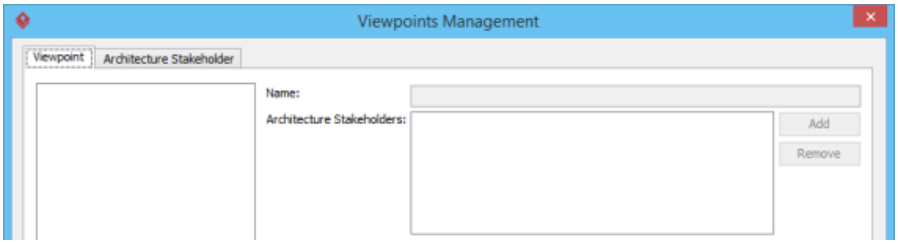
Viewpoint allows enterprise architects, designers and any other stakeholders involved in building, maintaining and participating in an enterprise architecture to define their own views on the architecture. Stakeholders views the portions they are interested in and avoid reading information that they do not care as much about.

In this page, you will learn how to add a standard viewpoint which is suggested by OpenGroup ArchiMate 2.0 specification and how to create your own viewpoint.

Adding a Standard Viewpoint

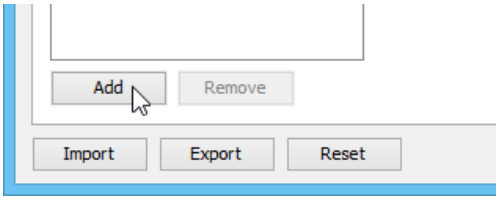
A set of standard viewpoints have been suggested by OpenGroup. In this section, you will learn how to add a standard viewpoint.

1. Select **Modeling > Manage Viewpoints...** from the toolbar.
2. In the **Viewpoint Management** window, the **Viewpoint** tab is opened by default.



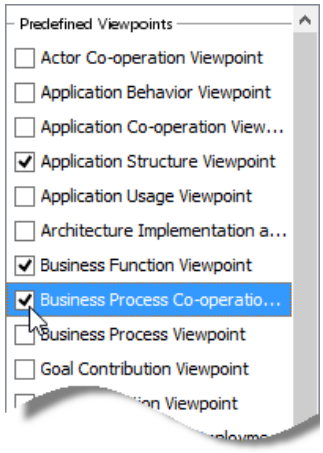
Manage viewpoints

3. At the bottom left, click **Add**.



Add a Viewpoint

4. A list of standard viewpoints are listed, known as the predefined viewpoints. Check the ones that are required by your enterprise architecture (project).



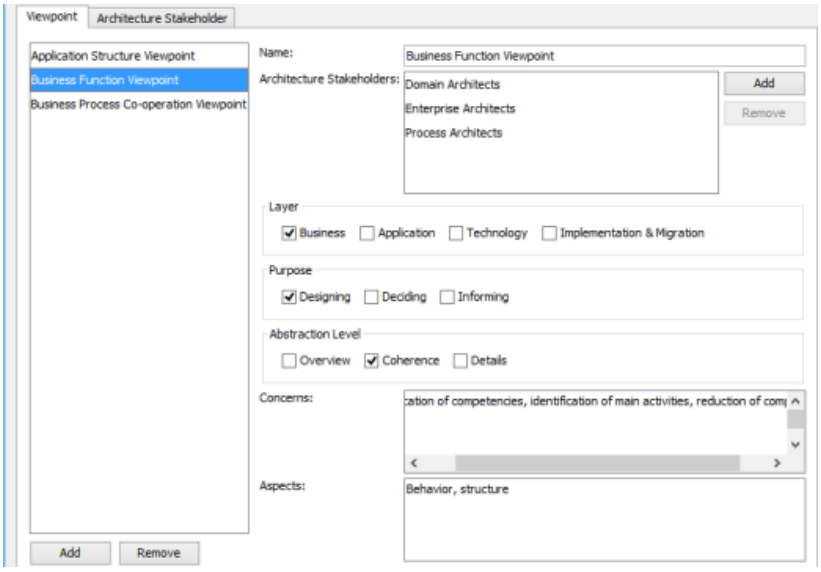
Add standard viewpoints to enterprise architecture

5. Click **Done** at bottom left to confirm your selection.
6. Each predefined viewpoint comes with a set of preset properties such as stakeholders, layers, purpose, abstraction level, concerns and aspects. You may change them by first selecting the viewpoint you want to change on the left hand side and editing the fields on the right hand side. Here is a description of properties:

Property	Description
Name	Name of the viewpoint.
Architecture Stakeholders	People who view the enterprise architecture through the viewpoint.
Layer	The perspective involved. The selection affects the visibility of tools that are available in diagram toolbar in ArchiMate diagram.

Purpose	One of the two dimensions in classifying a viewpoint. The Purpose dimension allows the classification of viewpoint base on the purpose of view. Designing - Support architectures and designers in the design process. Deciding - Assists stakeholders in decision-making. Informing - Inform stakeholders who have a need in understanding the enterprise architecture.
Abstraction Level	One of the two dimensions in classifying a viewpoint. The Abstract Level dimension classify viewpoint base on the level of detail of a view. Details - Typically consider one layer and one aspect from the ArchiMate framework. Coherence - Multiple layers or multiple aspects are spanned. Overview - Addresses both multiple layers and multiple aspects.
Concerns	Conerns of stakeholders to be addressed by the viewpoint.
Aspects	Certain aspects of enterprise architecture.

Caption Here



Properties of a viewpoint

7. Click **OK** to confirm viewpoint management and close the window.

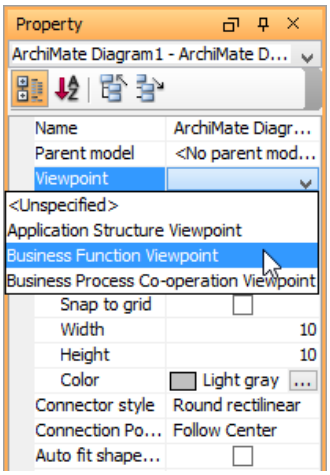
Creating Your Own Viewpoint

Instead of using standard viewpoints, you may also define your own viewpoints, with your own stakeholders. To create your own viewpoint:

1. Select **Modeling > Manage Viewpoints...** from the toolbar.
2. In the **Viewpoint Management** window, select the **Viewpoint** tab.
3. At the bottom left, click **Add**.
4. Click **New**.
5. Enter the name of the stakeholder.
6. You may optionally add the stakeholder to existing viewpoints now.
7. Specify the documentation.
8. At the bottom left, click **Done** to confirm the creation of stakeholder.

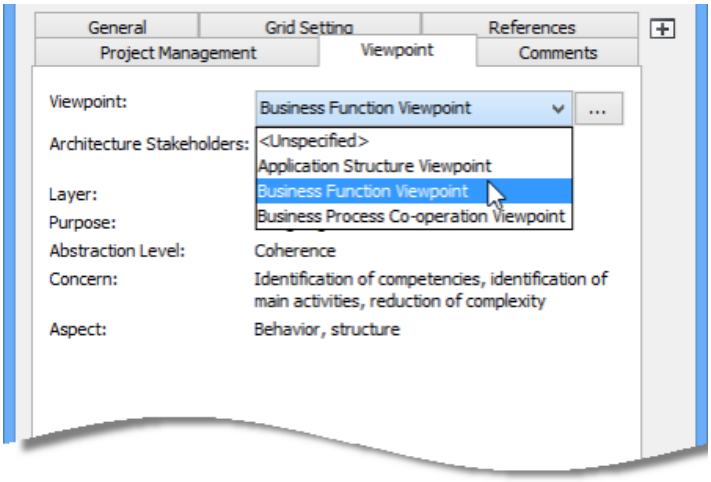
Assign a Viewpoint to Diagram (i.e. View in ArchiMate)

Once you have added or created a viewpoint, you can assign the viewpoint to diagram(s). The diagram with viewpoint specified is supposed to be designed for the stakeholders listed in that viewpoint. There are two possible ways to set viewopoint. The first way is to set via the **Property Pane**. To open the **Property Pane**, select **View > Panes > Property** from the toolbar.



Set viewpoint via Property Pane

The second way is to set via the diagram specification window. Right click on the diagram and select **Open specification**, open its **Viewpoint** tab and set viewpoint there.



Set viewpoint via diagram specification window

Creating a Stakeholder

When you add a standard viewpoint, the stakeholders involved will be added to your project automatically. In addition to these 'default stakeholders', you can create your own to suit your business and problem domain.

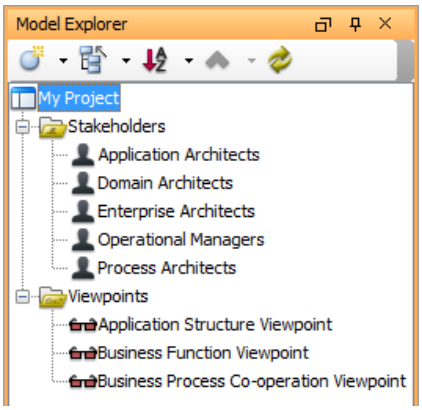
To create a stakeholder:

1. Select **Modeling > Manage Viewpoints...** from the toolbar.
2. In the **Viewpoint Management** window, select the **Architecture Stakeholder** tab.
3. At the bottom left, click **Add**.
4. Click **New**.

Browsing an Enterprise Architecture with Viewpoint

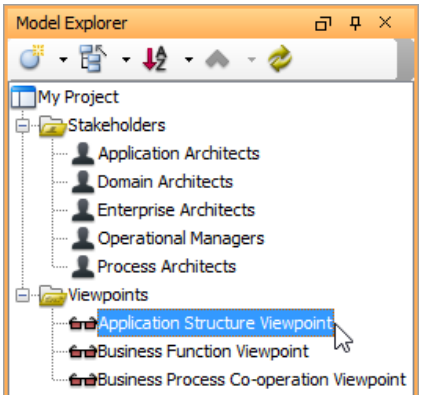
You can list the diagrams that have been assigned with certain viewpoint, or to list diagrams based on certain stakeholder. To do these:

1. Select **View > Panes > Model Explorer** in the toolbar. After openning the **Model Explorer**, you should see two nodes - Stakeholders and Viewpoints, with stakeholders and viewpoints listed.



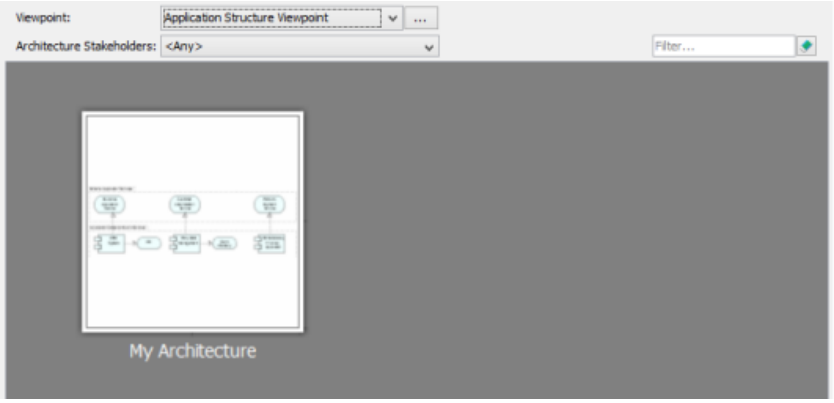
Model Explorer

2. If you want to browse enterprise architecture base on a stakeholder or viewpoint, double click on it.



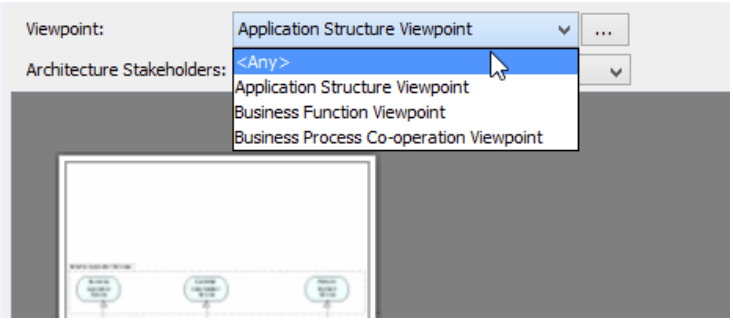
Browse an enterprise architecture from viewpoint

You will see the diagram(s) that is associated with the chosen stakeholder/viewpoint will be displayed.



Browsing enterprise architecture

3. If you want to browse with another stakeholder/viewpoint, you can update the selection in the drop down at the top of the panel.



View another stakeholder

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Full set of enterprise architecture tutorials

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business rule

A business rule defines guideline with necessary constraint(s) needed for executing certain business operations. You will learn how to manage business rules.

Managing business rules

Shows you how to use the rule editor to edit business rule.

Business rule grid

Shows you how to use the rule grid to manage business rules.

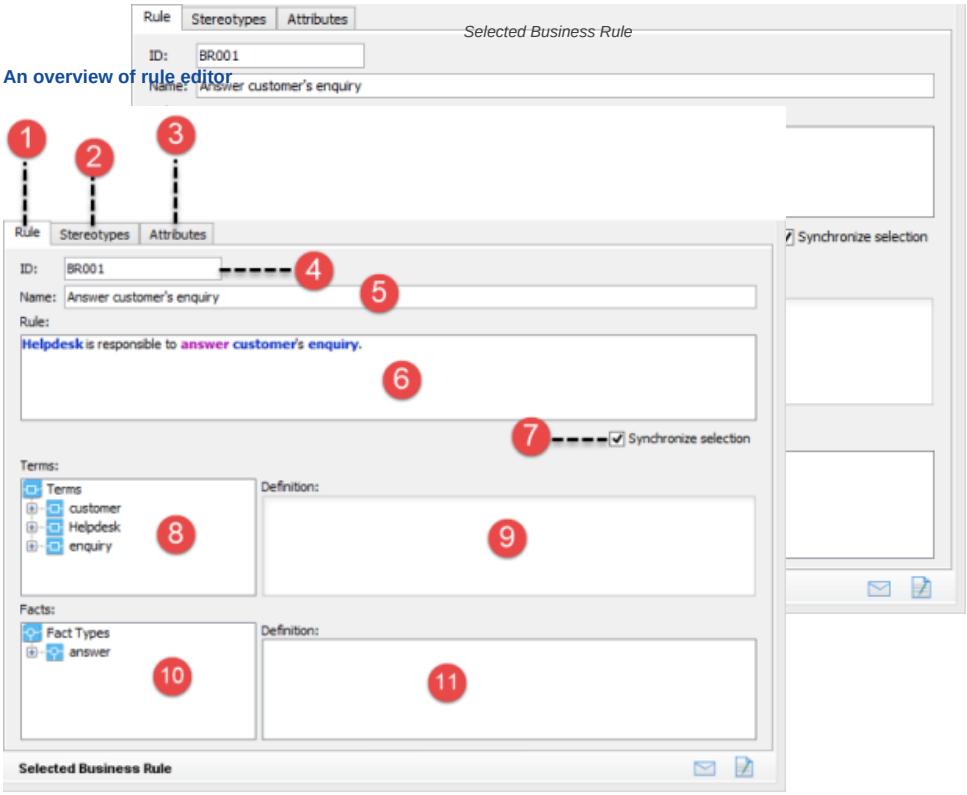
Managing business rules

A business rule defines guideline with necessary constraint(s) needed for executing certain business operations. You can record and describe business rules with rule editor as well as to identify the term (vocabulary) involved in the rule, which helps tracing fact concepts around rules.

Defining a business rule

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Business Rule**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.
6. This shows the rule editor. An ID is a set of number indicating the order of rule creation that is assigned automatically. You may change it if you like. Name the rule with a short and descriptive phrase. Fill in the rule content in the **Rule** field.

An overview of rule editor



Overview of rule editor

No.	Name	Description
1	Rule	A tab that shows the primary rule information such as its name and definition

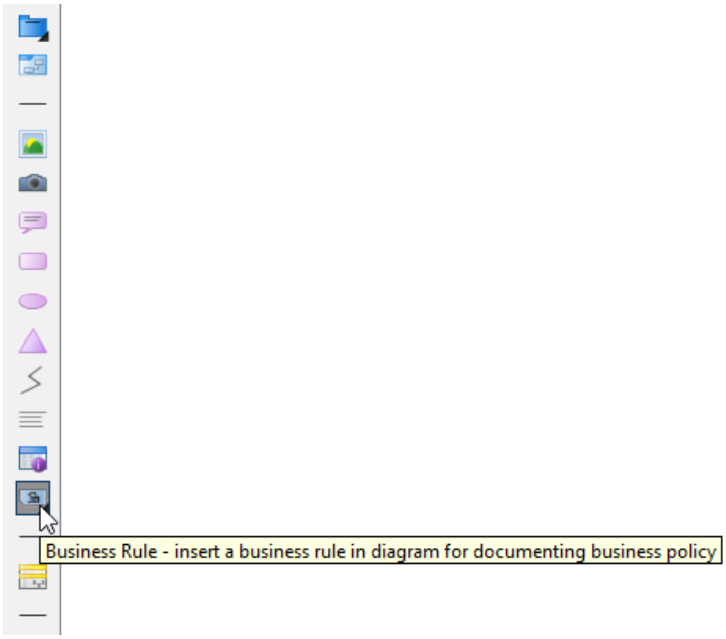
2	Stereotypes	A list of stereotypes applied to the rule. You can extend a rule from a stereotype to add domain specific meaning to it. For example, you can extend a rule from stereotype critical to represent an important rule.
3	Attributes	Read, add and remove attributes from the rule. Attributes can be added to denote extra properties to a rule.
4	ID	A value that makes each rule unique. When you create a rule, an ID will be assigned automatically. The assigned ID indicates the order of rule creation.
5	Name	A short phrase that describes the rule.
6	Rule definition	A longer and more detailed description of rule.
7	Synchronize selection	When the checkbox is checked, the **Terms** and **Facts**' active node selection will follow the selection as pointed by the mouse pointer in the **Rule** definition field.
8	Terms	A list of terms that involve in the rule definition.
9	Term definition	By selecting a term in Terms list, its definition will appear in the Definition field.
10	Facts	A list of facts that involve in the rule definition.
11	Term definition	By selecting a term in Facts list, its definition will appear in the Definition field.

Description of rule editor

Visualizing business rule on diagram

Instead of creating a business rule as described above, you can also create a business rule shape on a diagram. Although there is no specific type of diagram made for presenting business rule, you can draw business rule on any type of diagram. To draw a business rule on a diagram:

1. Select **Business Rule** from the toolbar.

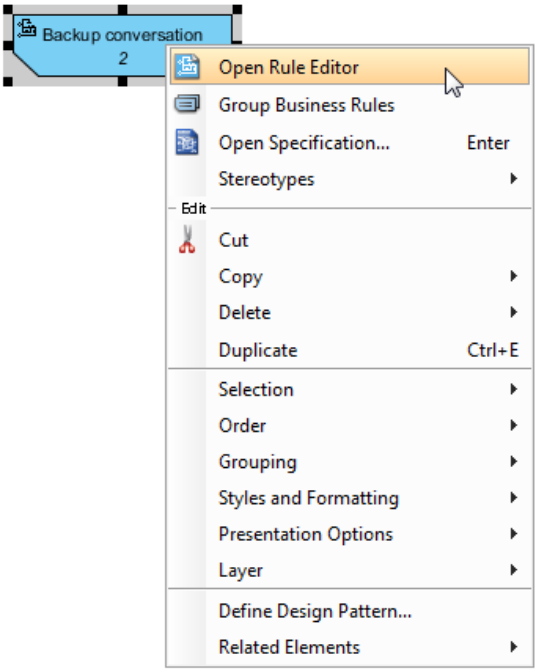


Selecting Business Rule in diagram toolbar

2. Click on the diagram to create a business rule. Name the rule with a short and descriptive phrase and press **Enter**to confirm.

A business rule is created

If you need to describe the rule in detail, right click on the rule shape and select **Open Rule Editor** from the popup menu. After that, fill in the rule definition in rule editor as mentioned above.

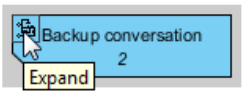


Open the rule editor to define the rule

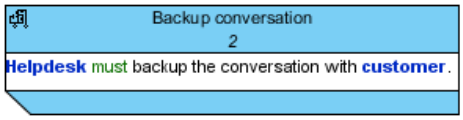
Expanding and collapsing rule

A business rule shape has two visual states - collapsed and expanded. While in collapsed state, the business rule hides away the definition of rule, in expanded state, a new compartment will appear in the middle rule shape for showing the rule definition.

To expand or collapse a rule shape, click on the top-left corner of rule shape.



Expand a collapsed rule shape

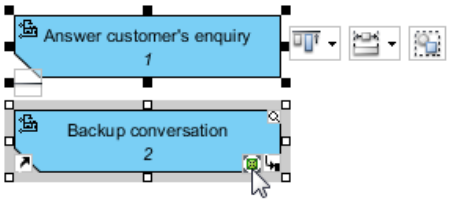


An expanded rule shape

Business rule group

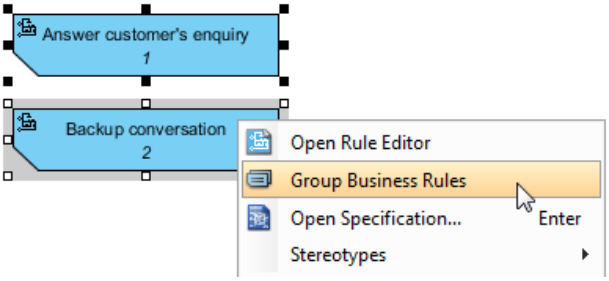
If you find a number of rules are of the same category, you can group them.

1. Select the rule shapes.

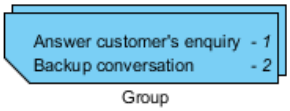


Select rules

2. Right click on the selection and select **Group Business Rules** from the popup menu.



To group rules



Rules are grouped

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

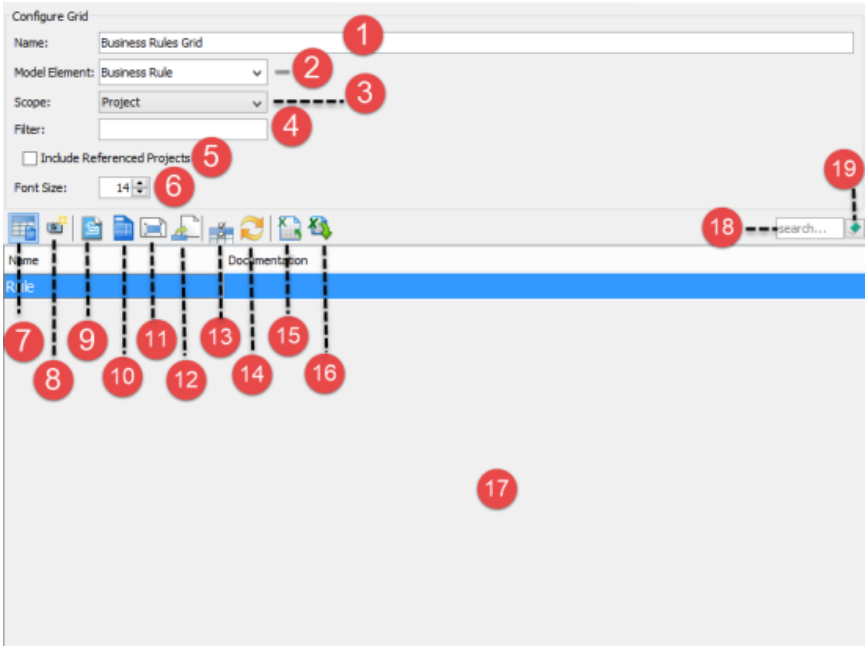
Business rule grid

Business rule grid is a table with business rules listed in it. It allows you to access all business rules in a project or diagram, lookup and create business rule.

Creating business rule grid

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Business Rule Grid**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

The overview of Business Rule Grid



The Business Rule Grid

No.	Name	Description
1	Name	The name of this **Business Rule Grid**.
2	Model Element	Change the type of model element to be listed in grid. Although **Business Rule** is selected by default, you may change it to other types.
3	Scope	The location to look for the business rules to list in grid. By default, business rules are found from the whole project. You can change to find use cases from specific model or package, or to find only business rules right at the root level. You can also restrict the scope to all diagrams, to within a specific diagram or to all business rules that has not been visualized in any diagram.
4	Filter	Apply filter to grid content. Text entered here is matched against the **Name** property of business rules listed in grid. Business rules that do not contain the entered text in their name are hidden.
5	Include Referenced Projects	Check it to list also business rules in referenced projects, in **Business Rule Grid**.
6	Font Size	Click to adjust the font size of text in **Business Rule Grid**.
7	Configure Grid	Click to show/hide the grid configuration panel, which allows you to enter the name of grid, the model element to be listed in grid, the scope and to apply filter to grid content.
8	New Business Rule	Click to create a business rule.
9	Open Rule Editor	Select a business rule in **Business Rule Grid** and click this button to open the rule editor.
10	Open Specification...	Select a business rule in **Business Rule Grid** and click this button to open its specification.
11	Open Business Rule Details	Select a business rule in **Business Rule Grid** and click this button to acces its details in **Business Rule Details** editor.
12	Show View...	Select a business rule in **Business Rule Grid** and click this button to list the diagrams that contains the view of the selected business rule.
13	Visualize...	Select a business rule in **Business Rule Grid** and click this button to show it in a new or existing diagram.

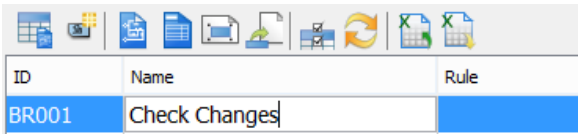
14	Configure Columns...	Click to select the property(ies) of business rules to be listed in the grid, as columns.
15	Refresh	Click to refresh the grid content by showing the most updated information of business rules listed.
16	Export to Excel	Click to export grid content to Excel file.
17	Import from Excel	Click to import grid content from exported Excel file.
18	List of business rules	Business rules are listed here.
19	Search	Find business rule(s) by entering search criteria.
20	Clear	Click to clear the text entered in **Search** box.

The fields in Business Rule Grid

Creating business rule in Business Rule Grid

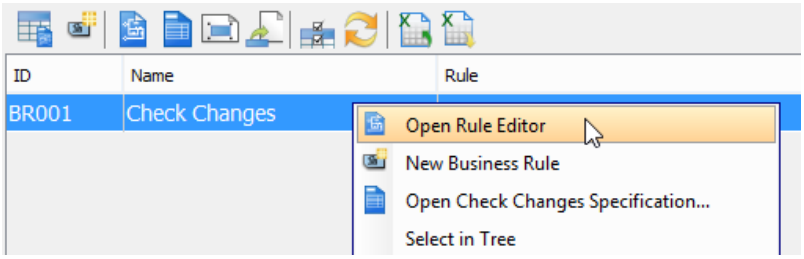
To create a business rule in **Business Rule Grid**:

1. Click on **New Business Rule** above the **Business Rule Grid**.
2. Enter the name of business rule.



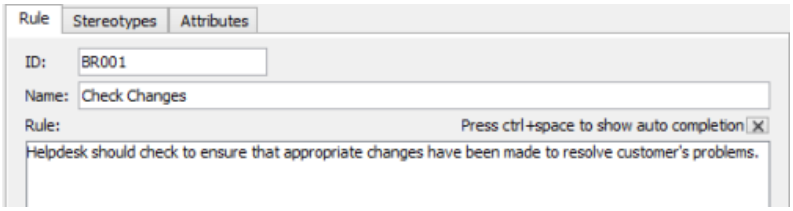
Creating business rule in Business Rule Grid

3. Press **Enter** to confirm editing.
4. To enter the details of business rule, right click on the new business rule and select **Open Rule Editor** from the pop-up menu.



Open Rule Editor

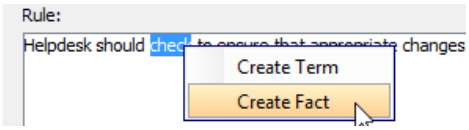
5. In **Business Rule Editor**, enter name and rule for the newly created business rule.



Business Rule Editor

Creating fact

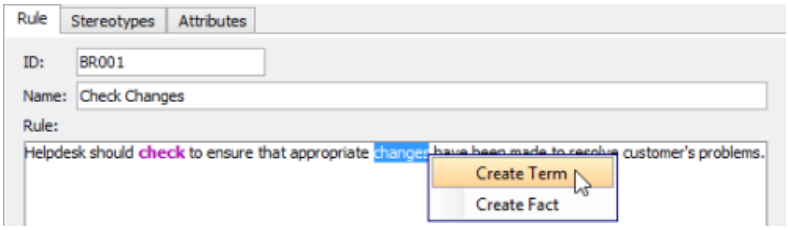
Highlight a word which you want to create to be a fact and select **Create Fact** from the pop-up menu.



Create check as fact

Creating term

Highlight a word which you want to create to be a term and select **Create Term** from the pop-up menu.



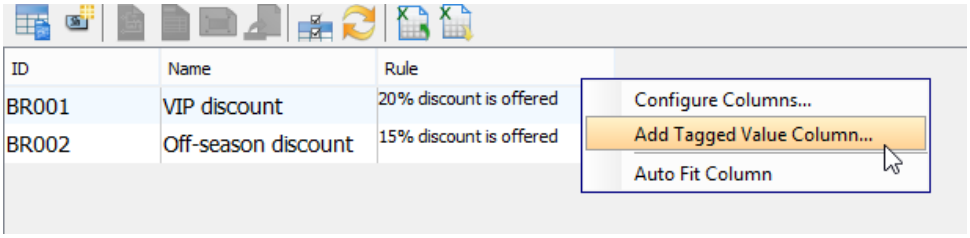
Create changes as term

Adding new tag to business rules in grid

Sometimes, you may find the standard modeling notations not enough in supporting your domain specific needs. For example, if you are working on a project that relies heavily on the use of third-party libraries, you might want to specify in UML component diagram the API source of those components modeled. Usually, this kind of specific needs cannot be directly supported by the standard notations.

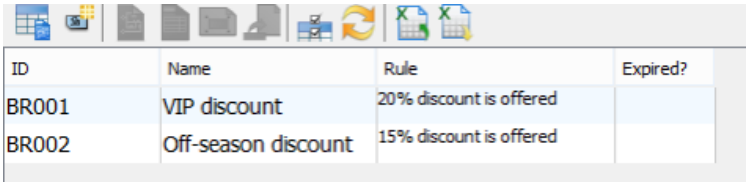
Visual Paradigm supports the use tagged values, which allows designers to extend the vocabulary of UML in order to create new model elements. With the use of stereotype and tagged values, designers can introduce model elements with domain/problem specific properties. There are several ways you can take to create tagged values to model elements. One is to create in a Grid. This method is extremely efficient when you want to add into and fill in the tags for multiple model elements in same type. To create a tag to business rules in Business Rule Grid:

1. Right click on the column header at the top of a grid and select **Add Tagged Value Column...** from the popup menu.



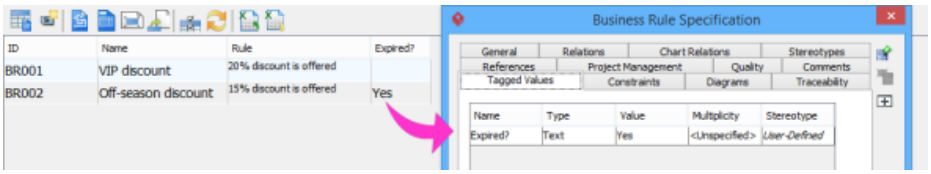
To add a tagged value column

2. Enter the tag name in the **Input** dialog box and click **OK** to confirm.
3. You will see a new column appear in the last column of the grid.



New "Expired?" column added to a grid

4. Now, you can double click on a cell in that column to enter the value of the property (tag). When and only when you have entered a value, a tag will be added into that specific model element. Note that once a tag is added, you cannot remove the tag by clearing the cell content. If you want to remove the tag, you have to do it in the model element specification window.



Tagged value added and specified

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Fact diagram

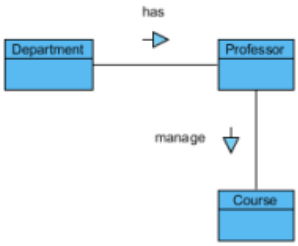
Record and related terminologies under specific business domain by using fact diagram.

Creating fact diagram

Shows you how to use the fact diagram to model the relationships between terms in a business domain.

Creating fact diagram

Record and relate terminologies under specific business domain by using fact diagram. In a fact diagram, terms are visualized as rectangular blocks. You can link terms up with connectors and specify the kind of relationship in between. A fact diagram is closely related to business rules. It helps you to identify the rules by studying the relationship between terms.

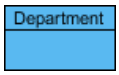
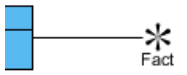
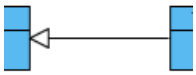


A sample fact diagram

Creating fact diagram

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Fact Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Notations

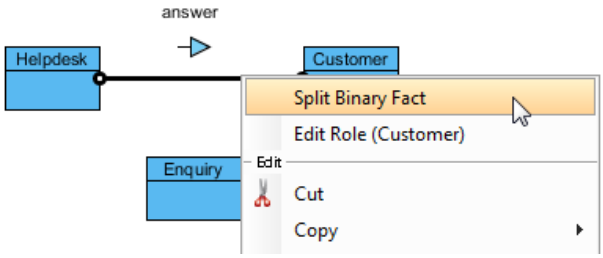
Name	Representation	Description
Term		A meaningful noun or noun phase that business participants recognize and use in communication.
Fact Type		A fact type represents the relationship between terms.
Fact Association		A fact association connects terms for visualizing the relationship in between.
Term-Fact Type Association		By connecting a term to a fact type, it represents a u-nary fact-type that provides a simple yes or no answer to business question.
Generalization		Represents a parent and child relationship between terms.

A list of supported notations in fact diagram

Spliting and joining binary fact

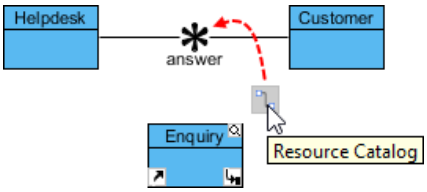
When a fact type involves more than two terms, we call it a n-ary fact type. For example, fact type worded "helpdesk answer customer's enquiry" involves three terms helpdesk, answer and enquiry. If you have already created a fact model that involves two terms and now want to add an additional one, you need to split a fact type and connect the split fact type with the new term.

1. Split a fact type by right clicking on the fact association and selecting **Split Binary Fact** from the popup menu.



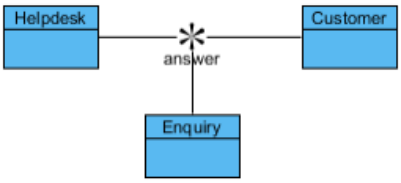
Split a binary fact

2. Move the mouse pointer over the added term. Press on the **Resource Catalog** button and drag to the split fact type.



To connect term with fact type

3. Release the mouse button. This connects the new term with split fact type.



Term is connected with fact type

On the contrary, you can join a split fact type by right clicking on an association and selecting **Join Fact Type** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Decision diagram

Represent complex decision situations and business rules in the most simplest, organized manner.

Creating decision table

Shows you how to create and understand decision table.

Documenting rules, conditions and actions in decision table

Shows you how to edit the documentation of rules, conditions and actions in decision table

Creating decision table

In order to develop a truly functional information system, features must be designed and developed based on users' business needs with behaviors following the business rules strictly. Decision table provides a compact way to represent complicated business rules. Thanks to the easy-to-comprehend layout, decision table can be understood by developers and end-users easily.

Decision table involves three sections - conditions, actions and rules. From developer's point of view, decision table is pretty much like the tabular form of an if-then-else statement. While business users use decision table to document business rules, system developers study a decision table to think about the right way to implement those rules.

	Rules					
Conditions	1	2	3	4	5	6
C1. Infant passengers (< 2)	Y	Y				
C2. Youth passengers (>2 and <16)			Y		Y	
C3. Domestic flights	Y					
C4. International flights		Y				Y
C5. Early reservation				Y	Y	
C6. Off-season traveling						Y
Actions	1	2	3	4	5	6
A1. Offer 10% discounts			Y	Y		
A2. Offer 15% discounts						Y
A3. Offer 20% discounts					Y	
A4. Offer 70% discounts		Y				
A5. Offer 80% discounts	Y					

Decision table sample

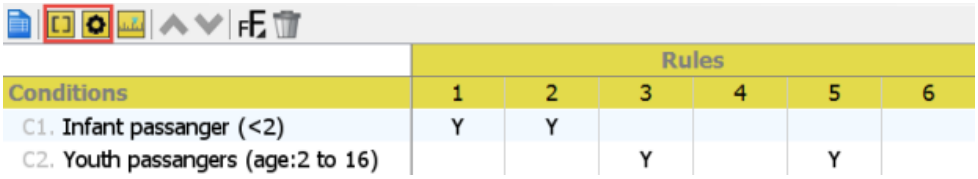
The condition rows in a decision table list out the factors that can influence the final decision. The action rows list out the possible operations to perform. Each of the rule columns represents a combination of condition(s) and action(s), meaning that when one or more conditions are met, action or multiple actions will be performed accordingly. Decision table does not enforce any rule regarding to the way cells are filled. Although people usually use simple true/false (or simply T/F, Y/N) values to represent the matching of conditions and actions, some prefer using ticks. There is real limitation though.

Creating decision table

1. Select **Diagram > New** from the toolbar.
2. In the **New Diagram** Window, select **Decision Table**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Creating conditions and actions

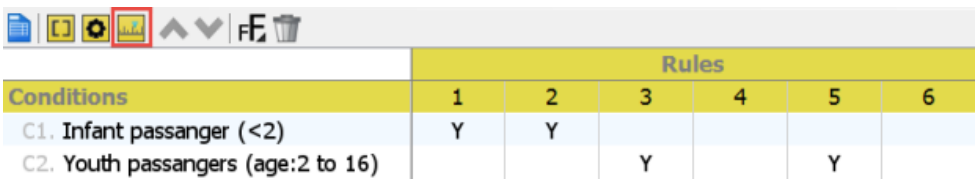
The first two buttons in the toolbar at the top of the decision table allows you to create conditions and actions respectively. By selecting an existing condition or action and click on the create button, a new condition/action will be inserted after the selected condition/action. Then, enter the description of condition/action. Alternatively, you can right click on a condition/action and add a new one after it via the popup menu.



Buttons to create conditions and actions

Creating rules

The third button in the toolbar at the top of the decision table allows you to create rules. By selecting an existing rule and click on the create button, a new rule will be inserted after the selected rule. Find the conditions and actions that match your rule. Double click on the corresponding cell and place a mark on it. For example, enter "Y" to indicate "Yes". Alternatively, you can right click on a rule and add a new one after it via the popup menu.



Buttons to create conditions and rules

Reorder conditions, actions and rules

To reorder conditions and actions, select them in the table and click on the **Move Up/Down** button in toolbar or right click on them and select **Move Up/Down** from the popup menu. To reoder rules, select and right click on them, select **Move Left/Right** from the popup menu.

Delete conditions, actions and rules

To delete conditions, actions and rules, select the rows or columns to delete, click on the **Delete** button in toolbar.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

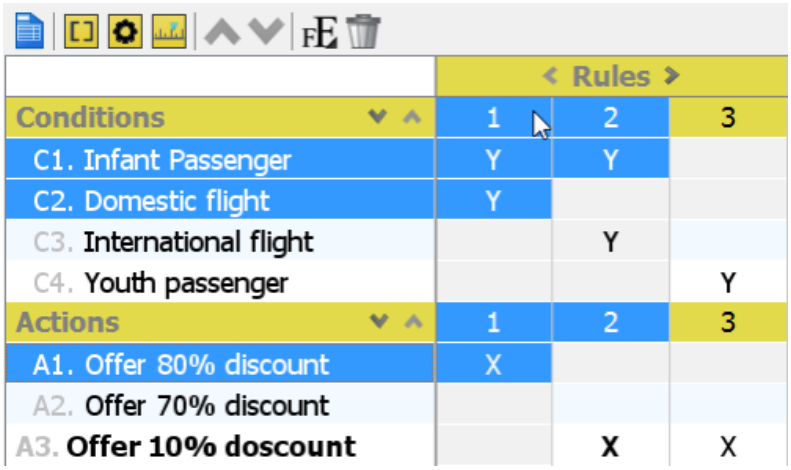
- What is decision table?
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Describing rules, conditions and actions in decision table

You may want to describe the reason why the rules are setup in the way they are in a decision table. You can do this by editing the description of rules added in a decision table. Besides rules, you can also edit the description of conditions and actions.

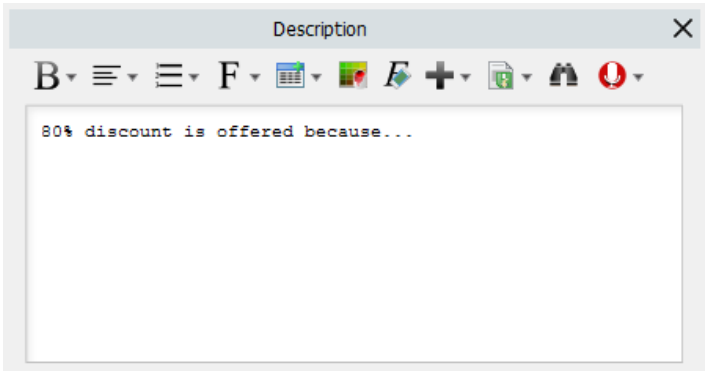
Describing rules, conditions and actions in decision table

1. Select the desired rule in the decision table.



Selecting a rule

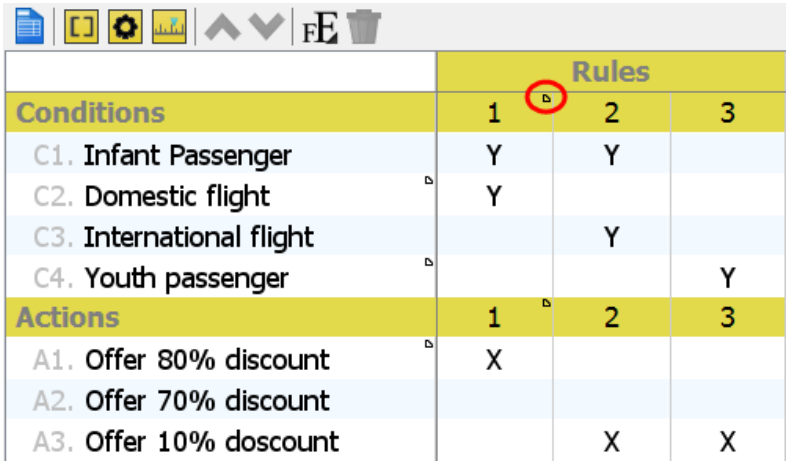
2. Click the **Show Description** button at the right hand side of the status bar to show the **Description** pane. Edit the description in the **Description** pane.



description of a rule

Identifying rules, conditions and actions that have description

Dog ear icons are shown in rules, conditions and actions that have description. You can read the description by selecting a rule/condition/action and viewing the **Description** pane.



Dog ear icon

Identifying conflicted rules

When two or more rules in a decision table are form with the same set of conditions, they are said to be conflicted. You will see an exclamation mark near those conflicted rules. It is recommended to resolve the conflicts to avoid ambiguity.

	Rules						
Conditions	1	⚠ 2	3	4	5	⚠ 6	7
C1. Address proof provided	N		Y	Y	Y		Y
C2. Identity proof provided		N	Y	Y	Y	N	Y
C3. Loan amount < monthly salary			Y				
C4. Loan amount >= monthly salary					Y		Y
C5. Loan purpose				Home purchase	Pay tax		Other
Actions	1	⚠ 2	3	4	5	⚠ 6	7
A1. Approve loan request immediately			X		X		
A2. Review loan request manually				X		X	X
A3. Reject loan request	X	X					

Conflicted rules in a Decision Table

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- What is decision table?
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

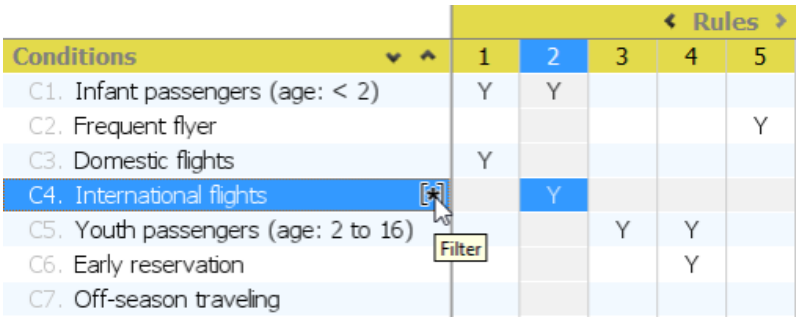
Filter in decision table

The size of a decision table may varies from a few rules to several thousand rules. While a small table is easy to read, a big table is not. To solve this problem, applying filter(s) on a decision table will be useful. A filter can assist you to screen out non-interested rule(s), so that you can focus on rule(s) that match(es) with a specific Conditions or Actions only.

How to apply a filter?

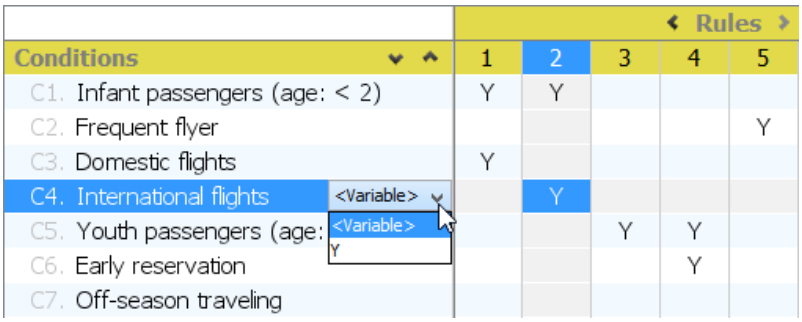
Filters can be applied in both the **Conditions** column and **Actions** column. Let's say you are planning to go for a trip and you want to know what kind of discount(s) you can get if you take an International flight. In this case, you need to set a filter in International flights (the Condition) and observe what the discount(s) (the Actions) is/are. To set a filter:

1. Click the **Filter** button on the right of the International flights column.



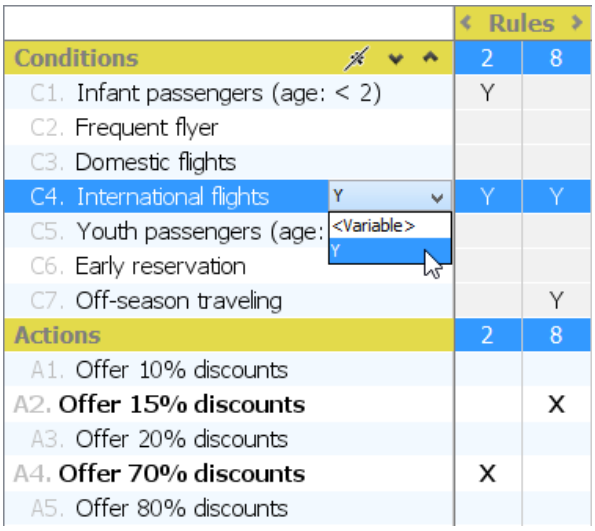
Set filter

2. Click the down arrow button. You can see the option <*Variable*> and *Y*.



Set filter

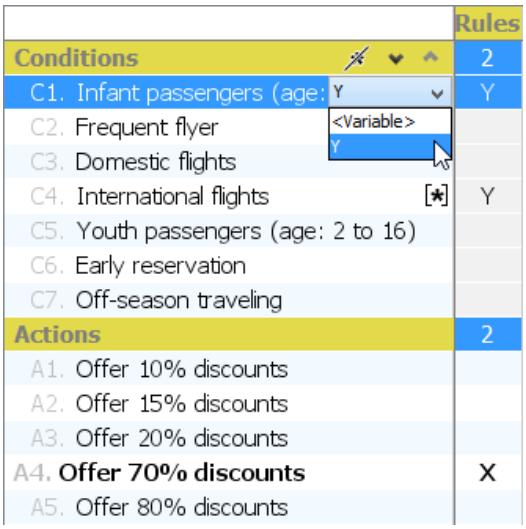
3. Click *Y*. As a result, only those Action(s) that meet the Condition, International flights is/are appeared below. In this case, the Actions of International flights are shown and other Actions which are unrelated to International flights are screened out. In this way, the table becomes more user-friendly to read.



Set filter

Moreover, you can set multiple filters to more than one Conditions/ Actions. For example, other than International flights, if you want to buy the flight ticket for your child which is under 2 year-old. In this case, one more Condition needs to be fulfilled which is *Infant passengers (age: <2)*. Then, you can set one more filter in the *Infant passengers (age: <2)* column.

Consequently, only the **Action**, *Offer 70% discounts* is shown in the Actions column. In other words, only this **Action** satisfies the **Conditions**, *International flights* and *Infant passengers (age: <2)*.



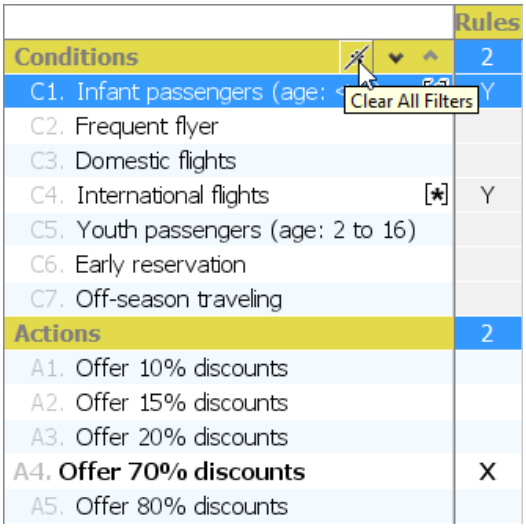
Set multipler filters

Please note that except setting filters in **Conditions** column to look for Action(s), you can also set filters in **Actions** column to look for **Conditions**.

How to clear all filters?

After applying filters to focus on certain information, you might want to see the whole decision table again. In this case, you need to clear all filters. To do this:

Select the **Clear All Filters** button near to the right of the **Conditions** column. The decision table will then be recovered.

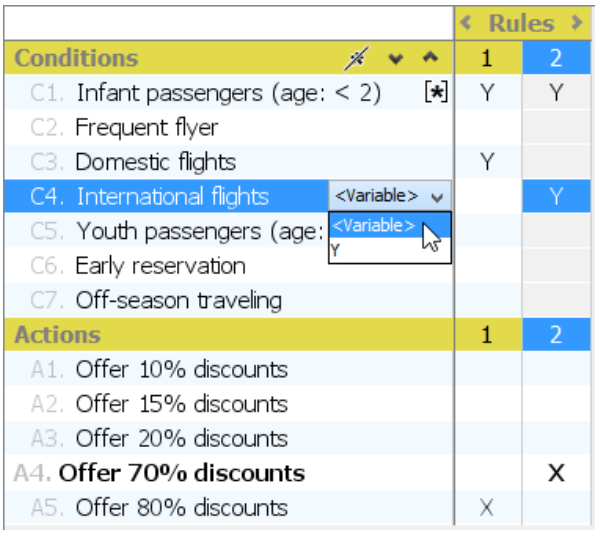


Clear all fitlers

How to clear a single filter?

Sometimes, instead of clearing all filters, you might want to clear only 1 Condition and keep the other filters. Consider the second case above where we set filters on both Conditions, International flights and Infant passengers (age: <2). For instance, you want to clear only the filter in International flights and keep the filter in, International flights and Infant passengers (age: <2). To do so:

1. Click the **Filter** button on the right of the *International flights* column.
2. Click the Down Arrow button. You can see the options <*Variable*> and *Y*
3. Click <*Variable*> this time. Then, only the filter in the **Conditions**, International flights will be removed.



Clear single filter

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- What is decision table?
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Contact us if you need any help or have any suggestion

Process Simulation

With process simulation, you can simulate the execution of business process for studying the resource consumption (e.g. human resources, devices, etc.) throughout a process, identifying bottlenecks, quantifying the differences between improvement options which help study and execute process improvements. This chapter provides you with detailed information about process simulation.

What is simulation?

Introduce simulation, and shows you some of the key features like simulation and resource chart.

Simulation control panel

The Simulation Control Panel is where you can configure and run a simulation. It consists of several parts, and will be described in detail.

Simulating business process

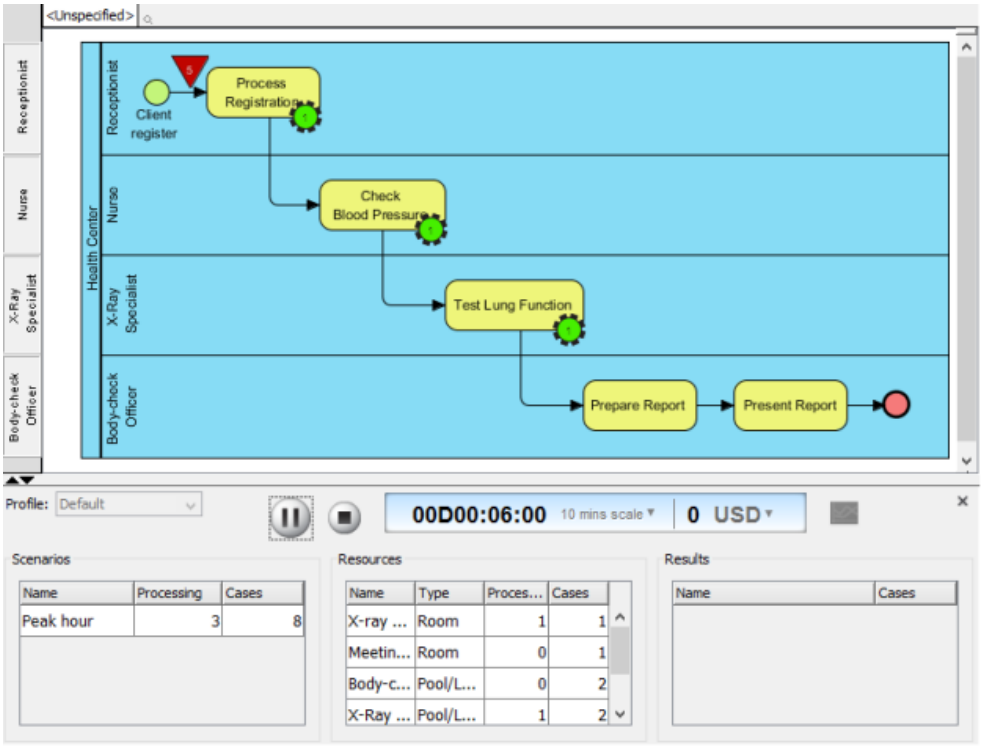
This page is aimed to give you an example to show you how to simulate a business process diagram.

Simulation charts

In addition to process simulation, you can produce charts to aid in the analysis of process performance.

What is process simulation?

The objective of performing business process modeling is to facilitate the communication with stakeholders to perform cost and benefit analysis and to perform process improvement, etc. Simulation is a set of value-added tools designed to aid business process modeling. With simulacian, you can simulate the execution of business process for studying the resource consumption (e.g. human resources, devices, etc.) throughout a process, identifying bottlenecks, quantifying the differences between improvement options which helps study and execute process improvements.

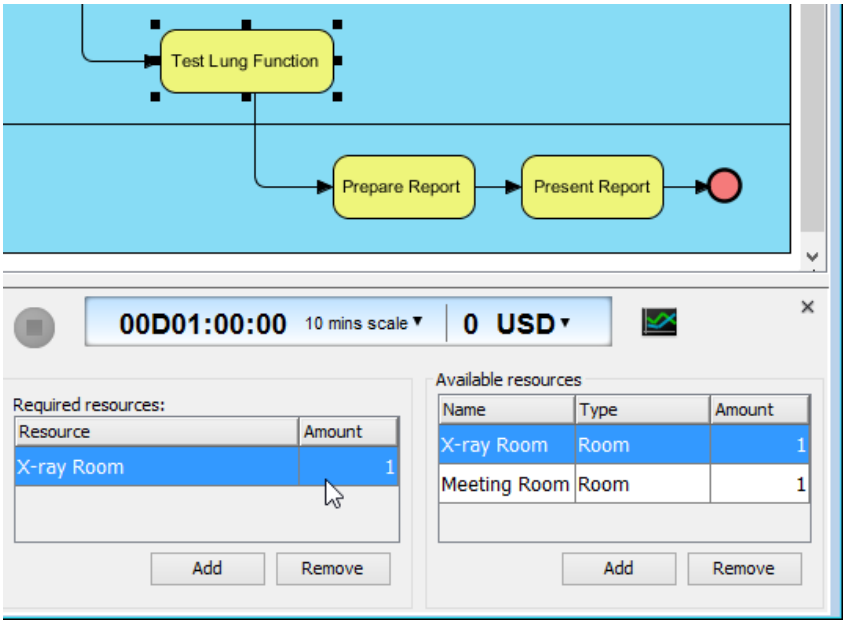


Simulating a business process

Key concepts

Resources

Resources refer to any kind and form of input essential for the execution of a process. Each resource has three properties - name, type and amount. There are two types of resources - available resources and required resources. Available resources refer to the resources that can be used by business process but may not be fully used. For example, a post office has 10 counters as resources but only 3 are in used at peak hours. Required resources is a flow object wide option. You can set the resource and the amount of resource required by completing a flow object. For example, task *Answering Enquiry* requires 1 counter as resource.



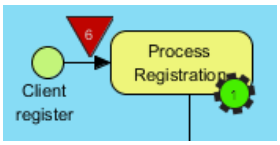
Required and available resources

Very often, the allocation of resource is critical to the efficiency of a business process. For example, if there are more available staffs and counters, this helps to increase the efficiency of customer service. But of course, if the available staffs and counters are more than enough, those non-used resources are wasted. With simulacian, you can determine the optimal resource allocation by evaluating the resource consumption of current process.

Duration
Duration is the time elapsed from the entering of flow object until the leaving of that flow object. It is understandably that the duration of flow object has significant effect to the efficiency of a business process. Imagine if it takes 5 minutes to complete the process of just one payment in a supermarket, a long queue will be acculumated to wait for trancation.

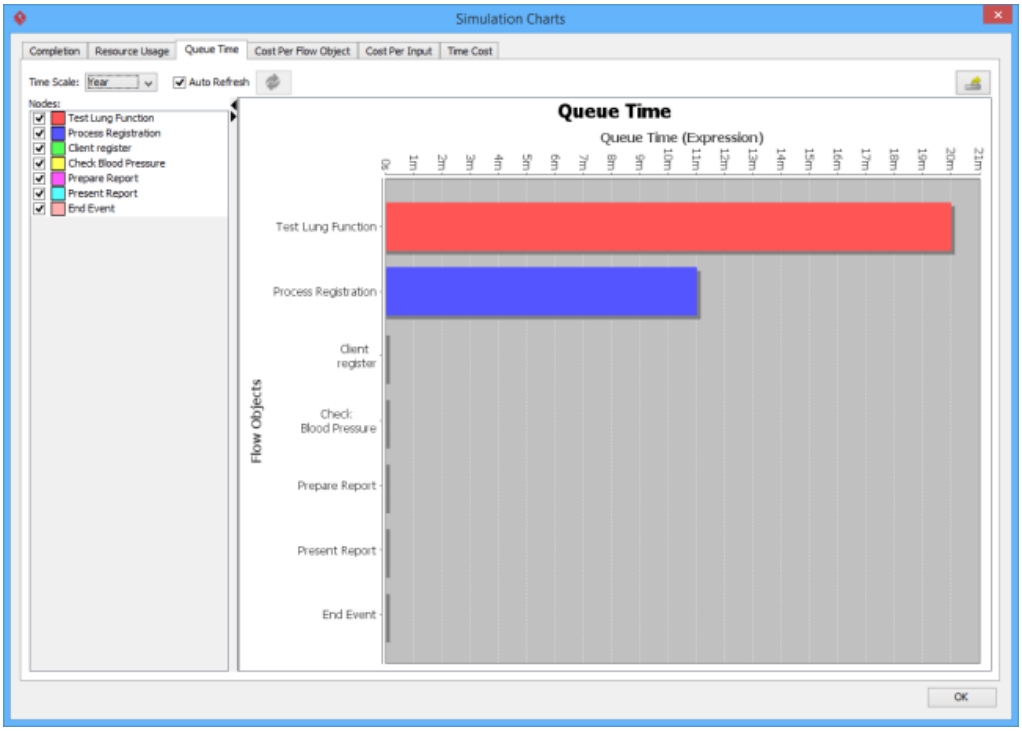
Input
Input is a way of simulating a given business process. It has a name that describe the input and an instance, which is a number that represents the number of time the input will happen at a particular instant. If you have modeled a general order processing system, you can add an input public holiday with instance 100 to represent the case that in public holiday, there will be 100 customers that need to undergo payment. In order to help you to improve your process, you must set input that reflects the reality. If you set 10000 as instance of input *public holiday* which will never happen, you will not obtain useful information to aid in process improvement.

Simulation
Once you have specified the available and required resources, the duration of flow objects and added input(s), you can run simulation. During simulation, diagram will be locked to avoid collision between your edit action and the simulation operation. Executing jobs are represented by a running green gear shape, with a number indicating the number of running job and are attached to the task where the job is being processed. Pending jobs are represented by inverted triangles, with a number indicating the number of pending jobs.



Executing and pending jobs

Performance analysis through charts
During simulation of business process, you can identify the bottleneck(s) by observing the occurrence of pending jobs (i.e. the inverted triangle). This works well in a relatively small process. However, if your business process diagram is large, you may not be able to study the simulacian outcome just by observation because the simulcian can be lengthy and involve several bottlenecks. Furthermore, you may want to know the exact figure of resource consumption for conducting a more accurate resource re-allocation plan. In those cases, you can produce simulacian charts that documents the completion of inputs, resource usage and queue time of flow objects against time.



Queue Time Chart

Related Resources
The following resources may help you to learn more about the topic discussed in this page.

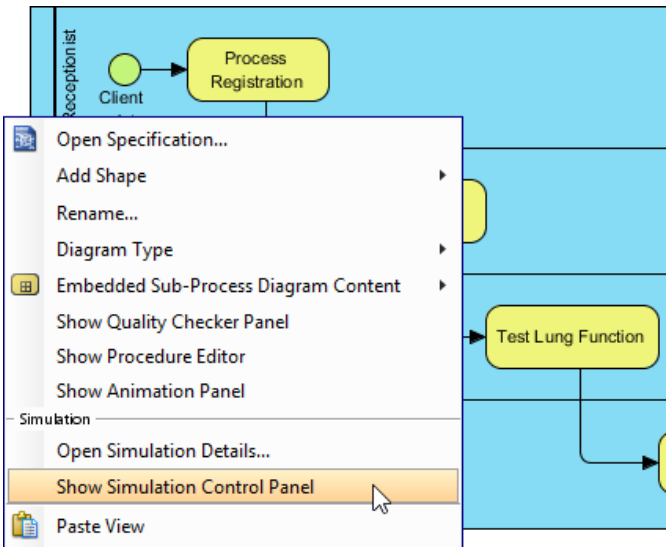
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Simulation control panel

In order to simulate a business process, you need to define simulation details like available and required resources, duration of tasks/sub-processes, instances of pools/lanes and scenarios. All these information can be defined in Simulation Control Pane which is a pane that display at the bottom of diagram, important for adjusting any settings related to simulation. The panel will be updated base on your selection in active diagram. Besides setting simulation details, start/pause of simulation can also be done in the panel.

Opening Simulation Control Panel

To open the Simulation Control Panel, right click on the business process diagram that you want to simulate and select **Show Simulation Control Panel** from the popup menu.

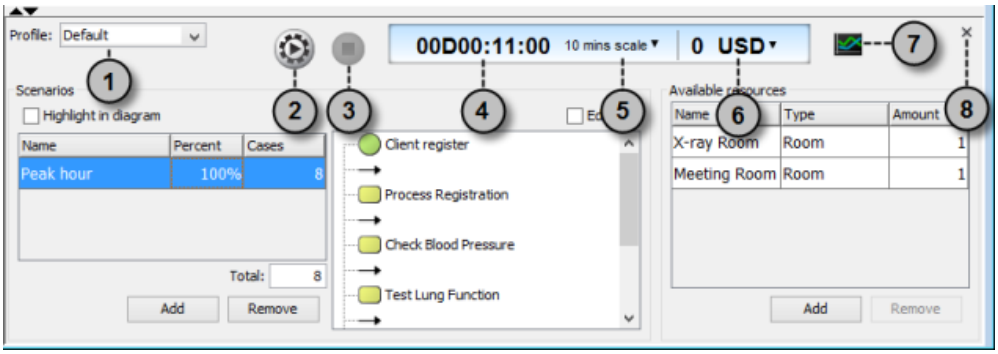


Opening Simulation Control Panel

NOTE: You can open Simulation Control Panel only for diagram that has selected **Simulation** as diagram type. To check/edit diagram type, right click on the background of business process diagram and select **Diagram Type** from the popup menu.

