

About Visual Paradigm for UML

Visual Paradigm for UML (VP-UML) is a powerful, cross-platform and yet the most easy-to-use visual UML modeling and CASE tool. VP-UML 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 VP-UML, 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 and 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

You can also export Business process diagram to BPEL.

Object-Relational mapping

Object-Relational Mapping enables you to access relational database in an object relational approach when coding. VP-UML 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:

- VP Teamwork Server (Need to buy Visual Paradigm Teamwork Server additionally)
- CVS
- Subversion
- Perforce

Interoperability

The interoperability support allows you to exchange model data with other tools. The following tools are supported:

	Import	Export
Telelogic Modeler	<input checked="" type="checkbox"/>	
Rational Rose	<input checked="" type="checkbox"/>	
ERwin Data Modeler project	<input checked="" type="checkbox"/>	
Rational Software Architect	<input checked="" type="checkbox"/>	
Rational DNX	<input checked="" type="checkbox"/>	
NetBeans 6.x UML diagrams	<input checked="" type="checkbox"/>	
Visio drawing	<input checked="" type="checkbox"/>	
BPEL for Oracle workflow engine	<input checked="" type="checkbox"/>	
BPEL for JBoss workflow engine	<input checked="" type="checkbox"/>	
Diagram (JPG, PNG, SVG, EMF, PDF)	<input checked="" type="checkbox"/>	
Microsoft Excel	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
EMF based UML2 model	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
XMI (1.0, 1.2, 2.1)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
XML (native)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
VP project	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Microsoft Word document (for use case model)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

List of standard or tools that are covered by interoperability support

Code engineering

Instant Reverse and Instant Generator are provided for reversing engineering and forward engineering. In addition, the Java Round-Trip engineering support lets you keep code and model in-sync. Below are the type of source code that can be reversed or generated through Instant Reverse and Instant Generator.

	Instant Reverse	Instant Generator
Java	✓	✓
C++	✓	✓
XML Schema	✓	✓
PHP	✓	✓
Python Source	✓	✓
Objective-C	✓	✓
CORBA IDL Source	✓	✓
.NET dll or .exe files (binary)	✓	
CORBA IDL Source	✓	
XML (structure)	✓	
JDBC	✓	
Hibernate	✓	
C#		✓
VB.NET		✓
ODL		✓
ActionScript		✓
Delphi		✓
Perl		✓
Ada95		✓
Ruby		✓

List of source code supported by Instant Reverse and/or Generator

IDE integration

Support full software development life-cycle, from analysis to design, and from design to implementation with your most favorite IDE. Below are the supported IDEs:

- Eclipse
- NetBeans/Sun ONE
- IntelliJ IDEA

Documentation generation

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

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

Editions

Below are the kinds of features that can be found in each edition of VP-UML. For the support of specific features, please refer to the Features section.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
UML Support	✓	✓	✓	✓	✓	✓	#
Requirements Management	✓	✓	✓	✓	✓	✓	#
User Interface Designer	✓						#
Business Process Modeling	✓						#
Animacian		✓					
Mind Mapping		✓					
Entity Relationship Diagram	✓	✓	✓	✓	✓	✓	#
ORM Diagram	✓	✓	✓	✓	✓	✓	#
Database Modeling	✓	✓					
Object-Relational Mapping (Java)	✓	✓					
Object-Relational Mapping (.NET)	✓						
General Visual Modeling	✓	✓	✓	✓	✓	✓	
Project Referencing	✓	✓	✓				
Visual Diff	✓	✓	✓				
Design Pattern Support	✓	✓	✓				
Model Element Nicknaming	✓	✓	✓				
Model Transitor	✓	✓	✓				
Style and Formatting	✓	✓	✓	✓	✓	✓	
Team Collaboration with VP Teamwork Server	✓	✓	✓	✓			
Team Collaboration with CVS Repository	✓	✓	✓				
Team Collaboration with Subversion Repository	✓	✓	✓				
Team Collaboration with Perforce	✓	✓	✓				
PDF and HTML report generation	✓	✓	✓	✓			
Word report generation	✓	✓	✓				
Project Publisher	✓	✓	✓				
Ad Hoc report creation	✓	✓	✓				
Printing	✓	✓	✓	✓	+	*	
IDE Integrations	✓	✓					

Reverse Engineering	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Code Generation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
State Machine Diagram Code Generation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Java Round-Trip Engineering	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Shape Editor	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Layout Facilities	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Interoperability A	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
• Import Telelogic Modeler project files			
• Import and export use case model to MS Word			
• Import Rational Rose project files			
• Import ERwin data modeler project files			
• Import Rational DNX			
• Import NetBeans 6.x UML diagrams			
• Import Visio drawings into VP-UML			
• Import and export Excel file			
Interoperability B	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
• Import and export EMF based on UML2			
• Import and export XMI 1.0, 1.2 and 2.1			
• Import and export VP project file format			
• Import Rational Software Architect files			
Import and export XML	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Generate BPEL code for Oracle and JBoss workflow engine	<input checked="" type="checkbox"/>		
Command-line operations	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Export diagrams as JPG, PNG, SVG, EMF and PDF	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Intuitive User Interface	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Automatic Updates	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Open Architecture	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

A summary of features supported by VP-UML

 This feature is available but the single watermark will be shown.

 The feature is available but the single watermark will be shown when the project has one diagram per diagram type. And the pattern watermark will be shown when the project has more than one diagram per diagram type.

 The feature is available but unable to make changes.

Features

UML support

Improve modeling efficiency with this easy-to-use, feature-rich and reliable UML 2.1 modeling tool.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
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	✓	✓	✓	✓	✓	✓	#
Use case detail editor	✓	✓	✓	✓	✓	✓	#
Use case flow-of-events listing	✓	✓	✓	✓	✓	✓	#
Generate sequence diagrams from flow of events lists	✓	✓	✓	✓	✓	✓	
Select attribute's getter or setter as call message's action	✓	✓	✓	✓	✓	✓	
Business model use case support	✓	✓	✓	✓	✓	✓	

The feature is available but unable to make changes.

UML Supports in different editions of VP-UML

Requirements management

Capture requirements with SysML Requirement Diagram, Use Case Modeling, Textual Analysis, CRC and more!

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Requirement diagram	✓	✓	✓	✓	✓	✓	#
Textual analysis	✓	✓	✓	✓	✓	✓	#
CRC Cards	✓	✓	✓	✓	✓	✓	#
User interface designer	✓						#
Identify candidate activity and action by textual analysis	✓	✓	✓	✓	✓	✓	
Define and customize requirement types	✓	✓	✓				
Display full set of requirements in tabular view	✓	✓	✓	✓	✓	✓	
Support generating ID for Requirements	✓	✓	✓				

The feature is available but unable to make changes.

Requirements management support in different editions of VP-UML

Business process modeling

Visualize, understand and improve your business process with the most comprehensive BPMN tool.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Business process diagram	✓						#
Data flow diagram	✓						#
Event-driven process chain diagram	✓						#
Process map diagram	✓						#
Export business process diagrams to BPEL	✓						
Identify candidate business process elements using textual analysis	✓						
Automatically stretch pools and lanes to fit diagram	✓						
Smart routing for connecting objects	✓						
Extend business process model with stereotype and tagged value	✓						
Set state for data object	✓						
Organization chart	✓						#
Relocate a branch of unit through drag and drop	✓						
Nested lanes support	✓						#

The feature is available but unable to make changes.

Business Process modeling support in different editions of VP-UML

Animacian

A set of tools to help animating paths in a diagram, or to export animation to flash for further analysis.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Animate business process using Animacian	✓						
Animate sequence diagram using Animacian	✓						
Animate activity diagram using Animacian	✓						
Travel along execution path with slider	✓						
Step forward and backward in an animation	✓						
Auto validate execution path	✓						
Auto calculate all possible execution paths	✓						
Eliminate paths using filter	✓						
Dim non-execution path	✓						
Support Full and Compact Mode	✓						
Protect diagram when animating	✓					#	
Export selected path to flash format	✓						
Animate and navigate animation in exported flash	✓					#	

The feature is available but unable to make changes.

Animacian support in different editions of VP-UML

Mind mapping

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Mind map diagram	✓						
Create link relationships between nodes	✓						
Smart layout for mind mapping nodes and diagram	✓						

Mind Mapping support in different editions of VP-UML

Database modeling

Enhance database documentation quality with sophisticated ERD and Object-Relational Mapping diagrams.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Entity relationship diagram	✓	✓	✓	✓	✓	✓	#
ORM diagrams	✓	✓	✓	✓	✓	✓	#
Reverse engineer existing databases to entity relationship diagrams (ERDs)	✓	✓					
Generate and execute database schema (DDLs)	✓	✓					
Conceptual, logical and physical ERD support	✓	✓	✓	✓	✓	✓	
Reverse engineer stored procedures to ERDs	✓	✓					

Foreign key auto-naming	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Define PK naming pattern	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Define FK relationship naming pattern	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Model primary key in object model by using the <>PK<> stereotype	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Download database drivers automatically	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	#
Display database architecture in object-relational mapping (ORM) pane	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Customizable SQL generation	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	#
Generate class diagrams from ERDs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Generate ERDs from class diagrams	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Select target diagram when first synchronized between class diagram and ERD	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Jump between ORM class and entity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Generate and reverse engineer database support for Oracle schema	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Generate or reverse engineer user-defined database types	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Database trigger and stored procedure modeling	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Reverse engineer DDL models to ERD models	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Database trigger and stored procedure generation and reversal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Unique and Index support for entities	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Convert a normal association to ORM association	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Support AbstractPersistable class for generating non ORM super class attributes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Configurable generated SQL statements in upper or lower case	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Model and generate DB Sequence (Oracle and DB2 only)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

The feature is available but unable to make changes.

Database modeling support in different editions of VP-UML

Object-Relational mapping

Know how to manipulate objects in Java, .NET and PHP? Then you now know how to access relational databases. No more tedious database programming! Just leave it to us.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Java ORM code generation	✓	✓					
.NET ORM code generation	✓						
PHP ORM code generation	✓						
Lazy collection fetching	✓	✓					
Database view support	✓	✓					
Custom ID generator support	✓	✓					
Map single classes to multiple tables	✓	✓					
Custom query support	✓	✓					
Automatic array table generation	✓	✓					
Criteria class generation	✓	✓					
DAO code generation	✓	✓					
Factory code generation	✓	✓					
POJO code generation	✓	✓					
Optional library selection	✓	✓					
Hibernate annotation support in ORM persistence	✓	✓					
Generate Hibernate version tags for optimistic concurrency control	✓	✓					
Formula support for ORM attribute	✓	✓					
User defined exception support in error handling	✓	✓					
Define default type mapping between database column and class attribute	✓	✓					
Configurable naming pattern for synchronization between Class Diagram and ERD	✓	✓					

Object Relational Mapping support in different editions of VP-UML

Visual modeling

Improve the user-friendliness of drawing diagrams, thus speed up the time need to spend on diagramming.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Reference other projects' model elements	✓	✓	✓				
Freehand shape support	✓	✓	✓	✓	✓	✓	✓
Justify shape name	✓	✓	✓	✓	✓	✓	✓
Support enable or disable minimum shape size	✓	✓	✓	✓	✓	✓	✓

Layer support to show/hide set of shape with single click	✓	✓	✓	✓	✓	✓
Support showing line jump in Arc, Square, Skip or normal	✓	✓	✓	✓	✓	✓
Align Connector Caption Base on Connector Orientation	✓	✓	✓	✓	✓	✓
Annotation with callout shapes	✓	✓	✓	✓	✓	✓
Annotation with freehand shapes	✓	✓	✓	✓	✓	✓
Organize model elements and diagrams using Model Explorer	✓	✓	✓	✓	✓	✓
Bookmark support	✓	✓	✓	✓	✓	✓
Handi-Selection tool	✓	✓	✓	✓	✓	✓
Customizable data types for use with multiple programming languages	✓	✓	✓	✓	✓	✓
Overview diagrams	✓	✓	✓	✓	✓	✓
Resource-centric interface	✓	✓	✓	✓	✓	✓
Model sharing	✓	✓	✓	✓	✓	✓
Cut, copy and paste	✓	✓	✓	✓	✓	✓
Copy diagrams as images for use in other applications	✓	✓	✓	✓	✓	✓
Undo and redo options	✓	✓	✓	✓	✓	✓
Mouse gestures	✓	✓	✓	✓	✓	✓
Reverse connector direction	✓	✓	✓	✓	✓	✓
Group creation support	✓	✓	✓	✓	✓	✓
Jump to feature for selecting a particular shape or model	✓	✓	✓	✓	✓	✓
Quick connect feature	✓	✓	✓	✓	✓	✓
Easy navigation to connected elements	✓	✓	✓	✓	✓	✓
Model commenting	✓	✓	✓	✓	✓	✓
Duplicate shapes and models	✓	✓	✓	✓	✓	✓
Selectable/non-selectable toggling for shapes	✓	✓	✓	✓	✓	✓
Diagram locking	✓	✓	✓	✓	✓	✓
Reference to any type of artifact	✓	✓	✓	✓	✓	✓
Advanced file and directory selector	✓	✓	✓	✓	✓	✓
Advanced tree support	✓	✓	✓	✓	✓	✓
Duplicate, move and reconnect connectors	✓	✓	✓	✓	✓	✓
Package headers for all types of diagrams	✓	✓	✓	✓	✓	✓

Tagged value display toggling for diagram elements	<input checked="" type="checkbox"/>	#					
Sub-diagrams and reference indicators	<input checked="" type="checkbox"/>						
Rectilinear, round rectilinear, oblique, round oblique and curve connector styles	<input checked="" type="checkbox"/>						
Space reclamation using Sweeper	<input checked="" type="checkbox"/>						
Space elimination using Magnet	<input checked="" type="checkbox"/>						
Create shapes with user-defined initial sizes	<input checked="" type="checkbox"/>						
Drag-and-drop creation of shapes using trees in diagrams	<input checked="" type="checkbox"/>						
Auto-fit shape sizes	<input checked="" type="checkbox"/>						
In-line editing	<input checked="" type="checkbox"/>						
Spell checking	<input checked="" type="checkbox"/>						
Visual alignment guides	<input checked="" type="checkbox"/>						
Numerous grid options	<input checked="" type="checkbox"/>	#					
Diagram information display in diagrams	<input checked="" type="checkbox"/>	#					
Jump to diagram feature	<input checked="" type="checkbox"/>						
Drag-and-drop copying, moving and reordering of classes and entity members	<input checked="" type="checkbox"/>						
Open view from model element	<input checked="" type="checkbox"/>						
Diagram renaming boxes	<input checked="" type="checkbox"/>						
Add folders as favorites	<input checked="" type="checkbox"/>						
Display Undo/Redo action names	<input checked="" type="checkbox"/>						
Inverse shape selection	<input checked="" type="checkbox"/>						
Create new attribute with Enter key	<input checked="" type="checkbox"/>						
Automatic reroute connector when overlapped with other shapes	<input checked="" type="checkbox"/>						
Hide shapes or type of shapes on a diagram	<input checked="" type="checkbox"/>						
Visualize related model element	<input checked="" type="checkbox"/>						
Enforcing master view between model element and shape	<input checked="" type="checkbox"/>						
Auto extend activation	<input checked="" type="checkbox"/>						
Model name completion for class	<input checked="" type="checkbox"/>						
Pinnable connector end	<input checked="" type="checkbox"/>						

The feature is available but unable to make changes.

Visual Modeling support in different editions of VP-UML

Visual diff

A set of tools that help to compare diagrams, and identify their differences.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Compare and show difference between diagrams visually	✓	✓	✓				
Support compare by name, id and transition options	✓	✓	✓				
Compare view, model element or both kinds of properties	✓	✓	✓				
Filter result by displaying all, addition, modification or deletion of items	✓	✓	✓				

Support of Visual Diff in different editions of VP-UML

Design pattern support

Define a pattern and save it for future use, or share with team members for making the design consistent.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Design pattern support for class diagrams	✓	✓	✓				
Full Gang of Four (GOF) design patterns support	✓	✓	✓				
Define pattern with a configurable number of subclasses	✓	✓	✓				
Define pattern with a configurable number of attributes and operations	✓	✓	✓				
Define pattern with implementation hierarchy	✓	✓	✓				
Define pattern with automatically named association end	✓	✓	✓				
Automatically rename elements in pattern	✓	✓	✓				
Design pattern support for sequence diagrams	✓	✓	✓				
Design pattern support for all UML diagram types	✓	✓	✓				
Design pattern support for entity relationship diagrams	✓	✓	✓				
Design pattern support for business process diagrams	✓						
Sharing Design Patterns through team collaboration support	✓	✓	✓				

Design Pattern support in different editions of VP-UML

Model element nicknaming

Define nickname for a set of models.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Assign nicknames to model elements	✓	✓	✓				

Model element nicknaming in different editions of VP-UML

Model transitor

Setup and maintain transition between models through the Model Transitor

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Generate and link model elements	✓	✓	✓				
Trace the origin of model elements (model traceability)	✓	✓	✓				
Trace the origin of model elements (model traceability)	✓	✓	✓				
Navigate between operation and sequence diagram	✓	✓	✓				
Transit Entity relationship diagram from Conceptual to Logical to Physical	✓	✓	✓				

Model Transitor support in different editions of VP-UML

Style and formatting

Define and apply style and formatting to shapes.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Manage and apply styles within project	✓	✓	✓	✓	✓	✓	✓
Customizable shape style and formatting	✓	✓	✓	✓	✓	✓	✓
Image incorporation in diagrams	✓	✓	✓	✓	✓	✓	#
Stereotyped element appearance	✓	✓	✓	✓	✓	✓	#
Rich text documentation	✓	✓	✓	✓	✓	✓	#
Add rich text elements to diagrams	✓	✓	✓	✓	✓	✓	
Shape format copier	✓	✓	✓	✓	✓	✓	
Enrich model documentation with images	✓	✓	✓	✓	✓	✓	#
Save/load template for model documentation	✓	✓	✓	✓	✓	✓	
Display stereotyped model element as image icon	✓	✓	✓	✓	✓	✓	#

The feature is available but unable to make changes.

Style and Formatting support in different editions of VP-UML

Team collaboration with VP teamwork server

Perform modeling collaboratively and concurrently with the VP Teamwork Server.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Concurrent and collaborative modeling with VP Teamwork Server	✓	✓	✓	✓			
Import projects to VP Teamwork Server	✓	✓	✓	✓			
View projects from VP Teamwork Server	✓	✓	✓	✓		#	
Commit project changes to VP Teamwork Server	✓	✓	✓	✓			
Update local project copy using VP Teamwork Server	✓	✓	✓	✓			
Review past revisions using VP Teamwork Server	✓	✓	✓	✓		#	
Compare past revisions using VP Teamwork Server	✓	✓	✓	✓		#	
Undo committed changes by reverting revisions	✓	✓	✓	✓			
Detect and resolve conflicts using VP Teamwork Server	✓	✓	✓	✓			
Branch and tag projects using VP Teamwork Server	✓	✓	✓	✓			
Merge branch changes using VP Teamwork Server	✓	✓	✓	✓			
Export multiple revisions from VP Teamwork Server	✓	✓	✓	✓			
Run VP Teamwork Server on common Java web servers	✓	✓	✓	✓			
Element based revision history	✓	✓	✓	✓			
Preview local and server changes before commit or update	✓	✓	✓	✓			

The feature is available but unable to make changes.

Support of Teamwork Collaboration with Teamwork Server in different editions of VP-UML

Team collaboration with CVS repository

Perform modeling collaboratively and concurrently with CVS.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Concurrent and collaborative modeling with CVS Repository	✓	✓	✓	✓			
Import projects to CVS Repository	✓	✓	✓	✓			
View projects from CVS Repository	✓	✓	✓	✓		#	
Commit project changes to CVS Repository	✓	✓	✓	✓			
Update local project copy using CVS Repository	✓	✓	✓	✓			
Review past revisions using CVS Repository	✓	✓	✓	✓		#	
Compare past revisions using CVS Repository	✓	✓	✓	✓		#	
Undo committed changes by reverting revisions	✓	✓	✓	✓			
Detect and resolve conflicts using CVS Repository	✓	✓	✓	✓			
Branch and tag projects using CVS Repository	✓	✓	✓	✓			
Merge branch changes using CVS Repository	✓	✓	✓	✓			
Export multiple revisions from CVS Repository	✓	✓	✓	✓			
Element based revision history	✓	✓	✓	✓			
Preview local and server changes before commit or update	✓	✓	✓	✓			

The feature is available but unable to make changes.

Support of Teamwork Collaboration with CVS Repository in different editions of VP-UML

Team collaboration with Subversion repository

Perform modeling collaboratively and concurrently with Subversion.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Concurrent and collaborative modeling with Subversion Repository	✓	✓	✓	✓			
Import projects to Subversion Repository	✓	✓	✓	✓			
View projects from Subversion Repository	✓	✓	✓	✓		#	
Commit project changes to Subversion Repository	✓	✓	✓	✓			
Update local project copy using Subversion Repository	✓	✓	✓	✓			
Review past revisions using Subversion Repository	✓	✓	✓	✓		#	
Compare past revisions using Subversion Repository	✓	✓	✓	✓		#	
Undo committed changes by reverting revisions	✓	✓	✓	✓			
Detect and resolve conflicts using Subversion Repository	✓	✓	✓	✓			
Branch and tag projects using Subversion Repository	✓	✓	✓	✓			
Merge branch changes using Subversion Repository	✓	✓	✓	✓			
Export multiple revisions from Subversion Repository	✓	✓	✓	✓			
Element based revision history	✓	✓	✓	✓			
Preview local and server changes before commit or update	✓	✓	✓	✓			

The feature is available but unable to make changes.

Support of Teamwork Collaboration with Subversion Repository in different editions of VP-UML

Team collaboration with Perforce

Perform modeling collaboratively and concurrently with the Perforce.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Concurrent and collaborative modeling with Perforce Repository	✓	✓	✓	✓			
Import projects to Perforce Repository	✓	✓	✓	✓			
View projects from Perforce Repository	✓	✓	✓	✓		#	
Commit project changes to Perforce Repository	✓	✓	✓	✓			
Update local project copy using Perforce Repository	✓	✓	✓	✓			
Review past revisions using Perforce Repository	✓	✓	✓	✓		#	
Compare past revisions using Perforce Repository	✓	✓	✓	✓		#	
Undo committed changes by reverting revisions	✓	✓	✓	✓			
Detect and resolve conflicts using Perforce Repository	✓	✓	✓	✓			
Branch and tag projects using Perforce Repository	✓	✓	✓	✓			
Merge branch changes using Perforce Repository	✓	✓	✓	✓			
Export multiple revisions from Perforce Repository	✓	✓	✓	✓			
Element based revision history	✓	✓	✓	✓			
Preview local and server changes before commit or update	✓	✓	✓	✓			

The feature is available but unable to make changes.

Support of Teamwork Collaboration with Perforce in different editions of VP-UML

Documentation generation

Share your design with your customers in popular document formats (PDF, HTML and MS Word).

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
PDF report generation	✓	✓	✓	✓			
MS Word report generation [1]	✓	✓	✓				
HTML report generation	✓	✓	✓	✓			
Project publisher	✓	✓	✓				
Ad Hoc report creation [1]	✓	✓	✓				
Intelligent element sorting during report generation	✓	✓	✓	✓			

[1] This feature is only available for the Windows platform.

Documentation generation support in different editions of VP-UML

Printing

Print out diagrams using the Print features.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Print multiple diagrams	✓	✓	✓	✓	+	*	
Preview printable pages	✓	✓	✓	✓	+	*	
Print clip marks	✓	✓	✓	✓	+	*	
Page margin, size and orientation alteration support	✓	✓	✓	✓	+	*	
Fit-to-pages option	✓	✓	✓	✓	+	*	
Fit-to-ratio option	✓	✓	✓	✓	+	*	
Customizable page header and footer	✓	✓	✓	✓	+	*	
Project name and diagram name display in header or footer	✓	✓	✓	✓	+	*	
Print with frame or border support	✓	✓	✓	✓	+	*	
Toggle gradient color printing	✓	✓	✓	✓	+	*	
Quick print support	✓	✓	✓	✓	+	*	

+ This feature is available but the single watermark will be shown.

***** The feature is available but the single watermark will be shown when the project has one diagram per diagram type. And the pattern watermark will be shown when the project has more than one diagram per diagram type.

Printing support in different editions of VP-UML

IDE integrations

Support full software development life-cycle, from analysis to design, and from design to implementation with your most favorite IDE.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Activate full UML environment from your favorite IDE	✓	✓					
Automatic code and model synchronization	✓	✓					
Simple integration of any IDE	✓	✓					
Import existing VP-UML project to IDE	✓	✓					
Integration with Eclipse	✓	✓					
Integration with NetBeans	✓	✓					
Integration with IntelliJ IDEA	✓	✓					
Multilingual support in IDE integration	✓	✓					

IDE integration support in different editions of VP-UML

Reverse engineering

Reverse engineer UML models from source code.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Reverse engineer source code or executables to class diagrams using Instant Reverse feature	✓	✓	✓				
Reverse engineer Java source code, classes and .jar files	✓	✓	✓				
Reverse engineer C++ source code	✓	✓	✓				
Reverse engineer .NET .dll and .exe files	✓	✓	✓				
Reverse engineer CORBA IDL source code	✓	✓	✓				
Reverse engineer Ada 9x source code	✓	✓	✓				
Reverse engineer XML	✓	✓	✓				
Reverse engineer XML Schema	✓	✓	✓				
Reverse engineer databases with JDBC	✓	✓	✓				
Reverse engineer Hibernate mapping files	✓	✓	✓				
Reverse engineer PHP 5.0 source code	✓	✓	✓				
Reverse engineer Python	✓	✓	✓				
On-demand Java reverse engineering	✓	✓	✓				
Template parameter support	✓	✓	✓				

Reverse Engineering support in different editions of VP-UML

Code generation

Generate various kind of source code from class models.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Instantly generate source code from class diagrams	✓	✓	✓				
Generate Java source code	✓	✓	✓				
Generate C# source code	✓	✓	✓				
Generate VB.NET source code	✓	✓	✓				
Generate PHP 5.0 source code	✓	✓	✓				
Generate Object Definition Language source code	✓	✓	✓				
Generate Flash ActionScript 3.0 source code	✓	✓	✓				
Generate IDL source code	✓	✓	✓				
Generate C++ source code	✓	✓	✓				
Generate Delphi source code	✓	✓	✓				
Generate Perl source code	✓	✓	✓				
Generate XML Schema source code	✓	✓	✓				
Generate Python source code	✓	✓	✓				
Generate Objective-C source code	✓	✓	✓				
Generate Ada source code	✓	✓	✓				
Generate Ruby source code	✓	✓	✓				
Template parameter support	✓	✓	✓				
Template support for generating source code with generic constructs	✓	✓	✓				
Customizable source code generation	✓	✓	✓				

Code generation support in different editions of VP-UML

State machine diagram code generation

Generate code from a state machine diagram for state machine.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Generate Java source code from state diagram	✓	✓					
Generate C++ source code from state diagram	✓	✓					
Generate C# source code from state diagram	✓	✓					
Generate VB.NET source code from state diagram	✓	✓					

State Machine Diagram code generation support in different editions of VP-UML

Java round-trip engineering

Update between UML class models and Java source code with the round-trip engineering functionalities.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Reverse engineer Java source code to class diagram	✓	✓					
Update Java source code based on class diagram	✓	✓					

Java Round-Trip support in different editions of VP-UML

Shape editor

Create domain specific shapes with Shape Editor to use in diagrams.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Design arbitrary shapes	✓	✓	✓	✓			
Import SVG shapes	✓	✓	✓	✓			
Incorporate different shapes into UML diagram	✓	✓	✓	✓			
Organize shapes by gallery, category and stencil	✓	✓	✓	✓			
Advanced shape design capabilities	✓	✓	✓	✓			

Shape Editor support in different editions of VP-UML

Layout facilities

Tidy up diagram content by performing layout with several mouse actions.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Automatic diagram layouts	✓	✓	✓	✓	✓	✓	✓
Shape alignment and centering	✓	✓	✓	✓	✓	✓	✓
Uniform shape width and height maintenance	✓	✓	✓	✓	✓	✓	✓
Automatic shape distribution	✓	✓	✓	✓	✓	✓	✓

Layout support in different editions of VP-UML

Interoperability

Exchange UML diagrams and models with other tools with industrial standard representation.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Import Telelogic Modeler project files	✓	✓	✓				
Import and export EMF based UML2 model	✓	✓	✓	✓			
Command-line operations	✓	✓	✓	✓			
Import and export XMI 1.0, 1.2 and 2.1	✓	✓	✓	✓			
Import and export XML	✓	✓	✓	✓	✓	✓	
Import and export VP project file format	✓	✓	✓	✓			
Import and export use case model to MS Word	✓	✓	✓				
Import Rational Rose project files	✓	✓	✓				

Import ERwin data modeler project files [2]	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Generate BPEL code for Oracle workflow engine	<input checked="" type="checkbox"/>					
Generate BPEL code for JBoss workflow engine	<input checked="" type="checkbox"/>					
Export diagrams as JPG, PNG, SVG and EMF image files	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> +	<input type="checkbox"/> *
Export diagrams as PDFs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> +	<input type="checkbox"/> +
Slice exported diagrams into smaller segments	<input checked="" type="checkbox"/>					
Import Rational Software Architect files	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Import Rational DNX	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import NetBeans 6.x UML diagrams	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Copy diagram elements as XML	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Import Visio drawings into Visual Paradigm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for use case diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for class diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for sequence diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for communication diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for state machine diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for activity diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for component diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for deployment diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for package diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for object diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for composite structure diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for interaction overview diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for requirement diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for basic diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for crc card diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for entity relationship diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for orm diagram	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Import and export Microsoft Excel file for business process diagram	<input checked="" type="checkbox"/>					
Import and export Microsoft Excel file for data flow diagram	<input checked="" type="checkbox"/>					

Import and export Microsoft Excel file for epc diagram	<input checked="" type="checkbox"/>
Import and export Microsoft Excel file for process map diagram	<input checked="" type="checkbox"/>
Import and export Microsoft Excel file for organization chart	<input checked="" type="checkbox"/>
Import and export Microsoft Excel file for mind mapping diagram	<input checked="" type="checkbox"/>

[2] This feature requires ERwin project on version 7.0 or above.

+ This feature is available but the single watermark will be shown.

***** The feature is available but the single watermark will be shown when the project has one diagram per diagram type. And the pattern watermark will be shown when the project has more than one diagram per diagram type.

Interoperability support in different editions of VP-UML

Intuitive user interface

User friendly user interface to let you achieve tasks effortlessly.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Group diagrams by category	<input checked="" type="checkbox"/>						
Advanced property pane	<input checked="" type="checkbox"/>	#					
Dockable user interface windows	<input checked="" type="checkbox"/>						
New project generation using predefined templates	<input checked="" type="checkbox"/>						
Easy-to-use "New Diagram" dialog	<input checked="" type="checkbox"/>						
Flexible zooming	<input checked="" type="checkbox"/>						
Palette-style toolbar	<input checked="" type="checkbox"/>						
Collapsible toolbar	<input checked="" type="checkbox"/>						
Display tool names for toolbar buttons	<input checked="" type="checkbox"/>						
Expand grouped toolbar buttons	<input checked="" type="checkbox"/>						
Display various diagram categories in toolbar	<input checked="" type="checkbox"/>						
Display stereotyped model types in toolbar	<input checked="" type="checkbox"/>						
Numerous looks-and-feels	<input checked="" type="checkbox"/>						
Import user preferences from existing workspaces	<input checked="" type="checkbox"/>						
Multilingual support	<input checked="" type="checkbox"/>						
Searchable options	<input checked="" type="checkbox"/>						
Customizable program shortcuts	<input checked="" type="checkbox"/>						

The feature is available but unable to make changes.

Support of 'intuitive user interface' in different editions of VP-UML

Automatic updates

Keep VP-UML up-to-date with the automatic updates features.

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Automatic online updating	✓	✓	✓	✓	✓	✓	✓
Maintain local update server with update synchronizer	✓	✓	✓	✓	✓	✓	✓

Support of 'automatic updates' in different editions of VP-UML

Open architecture

Extend VP-UML's functionalities using plug-in

	Enterprise	Professional	Standard	Modeler	Personal	Community	Viewer
Plug-in support (Java)	✓	✓	✓	✓	✓	✓	✓
VP model and XML interaction	✓	✓	✓	✓	✓	✓	✓

Support of 'open architect' in different editions of VP-UML

Licensing

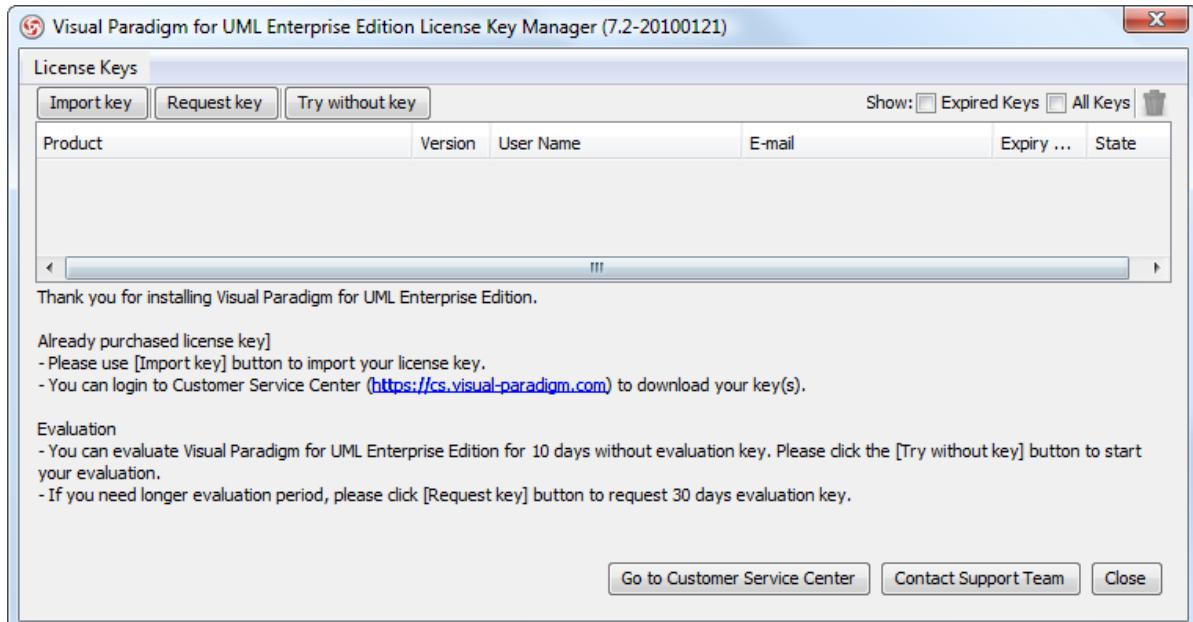
VP-UML need to run with a valid license key. A license key of a higher edition can be used on a lower edition. For example, you can run Standard Edition of VP-UML with Professional Edition of VP-UML key.

VP-UML also provides a 30-days evaluation key for trial. After the evaluation key expires, you can purchase the full license from our website or resellers, or un-install the program.

The license key for Community Edition will not expire. However, Community Edition cannot be used for commercial purposes.

Importing single-seat license key (For purchased customers)

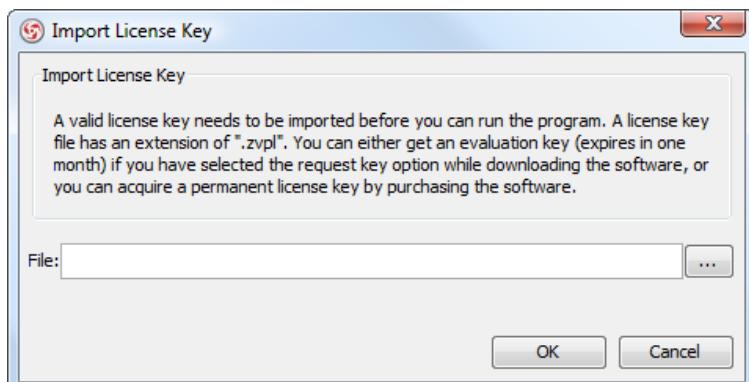
A Visual Paradigm's Single Seat (developer based) license allows a licensee to install the software on more than one machine, such as desktop and notebook, which belong to the licensee only. As the license is developer based, the software must be used by the licensee only, without running more than one instance concurrently. When you run VP-UML the first time, the **License Key Manager** pops up to let you import the license key.



