VP-UML
Quick Start

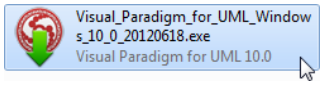
Last update: October 15, 2012

Table of Contents

Getting Started

Installing Visual Paradigm for UML (VP-UML)

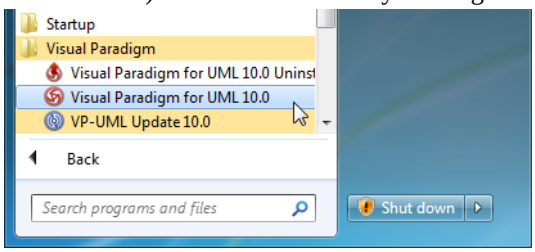
1. Run the VP-UML installer after your download it.



2. Click **Next** to proceed to the **License Agreement** page.
3. Read through the license agreement. Choose **I accept the agreement** after you finish reading the agreement and fully understand and accept the terms. Click **Next**.
4. Specify the directory for installing VP-UML. Click **Next** to continue.
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)** and **Visual Paradigm License File (*.zvpl)** checked if you want your system able to open the project file and the license key file. Click **Next** to start the file copying process.
7. Once the file copying is finished, you can choose to start VP-UML immediately, or just finish the installation without starting VP-UML. Choose the option **Don't Start** and click **Finish**. This will end the installation of VP-UML.

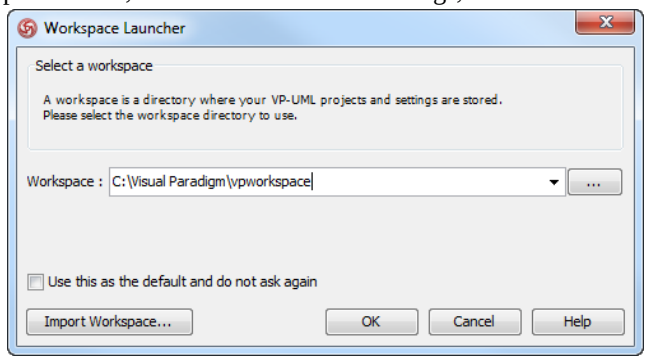
Starting VP-UML

Start VP-UML from the shortcut in the Start menu. If you selected not to create an entry in the Start menu (during the installation), you can look under the installation folder of VP-UML (the same path specified in step 4 in the section above) and start VP-UML by running **Visual Paradigm for UML 10.0.exe** in the **bin** folder.



Selecting Workspace

When you start VP-UML, you need to specify a folder for workspace. A workspace is a folder in which application preferences, like the look-and-feel settings, are stored.

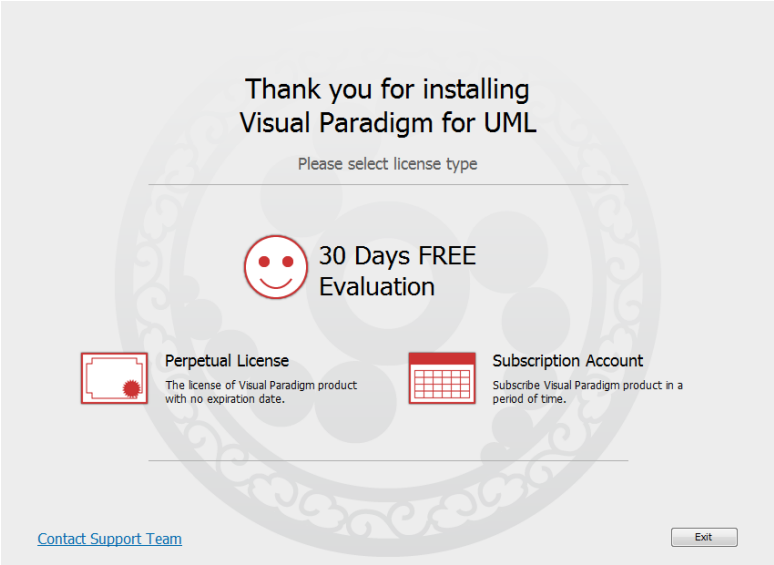


You need to select a workspace every time you start VP-UML but the workspace selected need not always be the same. If you want to keep applying the same set of preferences, always start VP-UML with the same workspace

folder. If you are moving to a new machine and want to keep the application preferences, simply copy the workspace folder over and choose it as workspace when starting VP-UML on the new machine. If you want a fresh working environment, select a new folder and proceed. Now, select any folder you like and click **OK** to continue. In this guide, we will choose *C:\Visual Paradigm\vpworkspace* as the workspace folder.

Selecting License Type

When you start VP-UML the first time, you are asked to select a way to "unlock" VP-UML.



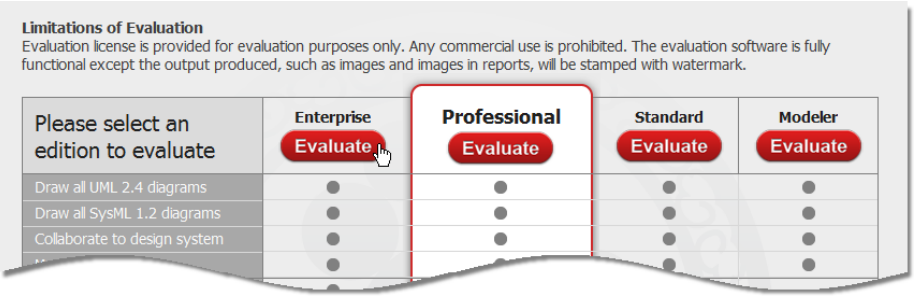
Depending on whether you own a purchased copy or an evaluation copy of VP-UML, you can proceed by following the steps below:

For Customers

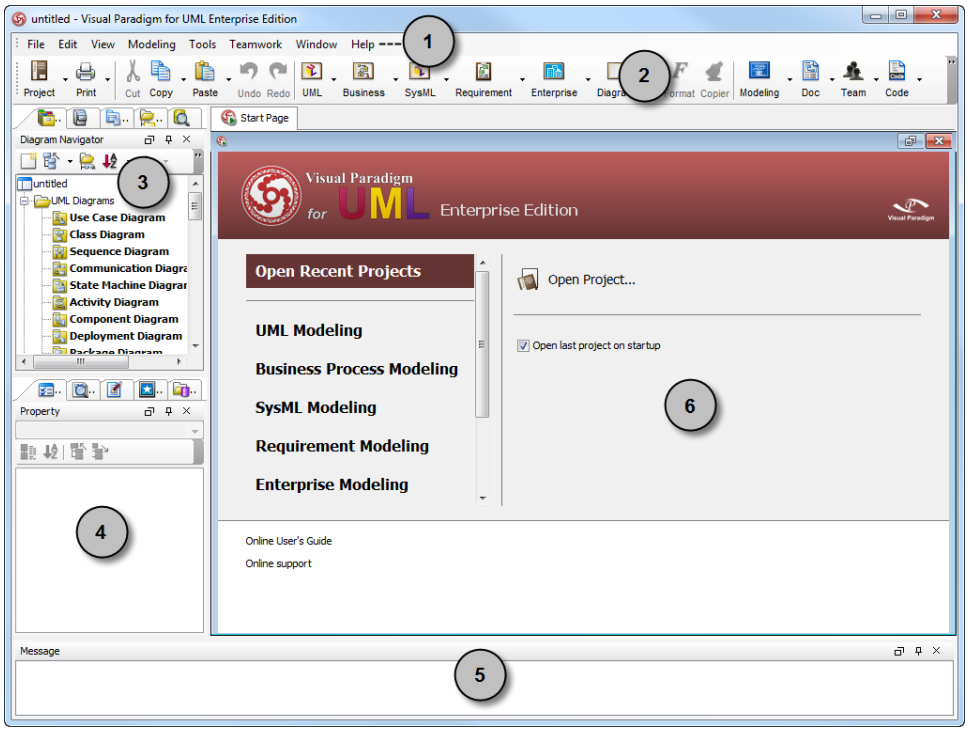
You should receive our notification (Email) with an activation code. The same activation code can also be found from the license key listed in your customer account. Copy the activation code first. Then, click on **Perpetual License**, paste the copied activation code and click **Activate** continue.

For Evaluators

If you want to evaluate VP-UML, click **30 Days FREE Evaluation**. You will then be asked to select the edition of product to evaluate. VP-UML features vary by product edition. 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. In this guide we will evaluate the Enterprise Edition.



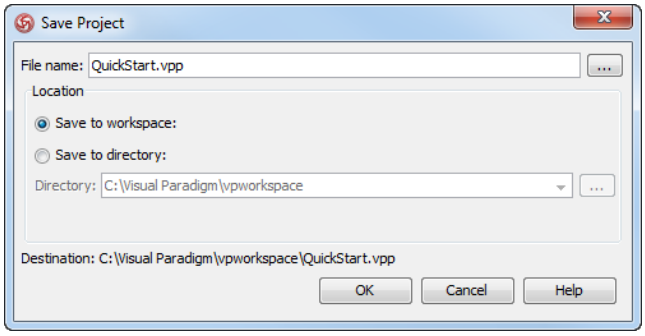
Environment



No.	Name	Description
1	Menu bar	A string of pull-down menus at the top of the window which allows you to select and perform various operations in VP-UML.
2	Toolbar	Located below the menu bar, it is an extension of the main menu. All buttons open up to groups of icons that perform various functions.
3	Diagram Navigator	A place where supported diagram types and existing diagrams are listed accordingly. It allows you to create or access existing diagrams by their types.
4	Properties Pane	A display of properties of the model element/shape currently being selected in the diagram pane.
5	Message Pane	All available information or warnings will be shown here.
6	Diagram Pane	The diagram will be displayed in diagram pane.

Saving and Opening Projects

To save your work, select either **File > Save Project** or **File > Save Project as…**. When you are saving a project for the first time, you will be asked to specify the location. You can save the new project either in your current workspace or a directory of your choice.



To open an existing project, select **File > Open Project…** from the main menu and select the project to open.