Overview of Simulation Control Panel

Common



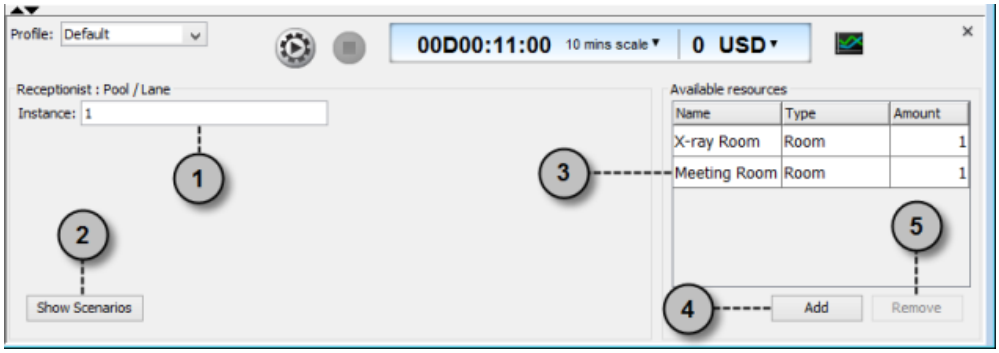
An overview of Simulation Control Panel

No.	Name	Description
1	Profile	To create a new profile for simulation, select **Configure Profile...** from the drop-down menu and name the newly created profile in the **Configure Profile** dialog box.
2	Start / Pause	Click to start simulation when paused or stopped base on the resource, duration and scenario settings or to pause a simulation when it is playing.
3	Stop	Stop a simulating business process.
4	Clock	Displays the time elapsed from the beginning of simulation until the current moment. It is for reflecting the duration of the execution of business process, and is based on the selection of time scale.
5	Time scale	Control the speed of simulation. For example, a selection of *10 mins scale* simulates the business process in speed 10 min per second.
6	Current unit	Click to select the current unit, such as US Dollar, Hong Kong Dollar and Yen, for the consumption during simulating.
7	Simulation charts	Click to display a new window that display the completion, resource usage and queue time charts base on the settings of resource, duration and scenario. You can treat it as a chart form of simulation outcome.

8	Close	Click to close the Simulation Control Panel. You can open it again by right clicking on the business process diagram and selecting **Show Simulation Control Panel** from the popup menu.

Description of Simulation Control Panel

When selected pool/lane

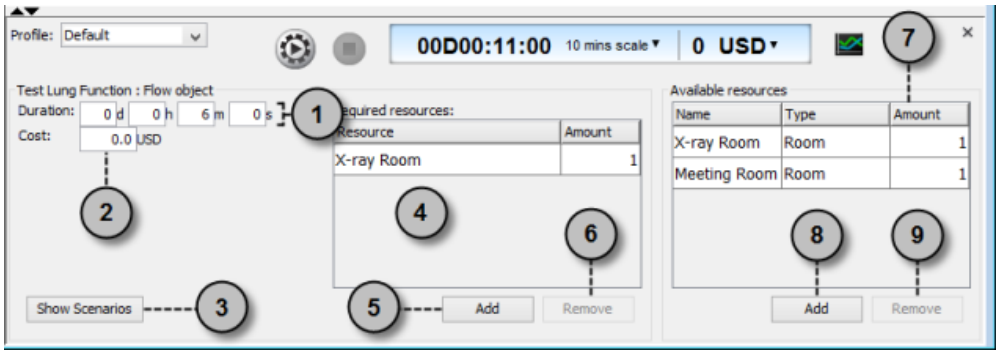


An overview of Simulation Control Panel while selected pool/lane

No.	Name	Description
1	Instance	When you model a business operation in business process diagram, you use pools and lanes to represent participants and sub-participants of the process, such as *Client* and *Receptionist*. No matter how many actual participants are there, you will still use a single pool (or lane) to represent all of them. For example, you will draw a pool *Receptionist* to represent all receptionists instead of drawing 5 pools for representing the fact that there are 5 receptionists. Here the **Instance** field which allow you to set the number of instances of the selected pools or lanes. If there are 5 receptionists, enter 5 for instance of pool/lane *Receptionist*. Provided that there are sufficient resources for performing jobs, the number of instances affects the performance of process - The more the instances, the more the efficient. During process improvement, you can adjust the instance to evaluate the impact of increasing or decreasing the number of staff to handle certain job.
2	Show Scenarios	Scenarios are the expected way of executing a business process during simulation. Click **Show Scenarios** to list the scenarios, and add/remove scenarios in further.
3	Available resources	The table of available resources list the resources needed by the business process. For example, a process of body checking has resources X-Ray room and reception counter. Each resource has a name, a type and its amount.
4	Add available resource	Click to add an available resource by giving its name, selecting/entering its type and setting its amount.
5	Remove available resource	Select an available resource and click this button to remove it.

Description of Simulation Control Panel while selected pool/lane

When selected flow objects



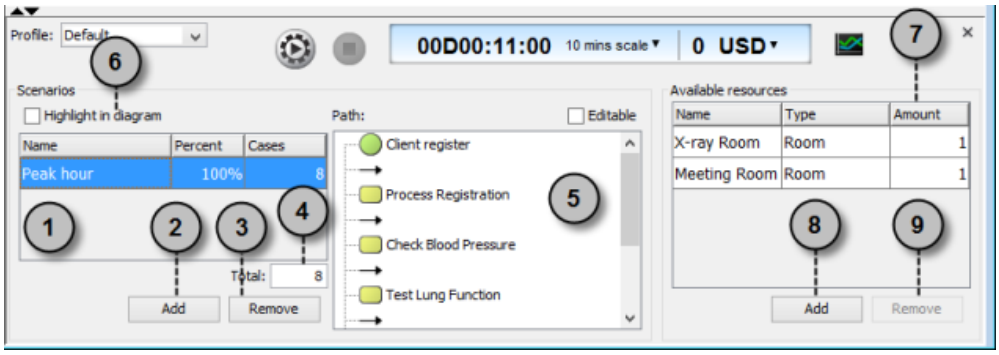
An overview of Simulation Control Panel while selected flow object

No.	Name	Description
1	Duration	The time the selected flow object need to take to process and complete a job. The 4 boxes d, h, m and s mean day, hour, minute and second, respectively. For example, if it takes five minutes to present a document to client in a body check operation, then set the business task *Present Document*'s duration to be 5 m, meaning 5 minutes. The duration setting affects the performance of process - The less time is needed, the more efficient it is. However, do not forget that the less time it takes to complete a task may affects the quality of work. During

		process improvement, you need to keep the balance between efficiency and quality of work and adjust the duration accordingly.
2	Cost	The cost the selected flow object need to spend to process and complete a job.
3	Show scenarios	Scenarios are the expected way of executing a business process during simulation. Click **Show Scenarios** to list the scenarios, and add/remove scenarios in further.
4	Required resources	The resources the participant needed in order to complete a job when processing the selected flow object. The **Resource** column shows the names of resources. The **Amount** column shows the number of resource needed to use in completing the task per participant. For example, to present document to client, you need one resource *Meeting Room*, and one resource *Projector*.
5	Add required resource	Click to add the resource the participant needed in order to complete a job when processing the selected flow object. Make sure you have available resources defined in order to select a required resource. Also note that you cannot set an amount that exceed the amount set in available resource. For example, if you have 6 available projectors (resources), the maximum number of projector a flow object can require is from 0 to 6.
6	Remove required resource	Select a required resource and click this button to remove it.
7	Available resources	The table of available resources lists out the resources needed by the business process. For example, a process of body checking has resources X-Ray room and reception counter. Each resource has a name, a type and its amount.
8	Add available resource	Click to add an available resource by giving its name, selecting/entering its type and setting its amount.
9	Remove available resource	Select an available resource and click this button to remove it.

Description of Simulation Control Panel while selected flow objects

When selected diagram background / showing scenarios

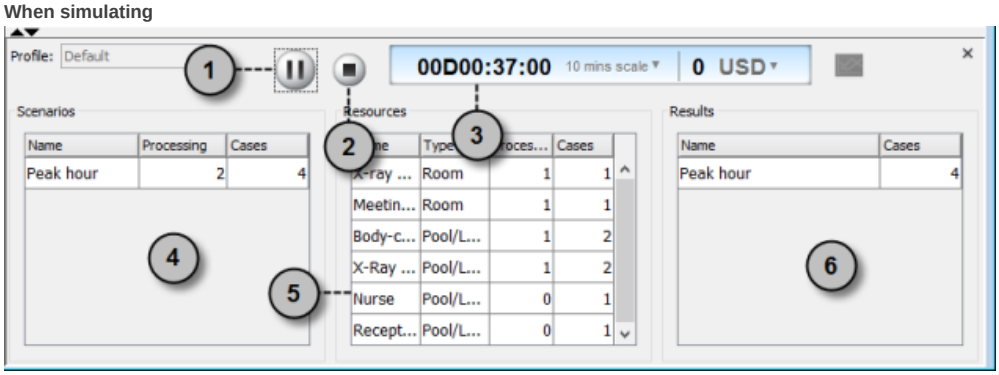


An overview of Simulation Control Panel while selected diagram background or showing scenarios

No.	Name	Description
1	Scenarios	Scenarios are the definitions of how to execute a business process during simulation. A scenario consists of a selection of possible execution path formed by flow objects in diagram with the number of instances, which represents the number of time the path will be executed at a specific instant. For example, if you want to simulate the case in which 10 clients need to perform body checking in a business process of body checking, to see whether the process can handle this situation well, you will add an scenario *Performing body check*, with 10 as number of instances.
2	Add scenarios	Click this button to add an scenario with name and number of instances.
3	Remove scenario	Select a scenario and click this button to remove it.
4	Total cases	The total number of cases of all scenarios.
5	Path	The flow objects involved in a scenario. If there is a gateway in your diagram, you need to make a decision to the outgoing path of the gateway object.
6	Highlight in diagram	Check this button to make the diagram highlight the path involved in the scenario selected in **Scenarios** table.
7	Available resources	The table of available resources list the resources needed by the business process. For example, a process of body checking has resources X-Ray room and reception counter. Each resource has a name, a type and its amount.
8	Add available resource	Click to add an available resource by giving its name, selecting/entering its type and setting its amount.
9	Remove available resource	Select an available resource and click this button to remove it.

Description of Simulation Control Panel while selected diagram background or showing scenarios

When simulating



An overview of Simulation Control Panel while simulating

No.	Name	Description
1	Pause	To temporarily pause a simulation.
2	Stop	To stop simulating the business process.
3	Clock	Display the time elapsed from the beginning of simulation until the current moment. It is for reflecting the duration of the execution of business process and it is based on the selection of time scale.
4	Scenarios	A list of scenarios with their completeness throughout the simulation. The **Processing** column shows the jobs under processing by the simulating process. The **Instances** column shows the amount of non-completed jobs. It should keep decreasing and will become 0 at the end of simulation.
5	Resources	A list of resources with the status of consumption throughout the simulation. The **Processing** column shows the amount of resources be consumed by the simulating process. The **Instances** column shows the total amount of resources. You can observe this table to study the current resource allocation.
6	Results	A list of completed scenarios. The **Instances** column shows the amount of completed scenarios.

Description of Simulation Control Panel while simulating

Related Resources

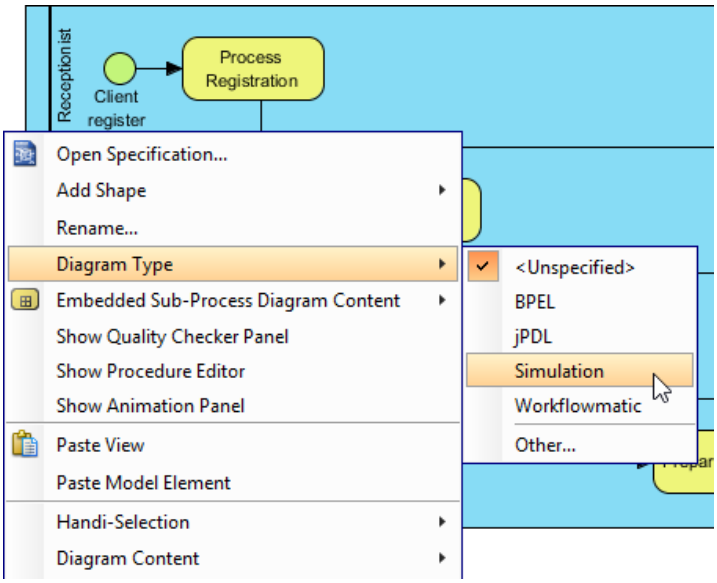
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Simulating business process

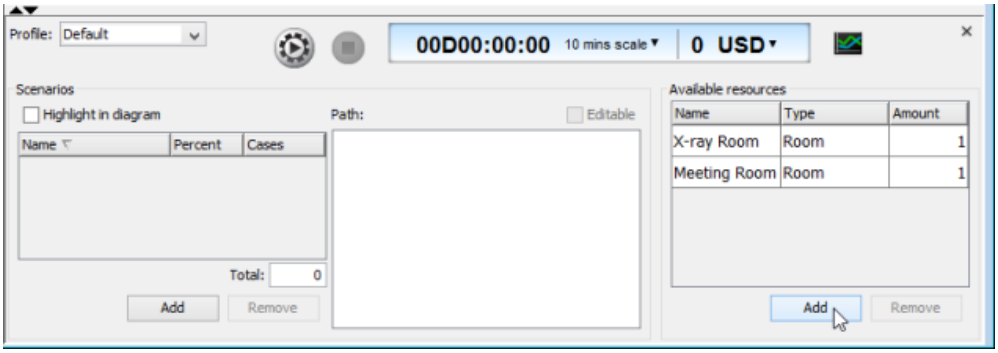
In order to simulate a business process, you must provide performance-related information to the business process diagram, such as resource consumption and duration of flow objects, so that the simulation tool can analyze the information and conduct a simulation. Below are what you have to do to start simulation.

1. Right click on the background of the business process diagram that you want simulate and select **Diagram Type > Simulation** from the popup menu.



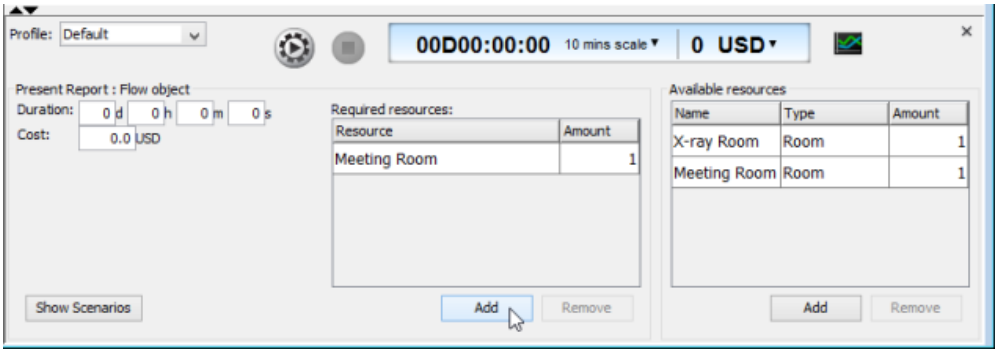
Set diagram's type to Simulation

2. In the **Simulation Control Panel**, click **Add** under **Available resources** to define the resources that can be used by the flow objects in the business process diagram in order to complete the tasks. If you want to know more about resources, please refer to the chapter *What is Process Simulation?*.



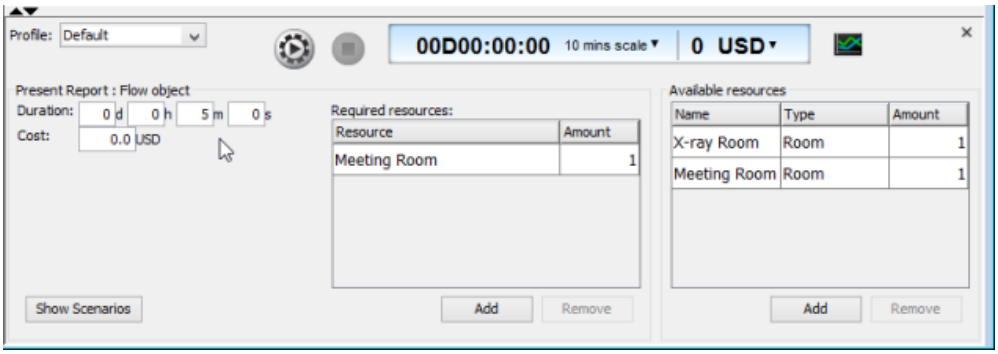
To add available resources

3. For each of the flow objects, select it in diagram and click **Add** under **Required Resources** in **Simulation Control Panel** to add the resource(s) the participant needed to complete the job when reaching the selected flow object.



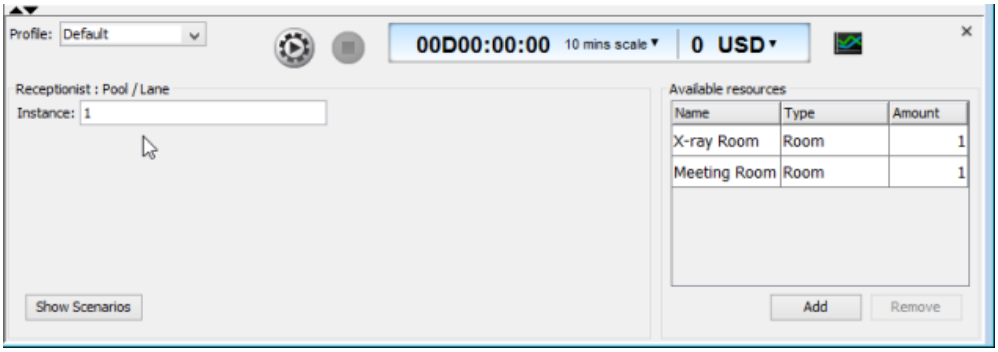
To add required resources

4. For each of the flow objects, select it in diagram and set in **Simulation Control Panel** its duration, which is the time it takes to get completed. If you want to know more about duration, please refer to the chapter *What is Process Simulation?*.



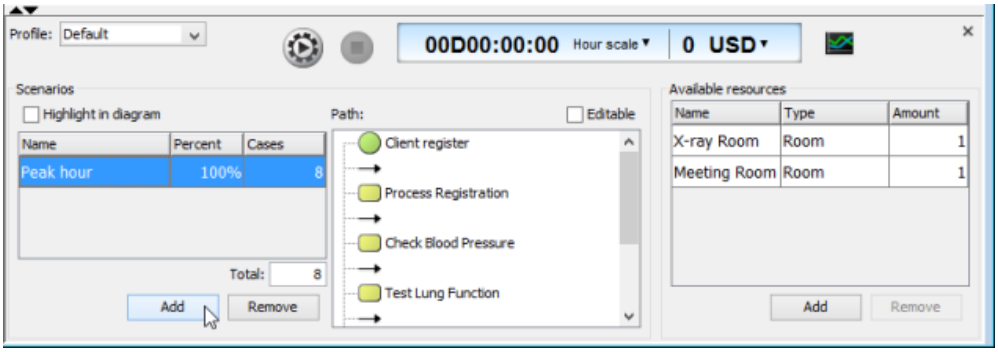
To set the duration of flow object

5. For each lane and pool (without lane), select it in diagram and set its instance in **Simulation Control Panel** which represents the number of participants that take part in the process.



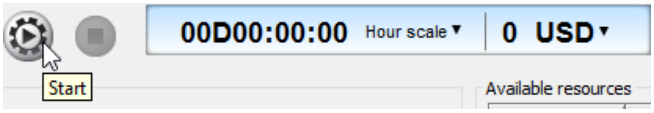
To set instance of pools or lanes

6. Click on the background of diagram or click **Show Scenarios** in **Simulation Control Panel**.
7. Click **Add** under Scenarios in **Simulation Control Panel** to add the paths to be executed during simulation. For each scenario, set the name that describe the path and the number of instances for the number of copies of the path to execute.



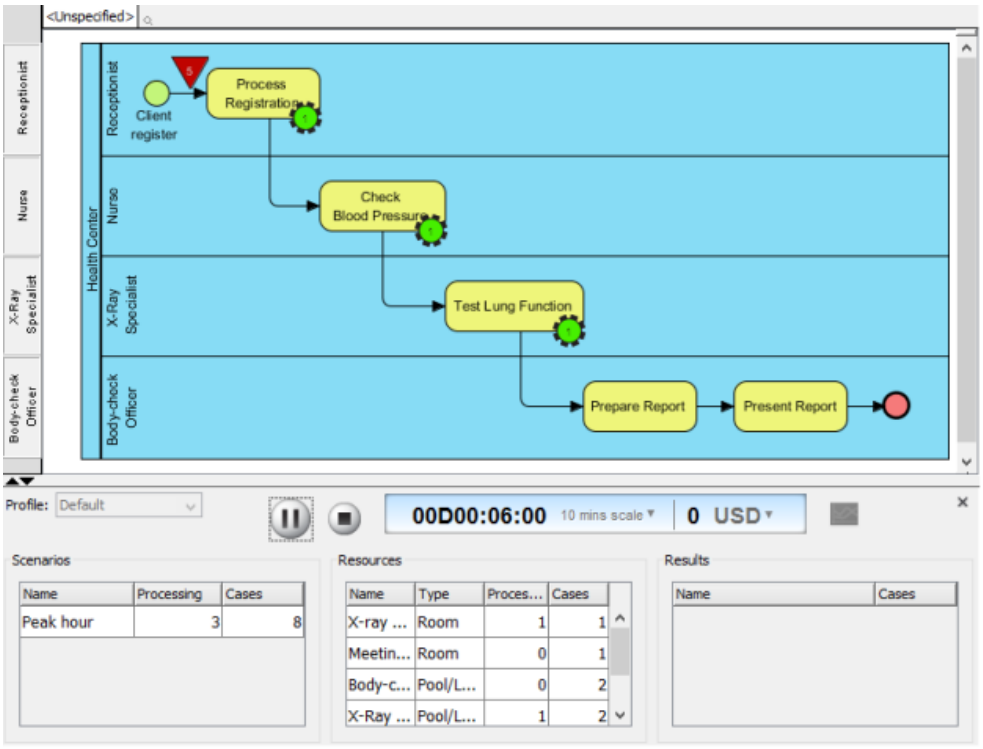
To add scenario

8. Click Start in **Simulation Control Panel** to start simulation.



Start simulation

Now, you can watch the simulation in diagram, to see the executions of scenarios and identify bottlenecks.



Simulation is running

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Simulation charts

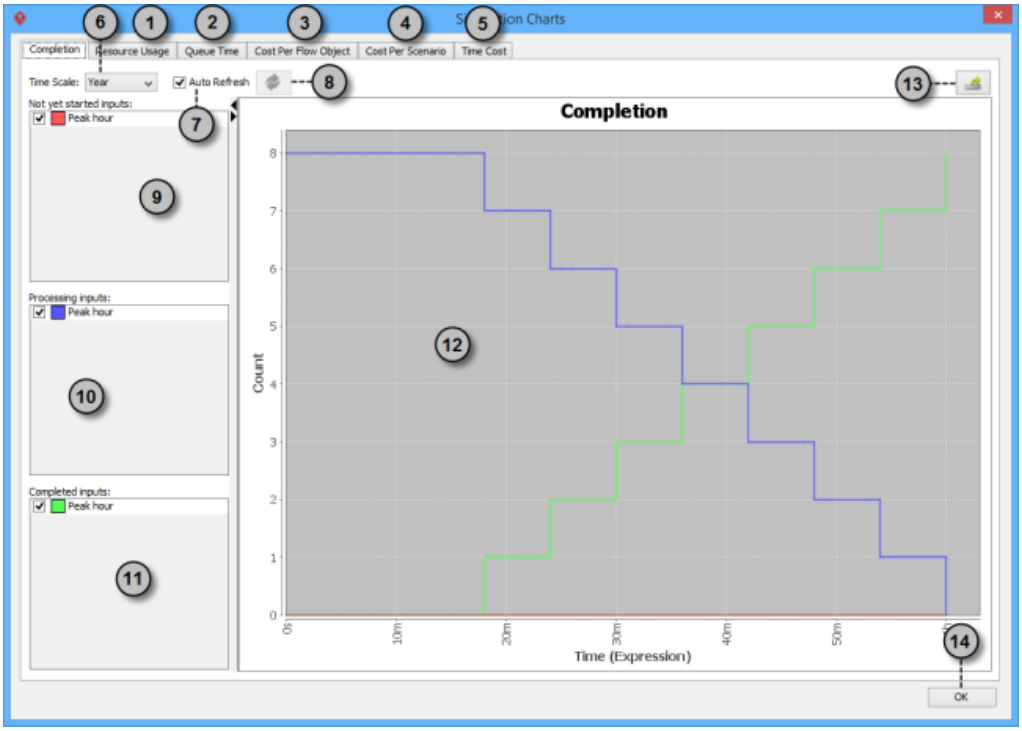
Sometimes, just by watching the Simulation outcome is not enough to find out the bottleneck, especially when the diagram is large and have many, many bottlenecks. In such cases, you can produce charts for Simulation outcome, which helps quantify resource consumption and queuing time for each flow object.

To read the charts, click **Simulation Charts** in Simulation Control Panel.

Open Simulation Charts window

Completion chart

The completion chart primarily shows the status of scenarios completion against time.



An overview of completion chart

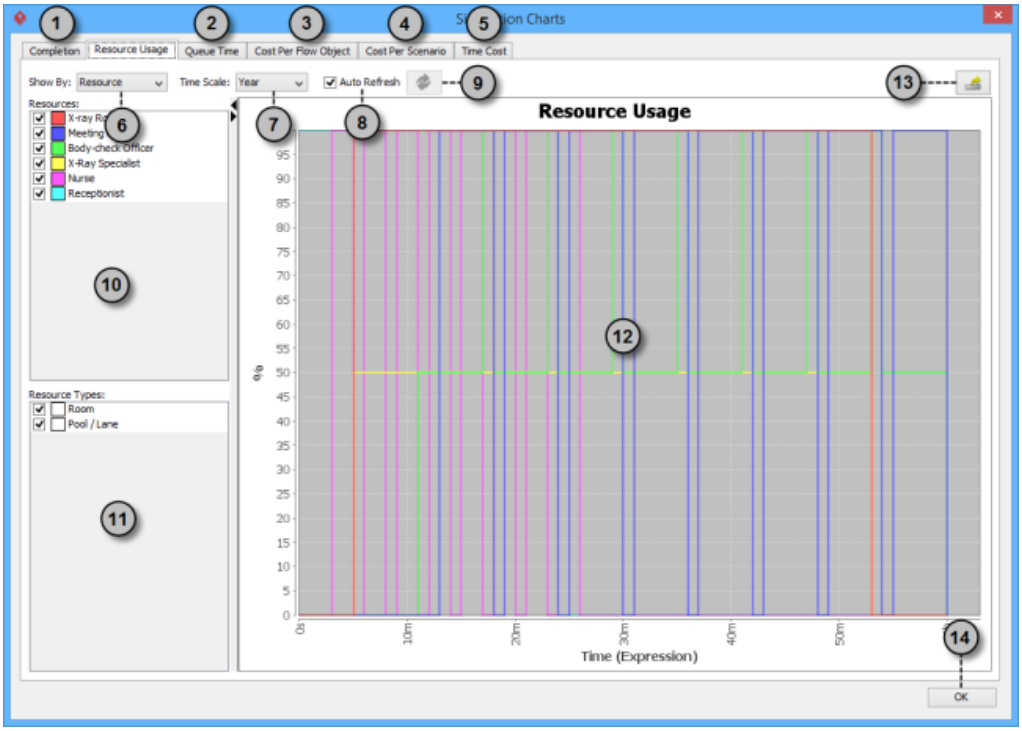
No.	Name	Description
1	Resource usage	Click to show resource usage chart.
2	Queue time	Click to show queue time chart.
3	Cost Per Flow Object	Click to show cost per flow object chart.
4	Cost Per Scenario	Click to show cost per scenario chart.
5	Time Cost	Click to show time cost chart.
6	Time scale	Control the length of chart by selecting a time scale.
7	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of scenarios. Uncheck to update manually by clicking **Refresh**.
8	Refresh	Click to update the chart body against the chart settings such as time scale and selection of scenarios. This button is available only when **Auto Refresh** is unchecked.
9	Not yet started scenarios	Check or uncheck the scenarios to show or hide in chart the change of the amount of not-yet-started scenarios throughout simulation.
10	Processing scenarios	Check or uncheck the scenarios to show or hide in chart the change of the amount of processing scenarios throughout simulation.
11	Completed scenarios	Check or uncheck the scenarios to show or hide in chart the change of the amount of completed scenarios throughout simulation.
12	Chart body	The chart body.

13	Export	Click to export the opening chart to Microsoft Excel or image file.
14	OK	Click to close the Simulation charts window and go back to the diagram.

Description of completion chart

Resource usage chart

The resource usage chart shows the percentage of resource consumption throughout simulation. If a resource has its peak reaching 100%, it means that the allocation of that resource is in optimum state. If the peak is below 100%, it means that some of the resources are not used throughout the simulation, which usually signifies that they are wasted, and you should consider to adjust the amount of available resource to optimize the resource consumption.



An overview of resource usage chart

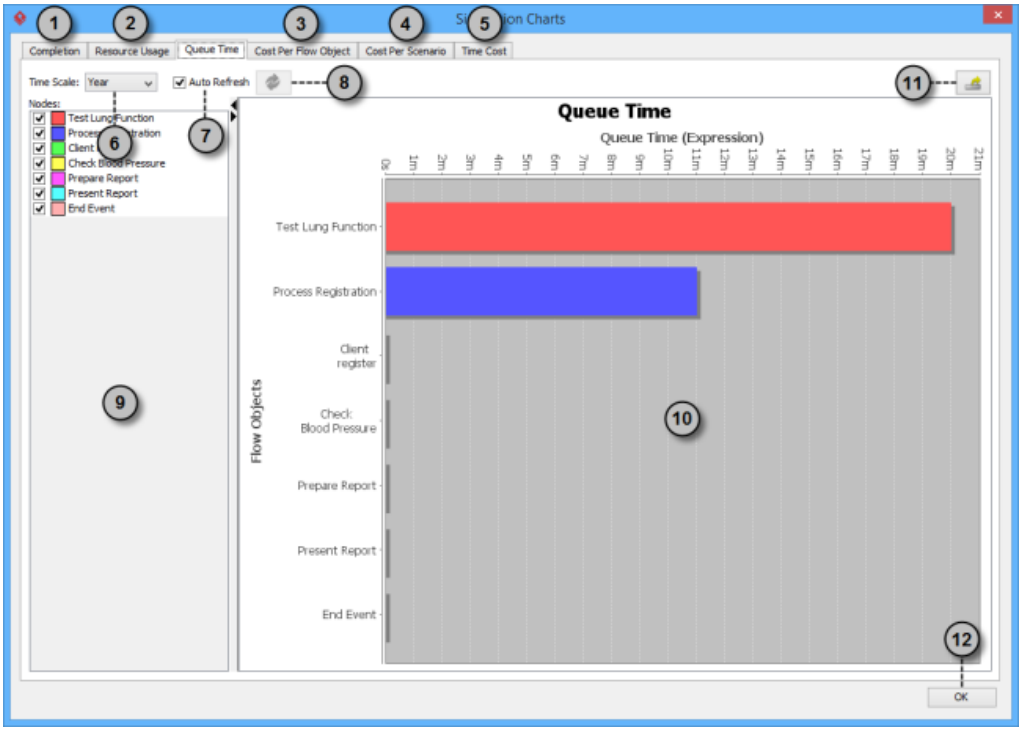
No.	Name	Description
1	Complete	Click to show completion chart.
2	Queue time	Click to show queue time chart.
3	Cost Per Flow Object	Click to show cost per flow object chart.
4	Cost Per Scenario	Click to show cost per scenario chart.
5	Time Cost	Click to show time cost chart.
6	Show by	Select what to present in the chart. Resource - Cause the chart to presents the resource consumption of each available resource. Resource Type - Cause the chart to presents the resource consumption of resource types.
7	Time scale	Control the length of chart by selecting a time scale.
8	Auto refresh	Check to let the chart body auto update against chart settings such as whether to show by resource or resource type, time scale and selection of resources and resources type. Uncheck to update manually by clicking **Refresh**.
9	Refresh	Click to update the chart body against the chart settings such as whether to show by resource or resource type, time scale and selection of resources and resources type. This button is available only when **Auto Refresh** is unchecked.
10	Resources	Check or uncheck the resources to show or hide their usage in chart. This pane is active only when **Resource** is selected in the drop down menu **Show By**.
11	Resource types	Check or uncheck the resource types to show or hide their usage in chart. This pane is active only when **Resource Type** is selected in the drop down menu **Show By**.
12	Chart body	The chart body.
13	Export	Click to export the opening chart to Microsoft Excel or image file.

14	OK	Click to close the Simulation charts window and go back to the diagram.

Description of resource usage chart

Queue time chart

The queue time chart shows the time the flow objects spent on waiting, which corresponds to the time an inverted triangle appear during simulation. You may study whether the queue time of certain flow object is reasonable or not and think of the improvement.



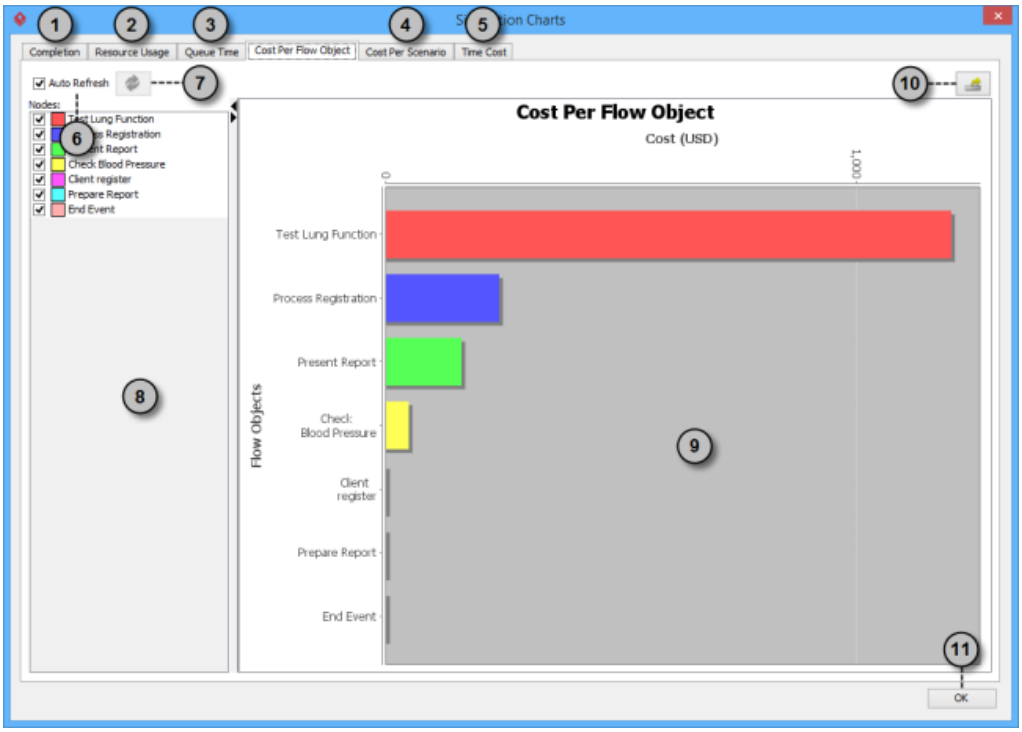
An overview of queue time chart

No.	Name	Description
1	Completion	Click to show completion chart.
2	Resource usage	Click to show resource usage chart.
3	Cost Per Flow Object	Click to show cost per flow object chart.
4	Cost Per Scenario	Click to show cost per scenario chart.
5	Time Cost	Click to show time cost chart.
6	Time scale	Control the length of chart by selecting a time scale.
7	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of nodes. Uncheck to update manually by clicking **Refresh**.
8	Refresh	Click to update the chart body against the chart settings such as time scale and selection of nodes. This button is available only when **Auto Refresh** is unchecked.
9	Nodes	Check or uncheck flow objects nodes to show or hide their queue time in chart.
10	Chart body	The chart body.
11	Export	Click to export the opening chart to Microsoft Excel or image file.
12	OK	Click to close the simulacian charts window and go back to the diagram.

Description of queue time chart

Cost per flow object chart

The cost per flow object chart shows the cost each flow object consumed throughout simulation.



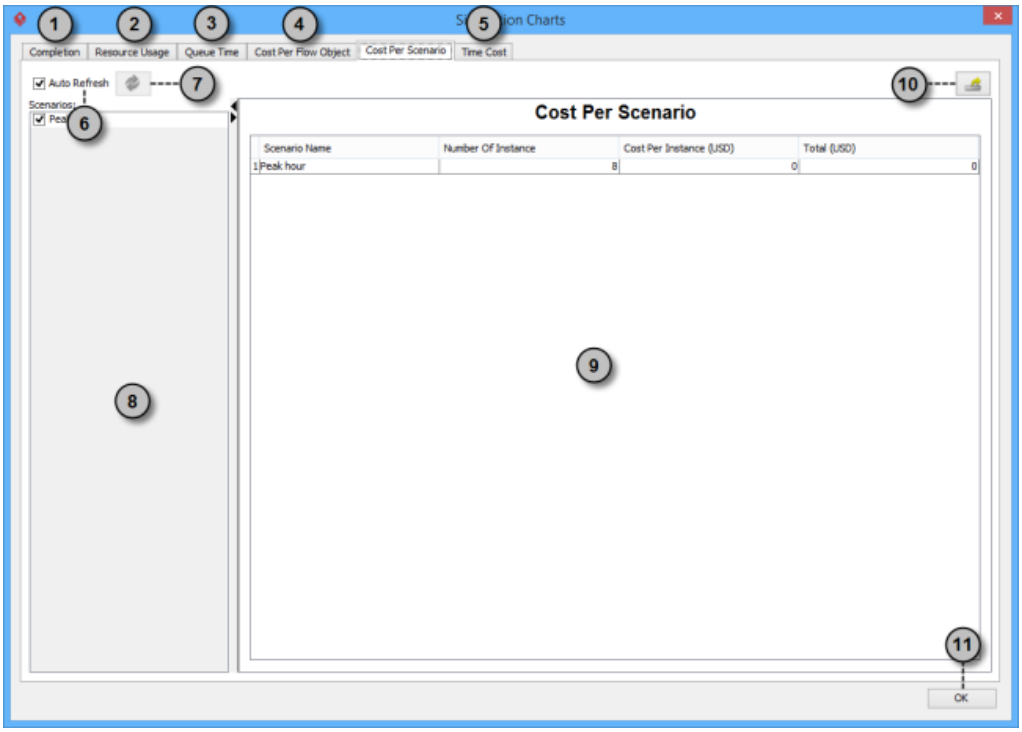
An overview of cost per flow object chart

No.	Name	Description
1	Completion	Click to show completion chart.
2	Resource usage	Click to show resource usage chart.
3	Queue Time	Click to show queen time chart.
4	Cost Per Scenario	Click to show cost per scenario chart.
5	Time Cost	Click to show time cost chart.
6	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of scenarios. Uncheck to update manually by clicking **Refresh**.
7	Refresh	Click to update the chart body against the chart settings such as time scale and selection of nodes. This button is available only when **Auto Refresh** is unchecked.
8	Nodes	Check or uncheck flow objects nodes to show or hide their queue time in chart.
9	Chart body	The chart body.
10	Export	Click to export the opening chart to Microsoft Excel or image file.
11	OK	Click to close the Simulation charts window and go back to the diagram.

Description of queue time chart

Cost per scenario chart

The cost per scenario chart shows the cost each scenario consumed throughout simulation.



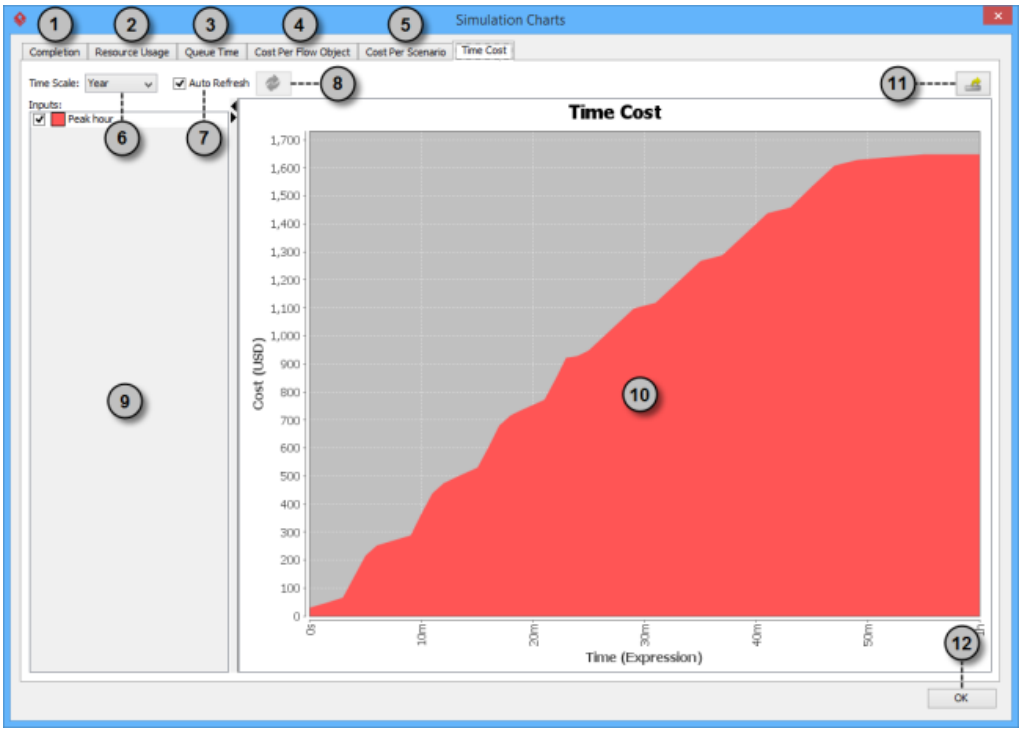
An overview of cost per scenario chart

No.	Name	Description
1	Completion	Click to show completion chart.
2	Resource Usage	Click to show resource usage chart.
3	Queue Time	Click to show queen time chart.
4	Cost Per Flow Object	Click to show cost per flow object chart.
5	Time Cost	Click to show time cost chart.
6	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of scenarios. Uncheck to update manually by clicking **Refresh**.
7	Refresh	Click to update the chart body against the chart settings such as time scale and selection of scenarios. This button is available only when **Auto Refresh** is unchecked.
8	Nodes	Check or uncheck flow objects nodes to show or hide their cost per scenario in chart.
9	Chart body	The chart body.
10	Export	Click to export the opening chart to Microsoft Excel or image file.
11	OK	Click to close the Simulation charts window and go back to the diagram.

Description of cost per flow object chart

Time cost chart

The time cost chart shows the cost consumed against time spent throughout simulation.



An overview of time cost chart

No.	Name	Description
1	Completion	Click to show completion chart.
2	Resource Usage	Click to show resource usage chart.
3	Queue Time	Click to show queen time chart.
4	Cost Per Flow Object	Click to show cost per flow object chart.
5	Cost Per Scenario	Click to show cost per scenario chart.
6	Time Scale	Control the length of chart by selecting a time scale.
7	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of scenarios. Uncheck to update manually by clicking **Refresh**.
8	Refresh	Click to update the chart body against the chart settings such as time scale and selection of scenarios. This button is available only when **Auto Refresh** is unchecked.
9	Nodes	Check or uncheck flow objects nodes to show or hide their cost per scenario in chart.
10	Chart body	The chart body.
11	Export	Click to export the opening chart to Microsoft Excel or image file.
12	OK	Click to close the Simulation charts window and go back to the diagram.

Description of cost per flow object chart

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Eclipse Integration

Overview and Installation of Eclipse Integration

Know how Visual Paradigm can work with Eclipse through Eclipse integration. Learn how to install the integration from Visual Paradigm.

Creating a UML Project in Eclipse

Learn how to create a UML project from Java project in Eclipse.

Opening a UML Project in Eclipse

Learn how to open a UML project created from a Java project.

Reverse Engineering in Eclipse

Learn how to reverse engineer class model from Java source code in Eclipse.

Code Generation from UML Model in Eclipse

Learn how to produce source files from class model in Visual Paradigm.

Selecting UML Class from Source File in Eclipse

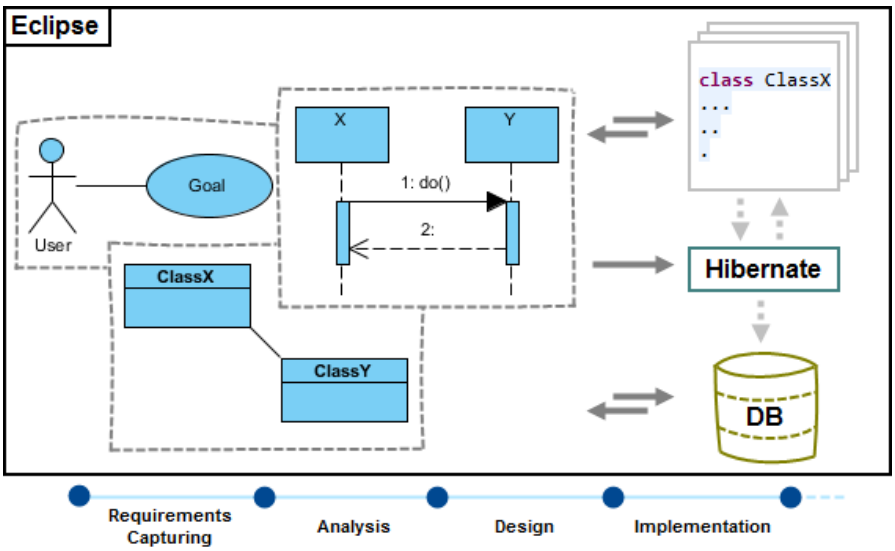
Learn how to select UML class model from a given source file.

Selecting Source File in Eclipse from UML Class

Learn how to select source file from a given UML class model.

Overview and installation of Eclipse integration

Visual Paradigm enables you to integrate the visual modeling environment with Eclipse, providing full software development life cycle support. By designing your software system in Visual Paradigm, you can generate programming source code from class diagram to an Eclipse project. Also, you can reverse engineer your source code into class models in Visual Paradigm.

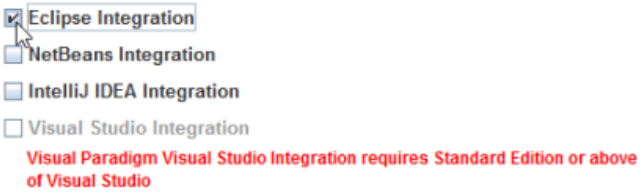


Eclipse integration overview

Installation

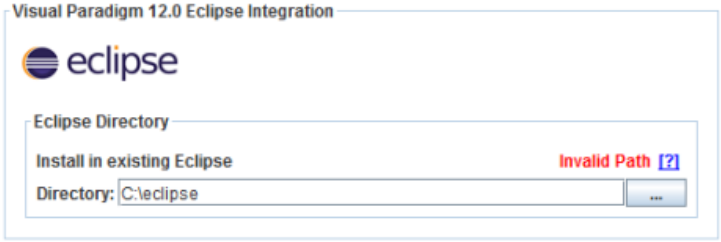
First of all, please make sure you have Eclipse 3.5 or above available. To install Eclipse Integration from Visual Paradigm:

1. In Visual Paradigm, select **Windows > Integration > IDE Integration...** from the toolbar.
2. Select Eclipse. You can run Visual Paradigm in multiple IDEs. In other words, if you need you can select multiple IDEs here. Click **Next**.



Select Eclipse Integration

3. Specify the folder path of Eclipse. Click **Next** to start copying files to your IDE.



Path of Eclipse

> **NOTE:** Eclipse integration can only be installed on one Eclipse directory only.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

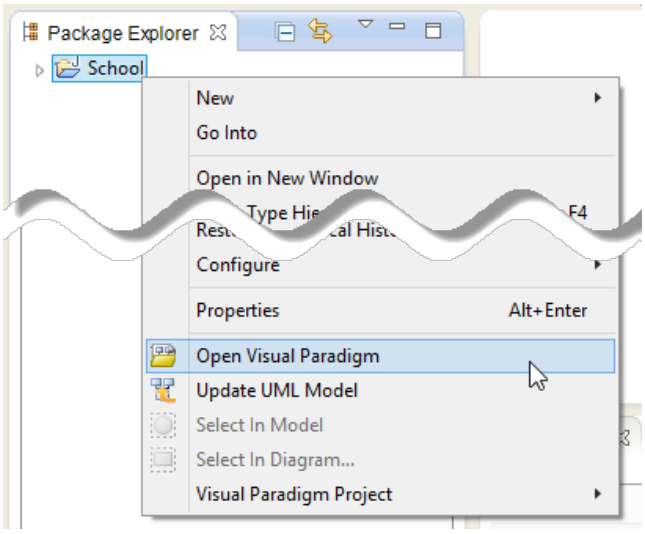
- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating a UML project in Eclipse

You can create UML project for any of your Java project in Eclipse. Note that one Java project can associate with at most one UML project and you cannot create UML project without associating it with any Java project. Once you have created a UML project for a Java project, you cannot remove it or de-associate it.

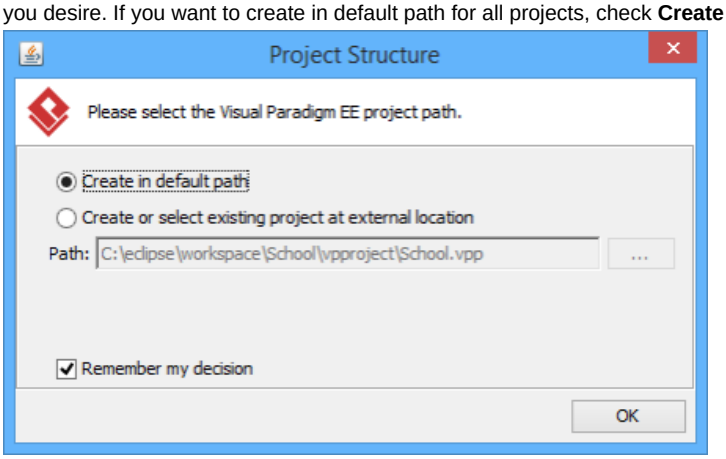
Creating a New UML Project

1. In Eclipse, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Open Visual Paradigm** from the popup menu.



Open Visual Paradigm from Java project

3. Select from the **Project Structure** window the location of the Visual Paradigm project is to be saved. The Visual Paradigm project with .vpp extension is the UML project file that is going to be associated with the selected Eclipse project file. Select **Create in default path** will save the UML project to **%Eclipse_Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.

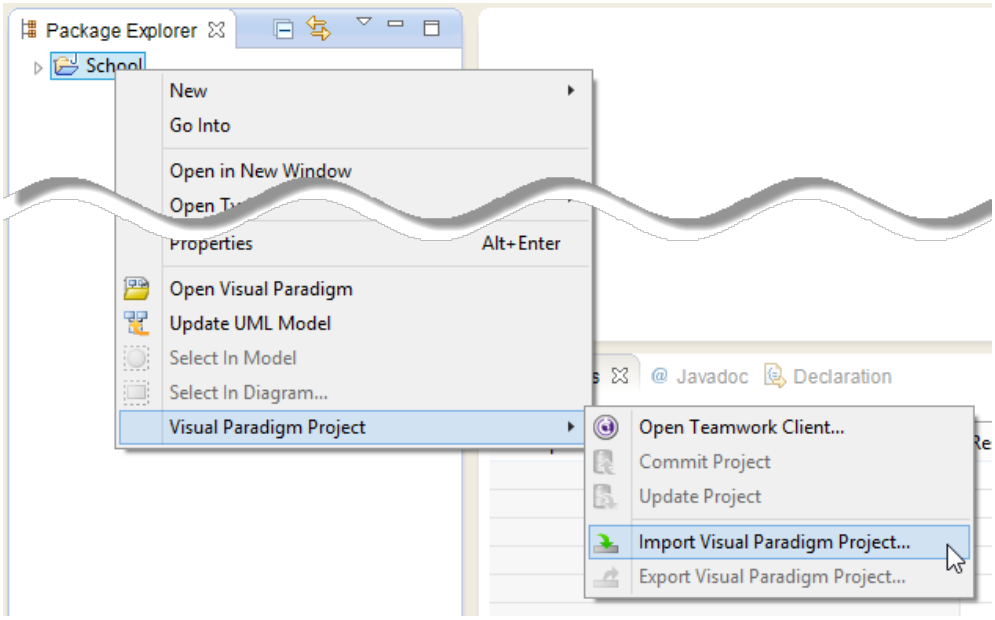


Create a new UML project

4. Click **OK**.

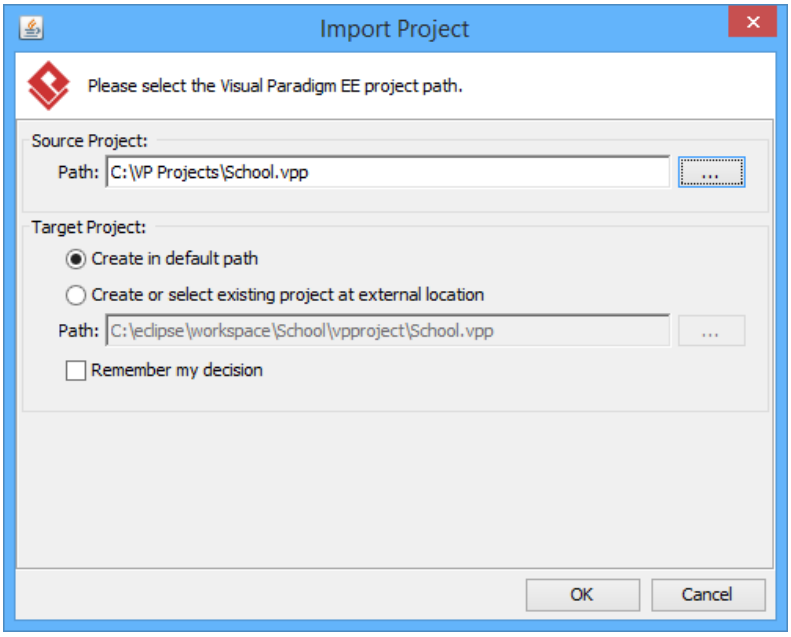
Creating a UML Project by Importing an Existing .vpp Project File

1. In Eclipse, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Visual Paradigm Project > Import Visual Paradigm Project...** from the popup menu.



Import Visual Paradigm project

3. Specify the path of source .vpp project as well as the location of the imported project file is to be saved. Select **Create in default path** will save the UML project to **%Eclipse_Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.



Import an existing .vpp project file

4. Click **OK**.

Related Resources

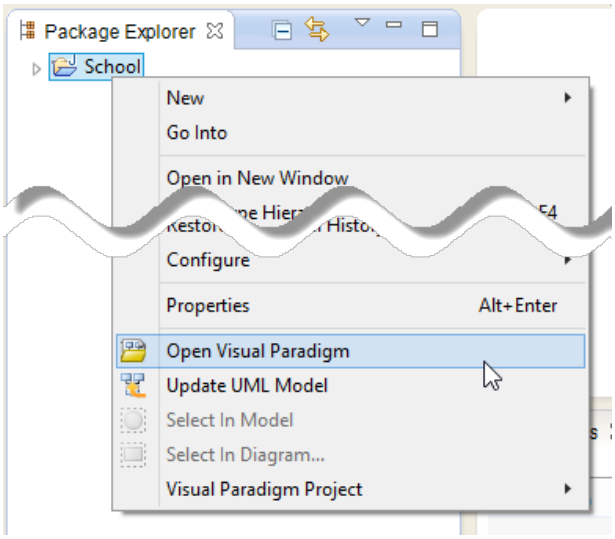
The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening a UML project in Eclipse

Opening a UML Project

1. In Eclipse, select the Java project where you want to open its UML project.
2. Right click on the project and select **Open Visual Paradigm** from the popup menu.



Open Visual Paradigm from Java project

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

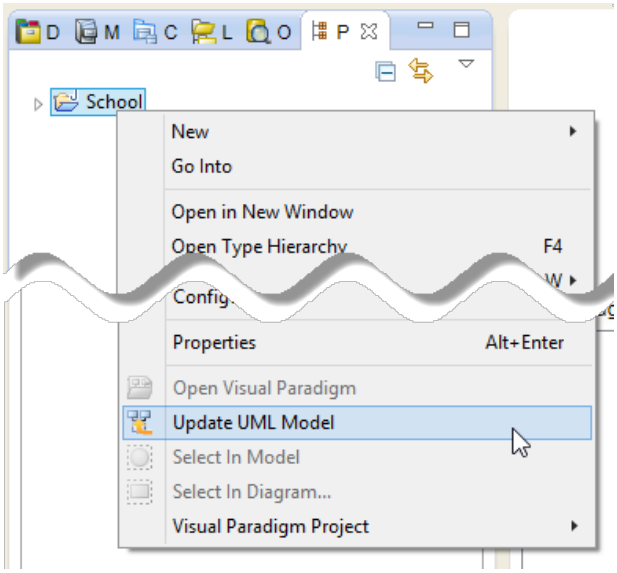
- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse engineering in Eclipse

Reverse engineering is the process to reverse engineer UML model from Java source. With reverse engineering, you can visualize your program or system with class diagram. Before reverse engineering, you must open the UML project from the Java project.

Project Based Reverse Engineering

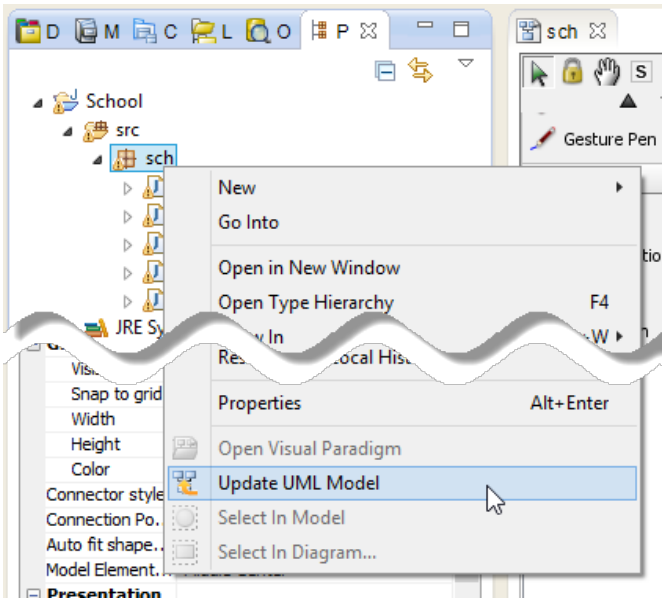
You can produce and update UML models from all source files in a Java project. Models of the selected project, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from an Eclipse project, right-click on the project node in Eclipse and select **Update UML Model** from the popup menu.



Update the whole UML model from a Java project

Package Based Reverse Engineering

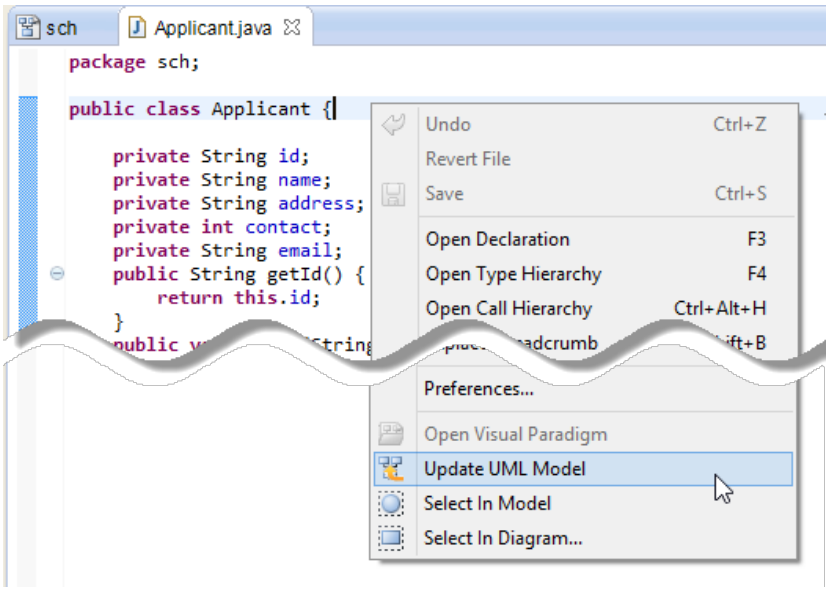
You can produce and update UML models from source files under a package. Models of the selected package, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from a package in a Java project, right-click on the package in any tree and select **Update UML Model** from the popup menu.



Update UML package and its containing classes from a package folder

Class Based Reverse Engineering

You can produce and update UML models from classes in Eclipse. Models of the selected class and child classes (inner class) will be created (if the models are not already exists) or updated. To reverse engineer code from a class in a Java project, right-click on the class file in any tree or in code editor and select **Update UML Model** from the popup menu.



Update UML model from source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

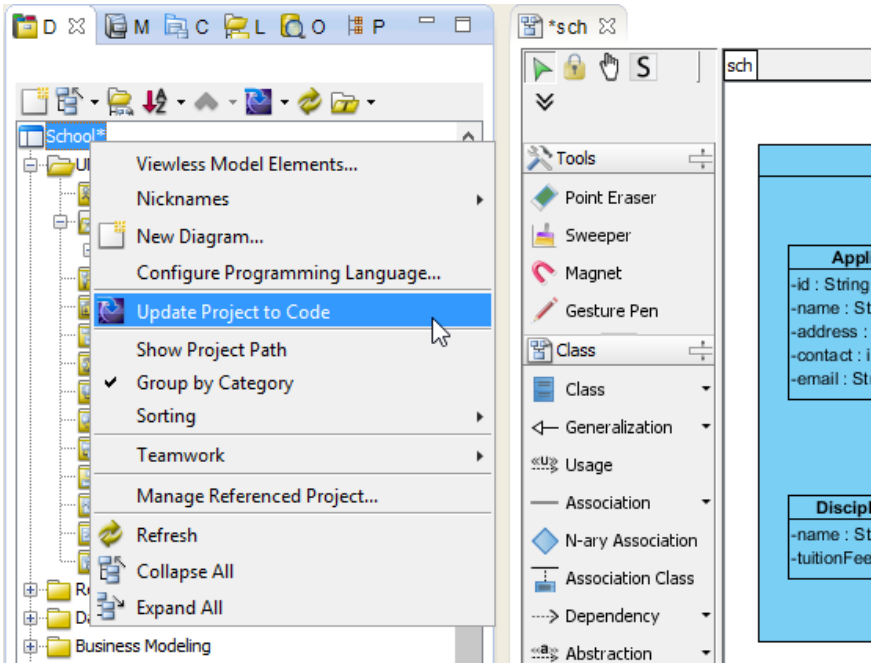
Code generation from UML model in Eclipse

Code generation creates and updates source files in a Java project from UML models. You can select to update the whole project, package(s) and class(es) from Visual Paradigm to Eclipse. Before updating source files, you must open the UML project from the Java project.

Project Based Code Generation

You can generate and update source files from the whole UML project. Packages and classes will be created (if not already exists) or updated. To generate/update source files from UML project, perform any of the steps below:

- Click in Eclipse toolbar.
- Click at the top of **Diagram Navigator**.
- Right click on the root node of **Diagram Navigator** and select **Update Project to Code** from the popup menu.