License Key Manager

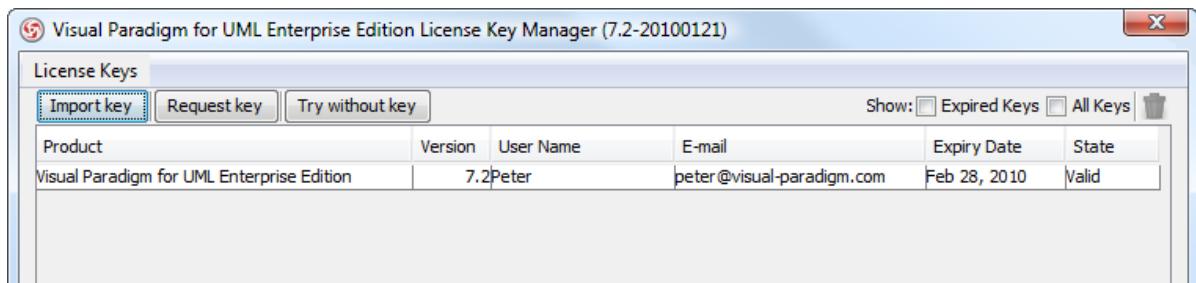
NOTE: To start License Key Manager in VP-UML, select **Tools > Key Manager...** from the main menu.

To import a single-seat key, select **License Keys > Import...** from the menu. In the **Import License Key** dialog box, specify the filepath of the key and press **OK** to confirm.



Entering the filepath of license key

If the imported key is valid, information of key will appear in the **License Key Manager**.



License key is imported

Importing evaluation key (For evaluation users)

You can evaluate VP-UML for 10 days without importing any keys. To do this, click **Close** to close the key manager and continue running VP-UML. After 10 days gone, the key manager will popup again when you start VP-UML, requesting for a key. At that moment, you must supply a valid key in order to continue.

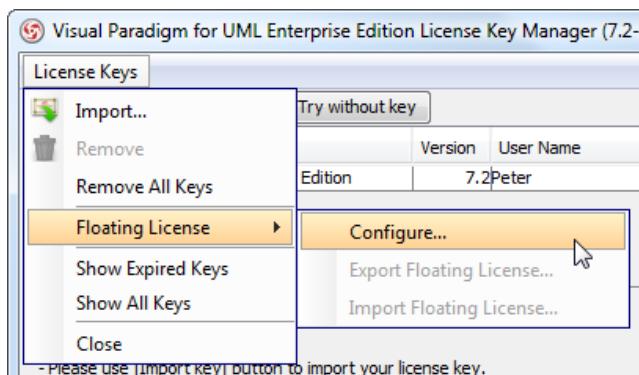
If you want to evaluate for a longer period of time, click **Request key** to request for an evaluation key that enables you to run VP-UML for 30-days. Click **Import key** to import the evaluation key into the key manager and continue.

Working with floating license server

Before you use VP-UML with a floating license key, your machine need to access to the license server via LAN to acquire a license key first. For more details, you can refer to the Floating License Server Installation Guide about floating license server installation for more details.

Configuring floating license server

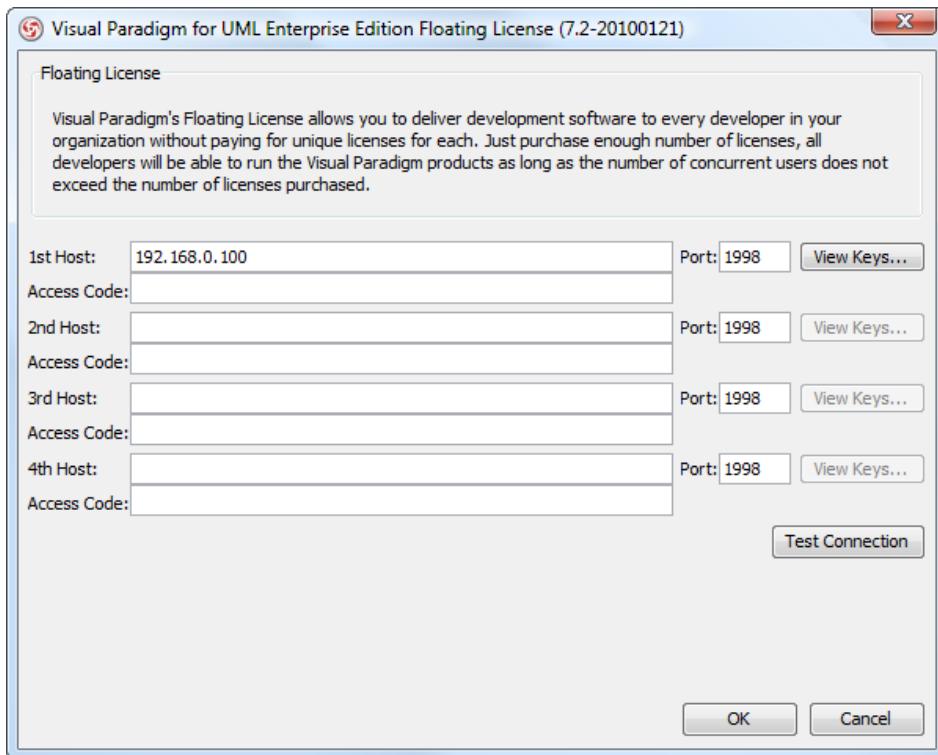
Floating license need to be acquired from server, through the **License Key Manager**. When you start VP-UML the first time, the **License Key Manager** will show up automatically. From the **License Key Manager**, select **License Keys > Floating License > Configure...** in the menu.



To confirm floating license

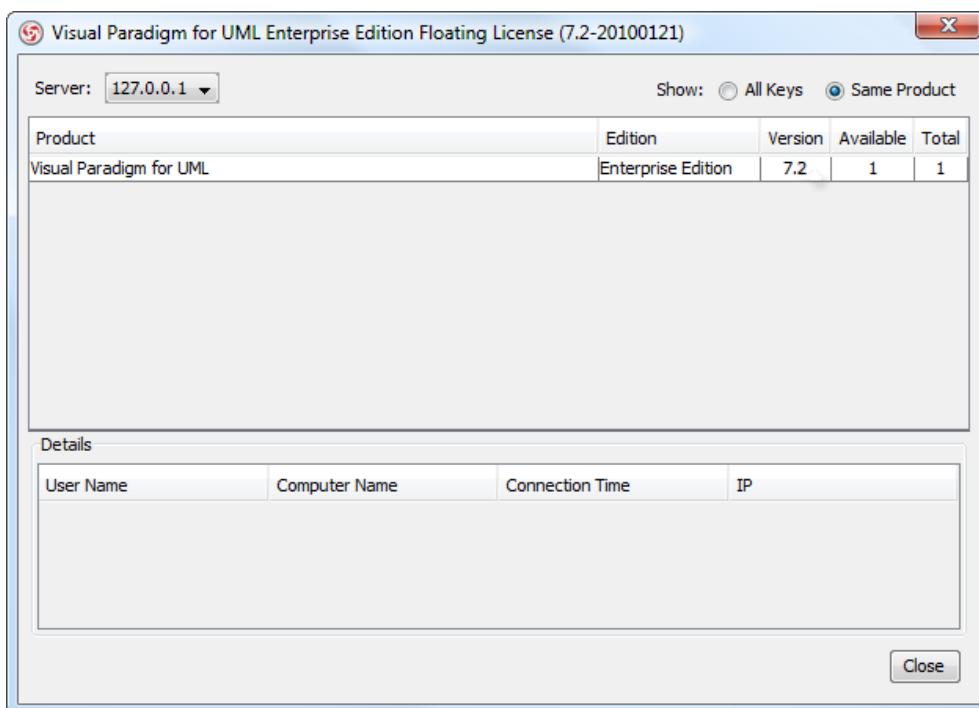
NOTE: To start License Key Manager in VP-UML, select **Tools > Key Manager...** from the main menu.

In the configuration page, specify the host IP and adjust the port (if necessary). You may need to specify the access code if it has been defined in floating license server. You should contact your administrator about this matter.



Specifying host of license server

In order to verify the connection, press **Test Connection**. Alternatively, you may click on the **View Keys** button to see how many keys are available in server.



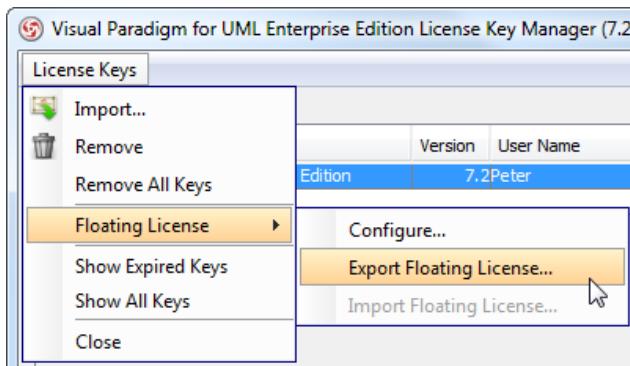
Viewing license in license server

Once the configuration is done correctly, license key will be acquired from license server. By closing the **License Key Manager**, VP-UML will start.

Exporting floating license key from server

In order to acquire a floating license key, you must have access to the floating license server machine through network. If you need to use VP-UML offline, which means, at a location that cannot reach the server machine (e.g. when having a meeting outside the office), you can export a floating license keys to your laptop for running VP-UML with a local license key. Once a key has been exported from server, the number of keys in the server will decrease by one. The volume will reset once you have imported the licence key back to the server.

To export floating license key, start the **License Key Manager** by selecting **Tools > Key Manager...** in the main menu of VP-UML, and select **License Keys > Floating License > Export Floating License...** in the menu of **License Key Manager**.



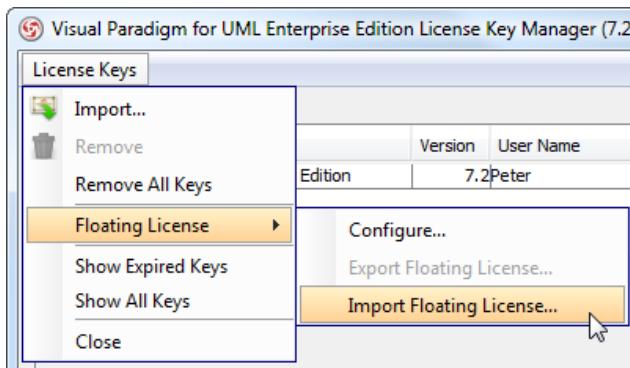
To export floating license key

A message appears informing you about the export of license key.

From the moment the key is exported from server, you do not need to connect to the license server anymore until the moment you import the key back to the server.

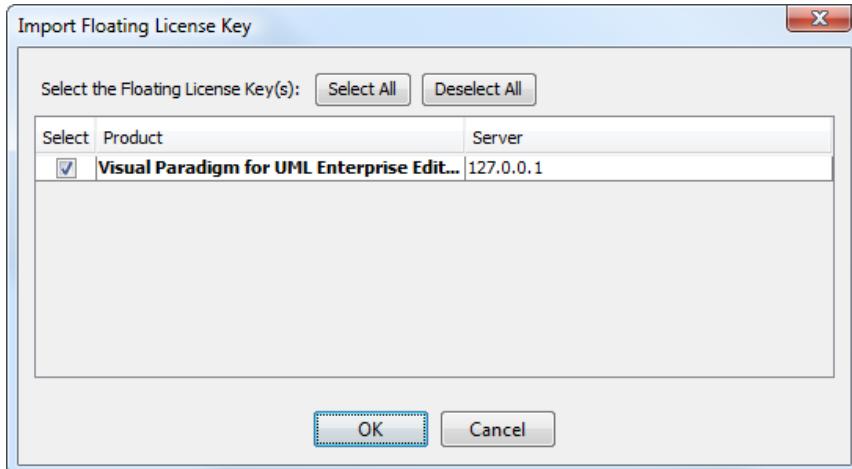
Importing floating license key to server

To import an exported floating license key back to license server, start the License Key Manager by selecting **Tools > Key Manager...** in the main menu of VP-UML, and select **License Keys > Floating License > Import Floating License...** in the menu of **License Key Manager**.



To import floating license key

In the **Import Floating License Key** dialog box, select the key you need to import and click **OK** to confirm.



Select the license key to import

This finished importing the key back to server.

Switching from evaluation license key to purchased license key

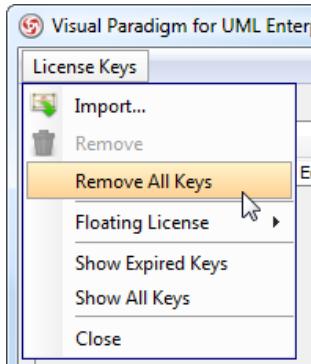
An evaluation key is required for evaluating VP-UML. But once you have purchased, you will receive a purchased license key from Visual Paradigm. In order to make the purchased key take effect, you need to remove the evaluation keys, then import the purchased license key. To switching from an evaluation license key to a purchased license key:

1. Start VP-UML.

- Start the License Key Manager by selecting **Tools > Key Manager...** from the main menu.

NOTE: If your evaluation has expired, you can skip this step as the **License Key Manager** was shown when starting VP-UML (step 1) due to no valid license key was found.

- Remove all license keys by selecting **License Keys > Remove All Keys** from the menu of **License Key Manager**.



To remove all keys in License Key Manager

This cause all the keys to be removed from the **License Key Manager**. Therefore, you should not see any entry in the manager.

- You can import the purchased license key now. For single-seat key user, select **License Keys > Import...**, then specify the filepath of the purchased license key. For floating license user, configure the license server connection by selecting **License Keys > Floating License > Configure...** from the menu. For details, refer to the previous section.

Various licensing options

Single seat license

Visual Paradigm's single-seat (team-member-based) license allows a 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.

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.

In order to work with the floating license, installation of a floating license 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>.

Academic license

Academic licenses are available for higher education, with the aim of providing free site licenses for the teaching of software engineering. Educational institutions that join the Academic Partners Program are entitled to free licenses for the Standard Edition of Visual Paradigm's software, 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/>.

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

From time to time, Visual Paradigm releases new versions of its software. Normally there are two or three versions released every year, with each new version having around five to ten distinct new features as well as a number of improvements. The easiest way to get the latest version from Visual Paradigm is to purchase the software maintenance package. Both minor upgrade (e.g., 6.1 to 6.1 SP1) and major upgrade (e.g., 7.0 to 7.1 or 8.0) of Visual Paradigm products are included in the software maintenance package, thus entitling you to get all of the version upgrades issued within the 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/technicalsupport.jsp>

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 delivering 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.

LIVE help

Our Technical Support Team uses a real-time, web-based chat program to discuss the problem with you directly.

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. To make a call, visit: <http://www.visual-paradigm.com/support/callme.jsp>

Price

Software maintenance is purchased in year-long terms (e.g., June 20, 2007 to June 19, 2008).

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	\$279.5	\$419.5
Professional	\$139.5	\$209.5
Standard	\$59.5	\$89.5
Modeler	\$19.5	\$29.5
Personal	\$11.5	\$17.5
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	\$363.5	\$545.5
Professional	\$181.5	\$272.5
Standard	\$77.5	\$116.5
Modeler	\$25.5	\$38.5
Personal	not applicable	not applicable
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

System requirements

Hardware requirements

- Intel Pentium III Compatible Processor at 1.0 GHz or higher.
- Minimum 512MB RAM, but 1.0 GB is recommended.
- Minimum 800MB disk space.
- Microsoft Windows (98/2000/XP/2003/Vista/7), Linux, Mac OS X, Solaris or all other Java-enabled platforms.

IDE requirements (for IDE integration)

- Eclipse 3.1 or above
- IntelliJ IDEA 4 or above (8.1 ready)
- NetBeans 4.0 or above (6.8 ready)

Installing Visual Paradigm for UML on Windows 2000/NT/2003/XP/Vista

Having downloaded the installer of Visual Paradigm Suite (VP Suite), you can execute it, run through the installation to install the suite as well as VP-UML. If you are using the no-install zip version, you just need to unzip it, launch the suite and install VP-UML. In this chapter, we will go through the installation of VP-UML both with installer (.exe) and no-install (.zip).

Using installer (.exe)

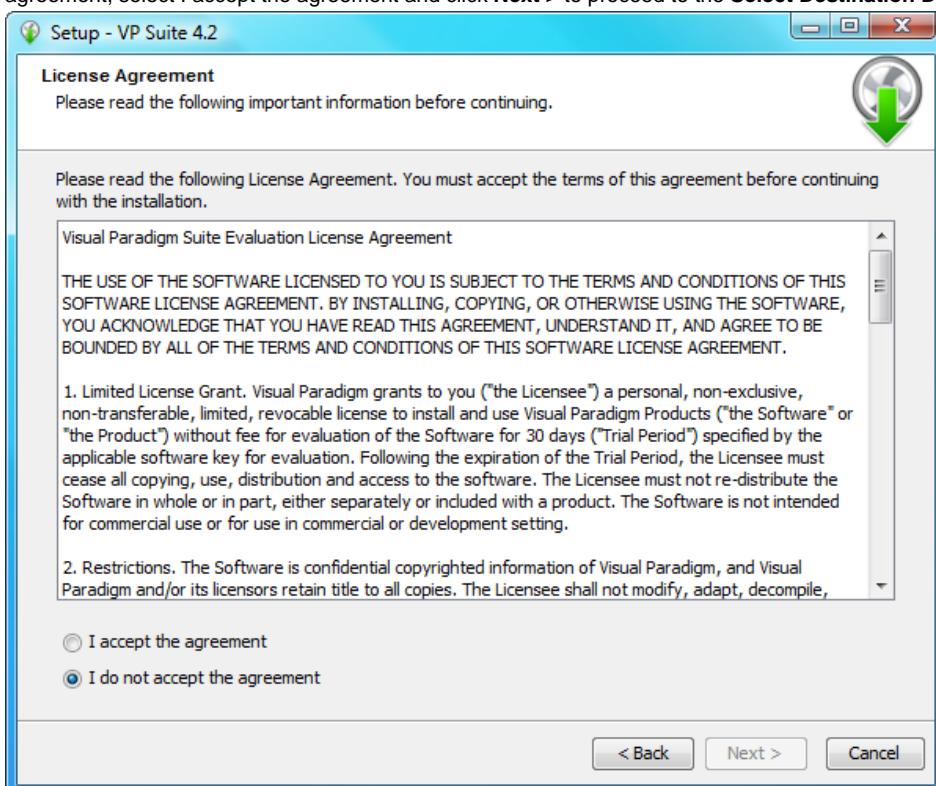
1. Execute the downloaded VP Suite installer file. The setup wizard appear as below.



VP Suite Welcome screen

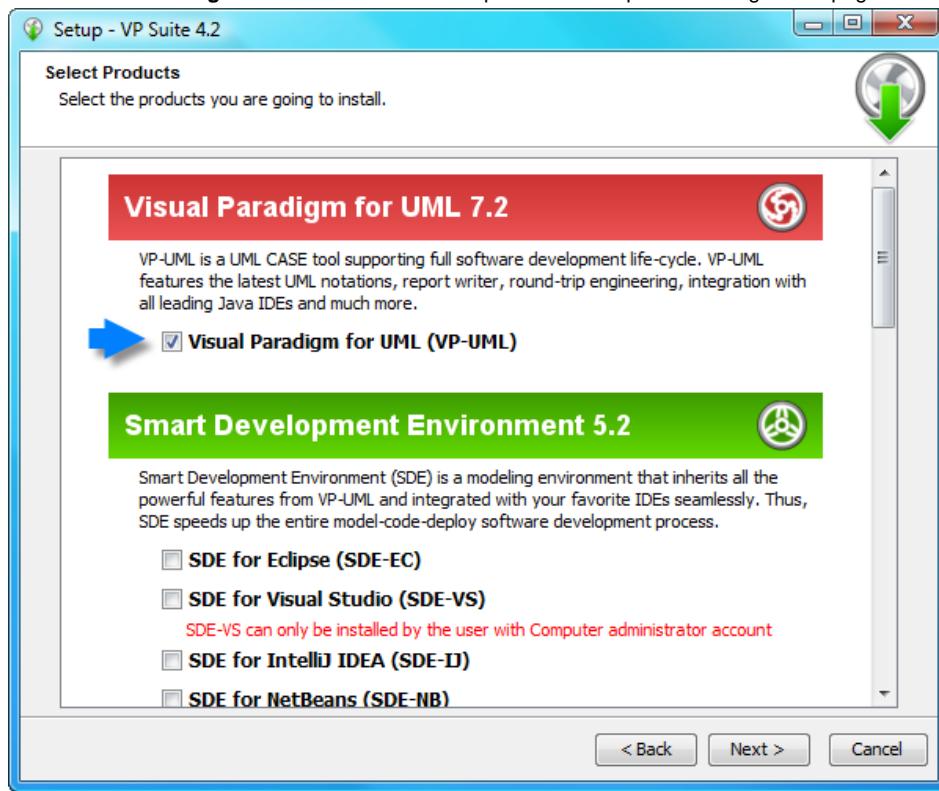
2. Click **Next** to proceed to the License agreement page.

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.



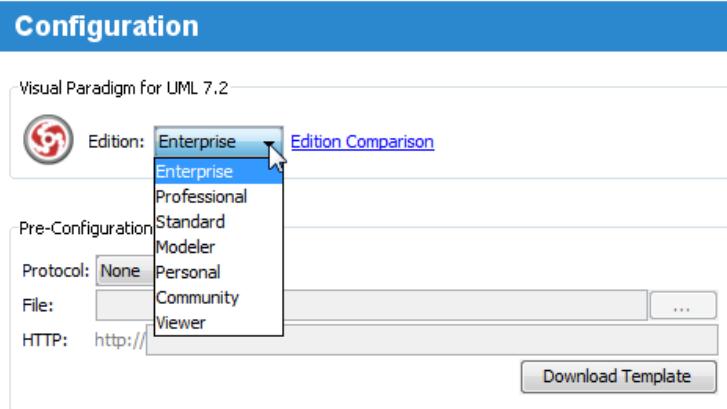
The License Agreement

4. Specify the directory for installing VP Suite. 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)** 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 proceed to the product selection page.
7. Select **Visual Paradigm for UML**. Click **Next >** to proceed to the product configuration page.



The product selection page

8. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.



Select product edition

9. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

Selecting **Enable Product Selector** for Product Selection will result in creating a shortcut under the Start Menu for starting the Product Selector, a utility that lets you realize the installed products with available keys in the floating license server.

Pre-Configuration File [?]

Protocol: **None** ▾

File: ...

HTTP: http://

Product Selector (for floating license)

Enable Product Selector

Pre-configure floating license server connection and enable product selector for floating license user

10. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

Download Documentation

Download Document from Internet:

Download Online Help 

Download PDF Version 

Download HTML Version 

Proxy Setting

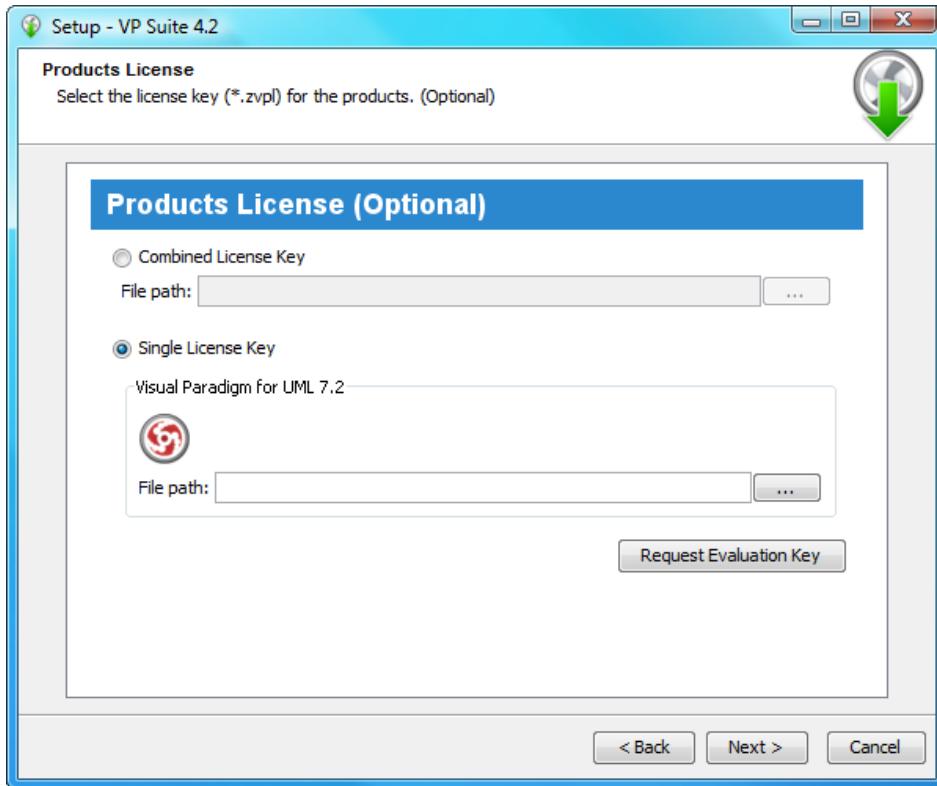
Enable Proxy

Address: Port:

User Name: Password:

Download documentation

11. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.
A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.
The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on **Request Evaluation Key** to ask for another one. This time, try not to select sending the key as attachment.

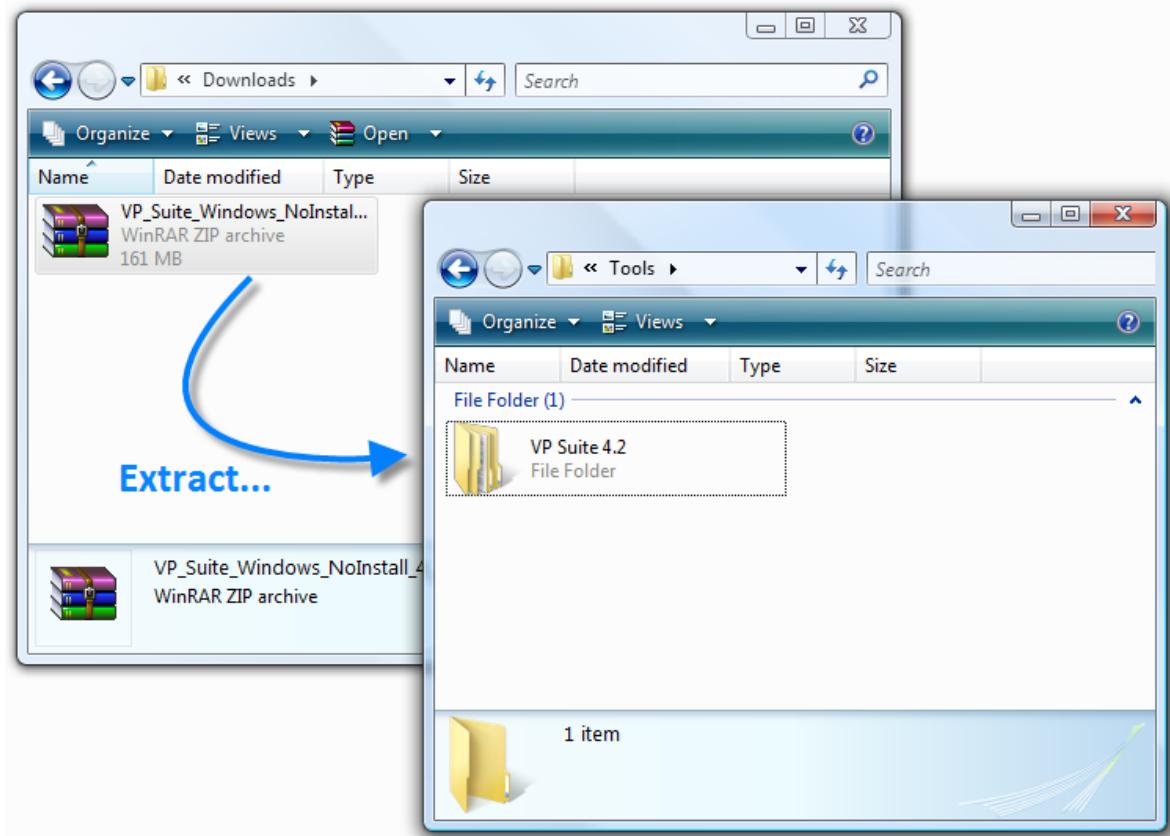
12. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



Installation completed screen

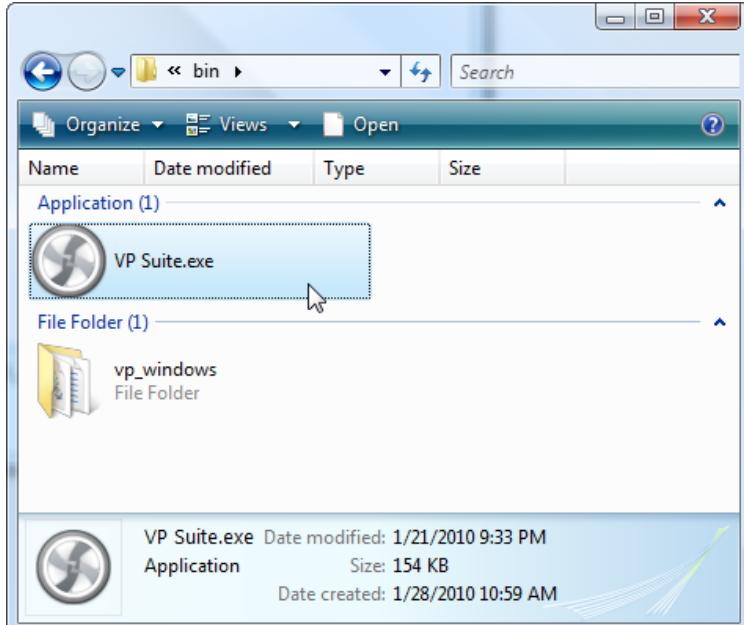
Using no install version (.zip)

1. Decompress the downloaded zip file into a directory. This should create a subdirectory named "VP Suite 4.0" where 4.0 is the version number.



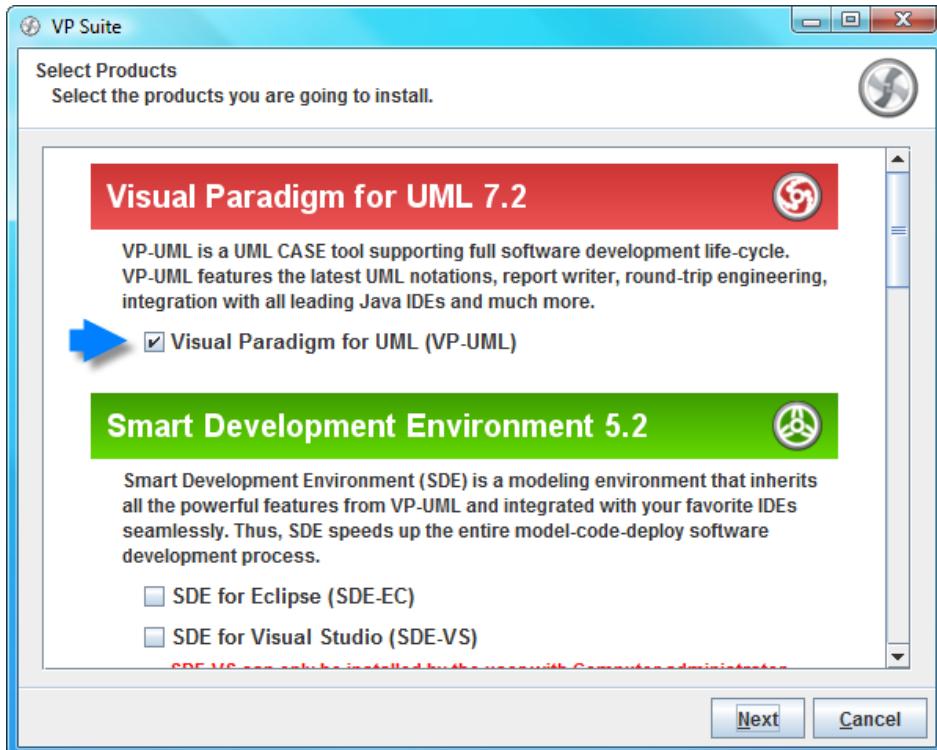
Extracting No-Install zip file

2. Change directory to "VP Suite 4.0\bin" and execute VP Suite in it.



Launching VP Suite

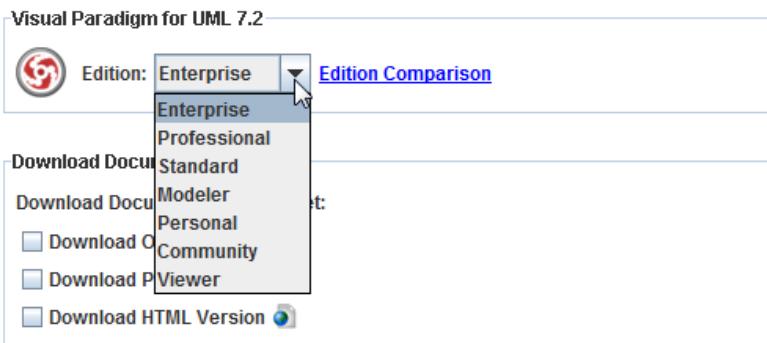
3. Select Visual Paradigm for UML. Click **Next >** to proceed to the product configuration page.



The product selection page

4. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.

Configuration



Select product edition

5. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

6. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

Download Documentation

Download Document from Internet:

- Download Online Help
- Download PDF Version
- Download HTML Version

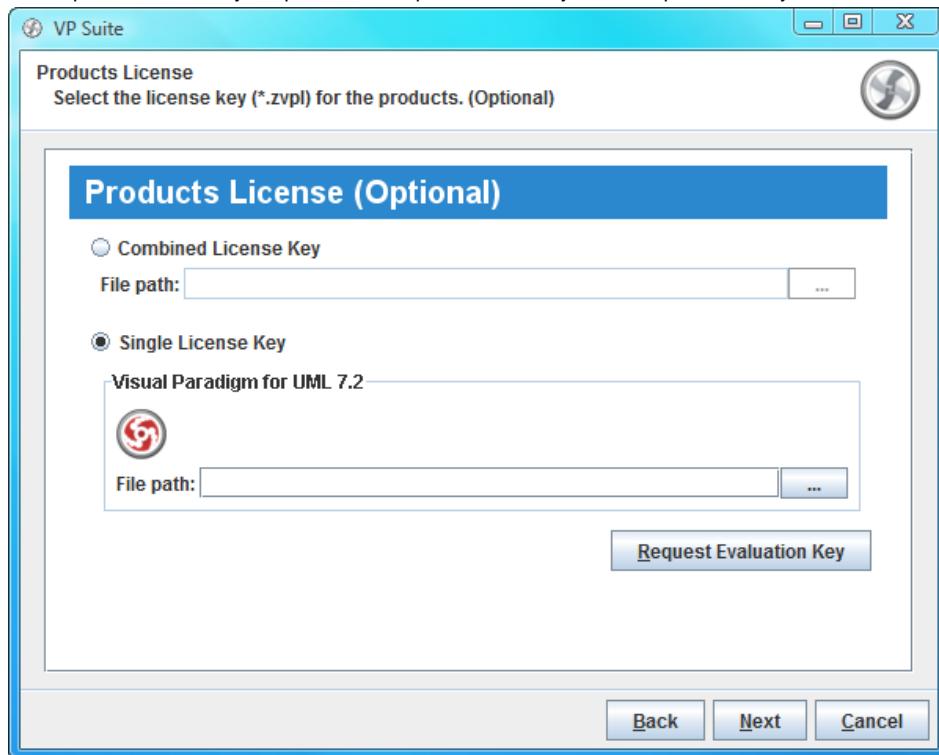
Proxy Setting

- Enable Proxy

Address: Port:
User Name: Password:

Download documentation

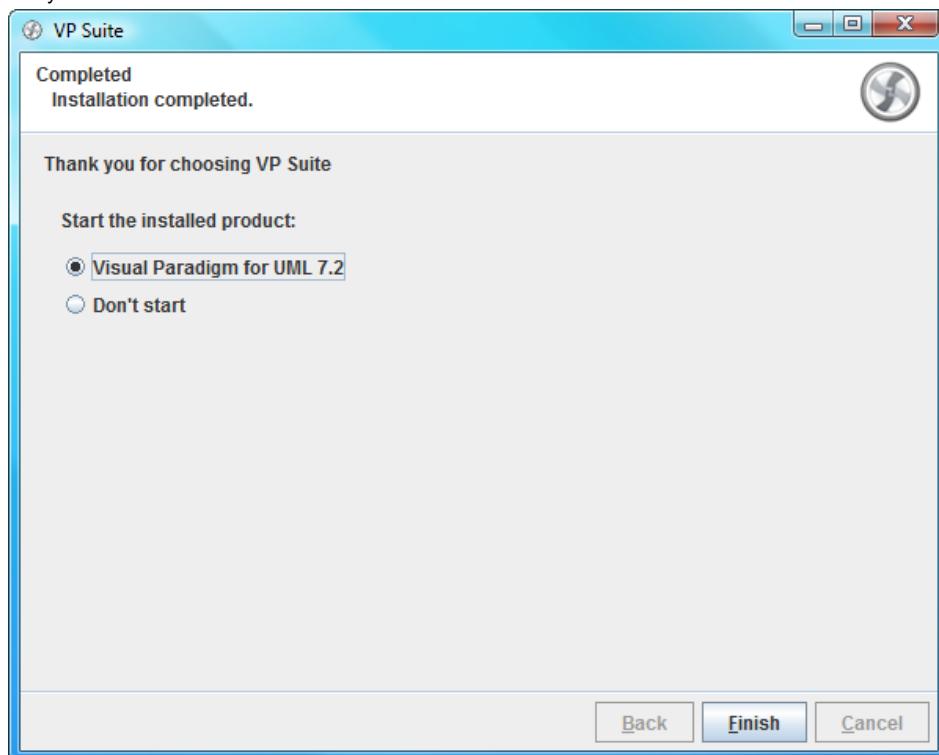
7. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.
 A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.
 The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on **Request Evaluation Key** to ask for another one. This time, try not to select sending the key as attachment.

8. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



Installation completed screen

Installation FAQ

Question: What is the difference between Installer and "Not Install 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 No Install 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 (support-team@visual-paradigm.com) for assistance. It is recommended to include the vp.log file, which can be found at the bin folder of VP Suite installation directory, for our team to diagnose in further.

Question: I don't have administrator right, can I install the software?

Answer: Yes, you can.

Question: Can I change the Edition without re-install the software?

Answer: Yes, you can. Product edition can be changed by running VP Suite Product Edition Manager under the bin folder of VP Suite installation directory. Change of edition takes effect after the restart of affected products. For more details, please read the section Switching Edition a few pages later.

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 (support-team@visual-paradigm.com) or the virus scanner vendor for assistance.

Installing Visual Paradigm for UML on Mac OSX

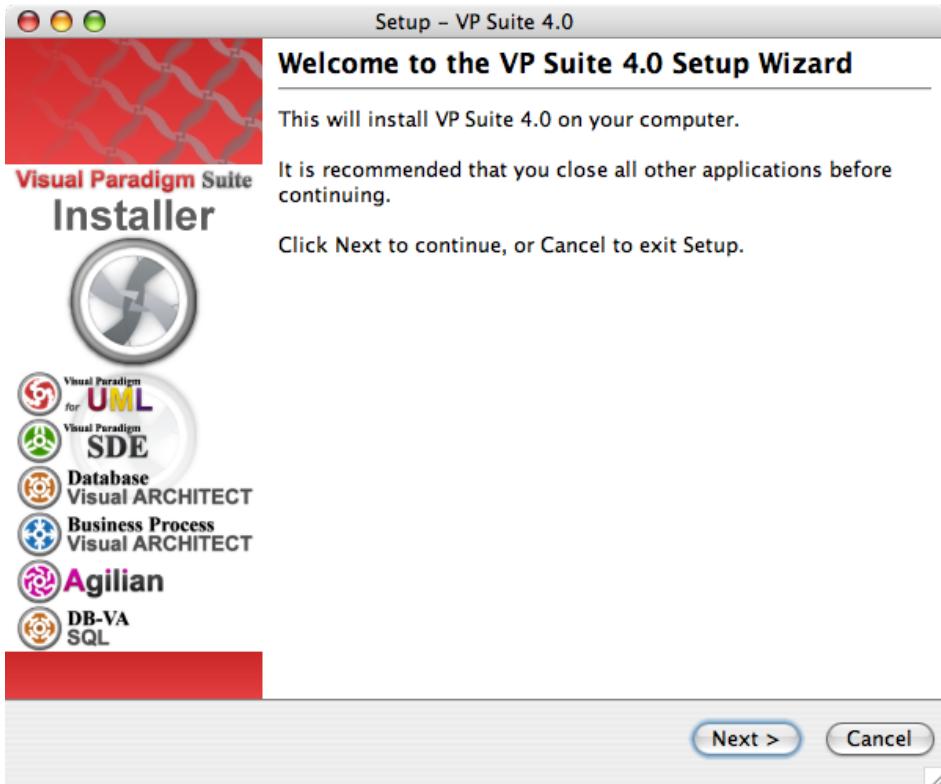
Having downloaded the installer of Visual Paradigm Suite (VP Suite), you can execute it, run through the installation to install the suite as well as VP-UML. If you are using the no-install zip version, you just need to unzip it, launch the suite and install VP-UML. In this chapter, we will go through the installation of VP-UML both with installer (.dmg) and no-install (.zip).

Using installer (.dmg)

1. Execute the downloaded VP Suite installer file:

```
bash ./VP-SUITE-INSTALLER-FILENAME% (e.g. bash ./VP_Suite_Linux_7_0.sh)
```

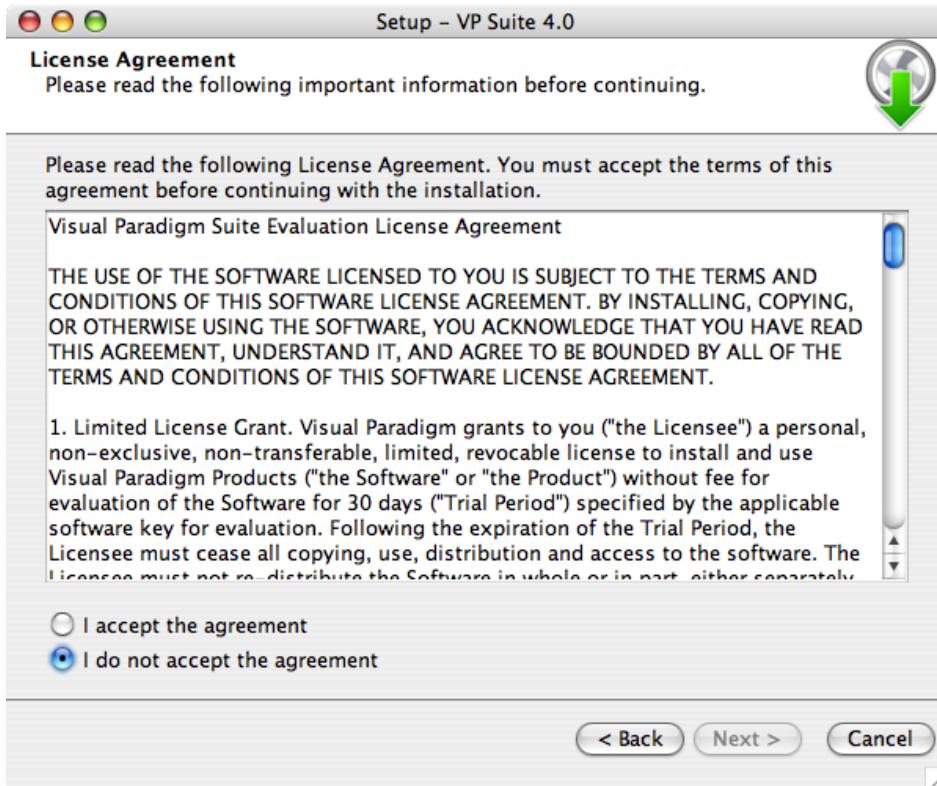
The setup wizard appear as below.



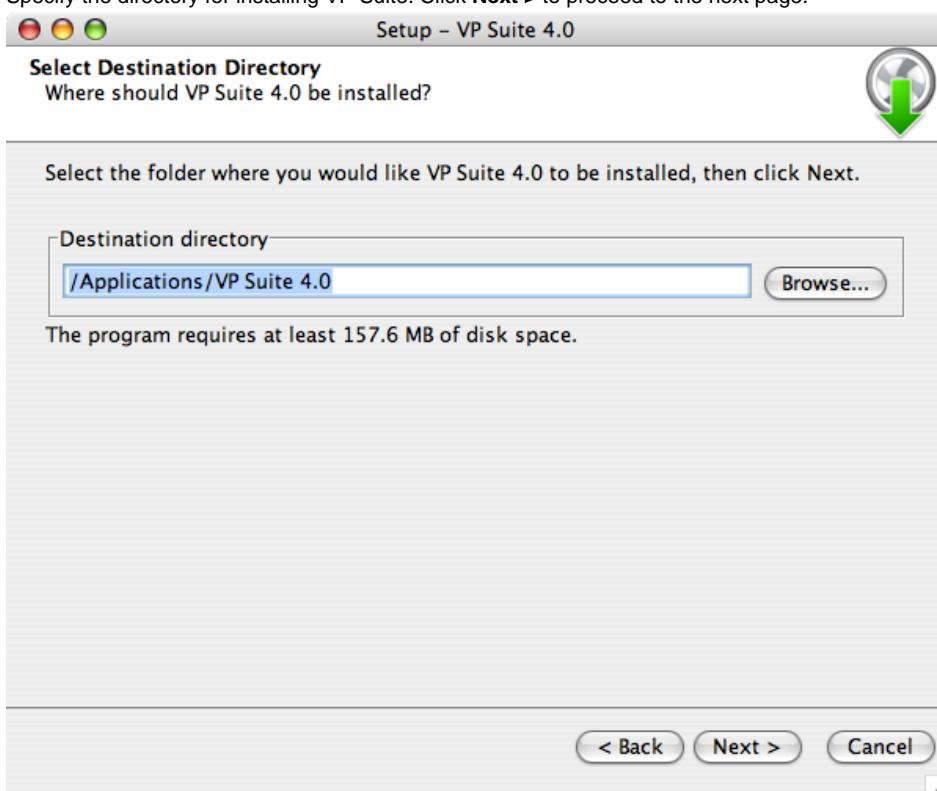
VP Suite Welcome screen

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.

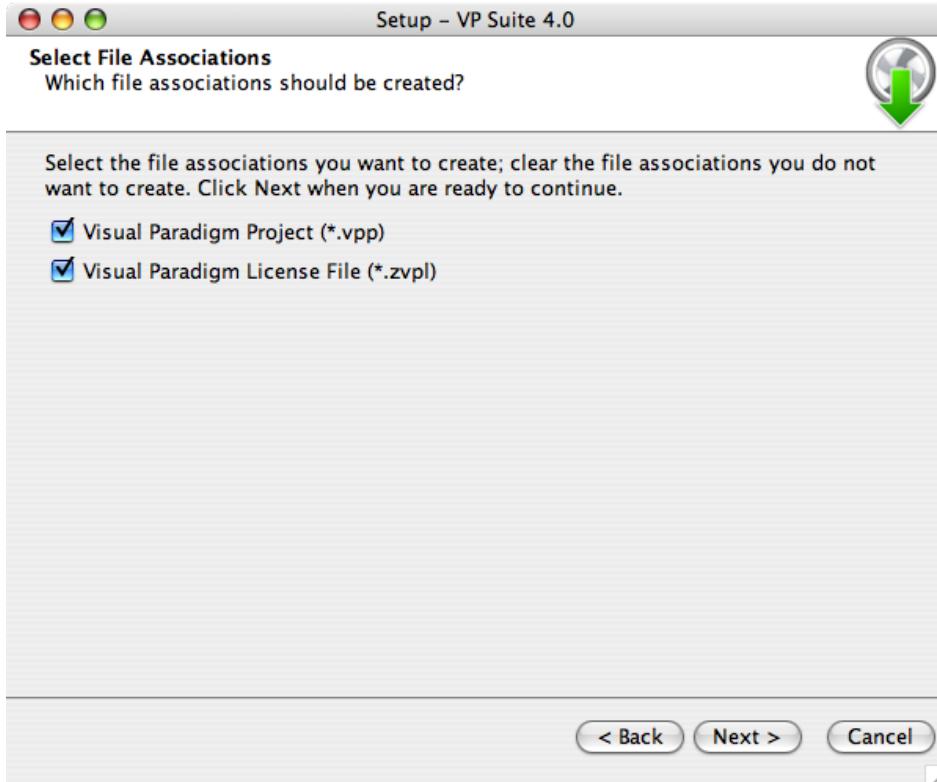


4. Specify the directory for installing VP Suite. Click **Next >** to proceed to the next page.



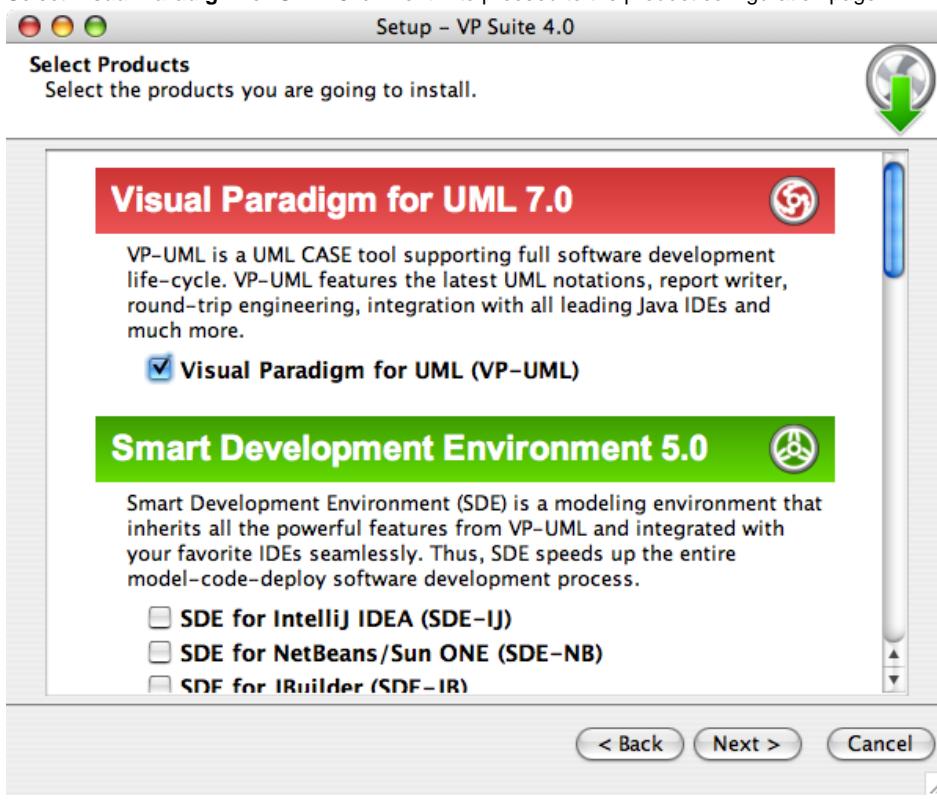
The Select Destination Directory page

5. 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 proceed to the product selection page.



The Select File Associations page

6. Select **Visual Paradigm for UML**. Click **Next >** to proceed to the product configuration page.



The product selection page

7. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.

Configuration

The screenshot shows the 'Configuration' window for Visual Paradigm for UML 7.0. At the top left is the logo and the text 'Visual Paradigm for UML 7.0'. Below it is a dropdown menu labeled 'Edition' with the following options: Enterprise (selected), Professional, Standard, Modeler, Personal, Community, and Viewer. To the right of the dropdown is a link 'Edition Comparison'. The main configuration area includes sections for 'Pre-Configuration File', 'Protocol' (set to 'None'), 'File', and 'HTTP' (set to 'http://').

Select product edition

8. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

Selecting **Enable Product Selector** for Product Selection will result in creating a shortcut under the Start Menu for starting the Product Selector, a utility that lets you realize the installed products with available keys in the floating license server.

The screenshot shows the 'Pre-Configuration File' section. It includes fields for 'Protocol' (set to 'None'), 'File' (with a browse button '...'), and 'HTTP' (set to 'http://').

Pre-configure floating license server connection for floating license user

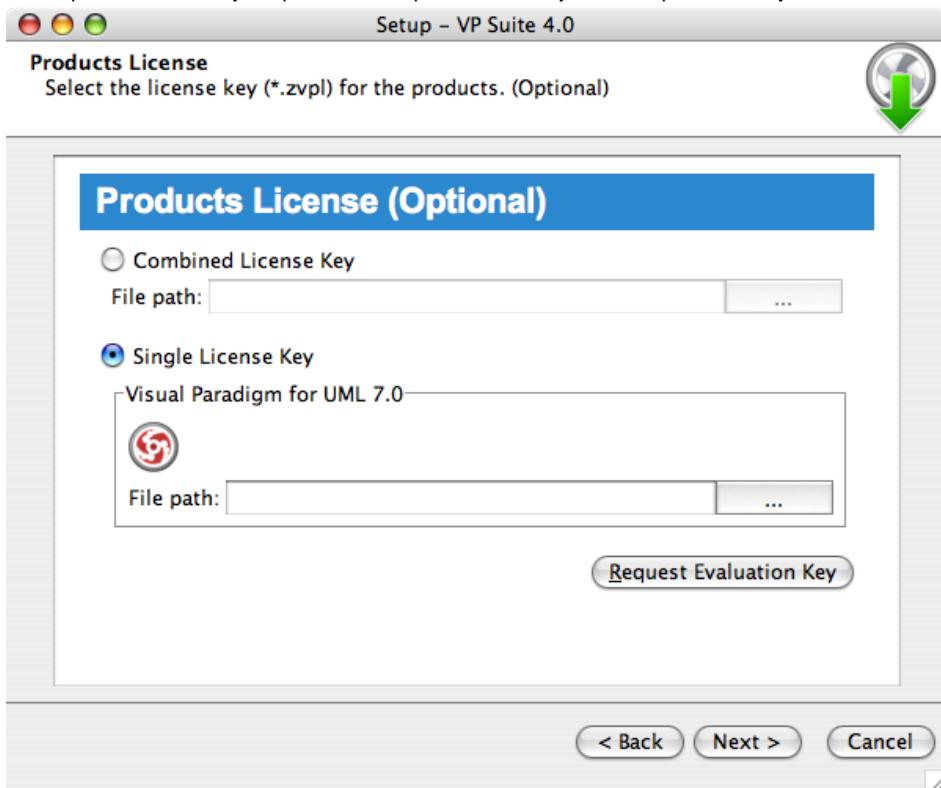
9. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

The screenshot shows the 'Download Documentation' section. It includes a checkbox 'Download Document from Internet:' followed by three options: 'Download Online Help' (checkbox checked), 'Download PDF Version' (checkbox checked), and 'Download HTML Version' (checkbox checked).

The screenshot shows the 'Proxy Setting' section. It includes a checkbox 'Enable Proxy', and fields for 'Address' and 'Port' (both empty), and 'User Name' and 'Password' (both empty).

Download documentation

10. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.
A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.
The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on Request Evaluation Key to ask for another one. This time, try not to select sending the key as attachment.

11. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



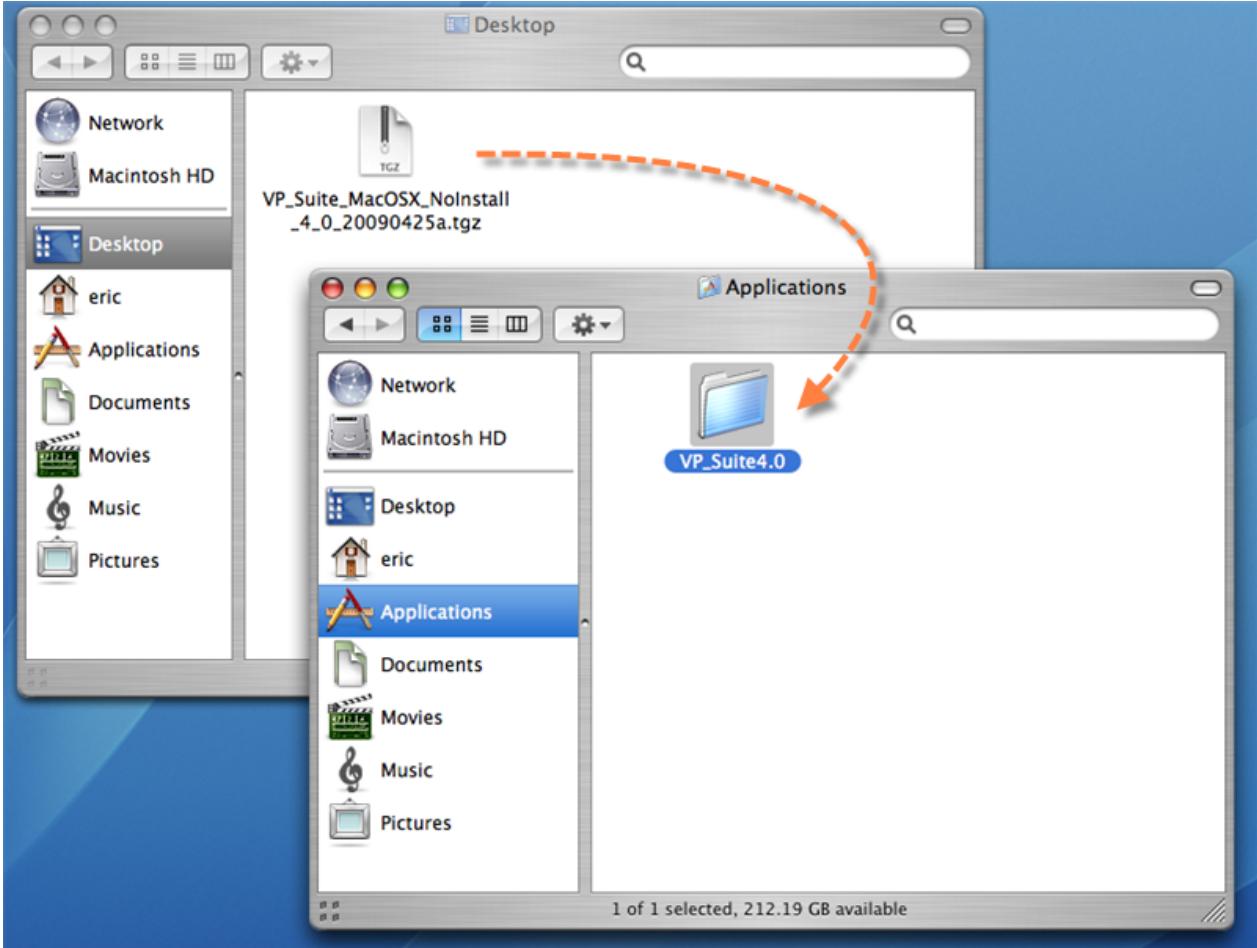
Installation completed screen

Using no install version (.tgz)

1. Decompress the downloaded zip file into a directory.

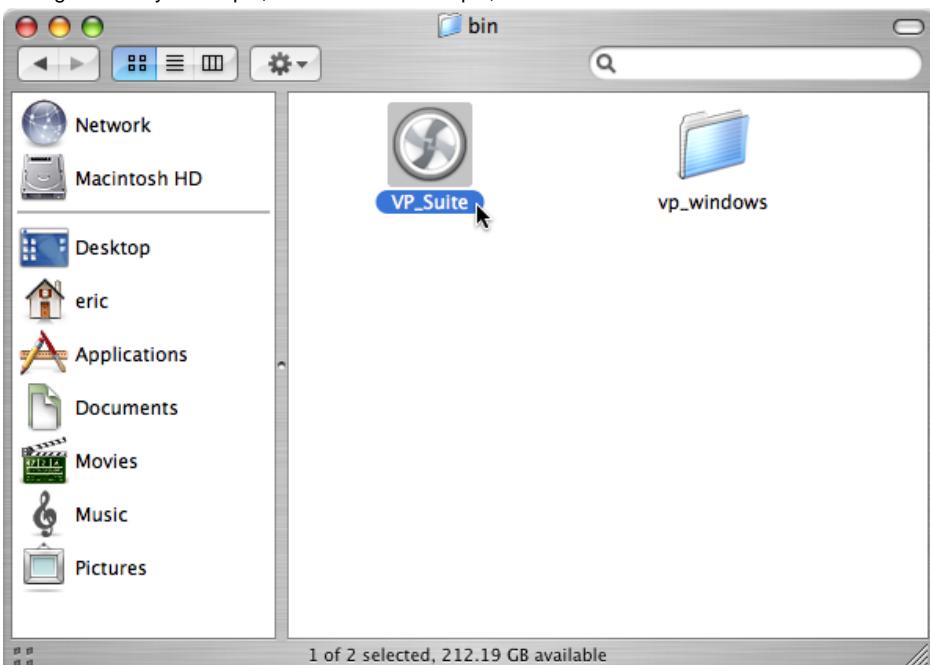
```
tar -zxf %NO-INSTALL-FILE.tar.gz% -C %DESTINATION-FOLDER%
```

This should create a subdirectory named "VP Suite 4.0" where 4.0 is the version number.



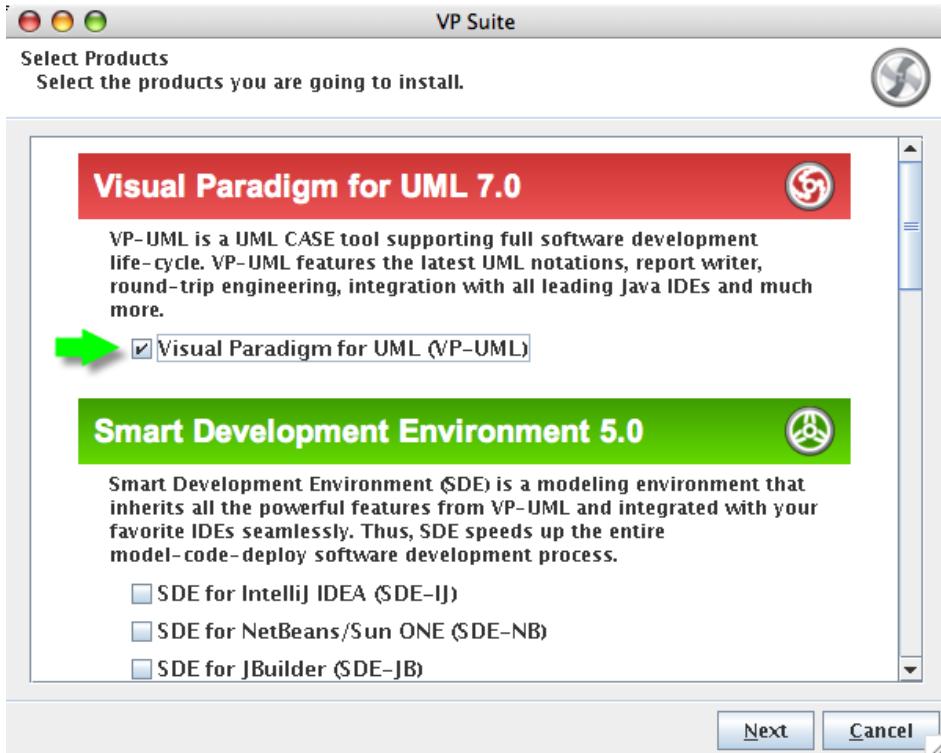
Extracting No-Install zip file

2. Change directory to "VP Suite 4.0/bin" and execute VP Suite in it.



Launching VP Suite

3. Select Visual Paradigm for UML. Click **Next >** to proceed to the product configuration page.



The product selection page

4. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.

Configuration

The screenshot shows the configuration page for 'Visual Paradigm for UML 7.0'. It includes fields for 'Edition' (set to 'Enterprise'), 'Pre-Configuration File' (empty), 'Protocol' (set to 'None'), 'File' (empty), and 'HTTP' (set to 'http://'). A dropdown menu is open over the 'Edition' field, listing seven options: Enterprise, Professional, Standard, Modeler, Personal, Community, and Viewer. The 'Enterprise' option is highlighted.

Select product edition

5. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

The screenshot shows the 'Pre-Configuration File' section. It includes a 'Protocol' dropdown (set to 'None'), an empty 'File' input field, and an 'HTTP' input field set to 'http://'. A tooltip at the bottom of the 'File' field reads: 'Pre-configure floating license server connection for floating license user'.

Pre-configure floating license server connection for floating license user

6. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

Download Documentation

Download Document from Internet:

- Download Online Help** 
- Download PDF Version** 
- Download HTML Version** 

Proxy Setting

- Enable Proxy**

Address:

Port:

User Name:

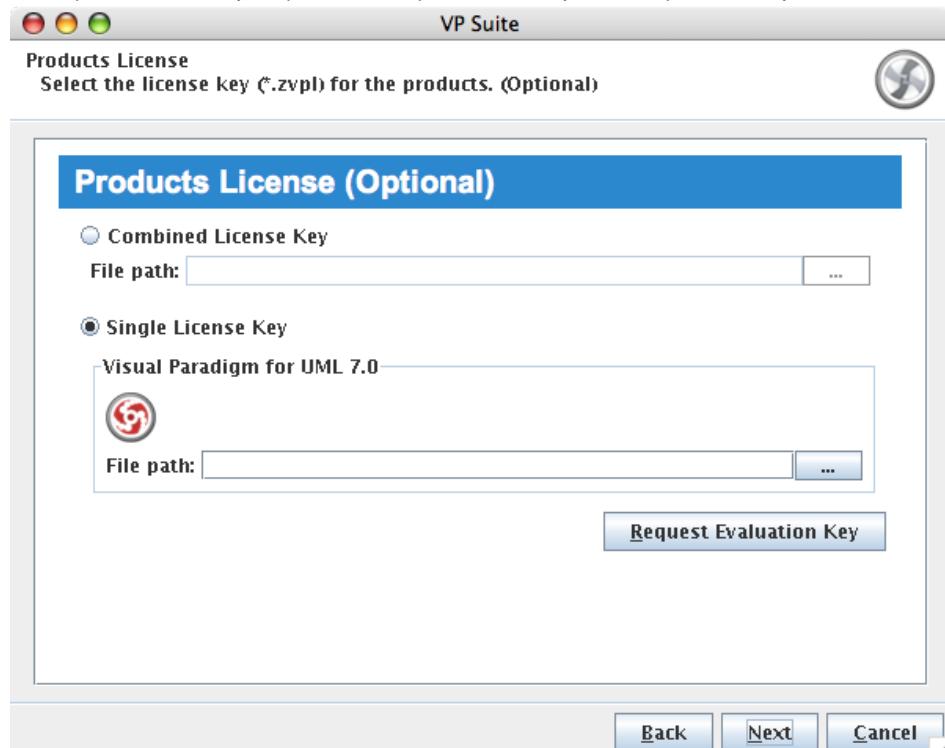
Password:

Download documentation

7. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.

A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.

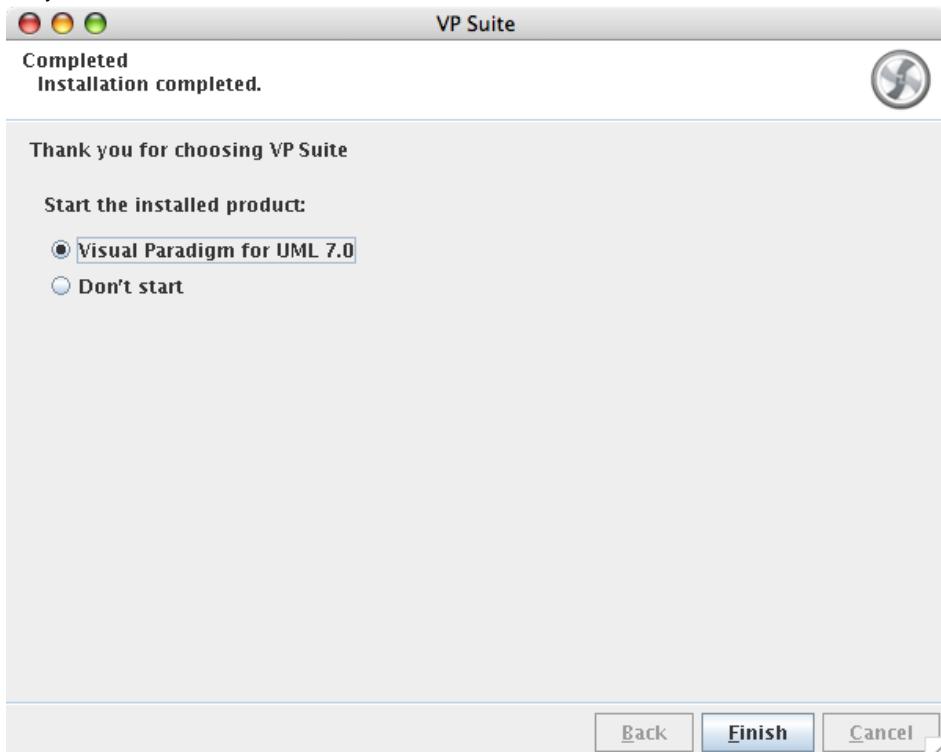
The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on **Request Evaluation Key** to ask for another one. This time, try not to select sending the key as attachment.

8. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



Installation completed screen

Installation FAQ

Question: What is the difference between Installer and "Not Install 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 No Install 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 (support-team@visual-paradigm.com) for assistance. It is recommended to include the vp.log file, which can be found at the bin folder of VP Suite installation directory, for our team to diagnose in further.

Question: I don't have administrator right, can I install the software?

Answer: Yes, you can.

Question: Can I change the Edition without re-install the software?

Answer: Yes, you can. Product edition can be changed by running VP Suite Product Edition Manager under the bin folder of VP Suite installation directory. Change of edition takes effect after the restart of affected products. For more details, please read the section *Switching Edition* a few pages later.

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 (support-team@visual-paradigm.com) or the virus scanner vendor for assistance.

Installing Visual Paradigm for UML on Linux and Unix

Having downloaded the installer of Visual Paradigm Suite (VP Suite), you can execute it, run through the installation to install the suite as well as VP-UML. If you are using the no-install zip version, you just need to unzip it, launch the suite and install VP-UML. In this chapter, we will go through the installation of VP-UML both with installer (.sh) and no-install (.zip).

Using installer (.sh)

1. Execute the downloaded VP Suite installer file:

```
bash ./VP-SUITE-INSTALLER-FILENAME% (e.g. bash ./VP_Suite_Linux_7_0.sh)
```

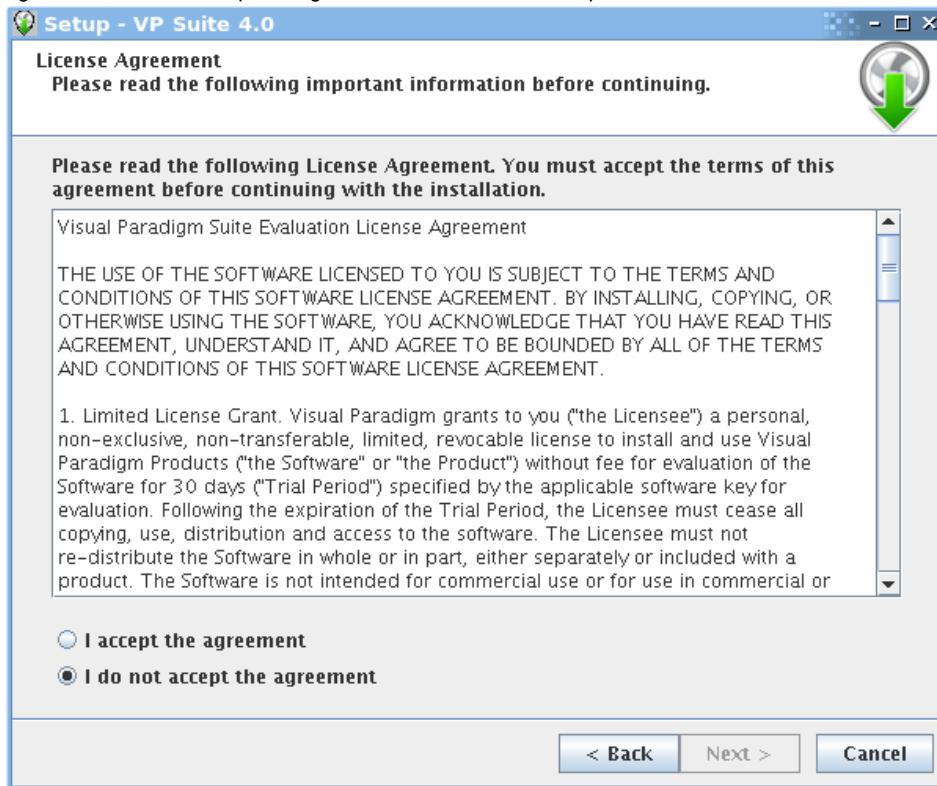
The setup wizard appear as below.