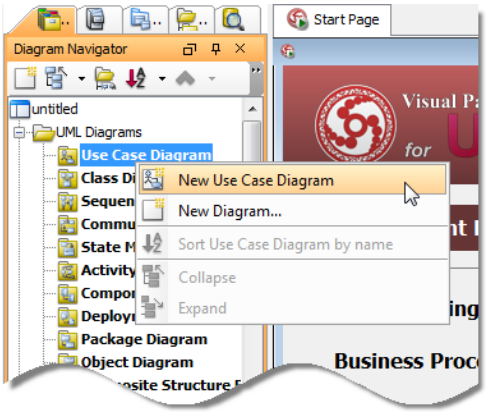
Diagramming

This section will go through the steps of creating diagrams, creating entities and connecting them. In addition, you will learn how to document entities and make diagrams more readable by using different colors in entities.

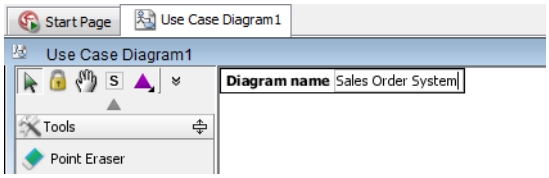
Creating Diagram

Let's create your first diagram in VP-UML. To create a diagram (e.g. a use case diagram):

1. Right-click on the **Use Case Diagram** node in the **Diagram Navigator**.
2. Select **New Use Case Diagram** from the popup menu.



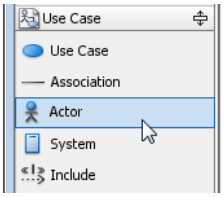
3. A blank use case diagram is created. You can give the diagram a name in the text box at the top left corner of the diagram. Name the diagram *Sales Order System*.



Creating and Connecting Shapes

You can create shapes either through the **Diagram Toolbar** (which appears to the left of the diagram) or the **Resource-centric Interface**. Let's create an actor from the **Diagram Toolbar** for now.

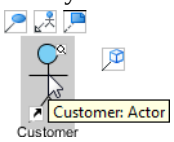
1. Select **Actor** from the **Diagram Toolbar**.



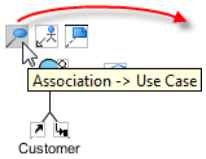
2. Click on the diagram to create an actor. Name the actor *Customer* and press **Enter** to confirm editing.

If you move your mouse pointer over a shape, you will see a number of resource icons surrounding the shape. From that shape, those icons help you create the next new shape (with connector) or create a connector to an existing shape. Let's create a use case from the actor above.

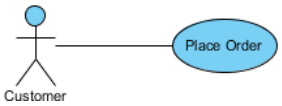
1. Place your mouse pointer over the actor shape we've just created.



2. Press on the resource icon **Association -> Use Case** and drag to the right.

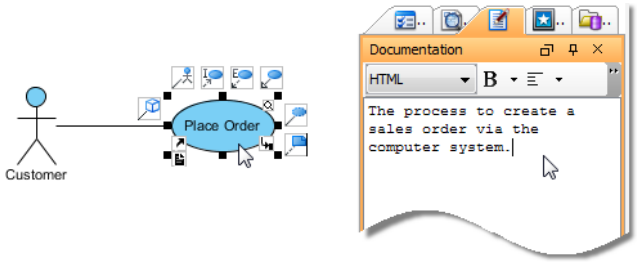


3. Release the mouse button. Name the use case *Place Order*. This will create a use case that associates with the *Customer* actor.

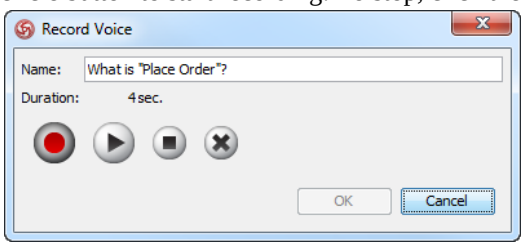


Documenting Model Elements

Generally speaking, naming model elements alone would not suffice to describe their details. To elaborate, you can enter additional details in the **Documentation Pane**. Simply go to the diagram and select the shape you want to document. At the bottom left of the application, open the **Documentation Pane** and fill in the details.



In addition to textual description, voice recording is also available. If your machine supports microphone usage, click the **Record** button at the bottom of the **Documentation Pane**. In the **Record Voice** window, click the red circle button to start recording. To stop, click the button with a square inside. To save your recording, click **OK**.

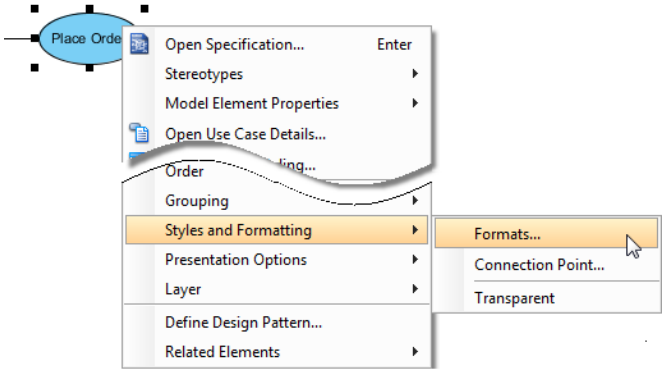


Note	Make sure your recording device is available when applying this feature.

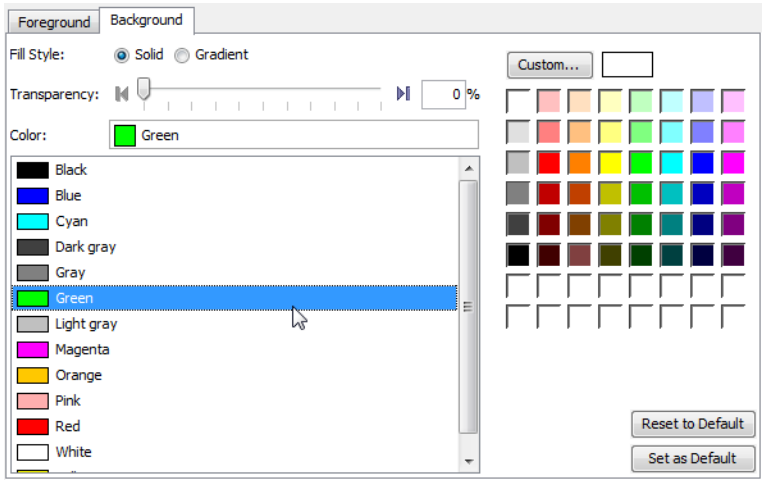
Setting Color, Line and Font Styles for Shapes

You can make your design more readable by formatting shapes differently, based on their natures. Let's change the color for the use case shape.

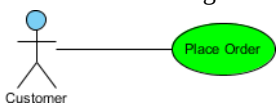
1. Right click on the use case shape and select **Styles and Formatting > Formats…** from the popup menu.



2. Open the **Background** tab in the **Formats** window. Select *Green* for color. Click **OK** to confirm the change.



The use case turns green.



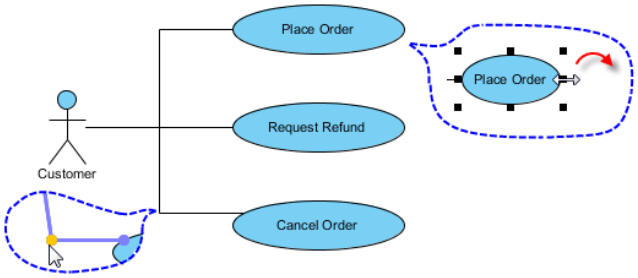
UML Modeling

Drawing Use Case Diagrams

A use case diagram is used to model and identify the functional requirements of a software system. In a use case diagram, all stakeholders and system goals are identified to elaborate how the system is formed. The main elements of a use case diagram include actor, use case and association (communication link).

An actor is any person or external system that interacts with the system to achieve a user goal (i.e. use case). The following simple use case diagram illustrates the use cases of a sales order system. *Customer*, an actor, interacts with the system to accomplish the goal of order placement, as modeled by the use case *Place Order*. There are other goals that the customer wants to accomplish, such as *Request Refund* and *Cancel Order*.

Now, apply the diagramming techniques described in the previous section to draw the diagram. You can drag the resize handler surrounding a shape to resize it. To reshape a connector, press on it and drag around to produce and move a pivot point (which appears as a bubble).

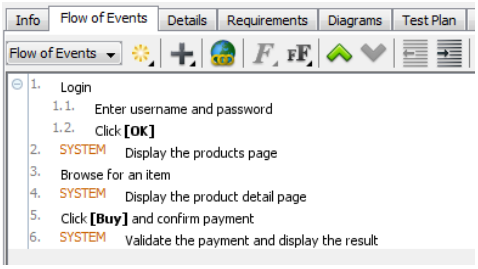


Documenting Use Case Flow of Events

[Professional Edition or above]

A use case's name tells us the 'what' aspect of a use case – what the users need. Meanwhile, the flow of events shows the 'how' aspect of a use case by explaining how a user's goal can be achieved. It is a technique for analyzing interaction between the actor and the system in accomplishing a use case. To work with the flow of events editor:

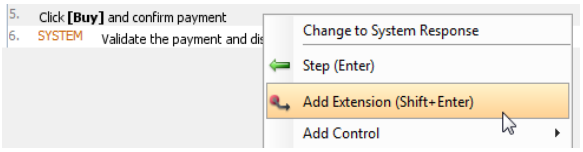
1. Right-click on a use case (e.g. *Place Order*) and select **Open Use Case Details…** from the popup menu.
2. Open the **Flow of Events** tab.
3. Enter the steps involved in accomplishing the **Place Order** use case.



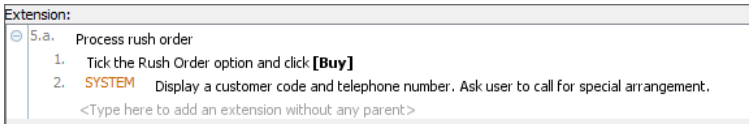
Tips:

- Create a new step by pressing **Enter**.
- Set a step as sub-step by pressing **Tab**.
- Add and declare a step responded by the system by clicking and selecting **System Response** from the popup menu.
- Make use of the formatting functions (e.g. bold, italic, etc) to format text.