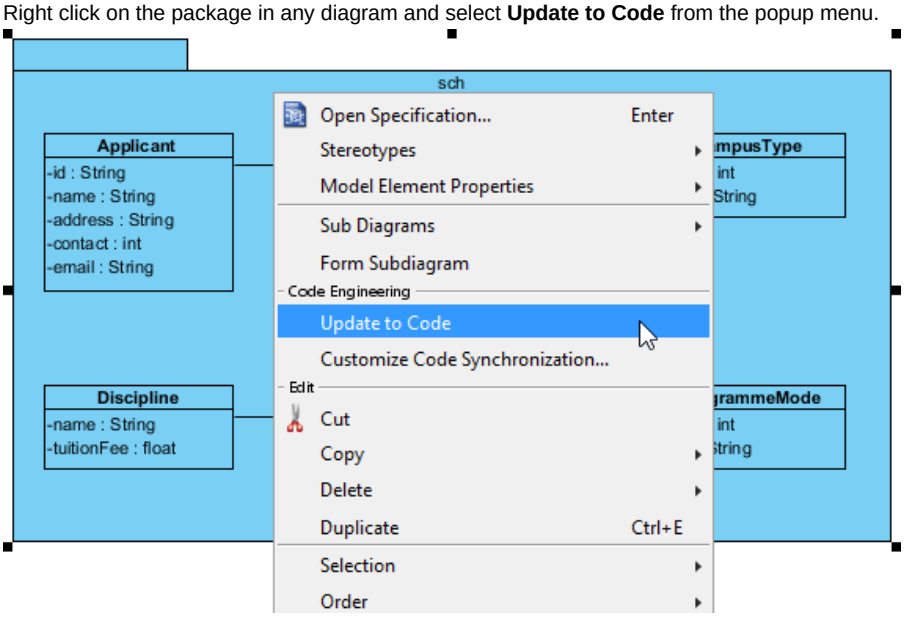


Update the whole Java project from UML project

Package Based Code Generation

You can generate and update package and its containing source file(s) from a UML package. Package and classes will be created (if not already exists) or updated. To generate/update source files from UML package, perform any of the steps below:

- Right click on the package in any diagram and select **Update to Code** from the popup menu.



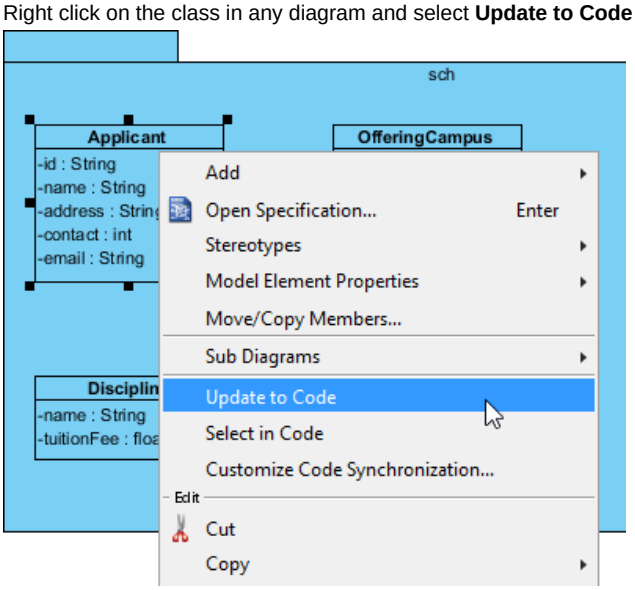
Update source files from UML package

- Right click on the package under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Class Based Code Generation

You can generate and update source file from a UML class. Class will be created (if not already exists) or updated. To generate/update source file from UML class, perform any of the steps below:

- Right click on the class in any diagram and select **Update to Code** from the popup menu.



Update source file from UML class

- Right click on the class under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

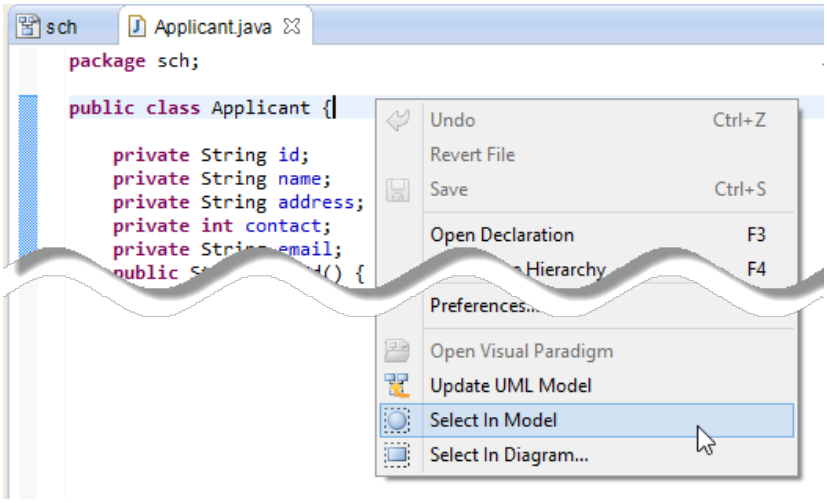
- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting UML class from source file in Eclipse

Once a UML class is associated with a Java source by code reversal/generation, you can select from source file the corresponding UML class in Visual Paradigm.

Selecting UML Class in Model Explorer

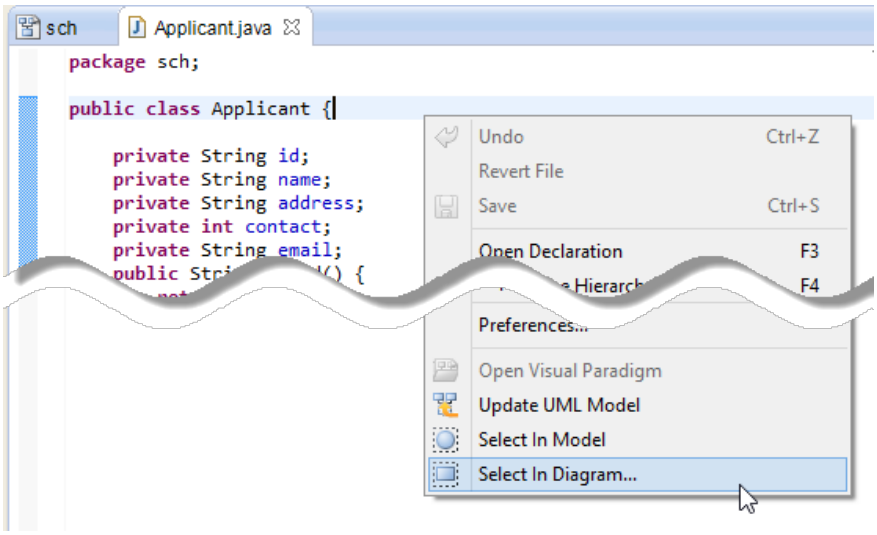
To select the UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Model** from the popup menu.



Open the UML class from a source file

Selecting UML Class in Diagram

To select the view of UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Diagram...** from the popup menu. If the class has been visualized with multiple views, you will be prompted to select a view to open.



Open the view of UML class from a source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

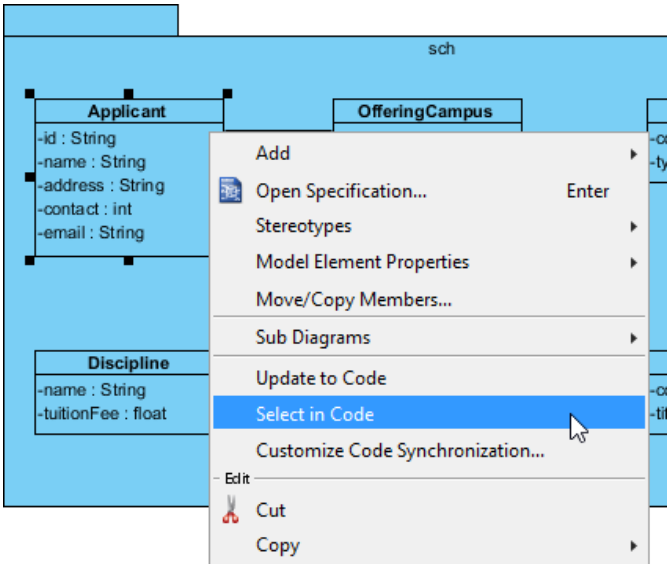
- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting source file in Eclipse from UML class

Once a UML class is associated with a Java source by code reversal/generation, you can select from UML class the corresponding Java source file in Eclipse.

Selecting Java Source from UML Class

To open a Java source file from a UML class, right-click on the UML class in **Diagram Navigator/Model Explorer/Class Repository** or in diagram and select **Select in Code** from the popup menu. This opens the corresponding source file in code editor.



Select source file from UML class

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Begin UML modeling in Eclipse
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Visual Studio Integration

Overview and Installation of Visual Studio Integration

Know how Visual Paradigm can work with Visual Studio through Visual Studio integration. Learn how to install the integration from Visual Paradigm.

Creating a UML Project in Visual Studio

Learn how to create a UML project from project in Visual Studio.

Opening a UML Project in Visual Studio

Learn how to open a UML project created from a Visual Studio project.

Reverse Engineering in Visual Studio

Learn how to reverse engineer class model from source code in Visual Studio.

Code Generation from UML Model in Visual Studio

Learn how to produce source files from class model in Visual Paradigm.

Selecting UML Class from Source File in Visual Studio

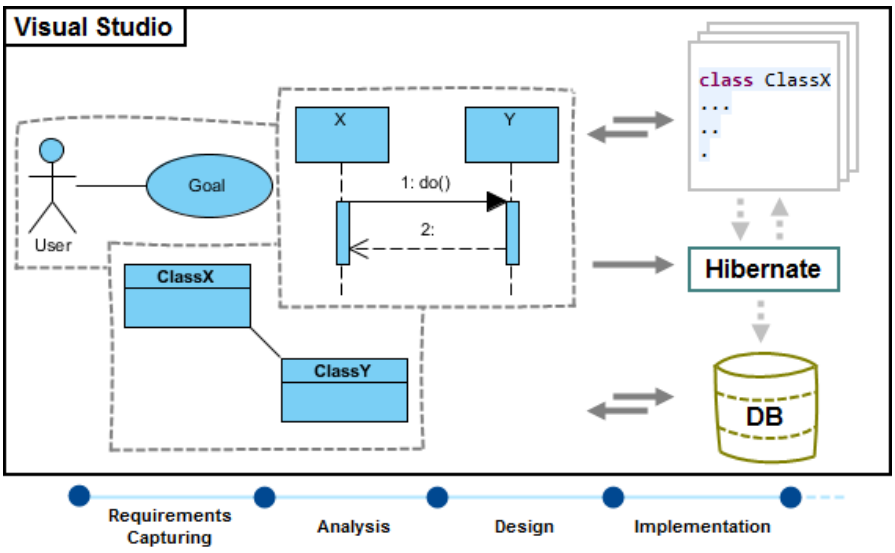
Learn how to select UML class model from a given source file.

Selecting Source File in Visual Studio from UML Class

Learn how to select source file from a given UML class model.

Overview and installation of Visual Studio integration

Visual Paradigm enables you to integrate the visual modeling environment with Visual Studio, providing full software development life cycle support. By designing your software system in Visual Paradigm, you can generate programming source code from class diagram to an Visual Studio project. Also, you can reverse engineer your source code into class models in Visual Paradigm.

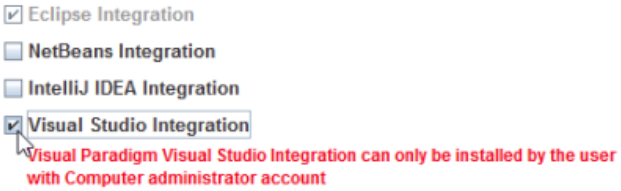


Visual Studio integration overview

Installation

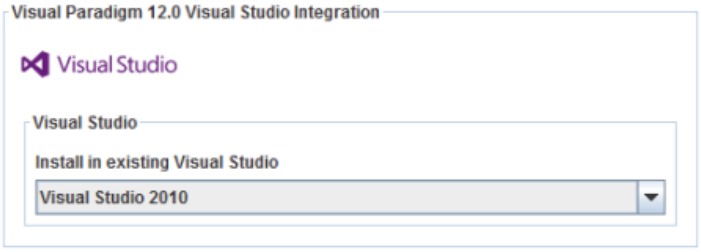
First of all, please make sure you have Visual Studio 2008 or above installed. To install Visual Studio Integration from Visual Paradigm:

1. In Visual Paradigm, select **Windows > Integration > IDE Integration...** from the toolbar.
2. Select Visual Studio. You can run Visual Paradigm in multiple IDEs. In other words, if you need you can select multiple IDEs here. Click **Next**.



Select Visual Studio Integration

3. Select the version of Visual Studio to integrate with. Click **Next** to start copying files to your IDE.



Version of Visual Studio

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

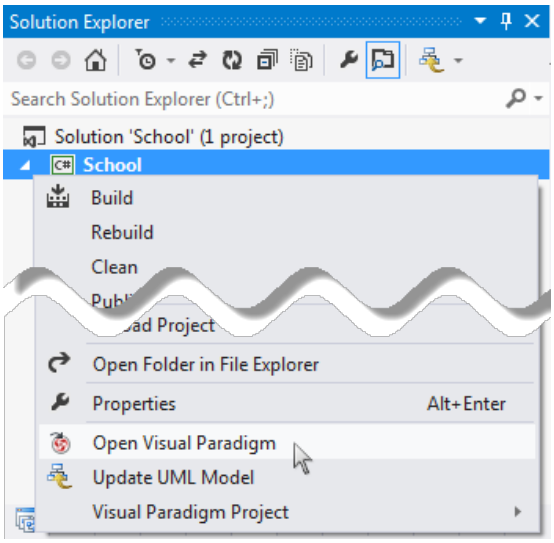
- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating a UML project in Visual Studio

You can create UML project for any of your project in Visual Studio. Note that one Visual Studio project can associate with at most one UML project and you cannot create UML project without associating it with any Visual Studio project. Once you have created a UML project for a Visual Studio project, you cannot remove it or de-associate it.

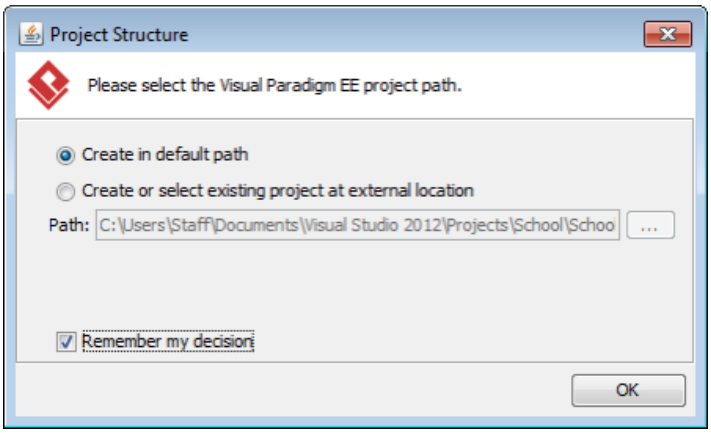
Creating a New UML Project

1. In Visual Studio, select the project where you want to create a UML project for it.
2. Right click on the project and select **Open Visual Paradigm** from the popup menu.



Open Visual Paradigm from Visual Studio project

3. Select from the **Project Structure** window the location of the Visual Paradigm project is to be saved. The Visual Paradigm project, with .vpp extension is the UML project file that is going to be associated with the selected Visual Studio project file. Select **Create in default path** will save the UML project to **%Visual Studio _Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.

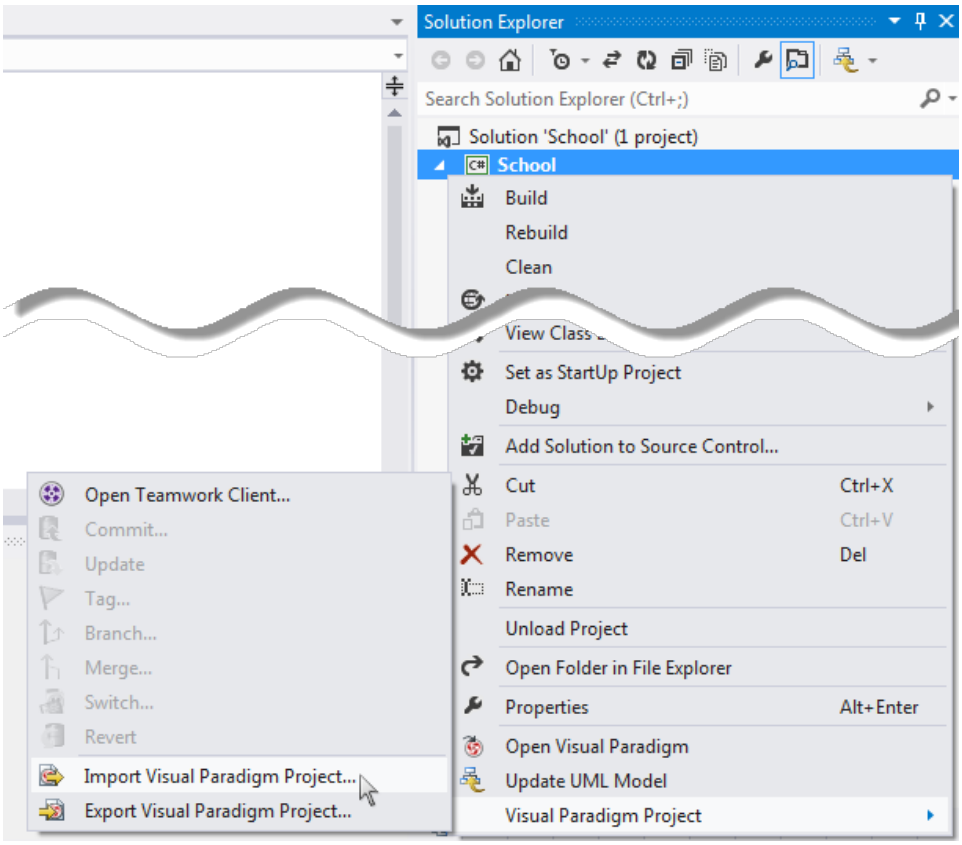


Create a new UML project

4. Click **OK**.

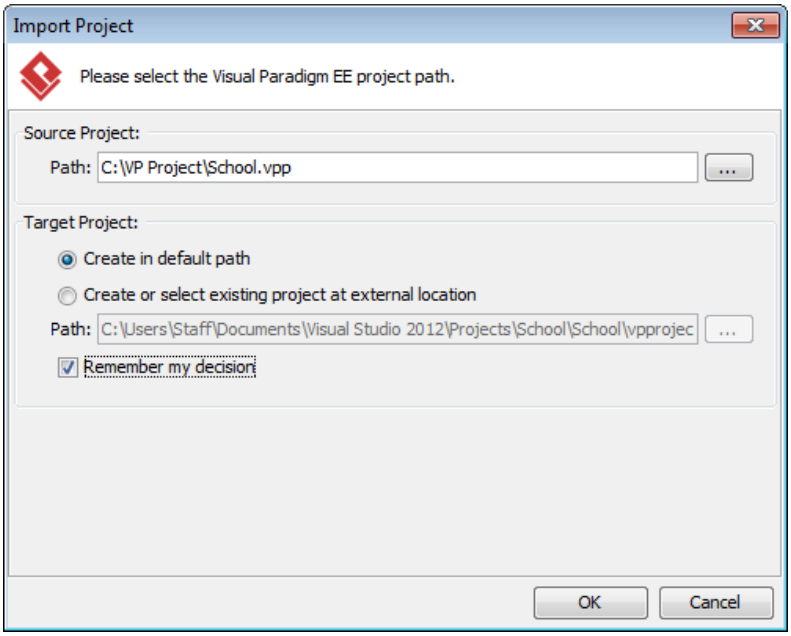
Creating a UML Project by Importing an Existing .vpp Project File

1. In Visual Studio, select the project where you want to create a UML project for it.
2. Right click on the project and select **Visual Paradigm Project > Import Visual Paradigm Project...** from the popup menu.



Import Visual Paradigm project

3. Specify the path of source .vpp project as well as the location of the imported project file is to be saved. Select **Create in default path** will save the UML project to **%Visual Studio _Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.



Import an existing .vpp project file

4. Click **OK**.

Related Resources

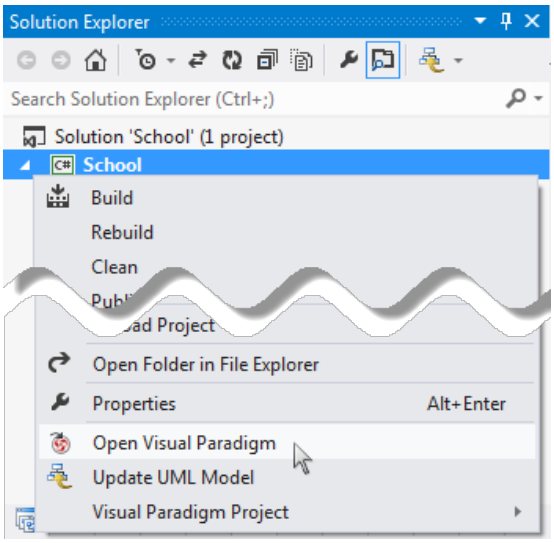
The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening a UML project in Visual Studio

Opening a UML Project

1. In Visual Studio, select the project where you want to open its UML project.
2. Right click on the project and select **Open Visual Paradigm** from the popup menu.



Open Visual Paradigm from Visual Studio project

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

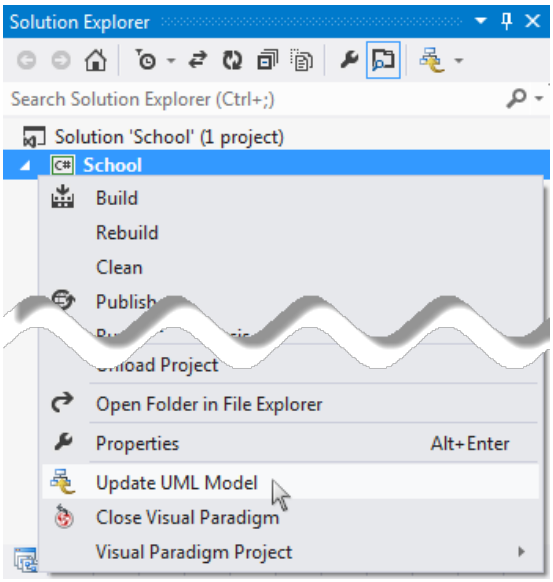
- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse engineering in Visual Studio

Reverse engineering is the process to reverse engineer UML model from source files in Visual Studio project. With reverse engineering you can visualize your program or system with class diagram. Before reverse engineering, you must open the UML project from the Visual Studio project.

Project Based Reverse Engineering

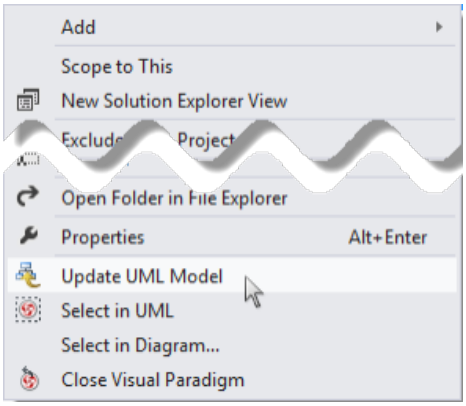
You can produce and update UML models from all source files in a Visual Studio project. Models of the selected project, child namespaces and classes will be created (if the models are not already exists) or updated. To reverse engineer from an Visual Studio project, right-click on the project node in Visual Studio and select **Update UML Model** from the popup menu.



Update the whole UML project from a Visual Studio project

Namespace Based Reverse Engineering

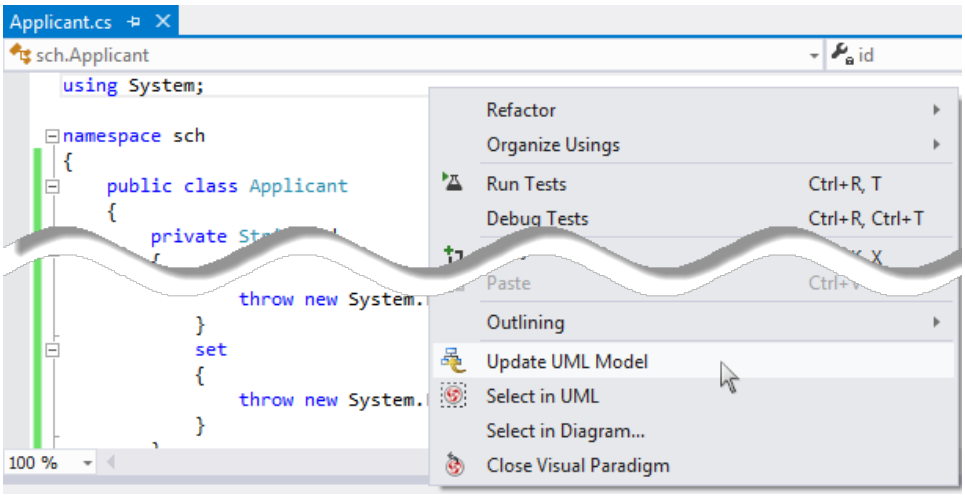
You can produce and update UML models from source files under namespace. Models of the selected namespace, child namespaces and classes will be created (if the models are not already exists) or updated. To reverse engineer from a namespace in a Visual Studio project, right-click on the namespace in any tree and select **Update UML Model** from the popup menu.



Update UML package and its containing classes from a namespace folder

Class Based Reverse Engineering

You can produce and update UML models from classes in Visual Studio. Models of the selected class and child classes (inner class) will be created (if the models are not already exists) or updated. To reverse engineer code from a class in a Visual Studio project, right-click on the class file in any tree or in code editor and select **Update UML Model** from the popup menu.



Update UML model from source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

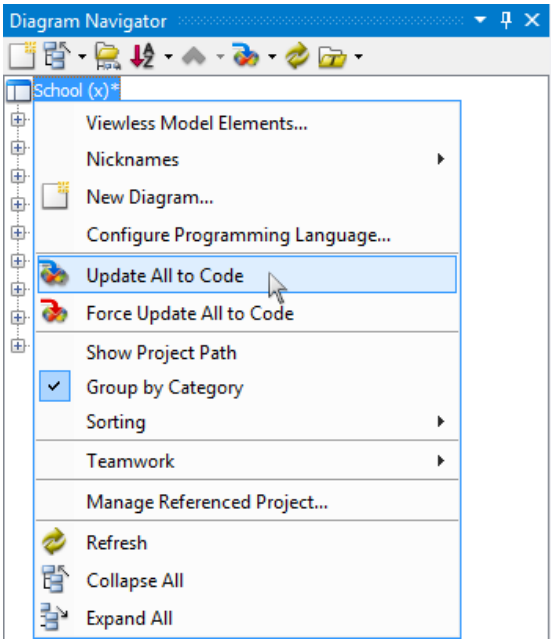
Code generation from UML model in Visual Studio

Code generation creates and updates source files in a Visual Studio project from UML models. You can select to update the whole project, package(s) and class(es) from Visual Paradigm to Visual Studio. Before updating source files, you must open the UML project from the Visual Studio project.

Project Based Code Generation

You can generate and update source files from the whole UML project. Packages and classes will be created (if not already exists) or updated. To generate/update source files from UML project, perform any of the steps below:

- Click  at the top of **Diagram Navigator**.
- Right click on the root node of **Diagram Navigator** and select **Update All to Code** from the popup menu.

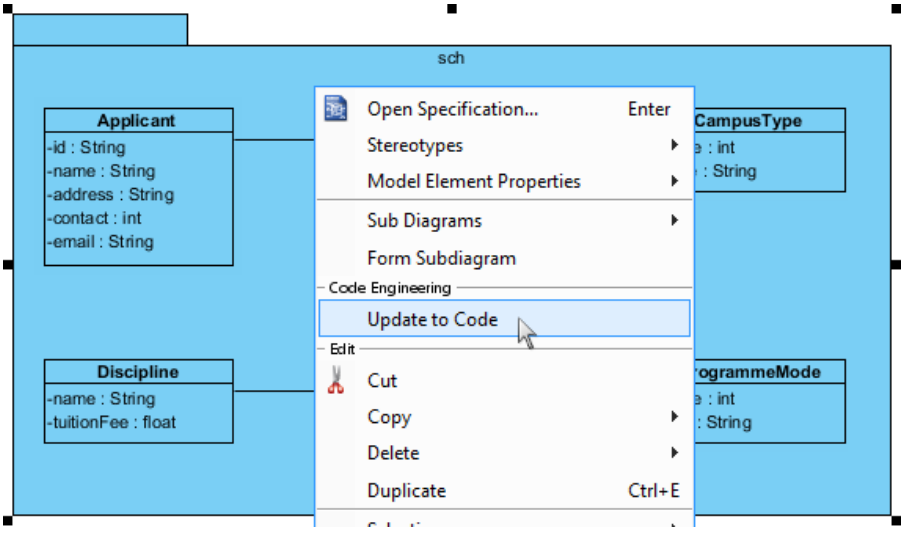


Update the whole project from UML project

Package (Namespace) Based Code Generation

You can generate and update namespace and its containing source file(s) from a UML package. Namespace and classes will be created (if not already exists) or updated. To generate/update source files from UML package, perform any of the steps below:

- Right click on the package in any diagram and select **Update to Code** from the popup menu.



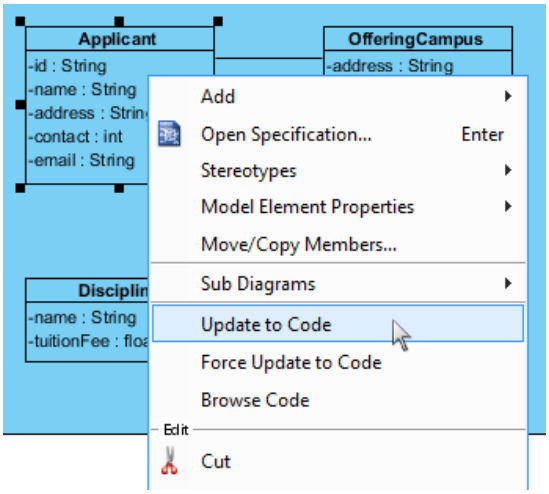
Update source files from UML package

- Right click on the package under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Class Based Code Generation

You can generate and update source file from a UML class. Class will be created (if not already exists) or updated. To generate/update source file from UML class, perform any of the steps below:

- Right click on the class in any diagram and select **Update to Code** from the popup menu.



Update source file from UML class

- Right click on the class under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

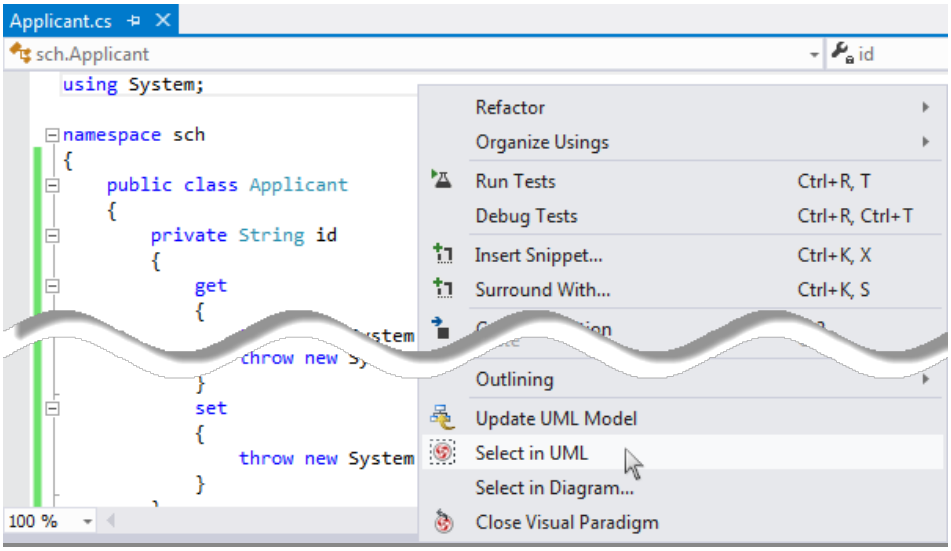
- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting UML class from source file in Visual Studio

Once a UML class is associated with a source file by code reversal/generation, you can select from source file the corresponding UML class in Visual Paradigm.

Selecting UML Class in Model Explorer

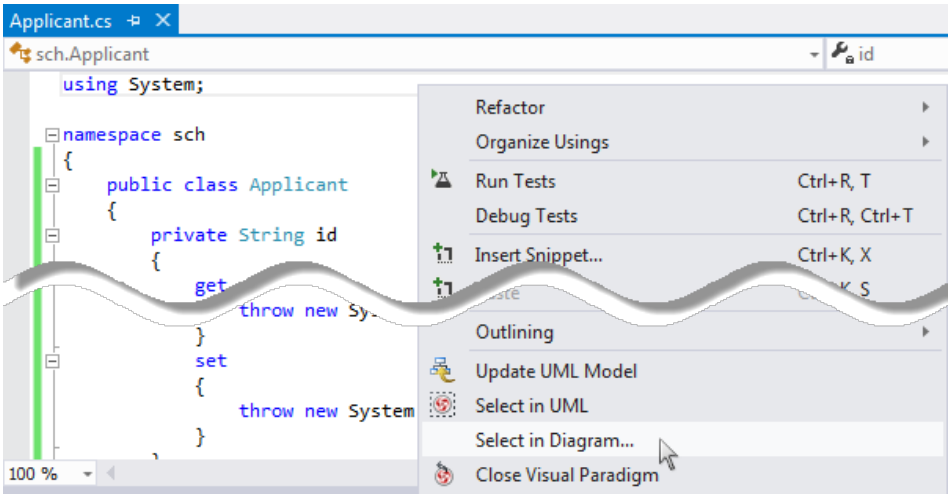
To select the UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Model** from the popup menu.



Open the UML class from a source file

Selecting UML Class in Diagram

To select the view of UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Diagram...** from the popup menu. If the class has been visualized with multiple views, you will be prompted to select a view to open.



Open the view of UML class from a source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

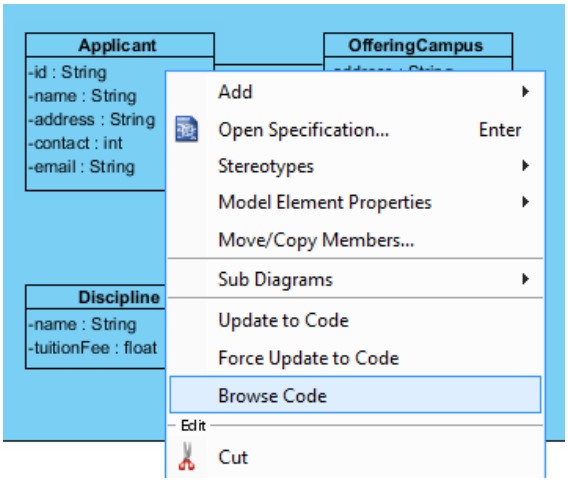
- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting source file in Visual Studio from UML class

Once a UML class is associated with a source file by code reversal/generation, you can select from UML class the corresponding source file in Visual Studio.

Selecting Source File from UML Class

To open a source file from a UML class, right-click on the UML class in **Diagram Navigator/Model Explorer/Class Repository** or in diagram and select **Browse Code** from the popup menu. This opens the corresponding source file in code editor.



Select source file from UML class

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - Generate C# source from UML class diagram in Visual Studio
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

NetBeans Integration

Overview and Installation of NetBeans Integration

Know how VP-UML can work with NetBeans through NetBeans integration. Learn how to install the integration from Visual Paradigm.

Creating a UML Project in NetBean

Learn how to create a UML project from Java project in NetBeans.

Opening a UML Project in NetBean

Learn how to open a UML project created from a Java project.

Reverse Engineering in NetBean

Learn how to reverse engineer class model from Java source code in NetBeans.

Code Generation from UML Model in NetBean

Learn how to produce source files from class model in Visual Paradigm.

Selecting UML Class from Source File in NetBean

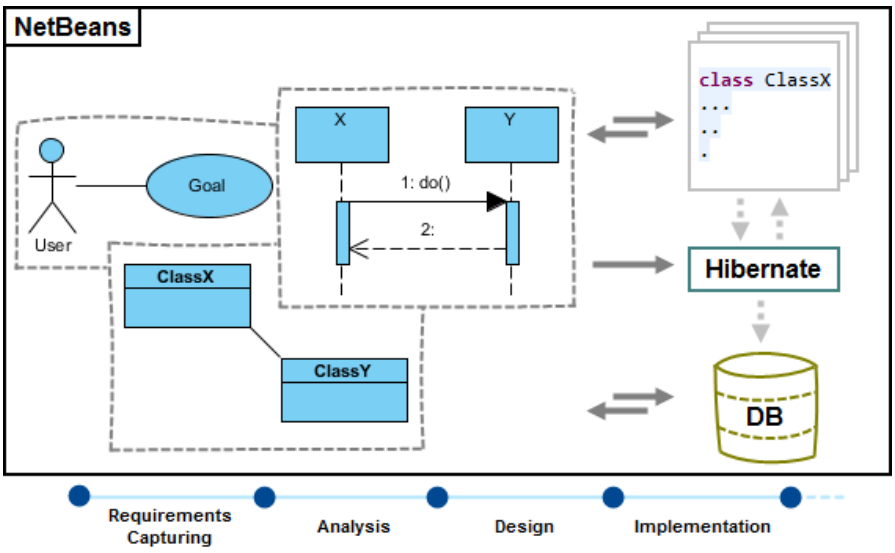
Learn how to select UML class model from a given source file.

Selecting Source File in NetBeans from UML Class

Learn how to select source file from a given UML class model.

Overview and installation of NetBeans integration

Visual Paradigm enables you to integrate the visual modeling environment with NetBeans, providing full software development life cycle support. By designing your software system in Visual Paradigm, you can generate programming source code from class diagram to an NetBeans project. Also, you can reverse engineer your source code into class models in Visual Paradigm.

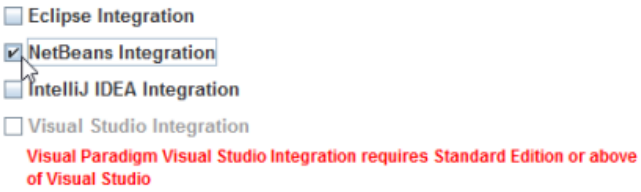


NetBeans integration overview

Installation

First of all, please make sure you have NetBeans 6.7 or above available. To install NetBeans Integration from Visual Paradigm:

1. In Visual Paradigm, select **Windows > Integration > IDE Integration...** from the toolbar.
2. Select NetBeans. You can run Visual Paradigm in multiple IDEs. In other words, if you need you can select multiple IDEs here. Click **Next**.



Select NetBeans Integration

3. Specify the folder path of NetBeans. Click **Next** to start copying files to your IDE.

Path of NetBeans

> **NOTE:** NetBeans integration can only be installed on one NetBeans directory only..

> **NOTE:** If you cannot find any Visual Paradigm menus in NetBeans after the installation, it could be due to our plug-in failed to be recognized by NetBeans. To solve this problem, please deactivate any plug-in in NetBeans and restart NetBeans.
> You can manage plug-in in NetBeans by selecting **Tools > Plugin** in NetBeans.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

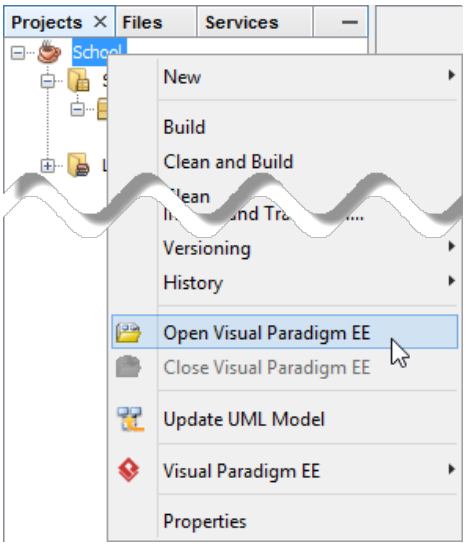
- Contact us if you need any help or have any suggestion

Creating a UML project in NetBeans

You can create UML project for any of your Java project in NetBeans. Note that one Java project can associate with at most one UML project and you cannot create UML project without associating it with any Java project. Once you have created a UML project for a Java project, you cannot remove it or de-associate it.

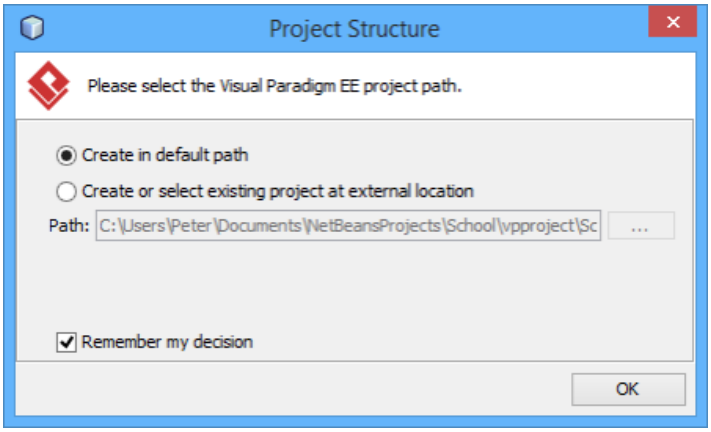
Creating a New UML Project

1. In NetBeans, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Open Visual Paradigm EE** from the popup menu.



Open Visual Paradigm from Java project

3. Select from the **Project Structure** window the location of the Visual Paradigm project is to be saved. The Visual Paradigm project, with .vpp extension, is the UML project file that is going to be associated with the selected NetBeans project file. Select **Create in default path** will save the UML project to **%NetBeans_Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.

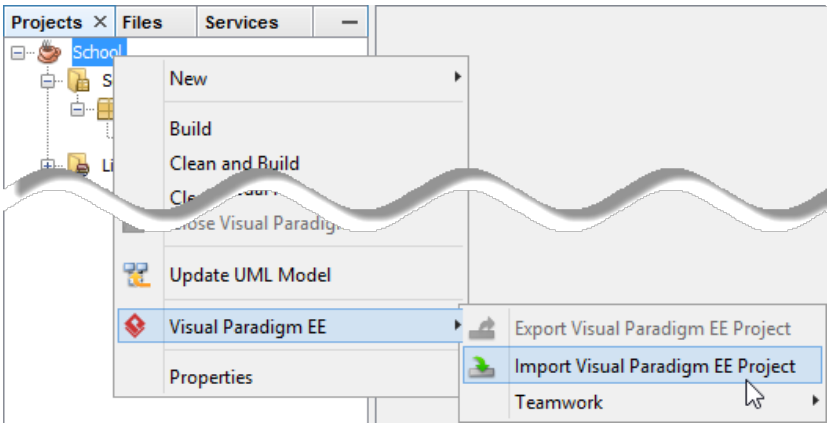


Create a new UML project

4. Click **OK**.

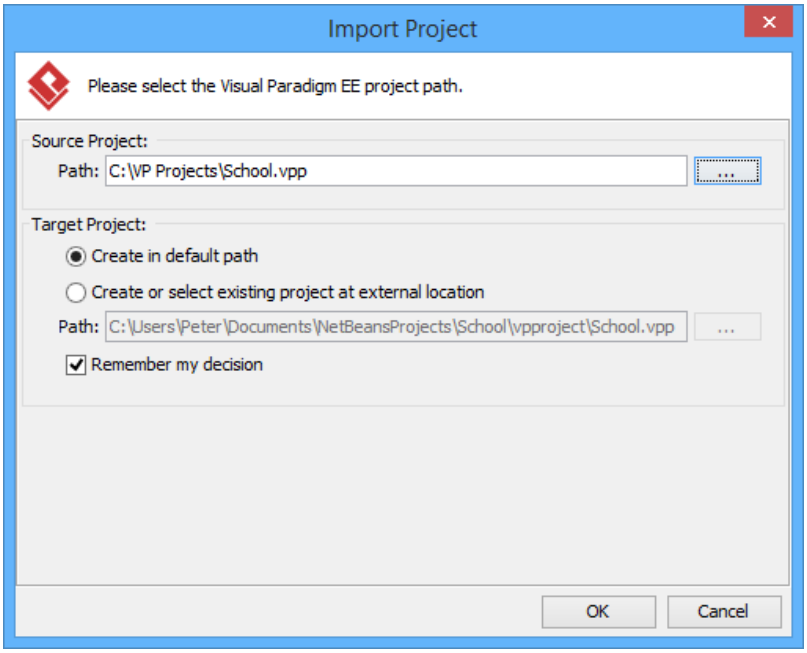
Creating a UML Project by Importing an Existing .vpp Project File

1. In NetBeans, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Visual Paradigm EE > Import Visual Paradigm EE Project...** from the popup menu.



Import Visual Paradigm project

3. Specify the path of source .vpp project as well as the location of the imported project file is to be saved. Select **Create in default path** will save the UML project to **%NetBeans_Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects check **Create in default path** and **Remember my decision**.



Import an existing .vpp project file

4. Click **OK**.

Related Resources

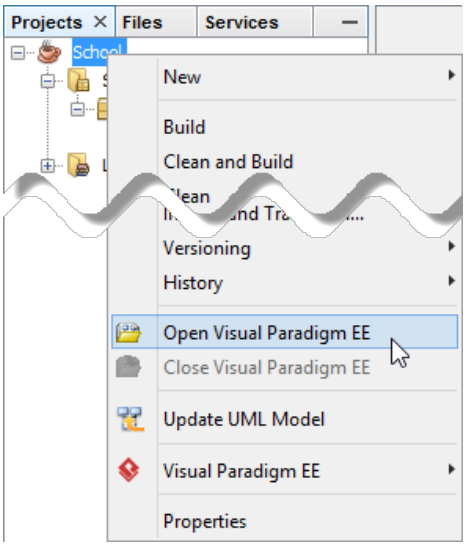
The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening a UML project in NetBeans

Opening a UML Project

1. In NetBeans, select the Java project where you want to open its UML project.
2. Right click on the project and select **Open Visual Paradigm EE** from the popup menu.



Open Visual Paradigm from Java project

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

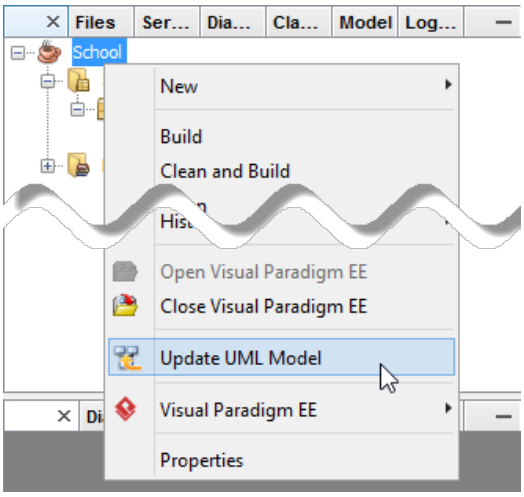
- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse engineering in NetBeans

Reverse engineering is the process to reverse engineer UML model from Java source. With reverse engineering, you can visualize your program or system with class diagram. Before reverse engineering, you must open the UML project from the Java project.

Project Based Reverse Engineering

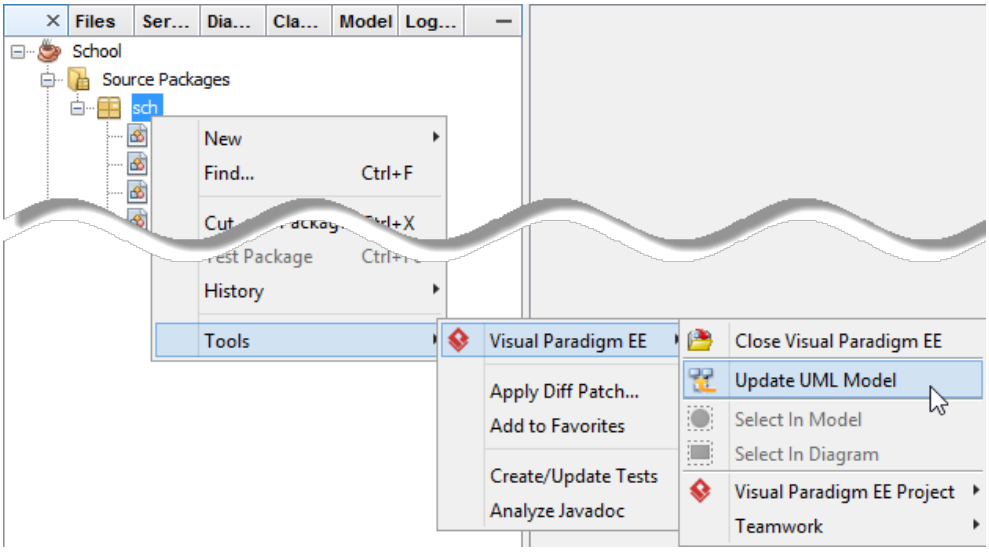
You can produce and update UML models from all source files in a Java project. Models of the selected project, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from an NetBeans project, right-click on the project node in NetBeans and select **Update UML Model** from the popup menu.



Update the whole UML model from a Java project

Package Based Reverse Engineering

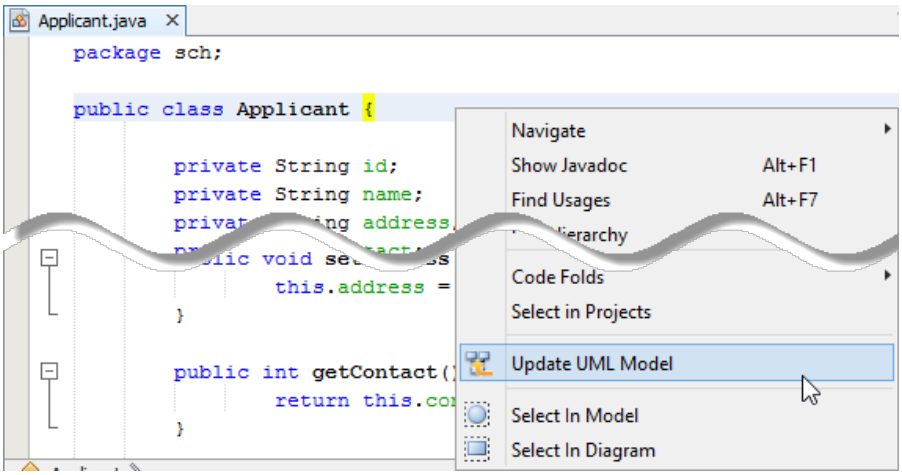
You can produce and update UML models from source files under a package. Models of the selected package, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from a package in a Java project, right-click on the package in any tree and select **Tools > Visual Paradigm EE > Update UML Model** from the popup menu.



Update UML package and its containing classes from a package folder

Class Based Reverse Engineering

You can produce and update UML models from classes in NetBeans. Models of the selected class and child classes (inner class) will be created (if the models are not already exists) or updated. To reverse engineer code from a class in a Java project, right-click on the class file in any tree or in code editor and select **Update UML Model** from the popup menu.



Update UML model from source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

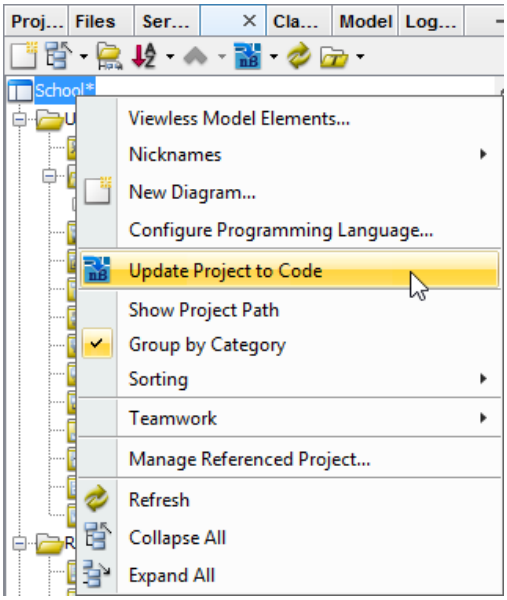
Code generation from UML model in NetBeans

Code generation creates and updates source files in a Java project from UML models. You can select to update the whole project, package(s) and class(es) from Visual Paradigm to NetBeans. Before updating source files, you must open the UML project from the Java project.

Project Based Code Generation

You can generate and update source files from the whole UML project. Packages and classes will be created (if not already exists) or updated. To generate/update source files from UML project, perform any of the steps below:

- Click in NetBeans toolbar.
- Click at the top of **Diagram Navigator**.
- Right click on the root node of **Diagram Navigator** and select **Update Project to Code** from the popup menu.

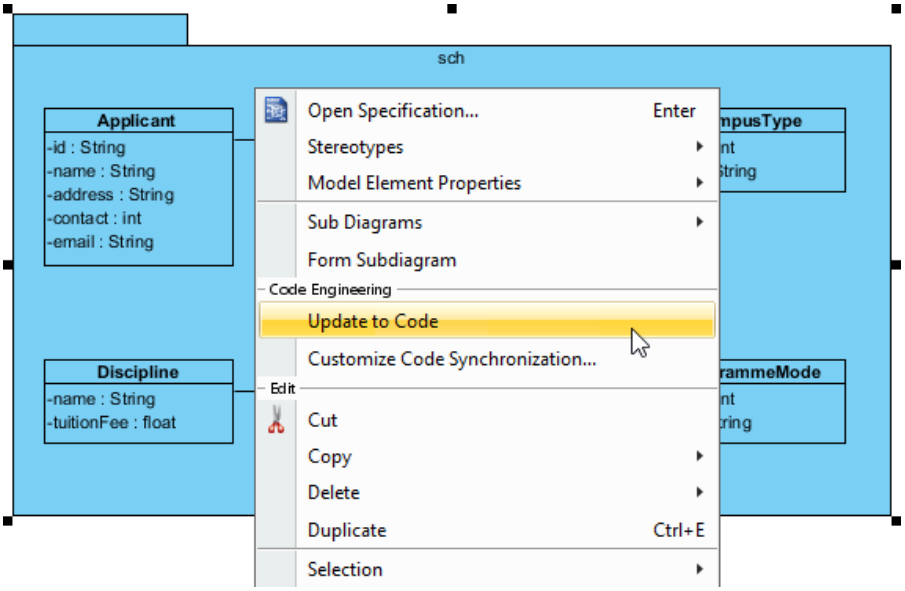


Update the whole Java project from UML project

Package Based Code Generation

You can generate and update package and its containing source file(s) from a UML package. Package and classes will be created (if not already exists) or updated. To generate/update source files from UML package, perform any of the steps below:

- Right click on the package in any diagram and select **Update to Code** from the popup menu.



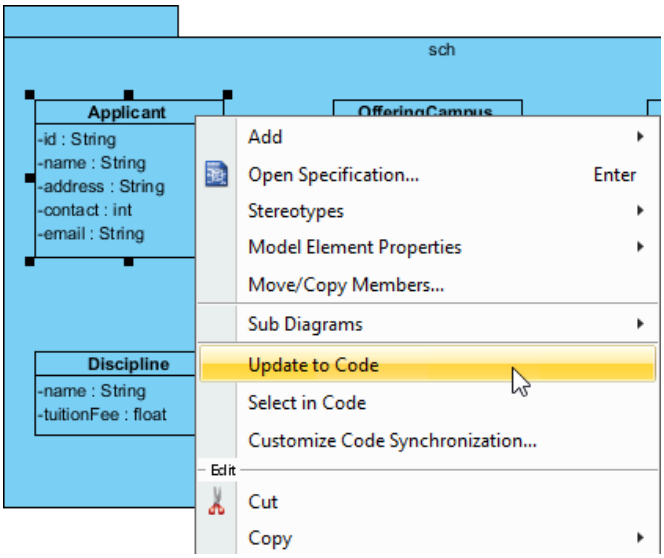
Update source files from UML package

- Right click on the package under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Class Based Code Generation

You can generate and update source file from a UML class. Class will be created (if not already exists) or updated. To generate/update source file from UML class, perform any of the steps below:

- Right click on the class in any diagram and select **Update to Code** from the popup menu.



Update source file from UML class

- Right click on the class under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

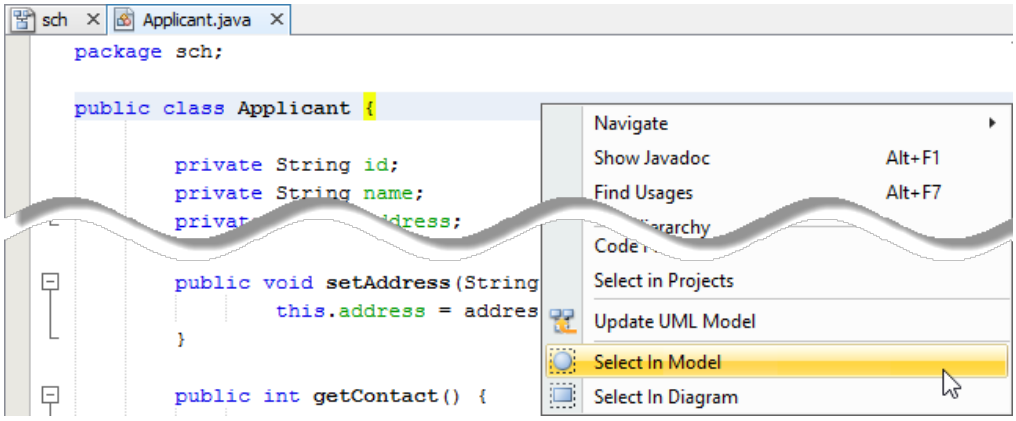
- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting UML Class from source file in NetBeans

Once a UML class is associated with a Java source by code reversal/generation, you can select from source file the corresponding UML class in Visual Paradigm.

Selecting UML Class in Model Explorer

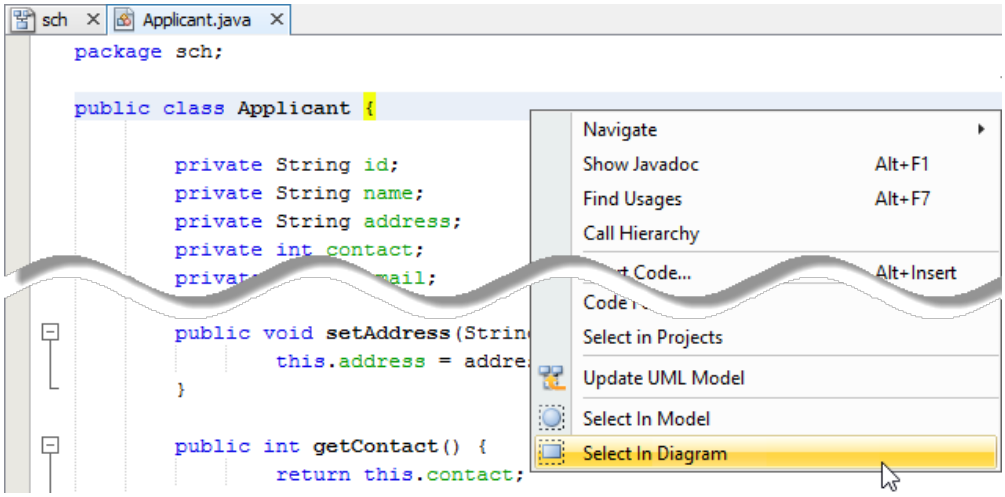
To select the UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Model** from the popup menu.



Open the UML class from a source file

Selecting UML Class in Diagram

To select the view of UML class from a source file, right-click on the class file in any tree or in code editor and select **Select In Diagram** from the popup menu. If the class has been visualized with multiple views, you will be prompted to select a view to open.



Open the view of UML class from a source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

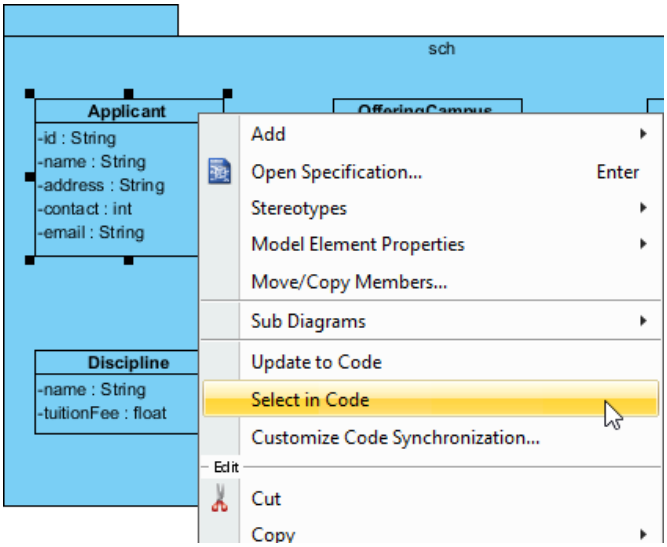
- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting source file in NetBeans from UML class

Once a UML class is associated with a Java source by code reversal/generation, you can select from UML class the corresponding Java source file in NetBeans.

Selecting Java Source from UML Class

To open a Java source file from a UML class, right-click on the UML class in **Diagram Navigator/Model Explorer/Class Repository** or in diagram and select **Select in Code** from the popup menu. This opens the corresponding source file in code editor.



Select source file from UML class

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- Tutorial - 4 quick steps to start UML modeling in NetBeans
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

IntelliJ IDEA Integration

Overview and Installation of IntelliJ IDEA Integration

Know how Visual Paradigm can work with IntelliJ IDEA through IntelliJ IDEA integration. Learn how to install the integration from Visual Paradigm.

Creating a UML Project in IntelliJ IDEA

Learn how to create a UML project from Java project in IntelliJ IDEA.

Opening a UML Project in IntelliJ IDEA

Learn how to open a UML project created from a Java project.

Reverse Engineering in IntelliJ IDEA

Learn how to reverse engineer class model from Java source code in IntelliJ IDEA.

Code Generation from UML Model in IntelliJ IDEA

Learn how to produce source files from class model in Visual Paradigm.

Selecting UML Class from Source File in IntelliJ IDEA

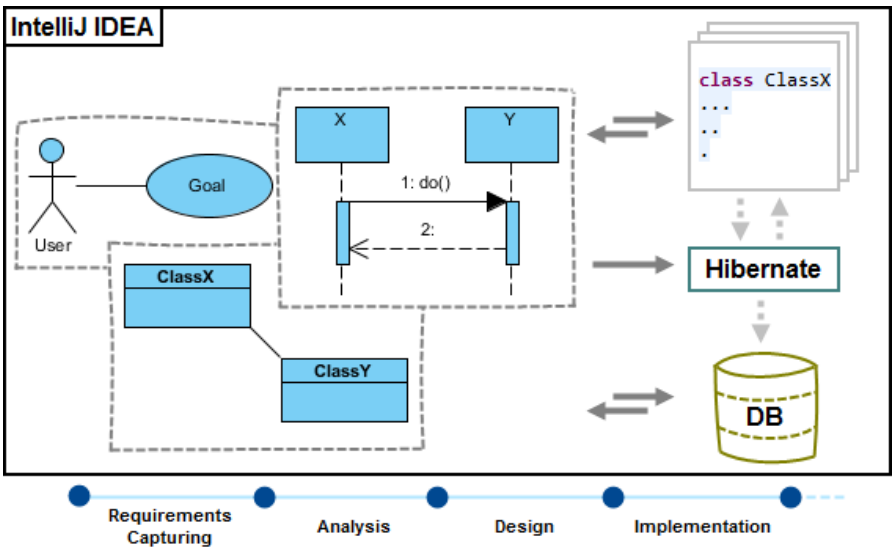
Learn how to select UML class model from a given source file.

Selecting Source File in IntelliJ IDEA from UML Class

Learn how to select source file from a given UML class model.

Overview and installation of IntelliJ IDEA integration

Visual Paradigm enables you to integrate the visual modeling environment with IntelliJ IDEA, providing full software development life cycle support. By designing your software system in Visual Paradigm, you can generate programming source code from class diagram to an IntelliJ IDEA project. Also, you can reverse engineer your source code into class models in Visual Paradigm.

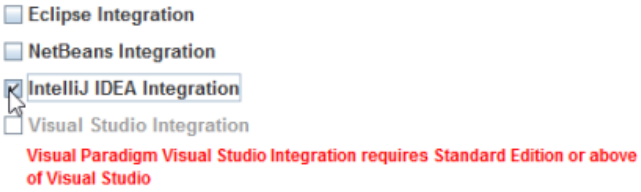


IntelliJ IDEA integration overview

Installation

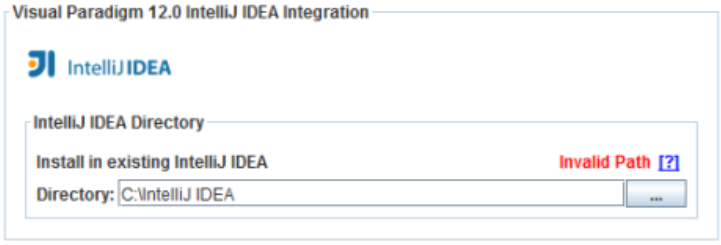
First of all, please make sure you have IntelliJ IDEA 11.0 or above available. To install IntelliJ IDEA Integration from Visual Paradigm:

1. In Visual Paradigm, select **Windows > Integration > IDE Integration...** from the toolbar.
2. Select IntelliJ IDEA. You can run Visual Paradigm in multiple IDEs. In other words, if you need you can select multiple IDEs here. Click **Next**.