VP Suite Welcome screen

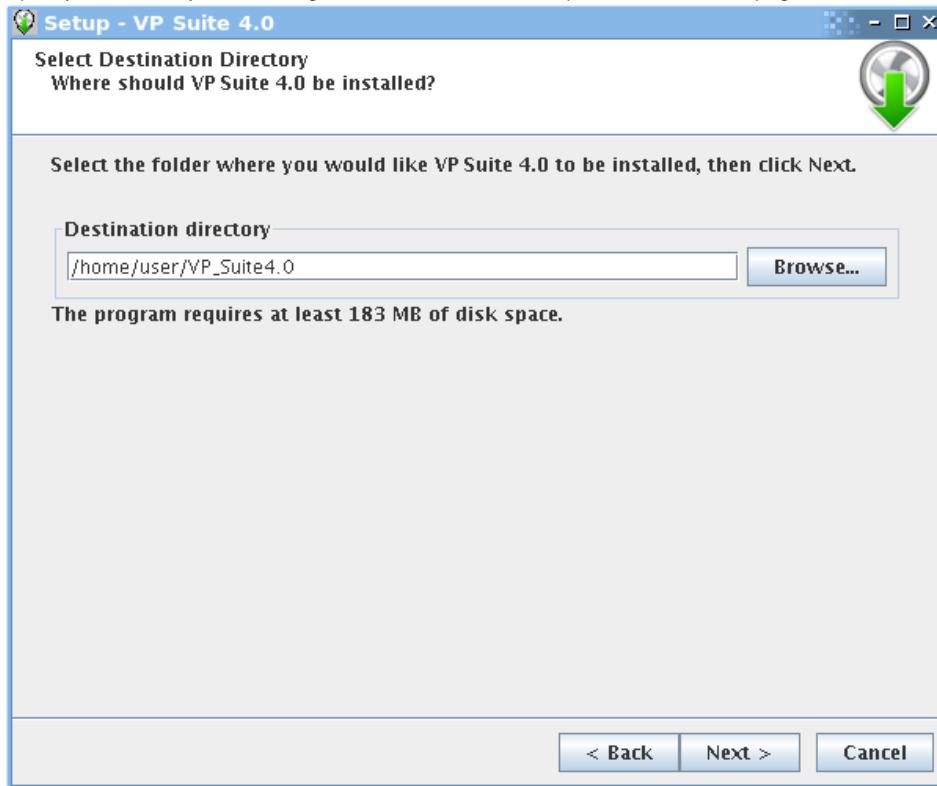
2. Click **Next** to proceed to the License agreement page.

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.



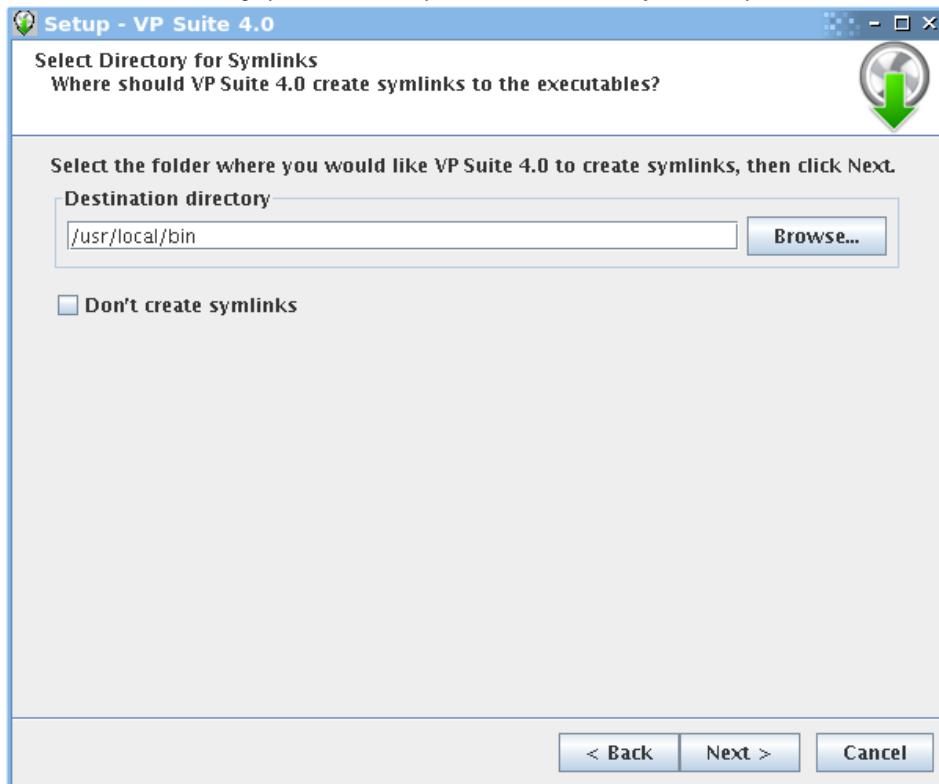
The License Agreement

4. Specify the directory for installing VP Suite. Click **Next >** to proceed to the next page.



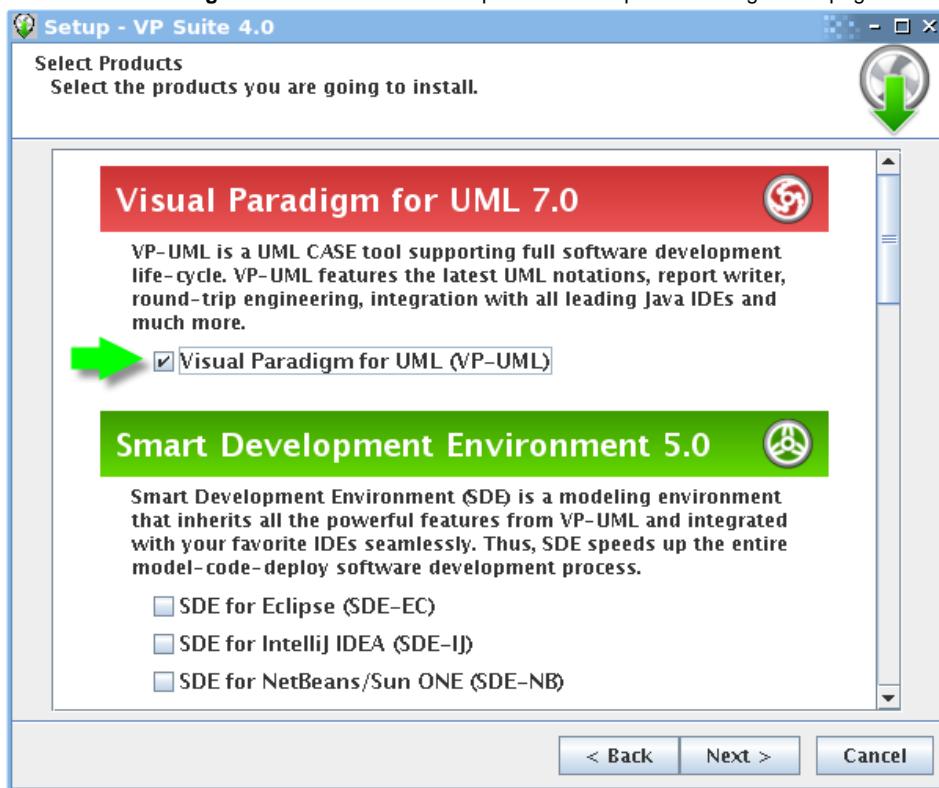
The Select Destination Directory page

5. select a folder for creating symlinks. You may select **Don't create symlinks** if you do not want to. Click **Next** to proceed.



The Select Symlinks Directory page

6. Select **Visual Paradigm for UML**. Click **Next >** to proceed to the product configuration page.



The product selection page

7. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.

Configuration

Visual Paradigm for UML 7.0

	Edition: Enterprise <input type="button" value="▼"/>	Edition Comparison
Pre- Configuration	Enterprise	
Protocol:	Standard	
File:	Modeler	
HTTP:	Personal	
	Community	
	Viewer	

Select product edition

8. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

Selecting **Enable Product Selector** for Product Selection will result in creating a shortcut under the Start Menu for starting the Product Selector, a utility that lets you realize the installed products with available keys in the floating license server.

Pre-Configuration File

Protocol: <input type="button" value="None ▼"/>
File: <input type="text"/> <input type="button" value="..."/>
HTTP: <input type="text"/> http://

Product Selector (for floating license)

Enable Product Selector

Pre-configure floating license server connection for floating license user

9. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

Download Documentation

Download Document from Internet:

- Download Online Help 
- Download PDF Version 
- Download HTML Version 

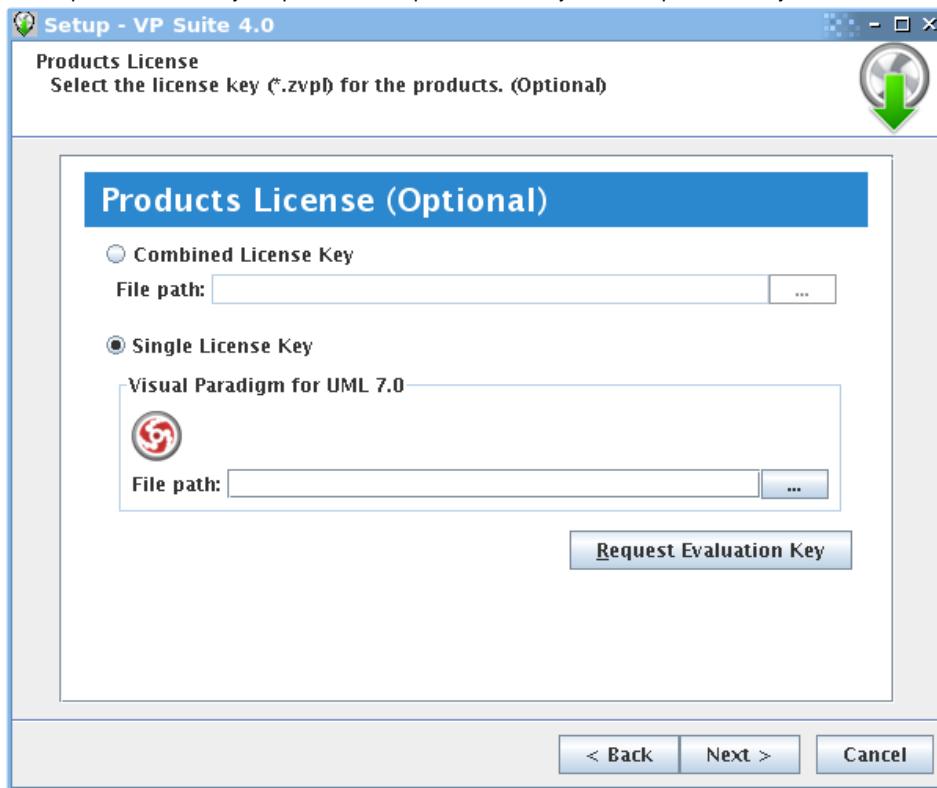
Proxy Setting

- Enable Proxy**

Address: <input type="text"/>	Port: <input type="text"/>
User Name: <input type="text"/>	Password: <input type="text"/>

Download documentation

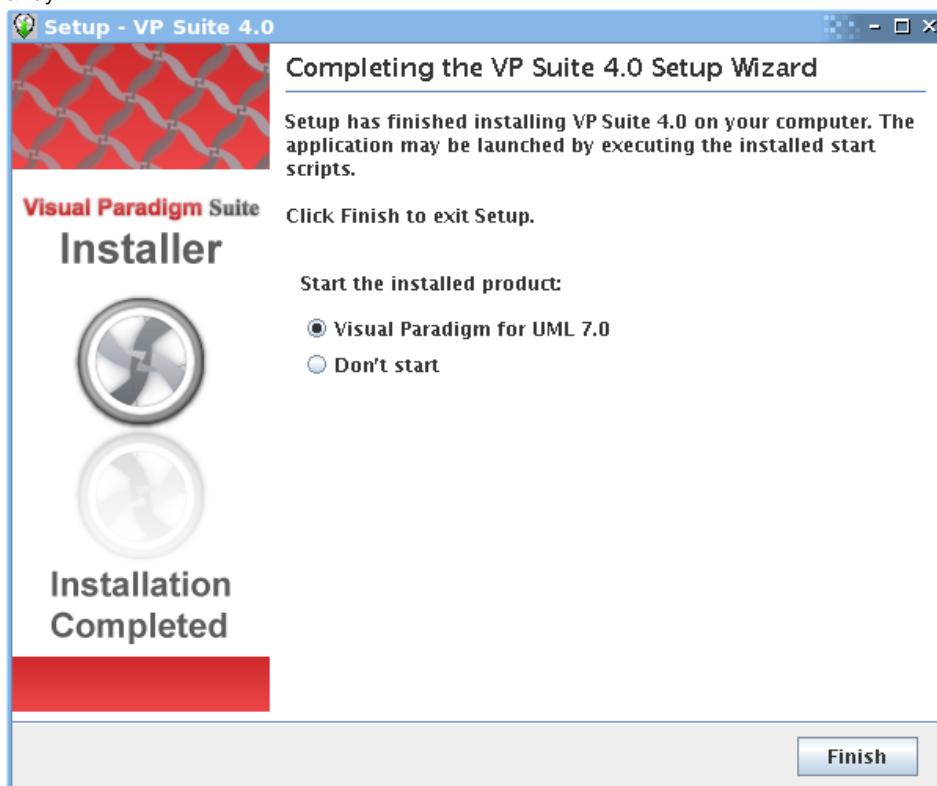
10. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.
A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.
The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on Request Evaluation Key to ask for another one. This time, try not to select sending the key as attachment

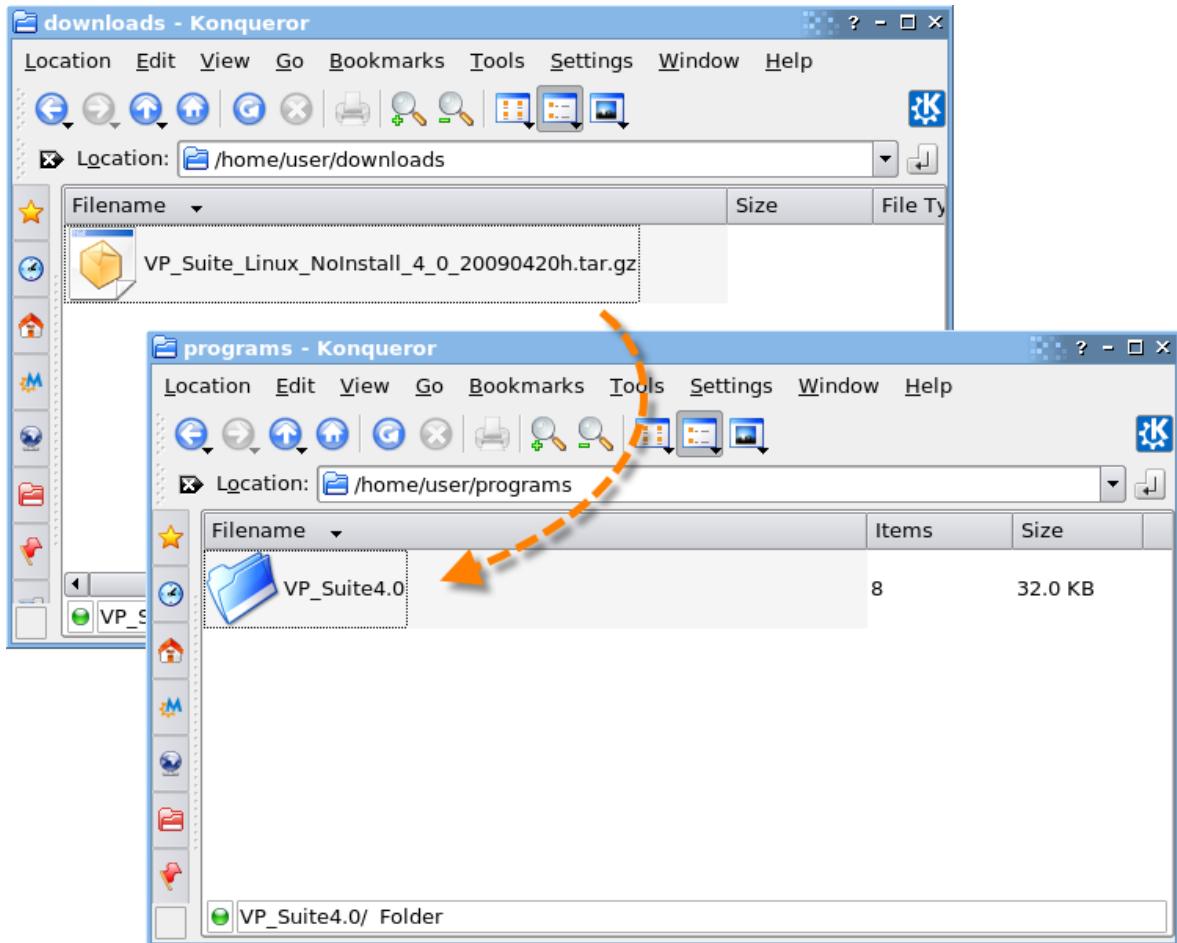
11. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



Installation completed screen

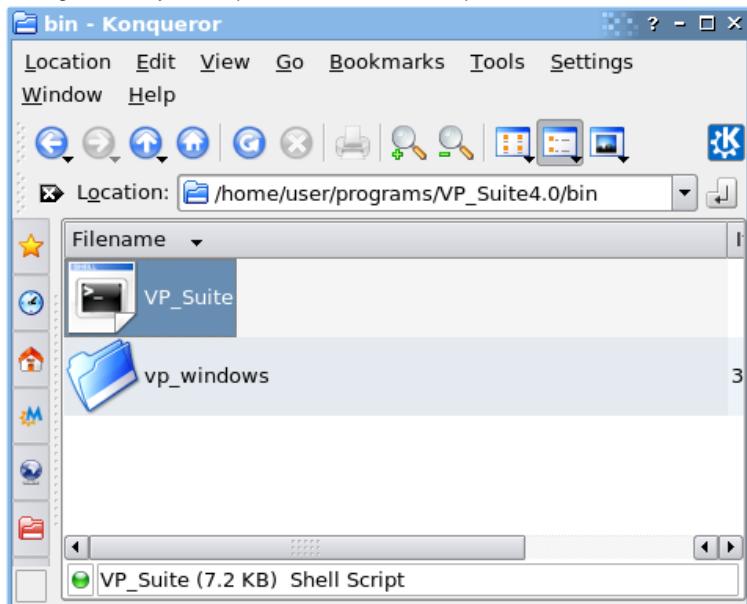
Using no install version (.tar.gz)

1. Decompress the downloaded zip file into a directory.
`tar -zxf %NO-INSTALL-FILE.tar.gz% -C %DESTINATION-FOLDER%`
This should create a subdirectory named "VP Suite 4.0"; where 4.0 is the version number.



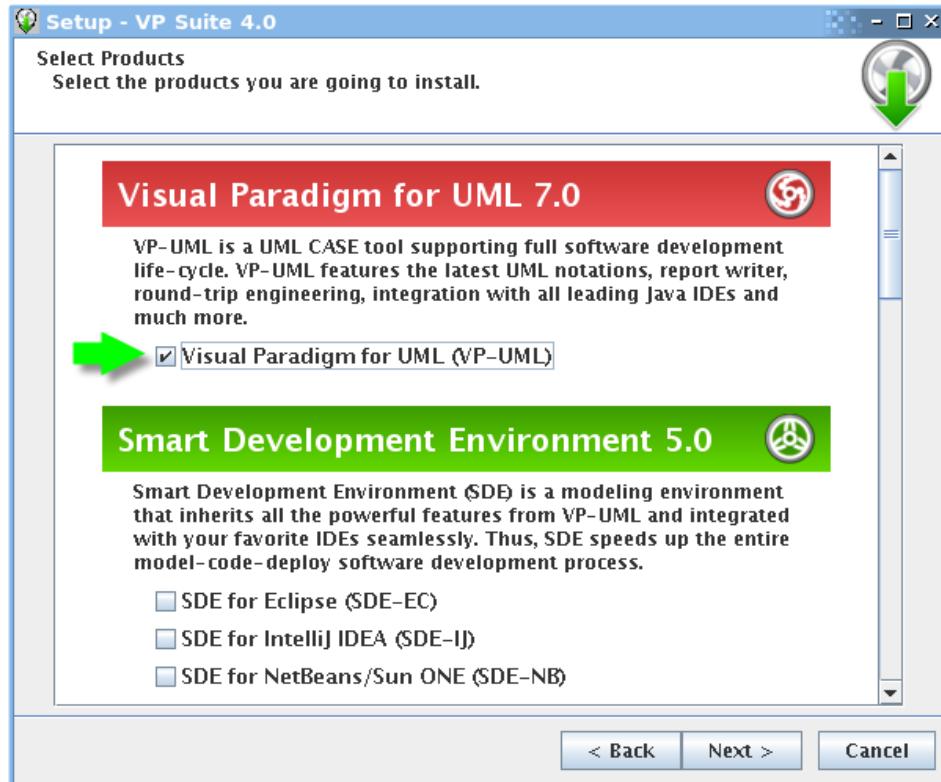
Extracting No-Install zip file

2. Change directory to "VP Suite 4.0/bin"; and execute VP Suite in it.



Launching VP Suite

3. Select Visual Paradigm for UML. Click **Next >** to proceed to the product configuration page.



The product selection page

4. VP-UML features vary by product edition. For more details on the features supported by different editions, click the hyperlink **Edition Comparison** which brings you to the web page of VP-UML feature comparison for different editions.

Configuration

Visual Paradigm for UML 7.0

	Edition:	Enterprise	Edition Comparison
Pre-Configurat	Enterprise		
Protocol:	Professional		
File:	Standard		
HTTP:	Modeler		
	Personal		
	Community		
	Viewer		

Select product edition

5. For users who are going to run VP-UML with floating license, the Pre-Configuration File section enables you to configure the connection(s) to floating license server by providing a configuration file. The pre configuration is optional. If you do not define the connection here, you will need to do so when starting VP-UML the first time. Below is the pre-configuration file file content:

```
?xml version="1.0" encoding="UTF-8"?>
<LicenseServer>
<Server accessCode="" host="" port="" />
</LicenseServer>
```

Pre-Configuration File

Protocol:	None
File:	<input type="text"/>
HTTP:	http://

Pre-configure floating license server connection for floating license user

6. Select **Download Online Help** if you want to be able to access the Help contents from within the tool. Select **Download PDF/HTML Version** if you wish to read the documentation in this two types of formats. Press **Next >** to proceed to the license import page.

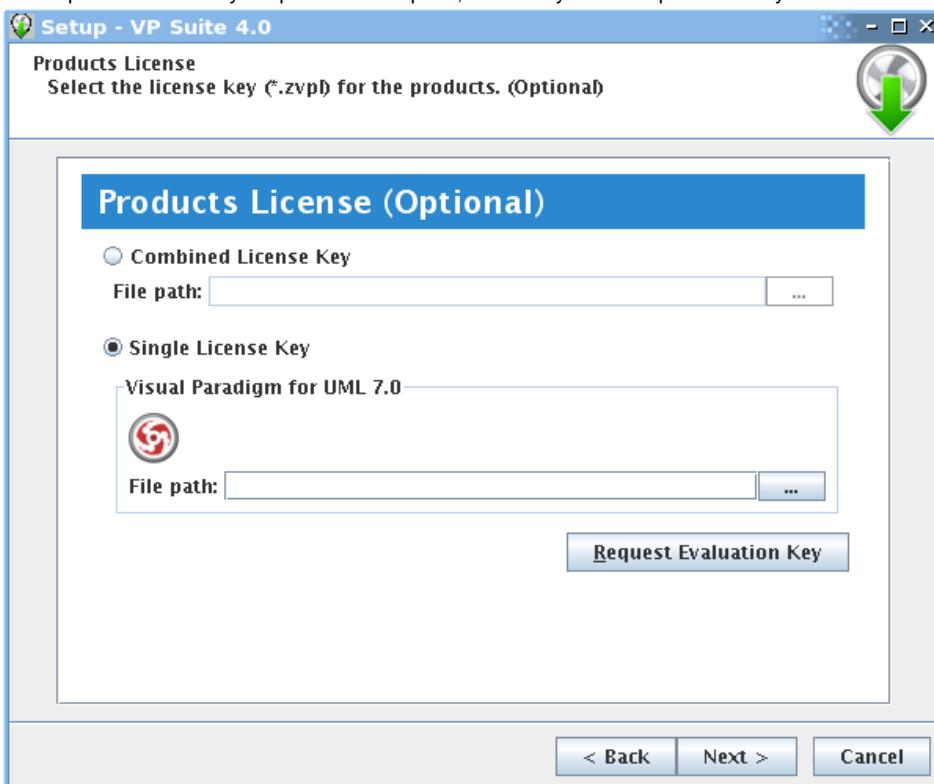


Download documentation

7. A combined license key is a key file which allows unlocking multiple products. To import a combined license key, select Combined License Key and specify the file path of the key.

A single license key is a key file which allows unlocking only one product. To import a single license key, select Single License Key and specify the file path of the key.

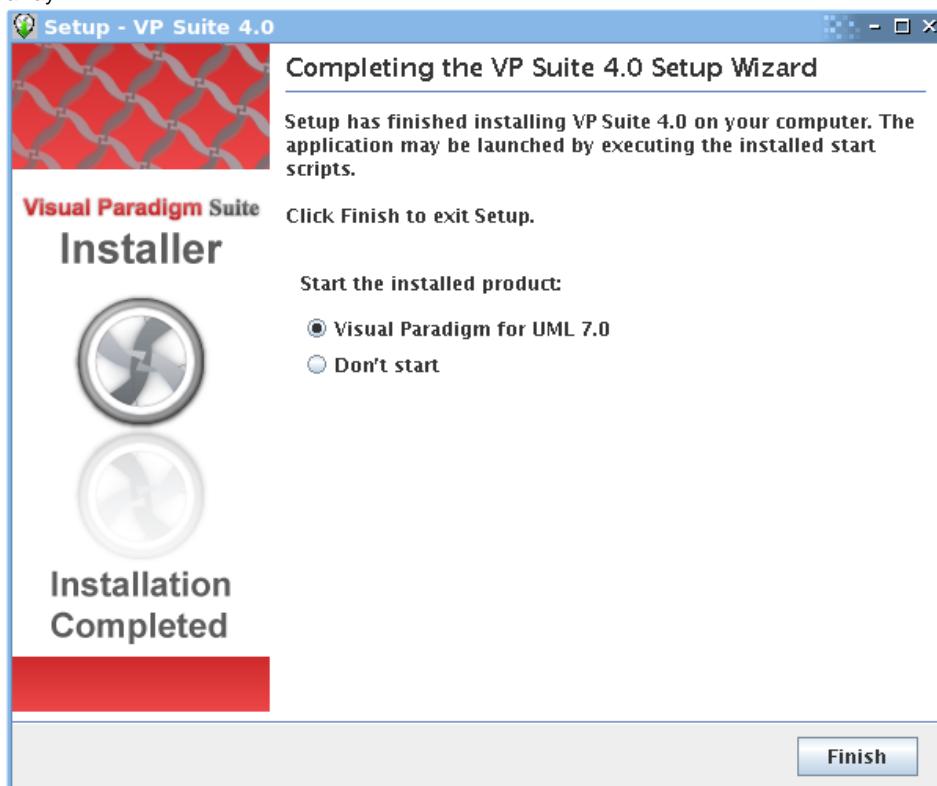
The import of license key is optional at this point, because you can import it when you run VP-UML. Click **Next >** to proceed with copying files.



The license key selection page

NOTE: For evaluation, the key should be sent to your email box (unless you have chosen not to receive it before download). If you have not received the email with key yet, and if you have selected to receive the key as attachment, the email might be treated as spam by mistake. Click on **Request Evaluation Key** to ask for another one. This time, try not to select sending the key as attachment.

8. Upon finishing, you can select whether to start VP-UML or not. Keep **Visual Paradigm for UML** selected and click **Finish** will run VP-UML right away.



Installation completed screen

Installation FAQ

Question: What is the difference between Installer and "Not Install 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 No Install 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 (support-team@visual-paradigm.com) for assistance. It is recommended to include the vp.log file, which can be found at the bin folder of VP Suite installation directory, for our team to diagnose in further.

Question: I don't have administrator right, can I install the software?

Answer: Yes, you can.

Question: Can I change the Edition without re-install the software?

Answer: Yes, you can. Product edition can be changed by running VP Suite Product Edition Manager under the bin folder of VP Suite installation directory. Change of edition takes effect after the restart of affected products. For more details, please read the section Switching Edition a few pages later.

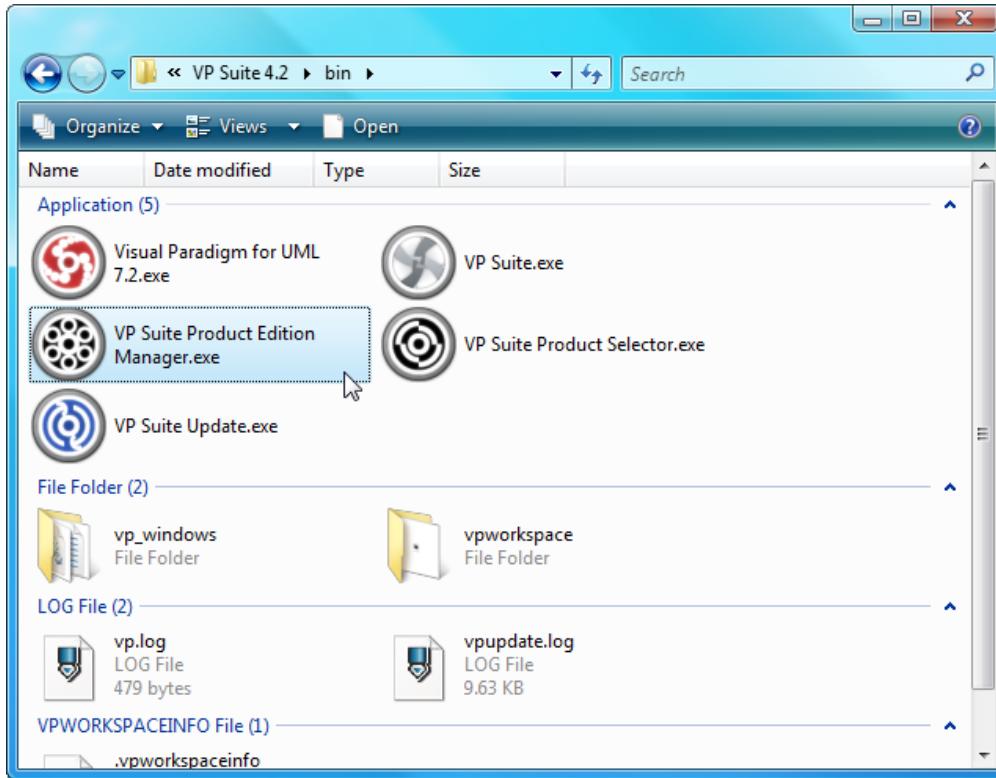
Question: The installer file is detected to contain a virus. What can I do?

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 (support-team@visual-paradigm.com) or the virus scanner vendor for assistance.

Switching edition

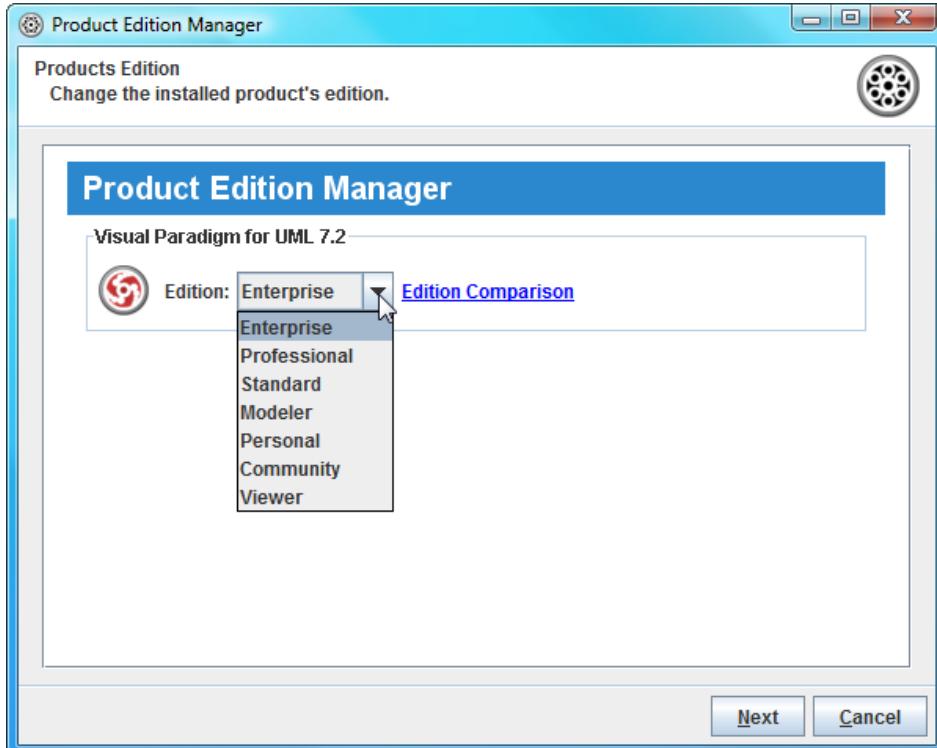
Product edition can be changed through the Product Edition Manager without the need of re-installation. When you attempt to switch to another edition, make sure you have the required key in order to run the product in new edition.

1. Launch the **Product Edition Manager** under the bin folder of VP Suite installation folder.



Launching Product Edition Manager

2. Select the edition to switch to in the edition manager.



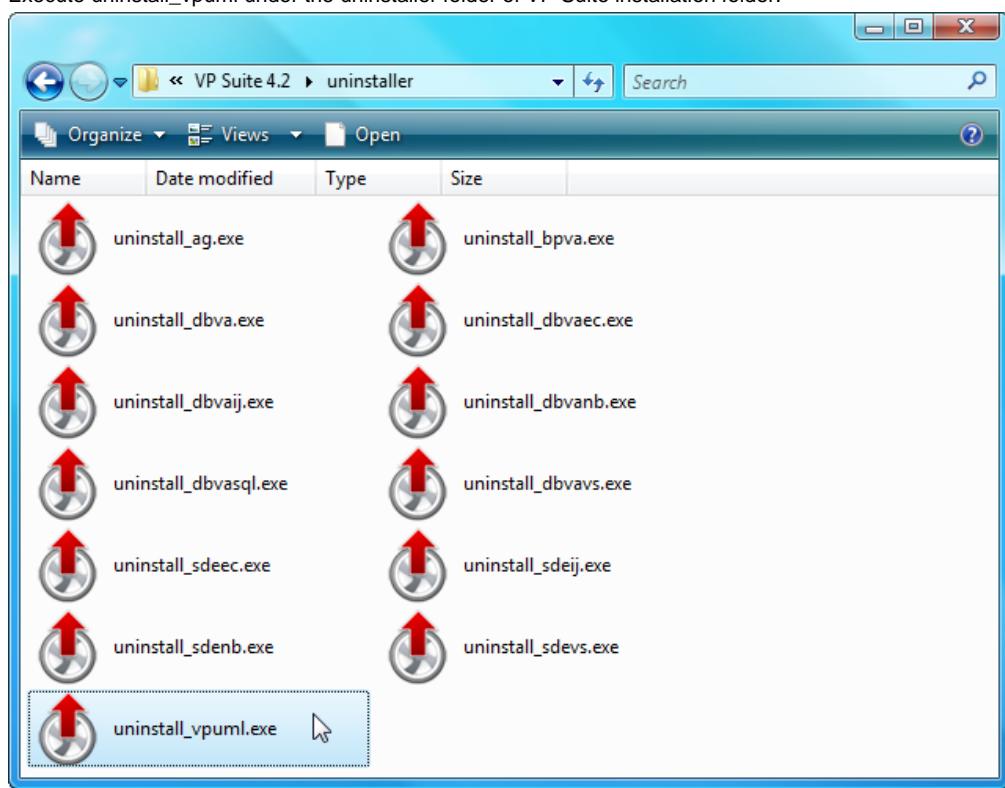
Select product edition

3. Click **Next** to confirm.

Uninstalling Visual Paradigm for UML

Uninstalling VP-UML will cause VP-UML to be removed from VP-Suite.