4. At the bottom of the editor you can find the **Extension** section. An extension represents a variation of the use case being extended. The variation may be triggered when walking through the main flow, under certain conditions. Let's assume the place order use case is capable in handling rush order. Right click on step 5 where a user proceeds with buying. Select **Add Extension** from the popup menu.



5. Fill in the steps required for handling rush order.

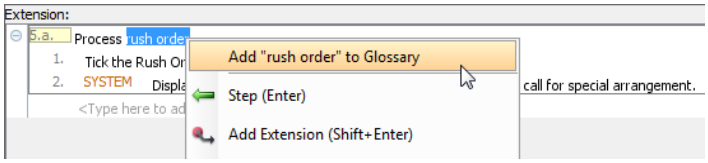


Building Glossary

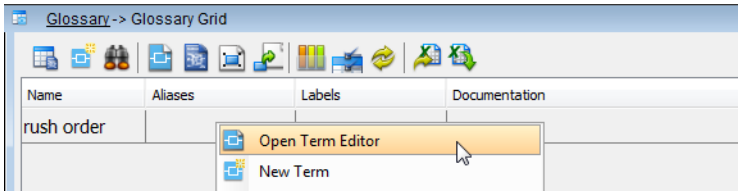
[Modeler Edition or above]

A glossary is a place where domain-specific vocabularies are stored and managed. And you can build a glossary by identifying terms in a flow of events.

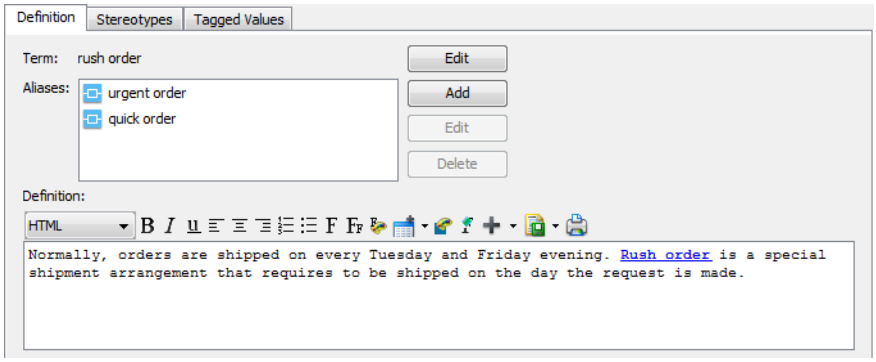
1. Suppose *rush order* is a key phrase that requires definition. Highlight it in the flow of events, right-click to select **Add "rush order" to Glossary** from the pop-up menu to make it a term.



2. This opens a glossary with the term online system homepage added. Right-click on the term to select **Open Term Editor** from the pop-up menu.



3. Specify its alias. In the **Term Editor**, click **Add** and enter *urgent order*. Add also *quick order* as alias. Enter the term's definition in the **Definition** section below.

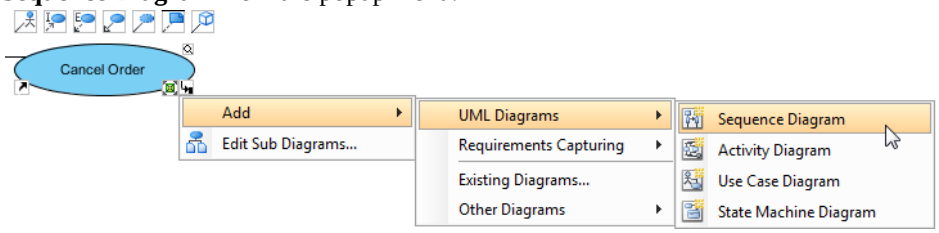


Note	Move your mouse pointer to the underlined term *online system homepage* in the **Flow of Events**, the documentation of the term will appear automatically.

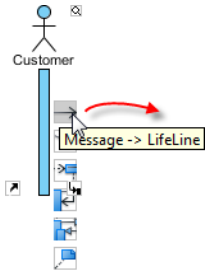
Drawing Sequence Diagram

A sequence diagram is used primarily to show interactions between objects that are represented as lifelines in a sequential order. More importantly, lifelines show all of their interaction points with other objects in events. A sequence diagram can be created by right-clicking **Sequence Diagram** in the **Diagram Navigator** and then selecting **New Sequence Diagram** from the pop-up menu. Alternatively, you can create a sequence diagram as a sub-diagram of a use case to model the interaction of that use case. Let's see how that works.

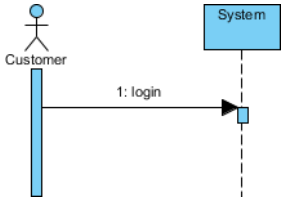
1. Move the mouse pointer over the use case *Cancel Order*.
2. Click on the tiny resource icon (**Sub Diagrams**) at bottom right and select **Add > UML Diagrams > Sequence Diagram** from the popup menu.



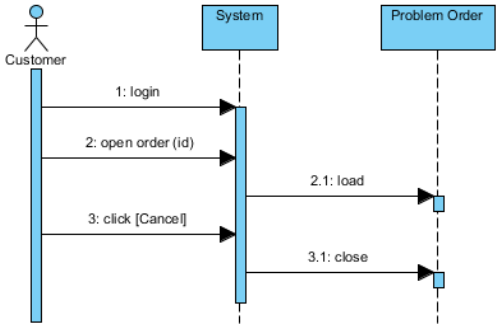
3. The actor *Customer* is created automatically. Let's model the interaction of how customer can cancel an order. Move the mouse pointer over the actor. Press on the **Message -> LifeLine** resource and drag it out.



4. Release the mouse button. Name the lifeline *System*, and the message *login*.



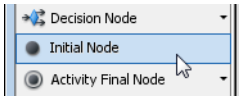
5. Complete the interaction by adding other messages and the *Order* lifeline.



Drawing Activity Diagram

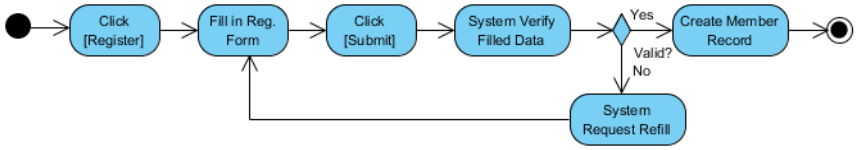
An activity diagram is essentially a flowchart, showing flow of control from one activity to another. Unlike a traditional flowchart, it can model the dynamic aspects of a system because it involves modeling the sequential steps in a computational process. Let's make use of activity diagram to model the registration process.

1. Create an activity diagram via the **Diagram Navigator**. Name the diagram *Register*.
2. Select **Initial Node** from the **Diagram Toolbar**. Click on the diagram to create an initial node, which represents the beginning of a flow.

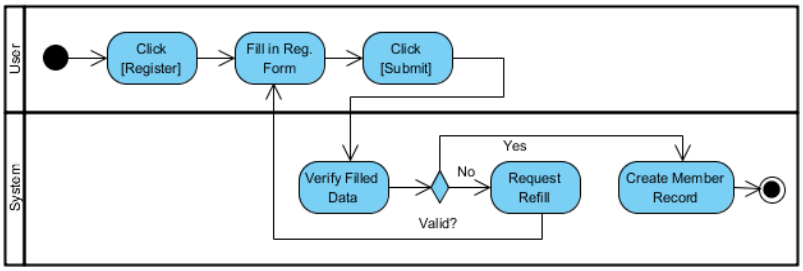


3. Press on the initial node's resource icon **Control Flow -> Action** and drag it. Release the mouse button to create an action and name it *Click [Register]*. This is the first action of the flow.

4. Complete the rest of the flow as shown below. The diamond shape is a decision node which leads to two possible subsequent flows.



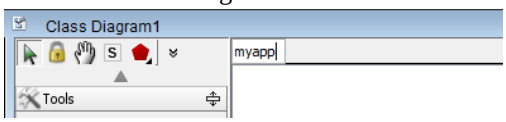
5. You can use a swimlane to group actions by participant. Select **Horizontal Swimlane** from the **Diagram Toolbar** and click on the diagram to create one. Double-click on the header of the partitions to name them. Drag the actions and other flow elements into the partitions appropriately.



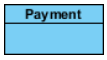
Drawing Class Diagram

A class diagram models the blueprints of objects required by a system and the relationships between them. Let's make use of class diagram to model the domain classes of the order processing system.

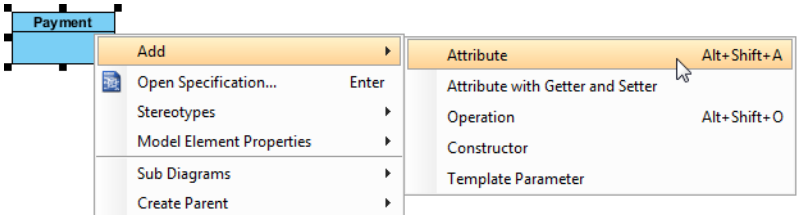
1. Create a class diagram via the **Diagram Navigator**. When the diagram is created, you are prompted to enter the package header. A package will be created with the entered string as name. The class diagram and the classes in the diagram will all be contained by the package. Enter *myapp*.



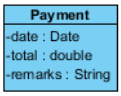
2. Name the diagram *Domain Model*.
3. Select **Class** from the **Diagram Toolbar** and click on the diagram to create a class. Name the class *Payment*.



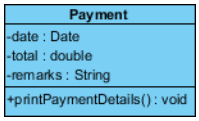
4. Add attributes to the class. Right click on the class and select **Add > Attribute** from the popup menu. Name it *date : Date* (The text after colon stands for attribute type).



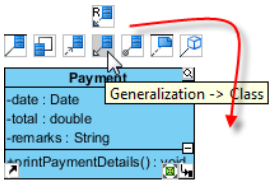
You can create as many attributes as you need by pressing **Enter** after inputting a new entry. Add two more attributes: *total : double* and *remarks : String*.