Select IntelliJ IDEA Integration

3. Specify the folder path of IntelliJ IDEA. Click **Next** to start copying files to your IDE.



Path of IntelliJ IDEA

> **NOTE:** IntelliJ IDEA integration can only be installed on one IntelliJ IDEA directory only.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

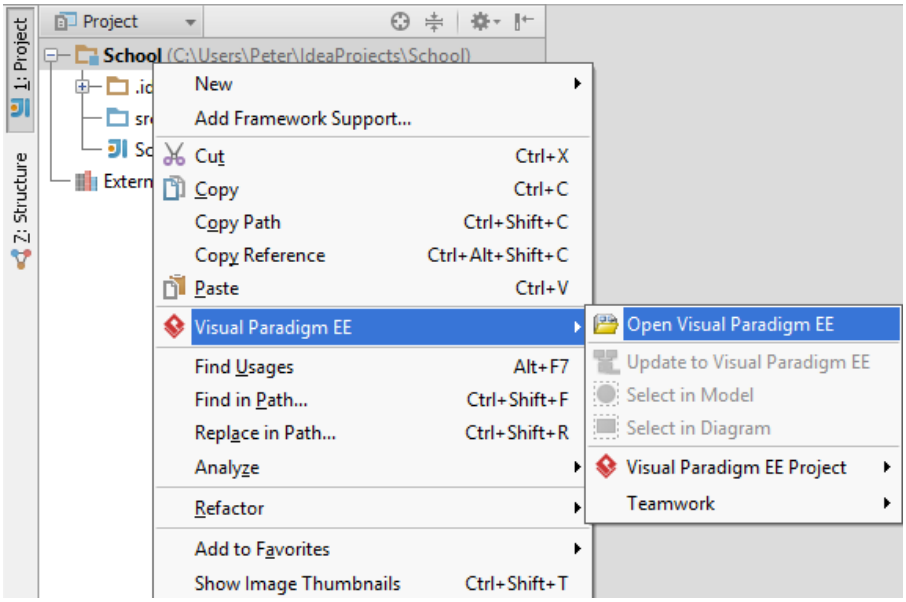
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Creating a UML project in IntelliJ IDEA

You can create UML project for any of your Java project in IntelliJ IDEA. Note that one Java project can associate with at most one UML project and you cannot create UML project without associating it with any Java project. Once you have created a UML project for a Java project, you cannot remove it or de-associate it.

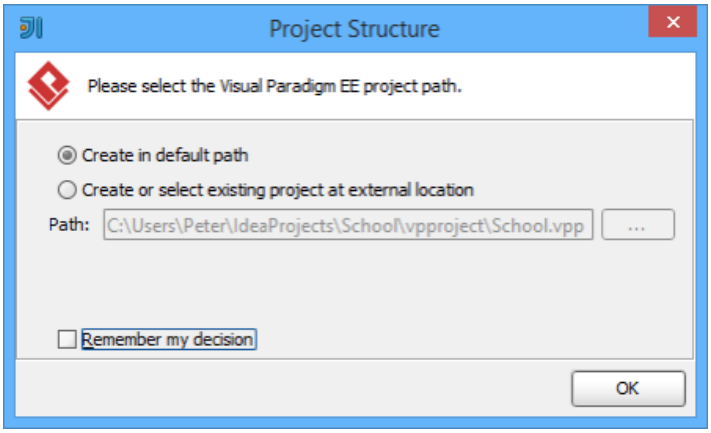
Creating a New UML Project

1. In IntelliJ IDEA, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Visual Paradigm EE > Open Visual Paradigm EE** from the popup menu.



Open Visual Paradigm from Java project

3. Select from the **Project Structure** window the location of the Visual Paradigm project is to be saved. The Visual Paradigm project, with .vpp extension, is the UML project file that is going to be associated with the selected IntelliJ IDEA project file. Select **Create in default path** will save the UML project to **%IntelliJ IDEA _Project_Directory%/vpproject** while selecting **Create at external location** require you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.

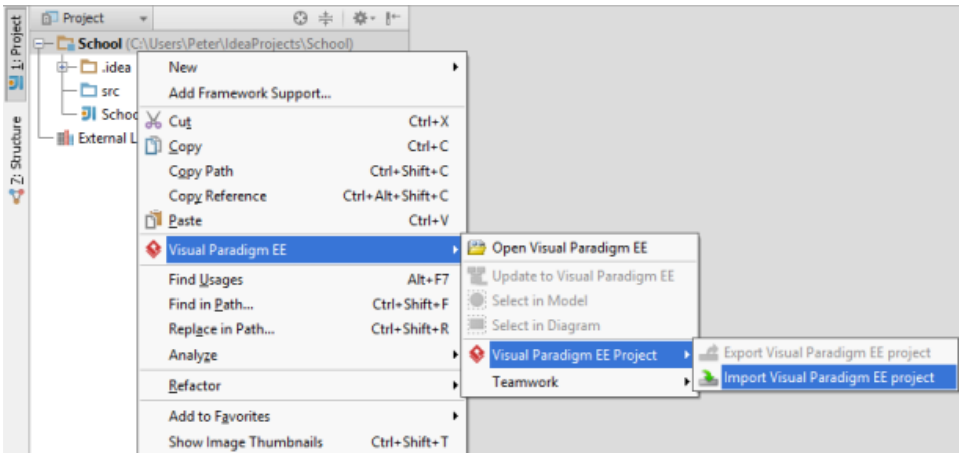


Create a new UML project

4. Click **OK**.

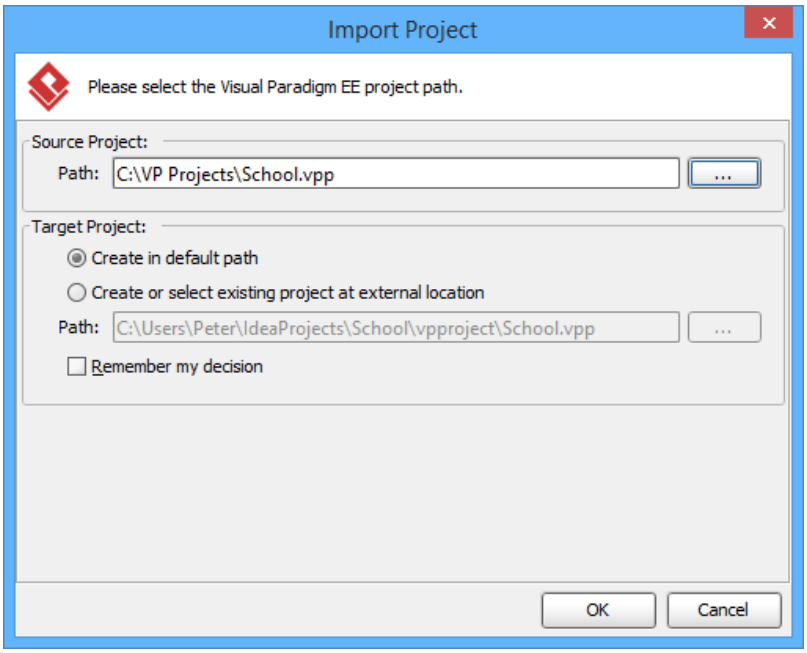
Creating a UML Project by Importing an Existing .vpp Project File

1. In IntelliJ IDEA, select the Java project where you want to create a UML project for it.
2. Right click on the project and select **Visual Paradigm EE > Visual Paradigm EE Project > Import Visual Paradigm EE Project...** from the popup menu.



Import Visual Paradigm project

3. Specify the path of source .vpp project as well as the location of the imported project file is to be saved. Select **Create in default path** will save the UML project to **%IntelliJ IDEA _Project_Directory%/vpproject** while selecting **Create at external location** requires you to specify the project path you desire. If you want to create in default path for all projects, check **Create in default path** and **Remember my decision**.



Import an existing .vpp project file

4. Click **OK**.

Related Resources

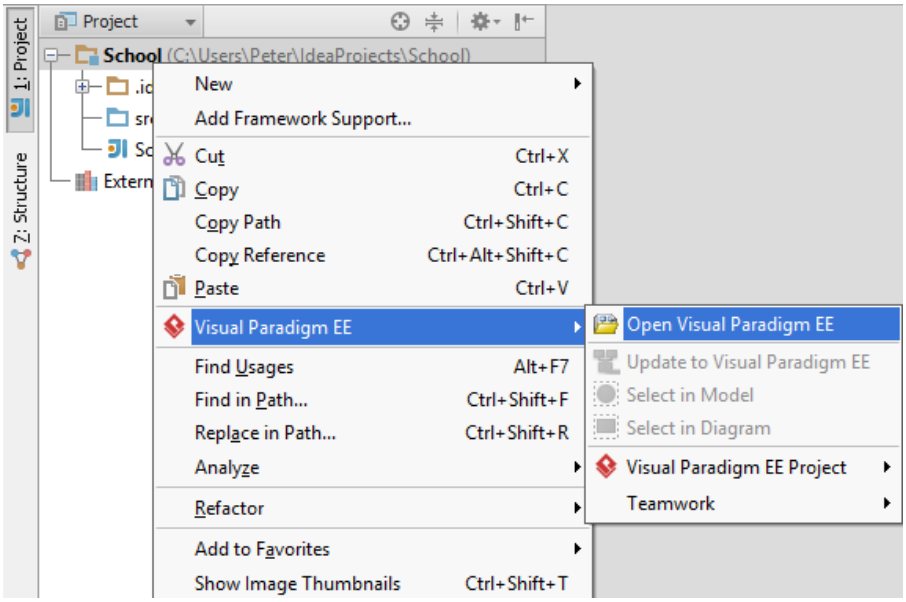
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Opening a UML project in IntelliJ IDEA

Opening a UML Project

1. In Eclipse, select the Java project where you want to open its UML project.
2. Right click on the project and select **Visual Paradigm EE > Open Visual Paradigm EE** from the popup menu.



Open Visual Paradigm from Java project

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

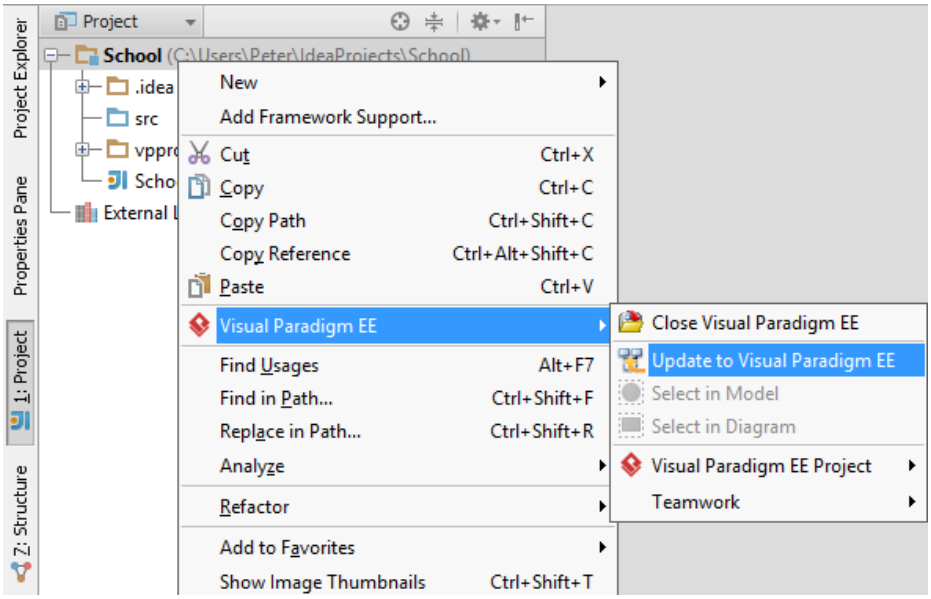
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Reverse engineering in IntelliJ IDEA

Reverse engineering is the process to reverse engineer UML model from Java source. With reverse engineering you can visualize your program or system with class diagram. Before reverse engineering, you must open the UML project from the Java project.

Project Based Reverse Engineering

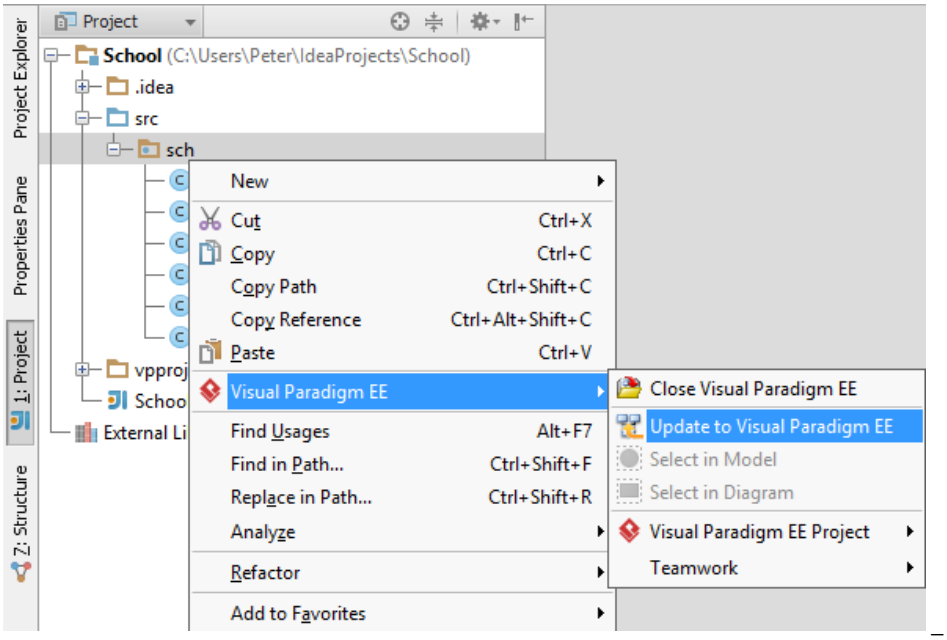
You can produce and update UML models from all source files in a Java project. Models of the selected project, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from an IntelliJ IDEA project, right-click on the project node in IntelliJ IDEA and select **Visual Paradigm EE > Update to Visual Paradigm EE** from the popup menu.



Update the whole UML model from a Java project

Package Based Reverse Engineering

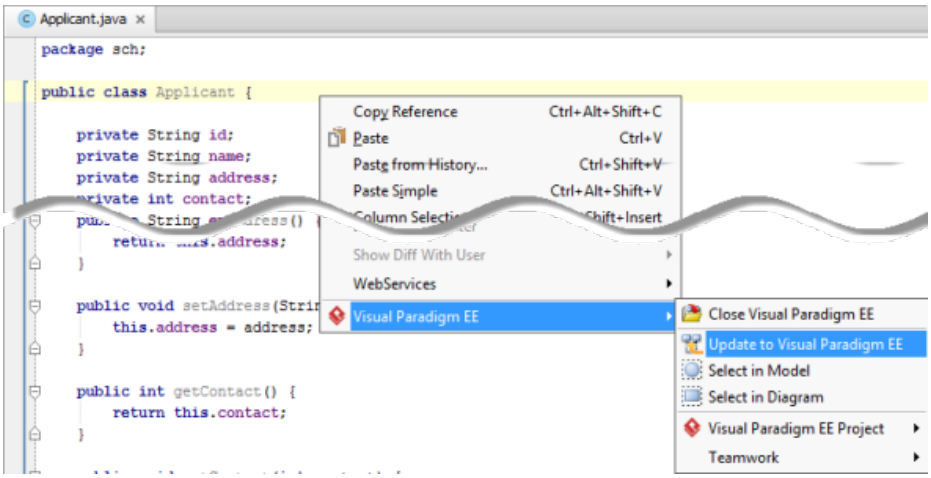
You can produce and update UML models from source files under a package. Models of the selected package, child packages and classes will be created (if the models are not already exists) or updated. To reverse engineer from a package in a Java project, right-click on the package in any tree and select **Visual Paradigm EE > Update to Visual Paradigm EE** from the popup menu.



Update UML package and its containing classes from a package folder

Class Based Reverse Engineering

You can produce and update UML models from classes in IntelliJ IDEA. Models of the selected class and child classes (inner class) will be created (if the models are not already exists) or updated. To reverse engineer code from a class in a Java project, right-click on the class file in any tree or in code editor and select **Visual Paradigm EE > Update to Visual Paradigm EE** from the popup menu.



Update UML model from source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

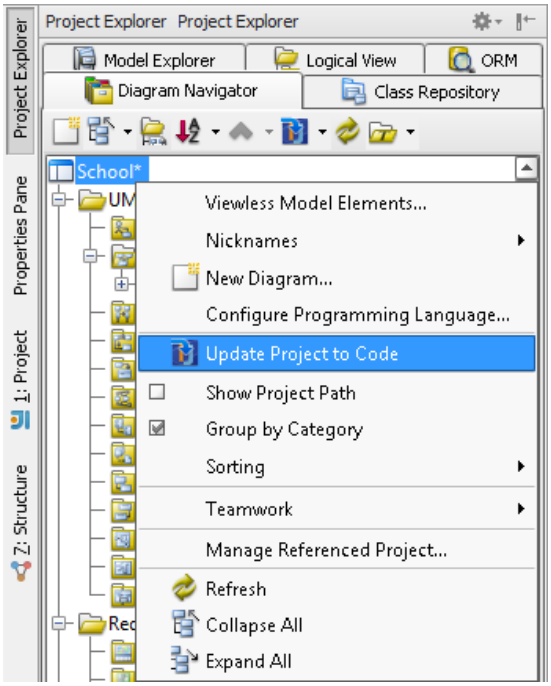
Code generation from UML model in IntelliJ IDEA

Code generation creates and updates source files in a Java project from UML models. You can select to update the whole project, package(s) and class(es) from Visual Paradigm to IntelliJ IDEA. Before updating source files, you must open the UML project from the Java project.

Project Based Code Generation

You can generate and update source files from the whole UML project. Packages and classes will be created (if not already exists) or updated. To generate/update source files from UML project, perform any of the steps below:

- Click in IntelliJ IDEA toolbar.
- Click at the top of **Diagram Navigator**.
- Right click on the root node of **Diagram Navigator** and select **Update Project to Code** from the popup menu.

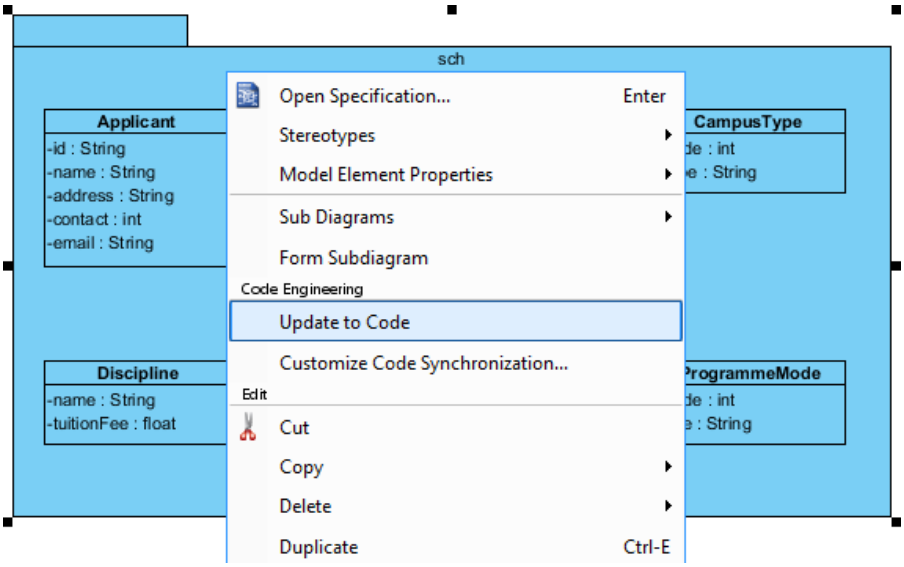


Update the whole Java project from UML project

Package Based Code Generation

You can generate and update package and its containing source file(s) from a UML package. Package and classes will be created (if not already exists) or updated. To generate/update source files from UML package, perform any of the steps below:

- Right click on the package in any diagram and select **Update to Code** from the popup menu.



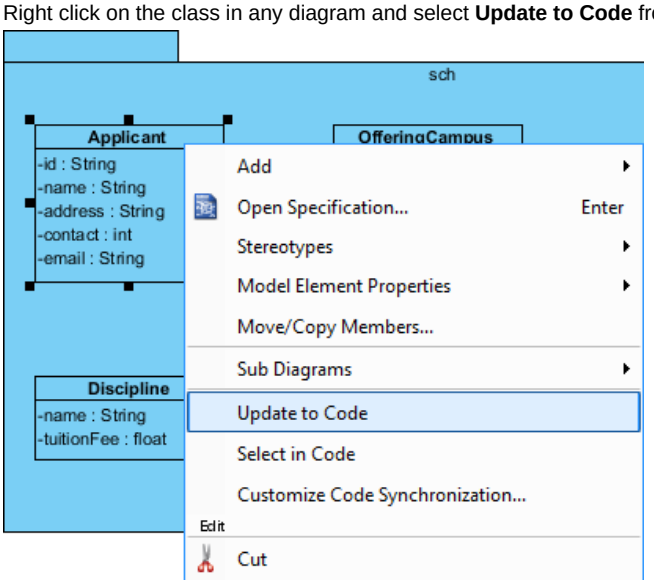
Update source files from UML package

- Right click on the package under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Class Based Code Generation

You can generate and update source file from a UML class. Class will be created (if not already exists) or updated. To generate/update source file from UML class, perform any of the steps below:

- Right click on the class in any diagram and select **Update to Code** from the popup menu.



Update source file from UML class

- Right click on the class under **Diagram Navigator/Model Explorer/Class Repository** and select **Update to Code** from the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

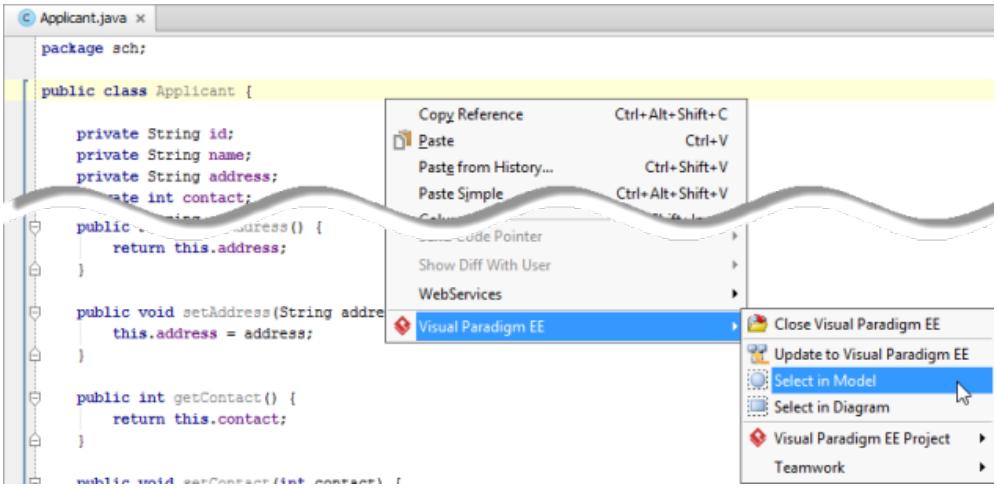
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting UML class from source file in IntelliJ IDEA

Once a UML class is associated with a Java source by code reversal/generation, you can select from source file the corresponding UML class in Visual Paradigm.

Selecting UML Class in Model Explorer

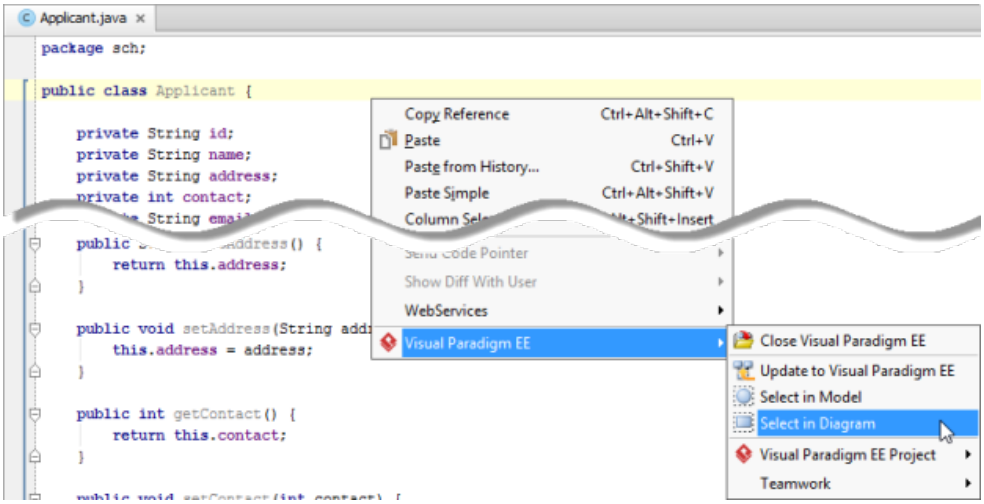
To select the UML class from a source file, right-click on the class file in any tree or in code editor and select **Visual Paradigm EE > Select in Model** from the popup menu.



Open the UML class from a source file

Selecting UML Class in Diagram

To select the view of UML class from a source file, right-click on the class file in any tree or in code editor and select **Visual Paradigm EE > Select in Diagram** from the popup menu. If the class has been visualized with multiple views, you will be prompted to select a view to open.



Open the view of UML class from a source file

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

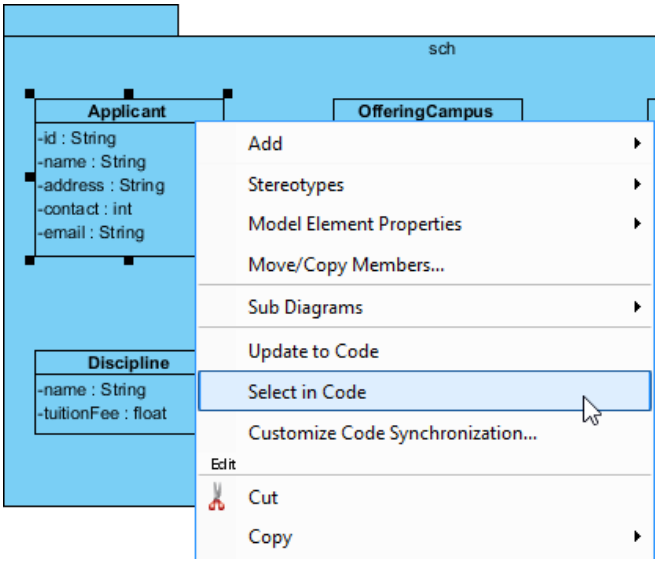
- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Selecting source file in IntelliJ IDEA from UML class

Once a UML class is associated with a Java source by code reversal/generation, you can select from UML class the corresponding Java source file in IntelliJ IDEA.

Selecting Java Source from UML Class

To open a Java source file from a UML class, right-click on the UML class in **Diagram Navigator/Model Explorer/Class Repository** or in diagram and select **Select in Code** from the popup menu. This opens the corresponding source file in code editor.



Select source file from UML class

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import XML

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. This chapter focuses on the interoperability with XML file.

Exporting XML

Shows you how to export project data to XML file.

Importing XML

Shows you how to import XML to an opening project.

Exporting XML

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. Visual Paradigm supports interoperability with XML file. You can export project data to an XML, manipulate it externally, and feed the changes back to Visual Paradigm. In this chapter, you will see how to export XML file of whole project or specific diagram in project.

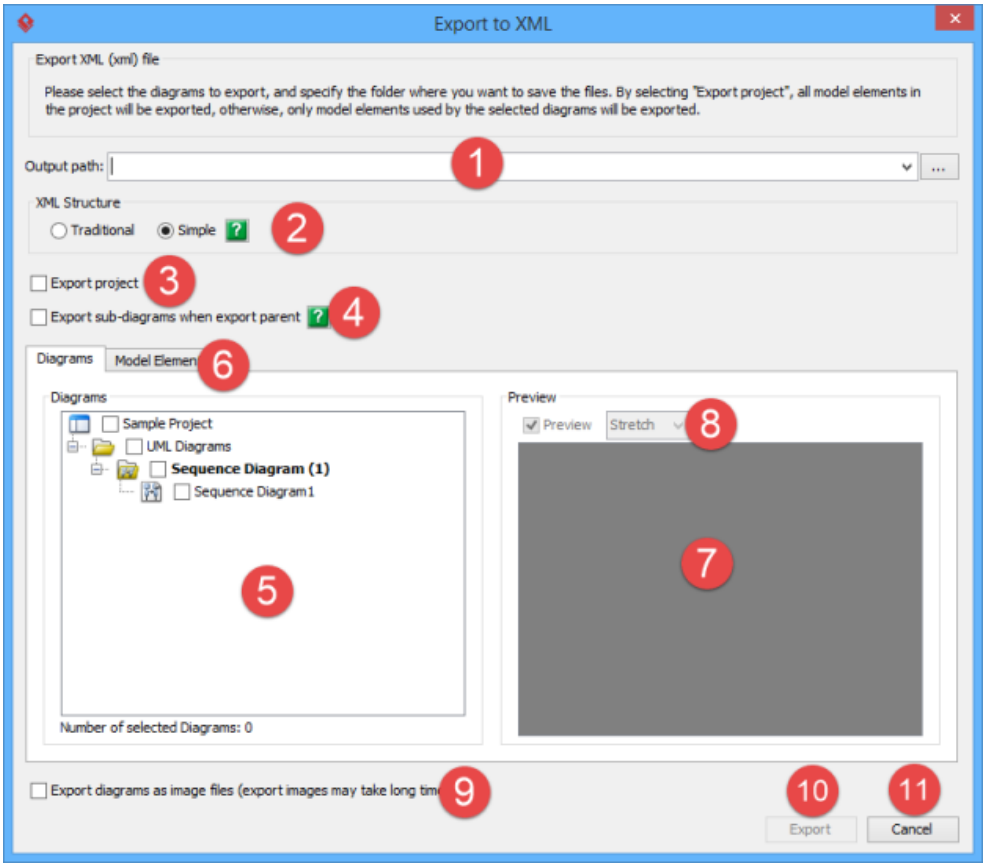
Exporting whole project to XML

1. Select **Project > Export > XML...** from the toolbar.
2. Specify the output destination.
3. Check **Export project** to export everything in the project to XML or keep it unchecked and check the diagram(s) to export only their content.
4. Click **Export**. Upon finishing, you can visit the output destination specified to obtain the XML.

Exporting active diagram to XML

1. Right click on diagram and select **Export > Export XML...** from the popup menu.
2. In the **Export to XML** window, specify the output destination which is a folder for containing the exported XML files and optionally the image files of diagrams.
3. Click **Export**.

Overview of exporting XML



Overview of Export XML window

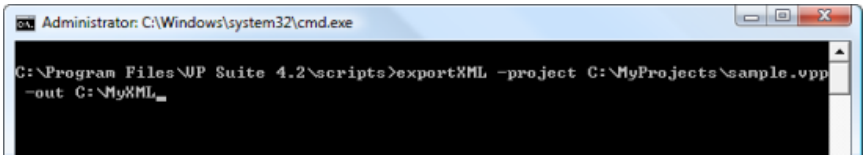
No.	Name	Description
1	Output destination	The location where you want to save the file.
2	XML structure	Controls how model data is to be presented in the exported XML file. **Traditional**: XML elements are named in more general manner. For example, Model, StringProperty, etc. **Simple**: XML elements are named using the name of the model. For example, UseCase, Frame, etc.
3	Export project	By checking **Export project**, all models in the project will be exported.
4	Export sub-diagrams when export parent	By selecting **Export sub-diagrams when export parent**, the sub-diagram(s) will also be exported when the parent diagram(s) is exported. For example, package *MyWorks* contains diagrams A and B. If you select to export diagram A, its parent "MyWork" will get exported, too. With the option **Export sub-**

		diagrams when export parent on, B will get exported too since B is a sub-diagram of package *MyWork*s. By turning off the option, B will be ignored when export.
5	Diagram list	A list of diagram of your project. Select the diagrams to export to XML.
6	Model elements	A list of model elements of your project. Select the model elements to export to XML.
7	Preview window	By checking the selected diagram and **Show preview**, it will be shown in preview window.
8	Preview mode	You can choose either **Stretch** or **Real size** to preview your diagram. **Stretch**: The ratio of your diagram will be fit in the size of preview window. **Real size**: The ratio of your diagram will be shown on the preview window as its real size.
9	Export diagrams as images	When checked, image file of the selected diagrams will also be exported along with the exported XML.
10	Export	Click **Export** to proceed with exporting to XML.
11	Cancel	Click **Cancel** to discard exporting to XML.

Description of Export XML window

Exporting diagrams to XML with command line interface

1. Start the command console.
2. In the console, change directory to the scripts folder and execute **ExportXML** with parameters required.



Parameters for ExportXML

This displays the usage of the command. Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	Folder for storing the exported XML and images	C:\Demo\Output
-diagram	One or more diagrams to be exported	"Diagram A" "Diagram B"
-noimage	Do not export image files for diagrams	N/A

Parameters for ExportXML

Upon finishing, you can visit the output destination specified to obtain the XML files.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing XML

You can import changes made externally in XML back to Visual Paradigm, to update the project data. In this chapter, you will see how to import an XML exported before. Notice that XML import is made in response to XML export in Visual Paradigm. Only XML exported from Visual Paradigm can be imported.

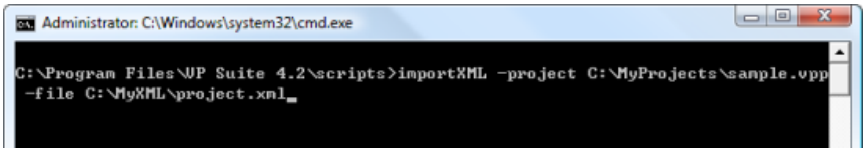
Importing XML to current project

1. Select **Project > Import > XML...** from the toolbar.
2. Specify the file path of the XML to import.
3. Click **OK**.

> **NOTE:** All changes made in project will be overwritten by data in XML. For example, if class Foo is renamed to Bar. By importing an XML exported before renaming class, Bar will be renamed to Foo.

Importing XML to project with command line interface

1. Start the command console.
2. In the console, change directory to the scripts folder and execute **ImportXML** with the parameters required.



Parameters for ImportXML

This displays the usage of the export command. Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the XML file to import	C:\Demo\input\sample.xml

Parameters for ImportXML

Upon finishing, the project file will be updated with the data presented in the XML file.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import VP project

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. This chapter focuses on the interoperability with native .vpp project file.

Exporting VP project

Shows you how to export project data to project file.

Importing VP project

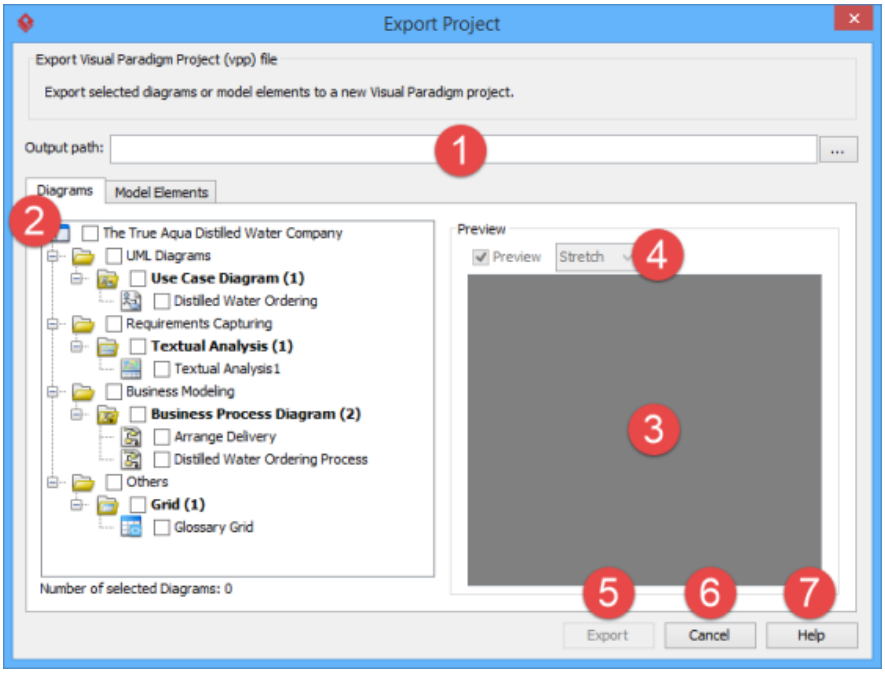
Shows you how to import project file to an opening project.

Exporting VP project

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. Visual Paradigm supports interoperability with native .vpp project file. You can export some of the diagrams to a project file, send to your team member for editing and feed the changes back to Visual Paradigm. In this chapter, you will see how to export project file for diagrams in project. To export VP project:

1. Select **Project > Export > Visual Paradigm Project...** from the toolbar.
2. Specify the output destination.
3. Check in the diagram tree the diagrams to export. If you want to export the whole project, check the top most root node.
4. Click **Export** button. Upon finishing, you can visit the output destination specified to obtain the .vpp project file.

Overview of exporting VP project



*Overview of **Export Project** dialog box*

No.	Name	Description
1	Output destination	The location where you want to save the file.
2	Diagrams / Model Elements	Select the diagram(s) or model element(s) to export.
3	Preview window	By checking the selected diagram and Show preview, it will be shown in preview window.
4	Preview mode	You can choose either **Stretch** or **Real size** to preview your diagram. **Stretch**: The ratio of your diagram will be fit in the size of preview window. **Real size**: The ratio of your diagram will be shown on the preview window as its real size.
5	Export	Click **Export** to proceed with exporting to VP project.
6	Cancel	Click **Cancel** to discard exporting to VP project.
7	Help	More information about how to export VP Project can be obtained by clicking this button.

*Description of **Export Project** dialog box*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing VP project

You can import a VP project to Visual Paradigm for importing changes made in an exported project, or to import model in another project. In this chapter, you will see how to import a VP project. To import a project:

1. Select **Project > Import > Visual Paradigm Project...** from the toolbar.
2. Select the file path of the .vpp file to import in file chooser.
3. Click **Open**.

> **NOTE:** All changes made in project will be overwritten by imported project.
> For example, if class Foo is renamed to Bar. By importing a project exported before renaming class, Bar will be renamed to Foo.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import Microsoft Excel

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. This chapter focuses on the interoperability with Microsoft Excel file.

Exporting to Microsoft Excel

Shows you how to export project data to Excel file.

Importing Microsoft Excel file

Shows you how to import Excel file to an opening project. (The Excel must be exported from Visual Paradigm through the export feature)

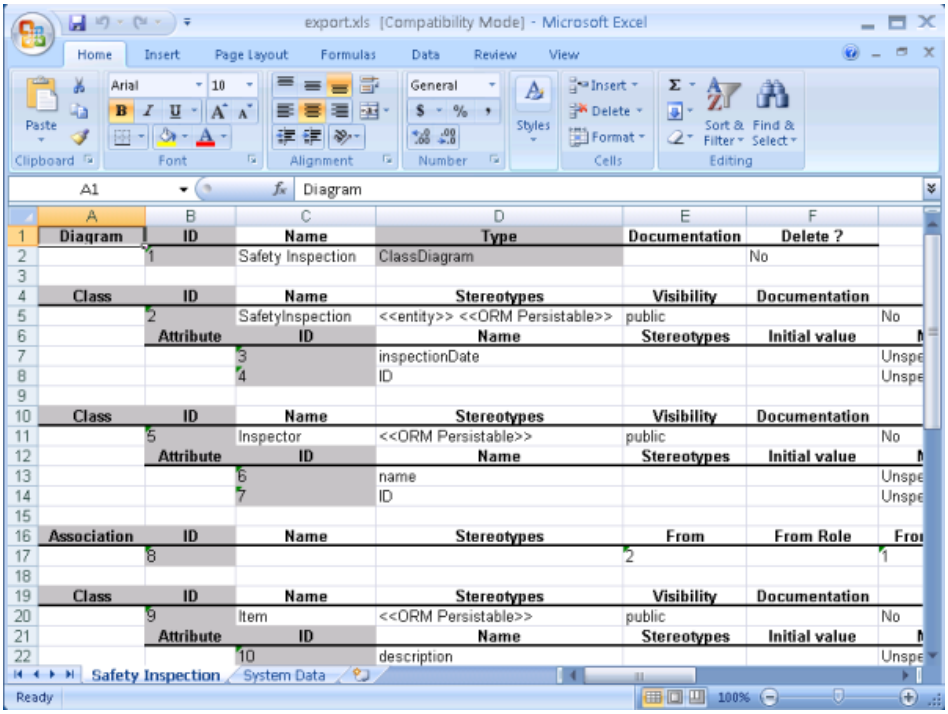
Excel modification guidelines

Non-desired changes made in Excel may damage its structure and cause import failed to work. This page outlines some of the points you need to pay attention to when modifying an exported Excel file.

Exporting diagrams to Microsoft Excel format

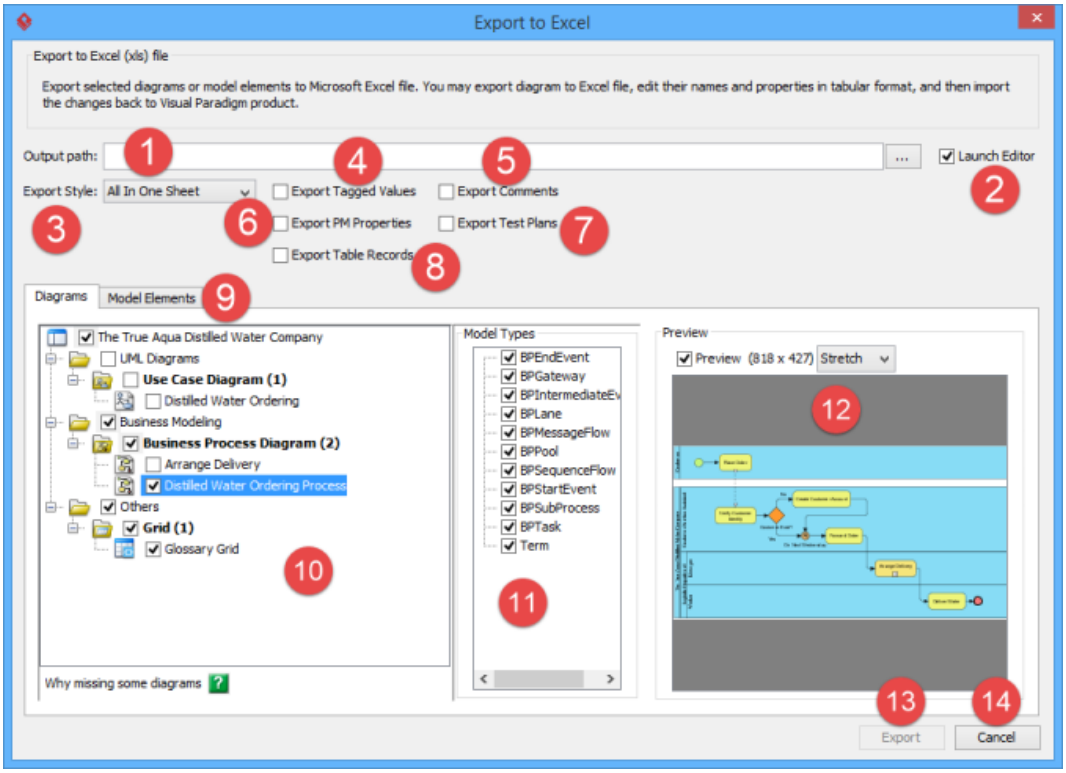
Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. Visual Paradigm supports interoperability with Microsoft Excel file. You can export project data to an excel file, make changes in exported Excel's worksheets or further processing like to query data with formula and eventually, feed the changes back to Visual Paradigm. In this chapter, you will see how to export an Excel file from a project. To export Excel:

1. Select **Project > Export > Excel...** from the toolbar.
   This opens the **Export Excel** dialog box.
2. Specify the output path of Excel file.
3. Select the diagrams to export to Excel.
4. On the **Model Types** list at the right hand side, you can optionally select the type(s) of model elements to be exported.
5. Click **Export**. Upon finishing, you can visit the output destination specified to obtain the Excel file.



The exported Excel file

An overview of Export Excel dialog box



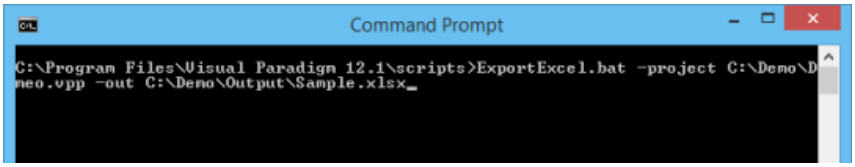
*Overview of the **Export Excel** dialog box*

No.	Field	Description
1	Output Path	The file path of the Excel file.
2	Launch Editor	Check to open the exported Excel file after exported.
3	Export Style	Determine how data will be presented in Excel. **Diagram Per Sheet** - Selected diagrams will be exported to separate sheets. **All in One Sheet** - Selected diagrams will be exported to a single sheet. **Model Type Per Sheet** - Selected model elements will be exported to separate sheets, grouped by type. **Raw** - Only one sheet will be generated with all elements listed in it. All the properties of selected elements will be listed as columns in which each row represents a model element.
4	Export Tagged Values	Tagged values can be added to model elements to specify domain specific properties. You can add tagged values in the specification dialog box of model element. Check **Export Tagged Values** if you want to export tagged values of model elements.
5	Export Comments	You can add your own comments to model elements in their specification dialog box. Check **Export Comments** if you want to export the comments to the Excel file.
6	Export PM Properties	PM properties is the short form of project management properties. You can set project management properties, such as version, phrase, author, in the specification dialog box of a model element. Check **Export PM Properties** if you want to export PM properties of model elements.
7	Export Test Plans	Test plan refers to the test plans that can be specified for Test Case elements in requirement diagram. Check **Export Test Plans** to export the information of test plan to a separate sheet in Excel file.
8	Export Table Records	You can specify the table records of entities created in ERDs. Check **Export Table Records** to export these table records.
9	Model Elements	Select this tab and check the model elements you want to export to Excel.
10	Diagrams	All diagrams in the opening project are listed here. Select the diagrams to export to Excel.
11	Model Types	Check or uncheck model types to include and exclude the types of model elements in exporting.
12	Model Types	When you have updated the diagram selection in **Diagrams** list, the **Model Types** list will be updated to list the types of diagram elements that appear in the chosen diagrams. By default, all diagram element types are checked, meaning that diagram elements in those types will be exported to Excel. You can uncheck type(s) to ignore certain type(s) of diagram elements when exporting.
13	Export	Click **Export** to proceed with exporting Excel.
14	Cancel	Click **Cancel** to diacard exporting to Excel.

*Overview of the **Export Excel** dialog box*

Exporting diagrams to Microsoft Excel with command line interface

1. Start the command console.
2. In the console, change directory to the **scripts** folder and execute **ExportExcel** with parameters required.



Parameters for ExportExcel

This displays the usage of the command. Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The filepath of the Excel file to import	C:\Demo\Output\Sample.xlsx

Parameters for ExportExcel

Upon finishing, you can visit the output destination specified to obtain the Excel file.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

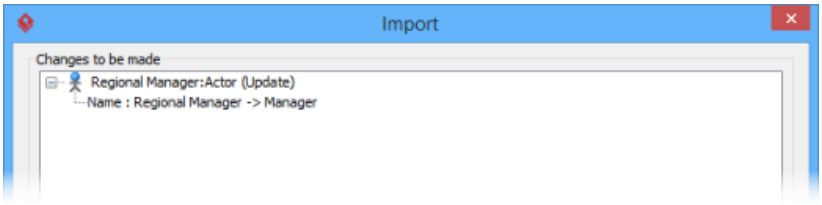
- Contact us if you need any help or have any suggestion

Importing Microsoft Excel file

You can import changes made externally in Excel file back to Visual Paradigm, to update the project data. In this chapter, you will see how to import an Excel exported before. Notice that Excel import is made in response to Excel export in Visual Paradigm. Only Excel exported from Visual Paradigm can be imported.

Importing an Excel file

1. Select **Project > Import > Excel...** from the toolbar.
2. In the file chooser, select the Excel file to import and click **Open** to confirm.
3. In the **Import** dialog box, you can preview the changes you have made in the previous Excel file. If you want to keep a record of changes, click on **Save Details...** at the bottom of dialog box to export the changes to an Excel file.



***Import** window*

4. Click **OK** button to proceed.

Importing Excel to project with command line interface

1. Start the command console.
2. In the console, change directory to the scripts folder and execute **ImportExcel** with the parameters required.

Parameters for ImportExcel

This displays the usage of the export command. Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the Excel file to import	C:\Demo\input\sample.xlsx

Parameters for ImportExcel

Upon finishing, the project file will be updated with the data presented in the XML file.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Excel modification guidelines

When you are going to modify Excel file exported from Visual Paradigm, make sure the changes you make are all valid. Making invalid changes will cause the file cannot be imported back to Visual Paradigm. In this chapter, we will go through some of the points that you need to pay attention to when editing.

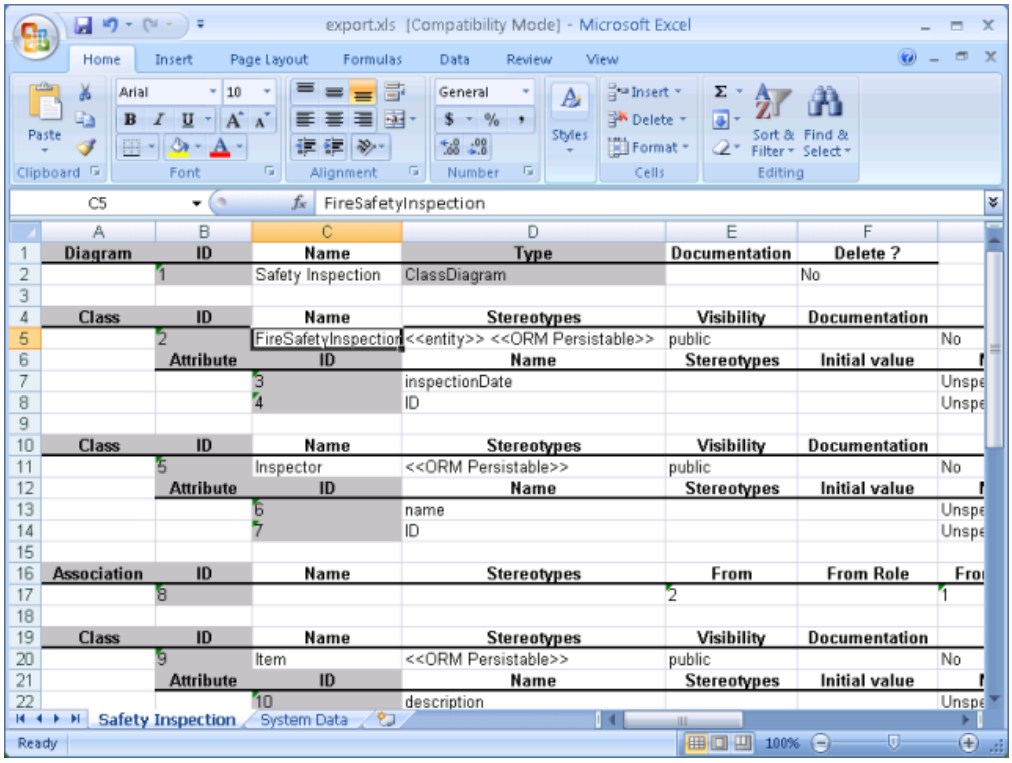
Caution

When you start to modify the Excel, pay attention to the following points to avoid having problems when importing the file back to project:

- Do NOT modify the gray cells
- Do NOT just delete a row for deleting a model element. Instead, change the value *No* to *Yes* under the **Delete?** column.
- Do NOT modify the System Data sheet

Renaming a model element

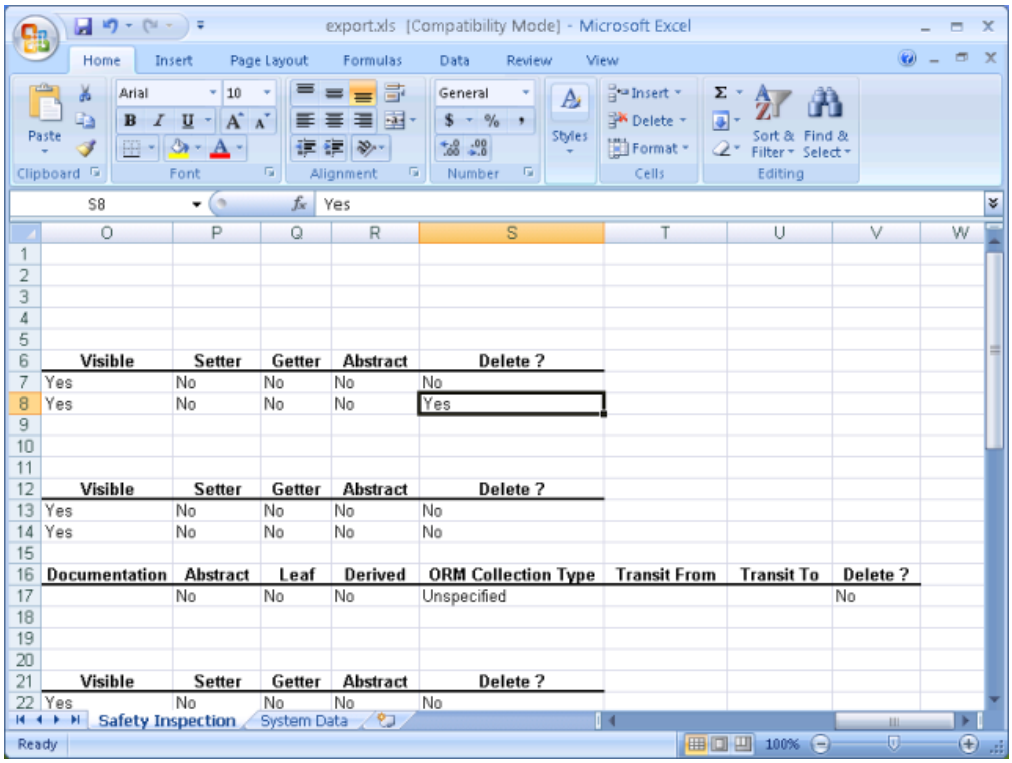
Double click on a cell and enter a new name.



Renaming class in Excel

Deleting a model element

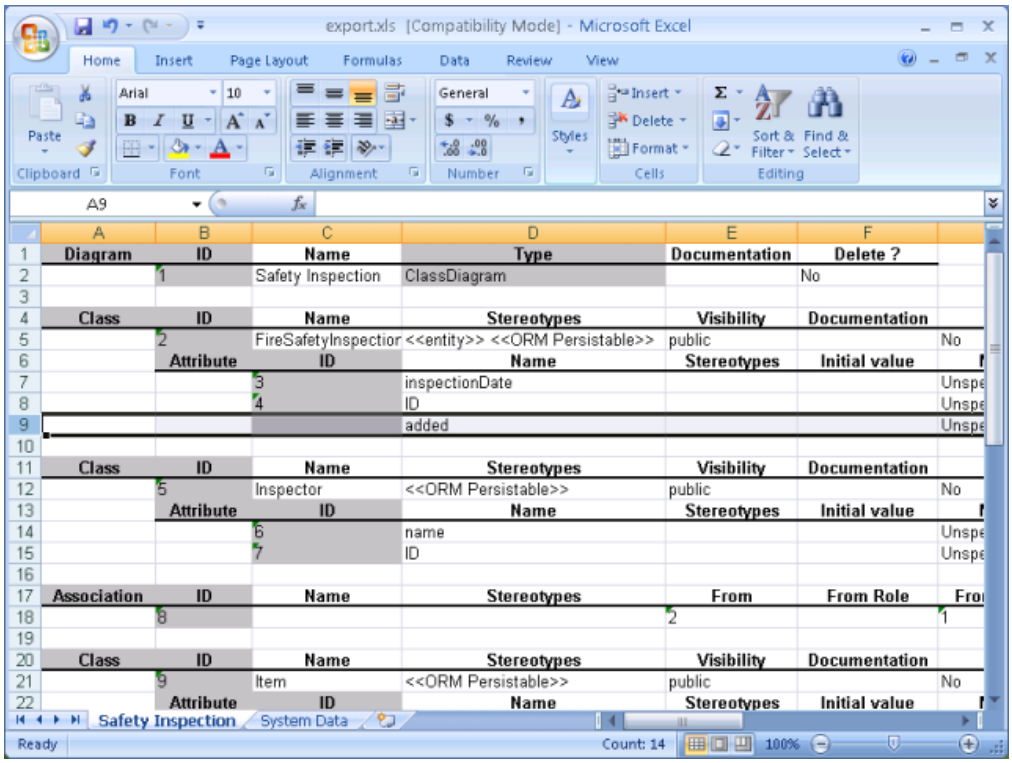
To delete a model element, change *No* to *Yes* under the **Delete?** column. Do NOT delete the row.



Deleting attribute in Excel

Adding a model element

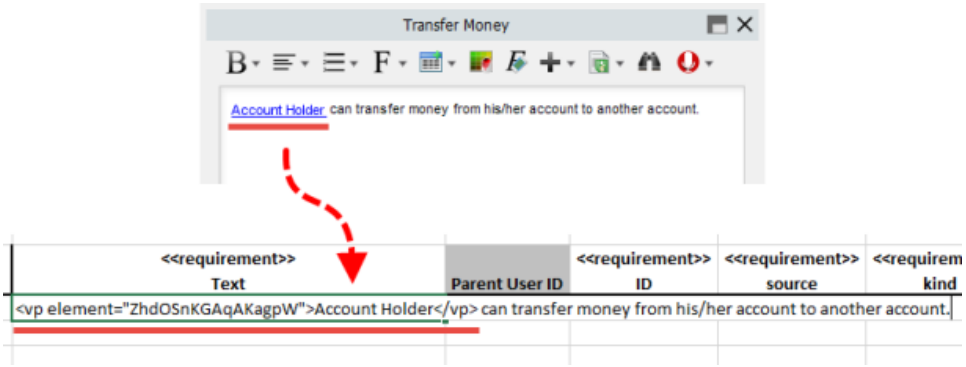
Suppose you want to add an attribute, select the last attribute row and insert a row in Excel, right under the last one. Then, start editing it. The gray cells can be left blank.



Adding attribute in Excel

Retaining element links

Properties like text and description (of requirement) may contain element links. In exported Excel, we will try to keep the links for you by converting it into a text like, formatted like **<vp element="${element=id}">link text</vp>**. To avoid damaging the link when importing the file back to Visual Paradigm, make sure **no** modification has been made to the link externally in Excel.



Appearance of element link in Excel

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import XMI

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. This chapter focuses on the interoperability with XMI file.

Exporting XML

Shows you how to export project data to XMI file.

Importing XML

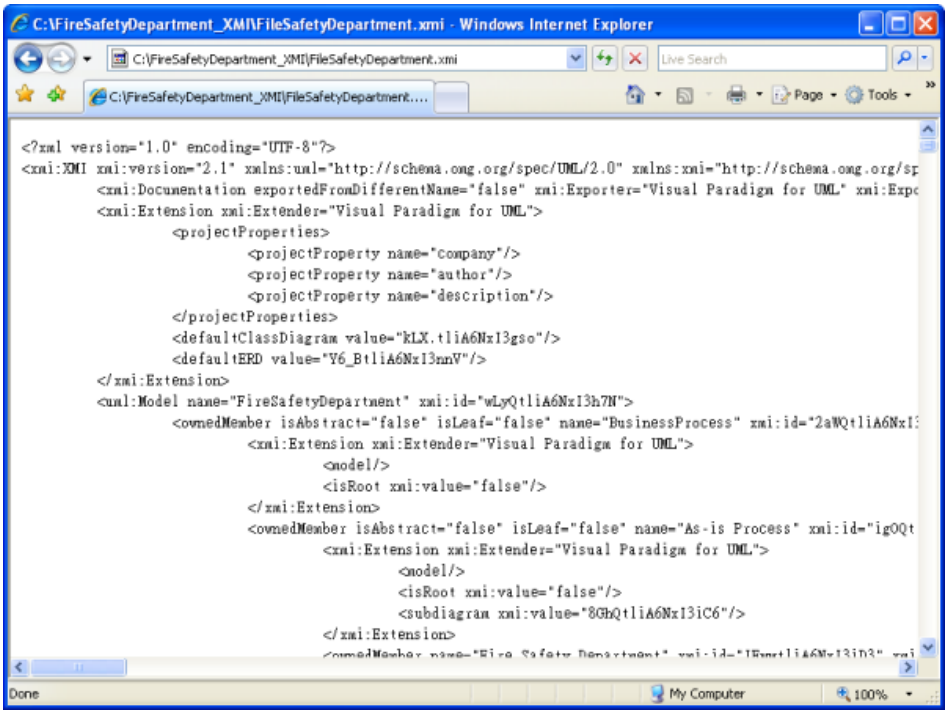
Shows you how to import XMI to an opening project.

Exporting XMI

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. Visual Paradigm supports interoperability with XMI file, a standard made for data exchange. You can export project data to an XMI, edit it externally with other softwares that accepts XMI. In this chapter, you will see how to export XMI file.

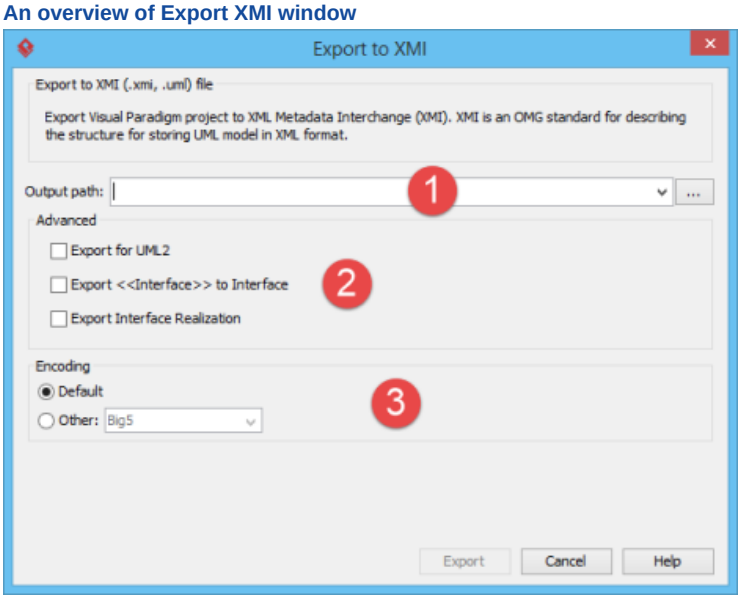
Exporting project to XMI

1. Select **Project > Export > XMI...** from the toolbar. This displays the **Export to XMI** window.
2. Specify the file path of the XMI file.
3. Configure the necessary export options and then click **OK** button to start exporting. When finished, you can visit the output destination specified to obtain the XMI.



Review exported XMI

An overview of Export XMI window



An overview of the Export to XMI window

No.	Name	Description
1	File path	The file path for the XMI file to export.
2	Advanced	• **Export for UML 2** - UML2 is an EMF-based implementation of the Unified Modeling Language (UML) 2.x OMG metamodel for the Eclipse platform. If you want to export an XMI that can be accepted by UML2 in Eclipse, check this option. By checking this option, the following options will appear in further.

- **Export Data Type to** - Determine whether to export data type to UML or Ecore primitive type
- **Export Java Annotation to EAnnotation** - Determine whether to export Java annotation to EAnnotation

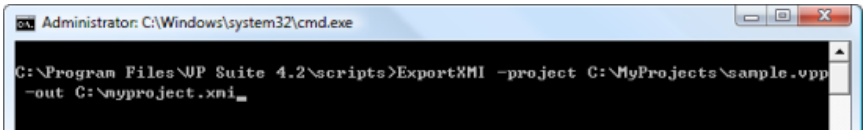
- **Export <<Interface>> to Interface** - Determine whether to export stereotype "interface" as stereotype or a segment of interface element.
- **Export Interface Realization** - Determine whether to keep exporting realization between interface and concrete class as realization or export it as interface realization.

3	Encoding	Select the encoding of XMI file.

*Description of **Export XMI** window*

Exporting diagrams to XMI with command line interface

1. Start the command console.
2. In the console, change directory to the scripts folder and execute **ExportXMI** with the parameters required.