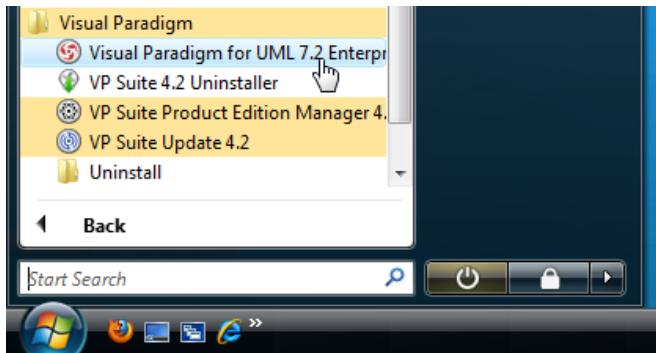
1. Close VP-UML if it is running.
2. Execute `uninstall_vpuml` under the `uninstaller` folder of VP Suite installation folder.



Starting Visual Paradigm for UML

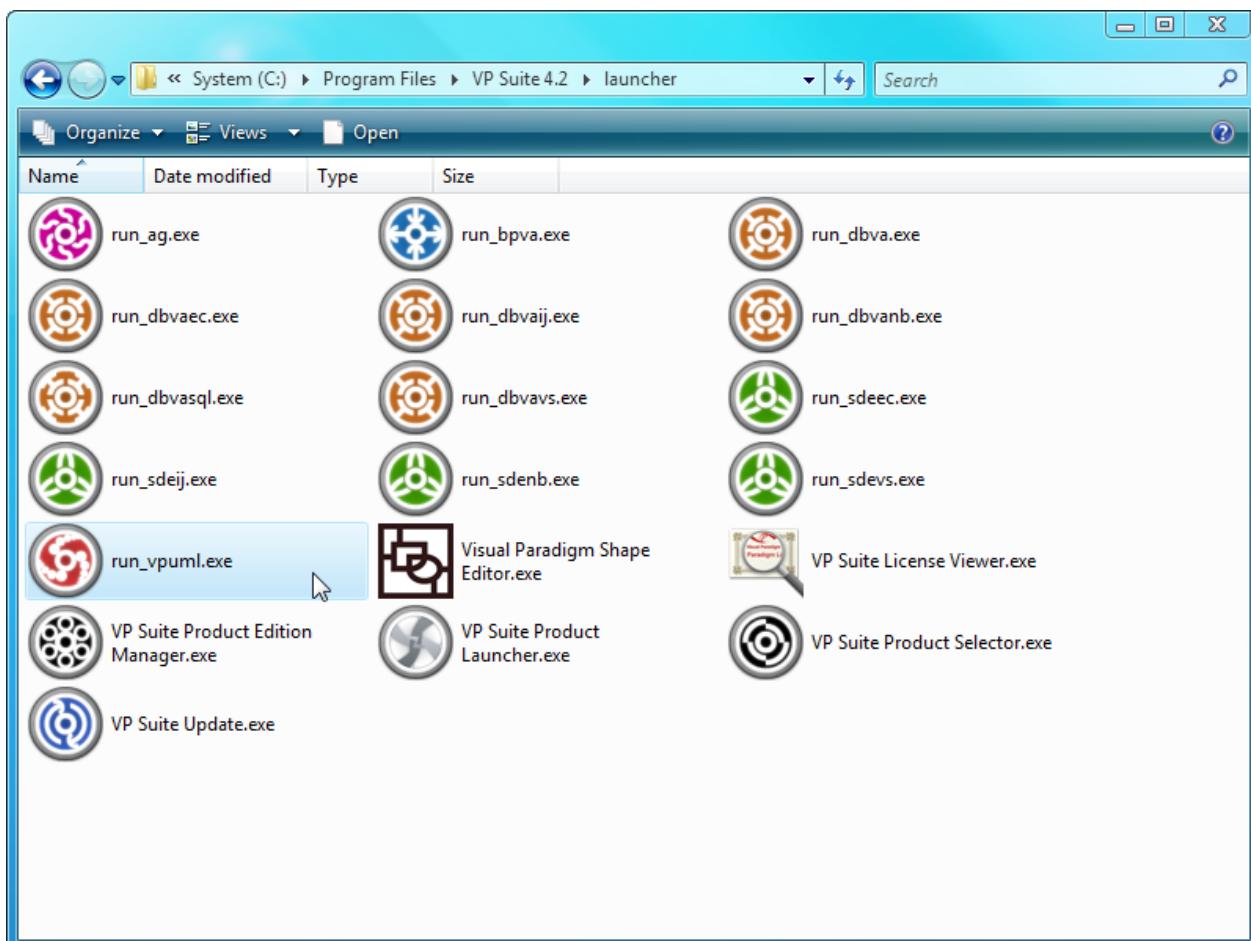
Ways of starting Visual Paradigm for UML

VP-UML can be launched through accessing the Start Menu.



Starting Visual Paradigm for UML via Start Menu

Alternatively, you can start VP-UML by executing the launcher `run_vpuml` in the launcher folder of VP Suite installation directory. Users in all operating system can start VP-UML in this way.



Starting Visual Paradigm for UML with launcher

For Linux users, VP-UML can be started through the shortcuts in desktop, created by the installer.



Starting Visual Paradigm for UML in Linux using the icon in desktop

Starting Visual Paradigm for UML (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 VP-UML with a startup script in order to connect to the server. Here are the steps:

1. Copy VP-UML.bat under the scripts folder of VP Suite 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\  
lib\lib04.jar;..\lib\lib05.jar;..\lib\lib06.jar;..\lib\lib07.jar;..\lib\lib08.jar;..\lib\  
lib\lib09.jar;..\lib\lib10.jar" -Djava.net.preferIPv4Stack=true RV %*  
popd
```

Editing the start up script

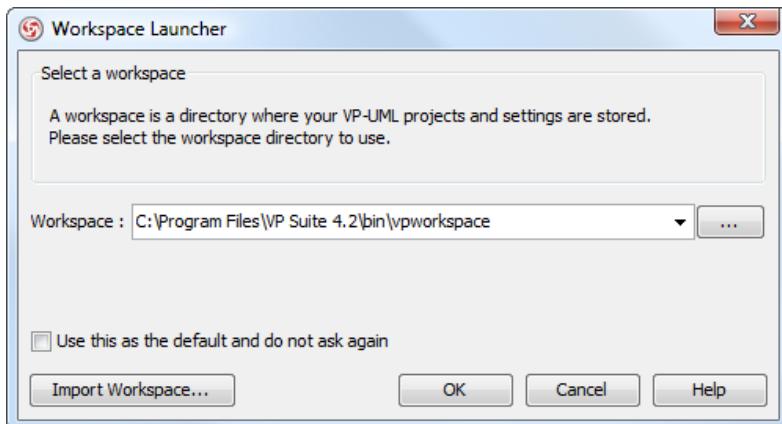
4. Save

5. From now on, execute Startup.bat to run VP-UML

Select workspace

A workspace is a directory used to store all settings, user interface perspectives and other preferences defined for the working environment (settings can be configured via **Tools > Options...** in VP-UML). A workspace also stores all the teamwork login information and local copies of teamwork projects. In the case of switching computers, you simply need to copy the whole workspace directory to the new computer and specify the new workspace when starting VP-UML. All your teamwork information and settings will then be transferred to the new computer.

The Workspace Launcher appear when running VP-UML.



Workspace Launcher

Specify the workspace folder.

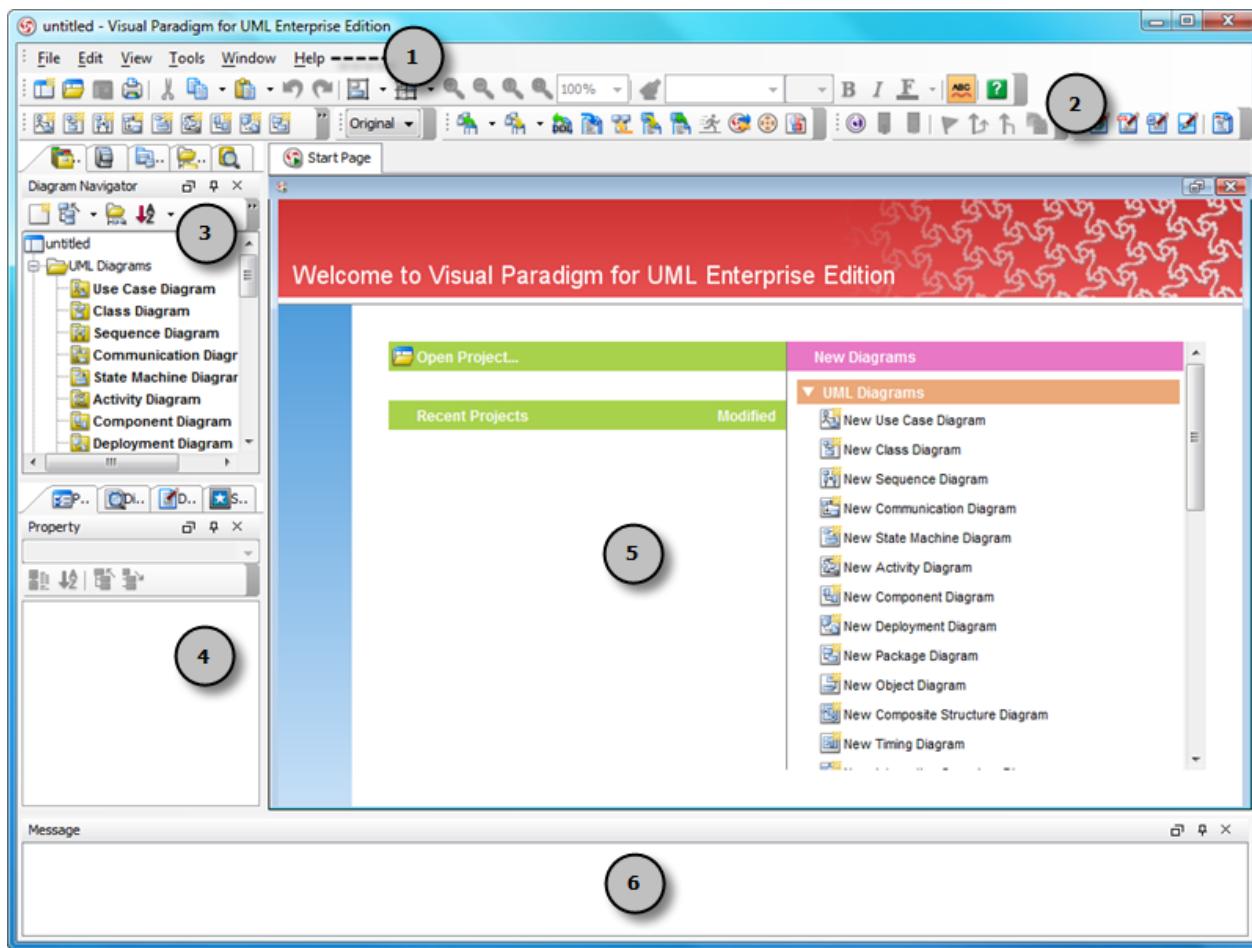
If you do not want this dialog box to appear again, check **Use this as the default and do not ask again**. This will cause VP-UML to open the specified workspace folder automatically the next time.

If you already have an existing workspace, you can import the settings from there by clicking **Import Workspace....**

Click **OK** to continue.

Interface overview

VP-UML's user interface comprises a menu bar, a toolbar several panes for model navigation, a message pane and a diagram pane that occupy most spaces.



User Interface of Visual Paradigm for UML

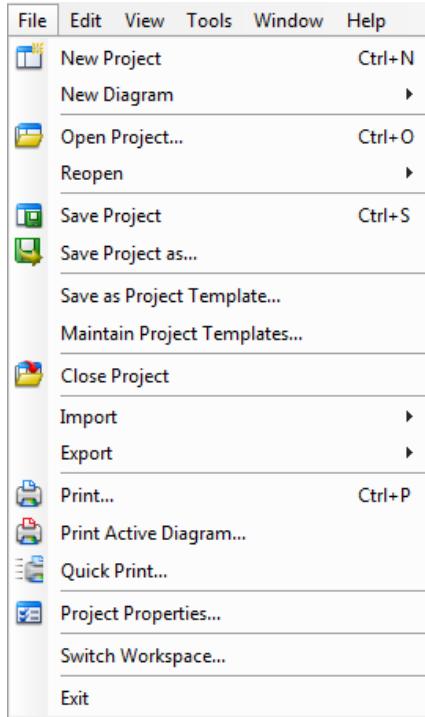
No.	Name	Description
1	Menu bar	The menu bar at the top of the window allows you to select and perform various operations in Visual Paradigm for UML.
2	Toolbar	Toolbar, which is below the menu bar, is the extension of menu. All buttons are presented as groups of icons that handily placed for users.
3	Diagram navigator	A place where diagrams are listed, and where you can create and access diagrams base on their types.
4	Property pane	The properties of chosen model/ shapes will be shown on properties pane upon selection.
5	Diagram pane	The diagram will be displayed in diagram pane.
6	Message pane	Information, notification or warnings will be shown here.

Description of user interface

Main menu

The main menu, which is on the top of the window, allows you to select and perform various operations in Visual Paradigm for UML.

File menu

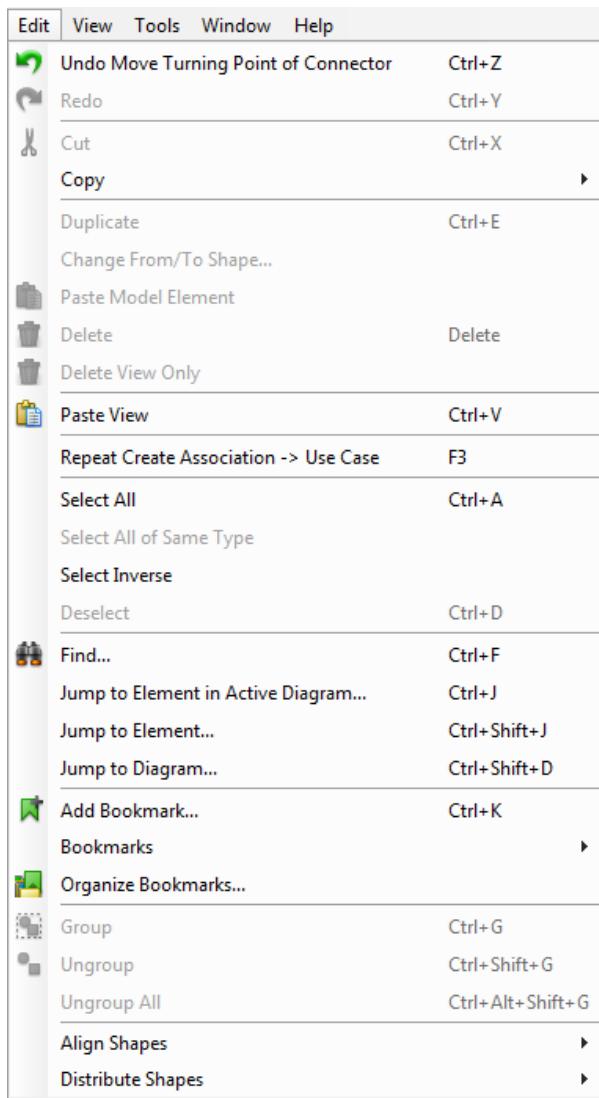


The *File* menu

The **File** menu enables you to:

- Create a project
- Create a diagram
- Open project
- Save project
- Save and manage project template.
- Import project data from the following media: VP-UML Project, Rose Project, XMI, XML, Erwin Project, Telelogic Rhapsody Project, Telelogic System Architect, Rational Model, Rational DNX, MS Word (Use Case Model documentation exported from VP), Excel (Exported from VP), Visio, NetBeans
- Export project into the following formats: VP-UML Project, XMI, XML, MS Word (for Use Case Model), Excel
- Export images (JPG, PNG, SVG, EMF, PDF)
- Printing
- Set project properties
- Exit

Edit menu

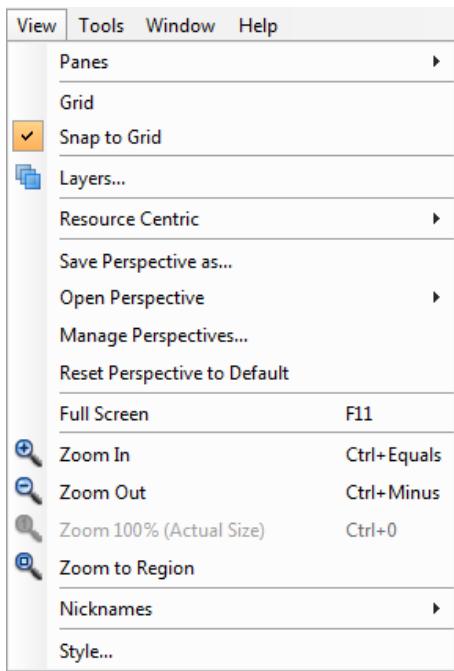


*The **Edit** menu*

The **Edit** menu enables you to:

- Undo and Redo
- Cut
- Copy
- Duplicate
- Delete
- Change the end model element of connector
- Repeat an action
- Select everything in diagram
- Find a model element/diagram
- Jump to a diagram or an element
- Add or manage bookmarks
- Grouping
- Shapes alignment and distribution

View menu

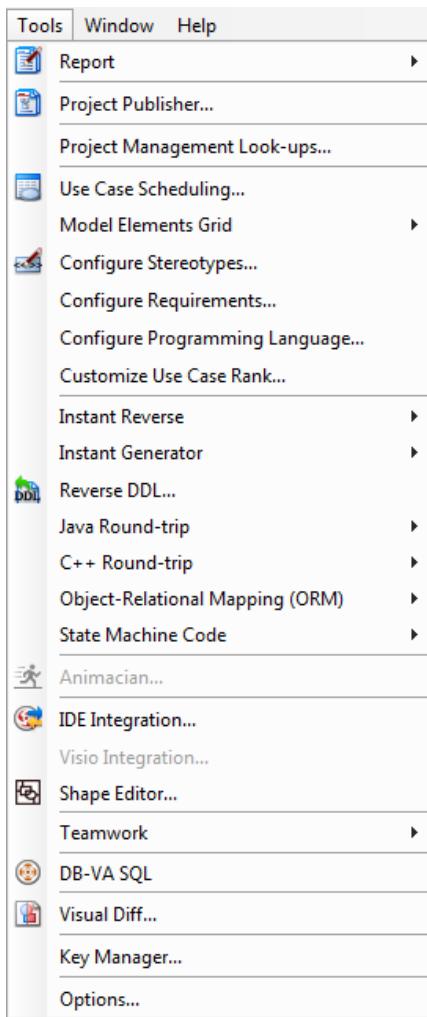


The **View** menu

The **View** menu enables you to:

- Show/Hide a pane
- Show and manage grid
- Manage Layers
- Change Resource Centric behavior
- Save, open and manage perspective
- Change VP-UML to show in full screen
- Zoom diagram in and out
- Manage nickname
- Manage style

Tools menu

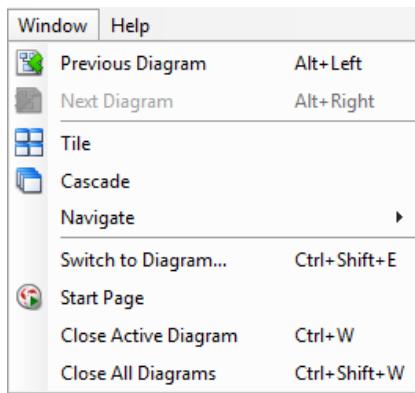


The **Tools** menu

The **Tools** menu enables you to:

- Generate report
- Publish project
- Perform use case scheduling
- Open various kind of element grid
- Configure stereotypes
- Configure requirements
- Configure programming language
- Customize use case ranks
- Reverse and Forward engineering with Instant Reverse and Instant Generator
- Perform round-trip engineering
- Reverse DDL
- Perform Object Relational Mapping (ORM)
- Perform State Machine Code Generation
- Launch Animacian
- Perform IDE integration
- Perform Visio integration
- Launch Shape Editor
- Perform Teamwork operations
- Launch DB-VA SQL
- Compare diagrams with Visual Diff
- Launch License Key Manager
- Configure application options through the Options dialog box

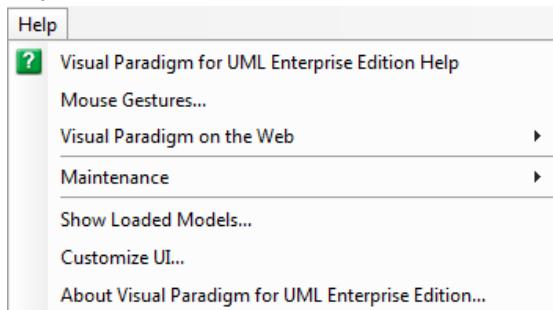
Window menu



The **Window** menu enables you to:

- Navigate between diagram
- Rearrange diagram windows
- Switch to another diagram
- Show the Start Page
- Close Diagrams

Help menu



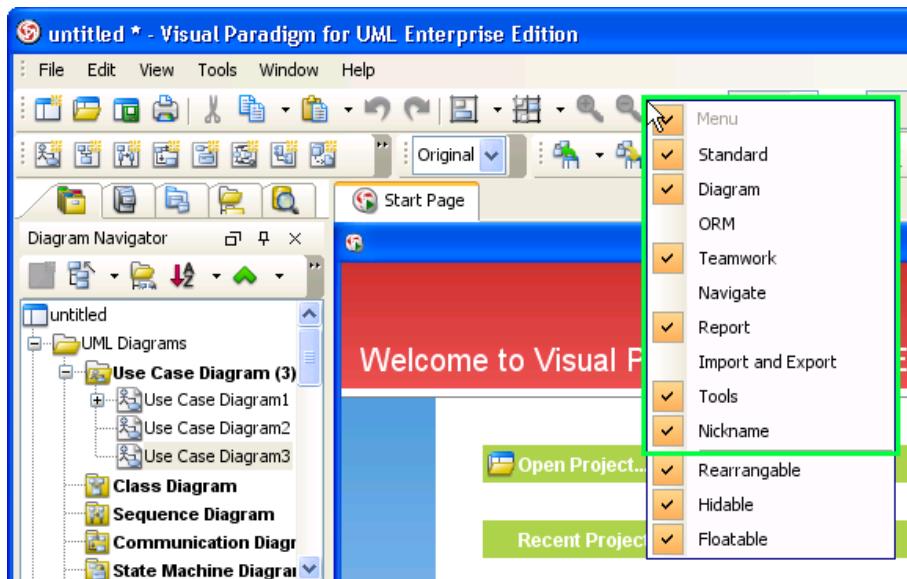
The **Help** menu enables you to:

- Browse the help contents
- Check the instruction of Mouse Gesture
- Visit Visual Paradigm online support
- Repair project
- Customize the user interface
- Check the environment using the About dialog box

Toolbar

Show/Hide toolbar(s)

A toolbar can be shown or hidden. To show a toolbar, right-click on any toolbar, and select the toolbar to show. Similarly, you can uncheck a toolbar to hide it.



Show or hide toolbar(s) by unchecking toolbar(s) from toolbars' popup menu

Reposition toolbars

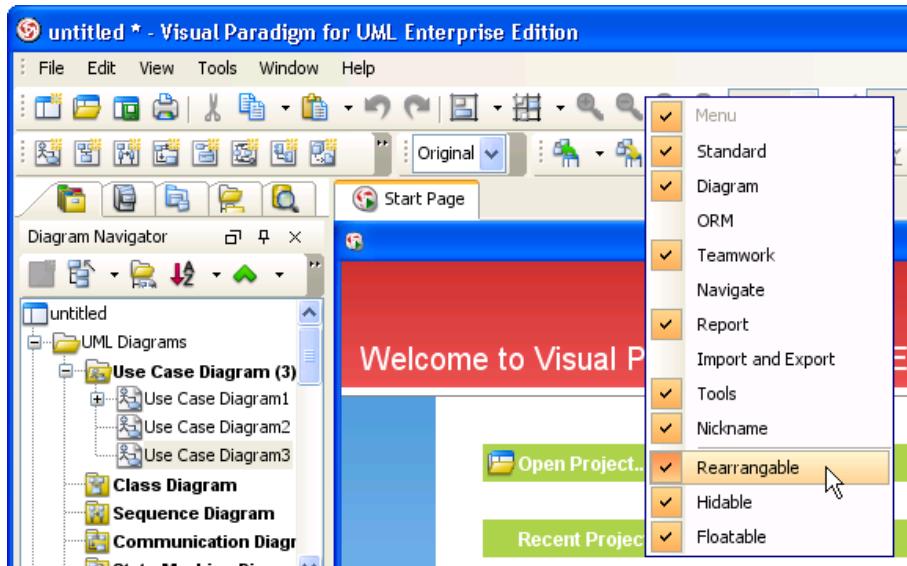
A toolbar can be repositioned by pressing on the handle of toolbar, which appear on the left hand side of a toolbar, and drag to target position.



Drag toolbar by pressing and moving the left of toolbar

Locking toolbar

To freeze toolbars' position and make the toolbars not movable, right click on any toolbar and uncheck **Rearrangable** from the popup menu.



Make toolbars not movable by deselecting the Rearrangable option in toolbars' popup menu

Standard toolbar



Icon	Name	Description
	New Project	Create a project
	Open Project	Open a project
	Save Project	Save the changes made in the opening project
	Print	Print diagram(s)
	Cut	Cut selected diagram elements
	Copy within VP-UML	Copy selected diagram elements ready to be used within VP-UML
	Copy to Clipboard as Image (JPG)	Copy selected diagram elements as JPG image
	Copy to Clipboard as Image (EMF)	Copy selected diagram elements as EMF image
	Copy to Clipboard as XML	Copy selected diagram elements as XML data
	Paste View	Paste copied diagram elements as view of original model element
	Paste Model Element	Paste copied diagram elements as a new model elements
	Undo	Roll back undesired changes
	Redo	Rerun an undone task
	Align Shapes	Open the Align Shapes dialog box for aligning shapes
	Align Top	Align selected shapes to the top of the range of selected shapes
	Align Bottom	Align selected shapes to the bottom of the range of selected shapes
	Align Left	Align selected shapes to the left of the range of selected shapes
	Align Right	Align selected shapes to the right of the range of selected shapes
	Align Horizontal Center	Align selected shapes to the center of the range of selected shapes
	Align Vertical Center	Align selected shapes to the vertical center of the range of selected shapes
	Same Width	Make selected shapes in same width, following the last selected shape
	Same Height	Make selected shapes in same height, following the last selected shape
	Same Width and Height	Make selected shapes in same width and height, following the last selected shape
	Distribute Shapes	Open the Distribute Shapes dialog box for distributing shapes
	Distribute Shapes Horizontally	Distribute selected shapes horizontally
	Distribute Shapes Vertically	Distribute selected shapes vertically
	Distribute Shapes by Left Edges	Distribute selected shapes by their left edges
	Distribute Shapes by Horizontal Centers	Distribute selected shapes by their horizontal centers
	Distribute Shapes by Right Edges	Distribute selected shapes by their right edges
	Distribute Shapes by Top Edges	Distribute selected shapes by their top edges

	Distribute Shapes by Vertical Centers	Distribute selected shapes by their vertical centers
	Distribute Shapes by Bottom Edges	Distribute selected shapes by their bottom edges
	Zoom In	Magnify the opening diagram
	Zoom Out	Diminish the opening diagram
	Zoom to 100% (Actual Size)	Reset zoom ratio back to 100%
	Zoom to Region	Zoom by selecting the range to view on diagram
<input type="button" value="100% ▾"/>	Zoom Ratio	Current zoom ratio. You can change it by selecting a ratio from the drop down menu or by typing a new one.
	Format Copier	Copy selected diagram element's format and apply on another shape
<input type="button" value="Dialog ▾"/>	Font Name	Name of font being used by the selected diagram element
<input type="button" value="11 ▾"/>	Font Size	Font size of the selected diagram element
B	Bold	Set the text of selected diagram element bold or not bold
<i>I</i>	Italic	Set the text of selected diagram element italic or not italic
F	Font Color	Select the font color of selected diagram element
	Auto Spell Check	Enable or disable automatic spell checking
	Help	Launch the Help Contents

Diagram toolbar



The Diagram toolbar

Icon	Name	Description
	New Use Case Diagram	Create a Use Case Diagram
	New Class Diagram	Create a Class Diagram
	New Sequence Diagram	Create a Sequence Diagram
	New Communication Diagram	Create a Communication Diagram
	New State Machine Diagram	Create a State Machine Diagram
	New Activity Diagram	Create an Activity Diagram
	New Component Diagram	Create a Component Diagram
	New Deployment Diagram	Create a Deployment Diagram
	New Package Diagram	Create a Package Diagram
	New Object Diagram	Create an Object Diagram
	New Composite Structure Diagram	Create a Composite Structure Diagram
	New Timing Diagram	Create a Timing Diagram
	New Interaction Overview Diagram	Create an Interaction Overview Diagram

 New Textual Analysis	Create a Textual Analysis
 New Requirement Diagram	Create a Requirement Diagram
 New Basic Diagram	Create a Basic Diagram
 New CRC Card Diagram	Create a CRC Card Diagram
 New Entity Relationship Diagram	Create an Entity Relationship Diagram
 New ORM Diagram	Create an ORM Diagram
 New Business Process Diagram	Create a Business Process Diagram
 New Data Flow Diagram	Create a Data Flow Diagram
 New EPC Diagram	Create a EPC Diagram
 New Process Map Diagram	Create a Process Map Diagram
 New Organization Chart	Create an Organization Chart
 New EJB Diagram	Create a EJB Diagram
 New Overview Diagram	Create an Overview Diagram
 New User Interface	Create a User Interface
 New Mind Mapping Diagram	Create a Mind Mapping Diagram
 New Matrix Diagram	Create a Matrix Diagram
 New WSDL Diagram	Create a WSDL Diagram

ORM toolbar



The ORM toolbar

Icon	Name	Description
 ORM Wizards	Open the ORM Wizards	
 Database Configuration	Open the Database Configuration dialog box to configure database connections	
 Reverse Database	Reverse engineering from database	
 Reverse Java Classes	Reverse engineering from Java Class	
 Reverse Hibernate	Reverse engineering from Hibernate mapping file	
 Reverse Enterprise Object Framework	Reverse engineering from Enterprise Object Framework	
 Synchronize to Class Diagram	Synchronize from Entity Relationship Diagram to Class Diagram	
 Synchronize to Entity Relationship Diagram	Synchronize from Class Diagram to Entity Relationship Diagram	
 Generate Database	Open the Database Code Generation dialog box to generate database	
 Generate ORM Code	Open the Database Code Generation dialog box to generate code	

Teamwork toolbar



The Teamwork toolbar

Icon	Name	Description
	Teamwork Client	Open the Teamwork Client dialog box
	Commit	Commit changes to server
	Update	Update changes from server
	Tag	Create a tag
	Branch	Create a branch
	Merge	Merge changes between trunk and branches
	Switch	Switch between trunk, branches and tags

Navigate toolbar



The Navigate toolbar

Icon	Name	Description
	Previous Diagram	Navigate to the previous diagram
	Next Diagram	Navigate to the next diagram
	Diagram Navigator	Open the Diagram Navigator pane
	Model Explorer	Open the Model Explorer pane
	Class Repository	Open the Class Repository pane
	Logical View	Open the Logical View pane
	ORM	Open the ORM pane
	Stencil	Open the Stencil pane
	Property	Open the Property pane
	Diagram Overview	Open the Diagram Overview pane
	Documentation	Open the Documentation pane
	Documentation	Open the Documentation pane

Report toolbar



The Report toolbar

Icon	Name	Description
	Generate HTML Report	Open the Generate HTML dialog box to generate HTML report
	Generate PDF Report	Open the Generate PDF dialog box to generate PDF report

 Generate Word Report	Open the Generate WORD dialog box to generate WORD report
 Report Writer	Open Report Writer
 Project Publisher	Publish Project to Web pages through Project Publisher

Import and export toolbar



The Import and Export toolbar

Icon	Name	Description
 Import VP-UML Project	Import a VP-UML project into the opening project	
 Import Rose	Import diagrams and model elements from a Rational Rose model	
 Import XMI	Import diagrams and model elements from XMI	
 Import XML	Import diagrams and model elements from XML	
 Import MS Word to Use Case Model Element	Import Use Case Report (MS Word) back into the opening project	
 Import Erwin Project (XML)	Import diagrams and model elements from Erwin Project	
 Import Telelogic Rhapsody Project	Import diagram and model elements from Telelogic Rhapsody Project	
 Import Telelogic System Architect	Import diagram and model elements from Telelogic System Architect	
 Import Rational Model	Import diagram and model elements from Rational Model	
 Import Rational DNX	Import diagram and model elements from Rational DNX	
 Import Excel	Import Excel file back into the opening project	
 Import Visio	Import Visio diagrams into VP-UML	
 Import NetBeans UML Project	Import NetBeans UML diagrams into VP-UML	
 Export VP-UML Project	Export the open project into a VP-UML project file	
 Export XMI	Export XMI from the opening project	
 Export XML	Export XML from the opening project	
 Export Active Diagram as Image	Export active diagram as image file	
 Export Diagram as Image	Export any diagram(s) as image file(s)	
 Export Selection as Image	Export selection on active diagram as image file	
 Export Use Case Model to MS Word	Export Use Case Report (MS Word) from Use Case diagrams and use case models	
 Export Active Diagram to Excel	Export active diagram into Excel report	
 Export Excel	Export any diagram(s) into Excel report(s)	

Tools toolbar



The Tools toolbar

Icon	Name	Description
	Instant Reverse	Reverse Engineering Class Diagram from many kinds of source code
	Instant Generator	Generate source code from the opening project
	Reverse DDL	Reverse data definition language file and form ERD
	Update to Code	Update source code from model
	Update to Model Element	Update model element from source code
	Generate State Machine Code	Generate state machine code from class diagram and state machine diagram
	Reverse State Machine Code	Reverse state machine code back into the opening project
	Animacian	Launch Animacian for animating the active diagram
	IDE Integration	Integrate with IDE(s) through IDE integration
	DB-VA SQL	Launch DB-VA SQL, a tool for querying data in database
	Visual Diff	Launch Visual Diff, a tool for comparing diagrams

Nickname toolbar



The Nickname toolbar

Manage and select the nickname to be applied on the opening project.

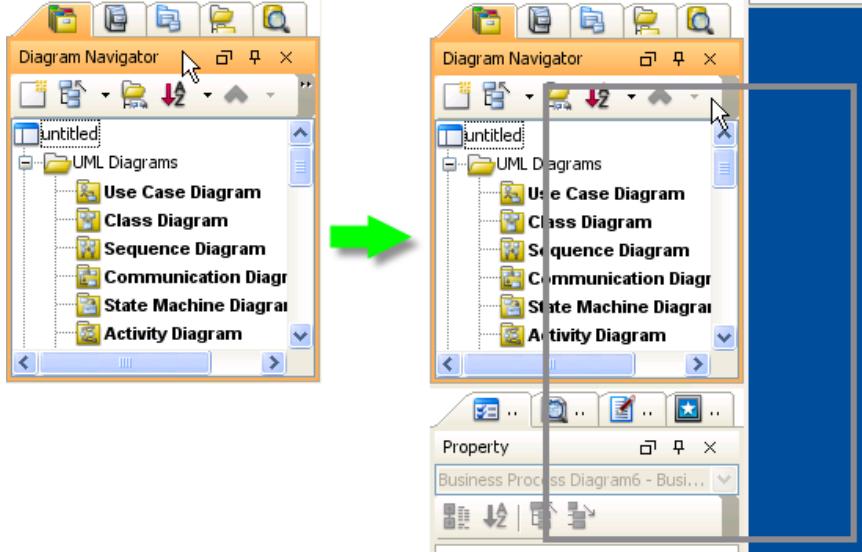
Dockable user interface

VP-UML adapts a Dockable User Interface which allows you to drag UI components around to customize your favorite working environment. You can save the environment as a perspective which you can reopen later. It allows you to use different perspectives for different purposes.

Using the dockable user interface

The Dockable User Interface is composed of a number of windows called dockable frames. A dockable frame may be standalone (floating) or docked into another container (split pane/tab pane).

You can press on the title bar of a dockable frame or press on a tab to drag it to anywhere you like.



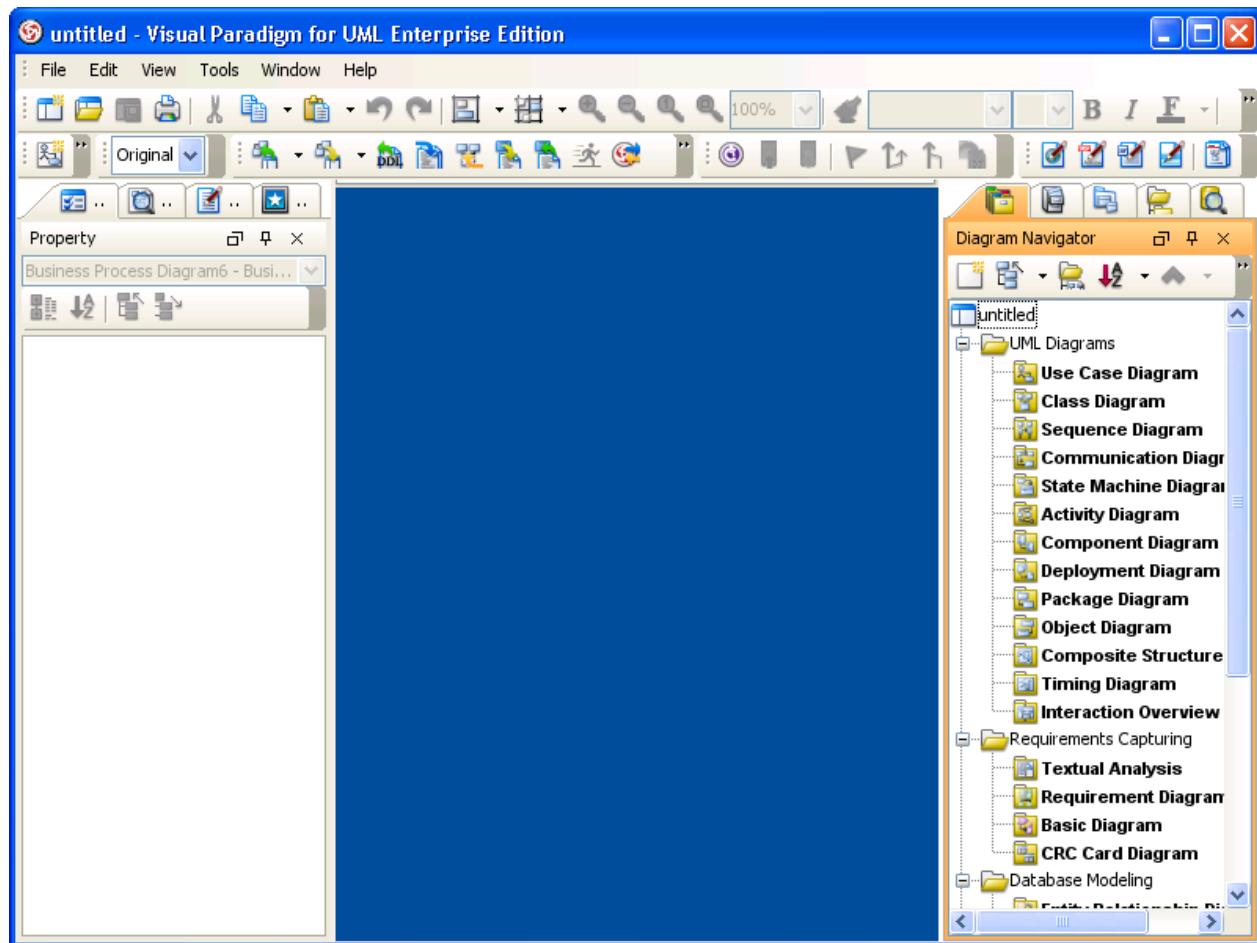
Frame can be dragged out and dock to elsewhere

You will notice a gray outline appears while you are dragging a frame/tab. This outline tells you where the dockable frame/tab will be docked to.

Docking a dockable frame to elsewhere

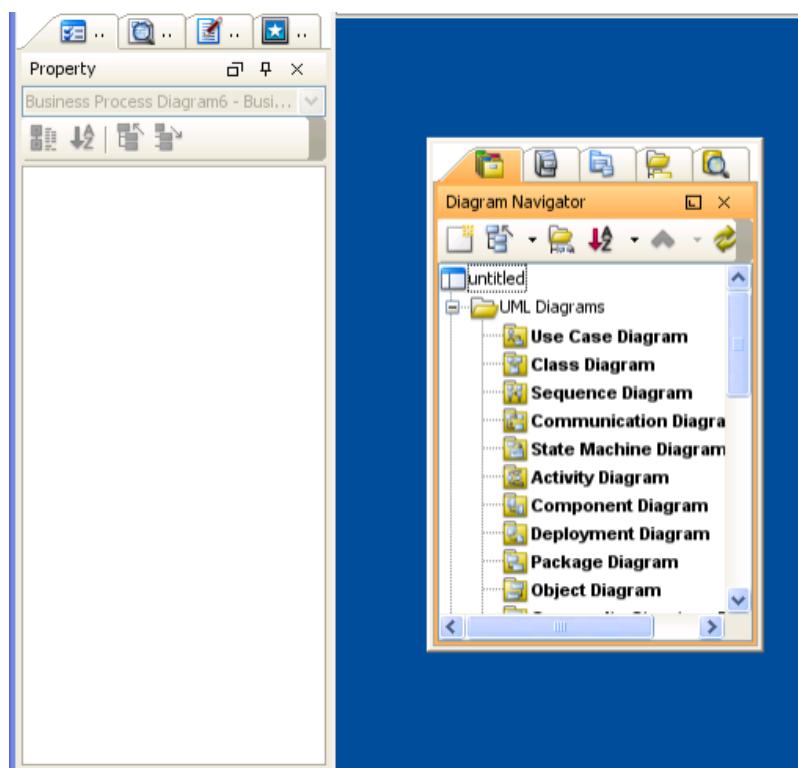