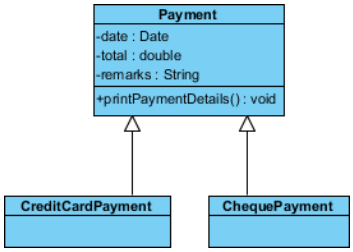
5. Add an operation by right clicking on the class and selecting **Add > Operation** from the popup menu. Name it *printPaymentDetails() : void*.



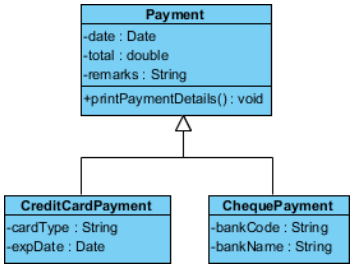
6. Generalization is a kind of relationship that models the "a-kind-of" relationship among two classes. Move the mouse over the *Payment* class, press on its resource icon **Generalization -> Class** and then drag it out.



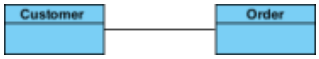
7. Release the mouse button to create the sub-class. Name it *CreditCardPayment*. Create another sub-class *ChequePayment* from *Payment* class.



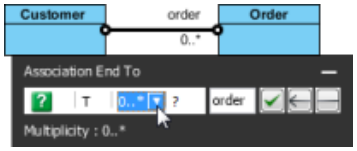
8. Add attributes to the sub-classes.



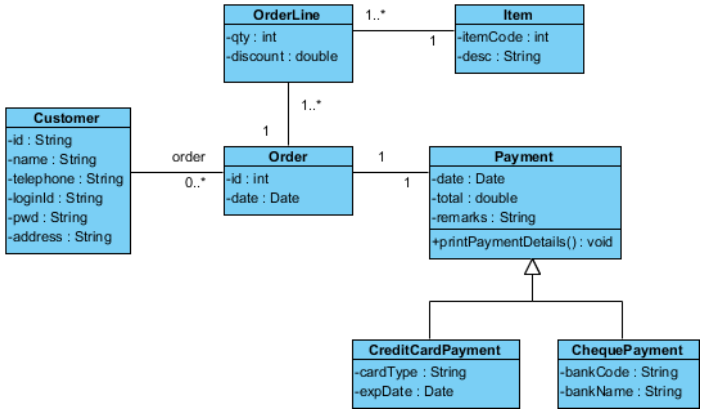
9. A class can be associated with another class. Create a class *Customer*. Move the mouse pointer over it and drag out the resource icon **Association -> Class**. Release the mouse button and name the new class *Order*.



10. To edit an association, double-click on it to open the **Association Editor**. Enter a name for the role of the association in the middle text box and adjust properties like multiplicity and navigability as needed.



11. Complete the diagram by creating other classes and relating them.



SysML Modeling

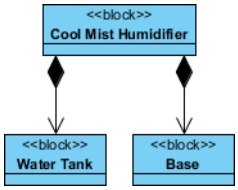
SysML, short for Systems Modeling Language, is a visual modeling language. SysML extends a subset of UML, making the language more software centric, catering the modeling needs for specifying, analyzing, designing, verifying and validating systems.

Drawing Block Definition Diagram

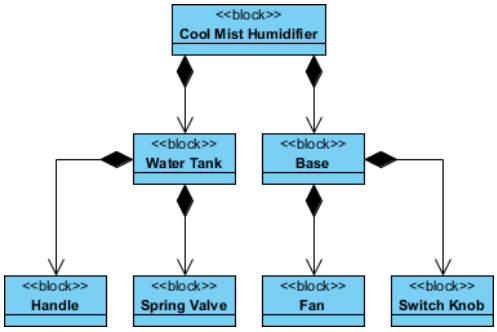
Block is one of the main notations in SysML. A block describes a composition of a system. Block definition diagram, composed of blocks, describes the system hierarchy, component classification, specification and inter-connections.

1. Let's model the parts of a cool mist humidifier in this section. We will go back to *QuickStart.vpp* in the next section. Create a new project. You can create a new project by selecting **File > New Project** from the main menu.
2. Create a block definition diagram via the **Diagram Navigator**. Name the diagram *Cool Mist Humidifier*.
3. Select **Block** in the **Diagram Toolbar** and click on the diagram to create a block. Name it *Cool Mist Humidifier*.

4. A cool humidifier consists of two main parts. The upper part is the water tank and the lower part is the base. To model this, move the mouse pointer over *Cool Mist Humidifier* and drag out the resource icon **Part Association -> Block**. Release the mouse button and name the new block *Water Tank*. Create another part and name it *Base*.



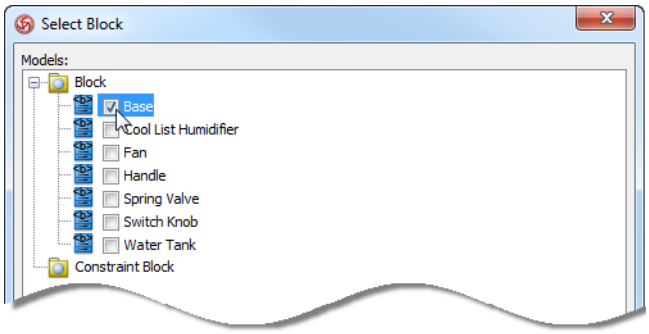
5. Water tank composes of a handle and a spring valve, while the base composes of a fan and a switch knob controlling the fan speed. Add the blocks and connect them with part association.



Drawing Internal Block Diagram

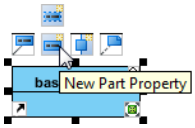
Internal Block Diagram describes the internal structure of a block in terms of properties and connectors between properties. We are going to draw an internal block diagram to describe the internal structure of a cool mist humidifier. Instead of including every little parts of a humidifier, we will just model a partial view that focuses on the parts that control the state and fan speed. The other parts will be omitted.

1. To describe the internal structure of the cool mist humidifier, right click on the *Cool Mist Humidifier* block in block diagram and select **Sub Diagrams > SysML > Internal Block Diagram > Create Internal Block Diagram** from the popup menu.
2. We want to focus on the base of the cool mist humidifier, which controls the state of humidifier as well as the amount of mist produced. Create a part for base. Select **Part Property** in diagram toolbar and click on the diagram. When you do this, you are prompted to select the type of block. Check *Base* and click **OK**.

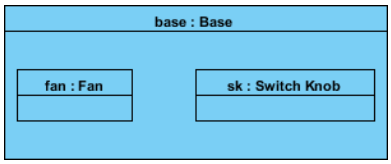


3. Name the part *base*.

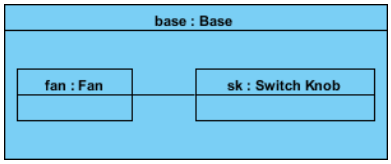
4. Fan and switch knob are both parts of a base. Move the mouse pointer over the base property and click on the resource **New Part Property** to create a part property that is contained by base.



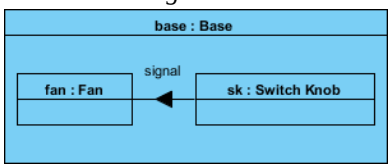
5. When you are asked to select a block as property type, select *Fan* and confirm. Name the property *fan*.
6. Create another property named *sk*, with *Switch Knob* as type.



7. Switch knob controls the fan speed. To represent this, move the mouse pointer over *sk : Switch Knob*. Drag out the resource icon **Connector > Part Property** and release the mouse button on *fan : Fan*.



8. Select **Item Flow** in diagram toolbar. Click at the connector between *sk : Switch Knob* and *fan : Fan*. Name the item flow *signal*.



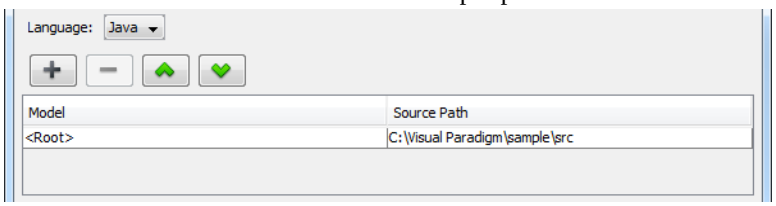
Code Generation

Java Round-Trip

[Standard Edition or above]

Round-trip engineering enables you to keep class model and source code in-sync. With Java round-trip, you can reverse a code-base to VP-UML as class model, analyze, and make changes such as adding missing classes, and then updating the changes to code, or vice versa.

To generate Java source code from class model, select **Tools > Code Engineering > Java Round-trip > Generate Code…** from the main menu. Enter the output path in the **Generate code** window and click **OK** to generate.



To reverse engineer a class from code, select **Tools > Code Engineering > Java Round-trip > Reverse Code…** from the main menu. The **Reverse Code** window will pop up asking you to select a source file path. Click **OK** to reverse.

C++ Round-Trip

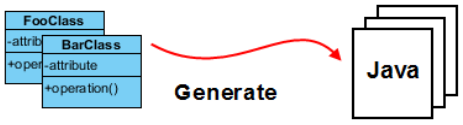
[Standard Edition or above]

To generate C++ for the whole project, click **Tools > Code Engineering > C++ Round-trip > Generate Code…** from the main menu. The **Generate Code** window will pop up asking you to select a path, click the **✚** button to add a path. After selecting one, click **OK** to generate.

To reverse class model from code, select **Tools > Code Engineering > C++ Round-trip > Reverse Code…** from the main menu. Select a source file path in the **Reverse Code** window and click **OK** to reverse.

Instant Generator

[Standard Edition or above]



Instant generator produces source code from your model at a particular instant. Unlike the code generation support in round-trip engineering, instant generator is a one-off. To generate code, select **Tools > Code Engineering > Instant Generator** from the main menu, then select the programming language in which to generate.