Export XMI using command line interface

Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The filepath of XMI file	C:\Demo\Output\sample.xmi
-type [optional]	Version of XMI. Unless specified, the lastly generated version will be selected. Here are the possible options: • 2.1 • 2.1UML2	2.1
-encoding [optional]	Encoding of XMI file	UTF-8

Parameters for ExportXMI

Upon finishing, you can visit the output destination specified to obtain the XMI.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing XMI

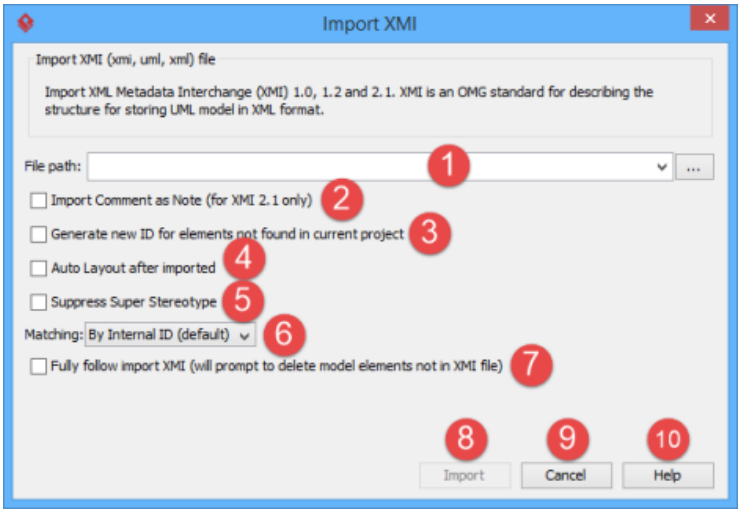
You can migrate your works done in another software to Visual Paradigm through XMI, provided that the software supports XMI. In this chapter, you will see how to import an XMI file.

Importing XMI to current project

1. Select **Project > Import > XMI...** from the toolbar. This displays the **Import XMI** window.
2. Specify the file path of the XMI to import and configure the import if necessary.
3. Click **OK**.

> **NOTE:** All changes made in project will be overwritten by data in XMI. For example, if class Foo is renamed to Bar. By importing an XMI exported before renaming class, Bar will be renamed to Foo.

An overview of Import XMI window



An overview of Import XMI window

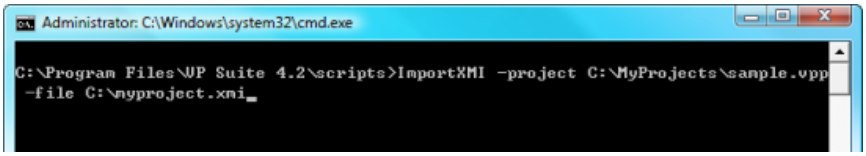
No.	Name	Description
1	File path	The file path of the XMI file to import.
2	Import Comment as Note (for XMI 2.1 only)	Determine whether to import comment as description or as note of model.
3	Generate new ID for elements not found in current project	You can enable this option to ensure the uniqueness of ID across projects.
4	Auto Layout after imported	Determine whether to run a layout on diagram after import. Note that running layout may takes time for a massive amount of diagram data.
5	Suppress Super Stereotype	Determine whether to ignore super stereotype when importing.
6	Matching	The importing of XMI is a procedure to merge the data in XMI into the opening project. The matching option is to determine how to match between data in XMI and the opening project during export and to perform changes accordingly.
7	Fully follow import XMI (will prompt to delete model elements not in XMI file)	By checking this option, the project will fully follow the imported XMI. You will be prompted to delete any data that does not exist in XMI.
8	Import	Click here to import the XMI.

9	Cancel	Click here to cancel importing.
10	Help	Click here to read the Help contents of import XMI.

*Description of **Import XMI** dialog box*

Importing XMI to project with command line interface

1. Start the command console.
2. In the console, change directory to the scripts folder and execute **ImportXMI** with the parameters required.



Import XMI using command line interface

Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the XMI file to import	C:\Demo\input\sample.xmi

Parameters for ImportXMI

Upon finishing, the project file will be updated with the data presented in the XMI file.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export and import BPMN 2.0

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. This chapter focuses on the interoperability with BPMN 2.0 file.

Exporting BPMN 2.0

Shows you how to export project data to BPMN 2.0 file.

Importing BPMN 2.0

Shows you how to import BPMN 2.0 to an opening project.

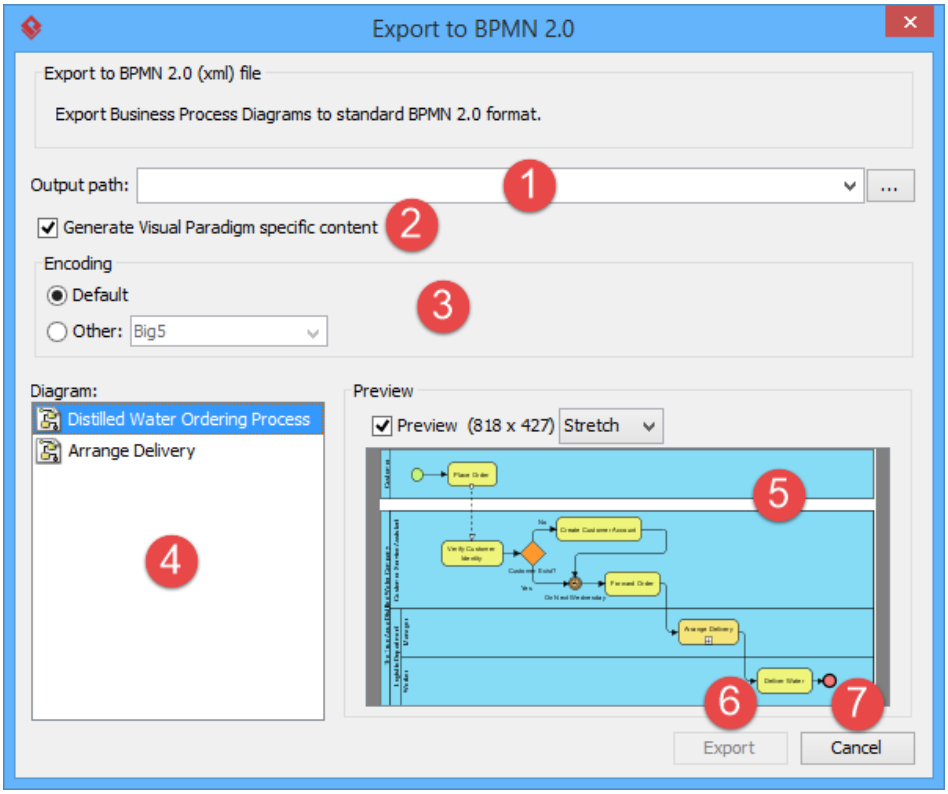
Exporting BPMN 2.0

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. Visual Paradigm supports interoperability with BPMN 2.0 XML. You can export project data, edit it externally with other softwares that accepts BPMN 2.0 XML. In this chapter, you will see how to export BPMN file.

Exporting project to BPMN

1. Select **Project > Export > BPMN 2.0...** from the toolbar.
   This displays the **Export BPMN** dialog box.
2. Specify the file path of the BPMN file.
3. Click **OK** button to start exporting. Upon finishing, you can visit the output destination specified to obtain the BPMN.

An overview of Export BPMN dialog box



*An overview of the **Export BPMN** dialog box*

No.	Name	Description
1	File path	The location where you want to save the file.
2	Generate Visual Paradigm specific content	Project specific content refers to contents that do not belong to BPMN. For example, project management properties.
3	Encoding	Encoding of the BPMN file.
4	Diagram	A list of diagram of your project. Select the diagrams to export to BPMN.
5	Preview	By checking the selected diagram and **Show preview**, it will be shown in preview window.
6	Export	Click to export BPMN file.
7	Cancel	Click to cancel exporting.

*Description of **Export BPMN** dialog box*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing BPMN 2.0

You can migrate your works done in another software to Visual Paradigm through BPMN XML 2.0, provided that the software supports BPMN 2.0. In this chapter, you will see how to import an BPMN 2.0 file.

Importing BPMN to current project

1. Select **Project > Import > BPMN 2.0...** from the toolbar.
2. Specify the file path of the XML to import.
3. Click **OK**.

> **NOTE:** All changes made in project will be overwritten by data in XML. For example, if task Foo is renamed to Bar. By importing an XML exported before renaming task, Bar will be renamed to Foo.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Visio drawing

You can import your previous work drawn in Visio to VP-UML through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

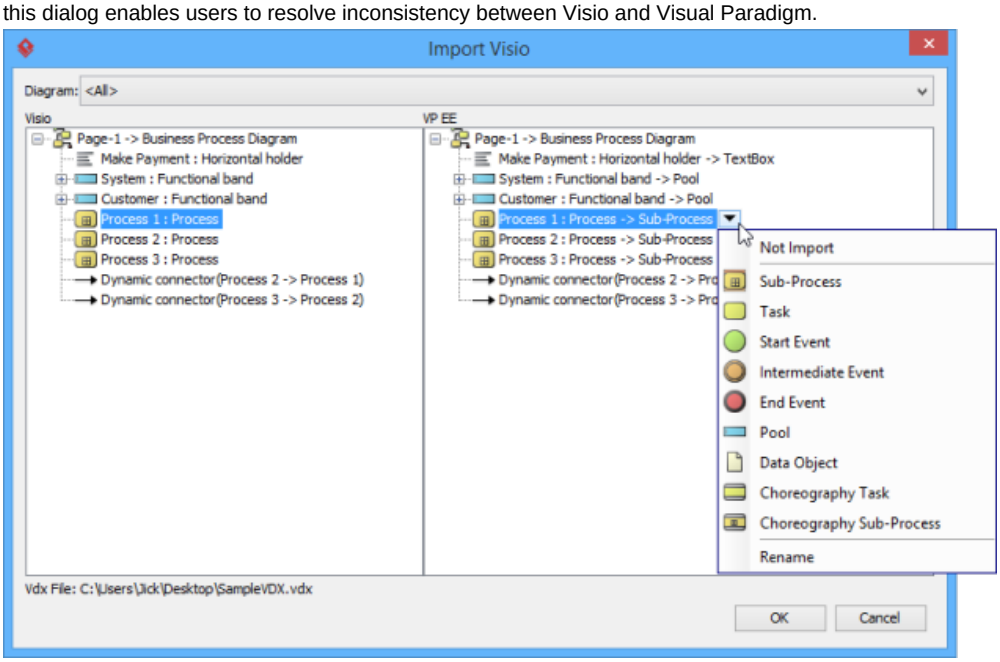
Importing Visio drawing

Outlines the steps involved in importing an Visio drawing.

Importing Visio drawing

You may have drew diagrams in Visio. You can now import your previous work through the import feature.

1. Save your Visio drawing as a .vdx or .vsdx file .
2. To import a Visio drawing into Visual Paradigm, select **Project > Import > Visio...** in the toolbar of Visual Paradigm.
3. Specify the file path of the Visio drawing.
4. Click **OK** to start importing. This popup another **Import Visio** dialog box. As the model structure is different among Visio and Visual Paradigm, this dialog enables users to resolve inconsistency between Visio and Visual Paradigm.



The Import Visio dialog box

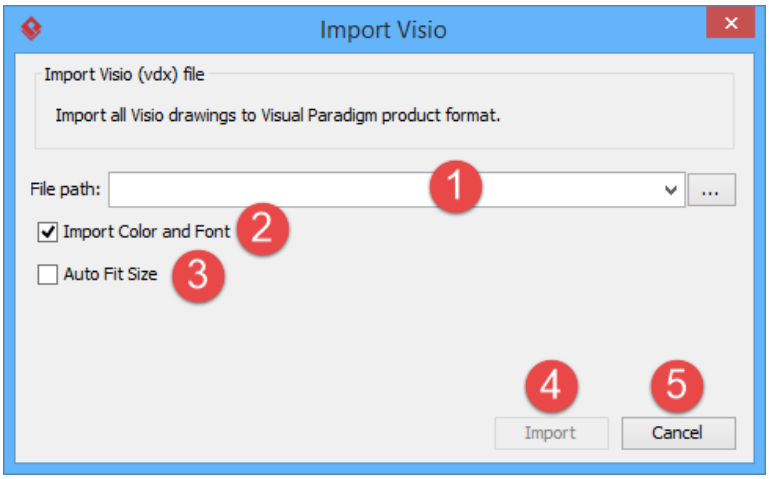
The left hand side of the dialog box represents the structure of Visio drawing, while the right hand side represents the expected outcome in Visual Paradigm through importing. Users can perform the following actions in this dialog box.

Action	Description and steps
Not to import a shape	Click on the button beside the shape node and select **Not Import** in the popup menu.
Rename a shape when importing	Click on the button beside the shape node and select **Rename** in the popup menu. Then, enter the new name of shape and press the **Enter** key to confirm renaming/.
Reset the shape to another type	Click on the button beside the shape node and select an appropriate shape type to reset to.

Description of available import options on individual shape

5. Click **OK** when the import is configured. The drawings can then be accessible in the **Diagram Navigator**. You can then double click on the diagram node to open the diagram.

An overview of Import Visio dialog box



*An overview of **Import Visio** dialog box*

No.	Name	Description

1	File path	The path of Visio .vdx file to be imported.
2	Import Color and Font	By selecting this option, colors and fonts of the shapes to be imported will remain unchanged. Otherwise, Visual Paradigm's default settings will be applied.
3	Auto Fit Size	By selecting this option, shapes' size will be optimized to their minimum possible size. Otherwise, the original size of the imported shapes will remain unchanged.
4	OK	Click to import.
5	Cancel	Click to cancel importing.

*Description of **Import Visio** dialog box*

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Rational Rose model

You can import your previous work drawn in Rose to Visual Paradigm through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

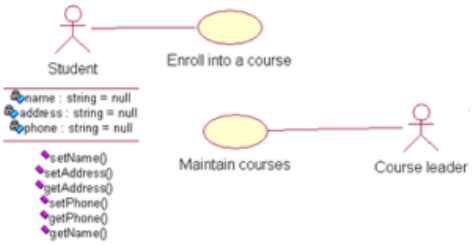
Importing Rational Rose Model

Outlines the steps involved in importing a Rational Rose model file.

Importing Rational Rose model

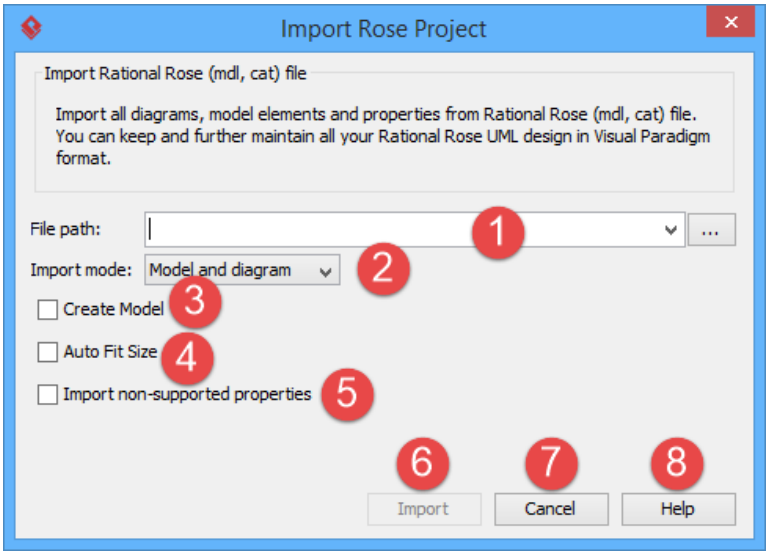
Rational Rose is one of the most widely used UML CASE Tool in the 90's. Visual Paradigm supports the importing of Rational Rose model. With this, users can import legacy design made in Rose into Visual Paradigm, with all the model data as well as formatting retained.

1. Save your work in Rose.



A Rose drawing

2. To import a Rose model into Visual Paradigm, select **Project > Import > Rose Project...** in the toolbar of Visual Paradigm.
3. Specify the file path of the Rose model.



Specifying Rose model path

No.	Name	Description
1	File path	The path of Rose .mdl file to be imported.
2	Import mode	You can choose the mode to be imported. **Model only**: By selecting this option, only the model elements (e.g. Actor, Use Case, Class, etc.) will be imported. NO diagrams will be imported. **Model and diagram**: By selecting this option, both model elements and diagrams will be imported.
3	Create Model	Create model for placing the imported data.
4	Auto Fit Size	Fit the size of the imported shapes.
5	Import non-supported properties	Convert non-supported properties into tagged values.
6	OK	Click to import.
7	Cancel	Click to cancel importing.
8	Help	Click to obtain more information from the help system.

Description of import rose properties

4. Click **OK** to start importing.
5. When import is completed, select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumnail to open the diagram. .

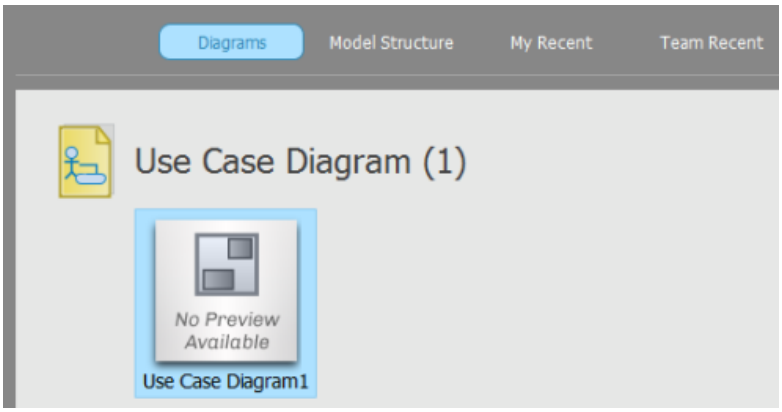


Diagram view lists the imported diagram(s)

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Rational Software Architect File

You can import your previous work drawn in Rational Software Architect to Visual Paradigm through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

Importing Rational Software Architect EMX

Outlines the steps involved in importing a .emx file.

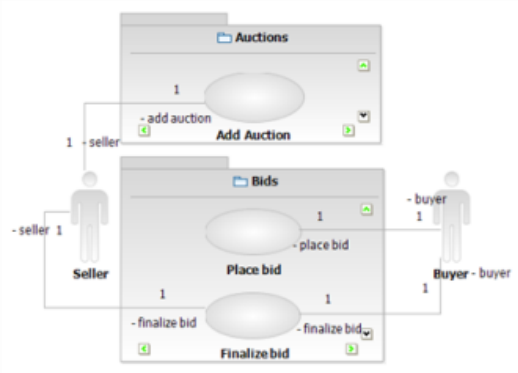
Importing Rational Software Architect DNX

Outlines the steps involved in importing a .dnx file.

Importing Rational Software Architect EMX

Rational Software Architect (RSA) is a modeling and development environment, which leverages UML for architectural design for C++ and Java 2 Enterprise Edition (Java2EE) applications and web services. Import of the RSA file, i.e. the .emx file, is supported in Visual Paradigm, so that users can simply migrate the work from RSA to Visual Paradigm and also perform further modeling on the imported models in Visual Paradigm.

1. Save your work in Rational.



A Rational drawing

2. To import a Rational model into Visual Paradigm, select **Project > Import > Rational Model...** in the toolbar of Visual Paradigm.
3. In the **Import Rational Software Architect UML Model** dialog box, specify the file path of the .emx file and click **OK**.
4. Select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumbnail to open the diagram.

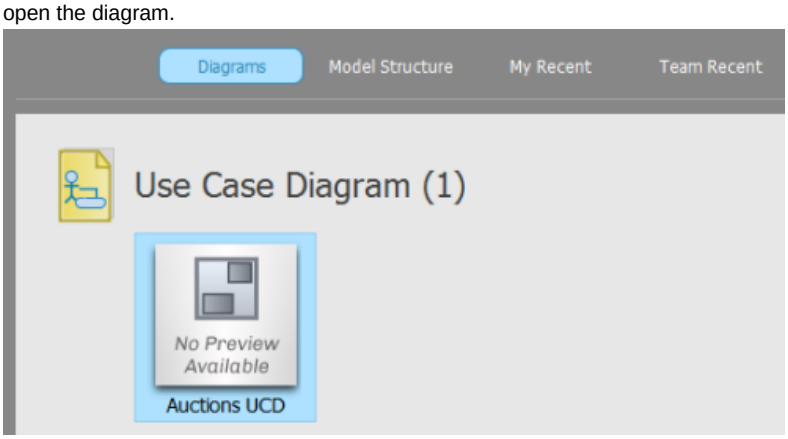


Diagram view lists the imported diagram(s)

Related Resources

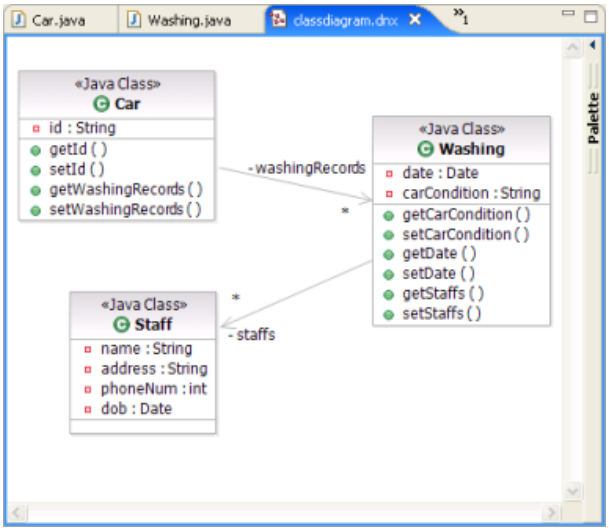
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Rational Software Architect DNX

Visual Paradigm supports importing drawing drew in Rational Software Architect with a .dnx extension. By importing a drawing, all diagrams, shapes and model information will be imported.

1. Save the drawing in Rational Software Architect.



A Rational drawing

2. To import the drawing into Visual Paradigm, select **Project > Import > Rational DNX...** in the toolbar of Visual Paradigm.
3. In the **Import Rational Diagram DNX** dialog box, specify the file path of the .dnx file and click **OK**.
4. After importing is completed, select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumnail to open the diagram.

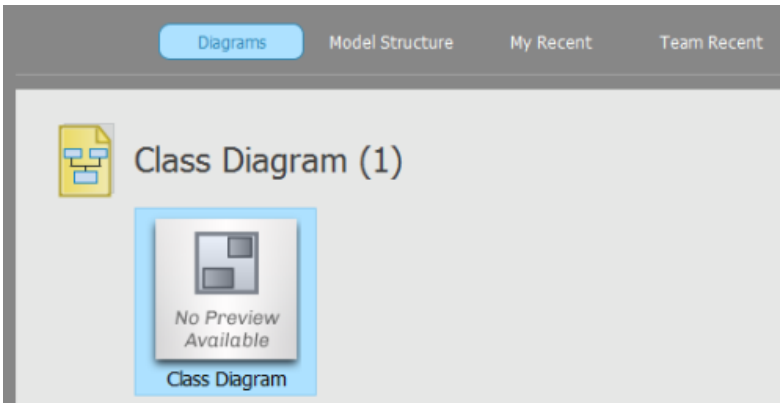


Diagram view lists the imported diagram(s)

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing ERwin data modeler project

You can import your previous work drawn in ERwin Data Modeler to Visual Paradigm through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

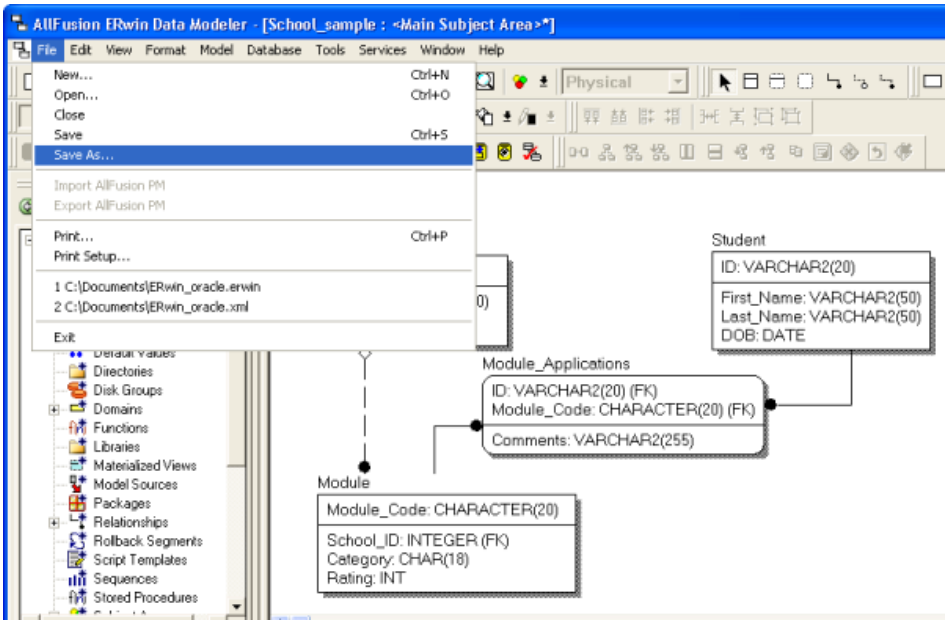
Importing ERwin data modeler project

Outlines the steps involved in importing an ERwin project.

Importing ERwin data modeler project

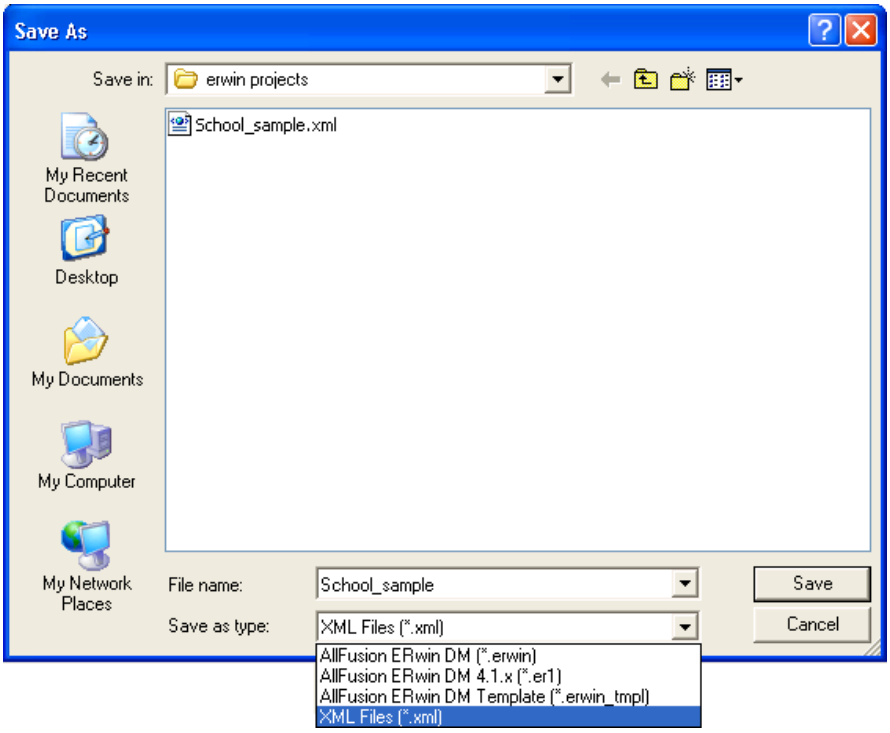
AllFusion ERwin Data Modeler is a popular tool for data modeling. You can import ERwin diagrams and entity models into Visual Paradigm with all properties preserved.

1. Here is a ERwin Data Modeler Project. In order to allow Visual Paradigm to import it, you need to save it as an XML file. Select **File > Save As...** from the menu.



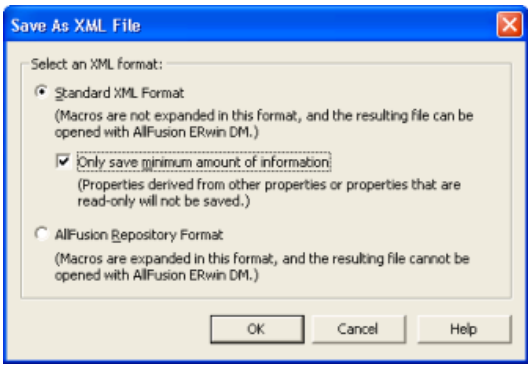
To save an ERwin Data Modeler project as XML

2. Select **XML Files (*.xml)** in **Save as type** and enter the file name in the **Save As** dialog box.



Save the ERwin Data Modeler project as XML

3. Click **Save**. This popup the **Save as XML File** dialog box.
4. Keep using the default settings **Standard XML Format** and **Only save minimum amount of information**.



Exporting the XML in standard XML format

5. Click **OK** to confirm. This saves an XML file that can be used for importing into Visual Paradigm.
6. To import an ERwin Data Modeler project into Visual Paradigm, select **Project > Import > ERwin Project(XML)...** in the toolbar of Visual Paradigm.
7. Specify the file path of the XML file.
8. Click **OK** to start importing. When import is completed, the **Open Imported Entity Relationship Diagram(s)** dialog box will appear.
9. Select the diagram(s) to open and click **Open** to open them. The drawings will then be opened.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Telelogic Rhapsody and System Architect project file

You can import your previous work drawn in Telelogic Rhapsody or Telelogic System Architect to Visual Paradigm through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

Importing Telelogic Rhapsody project

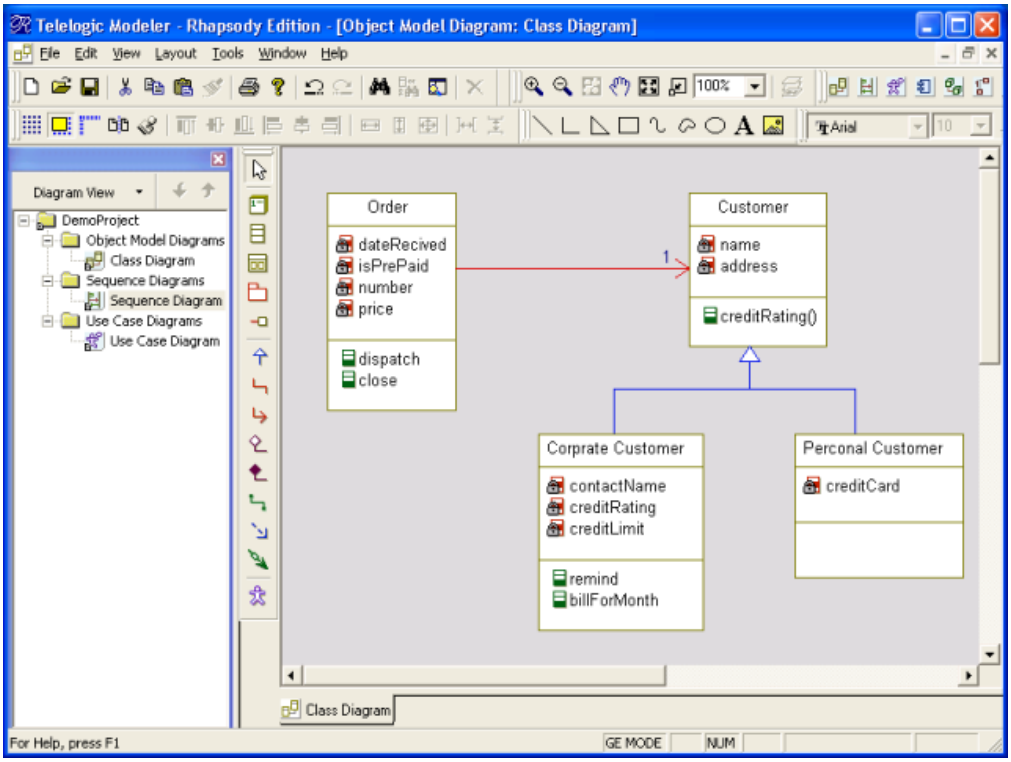
Outlines the steps involved in importing a Telelogic Rhapsody file.

Importing Telelogic System Architect

Outlines the steps involved in importing a Telelogic System Architect file.

Importing Rational Rhapsody project

1. Save your Rational Rhapsody Project.



A drawing in Telelogic Modeler

2. To import a Rational Phapsody project into Visual Paradigm, select **Project > Import > Rational Rhapsody Project ...** in the toolbar of Visual Paradigm.
3. Specify the file path of the Rational Rhapsody Project in the pop-up **Import Rhapsody** dialog box.
4. Click **OK** button when the import is configured. Select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumnail to open the diagram.

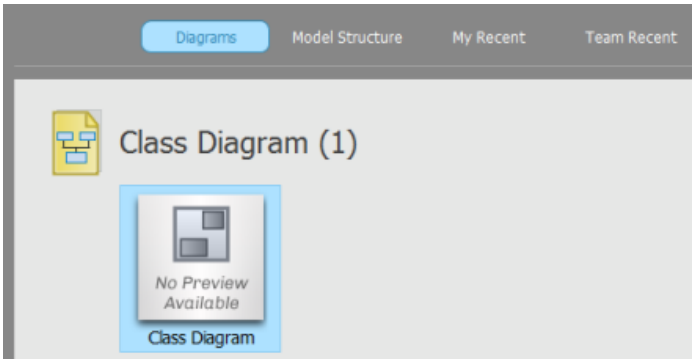


Diagram view lists the imported diagram(s)

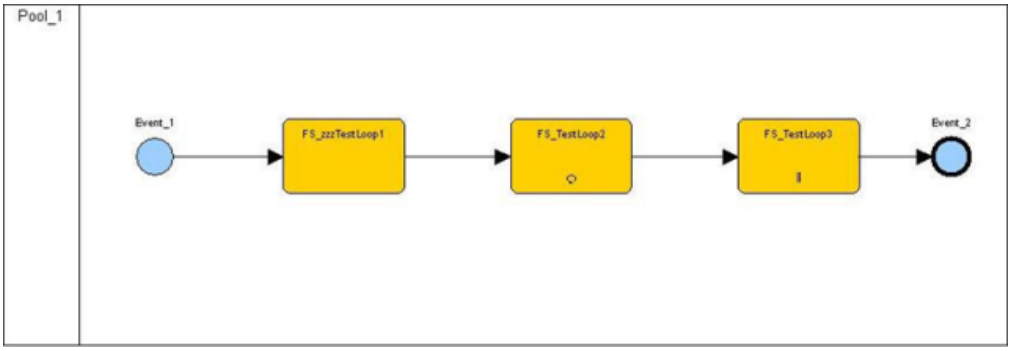
Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Rational System Architect

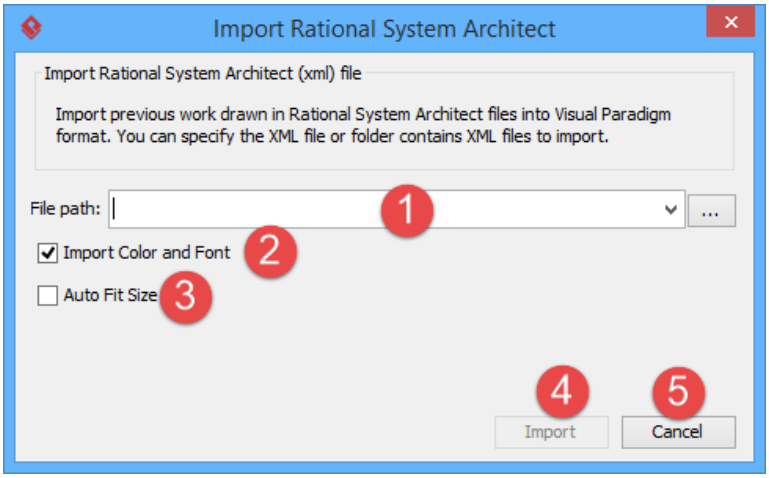
1. Save your Telelogic System Architect drawing:



A Rational System Architect drawing

2. To import a Rational System Architect project into Visual Paradigm, select **Project > Import > Rational System Architect ...** in the toolbar of Visual Paradigm.

3. Specify the file path of the Rational System Architect project.



Specifying Rational System Architect path

No.	Name	Description
1	File path	The path of Rational System Architect .xml file to be imported.
2	Import Color and Font	By selecting this option, colors and fonts of the shapes to be imported will remain unchanged. Otherwise, Visual Paradigm's default settings will be applied.
3	Auto Fit Size	By selecting this option, shapes' size will be optimized to their possible minimum size. Otherwise, the original size of the imported shapes will remain unchanged.
4	OK	Click **OK** to proceed with importing Rational System Architect.
5	Cancel	Click **Cancel** to discard importing XML.

*Description of **Import Rational System Architect** dialog box*

4. Click **OK** to start importing. When import is completed, the message pane will popup with a notification appear in it. Select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumnail to open the diagram.

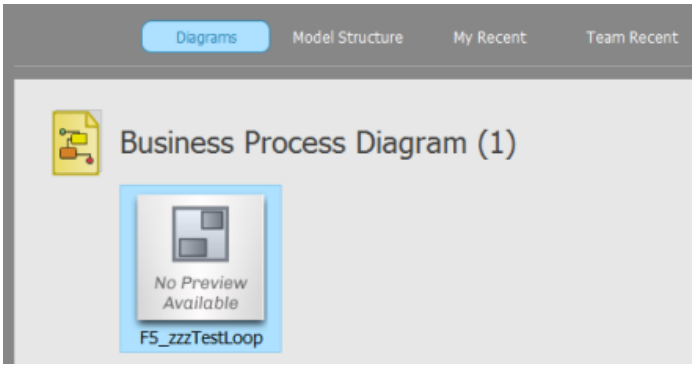


Diagram view lists the imported diagram(s)

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing NetBeans 6.x UML diagrams

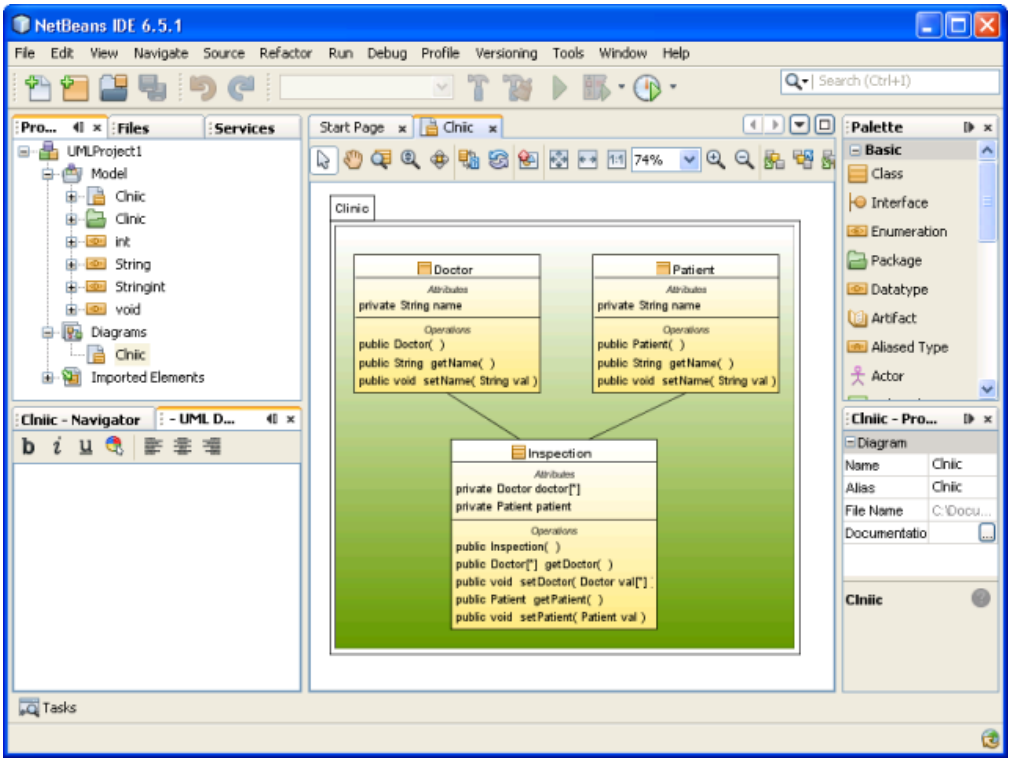
You can import the UML diagrams you previously drawn in NetBeans 6.x to Visual Paradigm through the import feature, and continue modeling in VP-UML. This chapter shows you how to do.

Importing NetBeans 6.x UML Diagrams into Agilian

Outlines the steps involved in importing NetBeans 6.x UML diagrams.

Importing NetBeans 6.x UML diagrams

1. Here is a NetBeans UML Diagram:

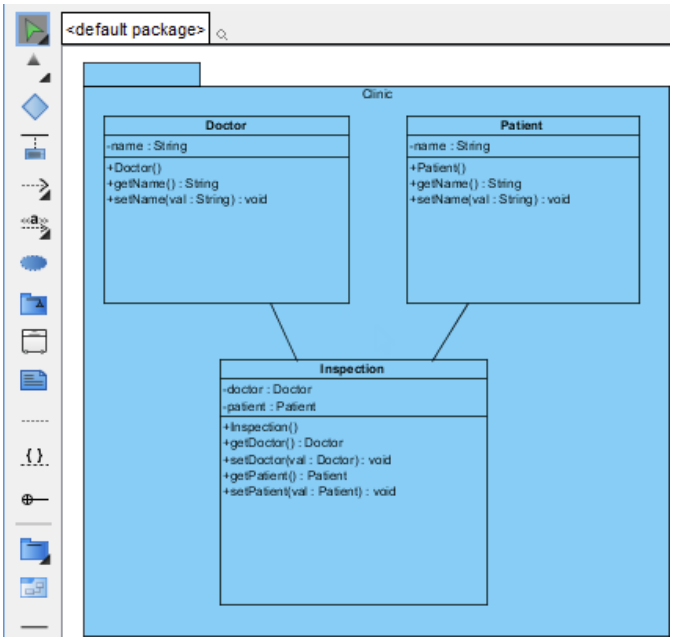


A Class Diagram drew in NetBeans

To import a NetBeans UML project into Visual Paradigm, select **Project > Import > NetBeans UML Project...** in the toolbar of Visual Paradigm.

2. Specify the file path of the NetBeans Java project folder.

3. Click **OK** to start importing. When import is completed, the message pane will pop up with a notification. The drawings can then be accessible in the **Diagram Navigator**. You can then double click on the diagram node to open the diagram.



A class diagram imported from NetBeans UML Project

NOTE: Due to different ways in presenting diagrams in Visual Paradigm and NetBeans, the imported shapes may be bigger than normal.
To fit a shape's size, move the mouse cover over it and press on the resource icon at the bottom right of shape.
To fit size for all shapes, right click on the diagram background and select **Diagram Content > Auto Fit Shapes Size** in the popup menu.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Importing Bizagi

You can import your previous work drawn in Bizagi to Visual Paradigm through the import feature, and continue modeling in Visual Paradigm. This chapter shows you how to do.

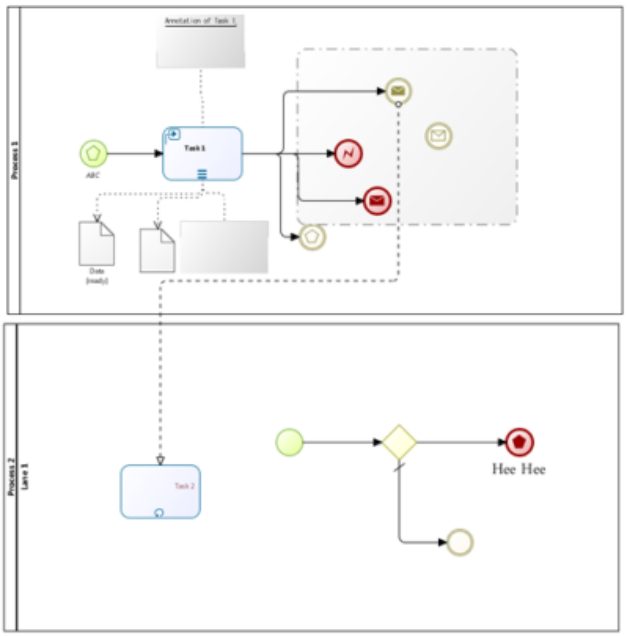
Importing Bizagi

Outlines the steps involved in importing a Bizagi project.

Importing Bizagi

Bizagi is one of the BPM softwares in the market. Since Visual Paradigm is compartible with Bizagi, you can import its .bpm file in Visual Paradigm. Importing Bizagi is as simple as migrating the work from Bizagi to Visual Paradigm. You can perform further modeling on the imported models in Visual Paradigm when necessary.

1. Save your Bizagi drawing.



A Bizagi drawing

2. To import a Bizagi project into Visual Paradigm, select **Project > Import > Bizagi ...** from the toolbar.
3. Specify the file path of the Bizagi project.
4. Click **OK** to start importing.
5. It will notify you that your Bizagi drawing is imported successfully in **Message** pane.
6. Select **View > Project Browser** in the toolbar. The **Diagrams** view lists the imported diagram(s). You can then double click on the thumbnail to open the diagram.

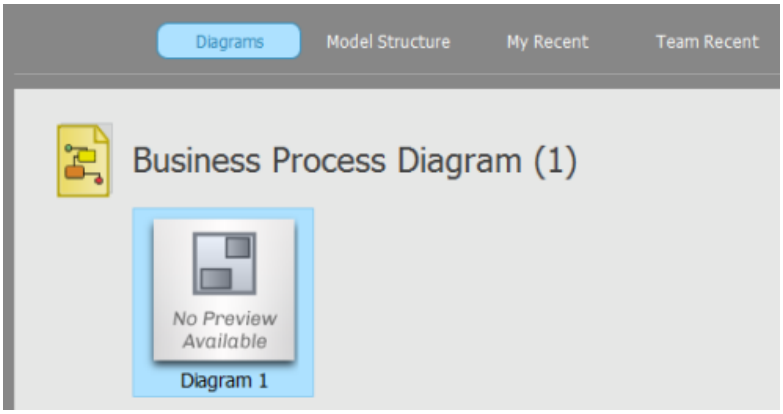


Diagram view lists the imported diagram(s)

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Export diagram to various graphic formats

You can export the opening diagram to image file. There are three ways of exporting. This chapter will shows you the instruction as well as some of the configuration of export, like how to slice diagram into parts.

Exporting active diagram as image

Shows you how to export the opening diagram to image.

Exporting multiple diagrams as images

Shows you how to export selected diagrams to image, as well as the steps of slicing image into parts.

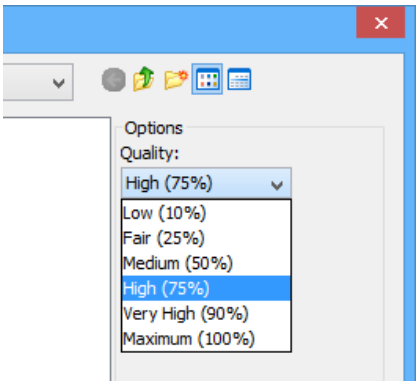
Exporting portion of diagram as image

You can export part of a drawing (i.e. some shapes) as an image. This page shows you how to do.

Exporting active diagram as image

You can export the opening diagram to image file. To export the active diagram as an image file:

1. Select **Project > Export > Active Diagram as Image...** from toolbar.
2. In the **Save** window, set the image quality. The higher the quality, the clearer the image, the larger the image size.



Set image quality

3. Select the image format at the bottom of window.

> **NOTE:** There are two options for exporting as PNG files - with and without background:
> - With background: export diagram's background color.
> - Without background: ignore the background color by exporting transparent background.

> **NOTE:** You can export diagrams to native PDF format. Since the exported PDF is of a small size, it can save a lot of space. Also, because the diagram in PDF is a vector, it is scalable. There are two export options:
> - PDF(diagram per page): selected diagrams will be exported to the same PDF file. Each diagram will occupy one page.
> - PDF(diagram per file): each diagram will be exported in one new PDF file.

4. Specify the filename.
5. Click **Save** to export.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

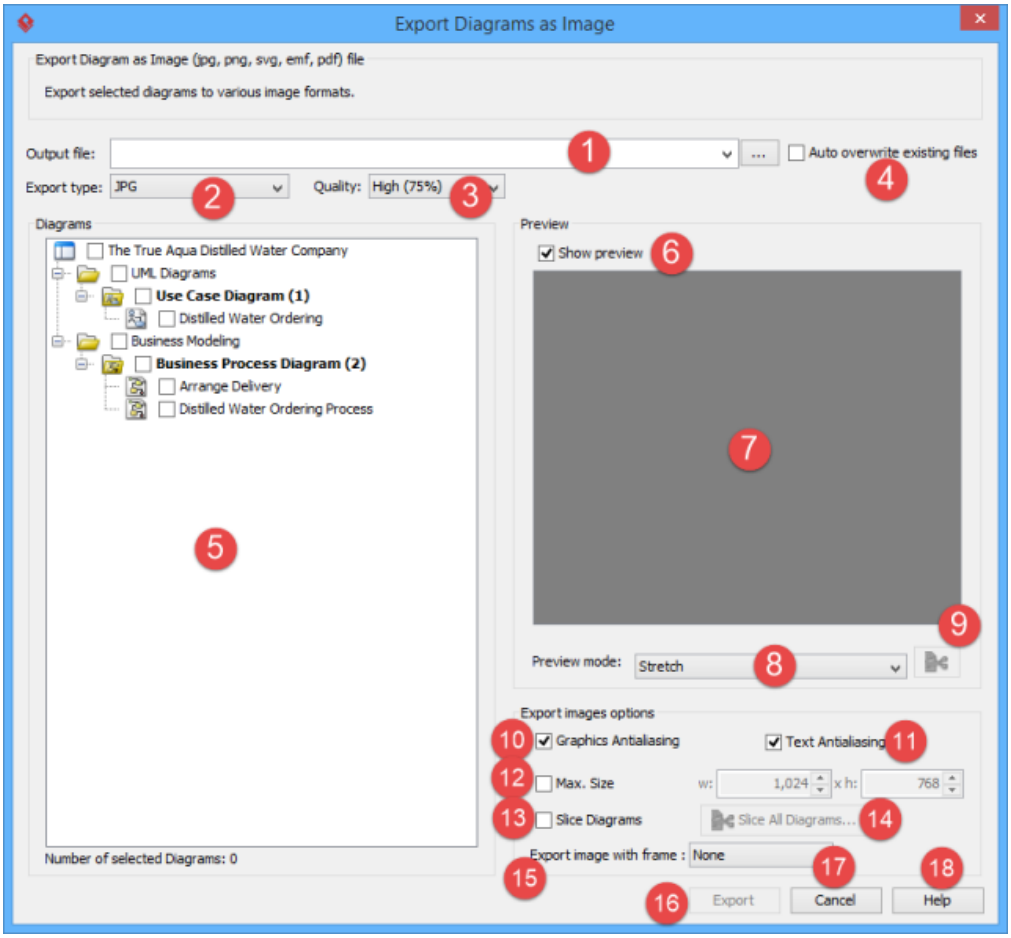
Exporting multiple diagrams as images

You can export diagrams in your project to image files. You can specify not only the quality and format (e.g. JPG, PNG, SVG, EMF, PDF) of images but also to slice diagram into pieces for easier insertion into documents.

Export diagrams

1. Select **Project > Export > Diagrams as Image...** from toolbar.
2. In the **Export Diagrams as Image** window, select the diagram(s) to export.
3. Specify the output destination for storing the image files.
4. Click **Export** button to export the diagrams.

An overview of Export Diagrams as Image



*An overview of **Export Diagrams as Image** window*

No.	Name	Description
1	Output destination	The **Output destination** is the directory where all the exported images are saved to. You can enter the path in the text field directly or you can click on the **...** button to browse for the directory.
2	Export type	To select the image format of the exported image, click on the pull-down box beside the **Export type** field and select the format you want to use. There are two options for exporting as PNG files - with and without background. With background will export diagram's background color. Without diagram will ignore the background color by exporting transparent background. You can export Visual Paradigm diagram to native PDF format. Since the exported PDF is of a small size, it can save a lot of space. Also, because the diagram in PDF is a vector, it is scalable. There are two different options when you export. For PDF (diagram per page), all the diagrams selected will be exported in the same PDF file. Each diagram will occupy one page. For PDF (diagram per file), each diagram selected will be exported in one new PDF file.
3	Quality	The quality of image. By applying a higher quality, the images will be clearer but larger in file size. By applying a lower quality, the images will look more blur but smaller in file size.
4	Auto overwrite existing files	You can check the **Auto overwrite existing files** checkbox to allow overwriting of files in the export process.
5	Diagrams	The **Diagrams** pane shows the diagrams in the current project. Check the checkbox beside the diagram you want to export. The number of selected diagrams is displayed at the bottom of the

		Diagram pane. The Preview pane also allows you to preview the exported image of the selected diagram.
6	Show preview	Check or uncheck to enable or disable the preview.
7	Preview	The **Preview** pane shows the preview of the exported image of the selected diagram in the Diagrams pane.
8	Preview mode	Select the size of the preview image by selecting from the pull-down box beside the **Preview mode** field. Selecting **Stretch** will show the image in scaled size that fits to the preview area, while selecting **Real size** will show the image in its actual size.
9	Diagram slicer	Click to configure how diagram is sliced into pieces. This is enabled only when the check box for **Slice Diagrams** (for slicing all diagrams) is unchecked. For details about slicing diagrams, please refer to the following section.
10	Graphics Anti-aliasing	Anti-aliasing is a method which handles the staircase pixels of slanted lines and curves to make them look smoother. You can apply anti-aliasing to the exported images. To apply anti-aliasing to graphics, check the **Graphics Anti-aliasing** checkbox .
11	Text Anti-aliasing	Anti-aliasing is a method which handles the staircase pixels of slanted lines and curves to make them look smoother. You can apply anti-aliasing to the exported images. To apply anti-aliasing to text, check the **Text Anti-aliasing** checkbox.
12	Max. Size	Maximum size of exported images. If the diagram size is larger than the maximum size, it will be resized.
13	Slice Diagrams	Enable it to slice all diagrams into pieces to obtain multiple image files for a single diagram. For details, please refer to the following section.
14	Slice all diagrams	Click to configure slice settings on all diagrams.
15	Export image with frame	A frame is a border that prints around a diagram. By selecting **None**, frame won't be printed. By selecting **Export with frame,** a frame will be added to exported images, making the diagram name show at the top left of diagram. By selecting **Export with border**, a black and thin border will be added to exported images.
16	Export	Click to proceed with exporting.
17	Cancel	Click to close the exporter without exporting diagrams.
18	Help	Click to show the help contents.

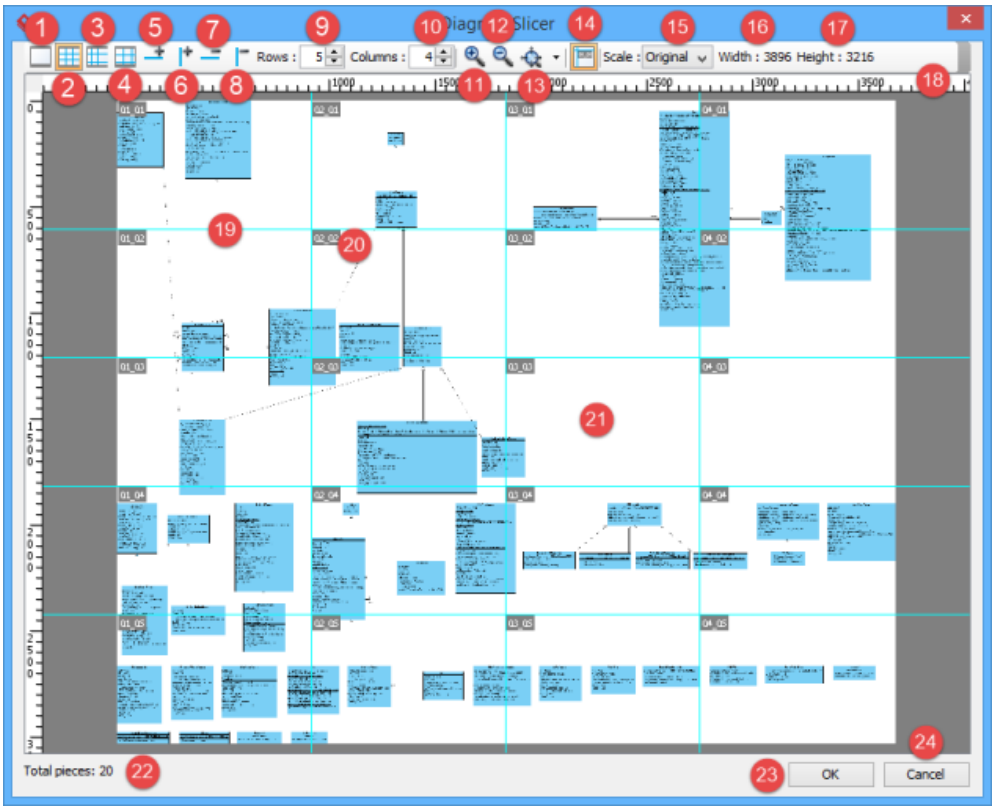
*Description of **Diagram Exporter** window*

Slice a diagram into pieces with diagram slicer

You can slice diagrams into pieces (number of files) as well as restrict the size of the exported diagrams.

The slice diagram window

The way how diagram is sliced can be set per diagram or to all diagrams. To slice a diagram, click on the slice button right under the diagram preview in the **Diagram Exporter** window. To slice all diagrams, enable **Slice Diagrams** and click on **Slice All Diagrams** button. Both ways open the **Diagram Slicer** for configuring how diagram(s) is to be sliced.



An overview of Diagram Slicer window

Below is the description of different parts of the window.

No.	Name	Description
1	No slicing	Do not slice diagram.
2	Fixed size	A simple strategy which slice exported diagram into pieces that have the same size.
3	Free slicing	User can customize how to slice the exported diagram by specifying the position of vertical slices and horizontal slices.
4	Fixed ratio	User can customize how to slice the exported diagram by specifying the position of vertical slices and horizontal slices.
5	Add row	Slice the diagram into specific number of rows.
6	Add column	Slice the diagram into specific number of columns.
7	Remove row	Reduce the number of rows to slice.
8	Remove column	Reduce the number of columns to slice.
9	Rows	The number of rows for slicing a diagram.
10	Columns	The number of columns for slicing a diagram.
11	Zoom in	Magnify the diagram content.
12	Zoom out	Magnify the diagram content.
13	Zoom fit to screen	Adjust the diagram to make it fit well on the slicer window.
14	Show label	Click to show/hide index label.
15	Scale	The way of scaling. When configuring slicing for all diagrams, this part will not be displayed.
16	Width	The total width of diagram.
17	Height	The total height of diagram.
18	Ruler	Shows the size of the diagram. When the slicing strategy **Free Slicing** is selected, a new row and column can be created by dragging a new one from the ruler.
19	Slice line	Lines that divide the diagram into pieces. The show the vertical and horizontal position that the diagram will be sliced at. When **Free Slicing** or **Fixed Ratio** is selected, the lines can be dragged and moved.

20	Index label	Shows the index of the pieces. This index will be printed on the exported file as well.
21	Diagram	The diagram being sliced.
22	Total pieces	The number of pieces to produced.
23	OK	Click to confirm the slice settings.
24	Cancel	Click to close the diagram slicer with changes discarded.

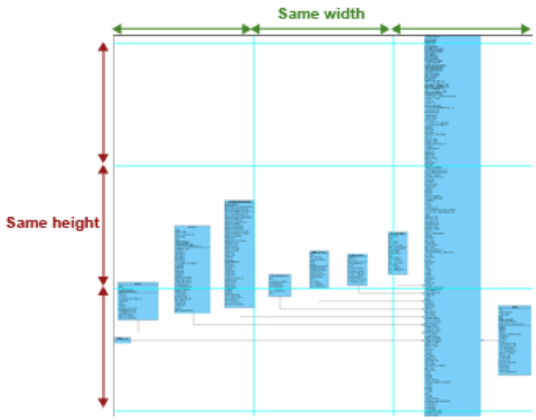
*Description of **Diagram Slicer** window*

Slicing strategies

There are three slicing strategies - **Fixed Size**, **Free Slicing** and **Fixed Ratio**. Each gives a distinct way of slicing images.

Fixed size

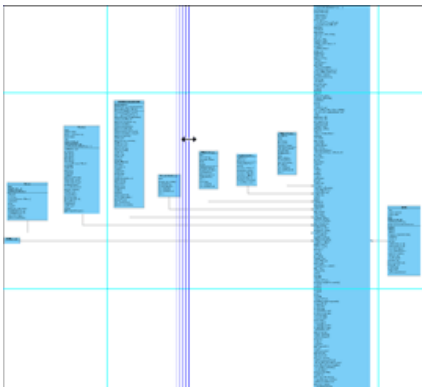
Fixed Size is a simple strategy which slice exported diagram into pieces that have the same size. User specifies the number of columns and rows to slice and then the exported diagram will be sliced into specific pieces.



Fixed Size

Free slicing

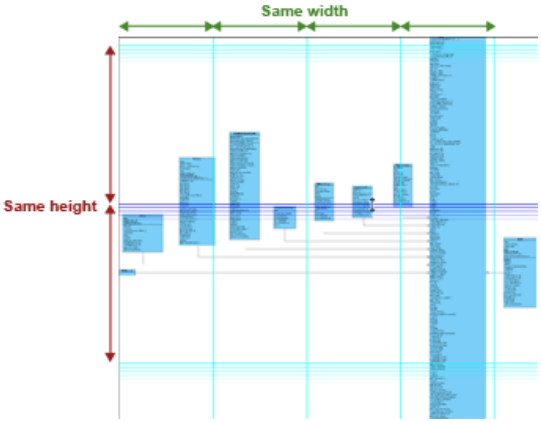
User can customize how to slice the exported diagram by specifying the position of vertical slices and horizontal slices. It is particularly useful to prevent a shape from being sliced into pieces.



Free Slicing

Fixed ratio

Fixed Ratio strategy gains the benefit of **Fixed Slice** and **Free Slicing**. The width and height of pieces are the same but last row and column. User can also customize the width and height of sliced pieces. Like **Free Slicing**, **Fixed Ratio** is size oriented. User modifies the size of pieces and **Diagram Slicer** calculates the number of row and column to slice.



Fixed Ratio

Controlling size of exported image

Diagram Slicer not only slice diagram into pieces but also controls the total size of the exported diagrams. There are scale controls on the right of the toolbar from the **Diagram Slicer** dialog. By default, the type of scale is **Original**. And it shows the size of diagram. The following are some of the possible ways of controlling diagram size.

Control by size

To control the total size of the exported diagram by specific width and height, select **Size** from the **Scale** combo box and then enter the width and height of the diagram. The ruler shows the size of the diagram.

Control by ratio

To control the total size of the exported diagram by specific ratio, select **Ratio** from the **Scale** combo box and enter the ratio in the field next to the combo box. The total size of the exported diagram shows next to that field.

Related Resources

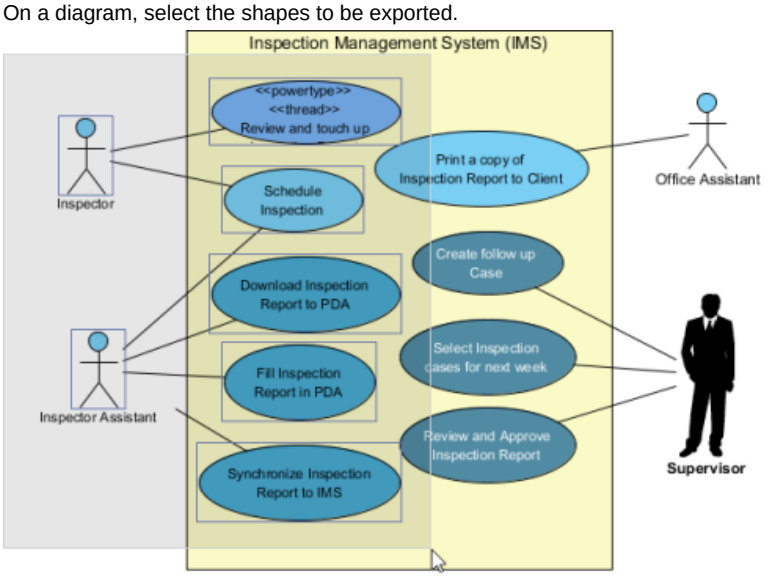
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Exporting portion of diagram as image

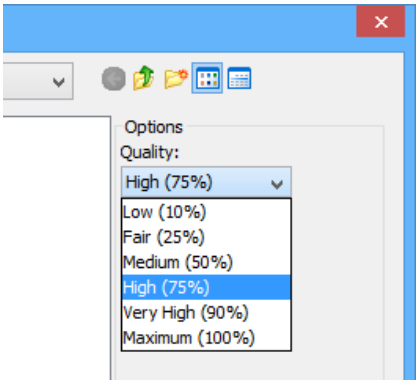
You can export some shapes in a diagram as an image file by selecting the shapes you want to export then perform export. To export selected shapes to image:

1. On a diagram, select the shapes to be exported.



Selecting shapes to export as image

2. Select **Project > Export > Selection as Image...** from toolbar.
3. In the **Save** dialog box, set the image quality. The higher the quality, the clearer the image, the larger the image size.