Instant Reverse

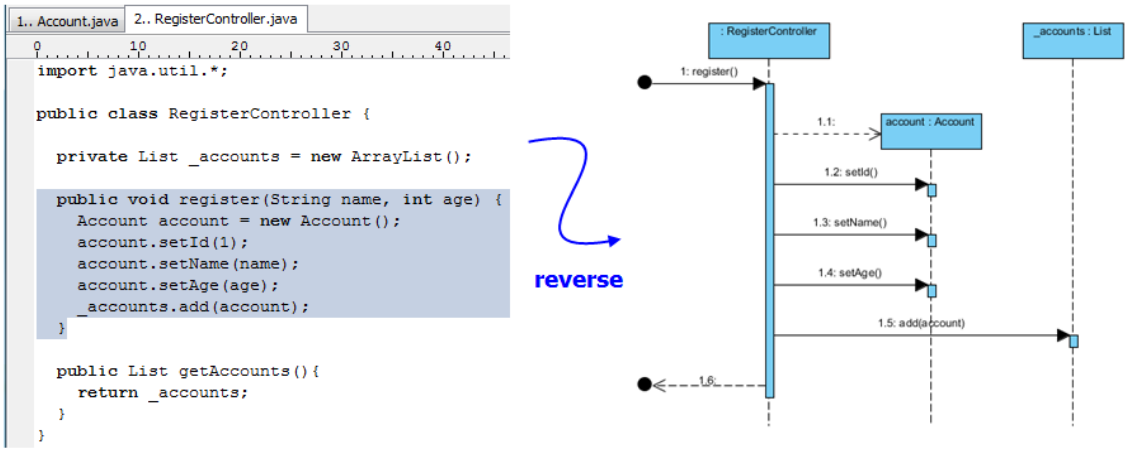
[Standard Edition or above]

Instant Reverse allows you to reverse different types of source into UML class models, such as Java source, Java classes, C++ source etc. To reverse, select **Tools > Code Engineering > Instant Reverse** from the main menu, then select the appropriate programming language. Select the source files and proceed.

Instant Reverse Java Code to Sequence Diagram

[Standard Edition or above]

Sequence diagram can help represent interactions between objects in runtime. VP-UML enables you to reverse your Java source code to sequence diagram. You can gain a better understanding of a piece of Java source code by reading its corresponding diagram, instead of looking at possibly a thousand lines of code. To reverse Java code to sequence diagram, select **Tools > Code Engineering > Instant Reverse > Java to Sequence Diagram…** from the main menu. Add the folder that contains the source code, continue and select the source file. Finally, visualize the code in a new diagram.



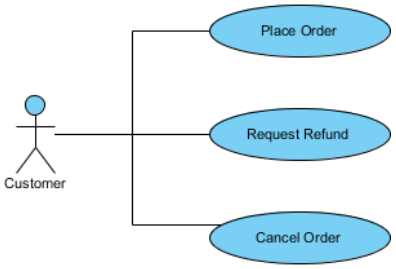
Reporting

Using Report Composer

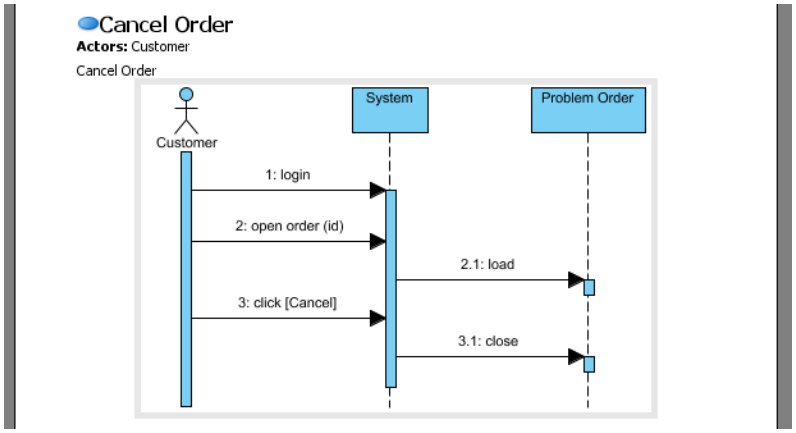
[Standard Edition or above]

You can develop professionally designed documentation using the Report Composer. Apart from the diagram-based report generation function, you can customize a report by adding elements to your report manually.

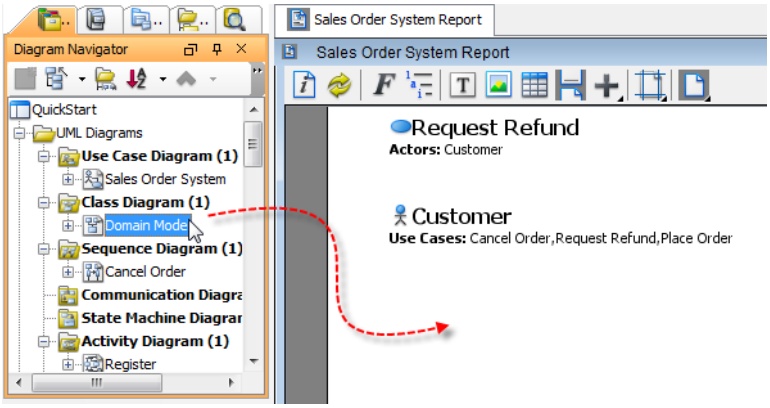
1. Let's continue with the *QuickStart.vpp*. Open the use case diagram *Sales Order System*. Create one if you do not have one already.



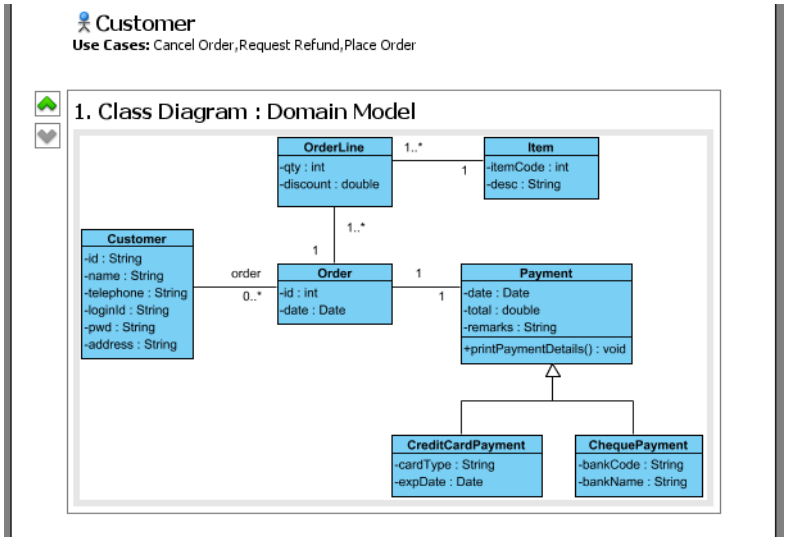
2. Right-click on the diagram and select **Utilities > Generate Use Case Report** from the popup menu. This creates a new report in report composer.



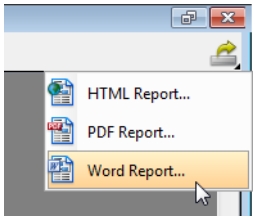
3. Scroll to the end of the report.
4. In the **Diagram Navigator**, press on the class diagram node *Domain Model*. Drag to the end of the report.



The result should look something like this.



5. Click on the **Export** button at the top right and select **Word Report…**.



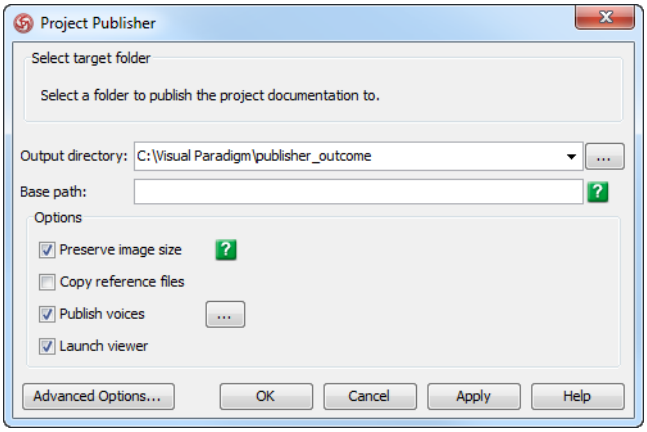
6. In the **Export Word** Report window, fill in the output path and click **Export** to produce a Word.

Project Publisher

[Standard Edition or above]

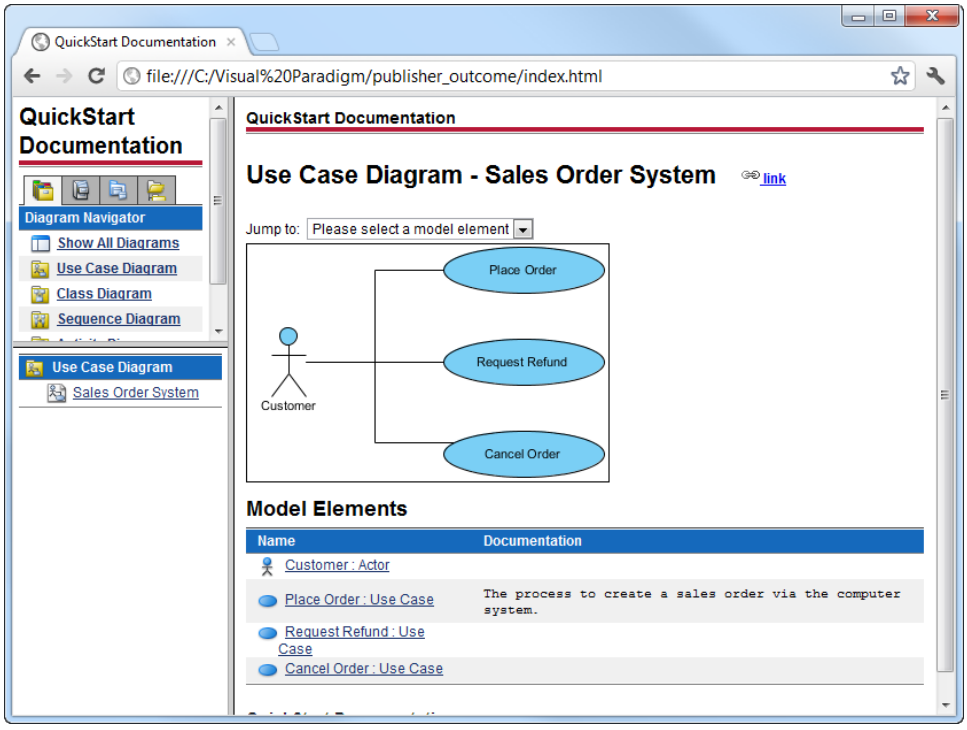
You can publish your project to Web format through the Project Publisher.