Set image quality

4. Select the image format at the bottom of dialog box.

> **NOTE:** There are two options for exporting as PNG files - with and without background:
> - With background: export diagram's background color.
> - Without background: ignore the background color by exporting transparent background.

> **NOTE:** You can export diagrams to native PDF format. Since the exported PDF is of a small size, it can save a lot of space. Also, because the diagram in PDF is a vector, it is scalable. There are two export options:
> - PDF(diagram per page): selected diagrams will be exported to the same PDF file. Each diagram will occupy one page.
> - PDF(diagram per file): each diagram will be exported in one new PDF file.

5. Specify the filename of the image file.
6. Click **Save** to export.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

The Open API

Plugin Support provides an interface for developers to integrate with Visual Paradigm. Developers can develop their plugins to achieve domain related operations. In this chapter, the overview of API architecture as well as the development of plugin will be covered.

Introduction to plugin support

Provides basic information about plugin support. Some of the key concepts like diagram, model element and action will be mentioned.

Implementing plugin

Detailed information about how to implement (code) a plugin will be covered.

Deploying plugin

Shows you how to deploy a plugin to Visual Paradigm.

Introduction to plugin support

Plugin Support provides an interface for developers to integrate with Visual Paradigm. Developers can develop their plugins for what they want. In this section, we will introduce the structure of a plugin.

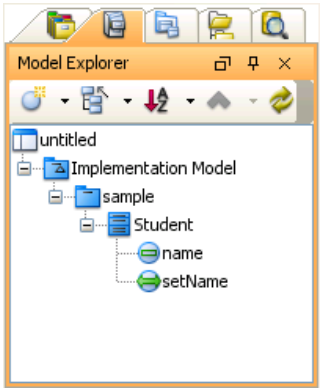
Plugin.xml

A plugin is defined in a XML file (plugin.xml). It includes the information (such as plugin id, provider, required libraries, etc...), custom actions (menu, toolbar and popup menu) and custom shapes/connector of the plugin.

For working with Visual Paradigm in plugin, there are 4 main components must be known by developers: Model, Diagram, Diagram Element and Action/Action Controller.

Model element

Model Elements are basic construct of a model. Plugin allows developer to create, retrieve, update and delete model elements through the popup menu context or through the project (by iterating model elements within a project).

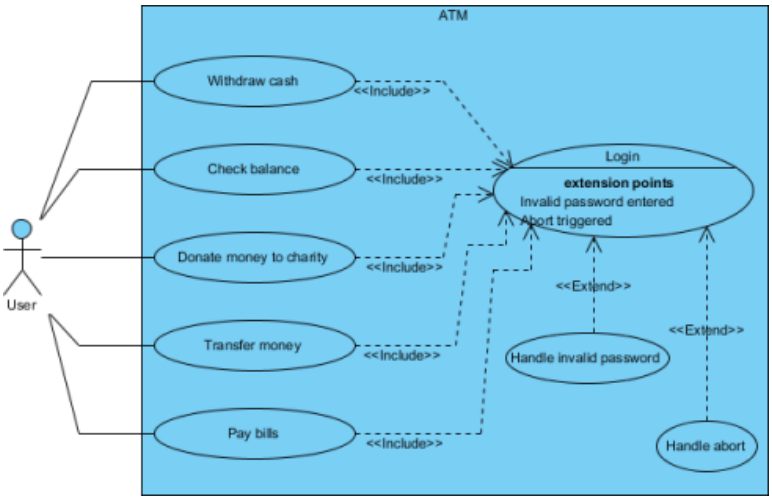


Model elements under Model Explorer

Diagram

Diagram is contains diagram elements on different domain (such as Use Case Diagram, Class Diagram, ERD, etc...).

Plugin allows developer to create, retrieve, update and delete diagrams through the popup menu context or through the project (by iterating diagrams within a project)



An opening Use Case Diagram

Diagram element

A model element does not contain information of appearance (such as x, y, width, height, etc...). It is the diagram element, which appear on the user interface, that owns the appearance data. Diagram Element represents a view of a model element. A model element can be shown on different diagrams (such as a class can be shown on 2 different class diagrams).

There are 2 kinds of diagram element: Shape and Connector. Shape represents the non-relationships diagram element (such as Class). Connector represents the relationships (such as Generalization). Plugin allows developer to create, retrieve, update and delete diagram elements through the popup menu context or through the project (to iterate all the diagrams and then the diagram elements appear on a diagram).

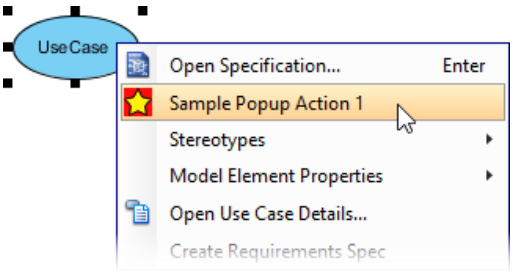
Actor, an example of diagram element

Action/Action controller

Action represents buttons and menus (menu, toolbar and popup menu), which contains the information on outlook (such as label, icon, mnemonic, etc...) and responses to trigger the function call.

Action is used to represent the button on 3 regions: menu/toolbar, popup menu and diagram toolbar

Action Controller is the control (function call) of actions. Developer needs to implement different Action Controller on different region's actions.



Popup menu with user-defined menu item

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Implementing plugin

Configuring development environment

The plugin API is in **%Visual-Paradigm-Install-Dir%/lib/openapi.jar**. In order to program a plugin, developer must import the jar into the development classpaths.

Beginning of plugin.xml

plugin.xml is the base of a plugin, to develop a plugin, you should start from writing the plugin.xml. The basic directory structure is "Visual-Paradigm-Install-DIR/plugins/YOUR_PLUGIN_ID/plugin.xml"

For improving the variability of the plugin.xml, a properties file (plugin.properties) can be used for storing the value of the xml. Developer can assign the value of the attributes in xml starts with '%', then the value will be read from the properties file. For example

In plugin.xml: <plugin id="sample.plugin name="%plugin.name" .../>

In plugin.properties: plugin.name=sample.plugin

Sample on XML:

```
<  plugin
        id= "sample.plugin"
        name= "Sample Plugin"
        description= "Sample Plugin"
        provider= "Visual Paradigm"
        class= "sample.plugin.SamplePlugin">
        <  runtime >
                < library path= "lib/sampleplugin.jar" relativePath= "true"/>
        </  runtime >
        <!-- to be continued -->
</  plugin >
```

Table shows the description of elements in the plugin.xml.

Element	Attribute	Description
plugin		The root element of plugin.xml, specify the basic information of the plugin (id, name, provider, etc...)
plugin	**class**	The class of the plugin, required to implements **com.vp.plugin.VPPlugin**.
runtime		The element specified the runtime environment of the plugin.
library (1..*)		Specify the .jar or directory as the classpaths required on the plugin. Such as the classes of the plugin and some libraries the plugin required.
library (1..*)	**Path**	The path of the .jar or directory.
library (1..*)	**relativePath** (optional, default: **true**)	Specify whether the path is relative path.

plugin.xml element description

Description on Code:

VPPlugin (com.vp.plugin.VPPlugin)

This class must be implemented and ref on <plugin class="xxx"... Otherwise, the plugin will not be loaded completely. In fact, the class can do nothing on it.

The following is the sample code:

```
package sample.plugin;
public class SamplePlugin implements com.vp.plugin.VPPlugin {
        // make sure there is a constructor without any parameters
        public void loaded(com.vp.plugin.VPPluginInfo info) {
                // called when the plugin is loaded
                // developer can get the current plugin's id and the
                // current plugin directory (default: %Visual-Paradigm%/plugins)of Visual-Paradigm from the VPPluginInfo.
        }
        public void unloaded() {
                // called when the plugin is unloaded (when Visual Paradigm will be exited)
        }
}
```

Implementing custom action

There are 2 main components for an Action: Action and Action Controller. Action represents the outlook, Action Controller responses to work as function call. In order to create custom action, developer needs to define the Action on xml, and implement the Action Controller on code.

Sample on XML:

```
<plugin>
        < actionSets>
                <!-- to be continued -->
```

```
        </ actionSets>
        <!-- to be continued -->
</plugin>
```

Table shows the description of elements in the above XML.

Element	Attribute	Description
actionSets		It is a collection of ActionSet. There 2 kinds of ActionSet: **actionSet** and **contextSensitiveActionSet**. **actionSet** is a set of actions which will be shown on menu/toolbar or diagram toolbar. **contextSensitiveActionSet** is set of actions which will be shown on popup menu.

XML sample for custom action

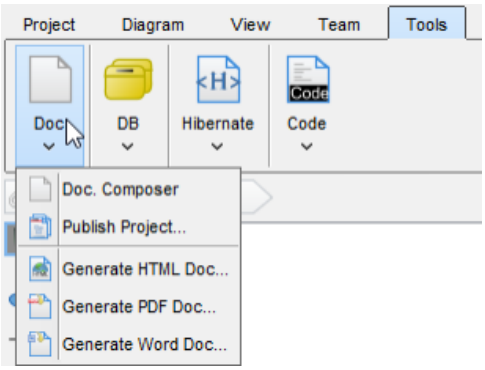
There are differences on xml definition and code implementation of the 3 kinds of Actions (menu/toolbar, popup menu, diagram toolbar).

Custom action on menu/Toolbar

Developer can define the menu, menu item, toolbar, toolbar button and etc... on the plugin.xml. In order to trigger the menu item and toolbar button's function call, Action Controller is required to be implemented and added into the Action. The Action Controller class on menu/toolbar actions is com.vp.plugin.action.VPActionController.

There are 2 important attributes used on menu, action and separator: **menuPath** and **toolbarPath** .

menuPath is the path specified where is the item placed on menu, toolbarPath is the path specified where is the item placed on toolbar. The path is formed by a set of 'name'. The 'name' is similar with the caption of the menu items (caption in English, ignores the "..." and remind the 'space'). '/' is used as delimiter of the path. '#' is used to represent the front of the menu. Here, 4 examples will be given:



Custom Action on MenuBar

Below is the menupaths required for implementing the menus shown in the above images.

Menu	"label" in XML	"menupath" in XML	Remarks
1	Tools	Tools	After the Tools menu
2	Tools/Document	Tools/Document	Under the Tools menu, after the Document menu
3	Tools/Document/#	Tools/Document/#	Under the Tools menu, and under the Document menu, place on the front
4	Tools/Document/Generate HTML Document	Tools/Document/Generate HTML Document	Under the Tools menu, and under the Document menu, after the Generate HTML Document menu item

Different menupaths settings

Sample on XML:

```
< actionSet id="sample.plugin.actions.ActionSet1">
        < toolbar
                        id= "sample.plugin.actions.Toolbar1"
                        orientation= "north"
                        index= "last"/>
        < menu
                        id= "sample.plugin.actions.Menu1"
                        label= "Sample Menu 1"
                        mnemonic= "M"
                        menuPath= "Tools/Document"/>
        < action
                        id= "sample.plugin.actions.Action1"
                        actionType= "generalAction"
                        label= "Sample Action 1"
                        tooltip= "Sample Action 1"
```

```
                    icon= "icons/red.png"
                    style= "normal"
                    menuPath= "Tools/Document"
                    toolbarPath= "sample.plugin.actions.Toolbar1/#">
                    < actionController class= "sample.plugin.actions.ActionController"/>
        </ action>
        < separator
                    id= "sample.plugin.actions.Separator1"
                    menuPath= "Tools/sample.plugin.actions.Action1"
                    toolbarPath= "sample.plugin.actions.Toolbar1/sample.plugin.action.Action1"/>
</ actionSet>
```

The table shows the description of elements in the above XML.

Element	Attribute	Description
actionSets		It is a collection of ActionSet. There are 2 kinds of ActionSet: **actionSet** and **contextSensitiveActionSet**. **actionSet** is a set of actions which will be shown on menu/toolbar or diagram toolbar. **contextSensitiveActionSet** is set of actions which will be shown on popup menu.
toolbar (0..*)		Specify a toolbar, contains the location information of the toolbar.
toolbar (0..*)	**orientation** [north | east |south | west]	Specify which side will be the toolbar placed on.
toolbar (0..*)	**index** [(number) | last |new]	Based on the orientation, where the toolbar will be placed. e.g. the orientation is "north" and there are 2 rows toolbars already. If the index is "0", then the toolbar will be placed on the first row's last position. If the index is "last", the toolbar will be placed on the last row, last position. If the index is "new", the toolbar will be placed on the third row (new row).
menu (0..*)		Specify a menu or a pull down button on menu bar or toolbar. It contains the outlook information of the menu.
action (0..*)		Specify a menu item or button on menu bar or toolbar. It contains the outlook information of the menu item.
action (0..*)	**actionType** [generalAction | shapeAction | connectorAction] (optional, default: generalAction)	There are 3 types: generalAction, shapeAction and connectorAction. As the action on menu/toolbar, generalAction should be assigned.
actionController		Specify the Action Controller for the action (the parent node in the xml).
actionController		The class name of the Action Controller. For the action on menu/toolbar, it is required to implement **com.vp.plugin.action.VPActionController**.
separator (0..*)		Specify a separator on menu bar or toolbar.

XML sample for menus and toolbars

Description on Code:

VPActionController (com.vp.plugin.action.VPActionController)

This class is used to perform the function call when the action is clicked. One Action Controller class refers to multi Actions is allowed.

Sample:

```
package sample.plugin.actions;
public class ActionController implements com.vp.plugin.action.VPActionController {
        // make sure there is an constructor without any parameters
        public void performAction(com.vp.plugin.action.VPAction action) {
                // called when the button is clicked, the parameter action represents the Action which be clicked.
                // developer also can set the properties of the action
        }
        public void update(com.vp.plugin.action.VPAction action) {
                // *for the actions located on menu bar only
                // when the parent menu is selected, this will be called,
                // developer can set the properties of the action before it is shown (e.g. enable/disable the menu item)
        }
}
```

Custom action on popup menu (context sensitive)

Developer can define the menu, menu item and separator on the popup menu shown on the diagram. The popup menu on diagram is context sensitive which based on what diagram element or diagram is selected. In order to make the menu item trigger the function call, Action Controller is required to be implemented. For popup menu, **com.vp.plugin.action.VPContextActionController** is the interface required developer to implement.

Same as Action on Menu/Toolbar, **menuPath** is used to specify the location of the action (menu/menu item on popup menu).

Sample on XML:

```
< contextSensitiveActionSet id= "sample.plugin.actions.ActionSet2">
	< contextTypes all= "false">
		< include type="Class"/>
		<!-- ignored when contextTypes.all = true -->
		< exclude type="Package"/>
		<!-- ignored when contextTypes.all = false -->
	</ contextTypes>
	<action
		id= "sample.plugin.actions.ContextAction1"
		label= "Sample Action [1]"
		icon= "icons/blue.png"
		style= "toggle"
		menuPath= "OpenSpecification">
		< actionController class= "sample.plugin.actions.ContextActionController"/>
	</action>
</contextSensitiveActionSet>
```

Table shows the description of elements in the above XML.

Element	Attribute	Description
contextSensitvieActionSet (0..*)		It is a collection of menu, action, separator on the popup menu of the plugin. The child elements should be ordered if they have the relationship on the position (e.g. developer prefers Action1 is placed into Menu1, then please define the Menu1 on the xml first
contextTypes		It is a collection of the model of diagram element of diagram types which the **contextSensitiveActionSet** is considering.
contextTypes	**all** [true | false] (optional, default: **false**)	Specify whether all the types of the models, diagram elements and diagrams will be considered by this actionSet.
Include		Specify the model, diagram element or diagram type will be considered by this ActionSet. (This will be ignored if the **contextType's** attribute **'all'** is assigned 'true'.
Include	**type**	It is type of the element. Such as "Class", "Actor", "ClassDiagram", "Attribute", etc...
exclude		Specify the model, diagram element or diagram type will not be considered by this ActionSet. (This will be ignored if the **contextType's** attribute **'all'** is assigned 'false'.
type		It is type of the element. Such as "Class", "Actor", "ClassDiagram", "Attribute", etc...
actionController		Specify the Action Controller for the action (the parent node in the xml)
actionController	**class**	The class name of the Action Controller. For the action on popup menu, it is required to implement **com.vp.plugin.action.VPContextActionController**.

XML sample for popup menu

Description on Code:

VPContextActionController (com.vp.plugin.action.VPContextActionController)

This class is used to perform the function call when the action is clicked. One Action Controller class refers to multi Actions is allowed.

Sample:

```
package sample.plugin.actions;
import java.awt.event.ActionEvent;
public class ContextActionController implements com.vp.plugin.action.VPContextActionController {
	// make sure there is an constructor without any parameters
	public void performAction(
		com.vp.plugin.action.VPAction action,
```

```
		com.vp.plugin.action.VPContext context,
		ActionEvent e
	) {
		// called when the button is clicked
	}
	public void update(
		com.vp.plugin.action.VPAction action,
		com.vp.plugin.action.VPContext context
	) {
		// when the popup menu is selected, this will be called,
		// developer can set the properties of the action before it is shown (e.g. enable/disable the menu item)
	}
}
```

VPContext (**com.vp.plugin.action.VPContext**)

Context will be passed into the Action Controller when the popup menu is shown or action is triggered. It is what the user selected on the diagram, can be model, diagram element or/and diagram.

A diagram may contain many diagram elements, when user right-click on the diagram element or the diagram, a popup menu will be shown. So, the context may be diagram element or diagram. However, the diagram element must be contained by diagram, then if popup menu is shown on a diagram element, the context must contain both diagram element and diagram. And the diagram element always represents for a model, so that is possible the context contains model, diagram element and diagram as same time. However, sometime, the popup menu is shown for a model only (e.g. select on an attribute of a class, because there is no diagram element for the attribute, the class's diagram element will be contained in the context).

Custom diagram element (shape and connector)

Developer can define the shape to connect on the specified diagram. But it is not allowed to develop a custom model. ActionSet and Action are used on definition of custom diagram element.

Sample on XML:

```
<actionSet id= "sample.plugin.actions.ShapeActionSet">
	<action
		id=  "sample.plugin.actions.ShapeAction1"
		actionType=  "shapeAction"
		label=  "Sample Action {1}"
		tooltip=  "Sample Action {1}"
		icon=  "icons/yellow.png"
		editorToolbarPath= "com.vp.diagram.ClassDiagram/Class">
		< shapeCreatorInfo
			shapeType= "sample.plugin.shape.Shape1"
			defaultWidth= "30"
			defaultHeight= "30"
			controllerClass= "sample.plugin.actions.ShapeController1"
			multilineCaption= "false"
			captionStyle= "north"
			resizable= "true"/>
	</action>
	<action
		id=  "sample.plugin.actions.ConnectorAction1"
		actionType= "connectorAction"
		label=  "Sample Action {2}"
		tooltip=  "Sample Action {2}"
		icon=  "icons/green.png"
		editorToolbarPath= "com.vp.diagram.ClassDiagram/sample.plugin.actions.ShapeAction1">
		<connectorCreatorInfo
			shapeType= "sample.plugin.connector.Connector1"
			fromArrowHeadStyle= "Arrow1"
			toArrowHeadStyle= "Arrow2"
			fromArrowHeadSize= "verySmall"
			toArrowHeadSize= "large"
			dashes= "7,10"
			lineWeight= "3"
			connectorStyle= "rectilinear">
			< connectionRules>
				< connectionRule
					fromShapeType= "sample.plugin.shape.Shape1"
					toShapeType= "sample.plugin.shape.Shape1"
					bidirection= "true"/>
				< connectionRule
					fromShapeType= "Class"
					toShapeType= "sample.plugin.shape.Shape1"
					bidirection= "true"/>
				<connectionRule
					fromShapeType= "Package"
					toShapeType= "sample.plugin.shape.Shape1"
					bidirection= "true"/>
			</connectionRules>
		</connectorCreatorInfo>
	</action>
```

```
</actionSet>
```

Table shows the description of elements in the above XML.

Element	Attribute	Description
Action		It is a collection of menu, action, separator on the popup menu of the plugin. The child elements should be ordered if they have the relationship on the position (e.g. developer prefers Action1 is placed into Menu1, then please define the Menu1 on the xml first
Action	**actionType** [generalAction | shapeAction | connectorAction] (optional, default: generalAction)	There are 3 types: generalAction, shapeAction and connectorAction. As the action for custom shape, "shapeAction" should be assigned. For custom connector, "connectorAction" should be assigned.
Action	**editorToolbarPath**	Specify which diagram toolbar contains this action. e.g. to add a shapeAction on class diagram after the button for creating a new class, "com.vp.diagram.ClassDiagram/Class" should be assigned. "com.vp.diagram.ClassDiagram" is the id of the class diagram. "/" is the delimiter. "Class" is the button id.
shapeCreatorInfo		If the actionType is "shapeAction", **shapeCreatorInfo** is required. It is used to specify the details of the custom shape.
shapeCreatorInfo	shapeType	The shape type assigned by developer, unique value is required.
shapeCreatorInfo	**captionStyle** [center | north | none] (optional)	Specify where the caption of the shape is displayed.
shapeCreatorInfo	**controllerClass**	The class name which the class is responsible to draw the shape on the diagram, **com.vp.plugin.diagram.VPShapeController** is required to be implemented. **com.vp.plugin.diagram.AbstractShapeController** is an abstract class of the **VPShapeController**.
connectorCreatorInfo		If the actionType is "connectorShape", **connectorCreatorInfo** is required. It is used to specify the details of the custom connector.
connectorCreatorInfo	**shapeType**	The shape type assigned by developer, unique value is required.
connectorCreatorInfo	**connectorStyle** [oblique | rectilinear] (optional, default: oblique)	Specify the style of the connector.
connectorCreatorInfo	**fromArrowHeadStyle** (optional)	Specify the arrow head style of the "from" side of the connector.
connectorCreatorInfo	**toArrowHeadStyle** (optional)	Specify the arrow head style of the "to" side of the connector.
connectorCreatorInfo	**fromArrowHeadSize** [verySmall | small | medium | large | extraLarge | jumbo | colossal] (optional)	Specify the arrow head size of the "from" side of the connector.
connectorCreatorInfo	**toArrowHeadSize** [verySmall | small | medium | large | extraLarge | jumbo | colossal] (optional)	Specify the arrow head size of the "to" side of the connector
connectorCreatorInfo	**dashes** (optional)	Specify the dashes pattern of the connector. A list of float, written in the pattern "%f, %f, %f, ...".
connectorCreatorInfo	**lineWeight** (optional)	Specify the weight of the connector.
connectorRules		It is a collection of **connectorRule**.
connectorRule (1..*)		Specify this connector can connect with what diagram element.

XML sample for diagram element

Description on Code:

VPShapeController (**com.vp.plugin.diagram.VPShapeController**)

It responses to handle the outlook of the shape on the diagram.

Sample:

```
package sample.plugin.actions;
// import the necessaries
public class ShapeController implement com.vp.plugin.diagram.VPShapeController {
	public void drawShape(
		Graphics2D g, Paint lineColor, Paint fillColor, Stoke stroke,
		Com.vp.plugin.diagram.VPShapeInfo shapeInfo
	){
		// draw the shape by the graphics
		// shapeInfo contains the information of the shape, e.g. the bounds of the shape.
	}
	public boolean contains( int x, int y, com.vp.plugin.diagram.VPShapeInfo shapeInfo) {
		// check whether the x, y is inside the shape,
		// it is used to checking what is selected by the user
	}
}
```

Working with models

Plugin Support provides interface for the developer to create, retrieve update and delete the models in Visual Paradigm. The base class of the model is **com.vp.plugin.model.IModelElement**. All models are contained in the project (**com.vp.plugin.model.IProject**). Each model has a model type to access all the model type, please refer to the class **com.vp.plugin.model.IModelElementFactory**, it is the class to create the models.

Creating model

Developer can use the model element factory (**com.vp.plugin.model.IModelElementFactory**) to create the model. Or based on a parent model (**com.vp.plugin.model.IModelElementParent**) to create a child model.

IModelElementFactory can be access by **IModelElementFactory.instance()**. It provides the functions to create all the models.

IModelElementParent is the subclass of **IModelElement**. It provides the function to create the child into it. If the parent class is more specified, it may support a more details function to create the child. For example, **IClass** is subclass of **IModelElementParent**, it provides **createOperation()** to create an operation into it.

Sample on Code:

```
/*
* create model by IModelElementFactory
* result of the 2 methods: "class model is created and added into the project"
*
// assume in a code segment
IClass classModel1 = IModelElementFactory.instance().createClass();
IClass classModel2 = (IClass) IModelElementFactory.instance().create(IModelElementFactory. MODEL_TYPE_CLASS);
/*
* create model by IModelElementParent
* result of the first 2 methods, "operation model is created and added into the class model"
* result of the last method, "actor model is created and added into project", because actor cannot be the child of class model
*/
// assume in a code segment
IOperation operationModel1 = classModel1.createOperation();
IOperation operationModel2 = (IOperation) classModel1.create(IModelElementFactory. MODEL_TYPE_OPREATION);
IActor actorModel1 = (IActor) classModel1.create(IModelElementFactory. MODEL_TYPE_ACTOR);
```

Retrieving model

Developer can use the project (**com.vp.plugin.model.IProject**) or the context (**com.vp.plugin.action.VPContext**) from ActionController to retrieve the models.

IProject is the project of Visual Paradigm. The project contains all models, diagram and diagram elements. It provides function (**modelElementIterator()**) for the developer to iterate the models.

VPContext is the context of a popup menu. Developer can access the context by popup menu's action controller (**com.vp.plugin.action.VPContextActionController**). Context may contain a model element if the popup menu is shown on a diagram element or model.

Sample on Code:

```
/*
* retrieve model by IProject
*/
// assume in a code segment
IProject project = ApplicationManager.instance().getProjectManager().getProject();
Iterator iter = project.modelElementIterator();
while (iter.hasNext()) {
	IModelEmenet modelElement = (IModelElement) iter.next();
	// model element retrieved
}
/*
*retrieve model by VPContext
*/
// assume on a sub-class of com.vp.plugin.action.VPContextActionController
```

```
public void update(VPAction action, VPContext context) {
	IModelElement modelElement = context.getModelElement();
	// model element retrieved, but please take care,
	// context.getModelElement() may return null if the popup menu is shown for the diagram
	// or the selected diagram element doesn't refer to a model element.
}
/*
* retrieve relationship model from a class model
* there are 2 kinds of relationships: IRelationship and IEndRelationship
*/
// assume in a code segment
IClass classModel = ...; // retrieved the class model from somewhere
// retrieve a generalization (IRelationship)
Iterator genIter = classModel.fromRelationshipIterator();
while (genIter.hasNext()) {
	IRelationship relationship = (IRelationship) genIter.next();
	// found out the another side's model of the relationship
	IModelElement otherModel = relationship.getTo();
}
// retrieve am association (IEndRelationship)
Iterator assoIter = classModel.fromRelationshipEndIterator();
while (assoIter.hasNext()) {
	IRelationshipEnd relationshipEnd = (IRelationshipEnd) assoIter.next();
	IModelElement otherModel = relationshipEnd.getEndRelationship().getToEnd().getModelElement();
}
```

Updating model

Developer can call a set of get/set methods on a model. Different model type has different properties. For setting and getting the model's property, cast the **IModelElement** into its sub-class is necessary. For example, developer gets the **IModelElement** from the popup menu's context. Developer checks whether the model is a **IClass**, then developer casts the **IModelElement** into **IClass**, and call the function **IClass.setVisibility(xxx)**.

Sample on Code:

```
/*
* update a class model
*/
// assume in a code segment
IModelElement model = ...; // model is retrieved from somewhere
If (IModelElementFactory. MODEL_TYPE_CLASS.equals(model.getModelType()) ) {
	IClass classModel = (IClassModel) model;
	// set the class to be 'private'
	classModel.setVisibility(IClass. VISIBILITY_PRIVATE);
	// set super class
	IClass superClassModel = ...; // another class model is retrieved, it will be set to be the previous model's super class
	IGeneralization generalizationModel = IModelElementFactory.instance().createGeneralization();
	generalizationModel.setFrom(superClassModel);
	generalizationModel.setTo(classModel);
	// get all "setName" operation from the class and set to be "protected final"
	Iterator operationIter = classModel.operationIterator();
	while (operationIter.hasNext()) {
		IOperation operation = (IOperation) operationIter.next();
		if ( "setName".equals(operation.getName()) ) {
			operation.getJavaDetail( true).setJavaFinal( true);
			operation.setVisibility(IOperation. VISIBILITY_PROTECTED);
		}
	}
}
```

Deleting model

Developer can delete the model by simple way, just call the **IModelElement.delete()**.

Working with diagrams/Diagram elements

Plugin Support provides interface for the developer to create, retrieve update and delete the diagrams or diagram elements in Visual Paradigm. The base class of the diagram is co **m.vp.plugin.diagram.IDiagramUIModel**. The base class of the diagram element is **com.vp.plugin.diagram.DiagramElement**. All diagrams are contained in the project (**com.vp.plugin.model.IProject**). And the diagram elements can be found in the diagrams. The diagram elements can be contained by the diagrams.

Creating diagrams/Diagram elements

Developer can create the diagram or diagram element by com.vp.plugin.DiagramManager. DiagramManager can be accessed by ApplicationManager.instance().getDiagramManager().

Sample on Code:

```
// assume in a code segment
DiagramManager diagramManager = ApplicationManager.instance().getDiagramManager();
/*
* create diagram
*/
IDiagramUIModel diagram = diagramManager.createDiagram(DiagramManager. DIAGRAM_TYPE_CLASS_DIAGRAM);
```

```
/*
* create diagram element with exists models
*/
IModelElement classModel1 = ...; // retrieved a class model from somewhere
IModelElement packageModel1 = classModel1.getParent(); // assume the class model is contained by a package
IDiagramElement packageDiagramElement1 = diagramManager.createDiagramElement(diagram, packageModel1);
IDiagramElement classDiagramElement1 = diagramManager.createDiagramElement(diagram, classModel1);
// class's diagram element should be a shape, not a connector
packageDiagramElement1.addChild((IShapeUIModel) classDiagramElement1);
/*
* create diagram element without models (the model will be created automatically)
*/
IDiagramElement newClassDiagramElement =
diagramManager.createDiagramElement(diagram, IClassDiagramUIModel. SHAPETYPE_CLASS);
IModelElement newClassModel = newClassDiagramElement.getModelElement();
/*
* open the created diagram
*/
diagramManager.openDiagram(diagram);
```

Retrieving diagrams/Diagram elements

Developer can use the project (**com.vp.plugin.model.IProject**) to retrieve the diagrams. Use a diagram (**com.vp.plugin.diagram.IDiagramUIModel**) to retrieve the contained diagram elements. Or use the context (**com.vp.plugin.action.VPContext**) from ActionController to retrieve the diagram and/ or diagram element.

IProject is the project of Visual Paradigm. The project contains all models, diagram and diagram elements. It provides function (**diagramIterator()**) for the developer to iterate the diagrams.

IDiagramUIModel is a diagram, which may contain many diagram elements.

VPContext is the context of a popup menu. Developer can access the context by popup menu's action controller (**com.vp.plugin.action.VPContextActionController**). Context may contain a diagram and/or diagram elements.

Sample on Code:

```
/*
* retrieve diagram from IProject
*/
// assume in a code segment
IProject project = ApplicationManager.instance().getProjectManager().getProject();
Iterator diagramIter = project.diagramIterator();
while (diagramIter.hasNext()) {
	IDiagramUIModel diagram = (IDiagramUIModel) diagramIter.next();
	/*
	* retrieve diagram element from IDiagramUIModel
	*/
	Iterator diagramElementIter = diagram.diagramElementIterator();
	while (diagramElementIter.hasNext()) {
		IDiagramElement diagramElement = (IDiagramElement) diagramElementIter.next();
	}
}
/*
* retrieve diagram and diagram element from VPContext
*/
// assume on a sub-class of com.vp.plugin.action.VPContextActionController
public void update(VPAction action, VPContext context) {
	IDiagramUIModel diagram = context.getDiagram();
	IDiagramElement diagramElement = context.getDiagramElement();
	// diagramElement may be null, if the popup menu shown for the diagram
}
/*
* retrieve connected connector from a shape
* because a connector can connected with both Shape and Connector, please check the
* both getToShape() and getToConnector() or getFromShape() and getFromConnector()
*/
// assume in a code segment
IShapeUIModel shape = ...; // retrieved the shape from somewhere
IConnectUIModel[] connectors = shape.toFromConnectorArray();
int count = connectors == null ? 0 : connectors. length;
for ( int i = 0; i < count; i++) {
	IDiagramElement toDiagramElement = connectors[i].getToShape();
	if (toDiagramElement == null) {
		toDiagramElement = connectors[i].getToConnector();
	}
}
```

Updateing diagrams/Diagram elements

IDiagramUIModel provides the functions to set the diagram outlook (size, background, etc...).

IDiagramElement is the super class of **IShapeUIModel** and **IConnectorUIModel**. Because there is a difference between shape and connector, the **IShapeUIModel** and **IConnectorUIModel** provide different set of functions to update them.

Sample Code:

```
/*
* update a shape's size and set a connector's connector style
*/
// assume in a code segment
IShapeUIModel shape = ...; // retrieved the shape from somewhere
shape.setBounds(20, 20, 400, 400);
IConnector connector = ...; // retrieved the connector from somewhere
connector.setConnectorStyle(IConnector. CS_CURVE);
```

Deleting diagrams/Diagram elements

Developer can delete the diagram and diagram element by simple way, just call the **IDiagramUIModel.delete()** and **IDiagramElement.delete()**.

Showing dialog on Visual Paradigm

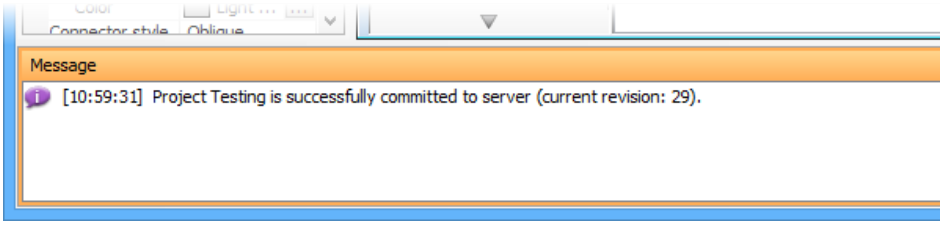
Since Visual Paradigm may be integrated with different platforms which may not support Swing (e.g. Eclipse, Visual Studio). That may make the process to be hung on if using the Swing dialog technology (e.g. JOptionPane and JDialog). So, it is necessary to use a special method to show the dialog with Swing technology.

com.vp.plugin.ViewManager is an interface provides function for developer to show the dialog as same as show dialog by JOptionPane. Besides that, **Viewmanager** supports developer to show message on Visual Paradigm's message pane and show custom dialog by implementing an interface (**com.vp.plugin.view.IDialogHandler**).

Same as JOptionPane, to show a dialog, it is better to have a component as the invoker/parent component. To get the component in Visual Paradigm, just call **ViewManager.getRootFrame()**.

Showing message on message pane

ViewManager provides function **showMessage(msg:String, msgTabId:String)** to show the message on Message Pane. The parameter **msg** is the content of the message, **msgTabId** is the id to identify the tab on Message Pane, which can be defined by developer.



Message in Message Pane

Sample on Code:

```
// assume in a code segment
ViewManager viewManager = ApplicationManager.instance().getViewManager();
viewManager.showMessage( "Thank you for reading Visual Paradigm Plugin Support User's Guide. >=)", "sample.plugin");
```

Showing simple message dialog

In Swing, we may use the **javax.swing.JOptionPane** to show a message dialog (e.g. **JOptionPane.showMessageDialog(...)**). **ViewrManager** provides the functions which simulate the JOptionPane. **ViewManger** provides a set of **showXXXXDialog(...)** functions for showing the dialog. The signature of the functions are same with the JOptionPane. Developer need not feel strange on calling the **showXXXXDialog(...)** functions.

Showing custom dialog

In Swing, we may implement the **javax.swing.JDialog** and add our component on the dialog's content pane. But in plugin, developer is required to implement an interface **com.vp.plugin.view.IDialogHandler** to work for the dialog.

IDialogHandler specify the behaviors of a dialog. There are 4 functions need to be implemented.

```
getComponent() : java.awt.Component
```

It is called once before the dialog is shown. Developer should return the content of the dialog (similar to the content pane).

```
prepare(dialog : com.vp.plugin.view.IDialog) : void
```

It is called after the **getComponent()**. A dialog is created on Visual Paradigm internally (it still not shown out). Developer can set the outlook of the dialog on **prepare()**, such as title, bounds and modal, etc... For your convenience, the dialog will be shown on the screen center as default. If developer don't want change the location, there is no necessary to call the **setLocation()** function.

```
shown()
```

It is called when the dialog is shown. Developer may need to do something when the dialog is shown, such as checking something before user to input data on the dialog.

```
canClosed()
```

It is called when the dialog is closed by the user clicking on the close button of the frame. Developer may not allow the user to close the dialog (e.g. failed on validation check), then please return 'false' on **canClosed()**.

Sample on Code:

```
package sample.plugin.dialog;
// assume imported necessary classes
public class CustomDialogHandler implements IDialogHandler {
	private IDialog _dialog;
	private Component _component;
	private JTextField _inputField1, _inputField2, _answerField;
	public Component getComponent() {
		this._inputField1 = new JTextField(10);
		this._inputField2 = new JTextField(10);
		this._answerField = new JTextField(10);
		JLabel addLabel = new JLabel( " + "); JLabel equalLabel = new JLabel( " = ");
		JButton okButton = new JButton( "Apply");
		okButton.addActionListener( new ActionListener() {
		public void actionPerformed(ActionEvent e) { ok();}
		} );
		JPanel pane = new JPanel();
		pane.add( this._inputField1); pane.add(addLabel); pane.add( this._inputField2);
		pane.add(equalLabel); pane.add( this._answerField); pane.add(okButton);
		this._component = pane;
		return pane;
	}
	public void prepare(IDialog dialog) {
		this._dialog = dialog;
		dialog.setModal(true);
		dialog.setTitle( "Maths Test");
		dialog.setResizable( false ); dialog.pack();
		this._inputField1.setText(String.valueOf(( int)(Math.random()*10000)));
		this._inputField2.setText(String.valueOf(( int)(Math.random()*10000)));
	}
	public void shown() {
		ApplicationManager.instance().getViewManager().showMessageDialog(
		this._component, "Maths Test is started, you have an half hour to finish this test.",
		"Maths Test", JOptionPane. INFORMATION_MESSAGE
		);
	}
	public boolean canClosed() {
		if ( this.checkAnswer()) { return true; }
		else {
			ApplicationManager.instance().getViewManager().showMessageDialog(
				this._component, "Incorrect",
				"Maths Test", JOptionPane. ERROR_MESSAGE
			);
			return false;
		}
	}
	private void ok() {
		if ( this.checkAnswer() ) { this._dialog.close(); }
		else {
			ApplicationManager.instance().getViewManager().showMessageDialog(
				this._component, "Incorrect",
				"Maths Test", JOptionPane. ERROR_MESSAGE
			);
		}
	}
	private boolean checkAnswer() {
		try {
			int a = Integer.parse Int( this._inputField1.getText());
			int b = Integer.parse Int( this._inputField2.getText());
			int c = Integer.parse Int( this._answerField.getText());
			return (a+b == c);
		}
		catch (Exception ex) { return false; }
	}
}
```

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Deploying plugin

After preparing all the required files for a plugin (plugin.xml, plugin.properties, classes/libraries and other resources), developer can plug the plugin into Visual Paradigm.

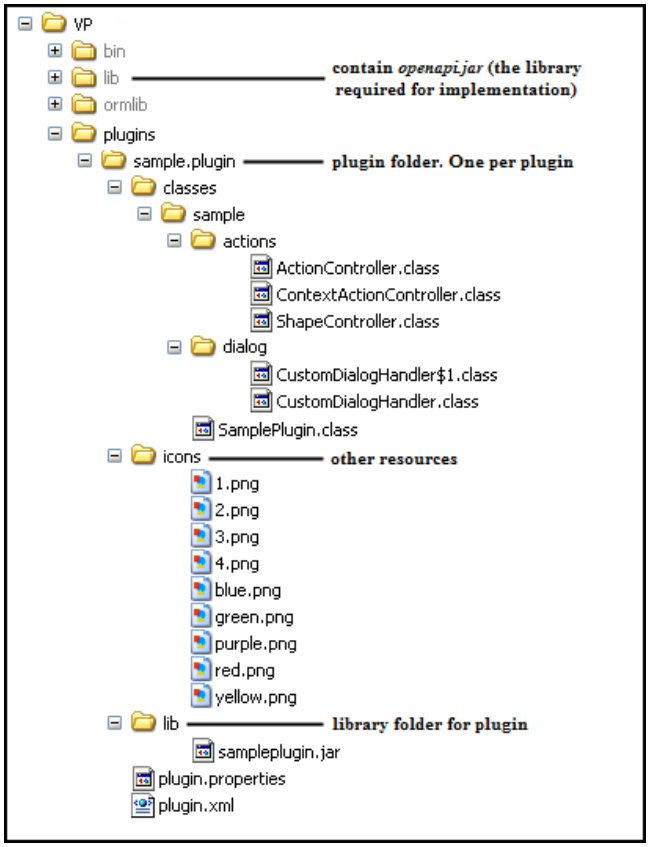
First, create a folder named **plugins** (notice the ' **s**') in the Visual Paradigm installation directory. Put the plugin files into "%Visual-Paradigm%\plugins \%PLUGIN_ID%\". %PLUGIN_ID% is a directory named as the plugin id (use the id as the directory name to avoid duplicated directories defined in **plugins**)

The following structure should be obtained:

%Visual Paradigm%

bin
lib
...
plugins
sample.plugin *(%PLUGIN_ID %)*
plugin.xml
plugin.properties
classes
sample (package)
... (other packages or classes or resources)
lib
sampleplugin.jar
... (others .jar)
icons (others resources)
red.png
...(other resources)

Below is an example of Vsiual Paradigm installation folder with plugin created in the **plugins** folder.



Plugin folder structure

After all, restart Visual Paradigm will see the plugin available. If not, make sure the code has been written correctly and can be compiled and you have setup the above folder structure correctly.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Command line interface

Instead of executing command through graphic user interface, you can also execute certain command in background, through the command line interface. In this chapter, all the supported command will be described in detail.

Exporting diagram image

The command needed to export diagram as image.

Exporting and importing XML

The command needed to export project data as XML.

Generating report

The command needed to generate HTML/PDF/Word report

Project publisher

The command needed to publish a project

Updating teamwork project from server

The command needed to update a local teamwork project by getting changes from server.

Executing operations with Apache Ant

How to execute commands with Ant script.

Exporting diagram image

To export images from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ExportDiagramImage** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ExportDiagramImage -project C:\Demo\Demo.vpp -out C:\Demo\Output -diagram "" -type jpg*
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	Folder for storing the exported images	C:\Demo\Output
-diagram	A list of diagram required to export images. User can enter "*" for representing all diagrams to supply the names of diagrams or to supply a text file which includes the names of all diagrams	diagram_1 diagram_2
-type [optional]	Type of diagrams. Here are the possible types: • png • png_with_background • jpg • svg • pdf	png

Parameters for ExportDiagramImage

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Exporting and import XMI

Visual Paradigm supports interoperability with XMI file, a standard made for data exchange. You can export project data to an XMI, edit it externally with other softwares that accepts XMI. In this chapter, you will see how to export and import XMI through the command line interface.

Exporting XMI

To export XMI from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ExportXMI** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ExportXMI -project C:\Demo\Demo.vpp -out C:\Demo\Output\Sample.xmi -type 2.1
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The filepath of XMI file	C:\Demo\Output\sample.xmi
-type [optional]	Version of XMI. Unless specified, the lastly generated version will be selected. Here are the possible options: • 2.1 • 2.1UML2	2.1
-encoding [optional]	Encoding of XMI file	

Parameters for ExportXMI

Importing XMI

To import XMI to a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ImportXMI** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ImportXMI -project C:\Demo\Demo.vpp -file C:\Demo\input\sample.xmi
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the XMI file to import	C:\Demo\input\sample.xmi

Parameters for ImportXMI

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Exporting and import XML

You can export project data to an XML, manipulate it externally and feed the changes back to Visual Paradigm. In this chapter, you will see how to export and import XML file through the command line interface.

Exporting XML

To export XML and images from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ExportXML** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ExportXML -project C:\Demo\Demo.vpp -out C:\Demo\Output
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	Folder for storing the exported XML and images	C:\Demo\Output
-diagram	One or more diagrams to be exported	"Diagram A" "Diagram B"
-noimage	Do not export image files for diagrams	N/A
-refmodel	Whether to embed referenced projects' content inline	true

Parameters for ExportXML

Importing XML

To import XML to a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ImportXML** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ImportXML -project C:\Demo\Demo.vpp -file C:\Demo\input\project.xml
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the XML file to import	C:\Demo\input\sample.xml

Parameters for ImportXML

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Exporting and import Excel

Visual Paradigm supports interoperability with Excel file. You can export project data to an Excel and edit it externally. In this chapter, you will see how to export and import Excel through the command line interface.

Exporting Excel

To export Excel from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ExportExcel** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ExportExcel -project C:\Demo\Demo.vpp -out C:\Demo\Output\Sample.xlsx
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The filepath of Excel file	C:\Demo\Output\Sample.xlsx

Parameters for ExportExcel

Importing Excel

To import Excel to a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ImportExcel** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ImportExcel -project C:\Demo\Demo.vpp -file C:\Demo\input\sample.xlsx
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the Excel file to import	C:\Demo\input\sample.xlsx

Parameters for ImportExcel

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating ORM code and/or database

Generation of ORM code and database can be done through the command line interface. To do this:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **GenerateORM** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   GenerateORM -project C:\Demo\Demo.vpp -out C:\Demo\Output -code -ddl
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	Folder for storing the generated files including the source code, required libraries and mapping XML	C:\Demo\Output
-db [optional]	The type of database management system of the target database. When specifying this option, please enter the number of the corresponding database listed below. 1: MySQL 2: MySQL 5.0.5 3: MariaDB 4: MS SQL Server 5: MS SQL Server 2008 6: Oracle 7: HSQL 8: Sybase ASE 9: Sybase SQL Anywhere 10: PostgreSQL 11: Cloudscape/Derby 12: Derby 10.7 13: DB2 14: Ingres 15: Ingres 10 16: OpenEdge 17: Informix 18: Firebird 19: FrontBase 20: OpenBase 21: Cache 22: SQLite 23: H2	1
-code [optional]	Include to generate code.	-code
-ddl [optional]	Include to export DDL	-ddl
-exportdb [optional]	Include to export database	-exportdb
-dbmode [optional]	The action that you want to execute upon your database. Here is a list of possible actions: - Create - Update - DropAndCreate - Drop	Create

Parameters for GenerateORM

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generating document through command line

To generate HTML/PDF/Word document from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **GenerateDocument** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   GenerateDocument -project C:\Demo\Demo.vpp -out C:\Demo\Output\Mydocument.pdf -type pdf -all
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The file or folder path of generated document file(s)	C:\Demo\Output\Mydocument.pdf
-type	Type of document to generate. Here are the possible options: • html • pdf • word	pdf
-all [optional]	By default, only the selected diagrams (saved in vpp) will be covered when generating document. By including -all, all diagrams will be covered.	-all

Parameters for GenerateDocument

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Generator

To generate code from a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **InstantGenerator** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   InstantGenerator -project C:\Demo\Demo.vpp -out C:\Demo\Output
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The folder path of generated source files	C:\Demo\Output
-template [optional]	The path of template folder. Unless specified, the default folder will be selected.	C:\MyTemplates
-lang [optional]	Specify the language to generate. Unless specified, the lastly selected language (saved in project file) will be generated. Here are the possible options: • Java • C# • VB .NET • PHP • ODL • ActionScript • IDL • C++ • Delphi • Perl • XSD • Python • Objective-C • Ada95 • Ruby	C++
-package [optional]	The package(s) to be included in code generation. Usage: -package {fully-qualified-package-list} You may use ";" as separator, and note that all the sub-packages will be included.	com.mypackage.model;com.mypackage.view
-class [optional]	The class(es) to be included in code generation. Usage: -class {fully-qualified-class-list} You may use ";" as separator.	com.mypackage.model.Account;com.mypacka

Parameters for InstantGenerator

> **NOTE:** Code generation through command line generates only classes selected when running Visual Paradigm. In other words, you must at least generate once in Visual Paradigm in order to make command line generation work.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Reverse

To reverse source code to a project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **InstantReverse** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   InstantReverse -project C:\Demo\Demo.vpp -path C:\Demo\MyProject\src -lang Java -pathtype folder -sourcetype source
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-path	The file or folder path of the source files to be reversed	C:\Demo\Output\src
-lang	Specify the language of the source code to reverse. Here are the possible options: • Java • "C++ Source" • ".NET dll or exe files" • "CORBA IDL Source" • Ada 9x Source" • XML • "XML Schema" • Hibernate • "PHP 5.0 Source" • "Python Source" • Objective-C	Java
pathtype	Useful only for Java, pathtype defines the type of the path supplied for -path. Here are the possible options: • file • folder • zip	file
sourcetype	Useful only for Java, sourcetype defines the type of source to reverse. Here are the possible options: • source • class	source
-overwrite | -update	Overwrite or update model from code	- overwrite

Parameters for InstantReverse

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Java code synchronization

To perform synchronization between model and Java code through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **JavaCodeSync** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   JavaCodeSync -project C:\Demo\Demo.vpp -src C:\Demo\MyProject\src -generate
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-src	The folder path of the source file	C: \Demo \Output \src
-generate | -reverse	Action to perform. Include -generate to indicate the update of code from model. Include -reverse to indicate the update of model from code.	- generate

Parameters for JavaCodeSync

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Project Publisher

To publish project through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **ProjectPublisher** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the **bin** folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ProjectPublisher -project C:\Demo\Demo.vpp -out C:\Demo\Output
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	The folder path of the files to be published	C:\Demo\Output
-nickname [optional]	By default, Project Publisher produces content for the active nickname. You can make it published a specific nickname by specifying the nickname parameter.	Spanish

Parameters for ProjectPublisher

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Updating teamwork project from server

To update Teamwork project from server through command line:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **UpdateTeamworkProject** and paste to the bin folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   UpdateTeamworkProject -project "C:\vpworkspace\teamwork_client\projects\MarketManagementSystem\MarketManagementSystem.vpp" -workspace "C:\vpworkspace"
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-workspace	The path of workspace of the supplied project	C:\vpworkspace

Parameters for UpdateTeamworkProject

Related Resources

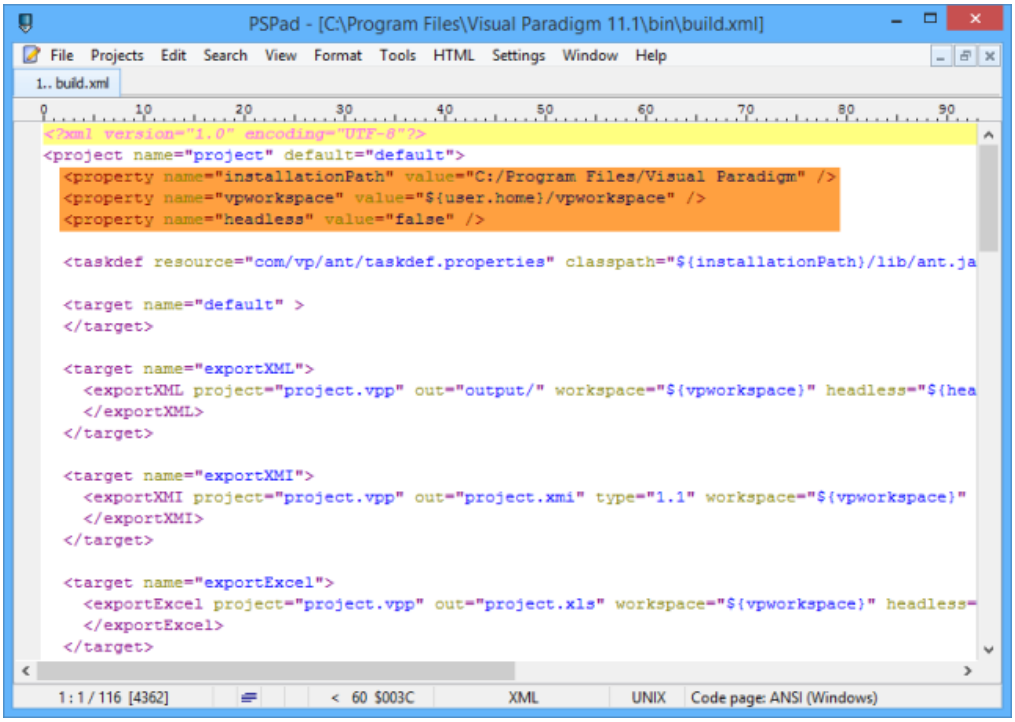
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Executing operations with Apache Ant in Visual Paradigm

Apache Ant is a software tool for automating software build processes. It is written in the Java language and is primarily intended for users with Java. If you are not familiar with Ant, you can find more information about it at Ant's webpage. To execute commands with Ant:

1. Browse the scripts folder under the Visual Paradigm installation directory.
2. Copy the script file **build.xml** and paste to the bin folder of Visual Paradigm installation directory.
3. Open the build file in any text editor. Modify the properties **vpsuiteInstallationPath**, **vpproduct**, **vpworkspace** and **headless** to suit your environment.



To modify basic properties in build.xml

4. Modify task(s) specific parts by changing the values of parameters. For details about the parameters, refer to previous sections.

Modify task(s) specific properties in build.xml

5. Save the changes and exit.
6. Start the command prompt and navigate to the bin folder of Visual Paradigm installation directory.
7. Enter **ant build.xml**, and then the task name to execute specific task.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Exporting document through command line (Doc. Composer)

To export documents created by Document Composer through command line:

1. Browse the **scripts** folder under the Visual Paradigm installation directory.
2. Copy the script file **ExportDocComposer** and paste to the **bin** folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   ExportDocComposer.bat -project "C:\Demo\Demo.vpp" -diagram "Doc1" -out "C:\Demo\Output" -type word
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-diagram	The name of the document(s) to be exported. You can provide a list of diagram like "Doc1" "Doc2", or enter "*" to export all document, or enter @diagramlist.txt to let the batch read the diagram list from file diagramlist.txt placed in bin folder.	"Doc1" "Doc2"
-out	The folder to store the exported file	C:\Demo\Output\
-type	Type of document to generate. Here are the possible options: • html • pdf • word	word

Parameters for ExportDocComposer

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Committing project through command line

To commit project through command line:

1. Browse the **scripts** folder under the Visual Paradigm installation directory.
2. Copy the script file **CommitTeamworkProject** and paste to the **bin** folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   CommitTeamworkProject.bat -workspace "C:\myworkspace" -project " C:\myworkspace\teamwork_client\projects\MyProject\ MyProject.vpp"
   Below is a description of parameters:

Parameter	Description	Example
-workspace	Workspace folder	C:\myworkspace
-project	Project path	C:\myworkspace\teamwork_client\projects\MyProject\MyProject.vpp

Parameters for CommitTeamworkProject

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Running Plug-in without starting Visual Paradigm

You can write plug-in for executing user-defined actions in Visual Paradigm. Generally, plug-in runs when Visual Paradigm starts. However, you can also run a plug-in without startubg Visual Paradigm.

To run a plug-in without starting Visual Paradigm:

1. Browse the **scripts** folder under the Visual Paradigm installation directory.
2. Copy the script file **Plugin** and paste to the **bin** folder of Visual Paradigm installation directory.
3. Start the command prompt.
4. Navigate to the bin folder of Visual Paradigm installation directory.
5. Execute the script by supplying the required parameters. For example:
   Plugin.bat -project "C:\Demo\Demo.vpp" -pluginid "sample.plugin"
   Below is a description of parameters:

Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-pluginid	The ID of the plugin you want to run. It's the id defined in the <plugin>, in plugin.xml.	"sample.plugin"
-pluginargs	Optional arguments of plugin.	

Parameters for Plugin

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Tips and tricks, Q&A and solutions about Visual Paradigm plug-ins
- Contact us if you need any help or have any suggestion

Printing diagrams

This chapters covers the steps needed to print diagram(s) to printing devices.

Printing diagrams

Introduces the standard way of printing diagram with description to several printing configuration.

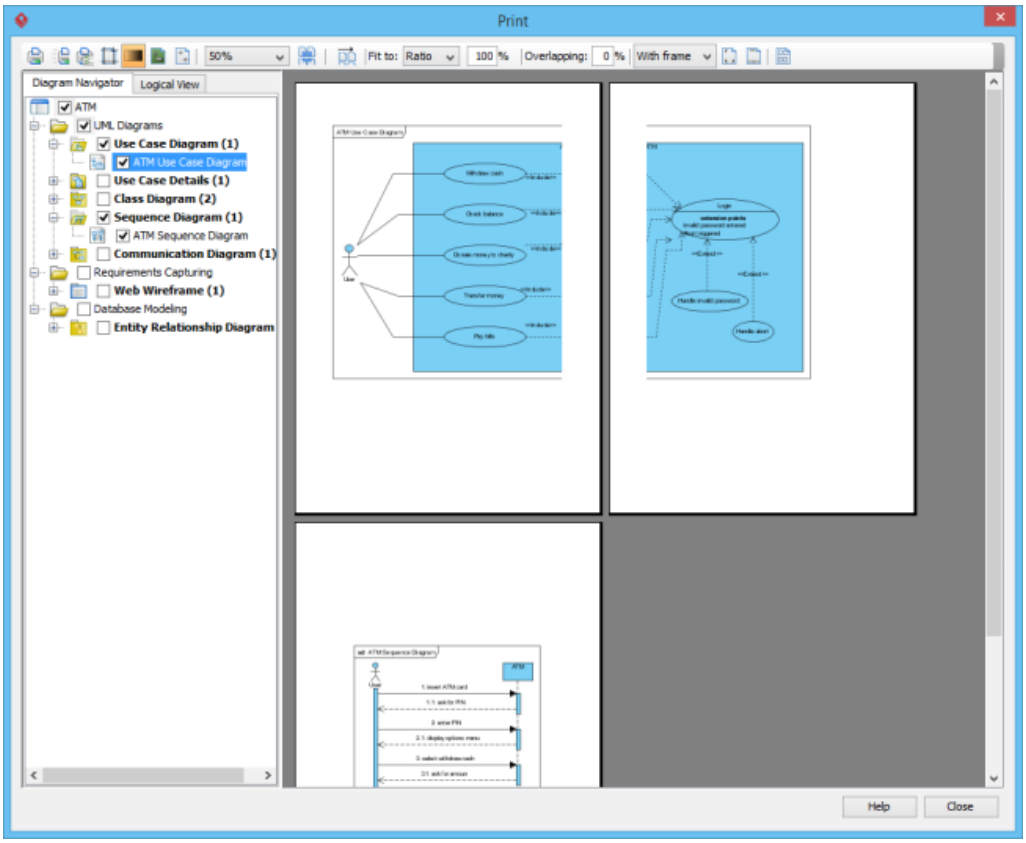
Quick print

Quick print is a lite way of printing diagram by ignoring some of the configuration options and preview of outcome.

Printing diagrams

The **Print** window dialog box allows you to preview the printout and provides a set of options for changing the printout style. To unfold the **Print** window dialog box, select **File > Print...** from the toolbar.

An overview of Print window



The Print window

The toolbar of the print preview pane allows you to configure the print settings. The buttons and their descriptions are shown in the table below:

Button	Name	Description
	Print	Print the diagram(s). The **Print** dialog box will be opened.
	Quick Print	Print the diagram(s) without previewing them. The **Quick Print** dialog box will be opened.
	Page Setup	Set up the page properties, such as paper size and orientation.
	Adjust Margin	Adjust the margins of the pages.
	Use Gradient Color	Select the use gradient color in printout. Since printing gradient color will use up lots of memory, it is recommended to turn this option off for better performance.
	Print Diagram Background	Click to print diagram's background when printing. When un-clicked, background color is ignored in printing.
	Global Page Number	Page number can be optionally displayed in printout. By default, the numbering of pages is diagram based, meaning that each diagram has its own set of numbers and the numbers reset for a new diagram. The **Global Page Number** option is to enable the continuous numbering of all diagrams.
50%	Zoom	Select the percentage to reduce/enlarge the print preview of diagrams.
/	Paper Base Layout/ Diagram Base Layout	If the **Fit to Pages** option is chosen, and there are multiple pages in the printout, choosing **Paper Base Layout** will cause the distribution of pages to be paper-oriented (the diagram size is ignored in arranging the preview); while choosing **Diagram Base Layout** will cause the distribution of pages to be diagram-oriented. Note that this option affects the preview only; the order of the printout remains unchanged.

/	Paper Place Style	Change the order of the printout. A large diagram is divided into many pages, choosing **From left to right** will arrange the printout order from the pages on the left to the pages on the right, while choosing **From top to bottom** will arrange the print order from the pages on the top to the pages on the bottom.
Fit to: Ratio 100 %	Fit to Ratio	Set the diagram size to fit to the specified ratio.
Fit to: Pages 1 x 1	Fit to Pages	Set the diagram to be printed on the specified number of pages.
Overlapping: 0 %	Overlapping	Set the percentage of the margins to overlap among adjacent pages.
With frame	Border	Determine whether to add frame or boder around diagram in printout.
	Show/ Hide Clip Marks on Page	Select/ deselect to show/hide the clip marks on the printout.
	Edit Header/ Footer	Edit the header and the footer of the printout.
	Multiple Page Mode	Switch to the Multiple Page Mode to set the multiple page options.

Details of toolbar

Printing a diagram with preview

You can use the Print command to select the printer. Set the range of pages and number of copies to be printed.

1. Select the desired diagram(s) for printing. The selected diagram(s) will be shown in the preview area.

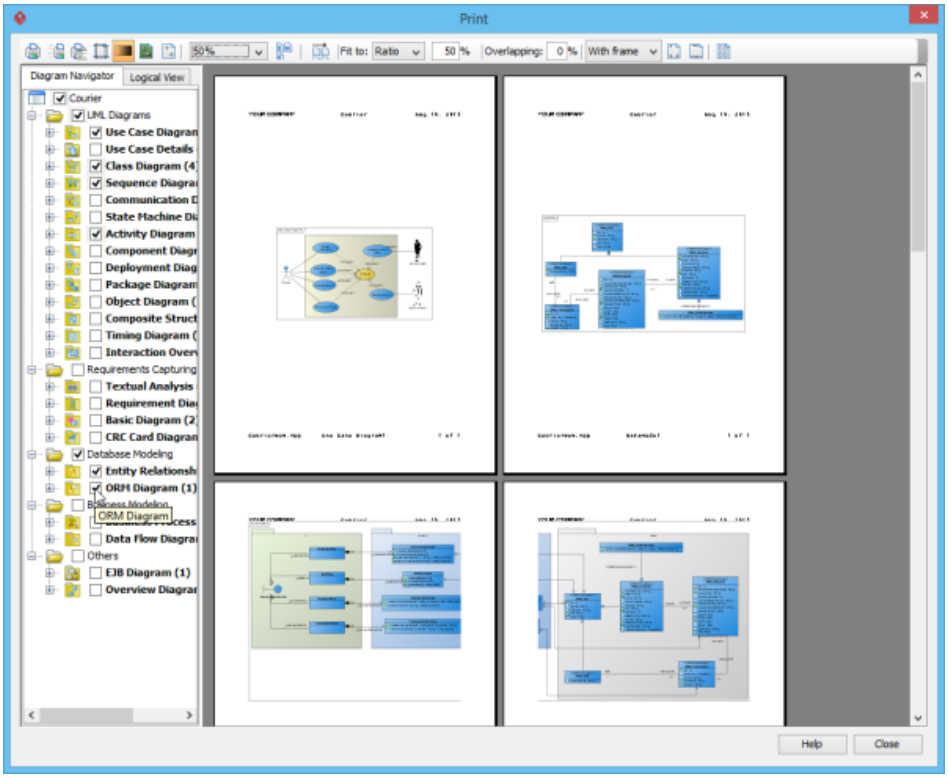
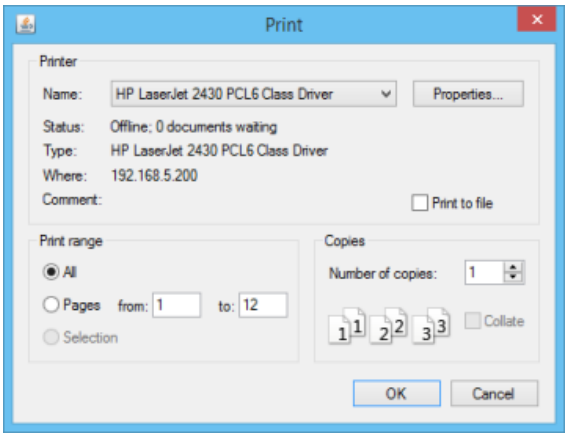


Diagram preview

2. Click the **Print** button on the Print preview toolbar. The **Print** dialog box will be shown.

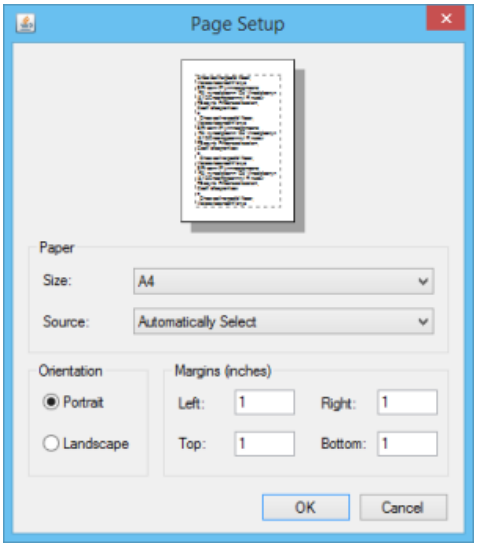


The Print dialog box

3. Select a specific printer, the page range and the number of copies to be printed. You may click the **Properties...** button to configure the printer-specific properties as well.
4. Click **OK** to start printing.

Page setup

Page Setup allows you to specify the page size, orientation, as well as the margins of the pages.

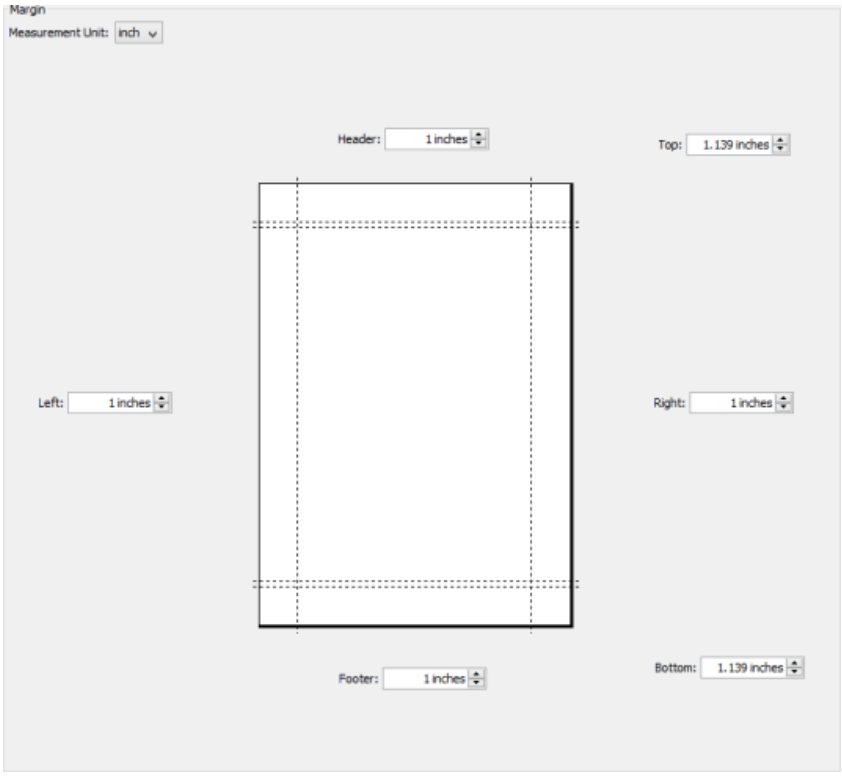


The Page setup dialog box

1. Click the **Page Setup** button on the toolbar. The **Page Setup** dialog box will appear.
2. You can click the **Size** drop-down menu to select the paper size for printing.
3. You can check either **Portrait** or **Landscape** udner **Orientation**.
4. You can enter the value into the **Left**, **Right**, **Top** and **Bottom** text fields to adjust the size of the corresponding margin.
5. Click **OK** to confirm the settings.

Adjusting margins

The Margins pane allows user to specify the margins of the pages, header and footer.



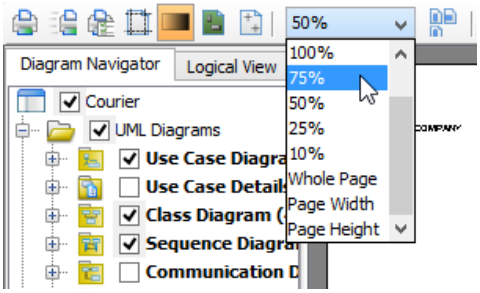
Adjusting margins

1. Click the **Adjust Margin** button on the toolbar. The margin setting page will be shown in the preview area.
2. You can edit the margins size by entering the sizes into the text fields. Alternatively, click the spinner buttons to increase/ decrease the margin sizes.
3. Click the **Finish Adjust Margin** button when you finish configuring the margin settings. The margin sizes will then be updated.

Zooming pages

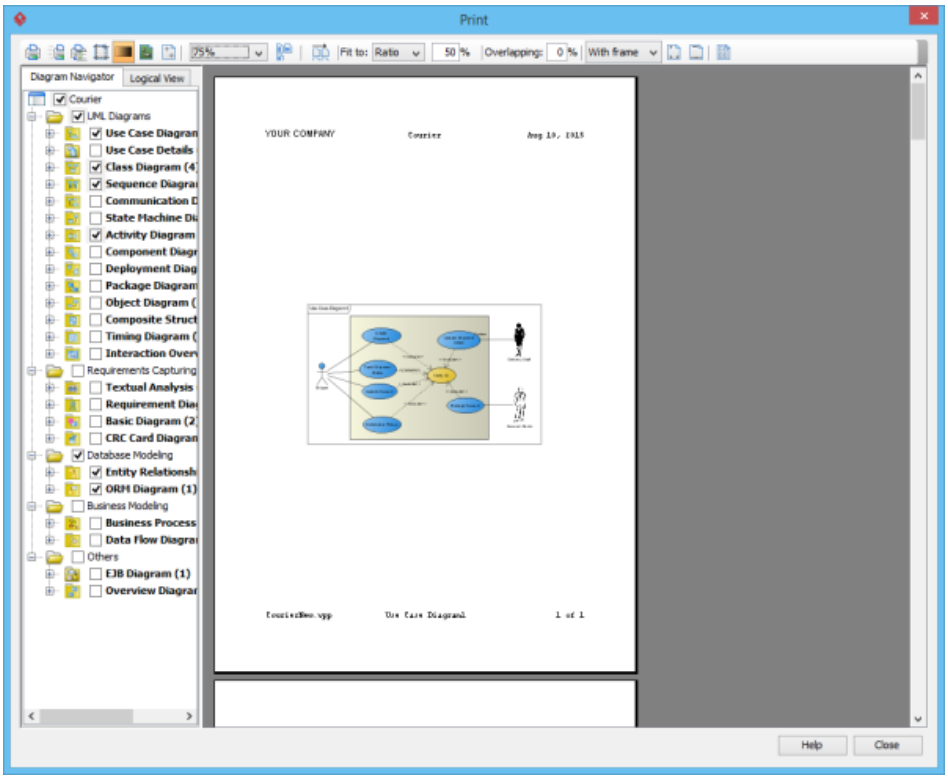
Diagrams can be zoomed in or zoomed out according to the user's preference.

1. Click the **Zoom** drop-down menu to select the desired zoom ratio.



Set zoom ratio

2. The preview area will show the diagrams in the zoom ratio that you have selected.



Previewing printout

Selecting the preview layout

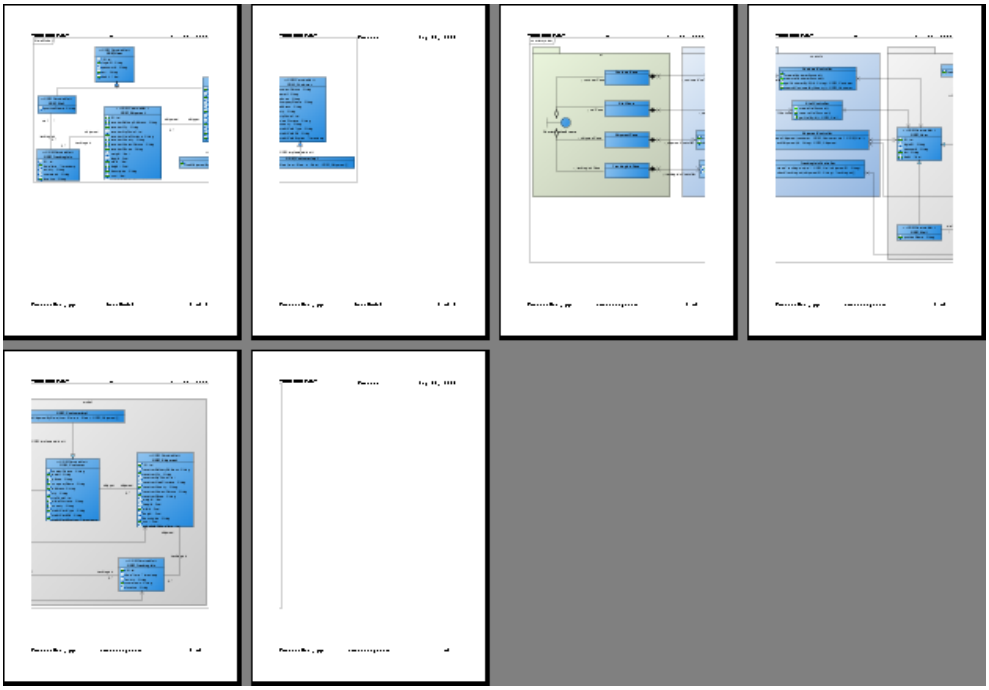
There are two layouts that you can choose for the print preview, the **Paper Base Layout** and the **Diagram Base Layout**.

If the **Fit to Pages** option is chosen and there are multiple pages in the printout, choosing **Paper Base Layout** will cause the distribution of pages to be paper-oriented (the diagram size is ignored in arranging the preview); while choosing **Diagram Base Layout** will cause the distribution of pages to be diagram-oriented.

Note that this option affects the preview only; the order of the printout remains unchanged.

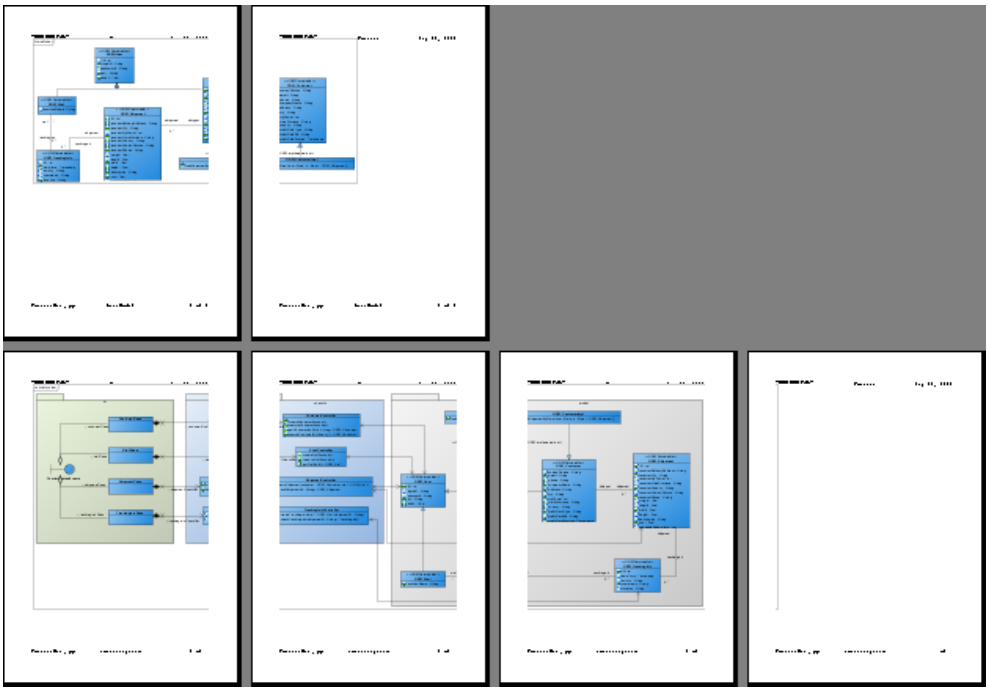
To select a layout of the preview, click the **Paper Base Layout** button or **Diagram Base Layout** button on the toolbar. A pop-up menu where you can choose the layout will appear.

The preview after applying the Paper Base Layout:



Preview in paper base layout

The preview after applying the Diagram Base Layout:



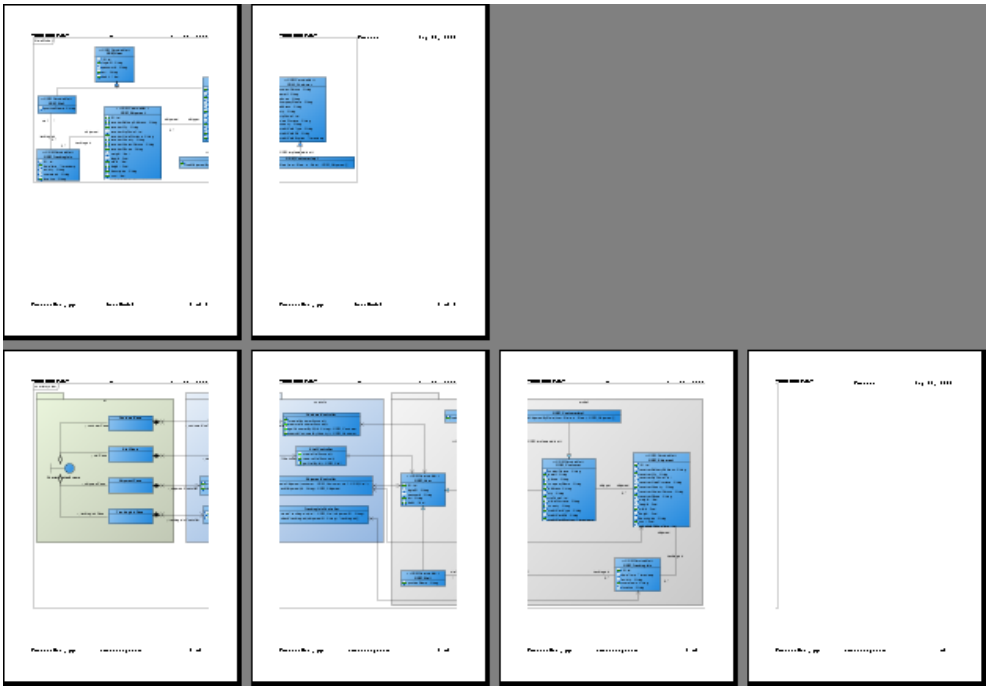
Preview in diagram base layout

Setting paper place style

You can select the paper place style to change the order of the printout. To select the paper place style, click the **Paper Place Style** button on the toolbar. A pop-up menu where you can choose a paper place style will appear.

Considering a large diagram is divided into many pages, choose **From left to right** will arrange the printout order from the pages on the left to the pages on the right, while choosing **From top to bottom** will arrange the print order from the pages on the top to the pages on the bottom.

The order of the printout after choosing **From left to right**:



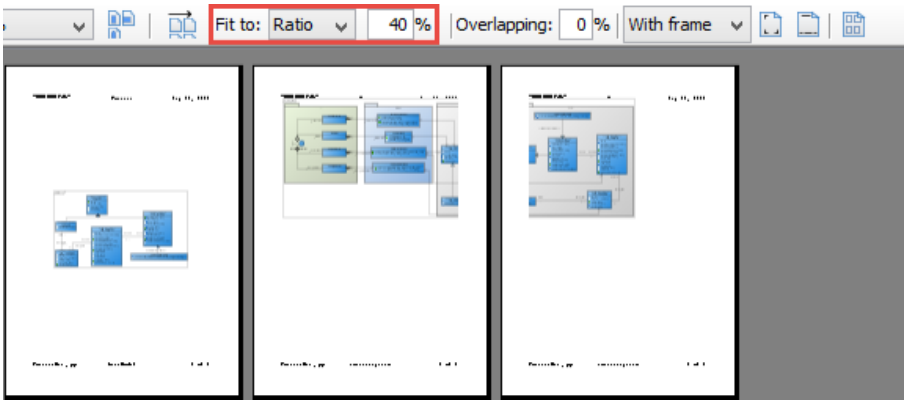
Printout order is left to right

Fit to ratio

Fit to Ratio is used to resize the diagrams in the printout to a specific ratio.

Click the **Fit to** drop-down menu and select **Ratio**.

You can enter the ratio into the text field. After editing the ratio, the diagrams in the printout will be resized at once.

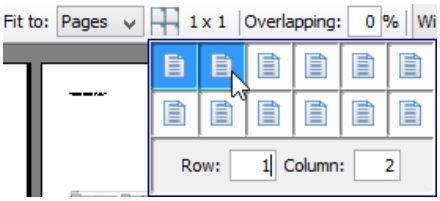


Fit to ratio

Fit to pages

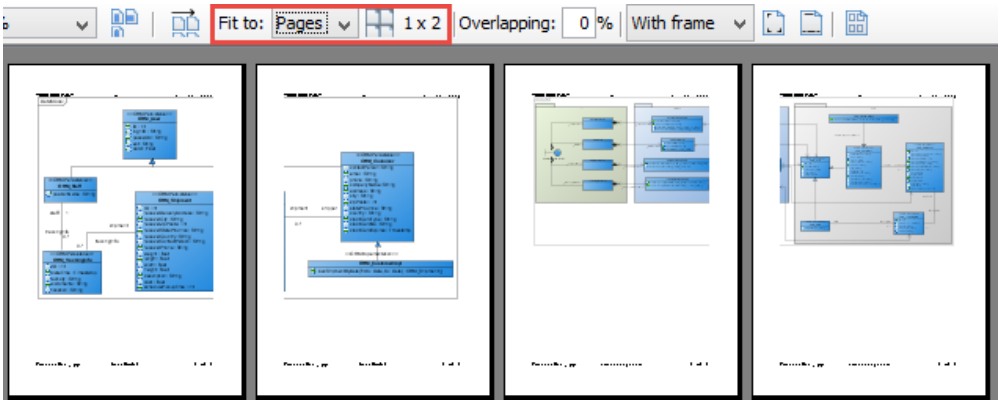
Fit to Pages is used to split the diagram to a desired number of pages when printing.

1. Click the **Fit to** drop-down menu and select pages.
2. Click the **Multiple Page Mode** button on the toolbar. The page selector will appear.



Select multiple pages

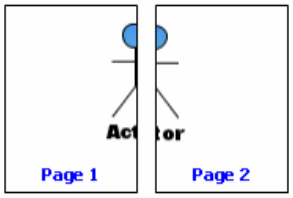
3. Click the row-column combination to select it (note that you can click and drag on the page selector to extend the selection). The diagram will be split into multiple pages by the rows and columns that you have selected.



Fit to page

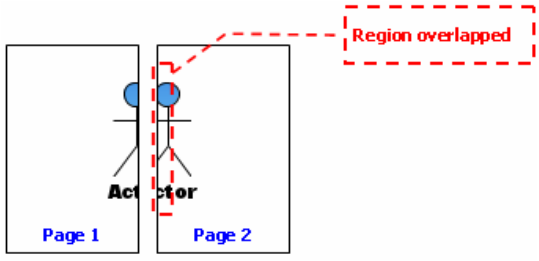
Setting the diagram overlap percentage

Overlapping is used when users want the diagrams to be overlapped at the boundaries between pages. This is particularly useful when you have a large diagram that span multiple pages and you want to stick the pages of the printout together; the overlapping area can then be used as a hint when sticking the pages.



Multiple page without overlap

1. Input the overlapping percentage and press **Enter** in **Overlapping** text field.
2. The printing area near the boundaries of the pages will be duplicated through the input value of overlapping percentage.



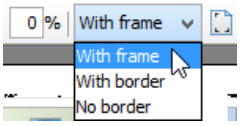
Multiple page with overlap

Printing with frame/Border option

You can print your diagram with a frame or border. There are three options available:

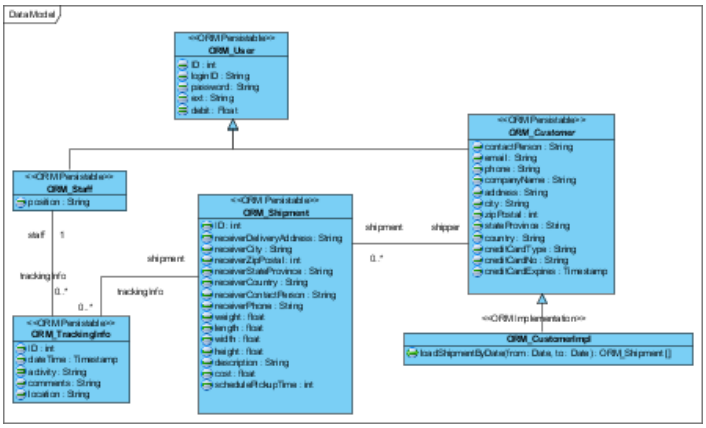
- With frame
- With border
- No border

Select **With frame**/ **With border**/ **No border** option from the drop-down menu.



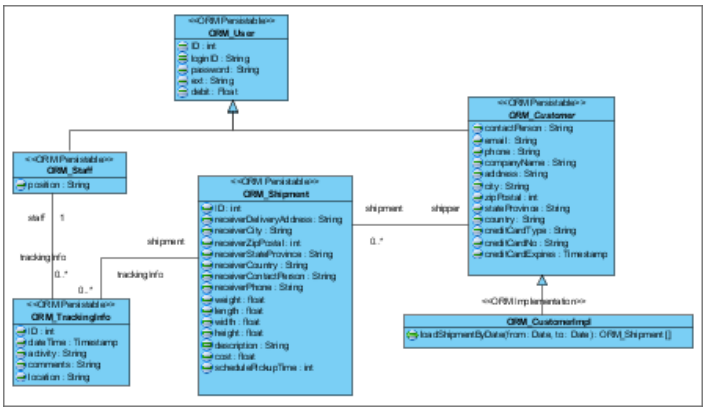
Select an option from drop-down menu

Output of printing with frame:



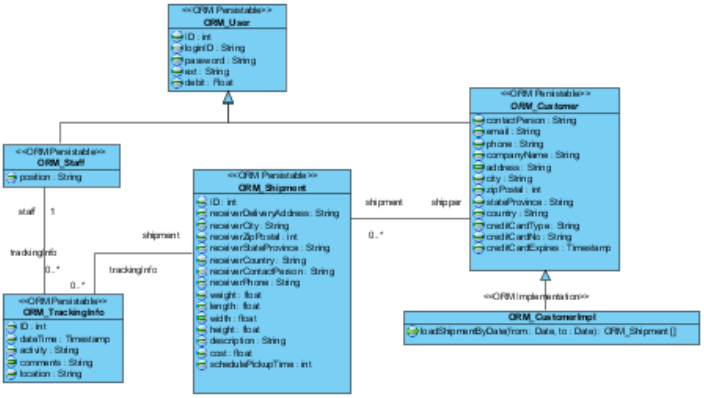
Printing with frame

Output of printing with border:



Printing with border

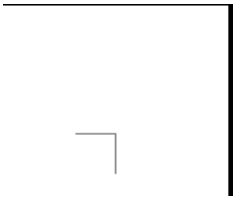
Output of printing with no border:



Printing with no border

Showing/Hiding clip marks on page

Clip marks act as an indication of the boundary of a page.

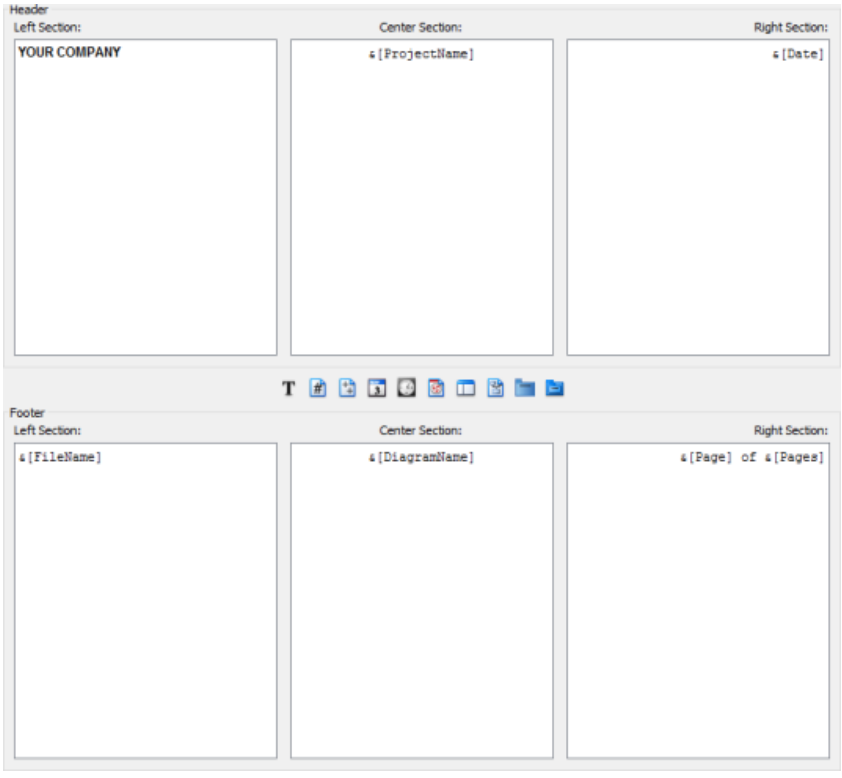


Clip marks

To show clip marks on the printout, click the **Show Clip Marks on Page** button . The boundaries of the pages will be surrounded by clip marks. To hide the clip marks, click the **Hide Clip Marks on Page** button again.

Editing header/footer of the pages

To edit the header/ footer of the printout, click the **Edit Header/Footer** button on the toolbar. You will then switch to the edit header/footer pane.



Editing header/footer of pages

You can edit the header and the footer in the Header panel and the Footer panel respectively. Each of the panel consists of the Left Section, Center Section and the Right Section, which represents the position that the content will be located in the header/footer.

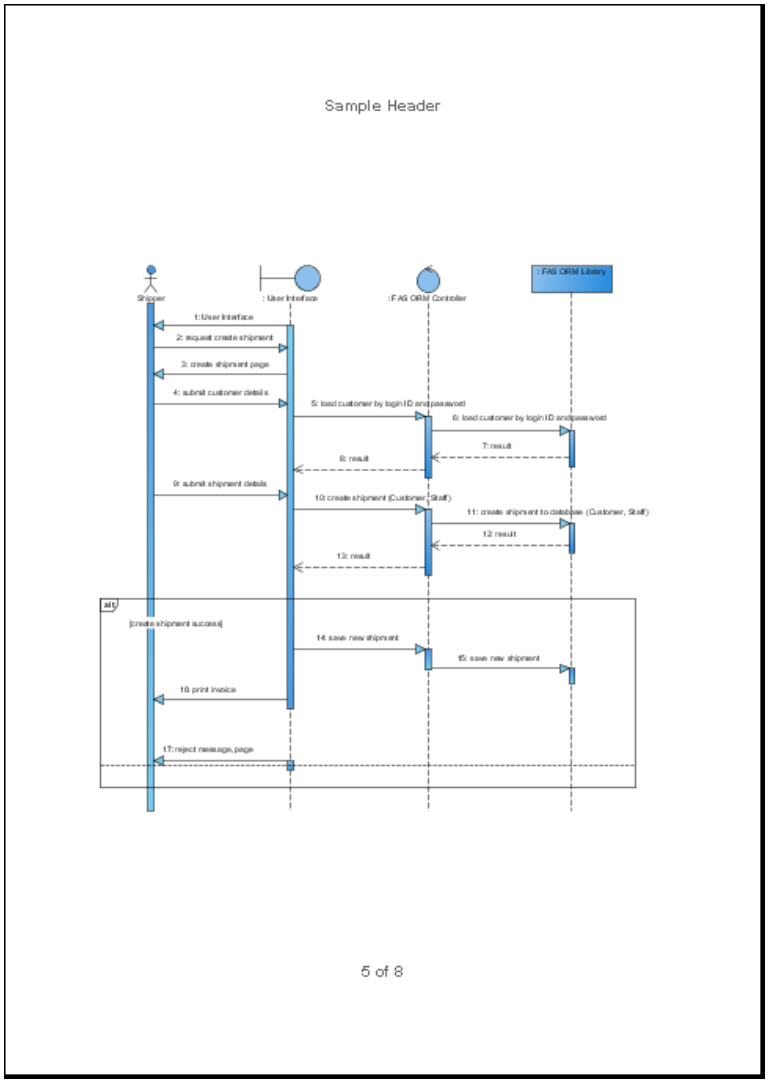
There is a toolbar between the Header panel and the Footer panel, which facilitates the editing of header/footer. The description of the buttons in the toolbar can be found in the following table:

Button	Name	Description

	Select Font	Select the font format. Note that you have to click the section you want its font to be formatted before you start setting the font format.
	Insert Page Number	Insert the page number. Note that you have to click the section you want page number to be inserted into before you click it.
	Insert Number of Pages	Insert the total number of pages. Note that you have to click the section you want the number of pages to be inserted into before you click it.
	Insert Date	Insert the date that the printing starts. Note that you have to click the section you want the date to be inserted into before you click it.
	Insert Time	Insert the time that the printing starts. Note that you have to click the section you want the time to be inserted into before you click it.
	Insert File Name	Insert the file name of the Visual Paradigm project. Note that you have to click the section you want the file name to be inserted into before you click it.
	Insert Project Name	Insert the name of the Visual Paradigm project. Note that you have to click the section you want the project name to be inserted into before you click it.
	Insert Diagram Name	Insert the diagram name. Note that you have to click the section you want the diagram name to be inserted into before you click it.
	Insert Parent Package	Insert the parent package. Note that you have to click the section you want the parent package to be inserted into before you click it.
	Insert Parent Hierarchy	Insert the parent hierarchy. Note that you have to click the section you want the parent hierarchy to be inserted into before you click it.

The description of editing of header/ footer toolbar

After you have finished editing the header/footer, click the **Close Edit Header/Footer** button to switch to the print preview mode. A sample page with the header and footer is shown in the figure below:

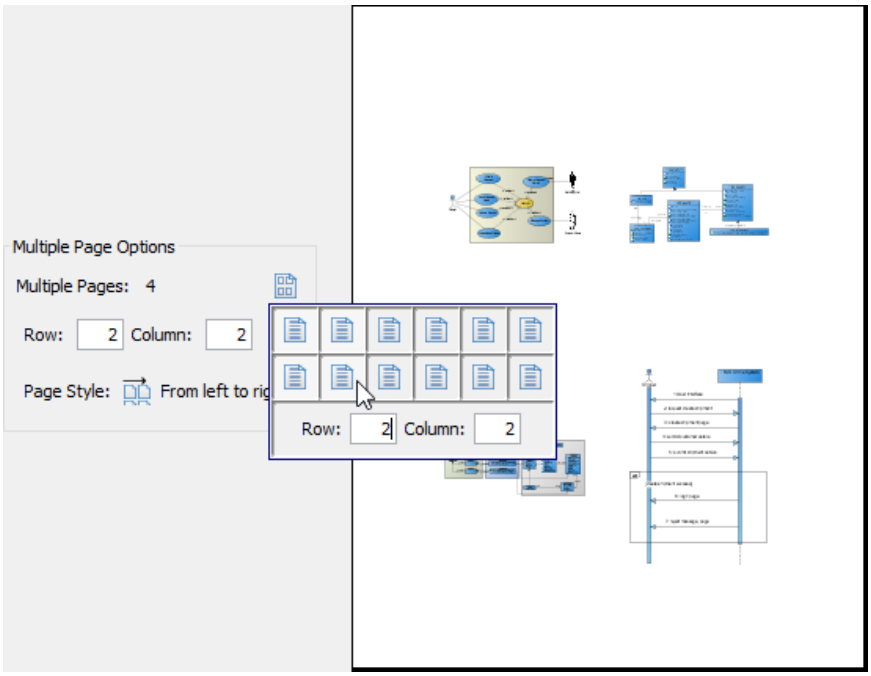


Page with header and footer

The multiple page mode

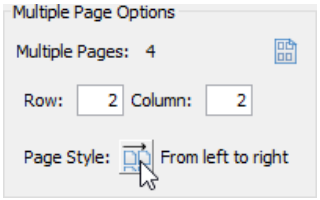
The Multiple Page Mode allows users to configure how the diagrams should be distributed in multiple pages. To switch to the Multiple Page Mode, click the **Multiple Page Mode** button on the toolbar.

Click the icon behind **Multiple Pages** will pop the page selector out, where you can select the row-column combination for the printout. Alternatively, you can type in the **Row** and **Column** text field directly.



Select multiple pages

Click the icon behind **Page Style** to change the printout order. Considering a large diagram is divided into many pages, choosing **From left to right** will arrange the printout order from the pages on the left to the pages on the right, while choosing **From top to bottom** will arrange the print order from the pages on the top to the pages on the bottom.



Distributes diagram in multiple page

After you have finished configuring the multiple page settings, click the **Close Multiple Page Mode** button to close it.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Printing a diagram with Quick Print

In Visual Paradigm, you can print a diagram with simple and easy setting by using quick print feature. The Quick Print feature allows you to print diagrams without previewing them, hence speeding up the print job.

Printing with quick print

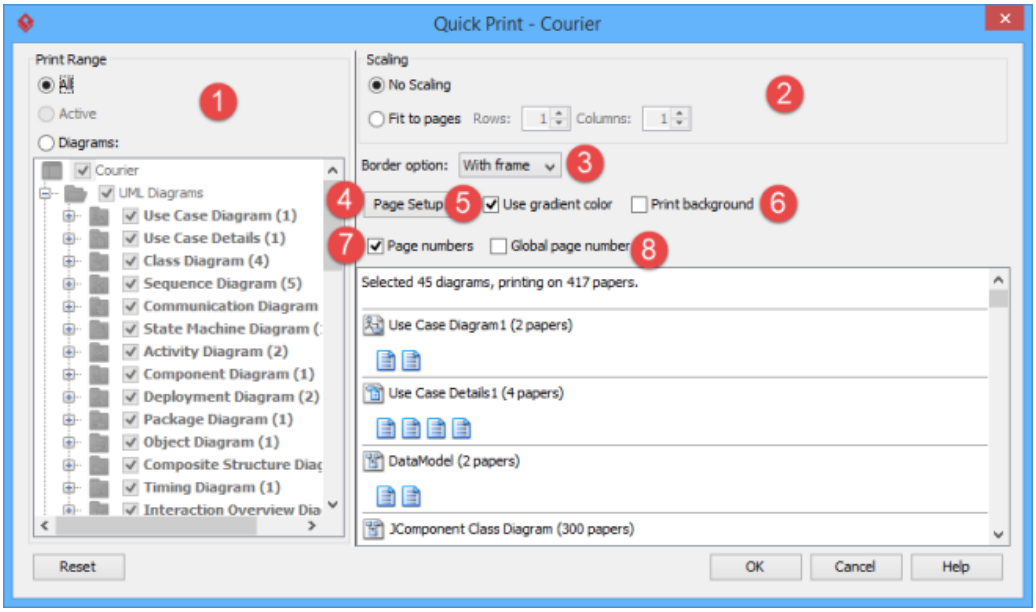
To perform quick print:

1. Select **File > Quick Print...** from the toolbar.

> **NOTE:** Alternatively, you can select **Print > Quick Print...** on the toolbar to launch quick print.

2. In the **Quick Print** dialog box, select printout setting for the diagram.
3. Click **OK** button to proceed printing.

The overview of Quick Print



***Quick Print** dialog box*

No.	Name	Description
1	Print Range	Click on one of the options to specify the print range: • All - Print all the diagrams within the current project. • Active - Print only the active diagram. • Diagrams - Check from the diagram tree to select the diagram(s) for printing.
2	Scaling	Select **No scaling** to print with diagrams' original size. Numbers of pages used for each diagram are subject to the scale of diagrams. Select **Fit to pages** to print with specified number of pages per diagram with respect to the specified number of rows and columns.
3	Border option	Select a border option of printout.
4	Page Setup...	It allows you to specify the page size, the orientation as well as the margins of the pages.
5	Use gradient color	Select the use gradient color in printout. Since printing gradient color will use up lots of memory, it is recommended to turn this option off for better performance.
6	Print background	Click to print diagram's background when printing. When un-clicked, background color is ignored in printing.
7	Page numbers	Select it to print diagrams with page number.
8	Global page number	Page number can be optionally displayed in printout. By default, the numbering of pages is diagram based, meaning that each diagram has its own set of numbers, and the numbers reset for a new diagram. The Global Page Number option is to enable the continuous numbering of all diagrams.

Details of quick print dialog box

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

General Options

Project Options

Option Name	Description
Default author for non-teamwork project	The name of person who edit the project. This will be filled in as the author within project management properties.
Auto save project	Save the opening project as backup periodically.
Auto save interval (mins)	Time needed to wait until the next auto saving.
Delete model element when deleting its view	Action that happens when all views of a model element are being deleted: • Ask - (default) Prompt if you want to delete the model element as well. • Delete without asking - Delete the model element as well. • Don't delete - Do not delete the model element.
Open exported image file	Action that happen after exporting images • Ask - (default) Prompt if you want to open the exported image file. • Yes - Open image file (when export single image) or image folder (when export multiple images) directly. • No - Do not take any action.
Backup level	The number of files that will be saved as backup. When reached the limit, the next backup will overwrite the earliest one.
Confirm diagram deletion	(default true) Prompt if you really want to close diagram.
Confirm shape deletion	(default true) Prompt if you really want to delete a shape on diagram.
Never delete model elements with views	(default false) If a model element has multiple views, delete any of them will never cause the model element to be deleted, even if you are deleting a master view.
Open last project on startup	(default true) If checked, it will open the last opened project immediately when starting Visual Paradigm. If unchecked, it will open new project.
Export as image with frame	• None - (default) Do not surround the diagram with neither frame nor border. • Export with frame - Add a frame around image to show a border with the name of diagram appear at top left of diagram. • Export with border - Add a thin border around image.
Show password in lock dialog pane	
Lock by password	

Project Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Referenced Project Options

Option Name	Description
Duplicate Model from Linked Project	Duplicating a model element from linked project will cause a local copy to be created. • Prompt - (default) Prompt if you really want to duplicate. • Yes - Confirm duplication. A local copy of selected shape(s) will be created. • No - Discard duplication.
Show warning when failed to add child to model from linked project	You cannot place a shape inside a linked shape, such as a linked package. If you try to do this, you will be warned by default. This option determines whether the warning will appear or not.

Caption Here

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Appearance Options

Option Name	Description
Look and feel	It controls the appearance of application screen.
Theme (for Office 2003 L&F only)	The tone of the application screen.
User Language	Language being applied on the user interface. This affects the text in menus, tooltips, dialog content, report content, etc.
Change application font	Select the font family and size of font to apply for the application user interface.
Date Format	Format of date values that appears in the application.
Date Sample	The sample of date is shown after you select the format of date.
Time Format	Format of time values that appears in the application.
Time Sample	The sample of time is shown after you select the format of time.
Measurement Unit	• inch - (default) Set the measurement unit to be inch • cm - Set the measurement unit to be centimeters

Appearence Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Connection Options

Option Name	Description
E-mail	Enter the Email field to specify your email address.
Use proxy	(default false) To check/uncheck in order to use a proxy server for connecting to the Internet.
Host	The host of the proxy server.
Port	The port of the proxy server.
Login name	The user name of the proxy server (if the proxy server required the user to login).
Password	The password of the proxy server (if the proxy server required the user to login).
ElaborView server Connection - Connection Name	A name to describe the connection of ElaborView server.
ElaborView server Connection - Hostname	The host of the ElaborView server.
ElaborView server Connection - Port	The port of the ElaborView server.
ElaborView server Connection - Email	The Email to log into the ElaborView server.
ElaborView server Connection - Password	The Password to log into the ElaborView server.
ElaborView server Connection - Add	VP can store multiple ElaborView connection. You may add a new one by clicking Add and specify the connection properties after added.
ElaborView server Connection - Remove	Click to remove the current connection.
ElaborView server Connection - Test Connection	Click to test if the connection is valid or not.

Connection Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Printing Options

Option Name	Description
Use gradient color when print diagrams	Control whether to use gradient or solid color for shapes and diagrams when printing.
Print diagram background when printing diagrams	Control whether to print the diagram background color when printing diagram.

Printing Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Edition Options

Option Name	Description
Enable non-supported feature	(default true) When checked, executing features that are not available in the running edition will be prompted, asking whether you want to advance to higher edition in order to use the feature. When unchecked, those non supported features will be disabled.

Edition Options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Teamwork Options

Option Name	Description
Show Export/Import Teamwork project	(default false) Determines whether the export/import Teamwork project menus will appear in the Projects menu of Teamwork Client dialog box

Teamwork Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Update Options

Option Name	Description
Auto update	Let the application checks for available updates with respect to the running build/version and notify you to perform update whenever possible. • Never - Do not inform product update • On every start - Check for updates everytime when starting Visual Paradigm • Daily - Check for updates daily • Weekly - (default) Check for updates weekly • Monthly - Check for updates monthly

Update Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagramming Options

Appearance Options

Option Name	Description
Grid - Show grid	(default false) Show grid lines on diagram
Grid - Color	Color of grid lines.
Grid - Width	Determines the horizontal spaces between grid lines.
Grid - Height	Determines the vertical spaces between grid lines.
Grid - Snap to grid	(default true) When checked, shapes will be docked to the closest grid line when being created/ moved. Otherwise, shapes can be moved freely as if the grid does not exist.
Enable diagram alignment guide	(default true) Show alignment guide when moving shapes.
Global Pallet Option - Show name	(default true) Determines whether the name of items will be shown in the pallet.
Global Pallet Option - Expand group	(default false) Determines whether the group will be expanded to display all items.
Show Hidden Layer Indicator	• Yes - When move the mouse over a shape, the hidden layer indicator is shown • No - The hidden layer indicator isn't shown even when move the mouse over a shape • Prompt -Ask if users want to show hidden layer indicator when move the mouse over a shape
Glossary Terms	• Highlight - Underline the text in shape name or documentation if it is a term. • Specify color - The color of glossary term highlight • Following Shape Foreground Color - Use the same color as the shape foreground

Appearance Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Environment Options

Option Name	Description
Shape Selection Detection	• Inside selection area - (default) When selecting a range of shape, only shapes that are completely inside the selection range are included in selection • Overlapped with selection area - When selecting a range of shape, shapes that are partly or completely covered by the selection range are included in selection
Copy as XML with RTF style	• Yes - When copy Use Case, the rich text format of Use Case Details will also be copied (size of copied content will increase considerably) • No - When copy Use Case, Use Case Details will be copied as plain text • Prompt - Ask if user want to copy rich text for Use Case Details when copying XML
Delay until showing Quick Preview in Diagram Navigator (seconds)	• Never show - Never show Quick Preview when moving mouse cursor over diagram node in Diagram Navigator • 1.0 - 3.5 - The number of seconds that a Quick Preview will disappear after moving the mouse cursor out of a diagram
Default Copy cction	• Within Visual Paradigm - (default) When triggering the hotkey for Copy (Ctrl-C, by default), it will perform copying within Visual Paradigm • Copy to Clipboard as Image (JPG) - When triggering the hotkey for Copy (Ctrl-C, by default), it will perform copying as JPG image • Copy to Clipboard as Image (EMF) - When triggering the hotkey for Copy (Ctrl-C, by default), it will perform copying as EMF image
Show copy and paste tips	Check to cause the help contents to be displayed when you paste a view on a diagram.
Copy as image with frame	• Unspecified - (default) Prompt for adding a frame or a border when copying shapes as image • None - Do not add border nor frame to image when copy shapes as image • Copy with frame - Add a frame around image to show a border with the name of diagram appear at top left of diagram • Copy with border - Add a thin border around image
Show shape content when dragging	(default true) Show the shape content such as shape name when dragging shape.
Show Logical View selector when naming new diagram	When there is at least one logical view in your project, you can optionally select the logical view to store a diagram during the creation of diagram. If you want to turn this option off, uncheck here.
Show diagram element tooltip	Show the tooltip of shape when moving mouse pointer over shape
Prompt for clearing undo history before applying design pattern	(default true) Prompt for applying design pattern even when there are remaining undo or redo due to the undo and redo records will be cleared after applying design pattern.
Enable Mouse Gesture	Mouse gesutre enables you to create and connect elements by forming a gesture with the right mouse button.
Number of stereotypes shown in popup menu	You can assign stereotypes to a shape through its popup menu. This option determines the number of stereotypes to display in that popup menu.
Auto expand diagram borders	Sometimes, the diagram border may be wrongly calculated. You may click the Reset to default button to reset the range of diagram border.
Undo limit	The number of available undo that can be performed. The lower the limit, the higher the application performance. Actions that beyond the undo limit will not be able to be undo. If you want to have no limit, check **Unlimited**.

Environment Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model Generation Options

Option Name	Description
Generate Sequence Diagram from Use Case Description Overwrite Existing Diagram	• Yes - Automatically overwrite Sequence Diagram generated from Use Case Details when generate again • Prompt - (default) Prompt for overwriting Sequence Diagram generated from Use Case Details when generate again
Generate Diagram from Scenario Overwrite Existing Diagram	• Yes - Automatically overwrite diagram generated from Scenario when generate again • Prompt (default) Prompt for overwriting diagram generated from Scenario when generate again
Overwrite Flow of Events when Synchronize from Sequence Diagram	• Ask - (default) Prompt for overwriting Flow of Events in Use Case Details when Synchronize from Sequence Diagram • Yes - Overwrite Flow of Events in Use Case Details automatically when Synchronize from Sequence Diagram

Model Generation Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Shape Options

Option Name	Description
Initial Shape Size When Set to Display Image	• Fit shape to image - Resize the image placeholder to fit the selected image • Fit image to shape - (default) Resize the image to fit into the image placeholder
Add Covered Shapes as Children After Resize	• Yes - Automatically contain the covered shapes after resizing a container to cover shapes • No - Do not contain covered shapes after resizing a container to cover shapes • Prompt - (default) Ask if you want to contain covered shapes after resizing a container to cover shapes
Show Added Member in Other View	When you try to create a new member, like a column of an entity on a shape and if there is another view of the model element, you can set whether to show the member on other view as well. • Yes - Show the member on other view(s) • No - Do not show the member on other view(s) • Prompt - (default) Ask if you want to show the member on other view(s)
Auto Expand Parent When Moving Child	Determines whether to expand the parent when moving the shape around the parent's border.
Create new line key	• <Ctrl> + <Enter> - Press Ctrl-Enter to create a new line when inline editing • <Alt> + <Enter> - (default) Press Alt-Enter to create a new line when inline editing • <Enter> - Press Enter to create a new line when inline editing
Show unlimited level in preview shape (May slower the diagram display)	(default false) Determines whether a diagram overview will show contents for all nested diagrams.
Show rich text content in preview shape	(default false) Determines whether diagram overview will show rich text content.
Resize shape when add member	(default true) For shapes that have member, like class and entitiy, when you add a member, it will or will not resize the owner against the length of member, base on this option.
Auto resize parent when resizing child	(default true) When you resize a shape that makes it touches the boundary of parent, this option determines whether the parent will be expanded accordingly.

Shape Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Connector Options

Option Name	Description
Default pin from connection point	(default false) Automatically pin connector's from end when connector is being created.
Default pin to connection point	(default false) Automatically pin connector's to end when connector is being created.
Show relationship connectors for dropped models	(default true) Show connectors when dragging and dropping inter-related model elements/views from tree to diagram.
Auto relocate connector when overlapped with other shapes	(default false) Auto relocate connector when connector is being overlapped by another shape.
Scroll connector delay (second)	• No delay - Immediately scroll to the other side of connector • 1 - 9 - Time provided for scrolling to the other side of connector
Highlight selected connector	(default true) Increase the thickness of selected connector(s).

Connector Options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Resource Centric Options

Option Name	Description
Show resources	(default true) Show resource icons around shapes.
User interface	Resource Catalog - Since Visual Paradigm 12.2, the classic Resource-Centric Interface had been enhanced to Resource Catalog for better usability. Choosing this option allows you to use Resource Catalog. Classic - The classic mode of Resource-Centric. It provides you with a set of action icons around each shape. You can create and connect shapes through accessing those icons.
Show resources delay (second)	0 - 2 - Time needed to wait from having mouse cursor hover on shape till the resource icons appear.
Auto hide resource delay (second)	Time needed to wait the resource icons to disappear when mouse cursor is moved out of a shape.
Show group resources	(default true) Show group resources that appear when selecting multiple shapes.
Show extra resources	(default false) Show also uncommon resource icons.
Show generic resources only	(default false) Show generic resource but hide other resource icons.
Always show model element indicators	(default true) Always show the reference, subdiagram, transitor, documentation resource icon at the bottom of shape no matter whether the shape has reference/sub-diagram/transitor/ documentation added/defined.

Resource Centric Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Class Options

Option Name	Description
Show parent style	• UML - (default) Show tooltip of child class in UML style like *Package::Class* • Java/C# - Show tooltip of child class in Java/C# style like *Package.Class*
Show fully qualified class and package name in tooltip	(default true) Enable to show fully qualified class and package name in tooltip like *Package::Class*/ *Package.Class* (depending on the setting of **Show parent style**). Disable to show only the hovering class or package name and type like *Class : Class*.

Class Options details

Auto Attribute Type

Option Name	Description
Name	When an attribute entered matches with the name defined here, the type and default value will be automatically filled.
Type	Automatically set attribute type value when the name user entered for an attribute matches with name specified in Name.
Default Value	Automatically set default value when the name user entered for an attribute matches with name specified in Name.

Auto Attribute Type of Class Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generalization Options

Option Name	Description
Direction of creation	• from General to Specific - (default) When creating a generalization, the arrow head will appear at the mouse release side • from Specific to General - When creaNessageting a generalization, the arrow head will appear at the firstly selected shape

Generalization Options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ERD & ORM Options

Auto Column Type

Option Name	Description
Name	When a column entered matches with the name defined here, the type and default value will be automatically filled.
Type	Automatically set column type value when the name user entered for a column matches with name specified in Name.
Nullable	Automatically set column nullable when the name user entered for a column matches with name specified in Name.
Default Value	Automatically set default value when the name user entered for a column matches with name specified in Name.

Auto Column Type of Entity Options details

Behavior & Presentation

Option Name	Description
Show table record editor	(default true) Show the table record editor at the bottom of ERD for adding default table records to database tables.

Behavior & Presentation of Entity Options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Interaction Options

Option Name	Description
Show sequence diagram text editor	You can edit sequence diagram through the text editor pane that appears at the bottom of diagram. This option controls the visibility of editor.
Auto fit message completion size	(default true) Fit message completion box's size everytime you activate it. If disabdeterminsled, size adjusted manually won't be remembered.
Show operation documentation in message completion	(default true) When selecting an Operation in the message completion box, the documentation of operation will appear next to the completion box.

Interaction Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business Process Options

Option Name	Description
Invalid Connection Handling	• Ignore all - Ignore all invalid actions related to connecting shapes • Cancel move - Cancel invalid actions related to connecting shapes • Prompt - (default) Prompt for an action when an invalid actions related to connecting shapes is discovered
Show Lane Handle	• Auto - (default) Show horizontal/vertical Lane header only when horizontal/vertical Lane exist • Always Show - Always show both horizontal and vertical Lane headers even when Lane does not exist • Always Hide - Always hide Lane headers
Show convert Sub-Process/Task warning	(default true) Show warning when trying to convert between Sub-Process and Task.

Business Process Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Documentation Options

Option Name	Description
When insert custom cover page, show disable generate cover page warning.	In a document in Doc. Composer, there are two ways for you to add a cover page. One is to add a custom cover page and another one is to fill in the details of cover page in Documentation Properties. You can choose either way to add a cover page, not both. By checking this option, you will be prompted to disable the Document Properties option when you add a custom cover page.
Level detection	• Max Level - The maximum number of heading levels that you want to apply numbering. • Levels detection - Set the style correspond to each level. If you select Heading 1 to be Level 1, Doc. Composer will make Heading 1 numbered. • Level style - The numbering format to be applied to different levels.

Documentation Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

View Options

View Options

Option Name	Description
Diagram Navigator Sort Type	• Sort by name - (default) Sort tree nodes in Diagram Navigator by their names • Sort by type - Sort tree nodes in Diagram Navigator by their types
Model Explorer Sort Type	• No sort - Do not sort tree nodes in Model Explorer • Sort by name - (default) Sort tree nodes in Model Explorer by their names • Sort by type - Sort tree nodes in Model Explorer by their types
Show data types	(default false) Show Data Types node in trees.
Show relationships	(default false) Show relationships in trees.
Show sub diagrams	(default true) Show subdiagrams in trees.
Show Activation in Diagram Navigator	(default true) Show activations in Diagram Navigator.
Show project path in Diagram Navigator	(default false) Show project path in Diagram Navigator.
Sort elements in tree with case sensitive	(default false) Make sorting of tree nodes case sensitive (consider the case).
Show carriage return character	(default true) Show carriage return character for line breaks of shape names that are in multiple lines.

View Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Reverse Options

Instant Reverse Options

Option Name	Description
.NET	• Not specified - (default) Do not specify whether Instant Reverse of .NET is enabled or not. • Enabled - Enable Instant Reverse for .NET • Disabled - Disable Instant Reverse for .NET
C++	• Not specified - (default) Do not specify whether Instant Reverse of C++ is enabled or not. • Enabled - Enable Instant Reverse for C++ • Disabled - Disable Instant Reverse for C++
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files that will be reversed • Other - Specify an encoding for the source files that will be reversed
Show Instant Reverse Form Diagram dialog after Instant Reverse	(default true) Show the **Instant Reverse Form Diagram** dialog box after Instant Reverse, so that you can form diagram after reversing code into Visual Paradigm.

Instant Reverse Options details

Related Resources
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ORM Options

General Options

Option Name	Description
Warn when rename table/constraint to over 30 chars if Oracle is selected as default database	To make your design fits the Oracle requirement, you can check this option to let Visual Paradigm warn you when naming table/constraint with more than 30 characters.

General Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

State Code Engine Options

State Code Engine Options

Option Name	Description
Browse output directory after generate code	(default false) Open the directory of generated state code.

State Code Engine Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Office Exchange Options

Office Exchange Options

Option Name	Description
Remember Import Decision	VP detects changes made in exported document and will suggest you to import changes back to your project. This option determines whether the import will be on or off. • Yes - Enable the import option. • No - Disable the import option. • Not Specified - (default) You will be asked if you want to import the changes from document to Visual Paradigm whenever changes are detected.
Launch viewer	(default true) Open the document after export.

Office Exchange Options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

User Path Options

User Path Options

A user path is a variable that refers to a base path in user's computer. You can add a reference to local file using user path so that the reference refers to a file relative to a user path, instead of an absolute path. This means you can move references files to a different location, or even to a different computer, and can still open them as long as the user path value is up-to-date.

Option Name	Description
Show user path	(default false) Select to show user paths in references, instead of displaying resolved absolute paths. A user path is displayed with its name enclosed by ${ }.
Prompt to specify user path	(default false) When adding a reference comprises a path that is not defined as a user path, you will be prompted to add path as user path.

User Path Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

File Types Options

File Types Options

Model Reference allows you to add reference(s) to external file or URL into diagram element. You can open the referenced file or URL
to get more information of the model in later stage. In the Options dialog box, you can configure to use specific application or command to open different types of file and specify your favorite web browser to open a URL. The system default handling method will be used if you have not configure the application or command to handle a particular file type.

Configure application/Command for file types

To configure application/command for file types:

1. Press on the upper **Add...** button. This shows a dialog box where you can add file extension.
2. Specify the **Extension**. Any file reference with this extension will be opened by the particular application or command. Note that for a valid extension a dot is required to put in front of the name of that extension, such as *.doc*.
3. Specify the **Application/Command**. The application or command for opening a file reference with file extension is same as that defined in the **Extension** field.
   A command can be entered directly to the text field and can include application arguments, while an application can be chosen from a file chooser by pressing **...** next to the text field.
4. Specify the **Name** of this application or command. This is an optional field for identifying this file extension.
5. Click **OK** to close the dialog box.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Spell Checking Options

Spell Checking Options

The Spell Checking feature supports spell checking in all inline editing, as well as in Textual Analysis. We support in-place editing of misspelled words, simply by right-clicking your mouse instead of using the complex spell-check box. Spell-check provides intelligent suggestions for words, and you can add your own words into your personal dictionary.

Options

Option Name	Description
Enable spell checking	(default true) Enable spell checking.
Dictionary	The choose of dictionary affects the judgment of correctness of words. • American - (default) Perform spell checking using an American dictionary. • British - Perform spell checking using an British dictionary. • Canadian - Perform spell checking using an Canadian dictionary.
Check spelling as you type	(default true) Check spelling when typing.
Ignore words in UPPERCASE	(default true) Do not classify the use of upper case in a word as a spelling mistake(unless the spelling is wrong).
Ignore words with numbers	(default true) Do not classify the inclusion of number in word as a spelling mistake (unless the spelling is wrong).
Ignore Internet and file address	(default true) Do not classify Internet and file address as a spelling mistake.

Spell Checking Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Keys Options

Keys Options

Customizable program shortcuts

Commands can be invoked by pressing certain keys in the keyboard as shortcuts. For example, holding down the Ctrl modifier key with the 'S' key invokes the save command. Now, key bindings, which is the assignment of keys to commands, can be customized. This permits you to use the familiar keystroke for invoking commands in Visual Paradigm.

To assign/re-assign a key:

1. Double-click on the binding cell of the desired action.
2. Click on the **Binding** field at the bottom of dialog box.
3. Press the key for invoking the command selected. The binding field will be updated accordingly.
4. Press **OK** button to confirm the updates. You will be prompted to restart the application in order to make the changes take effect. By restarting, you can invoke commands using the key defined.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Import/Export Options

XMI Options

Option Name	Description
Enable business process diagram for enterprise architect (experimental feature)	Check this to include BPD when import and export XMI.

XMI options

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Diagramming Options

Appearance

Environment

Model Generation

Shape

Connector

Class

Association

Generalization

ERD & ORM

Interaction

Use Case Diagram

Activity and State

Component Diagram

Deployment Diagram

Business Process

Requirement Diagram

DFD

Communication Diagram

Textual Analysis

Appearance Options

Option Name	Description
Graphics anti-aliasing	(default true) Smoothen the graphics.
Text anti-aliasing	(default true) Smoothen the text.
Description Type	Default type of description • HTML - (default) HTML text that consists of formatting such as bold, italic, underline, table • Plain text - Text without formatting
Model Element Name Alignment	• Top Left - Shape name will appear at top left of shape • Top Middle - Shape name will appear at top middle of shape • Top Right - Shape name will appear at top right of shape • Middle Left - Shape name will appear at middle left of shape • Middle - (default) Shape name will appear at middle middle of shape • Middle Right - Shape name will appear at middle right of shape • Bottom Left - Shape name will appear at bottom left of shape • Bottom Middle - Shape name will appear at bottom middle of shape • Bottom Right - Shape name will appear at bottom right of shape
Diagram background	Background color of diagrams.
Enable minimum size	(default true) Determines whether shapes are restricted to a built-in minimum size.
Fractional Metrics	(default true) When checked, fit size of shape will be performed correctly. When disabled, the shape may look better but size may not fit.
Show Package Name Style	• Within Package Body - The name of package will be shown at the top of package shape • Within Package Tab - The name of package will be shown inside the package tab

Appearance Options details

Related Resources
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Environment Options

Option Name	Description
Default HTML Description Font	The default font face, size, color, bold and italic status for HTML content in description pane.
Stereotype support HTML tagged value	Enables you to define tagged value in HTML format for stereotype.

Environment Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model Generation Options

Option Name	Description
Default generate diagram type from scenario	• Sequence Diagram - (default) Take Sequence Diagram to be the type of diagram that will be generated by Scenario • Interaction Overview Diagram - Take Interaction Overview Diagram to be the type of diagram that will be generated by Scenario
ID Generator Format	Specify the ID format of various element types. If you want an ID to be generated to a type of model element and you find the type does not exist in the table, click **Add** to add the type manually. Then, specify the format of ID by specifying prefix, number of degits and suffix.

Model Generation Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Shape Options

Option Name	Description
Font	Default font settings for shape content, inlcuding font, size, color, bold and italic.
Shape line format	The default line format for shapes.
Shape fill format	The default fill format for shapes.
Reset Formats to Default	Click to reset formatting properties to factory-default. By clicking it, you are asked if you also want to reset the formatting of all the shapes in project. Then, you are asked if you want to apply the setting to workspace. By applying to workspace, newly created project will follow the factory-default formatting setting, too.
Auto fit size (diagram-based)	(default false) Determines whether shapes in diagrams will fit in size automatically.

Shape Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Connector Options

Option Name	Description
Font	Default font settings for connector caption.
Connector Style	• Rectilinear - Set the default connector style to be rectilinear • Round Rectilinear - Set the default connector style to be round rectilinear • Oblique - (default) Set the default connector style to be oblique • Round Oblique - Set the default connector style to be round oblique • Curve - Set the default connector style to be curve
Connection Point Style	• Round the shape - (default) Set the connector end to attach the round the shape • Follow center - Set the connector end to point to the center of attached shapes
Line Jumps	• Off - (default) Disable line jump • Arc - Show connectors' intersections as an arcs • Gap - Show connectors' intersections as a gaps • Square - Show connectors' intersections as a squares
Line jump size	• Normal • Large • Extra large
Caption orientation	• Horizontal only - Enforce connector caption to appear horizontally regardless of connector angle • Horizontal or Vertical only - Enforce connector caption to appear either horizontally or vertically, depending on the connector angle • Follow Connector Angle - Enforce connector caption to appear an the same horizontal level as the connector • Follow Connector Angle and Keep Text Upright - Enforce connector caption to appear an the same horizontal level as the connector but keep the text upright
Foreground	Foreground color of connector.
Background	Background color of connector.
Paint connector through label	(default true) Captions' background will become transparent, so that connectors can show completely without having part of it covered by opaque caption.

Connector Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Class Options

Option Name	Description
Auto-synchronize role name	(default true) Rename role when the owner class is being renamed.
Auto-generate role name	(default false) Auto generate role names for a relationship when the relationship is created between.
Support multiple-line attribute	(default true) Allow to enter attribute name in multiple lines by pressing the new line key defined in **Diagramming > Shape**.
Support multiple-line class name	(default true) Allow to enter class name in multiple lines by pressing the new line key defined in **Diagramming > Shape.**
Update constructor after class renamed	• Auto rename - Automatic update constructor name when the class name is being updated • Do not rename - Do not update constructor name when the class name is being updated • Prompt - (default)
Synchronize documentation of Interface to sub-class	• Always synchronize • Do not synchronize • Prompt
Show row grid line within compartment of Classes in Class Diagram (diagram type-based)	(default false) Show a horizontal line between each attribute or operation in class.
Default parameter direction	• in - When creating a parameter in operation, the direction will be in • out - When creating a parameter in operation, the direction will be out • inout - (default) When creating a parameter in operation, the direction will be inout • return - When creating a parameter in operation, the direction will be return
Default Visibility - Class	• Unspecified - A new class will take Unspecified as visibility • private - A new class will take private as visibility • protected - A new class will take protected as visibility • package - A new class will take package as visibility • public - (default) A new class will take public as visibility • protected internal (.NET only) - A new class will take protected internal as visibility when programming language is set to be .NET • internal (.NET only) - A new class will take internal as visibility when programming language is set to be .NET
Default Visibility - Attribute	• Unspecified - A new attribute will take Unspecified as visibility • private - (default) A new attribute will take private as visibility • protected - A new attribute will take protected as visibility • package - A new attribute will take package as visibility • public - A new attribute will take public as visibility • protected internal (.NET only) - A new attribute will take protected internal as visibility when programming language is set to be .NET • internal (.NET only) - A new attribute will take internal as visibility when programming language is set to be .NET
Default Visibility - Operation	• Unspecified - A new operation will take Unspecified as visibility • private - A new operation will take private as visibility • protected - A new operation will take protected as visibility • package - A new operation will take package as visibility • public - (default) A new operation will take public as visibility • protected internal (.NET only) - A new operation will take protected internal as visibility when programming language is set to be .NET • internal (.NET only) - A new operation will take internal as visibility when programming language is set to be .NET
Default Attribute Value - Multiplicity	Specify the multiplicity of attribute apply when creating a new attribute.