1. Select **Tools > Project Publisher…** from the main menu.
2. In the **Project Publisher** window, specify the output directory, which is the folder for storing the files to publish.

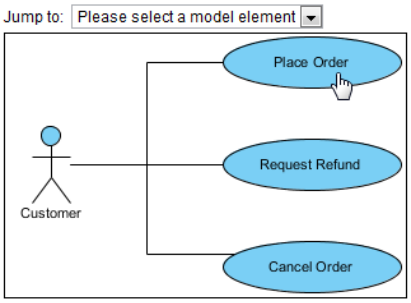


3. Click **OK**. When finished, you can read the published content in a web browser.

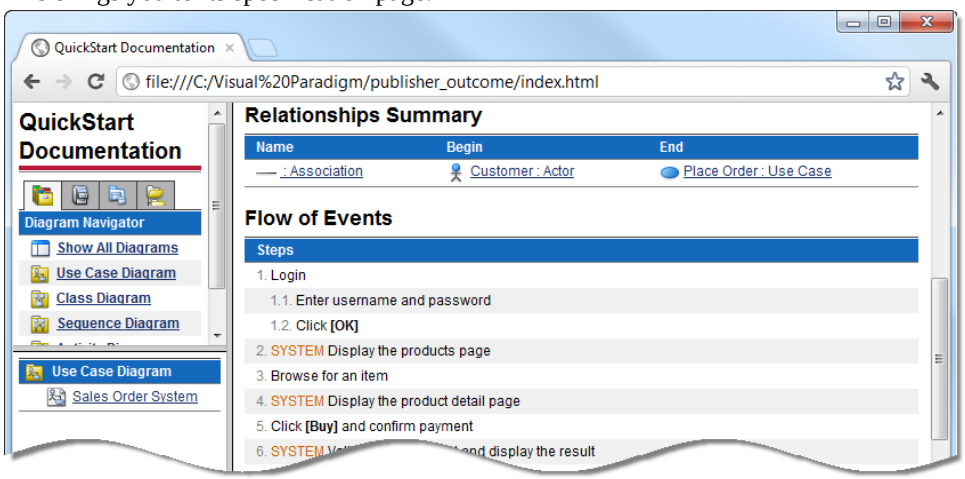
4. Open the use case diagram *Sales Order System* in published page.



5. Click on the use case *Place Order* on diagram.



This brings you to its specification page.



Modeling Collaboratively and Concurrently

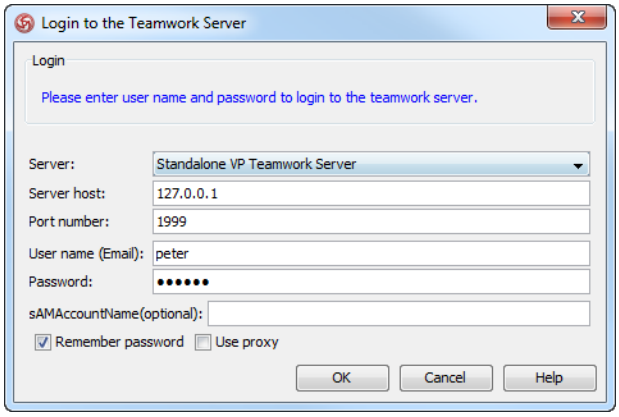
If you work as a team and need to share your design with your teammates, or work together on the same design, you need to make use of the team collaboration support. VP-UML supports version controlling systems like Teamwork Server, SVN, Perforce, ClearCase and CVS. In this guide, we will cover the import of project into Teamwork Server, and common operations like checkout, commit and update of project.

In this section, you need to simulate two persons in the same team - Peter and Stephen. They are both involved in the *Sales Order System* project. Before we continue, make sure you have Teamwork Server installed. You can download the Teamwork Server here. Installation of Teamwork Server is described in the user's guide page here.

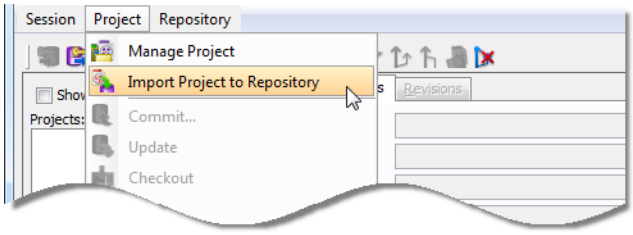
Share Design with Visual Paradigm Teamwork Server

Teamwork Server is a product developed by Visual Paradigm for supporting collaborative and concurrent modeling. To share your design with others:

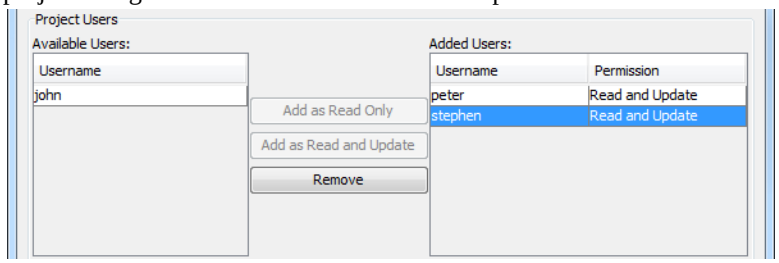
1. In VP-UML, select **Teamwork > Open Teamwork Client…** from the main menu.
2. In the **Login** window, keep selecting **Standalone VP Teamwork Server** as **Server** and fill in the connection information. Click **OK** to proceed. You should consult your system administrator about the login detail.



3. When the **Manage Project** window appears, skip it by clicking **OK** at the bottom of the window.
4. In the **Teamwork Client** window, select **Project > Import Project to Repository** from the main menu.



5. In the **Import Project** window, keep *QuickStart* to be the project name. Assign other teammates to this project and grant them with read and/or write permissions. Click **OK** to confirm importing.



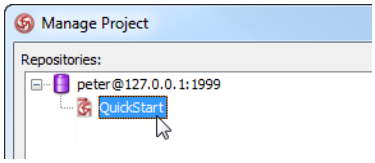
You have now imported the project into the server, and checked it out. Stephen, your colleague, can start working on the project you have imported.

Checkout and Open Project

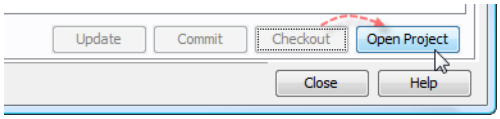
No matter you are using Teamwork Server, SVN, Perforce, ClearCase or CVS, you need to checkout a project from server and open it in order to work on it. If you are the one who imported the project into server, you do not need to perform checkout as this was done for you right after the import. However, you still need to open the project.

Now, play the role of another team member. Take the following steps to checkout and open project.

1. Start VP-UML in a new workspace.
2. Select **Teamwork > Open Teamwork Client…** from the main menu.
3. In the **Login** window, fill in the connection information of the other member and click **OK**.
4. In the **Manage Project** window, double click on the *QuickStart* project. Click **OK** to continue.



5. At the bottom right corner of the **Teamwork Client** window, click **Checkout**, and then click **Open Project**. Now, you have opened the project, and can start working on it.



Commit

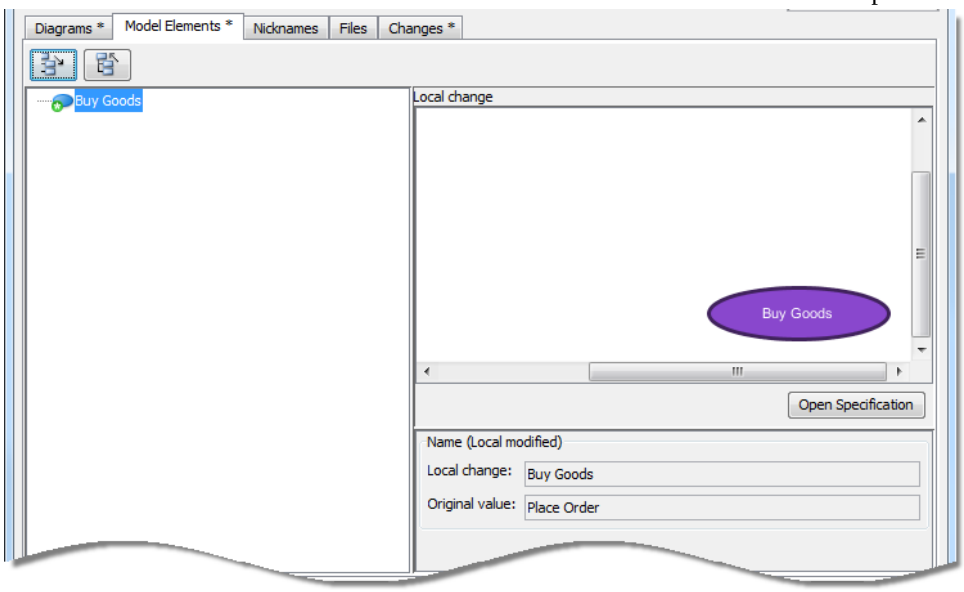
Commit is the process of uploading changes done in the working copy back to server. As you, the team member, make changes in a project, you can share your works by committing those changes to the server, and let others to update the changes from server. (Update will be discussed in next section)

Play the role of the team member who imported the project into server. Take the following steps to try out the commit process.

1. In **Teamwork Client** window, open the project you just imported to server.
2. Open the use case diagram *Sales Order System*.
3. Rename the use case *Place Order* to *Buy Goods*.