Default Attribute Value - Ordered	Specify the ordered property is checked or not when creating a new attribute.
Default Attribute Value - Unique	Specify the unique property is checked or not when creating a new attribute.

Class Options details

Presentation

Option Name	Description
Show attribute option	• Show all - (default) Show all attributes in Classes • Show public only - Show all public attributes in Classes • Hide all - Hide all attributes in Classes
Show type option	• Fully-qualified - Show attribute type, operation return type and parameter type as full qualified class name • Name only - (default) Show attribute type, operation return type and parameter type as class name • Relative - Show attribute type, operation return type and parameter type as relative class name
Show operation option	• Show all - (default) Show all operations in Classes • Show public only - Show all public operations in Classes • Hide all - Hide all operations in Classes
Visibility style	• Icon - Show icons for representing class members' visibilities • UML - (default) Show icons for representing class members' visibilities such as + for public, minus for private • None - Do not display visibilities
Show attribute initial value	(default true) Show initial value of attribute after its name.
Show attribute multiplicity	(default false) Show multiplicity of attribute after its name.
Show attribute getter/setter	(default false) Show getter and setter symbol for attribute, in front of attribute name.
Show operation signature	(default true) Show operation signature.
Show class member stereotype	(default true) Show the stereotypes set to attributes and operations.
Wrap class member	(default false) Automatic wrap class member against the class's width.
Show owner of class/package	(default false) Show the owner of class or package in class shape.
Show template parameter	(default true) Show template parameter of class.
Display as Robustness Analysis icon	(default true) Display class as robustness analysis icon for classes stereotyped as boundary/ control/entity.
Display as stereotype icon	(default false) Display stereotyped class as stereotype icon.
Show operation parameter name	(default true) Show operation parameter name. When disabled, only parameter type, if defined, would be shown.
Show empty compartments	(default false) Show compartments even when no members are defined.

Presentation of Class Options details

Auto Attribute Type

Option Name	Description
Default attribute type	Define attribute type that will be applied to newly created attributes.
Auto set attribute type by name	(default true) Automatically set attribute type and default value when the name user entered for an attribute matches with one of those listed in the table followed.

Auto Attribute Type of Class Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Association Options

Option Name	Description
Show association stereotype	(default true) Show the stereotypes assigned to an association.
Show from role name	(default true) Show the role name of the from end of association.
Show to role name	(default true) Show the role name of the to end of association.
Show from role visibility	(default true) Show the role visibility of the from end of association.
Show to role visibility	(default true) Show the role visibility of the to end of association.
Show from multiplicity	(default true) Show the role multiplicity me of the from end of association.
Show to multiplicity	(default true) Show the role multiplicity of the to end of association.
Show multiplicity constraints	(default false) Show multiplicity constraint such as {unique} for roles.
Show direction	(default false) Show a triangle mark on association for indicating direction.
Show association role stereotypes	(default true) Show stereotypes assigned to role.
Show owned association end as attribute	(default true) When you set the ownership at the end of an association (by right clicking on the association end and selecting "Owned by"), with this option on, an attribute will be created in the class on the opposite side.
Default Association End Navigable	• Unspecified - A new association will set Navigable as Unspecified for both ends • True - (default) A new association will set Navigable as True for both ends • False - A new association will set Navigable as False for both ends
Default Association End Visibility	• Unspecified - (default) A new association will set Visibility as Unspecified for both ends • private - A new association will set Visibility as private for both ends • protected - A new association will set Visibility as protected for both ends • package - A new association will set Visibility as package for both ends • public - A new association will set Visibility as public for both ends • protected internal (.NET only) - A new association will set Visibility as protected internal for both ends when programming language is set to be .NET • internal (.NET only) - A new association will set Visibility as internal for both ends when programming language is set to be .NET
Suppress implied "1" multiplicity for attribute and association end	(default false) Suppress implied "1" multiplicity for attribute and association end.
Synchronize association name with association class	• Prompt - Prompt if you want to make the name of association follow the association class when creating the association class • Yes - Make the name of association follow the association class when creating the association class • No - Do not make the name of association follow the association class when creating the association class
Synchronize association role name with referenced attribute name	(default true) You can create a "Referenced Attribute" from an association end in the Association End Specification. With this option on, when you rename the referenced attribute, the name of association end (i.e. the role) will be updated accordingly.

Association Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Generalization Options

Option Name	Description
Generalization set notation	• One Shape per Generalization - One generalization set shape per each Generalization relationship • Common Generalization Arrowhead - (default) Combine Generalization relationships' arrow head for the same set • Dashed Line

Generalization Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ERD & ORM Options

Option Name	Description
Show column type	(default true) Show data type of column.
Foreign key connector end points to associated column	(default false) Attach foreign key connector end points to the column associated.
Align column properties	(default true) Align the column properties, so that columns in entity will appear tidier.
Show extra column properties	(default true) Show extra column properties such as Nullable.
Show row grid line within compartment of Entities/Views in ERD (diagram type-based)	(default true) Show grid lines between row within Entities and Database Views in ERD.
Show row grid line within compartment of Entities/Views/Classes in ORM Diagram (diagram type-based)	(default true) Show grid lines between row within Entities, Database Views and Classes in ORM.
Warning on create ORM-Persistable Class in default package	(default true) Warn when creating ORM Persistable class at root.
Show schema name in ERD: ${name}.${tableName}	(default true) Show schema name, if defined, for entities.
Show table record editor	
Column Constraints Presentation	• Symbol • Text • Icon
Foreign key arrow head size	• Very Small • Small • Medium (default) • Large • Extra Large • Jumbo • Colossal
Primary Key Pattern	Pattern of primary keys that will be applied when synchronizing Class Diagram to Entity Relationship Diagram, which may create primary key.
Primary Key Constraint Patterm	Pattern of primary key constraint.
Foreign Key Pattern	Pattern of foreign key.
Foreign Key Relationship Pattern	Pattern of foreign key relationship pattern.
Index Pattern	Pattern of index.

ERD & ORM Options details

Auto Column

Option Name	Description
Default column type	Define column type that will be applied to newly created columns.
Default column nullable	Specify if a newly created column is nullable.
Auto set column type by name	(default true) Automatically set column type and default value when the name user entered for a column matches with one of those listed in the table followed.

Auto Column Type of ERD & ORM Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems

- Contact us if you need any help or have any suggestion

Interaction Options

Option Name	Description
Mark target lifeline stopped when attached by destroy message	The effect when attaching a Destroy Message to a Lifeline, to know whether the Lifeline will be marked stopped or not: • Yes - Lifeline will mark as stopped • No - Lifeline will not mark as stopped • Prompt - (default) Ask if you want to mark the Lifeline as stopped
Show sequence number in Communication Diagram	(default true) Show numbering on Sequence Message in Communication Diagram.
Show sequence number in Sequence Diagram	(default true) Show numbering on Sequence Message in Sequence Diagram.
Show messages operation signature in Sequence Diagram and Communication Diagram (diagram-based)	(default false) Show Sequence Messages' operation signatures in Sequence Diagram and Communication Diagram.
Show stereotype of message in Sequence Diagram and Communication Diagram (diagram-based)	(default true) Show Sequence Messages' stereotypes in Sequence Diagram and Communication Diagram.
Show activations in Sequence Diagram (diagram-based)	(default true) Show Activations in Sequence Diagram. If unchecked, sequence message will be attached to Lifeline instead of Activations.
Display as Robustness Analysis icon in Sequence Diagram	(default true) Display Lifeline as robustness analysis icon for Lifelines stereotyped as boundary/ control/entity.
Show associated diagram name of Interaction	(default false)

Interaction Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Use Case Diagram Options

Option Name	Description
Show Use Case Extension Points	(default true) Show Use Case Extension Points within Use Case shapes.
Show Use Case ID	(default false) Show the ID of use cases.
Rename Extension Point to Follow Extend Use Case	• Yes - Rename Extension Points to follow extend use case automatically when the name of extend use case is changed. • No - Even when the name of extend use case is changed, the name of Extension Points will not change to follow extend use case. • Prompt (default) - Ask if rename Extension Points to follow extend use case when the name of extend use case is changed.

Use Case Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Activity and State Options

Activity Diagram

Option Name	Description
Show Caption	• Initial Node - (default false) Show caption for Initial Node • Expansion Node - (default false) Show caption for Expansion Node • Activity Final Node - (default false) Show caption for Activity Final Node • Flow Final Node - (default false) Show caption for Flow Final Node • Decision Node - (default false) Show caption for Decision Node • Merge Node - (default false) Show caption for Merge NodeArtifactat • Fork Node - (default false) Show caption for Fork Node • Join Node - (default false) Show caption for Join Node
Show Partition Header	• Auto - (default) Show horizontal and/or vertical Partition headers if there is Partitions in that orientation • Always Show - Always show Partition headers regardless of the orientation of Partitions, even if there is no Partition • Always Hide - Always hide Partition headers
Decision/Merge Node connection point style	Determines how connector connects to decision/merge node. • Default (default) • Connect to vertex
Show Object Node Type	(default true) Show the type of object node inside the object node shape.

Activity Diagram Options details

State Machine Diagram

Option Name	Description
Show Caption (State Machine Diagram)	• Shallow History - (default false) Show caption for Shallow History • Deep History - (default false) Show caption for Deep History • Initial Pseudo State - (default false) Show caption for Initial Pseudo State • Junction - (default false) Show caption for Junction • Final State - (default false) Show caption for Final State • Terminate - (default false) Show caption for Terminate • Fork - (default false) Show caption for Fork • Join - (default false) Show caption for Join
Auto create Initial State on State Diagram	(default true) Automatic create an initial state when creating a State Machine Diagram.
Default location (in pixel)	Position of Initial State create by default.
Use state name tab	(default false) Name tab is a a tiny rectangle that appear on top of a state and at the left hand side, displaying the name of a state. Use state name tab is to enable such tab.
Show transition trigger	Triggers can be added to a Transition relationship. This option determines the visibility of Triggers. • Show - (default) Show Triggers information on a Transition connector • Hide - Do not show Triggers information on a Transition connector
Show precondition, postcondition and body of internal activities in State	(default true) Show the precondition, postcondition and body of internal activities in state.

State Machine Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Component Diagram Options

Option Name	Description
Show Component Option	• Keyword - Show only the keyword <<component>> at the top of Component • Icon - Show only an icon representing a Component at the top right of Component • Keyword and Icon - (default) Show both keyword and icon for a Component • None - Do not show keyword and icon for a Component

Component Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Deployment Diagram Options

Option Name	Description
Show Artifact Option	• Keyword - Show only the keyword <<artifact>> at the top of Artifact • Icon - Show only an icon representing an Artifact at the top right of Artifact • Keyword and Icon - (default) Show both keyword and icon for a Artifact • None - Do not show keyword and icon for a Artifact

Component Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business Process Options

Option Name	Description
Connect Gateway with Flow Object in Different Pool	• Prompt - (default) Prompt if user want to change the Message to Message Flow, Sequence Flow or cancel the action • Connect with Message Flow - Change or keep the relationship as Message Flow • Connect with Sequence Flow - Change or keep the relationship as Sequence Flow
Auto stretch pools	(default true) Stretch Pools automatically to the reach diagram bound.
Default Type of Sub-Process	The default type of newly created sub-process. • Unspecified (default) • Embedded • Reusable • Reference • Event
Business Process Diagram default language	The default value of language property of BPD
ID Generator Format - Show ID option	• Not Show - (default true) Do not display ID • Show Below Caption - Display ID as part of the caption, under the name • Show as Label - Display ID as a label attaching to shape • Show as Customized - Set the way to show ID per element type

Business Process Options details

Presentation

Option Name	Description
Connection Point Style	• Round the shape - Set the connection point style to Round the shape • Follow center - (default) Set the connection point style to Follow center
Connector Style	• Rectilinear - Set the connector style to Rectilinear • Round Rectilinear - (default) Set the connector style to Round Rectilinear • Oblique - Set the connector style to Oblique • Round Oblique - Set the connector style to Round Oblique • Curve - Set the connector style to Curve
Show Activities type icon (diagram-based)	(default true) Show icons that represent the type of Task and Sub-Process.
Data-based gateway markers visible	(default false) Show a cross in BPMN gateway shape when the gateway has a "Data-Based Exclusive Decision/Merge (XOR)" type.
Draw text annotation open rectangle folow connector end	(default true) Make the open rectangle of text annotation to follow the position of end of the attached connector.

Business Process Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Block Definition Diagram

Option Name	Description
Wrap compartment text	Wrap the text in compartments like part, reference, value and flow property in SysML block

Block Definition Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Requirement Diagram Options

Option Name	Description
Show Attributes	• Show All Attributes - (default) Show all Requirement attributes • Show Non-empty Attributes - Show only Requirement attributes that have values defined • Hide All Attributes - Hide all Requirement attributes
Wrap member	(default false) Wrap the Requirement members' content.
Support HTML Attribute	(default false) Allow to fill in attributes with rich text format.

Requirement Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

DFD Options

Option Name	Description
Add Data Stores and External Entities to Decomposed DFD	• Yes - When decompose a DFD, Data Stores and External Entities on the current diagram will be copied to the decompose diagram • No - When decompose a DFD, Data Stores and External Entities on the current diagram will not be copied to the decompose diagram • Prompt - (default) When decompose a DFD, prompt if user want the Data Stores and External Entities on the current diagram to be copied to the decompose diagram

DFD Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Communication Diagram Options

Option Name	Description
Use one message for one direction	On a link between lifeline, you can choose whether to group message arrows that follow the same direction into a single arrow or show as separate message arrows.

Communication Diagram Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Textual Analysis Options

Option Name	Description
Highlight Option	• Case insensitive - (default) Words which are the same as the entered word, even in different cases, are highlighted. • Case sensitive - Words which are the same as the entered word or/and with same case are highlighted.
Generate Requirement Text from Candidate Option	• Extracted text - (default) When create requirement from a candidate requirement, its text property will be filled by the candidate's extracted text. • Class description - When create requirement from a candidate requirement, its text property will be filled by the candidate's class description.

Textual Analysis options details

Related Resources

The following resources may help you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Instant Reverse Options

Instant Reverse Options

Option Name	Description
Create shape for parent model of dragged class/ package	(default false) When drag and drop an element from tree to diagram, add also their parent (e.g. Package) to diagram to contain the dropped shapes.
Calculate Generalization and Realization	(default false)
Reverse Operation's Implementation	(default false)

Instant Reverse Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ORM Options

General Options

Option Name	Description
Quote SQL Identifier	• Yes - Add quotes to SQL identifier to prevent potential violation when executing SQL • No - Do not add quotes to SQL identifier • Auto - (default) Quote only reserved words
Synchronize Name	• Yes - Auto update model element name when synchronize class diagram and ERD • No - Do not update model element name when synchronize class diagram and ERD • Prompt - (default) Prompt to update model element name when synchronize class diagram and ERD
Mapping File Column Order	• ERD - (default) Generate columns in mapping file in same order as ERD • Index - Generate index columns first in mapping file
Wrapping Servlet Request	• On - Automatically lock persistable object when get by HttpSession.getAttribute() • Off - (default) Do not lock object automatically
Getter/Setter Visibility	• Public - (default) Generate public getter/setter • Follow attribute -Getter/setter visibility follow attribute's visibility
Decimal Precision and Scale - Use default	(default true) Automatically determine the most suitable precision and scale when synchronize from attribute to column as decimal.
Decimal Precision and Scale - Precision	Specify the precision when synchronize from attribute to column as decimal.
Decimal Precision and Scale - Scale	Specify the scale when synchronize from attribute to column as decimal.
ID Generator	• assigned - allow the application to assign an identifier to the object before `save()` is called. • guid - uses a database-generated GUID string on MS SQL Server and MySQL. • hilo - uses a hi/lo algorithm to efficiently generate identifiers of type `long`, `short` or `int`, given a table and column as a source of hi values. The hi/lo algorithm generates identifiers that are unique only for a particular database. • identity - supports identity columns in DB2, MySQL, MS SQL Server, Sybase and HypersonicSQL. The returned identifier is of type `long`, `short` or `int`. • increment - generates identifiers of type `long`, `short` or `int` that are unique only when no other process is inserting data into the same table. *Do not use in a cluster.* • native - (default) picks `identity`, `sequence` or `hilo` depending upon the capabilities of the underlying database. • seqhilo - uses a hi/lo algorithm to efficiently generate identifiers of type `long`, `short` or `int`, given a named database sequence. • sequence - uses a sequence in DB2, PostgreSQL, Oracle. The returned identifier is of type `long`, `short` or `int`
Generate diagram from ORM wizards	(default true) Generate diagram when finish ORM wizards.
Export comment to database	(default true) Generate documentation to table/column.
ERD numeric to class type	• Automatic - (default) Automatically select attribute type when synchronize from column numeric type • Integer - Synchronize column numeric type to attribute as integer type • Float - Synchronize column numeric type to attribute as float type • Double - Synchronize column numeric type to attribute as double type • Big Decimal - Synchronize column numeric type to attribute as big decimal type
SQL Statement Case	• Upper Case - (default) Generate upper case keyword in SQL • Lower case - Generate lower case keyword in SQL
Formatted SQL	(default false) Generate pretty formatted SQL.

General Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Synchronization Options

Option Name	Description
Entity => Class Name - Prefix	Append characters/words in front of name
Entity => Class Name - Class name	• Capitalize - (default) The first character of each word becomes uppercase • Decapitalize - The first character of each word becomes lowercase • Upper case - All characters become uppercase • Lower case - All characters become lowercase • Upper camel case - words are joined without underscore ("_") and are capitalized • Lower camel case - Same as upper camel case except that the first character is lower case • Reverse camel case - each upper case character are considered as word separator, words are joined with underscore ("_") and are lower case • Reverse camel to upper case - each upper case character are considered as word separator, words are joined with underscore ("_") and are upper case • Don't change - Keep name unchanged
Entity => Class Name - Suffix	Append characters/words after name.
Column => Attribute Name - Prefix	Append characters/words in front of name.
Column => Attribute Name - Attribute Name	• Capitalize - The first character of each word becomes uppercase • Decapitalize - (default) The first character of each word become lowercase • Upper case - All characters become uppercase • Lower case - All characters become lowercase • Upper camel case - words are joined without underscore ("_") and are capitalized • Lower camel case - Same as upper camel case except that the first character is lower case • Reverse camel case - each upper case character are considered as word separator, words are joined with underscore ("_") and are lower case • Reverse camel to upper case - each upper case character are considered as word separator, words are joined with underscore ("_") and are upper case • Don't change - Keep name unchanged
Column => Attribute Name - Suffix	Append characters/words after name.
Class => Entity Name - Prefix	Append characters/words in front of name.
Class => Entity Name - Table name	• Capitalize - The first character of each word becomes uppercase • Decapitalize - The first character of each word becomes lowercase • Upper case - All characters become uppercase • Lower case - All characters become lowercase • Upper camel case - words are joined without underscore ("_") and are capitalized • Lower camel case - Same as upper camel case except that the first character is lower case • Reverse camel case - each upper case character are considered as word separator, words are joined with underscore ("_") and are lower case • Reverse camel to upper case - each upper case character are considered as word separator, words are joined with underscore ("_") and are upper case • Don't change - (default) Keep name unchanged
Class => Entity Name - Suffix	Append characters/words after name.
Attribute => Column Name - Prefix	Append characters/words in front of name.
Attribute => Column Name - Column name	• Capitalize - (default) The first character of each word becomes uppercase • Decapitalize - The first character of each word becomes lowercase • Upper case - All characters become uppercase • Lower case - All characters become lowercase • Upper camel case - words are joined without underscore ("_") and are capitalized • Lower camel case - Same as upper camel case except that the first character is lower case • Reverse camel case - each upper case character are considered as word separator, words are joined with underscore ("_") and are lower case

	• Reverse camel to upper case - each upper case character are considered as word separator, words are joined with underscore ("_") and are upper case • Don't change - Keep name unchanged
Attribute => Column Name - Suffix	Append characters/words after name.
Synchronize Name	• Yes • No • Prompt
Table per subclass FK Mapping	• Top Base Class • Super Class

Synchronization Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

State Code Engine Options

State Code Engine Options

Option Name	Description
Language	you can select the following language: • Java - (default) Generate code in Java • C# - Generate code in C# • VB .NET - Generate code in VB.NET • C++ - Generate code in C++
Synchronized transition methods	(default true) Generate synchronized keyword for transition methods.
Generate try catch block	(default true) Generate try catch block for method calls that may produce exception.
Generate debug messages	(default false) Generate debug message to help tracing problems that happen when running generated code.
Auto create transition methods	(default true) Auto generate operation to owner class by transition.
Re-generate transition methods	(default false) Overwrite the transition methods if already exists in source code.
Generate sample code	(default true) Generate sample code to help you to understand how to work with generated code.

State Code Engine Options details

Related Resources

The following resources may help to you learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Data Type Options

Data type options

UML is theoretically a modeling language independent to particular programming language(s). Yet, it is possible to transform between UML models to a software applications or systems. While the pre-defined data-type set works well in the UML world, there is enormous need to ensure the design can be applied to programming source code. Problems comes from the fact that programming languages, by nature, are unlikely to share the same set of data-types suggested by UML. A typical example is about the use of boolean. ‘boolean', ‘bool' and ‘Boolean' are adopted by UML and Java, C# and VB.NET respectively. But they are all referring to the same thing – boolean.

Visual Paradigm lets you choose a programming language that your UML project should be based on. When modeling, you can easily select a data-type that is allowed for the chosen language, without typing it. Besides, new languages and data types can be added, which increase the flexibility of working under different domains.

Configure programming language

1. In **Diagram Navigator / Model Explorer/ Class Repository**, right click on the project root node and select **Configure Programming Language...** from the pop-up menu.
2. In the **Programming Language** dialog box, select the language to switch to. The way how data-type will be mapped from the current language to the chosen language is listed in the table following the data-type definition of that language.

Customizing programming language and data types

By default, there are six types of predefined (programming) languages. Each of them consists of a set of supported data types. Besides working with those default languages and types, you can add your own languages and add data types. To do so:

1. Open the **Project Options** window by selecting **Tools > Project Options** from the main menu.
2. Choose **Data Type** from the list on the left hand side of the **Project Options** window.
3. In the **Data Type** page, click the **plus** button under **Languages:** to add a language.
4. Enter its name, and press **OK** button to confirm.
5. Press **Add...** button to add a data-type to the chosen language.
6. Enter its name and press **OK** to confirm. From now on, once you have set your own language as the language for your project, you can pickup the associated data-types as attribute type, operation return type and parameter type.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Code Synchronization Options

Code Options

Option Name	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified.
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified.
Default parameter type	(default int) Type that will be assigned to Parameter upon code generation when type is unspecified.
Generate Pre and Post Condition	
Reverse interface getter/ setter as association	
Auto realize interface	(default false) Generate operations defined in interface in sub-classes.
Remove method body after changed to abstract method	(default true) When an operation is set from non-abstract to abstract, updating code will remove the related method's body.
Use "is" prefix for getters that return boolean	(default true) Generate getter's name as isXXXX() for getters that return a boolean value.
Add import statement instead of using fully qualified type name	(default true) Add import statement for referencing classes in another package/namespace instead of using fully qualified name inline.
Import fully qualified type name for referenced type	(default false) Use fully qualified type name in import statements instead of using wildcard character * to represent importing all classes in package.
Java Collection	• Array - Generate one-to-many relationship as array • Collection - (default) Generate one-to-many relationship as collection
Use generic collections	(default true) Allow to use generic collection.
Generate annotation on	• Property method - Generate annotation on property method • Field - Generate annotation on field
Generate annotation in code convention	(default true) Generate annotation in code convention.
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files • Other -Specify an encoding for source files

Code Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Brace and Identation Options

Option Name	Description
Class declaration	• Same line - (default) Brace for class declaration appears at the same line as the declaration • Next line - Brace for class declaration appears at the line after the declaration
Constructor declaration	• Same line - (default) Brace for constructor appears at the same line as the declaration • Next line - Brace for constructor appears at the line after the declaration
Method declaration	• Same line - (default) Brace for method appears at the same line as the declaration • Next line - Brace for method appears at the line after the declaration
Enum declaration	• Same line - (default) Brace for enumeration appears at the same line as the declaration • Next line - Brace for enumeration to appear at the line after the declaration
Annotation type declaration	• Same line - (default) Brace for annotation type appears at the same line as the declaration • Next line - Brace for annotation type appears at the line after the declaration
Indentation policy	• Tabs - (default) Use a tab of space as indentation • Spaces - Use spaces as indentation. The number of spaces can be defined below
Indentation size	The number of spaces to indent.

Brace and Indentation Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

New Lines Options

Option Name	Description
Before package declaration	Number of blank lines to appear before Package declaration.
After package declaration	Number of blank lines to appear after Package declaration.
Before import declaration	Number of blank lines to appear before import statements.
After import declaration	Number of blank lines to appear after import statements.
Before first declaration	Number of blank lines to appear before the first declaration within Class declarations.
Before different kind declaration	Number of blank lines to appear before a different kind of declaration.
Before field declaration	Number of blank lines to appear before field declaration.
Before method declaration	Number of blank lines to appear before method declaration.
Before inner type declaration	Number of blank lines to appear before inner type declaration.
Number of lines to empty body	Number of blank lines to appear in empty method body.

New Lines Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Template Options

Option Name	Description
Operation Template	The content to fill in for operation body of code file synchronized from class.
Getter Template	The content to fill in for getter body of code file synchronized from class.
Setter Template	The content to fill in for setter body of code file synchronized from class.

Templates Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

C++ Code Synchronization Options

Code Options

Option Name	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified.
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified.
Default parameter type	(default int) Type that will be assigned to Parameter upon code generation when type is unspecified.
Text File Encoding	• System default - (default) The default system encoding will be selected as encoding for source files • Other - Specify an encoding for source files

Code Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Brace and Identation Options

Option Name	Description
Class declaration	• Same line - (default) Brace for class declaration appears at the same line as the declaration • Next line - Brace for class declaration appears at the line after the declaration
Constructor declaration	• Same line - (default) Brace for constructor appears at the same line as the declaration • Next line - Brace for constructor appears at the line after the declaration
Method declaration	• Same line - (default) Brace for method appears at the same line as the declaration • Next line - Brace for method appears at the line after the declaration
Enum declaration	• Same line - (default) Brace for enumeration appears at the same line as the declaration • Next line - Brace for enumeration tor appears at the line after the declaration
Annotation type declaration	• Same line - (default) Brace for annotation type appears at the same line as the declaration • Next line - Brace for annotation type appears at the line after the declaration
Indentation policy	• Tabs - (default) Use a tab of space as indentation • Spaces - Use spaces as indentation. The number of spaces can be defined below
Indentation size	The number of spaces to indent.

Brace and Indentation Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

New Lines Options

Option Name	Description
Before first declaration	Number of blank lines to appear before the first declaration within Class declarations.
Before different kind declaration	Number of blank lines to appear before a different kind of declaration.
Before field declaration	Number of blank lines to appear before field declaration.
Before method declaration	Number of blank lines to appear before method declaration.

New Lines Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Template Options

Option Name	Description
Operation Template	The content to fill in for operation body of code file synchronized from class.
Getter Template	The content to fill in for getter body of code file synchronized from class.
Setter Template	The content to fill in for setter body of code file synchronized from class.

Templates Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Model Quality Options

Model Quality Options

Option Name	Description
Enable model quality checking	Model quality checking is the ability to check project data for potential flaws. The chekcing is done automatically. Whenever a problem is detected, shapes will be underlined directly in diagram. You may turn this function off by unchecking this option.

Model Quality Options details

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Product update

Automatic update refers to the ability to detect and download possible updates from the Internet. This chapters shows you how to work with the automatic update feature.

Updating Visual Paradigm to latest build/version

How to perform manual update.

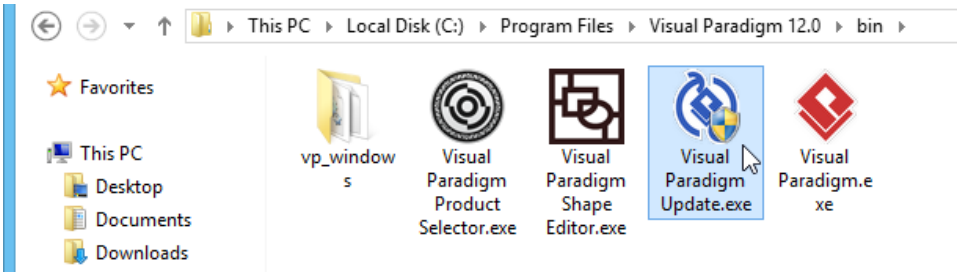
Automatic update notification

How automatic update works and how to react to update prompt.

Updating Visual Paradigm to Latest Build/Version

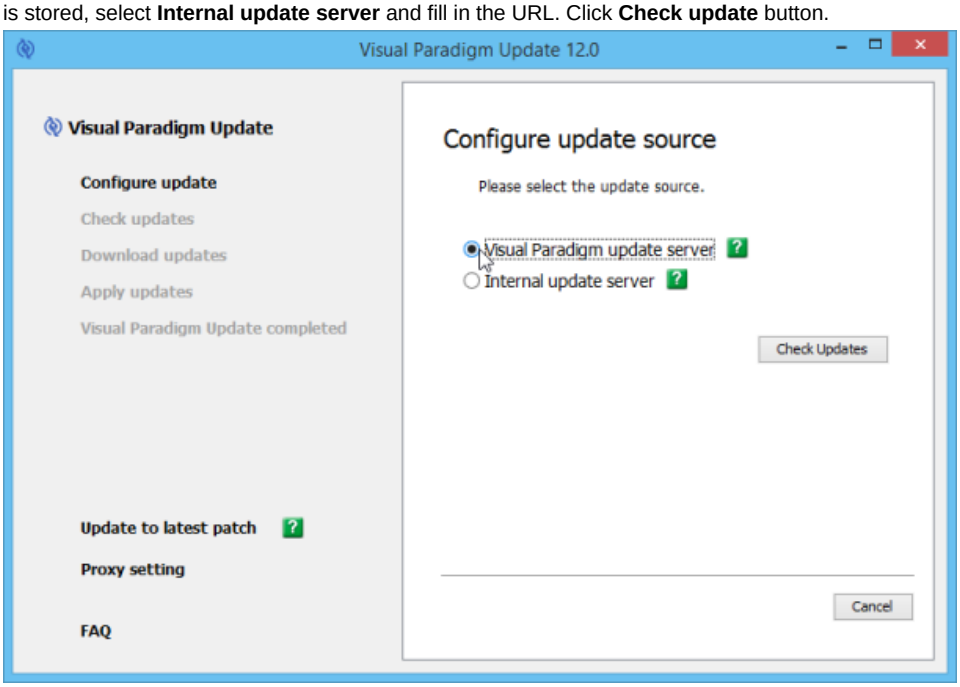
Visual Paradigm releases hotfixes in a monthly basis. It is recommended to run the update program monthly to make sure the installation is up-to-date. In this article you will learn how to update your installation to newer build. You can take the same steps to update your installation to a new version (if any).

1. Run the **Visual Paradigm Update** program. You can run it under **%Visual-Paradigm_Install_DIR%\bin**.



Run Visual Paradigm update

2. This shows the update program. Select the place where the update program can look for the update files. If you want to update Visual Paradigm to the latest build or version, select **Visual Paradigm update server** and click **Check update**. If you have a specific place where the update file is stored, select **Internal update server** and fill in the URL. Click **Check update** button.



Select Visual Paradigm update server

> **NOTE:** If you need to configure a proxy server for connection, click **Proxy setting** at bottom left.

> **NOTE:** A patch is special build made that contains specific bug fix/enhancement, made for specific users.
> When and only when you are asked by Visual Paradigm to update to the latest patch build, click **Update to latest patch**.

3. Confirm the build to update to and then click **Perform Update**. If there is a new version and if your software maintenance agreement is active, you will see an option for upgrading to the new version. If you choose to upgrade to the new version, the update program will help you to grab the new license upon the finishing of the upgrade.
4. Allow the program download and update the files for you. When a file is found modified both in the latest build and the installation, you need to select whether to keep the local copy, by clicking **Ignore update** or to apply the latest version by clicking **Overwrite**. When a file is found removed in the latest build but exist in the installation, it will be listed in red. Since the file is obsolete and has already been removed in the latest version, neither **Overwrite** button nor **Ignore update** button will be shown on screen. It will be removed without choice.
5. Click **Complete** when finish.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Automatic Update Notification

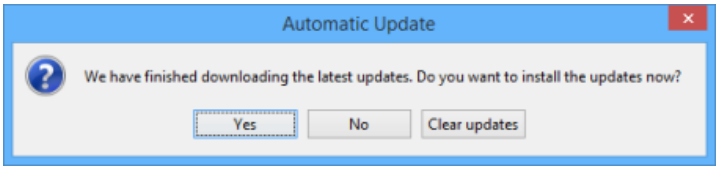
When you run Visual Paradigm, checking of possible updates is done in background. If there are available updates, you will be notified through the message pane. Then, you can perform an update to advance to the latest build.

Updating when running Visual Paradigm

1. When running Visual Paradigm, a message " **A new release of Visual Paradigm is available**" may popup in the **Message** pane. This message indicates that there is a build newer than the one that you are running and you are recommended to perform an update to advance to the latest build.

Notification of available updates

2. You can click on the message to start updating. If not, when you restart Visual Paradigm. You'll see the following dialog box:



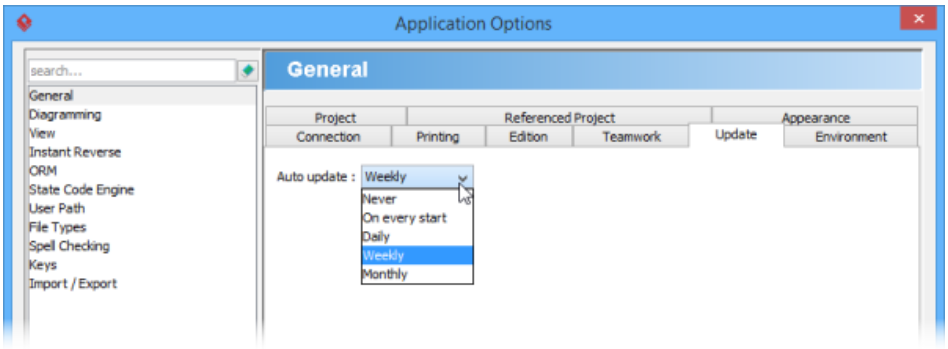
Prompting for updating when starting Visual Paradigm

3. By clicking **Yes**, the **Visual Paradigm Update** program will be launched. Update will be performed.
4. Click **Complete** when finish.

Setting the interval of checking updates

By default, update is checked weekly when starting Visual Paradigm. You can change the interval of checking updates through the **Application Options** window. To change:

1. Open the **Application Options** window by selecting **Tools > Application Options...** from the main menu.
2. In the **Application Options** window, select **General** from the list at the left hand side, then open the **Update** tab.
3. Select the interval of performing auto update from the **Auto update** drop down menu.



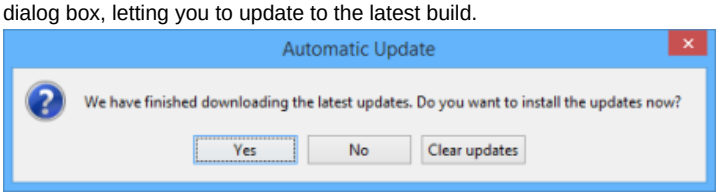
Selecting update interval

Here are the available options:

Option	Description
Never	Do not check for product updates anymore
On every start	Check for product updates everytime when starting Visual Paradigm
Daily	Check for product updates everyday, when starting Visual Paradigm
Weekly	Check for product updates every week, when starting Visual Paradigm
Monthly	Check for product updates every month, when starting Visual Paradigm

Available update interval

Click **OK** to confirm updating. From now on, once the interval elapsed and if there are available updates, you will see the **Automatic Update** dialog box, letting you to update to the latest build.



Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Connection rules

Model elements can be linked together using connectors. There are connection rules to control the type of model elements a connector support.

Use case diagram connection rules

Connection rules for shapes in use case diagram

Class diagram connection rules

Connection rules for shapes in class diagram

Sequence diagram connection rules

Connection rules for shapes in sequence diagram

Communication diagram connection rules

Connection rules for shapes in communication diagram

State machine diagram connection rules

Connection rules for shapes in state machine diagram

Activity diagram connection rules

Connection rules for shapes in activity diagram

Component diagram connection rules

Connection rules for shapes in component diagram

Deployment diagram connection rules

Connection rules for shapes in deployment diagram

Package diagram connection rules

Connection rules for shapes in package diagram

Object diagram connection rules

Connection rules for shapes in object diagram

Composite structure diagram connection rules

Connection rules for shapes in composite structure diagram

Interaction overview diagram connection rules

Connection rules for shapes in interaction overview diagram

Requirement diagram connection rules

Connection rules for shapes in requirement diagram

Basic diagram connection rules

Connection rules for shapes in basic diagram

Entity relationship diagram connection rules

Connection rules for shapes in entity relationship diagram

ORM diagram connection rules

Connection rules for shapes in ORM diagram

Business process diagram connection rules

Connection rules for shapes in business process diagram

Conversation diagram connection rules

Connection rules for shapes in conversation diagram

Data flow diagram connection rules

Connection rules for shapes in data flow diagram

EPC diagram connection rules

Connection rules for shapes in EPC diagram

Process map diagram connection rules

Connection rules for shapes in process map diagram

Organization chart diagram connection rules

Connection rules for shapes in organization chart diagram

Archimate diagram connection rules

Connection rules for shapes in archimate diagram

EJB diagram connection rules

Connection rules for shapes in EJB diagram

Overviewview diagram connection rules

Connection rules for shapes in overview diagram

Mind mapping diagram connection rules

Connection rules for shapes in mind mapping diagram

Use case diagram connection rules

	Actor	System	Collaboration	Use case
Actor	Association, Dependency, Realization, Generalization	Dependency, Realization	Dependency, Realization	Association, Dependency, Realization
System	Dependency, Realization	Dependency, Realization	Dependency, Realization	Dependency, Realization
Collaboration	Dependency, Realization	Dependency,	Dependency, Realization, Generalization	Association, Dependency, Realization
Use case	Dependency, Realization	Dependency, Realization	Association, Dependency, Realization	Association, Dependency, Include, Extend, Realization, Generalization

Connection rules in use case diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Class Diagram Connection Rules

	Class	NARY	Collaboration	Model
Class	Generalization, Realization, Usage, Association, Aggregation, Composition, Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace	Realization, Association, Aggregation, Composition, Dependency, Access, Import, Merge, Instantiation, Substitution	Realization, Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace	Realization Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace
NARY	Realization, Association, Aggregation, Composition, Dependency, Access, Import, Merge, Instantiation, Substitution	Realization, Dependency, Access, Import, Merge, Instantiation, Substitution	Realization, Dependency, Access, Import, Merge, Instantiation, Substitution,	Realization, Dependency, Access, Import, Merge, Instantiation, Substitution,
Collaboration	Realization, Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace	Realization, Dependency, Access, Import, Merge, Instantiation, Substitution	Generalization, Realization, Usage, Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution	Realization, Dependency, Binding Dependency, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace
Model	Realization, Dependency, Binding Dependency, Permission, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace	Realization, Dependency, Access, Import, Merge, Instantiation, Substitution	Realization, Dependency, Binding Dependency, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace	Generalization, Realization, Usage, Dependency, Binding Dependency, Access, Import, Merge, Instantiation, Substitution, Abstraction, Derive, Refine, Trace

Connection rules in class diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Sequence diagram connection rules

	Lifeline	Combined fragment	Interaction use	Frame	Actor	Concurrent	Continuation	Gate
Lifeline	Message				Message	Message		Message
Combined fragment								
Interaction use								
Frame								
Actor								
Concurrent	Message				Message			Message
Continuation								
Gate	Message				Message			

Connection rules in sequence diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Communication diagram connection rules

	Lifeline	Actor
Lifeline	Line, Dependency	Link, Dependency
Actor	Link, Dependency	Link, Generalization, Dependency

Connection rules in communication diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

State machine diagram connection rules

	State	Submachine state	Initial pspeudo state	Shallow history	Deep history	Choice	Junction	Fork	Join	Entry point	Exit point	Terminate	Final state
State	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Submachine state	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Initial pseudo state	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Shallow history		Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Deep history		Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Choice		Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Junction		Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Fork		Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Join	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Entry point	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition		Transition	Transition	Transition
Exit point	Transition	Transition		Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition	Transition
Terminate													
Final state													

Connection rules in state machine diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Activity diagram connection rules

	Activity	Activity parameter node	Action	Accept event action	Send signal action	Decision node	Merge node	Fork node	Join node	Initial node	Activity final node	Flow final node	Input pin	Output pin
Activity	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Activity parameter node	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	
Action	Control Flow	Object Flow	Control Flow, Exception handler	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow, Exception handler	Exception handler
Accept event action	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow, Exception handler	Exception handler
Send signal action	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow, Exception handler	Exception handler
Decision node	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Merge node	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Fork node	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Join node	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Initial node	Control Flow	Object Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Object Flow	Control Flow	Control Flow	Object Flow	
Activity final node														
Flow final node														
Input pin														
Output pin		Object Flow				Object Flow	Object Flow	Object Flow	Object Flow		Object Flow	Object Flow		
Value pin														
Object node	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow		Object Flow	Object Flow	Object Flow	
Central buffer node	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow		Object Flow	Object Flow	Object Flow	
Data store node	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow		Object Flow	Object Flow	Object Flow	
Interruptible activity region														
Expansion region													Exception handler	Exception handler
Expansion node	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow		Object Flow	Object Flow	Object Flow	

Swimlane														
Structured Activity node	Control Flow		Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow		Control Flow	Control Flow	Exception handler	Exception handler
Conditional node	Control Flow		Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow		Control Flow	Control Flow	Exception handler	Exception handler
Loop node	Control Flow		Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow		Control Flow	Control Flow	Exception handler	Exception handler
Sequence node	Control Flow		Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow	Control Flow		Control Flow	Control Flow	Exception handler	Exception handler

Connection rules in activity diagram A

	Value pin	**Object node**	**Central buffer node**	**Data store node**	**Interruptible activity region**	**Expansion region**	**Expansion node**	**Swimlane**	**Structured Activity node**	**Conditional node**	**Loop node**	**Sequence node**
Activity		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Activity parameter node		Object Flow	Object Flow	Object Flow			Object Flow		Object Flow	Object Flow	Object Flow	Object Flow
Action		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Accept event action		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Send signal action		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Decision node		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Merge node		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Fork node		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Join node		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Initial node		Object Flow	Object Flow	Object Flow			Object Flow		Control Flow	Control Flow	Control Flow	Control Flow
Activity final node		Object Flow	Object Flow	Object Flow			Object Flow					
Flow final node		Object Flow	Object Flow	Object Flow			Object Flow					
Input pin												
Output pin							Object Flow					
Value pin												
Object node		Object Flow	Object Flow	Object Flow			Object Flow		Object Flow	Object Flow	Object Flow	Object Flow
Central buffer node		Object Flow	Object Flow	Object Flow			Object Flow		Object Flow	Object Flow	Object Flow	Object Flow
Data store node		Object Flow	Object Flow	Object Flow			Object Flow		Object Flow	Object Flow	Object Flow	Object Flow

Interruptible activity region									
Expansion region	Exception handler								
Expansion node		Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow	Object Flow
Swimlane									
Structured Activity node	Exception handler					Control Flow	Control Flow	Control Flow	Control Flow
Conditional node	Exception handler					Control Flow	Control Flow	Control Flow	Control Flow
Loop node	Exception handler					Control Flow	Control Flow	Control Flow	Control Flow
Sequence node	Exception handler					Control Flow	Control Flow	Control Flow	Control Flow

Connection rules in activity diagram B

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Component diagram connection rules

	Component	Interface	Port	Instance specification
Component	Dependency, Generalization, Realization, Association, Aggregation, Composition, Usage	Dependency, Association, Aggregation, Composition, Usage	Dependency, Generalization, Realization, Association, Aggregation, Composition, Usage	Dependency, Realization
Interface	Dependency, Realization, Association, Aggregation, Composition, Usage	Dependency, Generalization, Realization, Association, Aggregation, Composition, Usage	Dependency, Generalization, Realization, Association, Aggregation, Composition	Dependency, Generalization, Realization, Association, Aggregation, Composition
Port	Dependency, Realization, Association, Aggregation, Composition, Usage	Dependency, Association, Aggregation, Composition, Usage	Dependency, Realization, Association, Aggregation, Composition	Dependency, Realization
Instance specification	Dependency, Realization	Dependency, Generalization, Realization, Association, Aggregation, Composition	Dependency	Dependency, Generalization, Realization, Link

Connection rules in component diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Deployment diagram connection rules

	Node	Artifact	Deployment specification	Component	Interface	Port	Instance specification
Node	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Deployment, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Deployment, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization
Artifact	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Dependency, Manifestation	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization
Deployment specification	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Dependency, Manifestation	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization
Component	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization	Dependency, Manifestation, Realization
Interface	Association, Aggregation, Composition, Dependency, Manifestation, Realization	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization	Association, Aggregation, Composition, Dependency, Manifestation, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization
Port	Dependency, Manifestation, Generalization, Realization	Dependency, Manifestation, Generalization, Realization	Dependency, Manifestation, Generalization, Realization	Association, Aggregation, Composition, Dependency, Manifestation, Generalization, Realization, Usage	Association, Aggregation, Composition, Dependency, Manifestation, Usage	Association, Aggregation, Composition, Dependency, Generalization, Realization	Dependency, Manifestation, Generalization, Realization
Instance specification	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization	Dependency, Manifestation, Realization	Association, Aggregation, Composition, Dependency, Manifestation, Generalization	Dependency, Manifestation, Realization	Link, Dependency, Manifestation, Deployment, Generalization, Realization

Connection rules in deployment diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Package diagram connection rules

	Package	Subsystem
Package	Dependency, Import, Access, Generalization, Realization, Merge	Dependency, Import, Access, Realization, Merge
Subsystem	Dependency, Import, Access, Realization, Merge	Dependency, Import, Access, Generalization, Realization, Merge

Connection rules in package diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Object diagram connection rules

	Instance specification	Class
Instance specification	Link, Dependency, Generalization, Realization	Association, Aggregation, Composition, Dependency, Generalization, Realization
Class	Association, Aggregation, Composition, Dependency, Generalization, Realization	Association, Aggregation, Composition, Dependency, Generalization, Realization

Connection rules in object diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Composite structure diagram connection rules

	Class	Part	Property	Interface	Port	Collaboration	Collaboration Use
Class	Dependency, Represents, Occurrence, Generalization, Realization	Dependency, Represents, Occurrence, Realization	Dependency, Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Generalization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Generalization, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization
Part	Dependency, Represents, Occurrence,	Connector, Dependency, Represents, Occurrence	Connector, Dependency, Represents, Occurrence	Dependency, Represents, Occurrence	Connector, Dependency, Represents, Occurrence	Dependency, Represents, Occurrence	Dependency, Represents, Occurrence,
Property	Dependency, Represents, Occurrence	Connector, Dependency, Represents, Occurrence	Connector, Dependency, Represents, Occurrence	Dependency, Represents, Occurrence	Connector, Dependency, Represents, Occurrence	Dependency, Represents, Occurrence	Dependency, Represents, Occurrence
Interface	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Generalization, Realization	Dependency, Represents, Occurrence, Realization	Dependency, Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Generalization, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Generalization, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization
Port	Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization	Connector, Dependency, Represents, Occurrence, Realization	Connector, Dependency, Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency, Represents, Occurrence	Connector, Association, Aggregation, Composition, Dependency, Represents, Occurrence, Realization	Dependency, Represents, Occurrence, Realization	Dependency, Represents, Occurrence, Realization
Collaboration	Association, Aggregation, Composition, Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency Represents, Occurrence	Association, Aggregation, Composition, Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency Represents, Occurrence, Realization
Collaboration Use	Association, Aggregation, Composition, Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency Represents, Occurrence	Dependency Represents, Occurrence, Realization	Association, Aggregation, Composition, Dependency Represents, Occurrence, Realization	Dependency Represents, Occurrence, Realization

Connection rules in composite structure diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Interaction overview diagram connection rules

	Interaction	Interaction use	Decision node	Merge node	Fork node	Join node	Initial node	Activity final
Interaction	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Interaction use	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Decision node	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Merge node	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Fork node	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Join node	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Initial node	Control flow	Control flow	Control flow	Control flow	Control flow	Control flow		Control flow
Activity final								

Connection rules in interaction overview diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Requirement diagram connection rules

	Requirement	Model	Testcase
Requirement	Composition, Derive, Trace		
Model	Satisfy, Refine		
Testcase	Verify,		

Connection rules in requirement diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Basic diagram connection rules

	Process	Decision	Business actor	Document
Process	Connector	Connector	Connector	Connector
Decision	Connector	Connector	Connector	Connector
Business actor	Connector	Connector	Connector	Connector
Document	Connector	Connector	Connector	Connector

Connection rules in basic diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Entity relationship diagram connection rules

	Entity	View	Sequence	Stored procedures	Stored procedure resultset	Triggers
Entity	One-to-one relationship, One-to-many relationship, Many-to-many relationship	One-to-one relationship, One-to-many relationship, Many-to-many relationship				
View	One-to-one relationship, One-to-many relationship, Many-to-many relationship	One-to-one relationship, One-to-many relationship, Many-to-many relationship				
Sequence						
Stored procedures						
Stored procedure resultset						
Triggers						

Connection rules in entity relationship diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

ORM diagram connection rules

	Class	Entity	View
Class	Association, Aggregation, Composition	Class-entity mapping	Class-entity mapping
Entity	Class-entity mapping	One-to-one relationship, One-to-many relationship, Many-to-many relationship	One-to-one relationship, One-to-many relationship, Many-to-many relationship
View	Class-entity mapping	One-to-one relationship, One-to-many relationship, Many-to-many relationship	One-to-one relationship, One-to-many relationship, Many-to-many relationship

Connection rules in orm diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business process diagram connection rules

	Task	Sub-process	Start event	Intermediate event	End event	Gateway	Choreography task	Choreography Sub-process	Call activity	Text annotation	Data Object	Pool/ Lane
Task	Sequence flow, Message flow	Sequence flow, Message flow	Message flow	Sequence flow, Message flow	Sequence flow,	Sequence flow, Message flow	Sequence flow,	Sequence flow,	Sequence flow, Message flow	Association, To-Direction Association	Association, To-Direction Association, Data Association	
Sub-process	Sequence flow, Message flow	Sequence flow, Message flow	Message flow	Sequence flow, Message flow	Sequence flow,	Sequence flow, Message flow	Sequence flow,	Sequence flow,	Sequence flow, Message flow	Association, To-Direction Association	Association, To-Direction Association, Data Association	
Start event	Sequence flow,	Sequence flow,		Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Association, To-Direction Association	Association, To-Direction Association	
Intermediate event	Sequence flow,	Sequence flow,		Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Association, To-Direction Association	Association, To-Direction Association	
End event	Message flow	Message flow	Message flow	Message flow					Message flow	Association, To-Direction Association	Association, To-Direction Association	
Gateway	Sequence flow, Message flow	Sequence flow, Message flow		Sequence flow, Message flow	Sequence flow,	Sequence flow, Message flow	Sequence flow,	Sequence flow,	Sequence flow, Message flow	Association, To-Direction Association	Association, To-Direction Association	
Choreography task	Sequence flow,	Sequence flow,		Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,		Association, To-Direction Association		
Choreography sub-process	Sequence flow,	Sequence flow,		Sequence flow,	Sequence flow,	Sequence flow,	Sequence flow,			Association, To-Direction Association		
Call activity	Sequence flow, Message flow	Sequence flow, Message flow	Message flow	Sequence flow, Message flow	Sequence flow,	Sequence flow, Message flow			Sequence flow, Message flow	Association, To-Direction Association	Association, To-Direction Association, Data Association	
Text annotation	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association		Association, To-Direction Association	Association, To-Direction Association
Data object	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association	Association, To-Direction Association			Association, To-Direction Association	Association, To-Direction Association	Data Association	
Pool/ lane										Association, To-Direction Association		

Connection rules in business process diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm

- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Conversation diagram connection rules

	Participant	Text annotation
Participant	Conversation link, Sub-conversation link, Call conversation link	Association
Text annotation	Association	

Connection rules in conversation diagram diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Data flow diagram connection rules

	Process	External entity	Data store
Process	Data Flow, Bidirectional data flow	Data Flow, Bidirectional data flow	Data Flow, Bidirectional data flow
External entity	Data Flow, Bidirectional data flow		
Data store	Data Flow, Bidirectional data flow		

Connection rules in data flow diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

EPC diagram connection rules

	Event	Function	And operator	Or operator	XOR operator	Organization unit	Process path	Information resource
Event		Control flow	Control flow	Control flow	Control flow		Control flow	
Function	Control flow	Control flow	Control flow	Control flow	Control flow			Information flow
And operator	Control flow	Control flow	Control flow	Control flow	Control flow			
Or operator	Control flow	Control flow	Control flow	Control flow	Control flow			
XOR operator	Control flow	Control flow	Control flow	Control flow	Control flow			
Organization unit		Organization unit assignment						
Process path	Control flow							
Information resource		Information flow						

Connection rules in EPC diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Process map diagram connection rules

	Process	Send	Receive
Process	Process link	Process link	Process link
Send	Process link	Process link	Process link
Receive	Process link	Process link	Process link

Connection rules in process map diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Organization chart connection rules

	Organization unit
Organization unit	Line

Connection rules in organization chart

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Fact diagram connection rules

	Term	Fact Type
Term	Fact Association, Generalization	Term-Fact Type Association
Fact Type		

Connection rules in fact diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Business motivation model diagram connection rules

	End	Vision	Goal	Objective	Means	Mission	Strategy	Tactic	Business Policy	Business Rule
End										
Vision										
Goal		Amplify	Composition							
Objective			Quantity	Composition						
Means										
Mission		Make Operative								
Strategy			Channel Efforts, Define, Require	Channel Efforts, Define, Require		Component of Plan for, Define, Require	Enable, Formulated Based on, Require, Composition		Define, Require	Define, Require
Tactic			Channel Efforts, Define, Require	Channel Efforts, Define, Require				Enable, Formulated Based on, Require, Composition	Define, Require	Define, Effect Enforcement Level, Require
Business Policy			Act as Regulation, Support Achievement	Act as Regulation, Support Achievement			Act as Regulation, Govern	Act as Regulation, Govern	Composition	Act as Regulation, Basis for
Business Rule			Act as Regulation,	Act as Regulation,			Act as Regulation, Govern	Act as Regulation, Govern		
Internal Influencer										
External Influencer										
Regulation										
Assessment	Affect Achievement, Use	Affect Achievement, Use	Affect Achievement, Use	Affect Achievement, Use	Affect Employment, Use	Affect Employment, Use	Affect Employment, Use	Affect Employment, Use	Affect Employment, Provide Impetus, Use	Affect Employment, Provide Impetus, Use
Risk									Provide Impetus	Provide Impetus
Potential Reward									Provide Impetus	Provide Impetus
Organization Unit	Define	Define	Define	Define	Define, Establish	Define, Establish	Define, Establish	Define, Establish	Define, Establish	Define, Establish
Business Process							Deliver, Realize	Deliver, Realize		
Asset										
Liability										

Connection rules in business motivation model diagram A

	Business Rule	Internal Influencer	External Influencer	Regulation	Assessment	Risk	Potential Reward	Organization Unit	Business Process	Asset	Liability
End											
Vision											

Goal											
Objective											
Means											
Mission											
Strategy	Define, Require							Define, Determine, Require		Define (Offerings), Require (Resources)	Define, Discharge, Require
Tactic	Define, Effect Enforcement Level, Require									Define,	Define, Discharge,
Business Policy	Act as Regulation, Basis for		Act as Regulation					Act as Regulation,	Act as Regulation, Govern	Act as Regulation, Govern	
Business Rule			Act as Regulation					Act as Regulation,	Act as Regulation, Govern, Guide	Act as Regulation, Govern	
Internal Influencer											
External Influencer											
Regulation											
Assessment	Affect Employment, Provide Impetus, Use	Judge, Use	Judge, Use	Judge, Use	Use	Identity, Use	Identity, Use				
Risk	Provide Impetus										
Potential Reward	Provide Impetus										
Organization Unit	Define, Establish	Define, Recognize, Source	Define, Recognize, Source	Define	Define, Make				Define, Responsible	Define (Offerings), Responsible	Define, Responsible
Business Process										Deliver (Offerings), Manage	
Asset											
Liability										Claim (Resources Only)	

Connection rules in business motivation model diagram B

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Archimate diagram connection rules

	Business actor	Business role	Business collaboration	Business process	Business function	Business interaction	Business event	Junction	Business service
Business actor	Aggregation, Composition, Specialization	Assignment							Realization
Business role	Assignment	Aggregation, Composition, Specialization		Assignment	Assignment	Assignment			
Business collaboration		Aggregation	Aggregation, Composition, Specialization			Assignment			
Business process		Assignment		Triggering, Aggregation, Composition, Specialization	Triggering	Triggering	Triggering	Triggering	Realization
Business function		Assignment		Triggering, Composition	Triggering, Aggregation, Composition, Specialization	Triggering	Triggering	Triggering	Realization
Business interaction		Assignment	Assignment	Triggering	Triggering	Triggering, Aggregation, Composition, Specialization	Triggering	Triggering	Realization
Business event				Triggering	Triggering	Triggering	Aggregation, Composition, Specialization		
Junction				Triggering	Triggering	Triggering	Triggering		
Business service				Used by	Used by	Used by			Aggregation, Composition, Specialization
Business interface		Association	Association						Assignment
Business object				Read access	Read access	Read access	Read access		Read access
Product	Used by								Aggregation
Contract									Read access
Representation									
Meaning									
Value									Association
Application collaboration						Assignment			
Application component				Assignment	Assignment	Assignment			
Application service				Used by	Used by	Used by			Used by
Application function									
Application interaction									
Application interface		Used by							
Data object									

Node
Device
System software
Infrastructure interface
Infrastructure service
Artifact

Connection rules in archimate diagram A

	Business interface	**Business object**	**Product**	**Contract**	**Representation**	**Meaning**	**Value**
Business actor	Line						
Business role	Association						
Business collaboration	Association						
Business process		Read access, Write access,					
Business function		Read access, Write access					
Business interaction		Read access, Write access					
Business event		Read access, Write access					
Junction							
Business service	Assignment	Read access, Write access		Read access, Write access			Association
Business interface	Aggregation, Specialization						
Business object		Association, Aggregation, Specialization			Association		
Product			Aggregation, Specialization	Aggregation			Association
Contract				Aggregation, Specialization			
Representation		Realization, Association			Aggregation, Specialization	Association	
Meaning					Association	Aggregation, Specialization	
Value			Association				Aggregation, Specialization
Application collaboration							
Application component							
Application service							
Application							

function	
Application interaction	
Application interface	
Data object	Realization
Node	
Device	
System software	
Infrastructure interface	
Infrastructure service	
Artifact	

Connection rules in archimate diagram B

	Application collaboration	**Application component**	**Application service**	**Application function**	**Application interaction**	**Application interface**	**Data object**
Business actor							
Business role						Require	
Business collaboration							
Business process		Assignment					
Business function							
Business interaction	Assignment	Assignment					
Business event							
Junction							
Business service				Used by	Used by		
Business interface		Used by					
Business object							
Product							
Contract							
Representation							
Meaning							
Value							
Application collaboration	Aggregation, Composition, Specialization	Association, Aggregation			Assignment	Provide, Require	
Application component	Association, Composition	Aggregation, Composition, Specialization	Realization	Assignment, Composition		Association, Provide, Require,	

						Composition	
Application service			Aggregation, Composition, Specialization		Assignment		Read access, Write access
Application function		Assignment	Realization	Aggregation, Composition, Specialization			Read access, Write access
Application interaction	Assignment		Realization		Aggregation, Composition, Specialization		Read access, Write access
Application interface		Association	Assignment			Aggregation, Composition, Specialization	
Data object			Read access, Write access	Read access, Write access	Read access, Write access		Association, Aggregation, Composition, Specialization
Node							
Device							
System software							
Infrastructure interface							
Infrastructure service							
Artifact		Realization					Realization

Connection rules in archimate diagram C

	Node	**Device**	**System software**	**Infrastructure interface**	**Infrastructure service**	**Artifact**
Business actor						
Business role						
Business collaboration						
Business process						
Business function						
Business interaction						
Business event						
Junction						
Business service						
Business interface						
Business object						

Product						
Contract						
Representation						
Meaning						
Value						
Application collaboration						
Application component						
Application service						
Application function						
Application interaction						
Application interface						
Data object						
Node	Communication path, Aggregation, Composition, Specialization	Composition	Composition	Provide, Require	Realization	Assignment
Device	Network	Aggregation, Composition, Specialization	Assignment	Provide, Require	Realization	Assignment
System software		Assignment	Aggregation, Composition, Specialization	Provide, Require	Realization	Assignment
Infrastructure interface				Aggregation, Composition, Specialization	Assignment	
Infrastructure service				Assignment	Aggregation, Composition, Specialization	Read access, Write access
Artifact	Assignment	Assignment	Assignment		Read access, Write access	Aggregation, Composition, Specialization

Connection rules in archimate diagram D

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Overview diagram connection rules

	Diagram overview
Diagram overview	Diagram containment, Directional generic connector

Connection rules in overview diagram

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Mind mapping diagram connection rules

	Node
Node	Branch, Link, From link, To link

Connection rules in mind mapping diagram

Related Resources

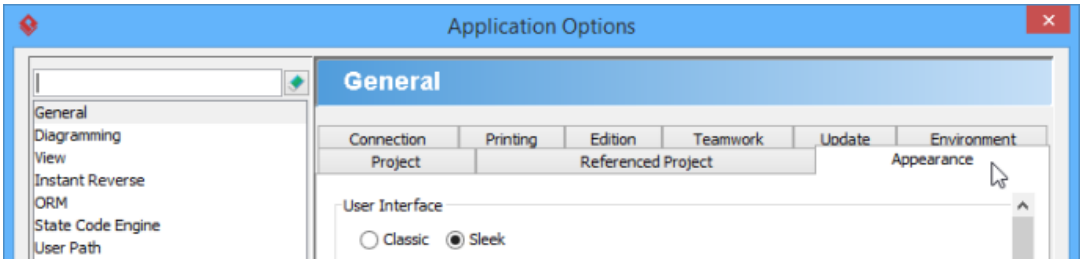
The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion

Multi-Languages support

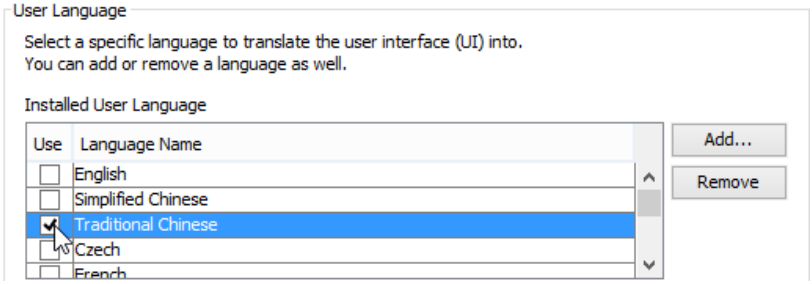
In Visual Paradigm, the default language is English. However, you might find it more convinient to use another language which is your first language to operate the software, so that you can understand the commands more easily and to perform the desired actions faster. Actually, Visual Paradigm supports the capability to allow users to switch to other langauges like Traditional Chinesem French and many others. In the following, we will show you the way to switch your User Interface into your preferred language. Please follow the simple steps below.

1. Select **Windows > Application** Options in the toolbar.
2. In **Application Options** window, select **General > Appearance**.



Application window

3. In the **Installed User Language** field, check your preferred language in the **Use** Column.



Check the preferred language

4. Click **OK**.
5. Restart Visual Paradigm.

Related Resources

The following resources may help you to learn more about the topic discussed in this page.

- New to Visual Paradigm? We have a lot of UML tutorials written to help you get started with Visual Paradigm
- Visual Paradigm on YouTube
- Visual Paradigm Know-How - Tips and tricks, Q&A, solutions to users' problems
- Contact us if you need any help or have any suggestion