4. Select **Teamwork > Commit…** from the main menu.

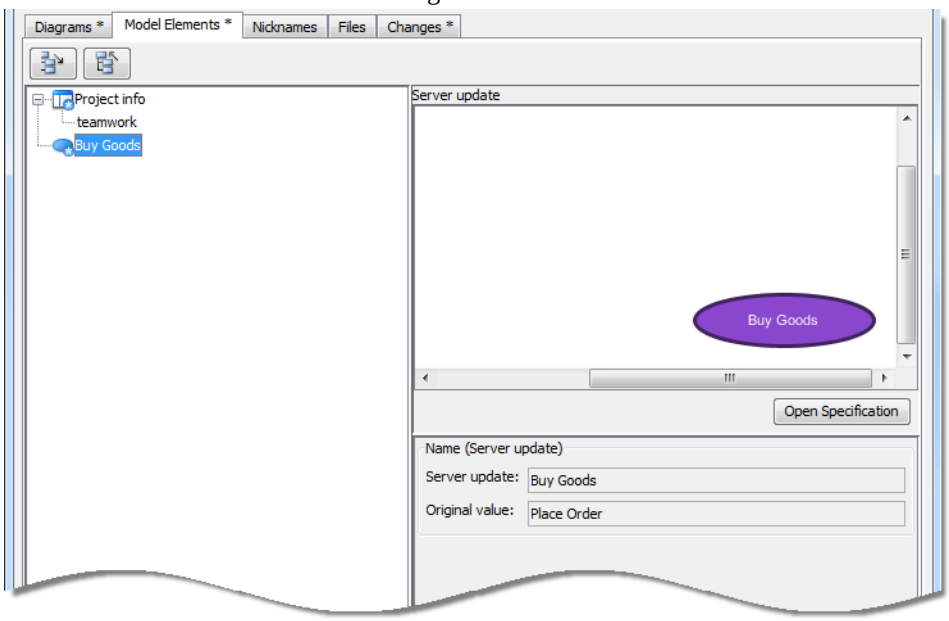
5. The **Commit** window appears. It lets you review the changes to be committed, solve conflicts (if any), and add comments to describe the commit action. Click **Commit** to continue and complete the commit process.



Update

Update is the process of refreshing the working copy by merging changes that others have made and committed to server. Let's say you are now Stephen. And you need to update the changes made by John.

1. Select **Teamwork > Update…** from the main menu.
2. The **Update** window appears. It lets you review the changes to be merged into your working copy. Open the **Model Elements** tab to see the changes.

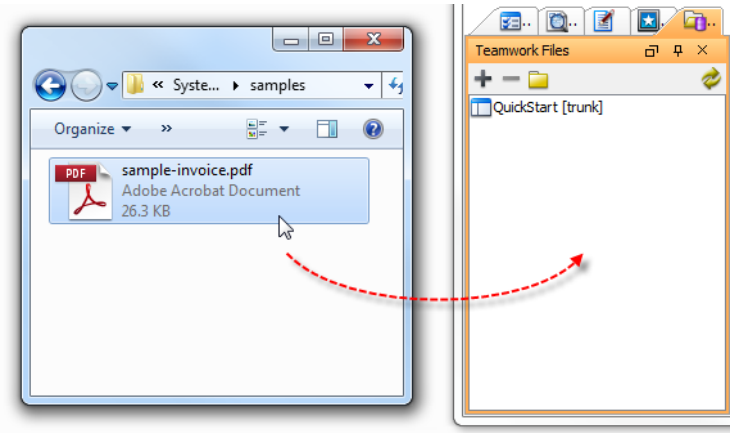


3. Click **Update** to continue and complete updating.

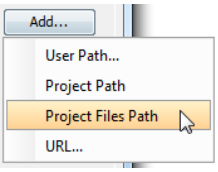
File Referencing with Teamwork Support

When modeling, there may be external resources you want to attach to a model which help describe it in detail or include data that cannot be modeled, like a text document. You can do this by adding file references to models. If you are working in a team-based environment with Teamwork Server, you do not need to copy any referenced files for other team members to open. Instead, you could commit your model along with the referenced files to the server.

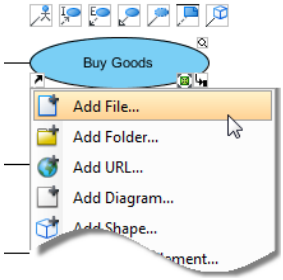
1. Open the **Teamwork Files** pane in the panes group at the bottom left of the user interface.
2. Drag a file to the **Teamwork Files** pane.



3. Click **OK** when you are asked to confirm putting the file to workspace. You must click **OK** here. If not, the file won't be added as a teamwork file nor committed to server.
4. Select **Tools > Application Options** from the main menu.
5. In the **Application Options** window, select **User Path** on the left hand side.
6. On the right hand side, click **Add...** and select **Project Files Path** from the popup menu. Click **OK** at the bottom of the **Options** window to close it.

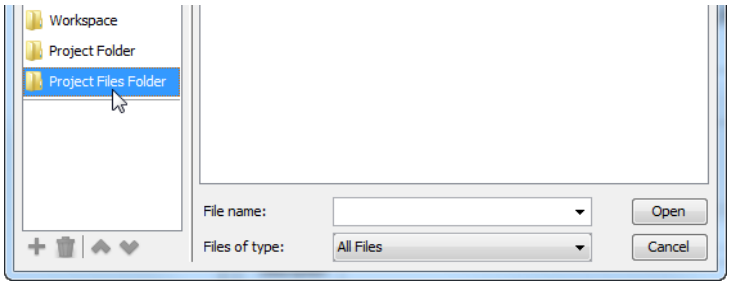


7. Move the mouse pointer over the use case *Buy Goods* to show the resource icons.
8. Click on the **References** resource icon at the bottom left of the requirement shape and select **Add File...** from the popup menu.

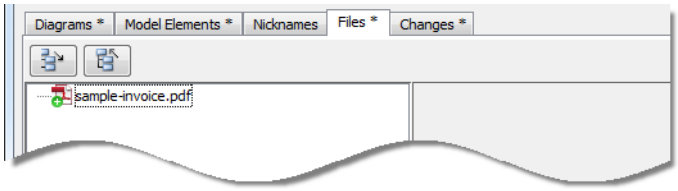


9. In the **Use Case Specification** window, click on the button **...** next to the **Path** field.

10. Click on the shortcut **Project Files Folder**. Note that if you have not completed step 4 to 6, you will not see this shortcut.



11. Select the teamwork file and click **Open** at the bottom right to choose it.
12. Click **OK** in **Use Case Specification** to return to the diagram.
13. Commit the changes to server. Note that the commit action will bring along the teamwork file(s) to server. You may check it in the **Commit** window.



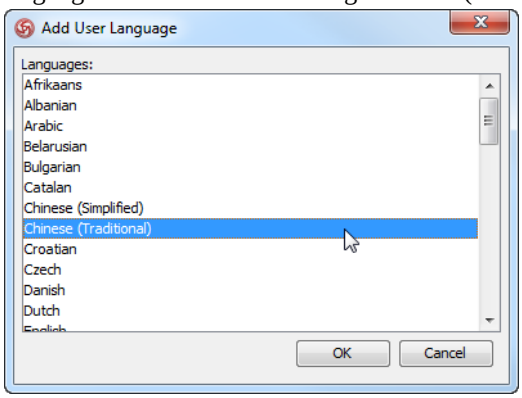
Advanced Modeling

Using Nicknamer

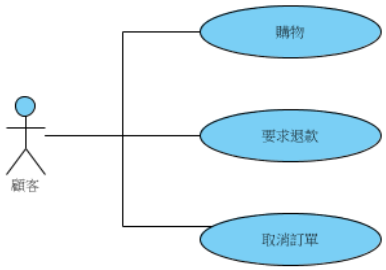
[Standard Edition or above]

Nicknamer is a feature which helps you to present a model with labels in different languages. This is particularly useful to multinational corporations where there's often a need in presenting a model in multiple languages for different regions. As localization is created on the fly when requested, there is no need to keep multiple versions for different languages. That means you need to modify one version only if there are any changes.

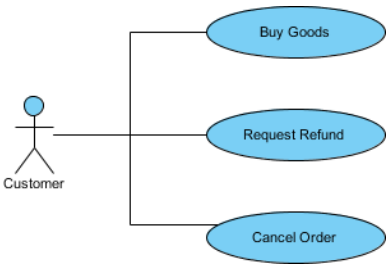
1. Select **View > Nicknames > Configure Nicknames…** from the main menu.
2. In the **Configure Nickname** window, click **Add User Language** and select in the popup window a language to add as nickname. E.g. Chinese (Traditional).



3. Click **OK** to close the popup window. Click **OK** again to return to the diagram. Modify the model for the added nickname.



4. To open the English (original) version of the model. Select **View > Nicknames > Original** from the main menu.

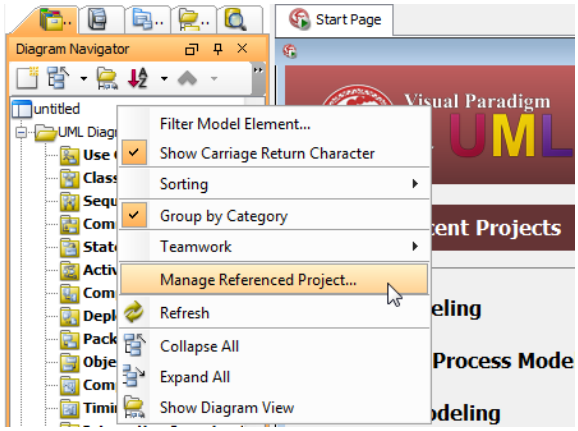


Project Referencing

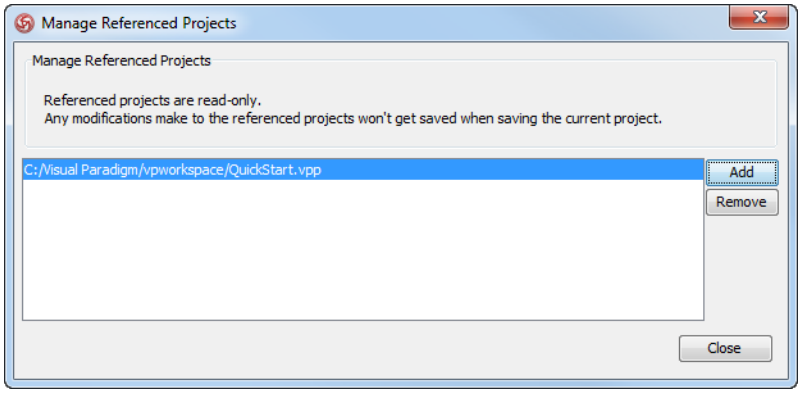
[Standard Edition or above]

To avoid creating the same things (e.g. a class) over and over again, it would be useful to have a generic library to keep components for reuse. When you make any changes to the components in the library, those changes will ripple down to where the components are actually used. In VP-UML, we call this generic library a "Reference Project."

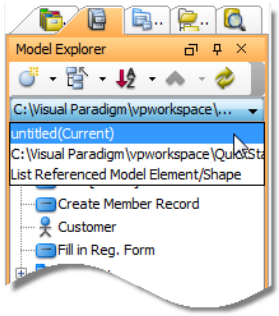
1. Create a new project in VP-UML. You can create a new project by selecting **File > New Project** from the main menu.
2. Right click on the background of **Diagram Navigator**/**Model Explorer** and select **Manage Referenced Project** from the popup menu.



3. Click **Add** in the **Manage Referenced Projects** window. Select the *QuickStart* project to reference to.

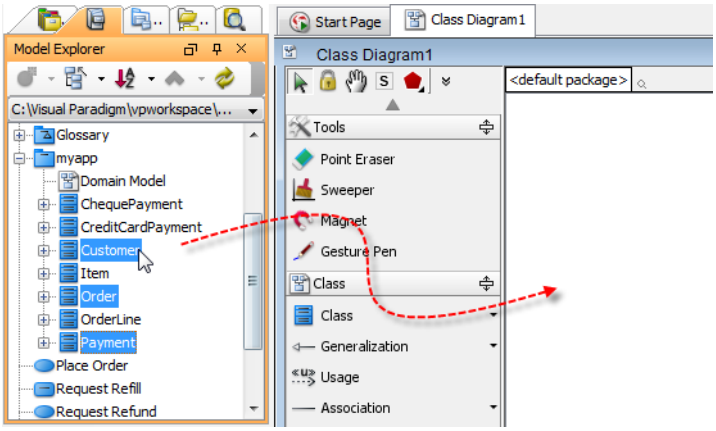


4. Click **Close**.
5. The referenced project(s) are listed in the drop down menu at the top of the **Model Explorer**. You can switch between the current project and the referenced project(s) through the drop down menu to see the elements in them.

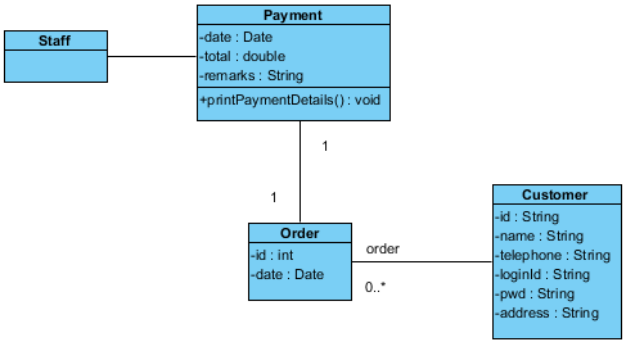


6. Create a new class diagram. Drag and drop the classes *Customer*, *Order* and *Payment* from **Model Explorer** to the diagram. Make sure you have selected the referenced project in the drop down menu in

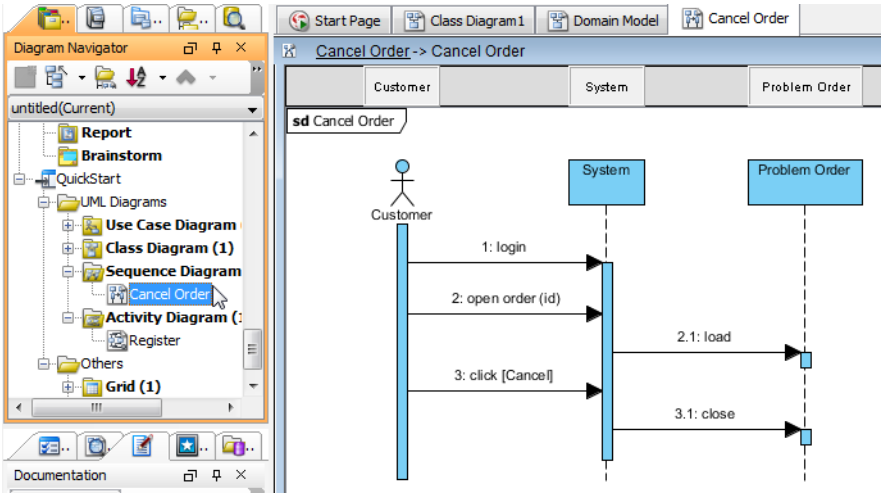
order to list the elements in it.



7. You can create elements in current project and connect them with referenced project data. Create a class *Staff* in the class diagram. Associate it with the *Payment* class referenced.



8. Open the **Diagram Navigator** and scroll to the bottom. Diagrams in the referenced project are listed there. You can double click on a diagram node to read it.



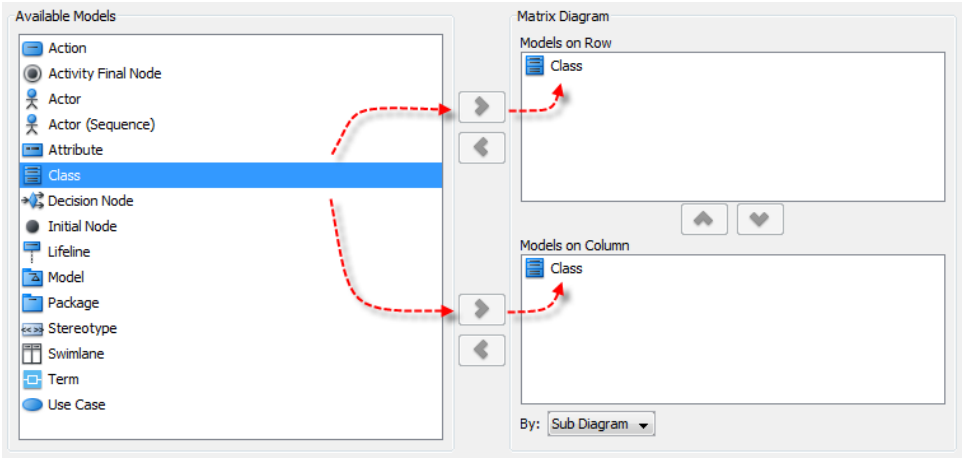
The benefit of using referenced model is to prevent your working project from becoming oversized as the information of referenced model will not be stored. However, since the referenced model is read-only from its source project, you cannot create a child to it. To deal with this problem, you can create mirror for parent-type elements such as package (Right click on a referenced element and select **Create Mirror Model Element**). The mirrored model element is also read-only on its properties, however, you can add a child model to it.

Impact Analysis with Matrix

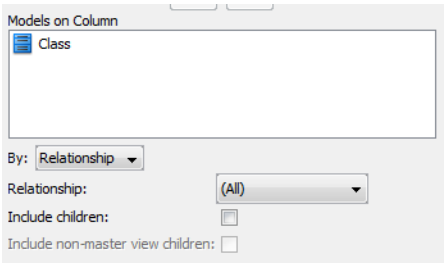
[Professional Edition or above]

If you were to make a change to some model elements, it would be important to know which other elements will get affected because of it. Impact Analysis can help you with that. There are three options, Matrix, Analysis Diagram and Chart, to choose from, depending on the scope of the analysis you need. Matrix (diagram) is a tool that helps you identify the relationship between model elements of specific type(s), so as to study the consequence of making certain changes. Let's try.

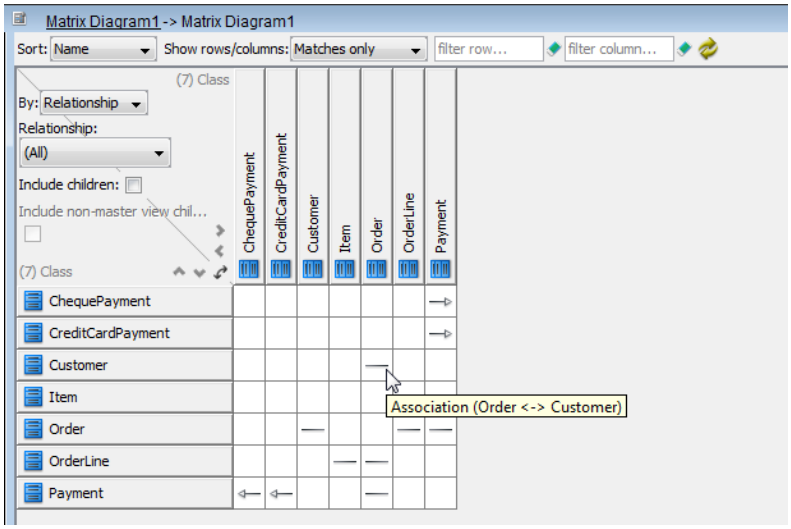
1. Continue with the *QuickStart* project (*QuickStart.vpp*). Create a matrix diagram via the **Diagram Navigator**, under the category of **Impact Analysis**.
2. You need to configure the type of elements to list in rows and columns. Let's say you want to delete some classes and you want to be certain that such action won't damage the integrity of the model. Therefore, you want to see the relationships among classes. Select **Class** from the list of available models. Click on the upper and lower arrow button to make classes list in both row and column.



3. You want to see the relationships (e.g. association, dependency, etc.) among classes. Select **Relationship** for **By** (By here means to compare row and column items by the selected criterion)



4. Click **OK**. This produces a matrix which lists the classes in rows and columns, showing their relationships in cells.



Let's say you are thinking about deleting the *Payment* class. From reading the matrix, you realize that the *Payment* class is a super class of *CreditCardPayment* and *ChequePayment*. Deleting the super class *Payment* may risk losing data. So you'd probably need to consider withdrawing the deletion or to move the data from super class to sub class, etc.