By dragging out a dockable frame/tab, and move the cursor to certain position, the frame/tab will be repositioned accordingly.

If you drag the dockable frame/tab and release it over another container, the gray outline will change its shape to fit the dockable area of the container. If you release the frame/tab, it will be docked into the underlying container and also removed from its original container.



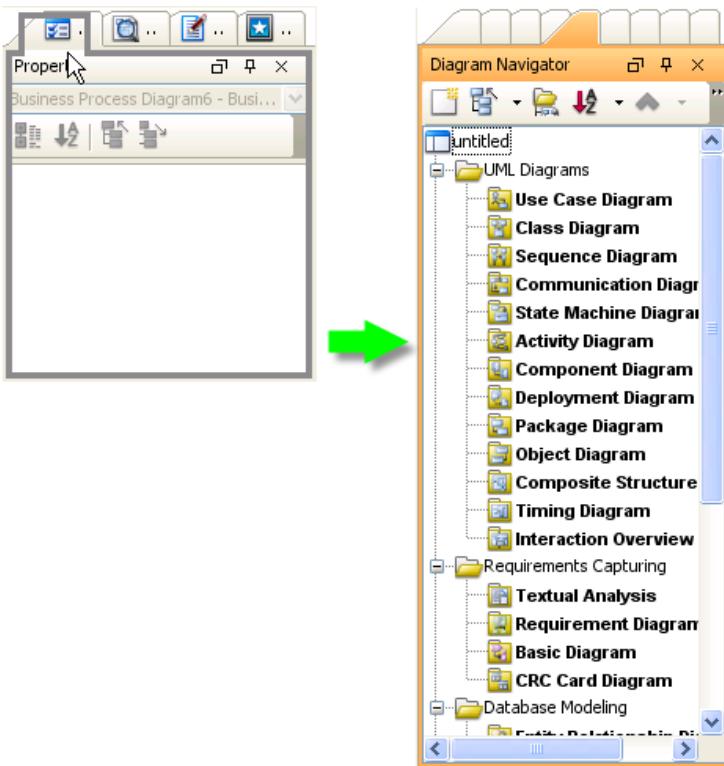
A frame docked to the right of application screen

You also can drag a frame/tab out to make it a floating window.



A frame float on top of the application

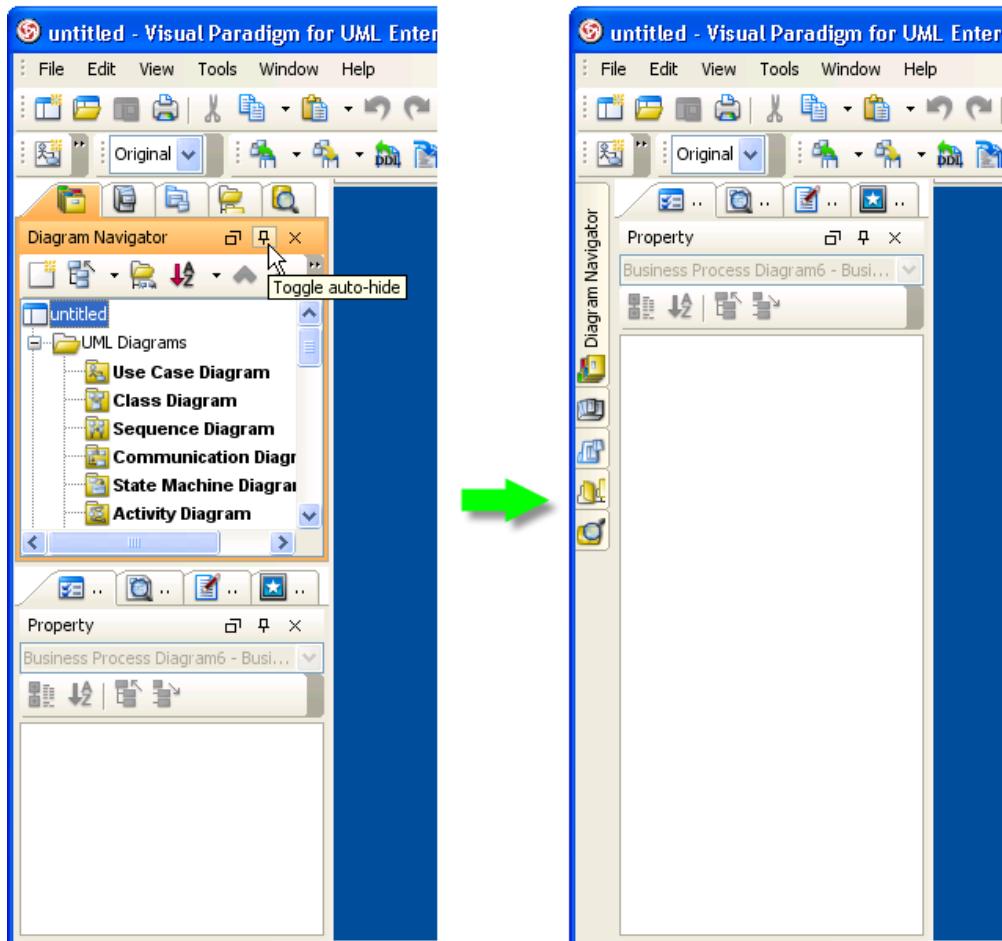
You can also drag a frame/tab and dock it into another tabbed pane. You will see the outline changed to a tab shape if you drag over a tab pane.



A frame docked to another frame

Auto-hiding a dockable frame

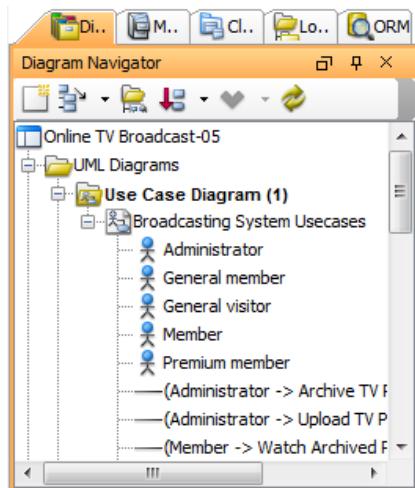
A dockable frame can be set to "auto hide" , meaning it will automatically disappear when not active. To set a dockable frame to "auto hide" , click on the **Toggle auto-hide** button on the upper right corner of the frame (the button with a pin as the icon, see figure below)



Automatically hiding Diagram Navigator

Diagram navigator

Diagram navigator is the location where diagrams are listed. You can create your own diagrams easily as they are listed by categories, such as UML Diagrams, Requirements Capturing, Database Modeling, Business Process Modeling and Others. With diagram navigator, you can create, open and delete diagrams.



The **Diagram Navigator**

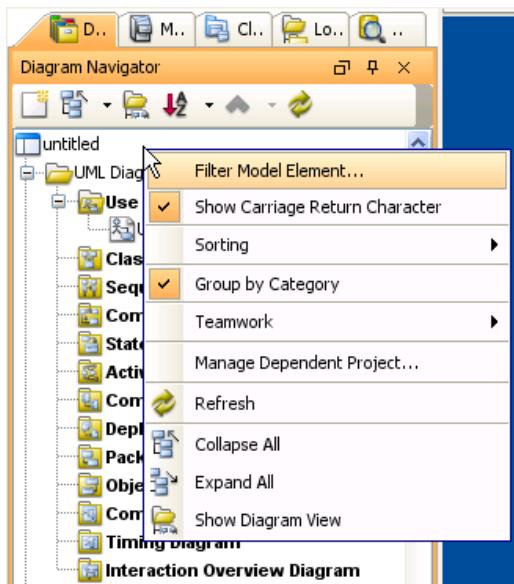
The toolbar

Name	Icon	Description
New diagram		To create a new diagram through the New Diagram dialog box.
Collapse		To collapse the selected diagram.
Expand		To expand the selected diagram.
Show Diagram View		To overview the diagram elements in the selected diagram.
Sort Diagram Element By Name		To sort diagram elements by their name in alphabetical order.
Sort Diagram Element By Type		To sort diagram elements by their type, disregard their name.
Move Selected Diagram		To move the selected diagram down or up. This option only works within the same diagram type when there are multiple diagrams.
Refresh		To update the content of diagram tree.

The description of icons on **Diagram Navigator**

Popup menu

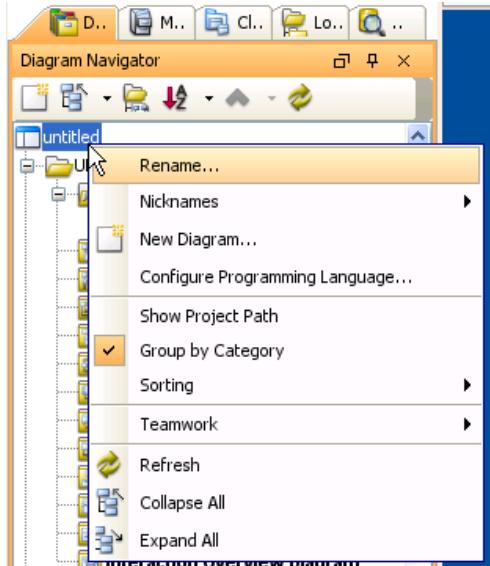
Popup menu of diagram navigator



The Popup menu of Diagram Navigator

Menu Title	Description
Filter Model Element...	Open the Model Element Filter dialog box to filter the diagram elements to appear in the Diagram Navigator
Show Carriage Return Character	Display line breaks of multi-lined diagram name as carriage return character
Sorting	Select the way to sort diagram elements in Diagram Navigator
Group by Category	Categorize diagrams base on their types
Teamwork	Perform teamwork activities
Manage Dependent Project...	Add or remove dependent project
Refresh	Refresh Diagram Navigator content
Collapse All	Collapse all tree nodes
Expand All	Expand all tree nodes
Show Diagram View	Make the Diagram Navigator to expand all diagram types' nodes to show the diagrams nodes, but do not display any diagram element nodes

Popup menu of project

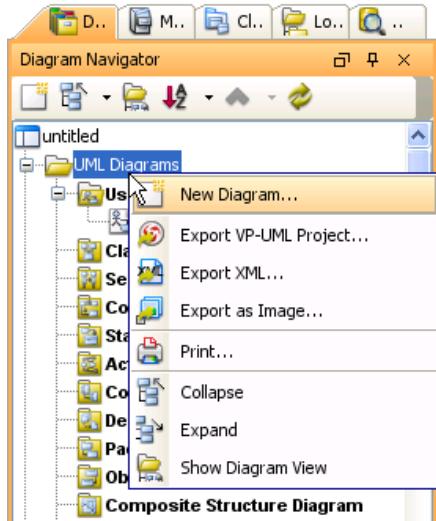


The Popup menu of project node in Diagram Navigator

Menu Title	Description

Rename...	Rename the project
Nicknames	Configure or switch to another nickname
New Diagram...	Create a diagram
Configure Programming Language	Change to another programming language or configure the type mapping for a language
Group by Category	Categorize diagrams base on their types
Sorting	Select the way to sort diagram elements in Diagram Explorer
Teamwork	Perform teamwork activities
Refresh	Refresh Diagram Navigator
Collapse All	Collapse the project node
Expand All	Expand the project node

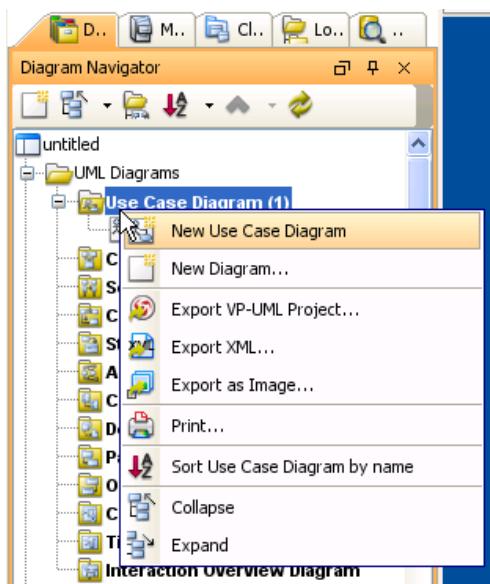
Popup menu of diagram category



The Popup menu of Diagram Category in Diagram Navigator

Menu Title	Description
New Diagram...	Create a diagram
Export VP-UML Project...	Export all diagrams in the selected category as VP-UML project
Export XML...	Export all diagrams in the selected category as XML
Export as Image...	Export all diagrams in the selected category as image files
Print...	Print the diagrams in the selected category
Collapse	Collapse the selected diagram category node
Expand	Expand the selected diagram category node
Show Diagram View	Make the Diagram Navigator to expand all diagram types' nodes to show the diagrams nodes, but do not display any diagram element nodes

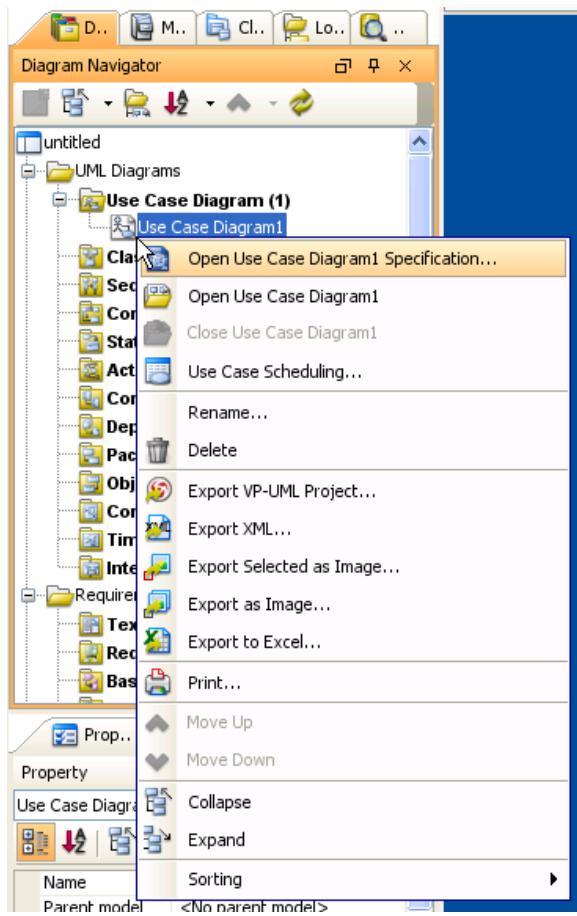
Popup menu of diagram type



The Popup menu of Diagram Type in Diagram Navigator

Menu Title	Description
New DIAGRAM_TYPE	Create a new diagram in the selected type
New Diagram...	Create a new diagram with popup dialog box
Export VP-UML Project...	Export all diagrams in the selected type as VP-UML project
Export XML...	Export all diagrams in the selected type as XML
Export as Image...	Export all diagrams in the selected type as image files
Print...	Print the diagrams in the selected type
Sort DIAGRAM_TYPE by name	Sort the diagram nodes of the selected type node in specific way
Collapse	Collapse the selected diagram type node
Expand	Expand the selected diagram type node

Popup menu of diagram



The Popup menu of Diagram Type in Diagram Navigator

Menu Title	Description
Open DIAGRAM_NAME Specification...	Open the specification of the selected diagram
Open DIAGRAM_NAME	Open the selected diagram if closed or inactive
Close DIAGRAM_NAME	Close the selected diagram if opened
Use Case Scheduling...	Open Use Case Scheduling dialog box
Rename...	Rename the selected diagram
Delete	Delete the selected diagram
Export VP-UML Project...	Export the selected diagram as VP-UML project
Export XML...	Export the selected diagram as XML
Export Selected as Image...	Export the selected diagram as image Excel
Export as Image...	Export diagram(s) as image via the Diagram Exporter
Export to Excel...	Export the selected diagram as image Excel
Print...	Print the selected diagram
Move Up	Move the selected diagram node upwards
Move Down	Move the selected diagram node downwards
Collapse	Collapse the selected diagram node
Expand	Expand the selected diagram node
Sorting	Change the way of sorting diagram elements

Close and open the diagram navigator

Diagram navigator 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 > Diagram Navigator** from the main menu.

Create a diagram

Approach 1 – Direct creation

If you want to create a diagram in a quick way and don't need to supply the documentation of diagram (for time-saving), use this approach. To create a diagram:

1. Right click on the diagram type that you want to create.
2. Select **New diagram type** from the pop-up menu. Here, diagram type means the type of diagram you want to create.

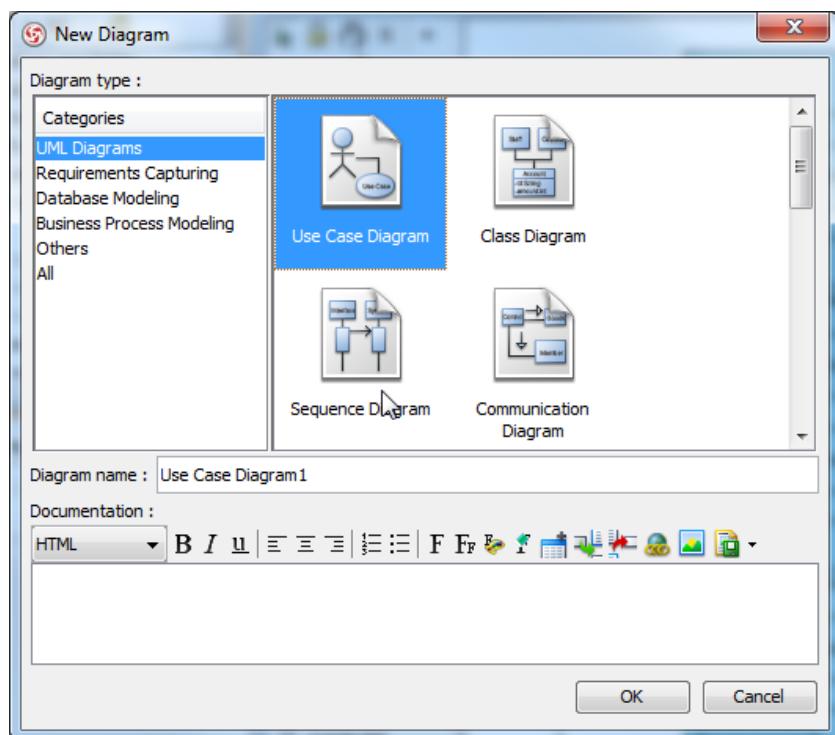
NOTE: You can immediately enter the diagram name at the top left corner of diagram

Approach 2 – Through the New Diagram dialog box

This approach let you create a diagram and enter the name and documentation simultaneously. To create a diagram:

1. Right click on the diagram type that you want to create.
2. Select **New Diagram** from the pop-up menu.

In the **New Diagram** dialog box, you can enter the name and documentation of diagram, and click **OK** to proceed.



New Diagram dialog box

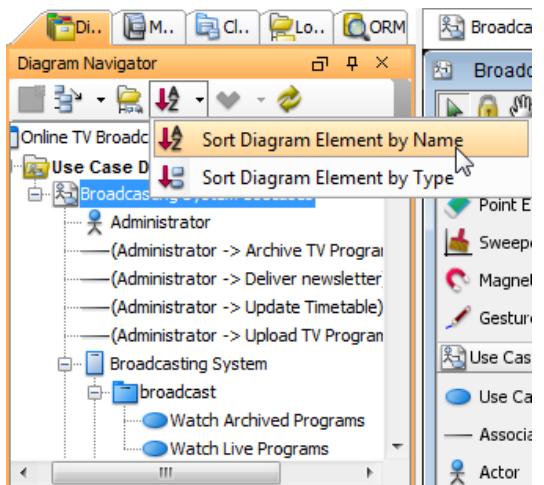
Open a diagram

Double click in the diagram tab you want to view in the diagram tab.

Sort diagram elements

In diagram navigator, diagram elements are listed under diagram nodes. You can sort diagram elements by their names, or by their types (e.g. use case, package, etc.)

To sort by name, click **Sort Diagram Element by Name** button. The elements will be listed by name, in alphabetical order.



Click **Sort Diagram Element by Name**

To sort by type, click **Sort Diagram Element by Type** button. As a result, the elements will be listed by type.

NOTE: The sort function applies to the entire diagram tree instead of the selected node.

Reorder diagrams

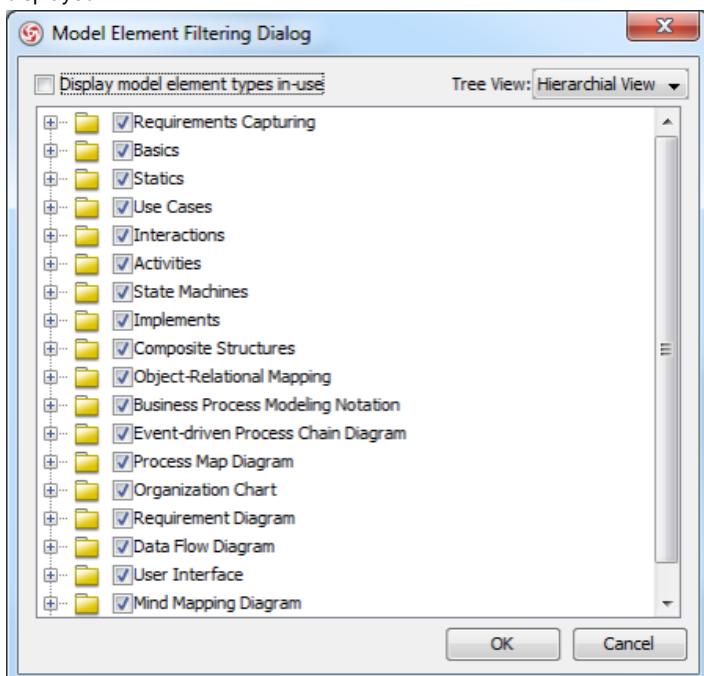
Select the diagram(s) you want to reorder. If there are multiple diagrams you want to select, just click one while pressing **Ctrl** button and make the other selection.

Click **Move Selected Diagram Up** button to move the selection upwards, or **Move Selected Diagram Down** button to move the selection downwards.

Filter model elements

You can choose which model elements you want to be displayed or not on the diagram tree.

1. Right click on the diagram navigator's background and select **Filter Model Element**. **Model Element Filtering Dialog** will then be pop-up.
2. Select the types of model element(s) you want to be displayed by checking to the type. Otherwise, uncheck the one you don't want to be displayed.



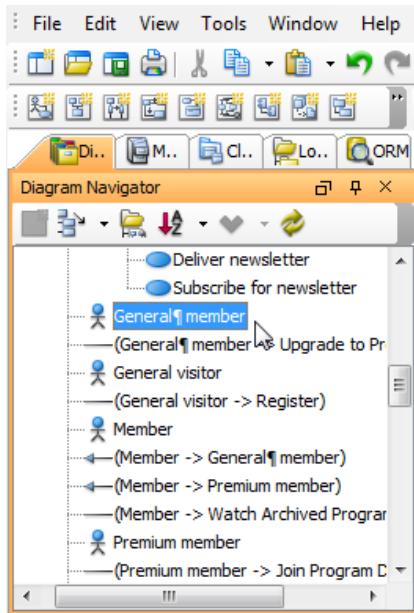
Model Element Filtering dialog box

NOTE: You can check **Display model element types in-use** to list only 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)

Show/hide carriage return character

If it is the case that the name of the diagram and diagram elements 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.

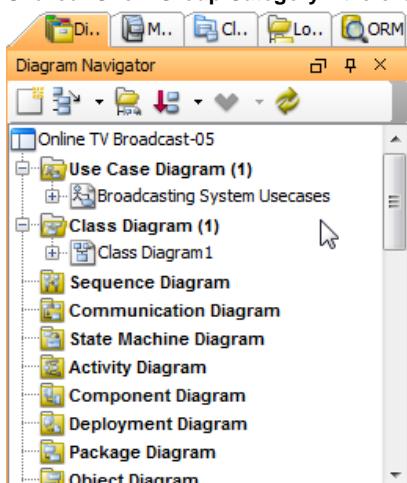
If you want to hide it, uncheck **Show Carriage Return Character** and the character will automatically be unshown.

Group or ungroup by category

As it is said before, default groups are automatically arranged by category

If you want to ungroup diagram:

1. Right click on the diagram navigator's background.
2. Uncheck **Show Group Category** if there is a tick in the box, and then all diagram types will be ungrouped.



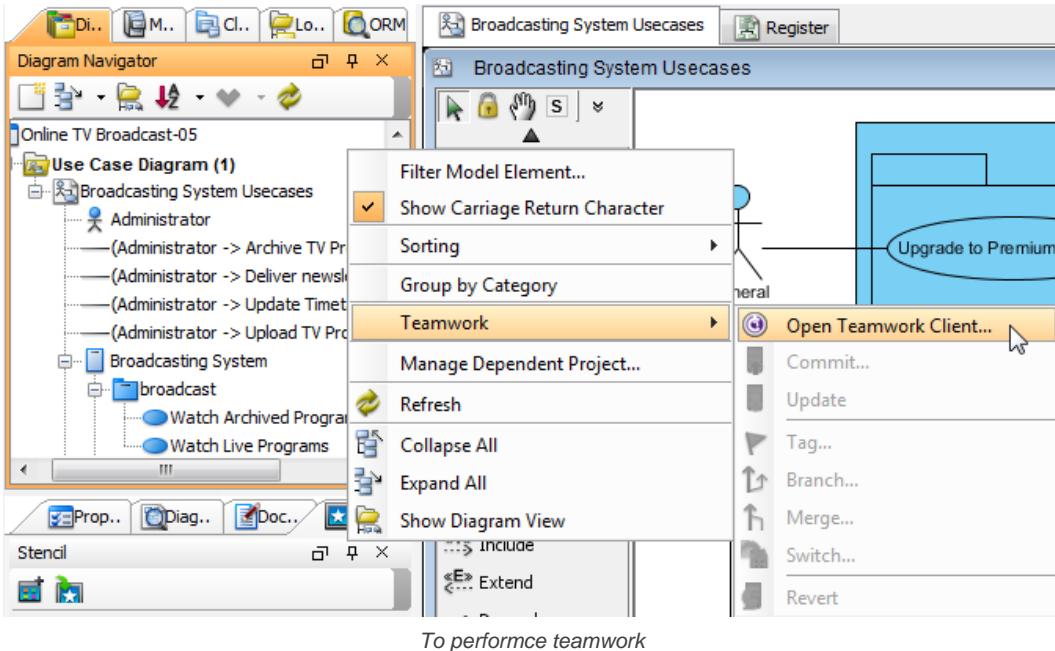
Ungrouped diagrams

Connect to server for team collaboration

VP-UML'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 diagram background.

2. Select **Teamwork** and the action you want to perform (e.g. commit and update, etc) from the pop-up menu.



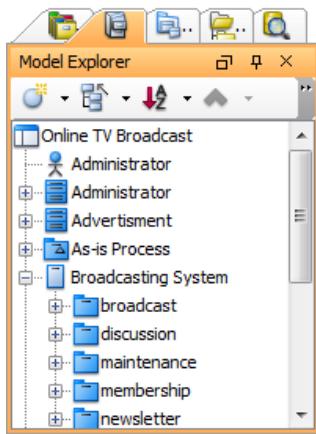
Display only diagram nodes but not diagram elements

If you only want the diagram nodes to be shown but not the diagram elements under them, just click on **Show Diagram View** button.

NOTE: This option applies to all diagram nodes, but not just the selected one.

Model Explorer

It would be much more efficient to make use of Model Explorer for your middle to large scale project which has considerable numbers of diagrams and model elements, rather than using Diagram Navigator.



The *Model Explorer*

The toolbar

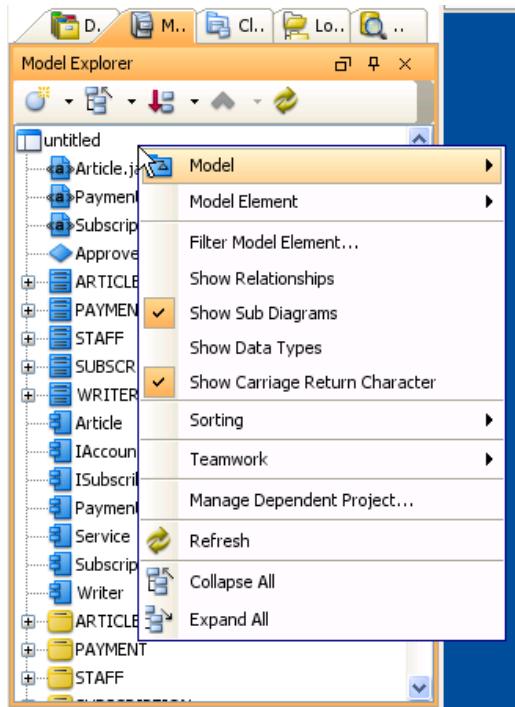
Name	Icon	Description
New Model Element		To create a new model element.
Collapse		To collapse the selected model element.
Expand		To expand the selected model element.
No Sorting		To arrange model elements without grouping.
Sort By Name		To sort model elements by their names in alphabetical order.
Sort By Type		To sort model elements by their types.
Move Selected Model Element		To move the selected model elements down or up. This option only works within the same model element type when there are multiple model elements.
Refresh		To update the content of model explorer

The description of icons on *Model Explorer*

Popup menu

Popup menu of model explorer

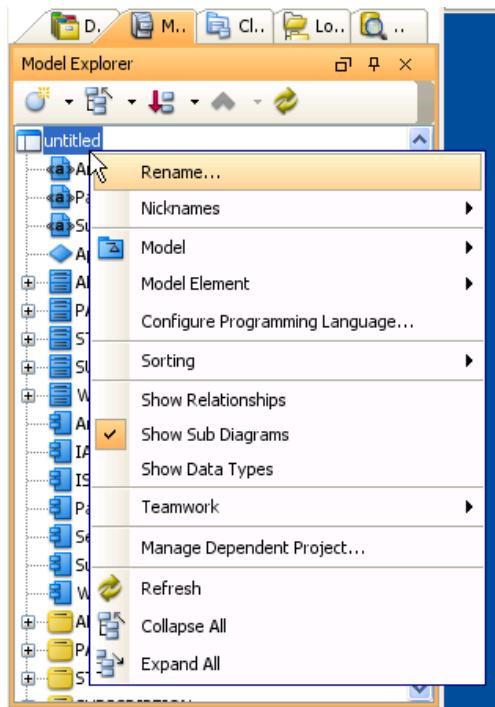
The Model Explorer lists all the model elements in the project.



The Popup menu of Model Explorer

Menu Title	Description
Model	Create a Model
New Model Element...	Create a new Model Element in Model Explorer without the need of creating through diagramming
Filter Model Element...	Open the Model Element Filter dialog box to filter the diagram elements to appear in the Diagram Navigator
Show Relationships	Show also Relationship model elements in Model Explorer (default hidden)
Show Sub Diagrams	Show Sub Diagrams in Model Explorer so that user can browse and open Sub Diagrams in Model Explorer
Show Data Types	Show Data Types in Model Explorer (default hidden)
Show Carriage Return Character	Display line breaks of multi-lined diagram name as carriage return character
Sorting	Select the way to sort model elements in Model Explorer
Teamwork	Perform teamwork activities
Manage Dependent Project...	Add or remove dependent project
Refresh	Refresh Model Explorer content
Collapse All	Collapse all tree nodes
Expand All	Expand all tree nodes

Popup menu of project



The Popup menu of project node in Model Explorer

Menu Title	Description
Rename...	Rename the project
Nicknames	Configure or switch to another nickname
Model	Create a Model
Model Element	Create a new Model Element in Model Explorer without the need of creating through diagramming
Configure Programming Language	Change to another programming language or configure the type mapping for a language
Sorting	Select the way to sort model elements in Model Explorer
Show Relationships	Show also Relationship model elements in Model Explorer (default hidden)
Show Sub Diagrams	Show Sub Diagrams in Model Explorer so that user can browse and open Sub Diagrams in Model Explorer
Show Data Types	Show Data Types in Model Explorer (default hidden)
Teamwork	Perform teamwork activities
Manage Dependent Project...	Add or remove dependent project
Refresh	Refresh Model Explorer
Collapse All	Collapse the project node
Expand All	Expand the project node

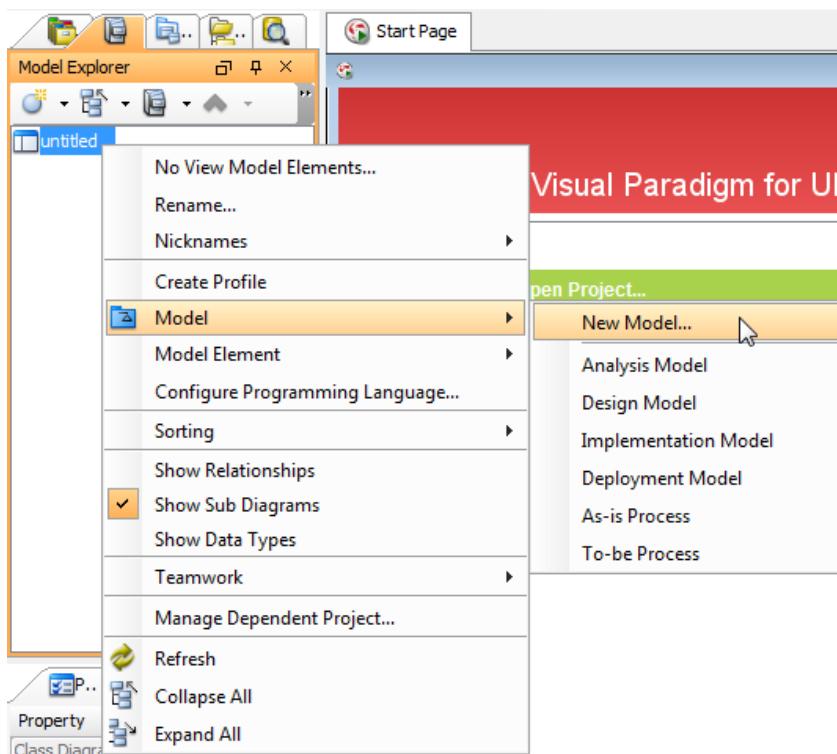
Closing and opening the model explorer

Model Explorer 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 > Model Explorer** from the main menu.

Creating a model

A model is a package like UML element that can store model elements and diagrams. Users are recommended to structure project by using model in order to maintain a clear structure for accessing project data and improve the application performance.

Right click on the root node in **Model Explorer** and select **Model** from the popup menu. You can either create a custom model by selecting **New Model...**, or create a pre-defined model by selecting it in the list.

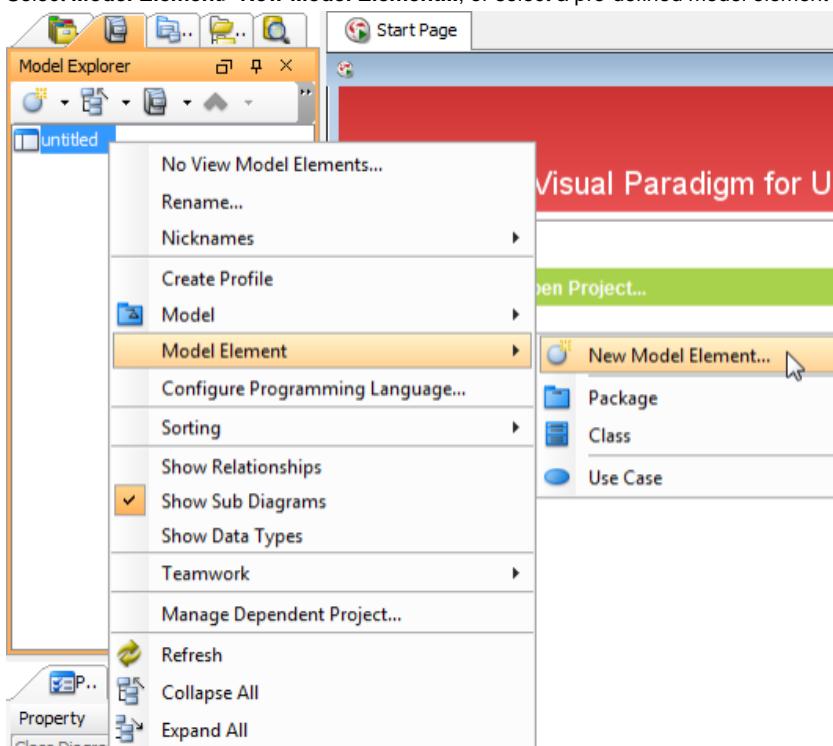


Selecting **New Model** in pop-up menu

Creating a model element

A model element is created when you create a shape on a diagram. If you want to create a model element without visualizing it, you can create it through the **Model Explorer**. To create a model element:

1. Right click on the root node.
2. Select **Model Element**>**New Model Element...**, or select a pre-defined model element from the pop-up menu.

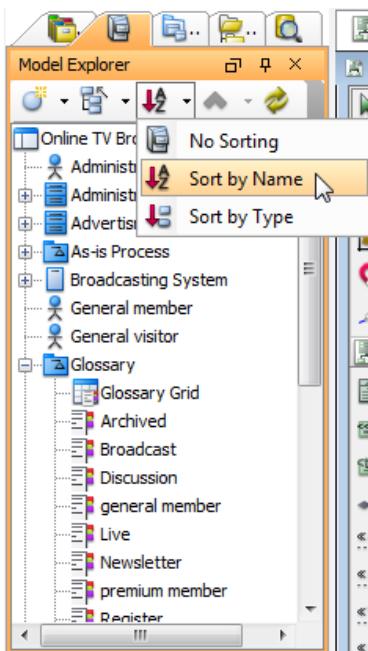


Selecting **New Model Element** in pop-up menu

Sorting model elements

In model explorer, model elements are listed under root nodes. You can sort model elements by their names, by their types (e.g. use case, package, etc.) or with no sorting.

To sort by name, click **Sort by Name** button. The elements will be listed by name, in alphabetical order.



Sort model elements by name

To sort by type, click **Sort by Type** button. As a result, the elements will be listed by type. To arrange model elements without sorting, click **No sorting**.

Reordering model elements

Select the model element(s) you want to reorder. If there are multiple model elements you want to select, just click one while pressing **Ctrl** button and make the other selections.

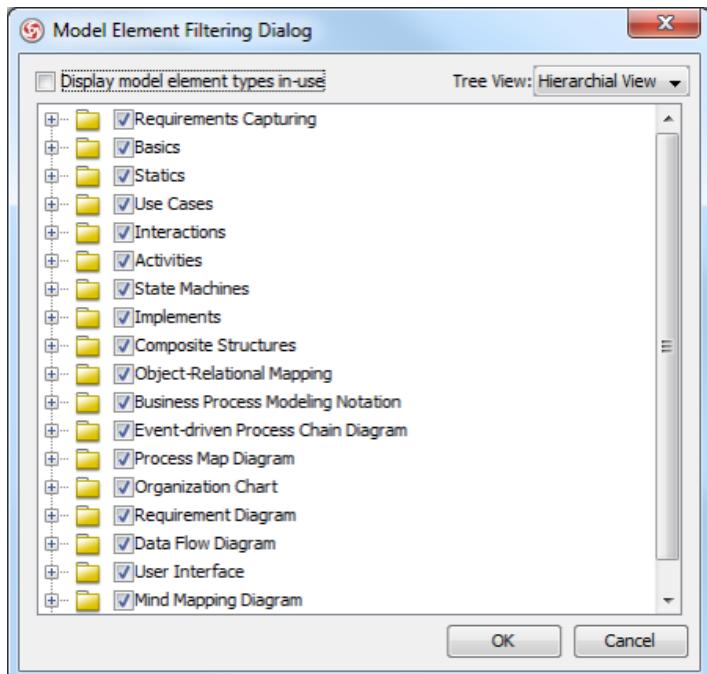
Click **Move Selected Model Element Up** button to move the selection upwards, or **Move Selected Model Element Down** button to move the selection downwards.

NOTE: This option only works within the same model element type when there are multiple model elements.

Filtering model elements

You can choose which model elements you want to be displayed or not on the model explorer.

Right click on the model explorer's background and select **Filter Model Element**. In **Model Element Filtering Dialog**, select the types of model element(s) you want to be displayed by checking the type. Otherwise, uncheck the one you don't want to be displayed.



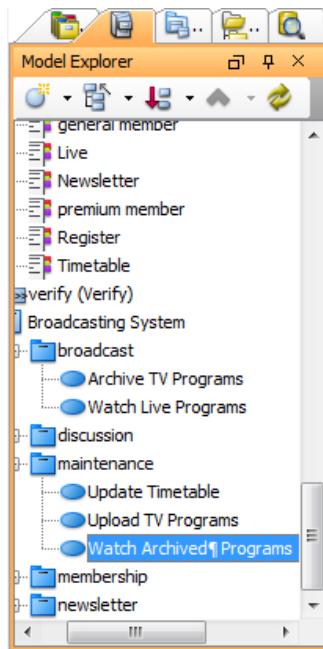
Model Element Filtering Dialog

NOTE: Instead of displaying all types of model elements, checking **Display model element types in-use** to display the types of model elements used in the project.

Showing/hiding carriage return character

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

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



Show Carriage Return Character is selected

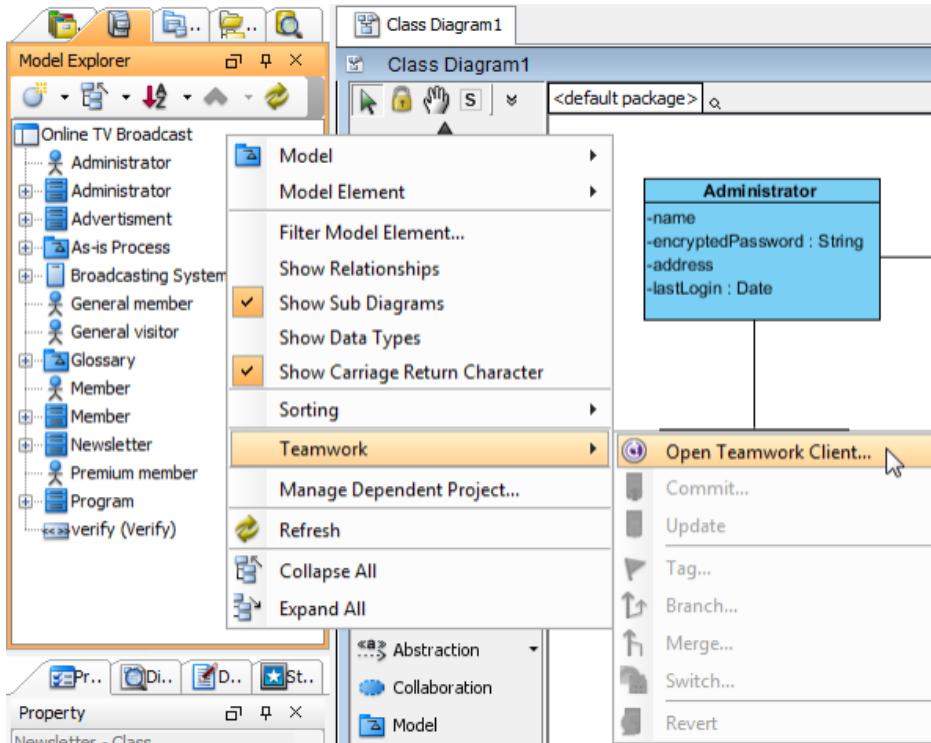
When off, the character ¶ is hidden.

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

Connecting to server for team collaboration

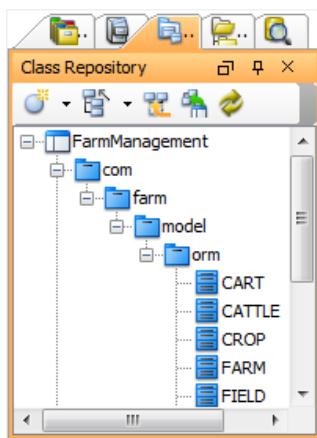
VP-UML's team collaboration support enables your team members work together on projects. To connect to server and perform related activities:

1. Right click on the model explorer's background.
2. Select Teamwork> Open Teamwork Client... from the pop-up menu.



Class Repository

Class repository is a pane where classes and container that contain classes, such as packages or subsystems, are listed. Further to accessing classes, you can also form class diagram by dragging classes from class repository to class diagram.



The Class Repository

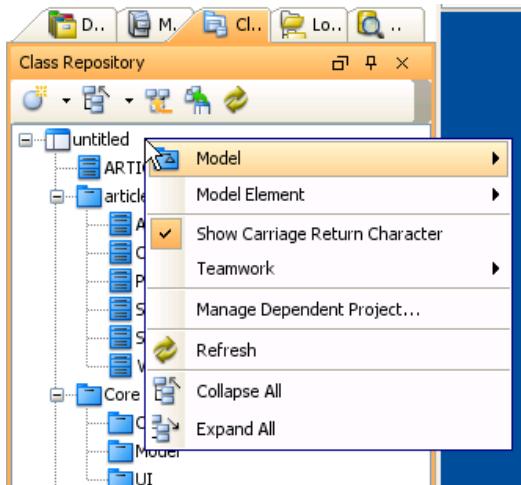
The toolbar

Name	Icon	Description
New Model Element		To create a new model element.
Collapse		To collapse the selected model element.
Expand		To expand the selected model element.
Reverse Code...		To reverse a code as class model for the whole project.
Instant Reverse...		To reverse different types of source into UML class models.
Refresh		To update the content of Class Repository

The description of icons on Class Repository

Popup menu

Popup menu of class repository

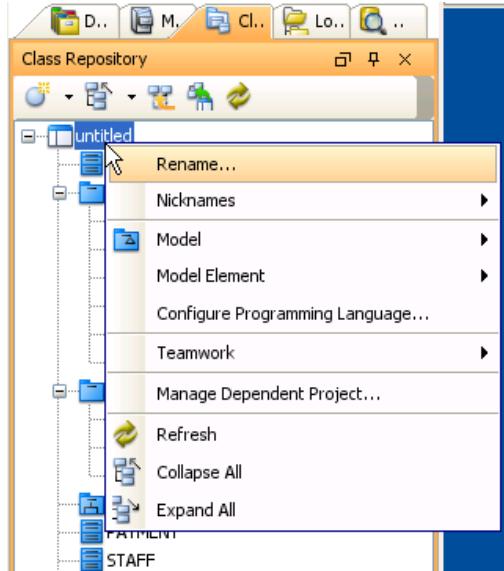


The Popup menu of Class Repository

Menu Title	Description
Model	Create a Model
Model Element...	Create a new Model Element in Class Repository without the need of creating through diagramming

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 Class Repository content
Collapse All	Collapse all tree nodes
Expand All	Expand all tree nodes

Popup menu of project



The Popup menu of project node in Class Repository

Menu Title	Description
Rename...	Rename the project
Nicknames	Configure or switch to another nickname
Model	Create a Model
Model Element	Create a new Model Element in Class Repository without the need of creating through diagramming
Configure Programming Language	Change to another programming language or configure the type mapping for a language
Teamwork	Perform teamwork activities
Manage Dependent Project...	Add or remove dependent project
Refresh	Refresh Class Repository
Collapse All	Collapse the project node
Expand All	Expand the project node

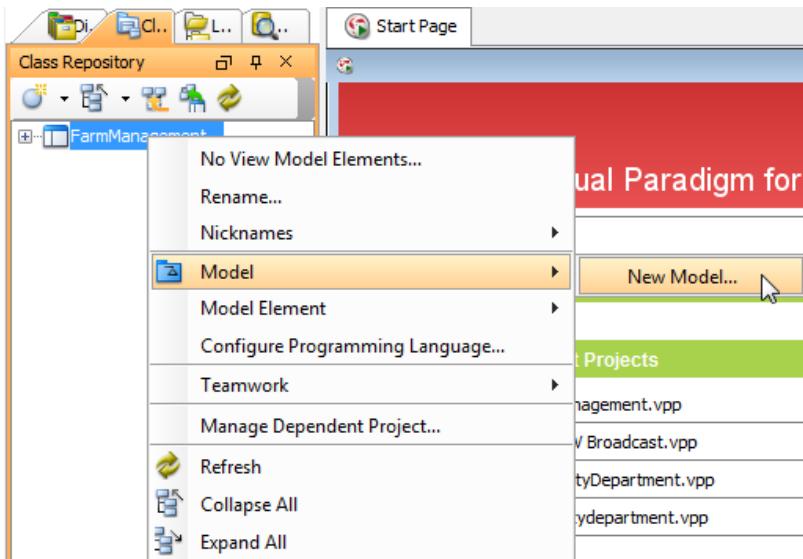
Closing and opening the Class Repository

Class Repository 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 > Class Repository** from the main menu.

Creating a model

A model is a package like UML element that can store model elements and diagrams. Users are recommended to structure project by using model in order to maintain a clear structure for accessing project data and improve the application performance.

Right click on the root node in Class Repository and select **Model** from the popup menu. You can either create a custom model by selecting **New Model....**

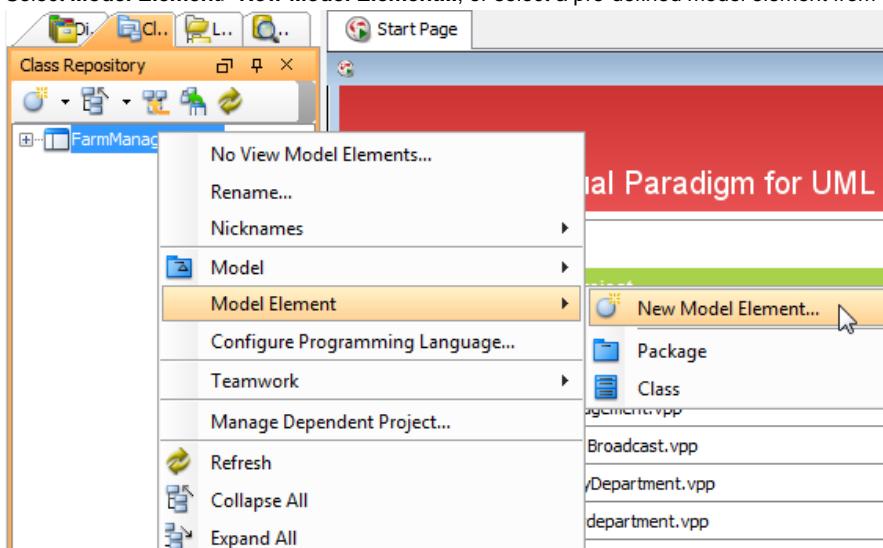


Selecting **New Model** in pop-up menu

Creating a model element

A model element is created when you create a shape on a diagram. If you want to create a model element without visualizing it, you can create it through the Class Repository. To create a model element:

1. Right click on the root node.
2. Select **Model Element> New Model Element...**, or select a pre-defined model element from the pop-up menu.

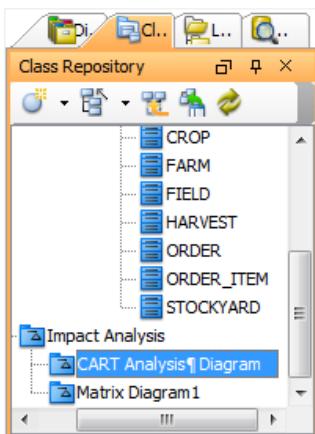


Selecting **New Model Element** in pop-up menu

Showing/hiding carriage return character

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

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



Show Carriage Return Character is revealed

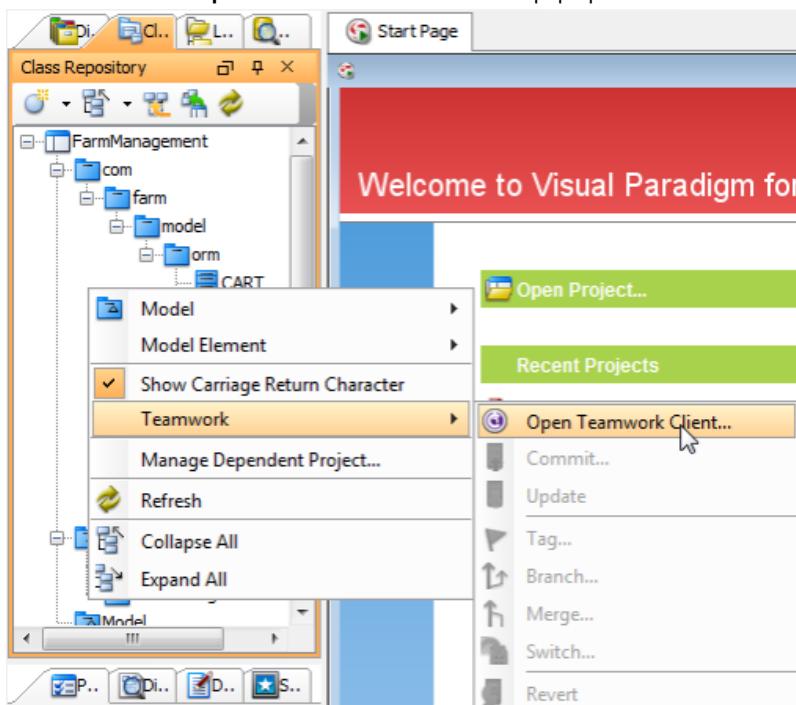
When off, the character ¶ is hidden.

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

Connecting to server for team collaboration

VP-UML'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 Class Repository's background.
2. Select Teamwork> Open Teamwork Client... from the pop-up menu.

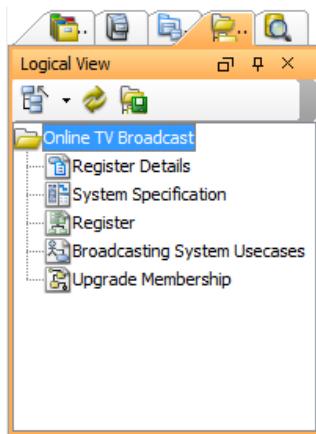


To perform teamwork

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

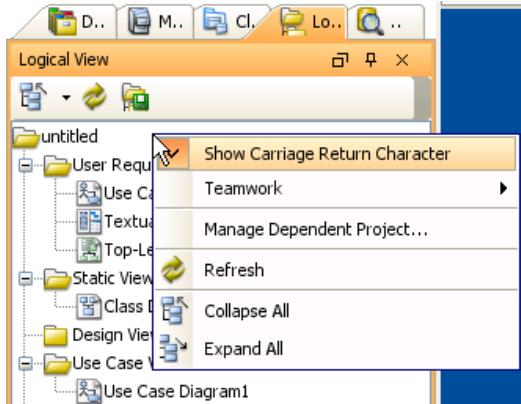
The toolbar

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

Popup menu

Popup menu of logical view

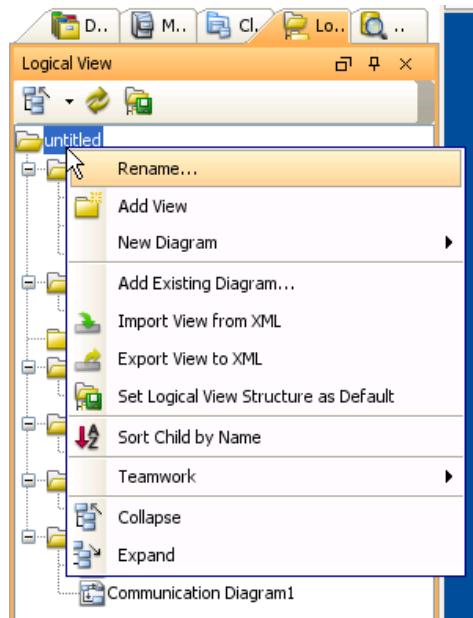


The Popup menu of Logical View

Menu Title	Description
Show Carriage Return Character	Display line breaks of multi-lined diagram name as carriage return character
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](#)

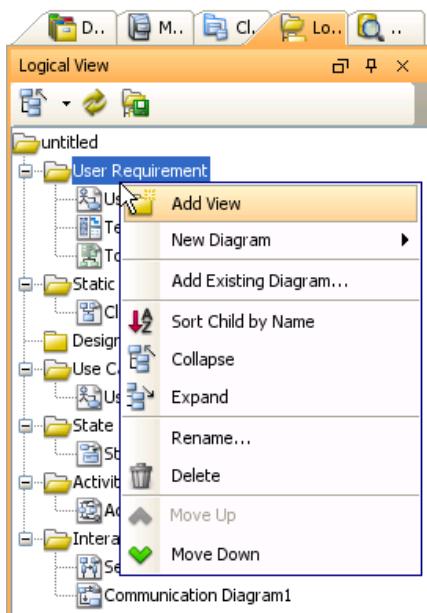
Popup menu of project



The Popup 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

Popup menu of view



The Popup menu of view in Logical View

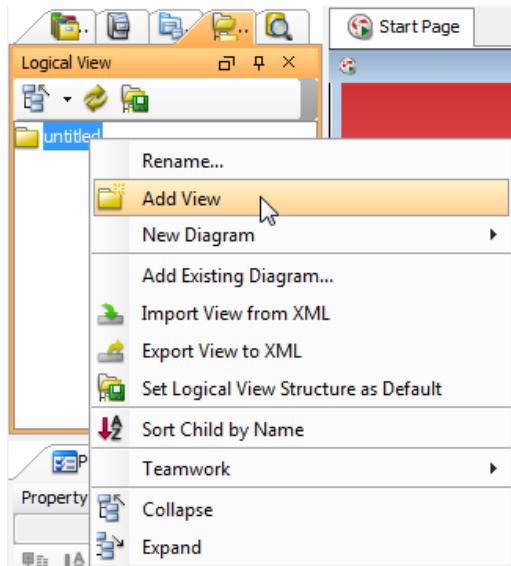
Menu Title	Description
Add View	Add a child view under the selected view
New Diagram	Create a diagram under the selected view
Add Existing Diagram...	Add an existing diagram under the selected view
Sort Child by Name	Sort the views/diagrams by name
Collapse	Collapse the selected view node
Expand	Expand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

Close and open 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.

Create a new view node

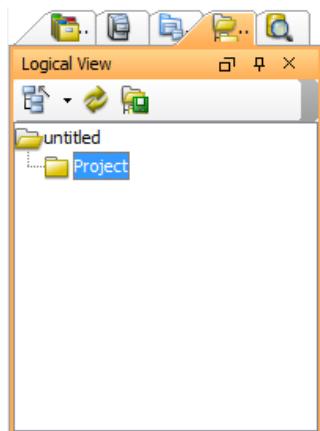
Right-click a root node on **Logical View** and select **Add View** from the pop-up menu.



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.

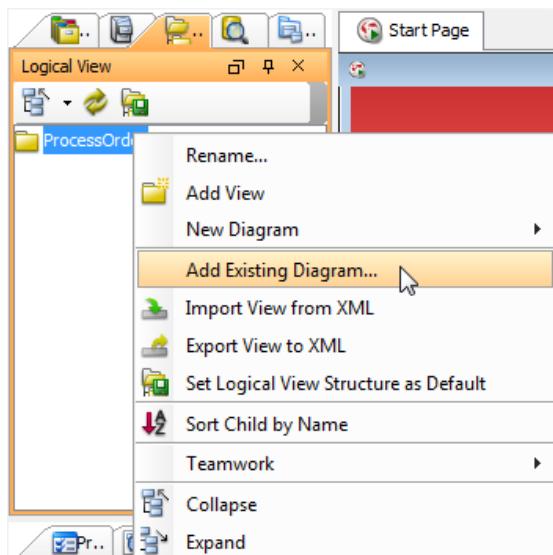
A new view node is, therefore, created under the chosen node.



Created new view node

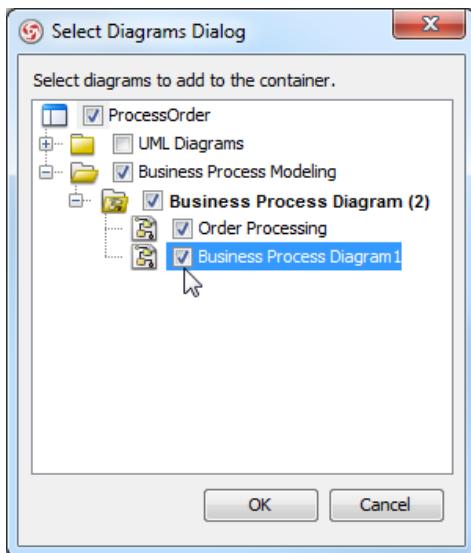
Add diagram to view

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.

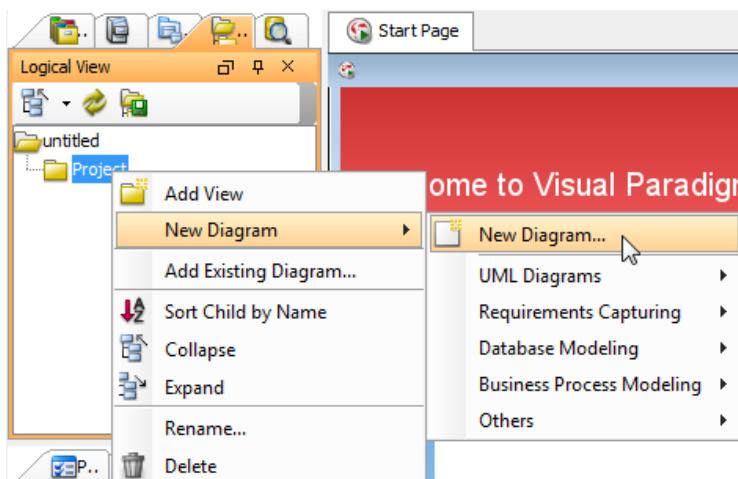


Check diagrams in **Select Diagrams Dialog**

Click **OK** to confirm the selection.

Create a new diagram

Right click the newly created view node, select **New Diagram** from the pop-up menu and 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.

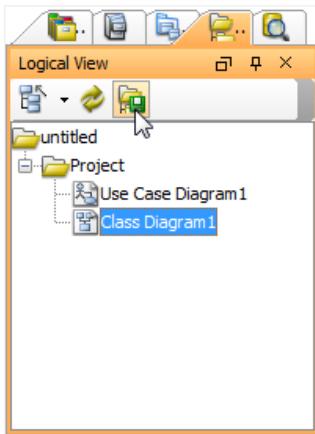
Open a diagram

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

Set Default View Structure

VP-UML 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**.

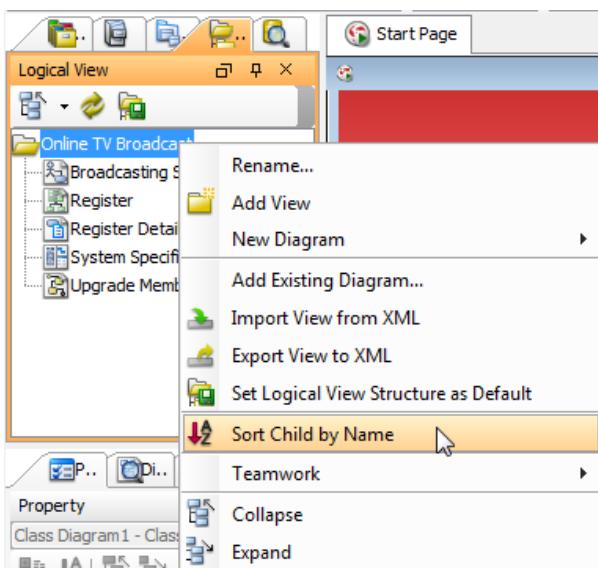


Click Set Logical View Structure as Default button

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.

Sort diagram by name

In logical view, diagrams are listed under diagram nodes by default. You can sort child diagrams by their names as well. To sort by name, click **Sort Child by Name** button. The child diagrams will be listed by name, in alphabetical order.



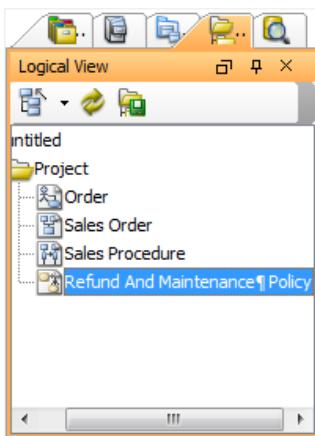
Click Sort Child by Name

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

Show/hide 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

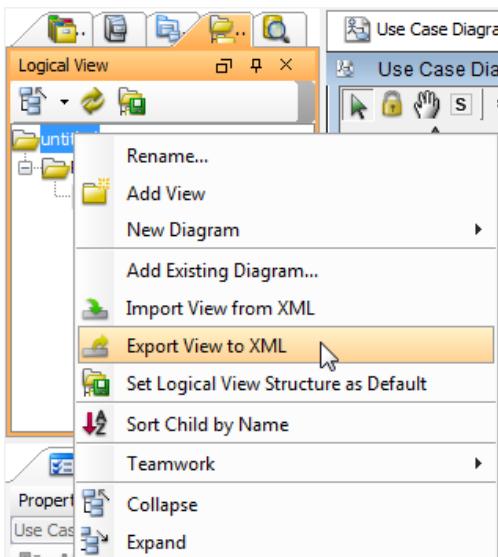
When off, the character ¶ is hidden.

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

Export View Structure to XML

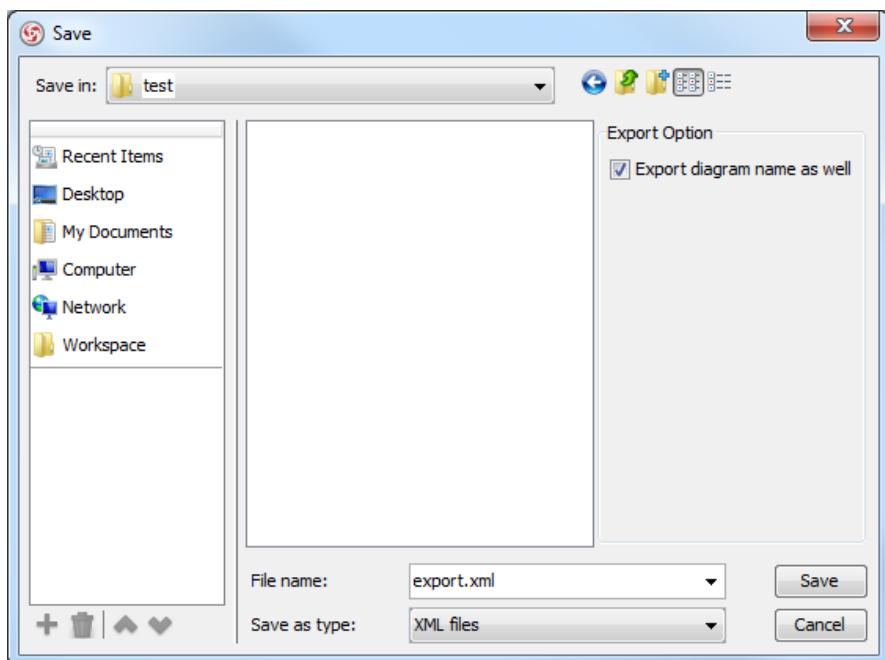
VP-UML allows you to export the current Logical View Structure as an XML file and to re-use it again on other projects.

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**.



Enter the file name in **Save** dialog box

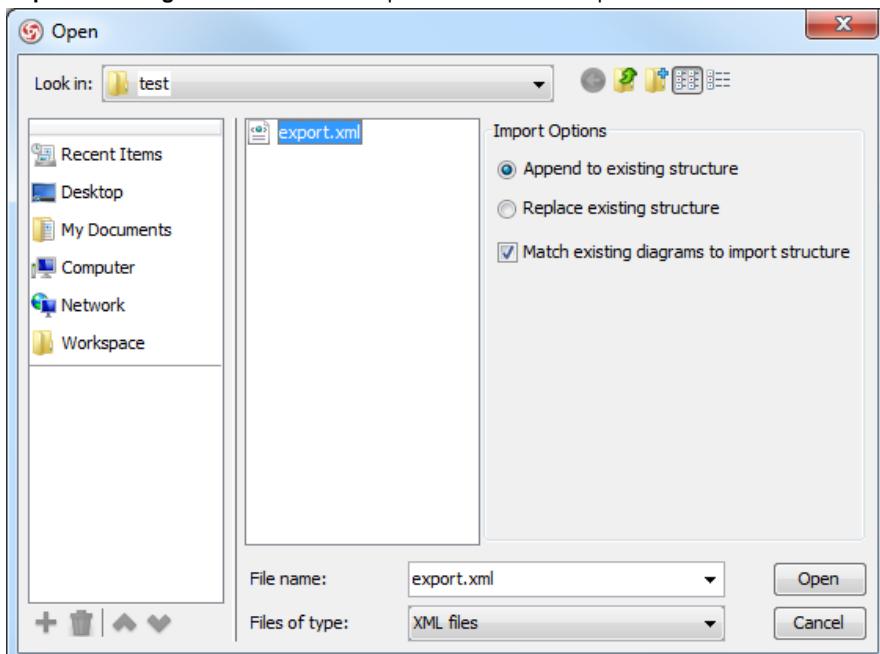
Import View Structure from XML

VP-UML 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.



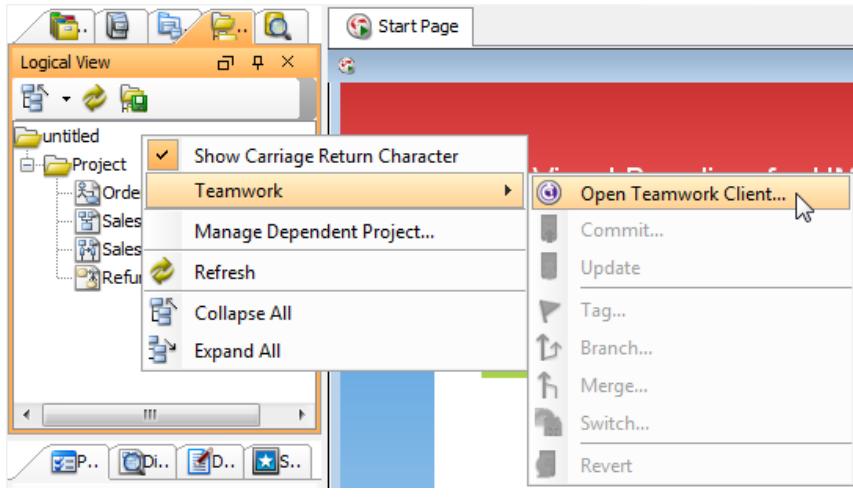
Select *export.xml* in **Open** dialog box

Connect to server for team collaboration

VP-UML'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.

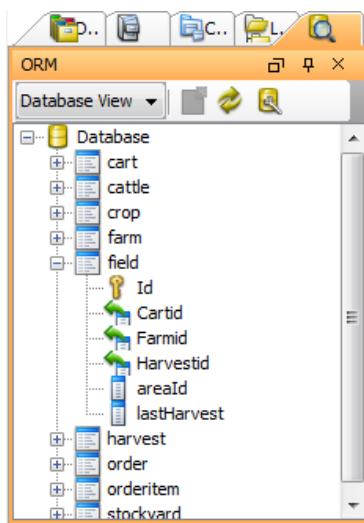
2. Select **Teamwork** and the action you want to perform from the pop-up menu.



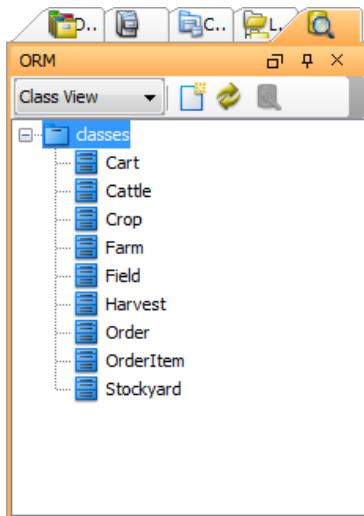
To perform teamwork

ORM pane

ORM pane consists of two views, class view and database view. They two serve two distinct purposes. The class view act as a media to convert domain source code into UML persistable class model, while the database view act as a media to convert database schema into entity models.



An ORM pane (Database view)



An ORM pane (Class view)

The toolbar

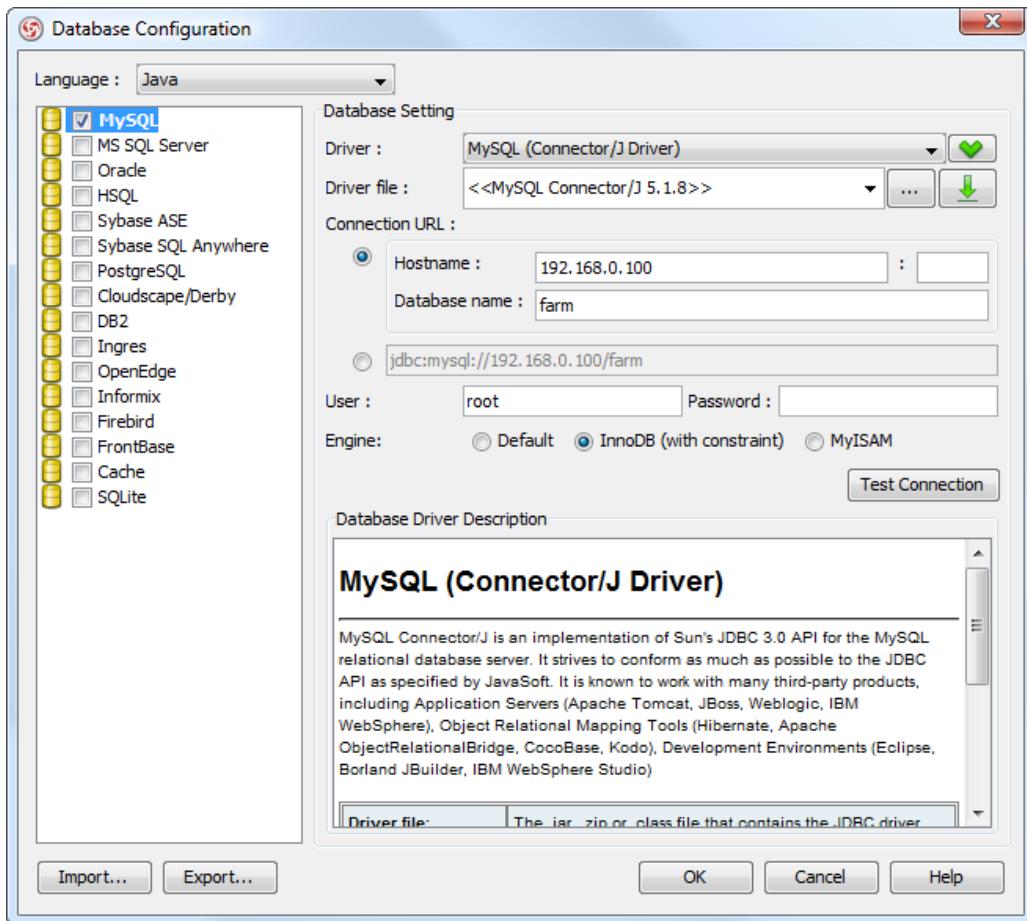
Name	Icon	Description
Database View / Class View		Switch between views. Class view - act as a media to convert domain source code into UML persistable class model. Database view - act as a media to convert database schema into entity models.
Classpath Configuration		Only available in class view, classpath configuration enables you to add or remove directories where Java class files are stored. Classes in class paths will be listed in the pane.
Refresh		By updating the class paths or database configuration, you can refresh the pane to show the updated class or entity listing.
Database Configuration		Only available in entity view, database configuration enables you to set the connection to database, so that ORM pane can connect to that database to read the schema to form the entity list.

The description of icons on ORM pane

Database view

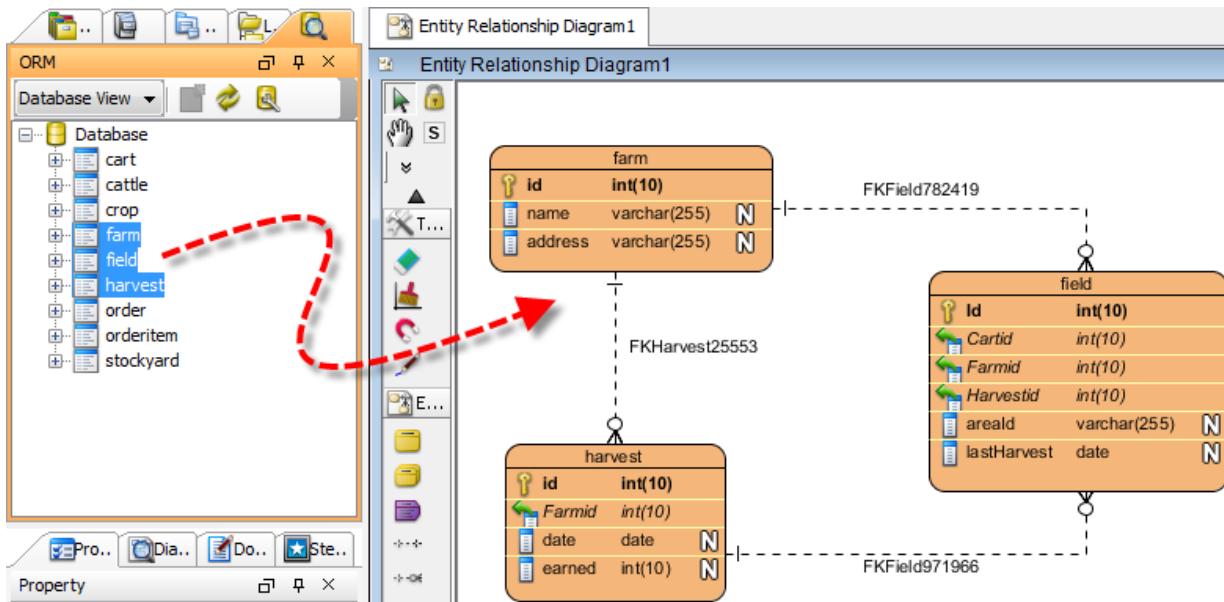
Database view lists tables from a chosen database. The tables are listed in a tree form. You can browse for its columns, and drag entities from tree to diagram, to form an entity relationship diagram.

In order to list tables, you need to configure the database connection first. Click  to open the **Database Configuration** dialog box. Then, specify the database connection for the database you want to have its tables list in ORM pane.



Configuring database connection

By confirming the configuration, tables, if any, will be listed in the ORM pane. You may form an entity relationship diagrams by dragging entities from the pane and releasing in diagram.

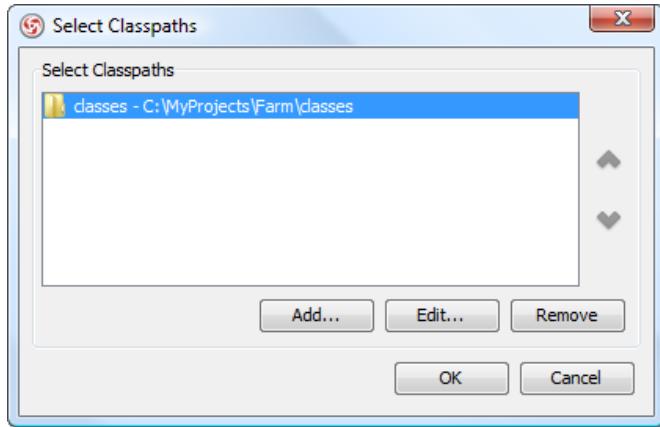


Entity relationship diagram is formed from entities in ORM pane

Class view

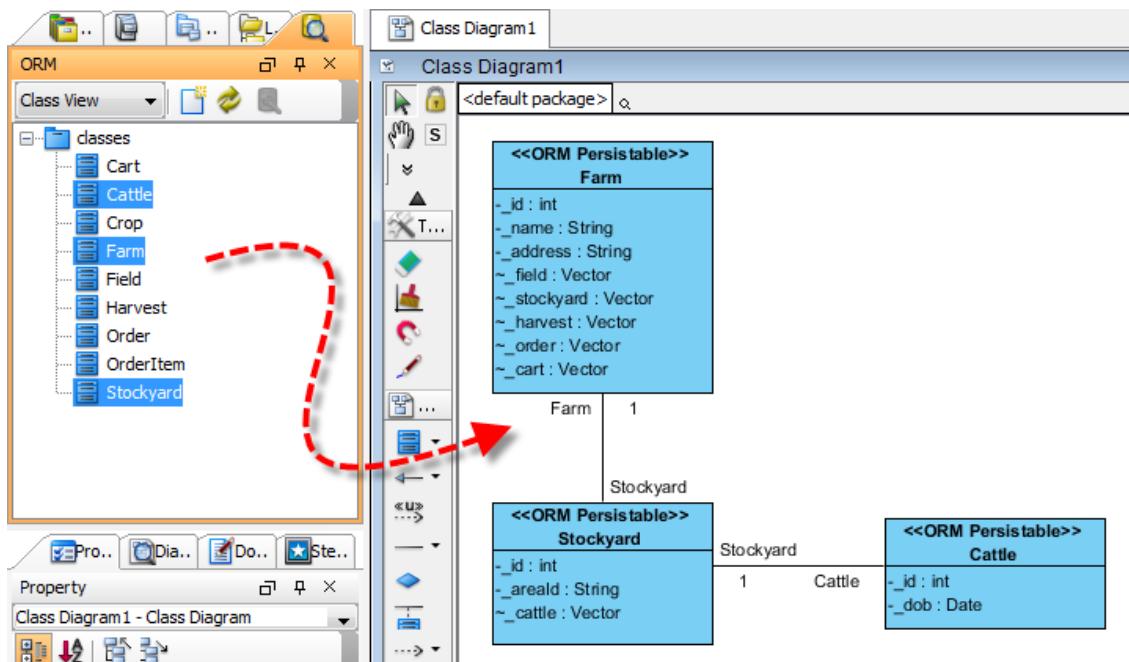
Class view lists classes from chosen classpaths. The main function is to let you convert source code of domain classes into class models that can be used to synchronize an ERD, to generate database in further. In other words, with the class view you can convert domain class (code) into database tables.

In order to list classes, you need to configure add classpaths first. Click  to open the **Classpath Configuration** dialog box. Then, add the classpaths of classes you want to list in ORM pane.



Adding classpaths

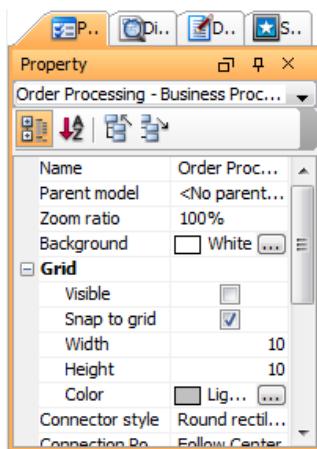
By confirming the classpath selection, classes, if any, will be listed in the ORM pane. You may form a class diagram by dragging classes from the pane and releasing in diagram. You can see the classes created will be extending the ORM Persistable stereotype. This means that they can be synchronized to an entity relationship diagram.



Class diagram is formed from classes in ORM pane

Property pane

Property pane is the location where the properties of a diagram, model element or shape are listed. With Property pane, you can view and edit properties directly.



The **Property** pane

The toolbar

Name	Icon	Description
Categorized View		To sort all properties by their category.
Alphabetical View		To sort all properties in ascending order base on their names.
Collapse		To collapse all properties of each subitem.
Expand		To expand all properties of each subitem.

The description of icons on *Property* pane

Close and open the Property pane

Property pane 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 > Property** from the main menu.

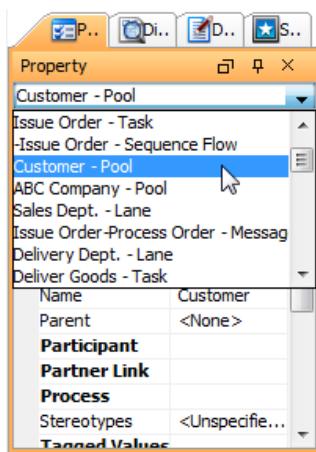
View the properties

Click a shape that you would like to view its properties directly when the shape is shown on diagram pane. There are not only the properties of shapes, but also the properties of diagrams in **Diagram Navigator** and the properties of model elements in **Model Explorer** that can be viewed in property pane.

Click a diagram on **Diagram Navigator** to view the properties of diagram while click a model element in **Model Explorer** to view the properties of model element.

View the properties in shortcut

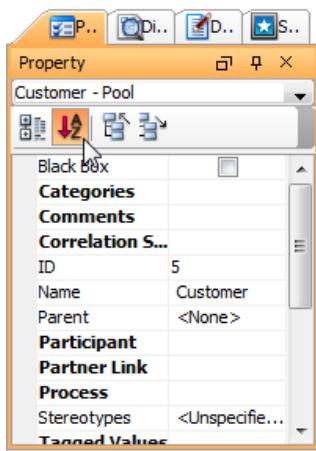
When a diagram, model element or shape is selected, its properties are shown on property pane. Meanwhile, you are allowed to view other properties, which are in the same diagram, in shortcut by selecting the diagram, model element and shape from dropdown menu at the top of the property pane.



Select **Customer - Pool** in dropdown menu

Sort the properties

In Property pane, the properties of a diagram, model element and shape are listed. You can sort the properties by their category, or in ascending order. To sort by the name of properties, click **Alphabetical View** button. All properties will be listed in ascending order base on their names.

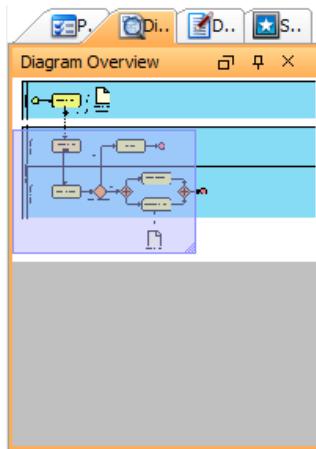


Click **Alphabetical Vie w**

To sort by category, click **Categorized View** button. As a result, all properties will be listed by category.

Diagram overview

Diagram overview is a pane where users can view and zoom in an active diagram directly and shortly.



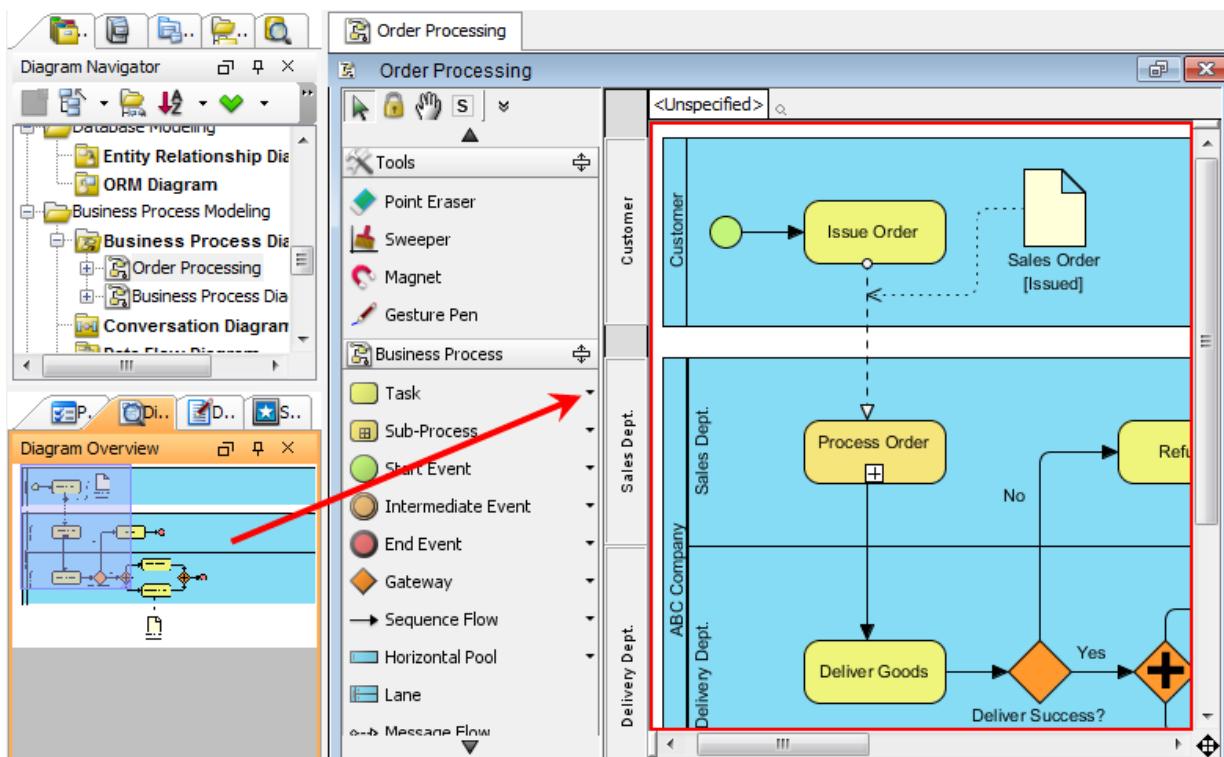
The Diagram overview

Close and open the diagram overview

Diagram overview 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 > Diagram Overview** from the main menu.

View an active diagram

An active diagram can be viewed in diagram overview automatically. As the diagram is shown on the diagram pane, it can be viewed in diagram overview.

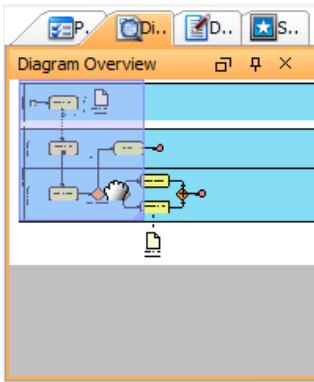


An active diagram is shown on Diagram Overview

Have a quick view on a particular part of diagram

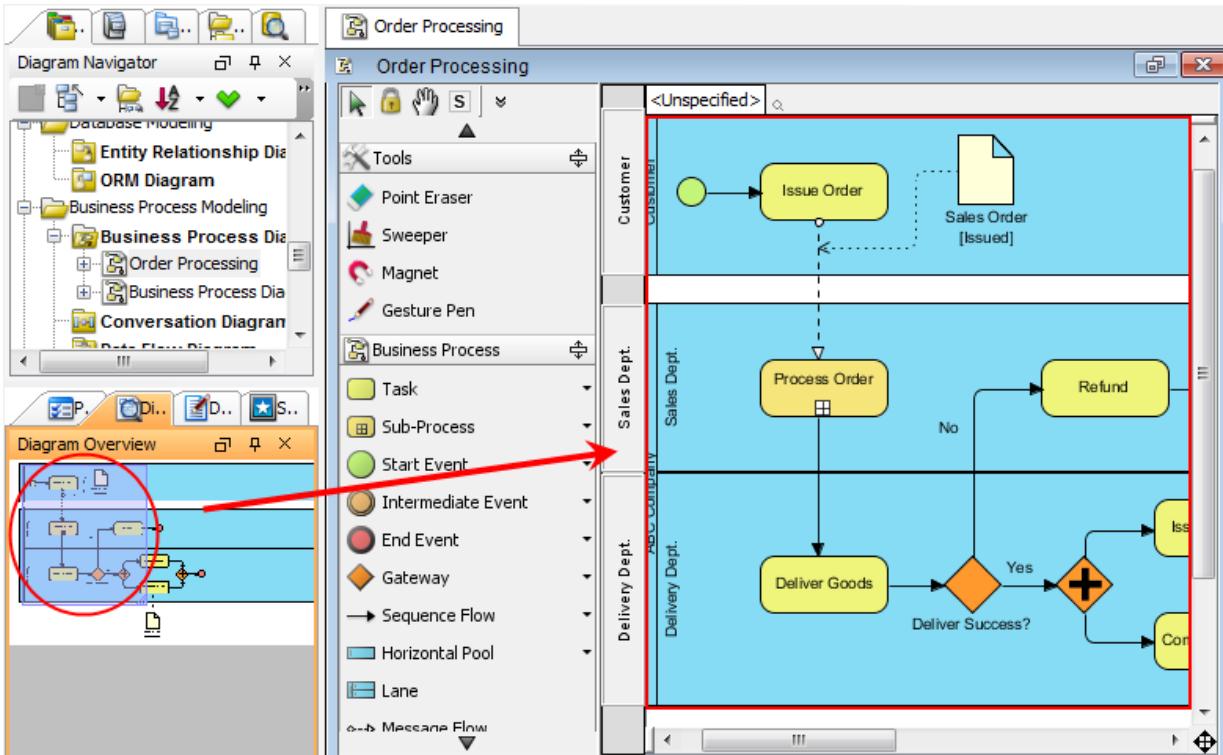
There is much truth in saying that viewing a large diagram is such an annoying task, especially a particular part of this large diagram is needed to focus on. In fact, a particular part of diagram can be navigated by moving the purple rectangle which represents the visible area of diagram in diagram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

- Press on the purple rectangle and drag it to the preferred part of diagram you would like to view.



Drag to the preferred part of diagram

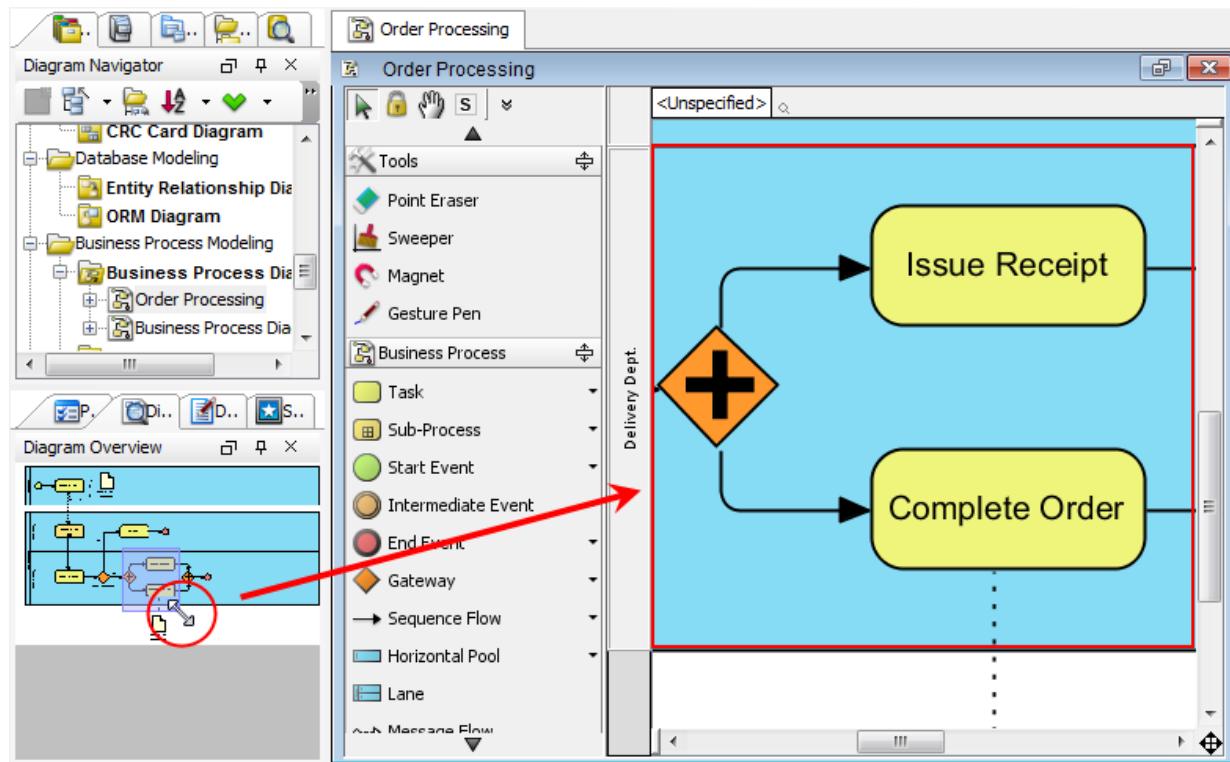
- As a result, the part of diagram will be subsequently shown on the diagram pane.



The particular part of diagram is viewed on diagram pane

Zoom 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.

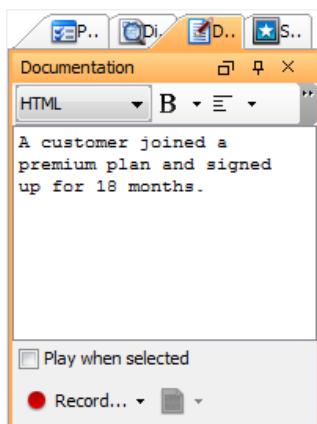


Drag the diagonal of purple rectangle

On the contrary, the larger you drag the purple rectangle, the more the part of diagram will be dwindled.

Documentation pane

Documentation 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 formatings like bold, italic, table, etc.



The **Documentation** pane

The toolbar

Name	Icon	Description
HTML		It enables you to read and edit 3 different types of content. The 3 types of content include HTML , HTML Source and Plain Text . HTML : Read and edit the original content. HTML Source : Read and edit the HTML source of content. Plain Text : Read and edit content without formats.
Bold		Set the highlighted text to bold.
Italic		Set the highlighted text to italic.
Underline		Underline the highlighted text.
Left Justify		Set the alignment of highlighted text to left.
Center Justify		Set the alignment of highlighted text to center.
Right Justify		Set the alignment of highlighted text to right.
Ordered list		Add a numbered list.
Un-ordered list		Add a list with bullet points.
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. A few formats of insertion for rows and columns can be selected, including: Insert Row Above , Insert Row Below , Insert Column on Left and Insert Column on Right . Insert Row Above : Insert a row above the row you selected. Insert Row Below : Insert a row below the row you selected. Insert Column on Left : Insert a column on the left of the column you selected. Insert Column on Right : Insert a column on the right of the column you selected.
Background color		Select the background color of highlighted text.
Clear formats		Clear formats of whole editor to convert the content to plain text.
Link		Add a hyperlink.

Image		Add an image.
Save as template...		Save the documentation as template.
Manage Template...		Preview a saved template.
Print		Print the custom content.

The description of icons on Documentation pane

Close and open the Documentation pane

Documentation pane 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 > Documentation** from the main menu.

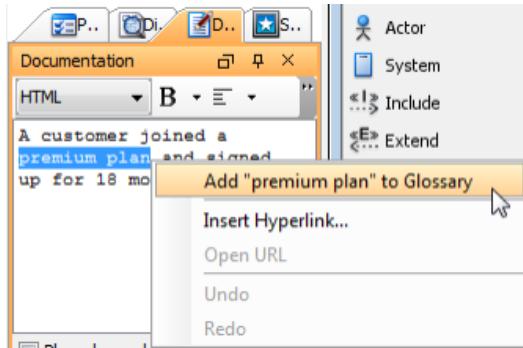
Document project data in text

Documentation pane enables users to type in the textual description for project data, for instance, model elements, shapes and diagrams.

Define a glossary item

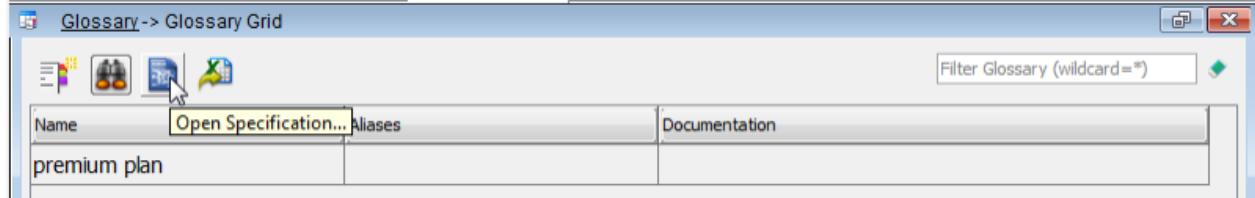
A word or a lexis can be defined as a glossary item for explication.

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



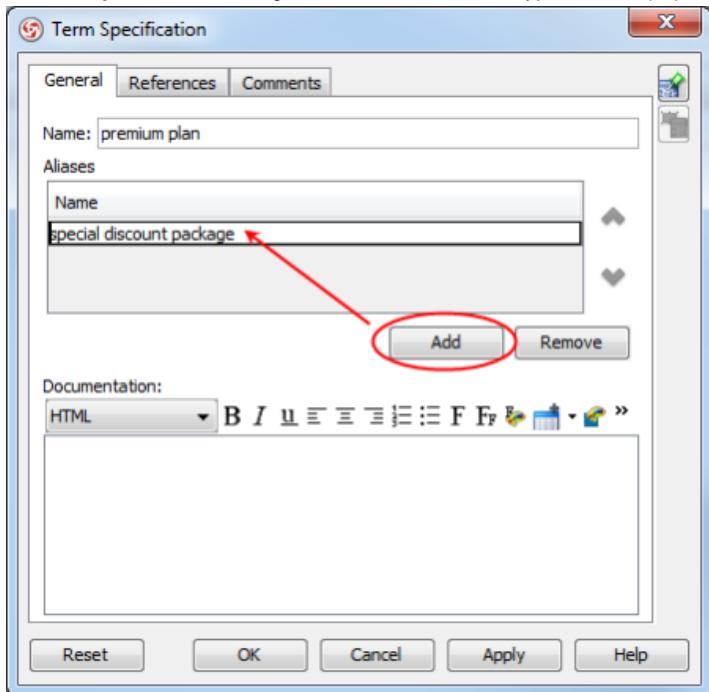
Select Add "premium plan" to Glossary from the pop-up menu

2. In **Glossary Grid**, click **Open Specification...** button in order to fill more details about the new item.



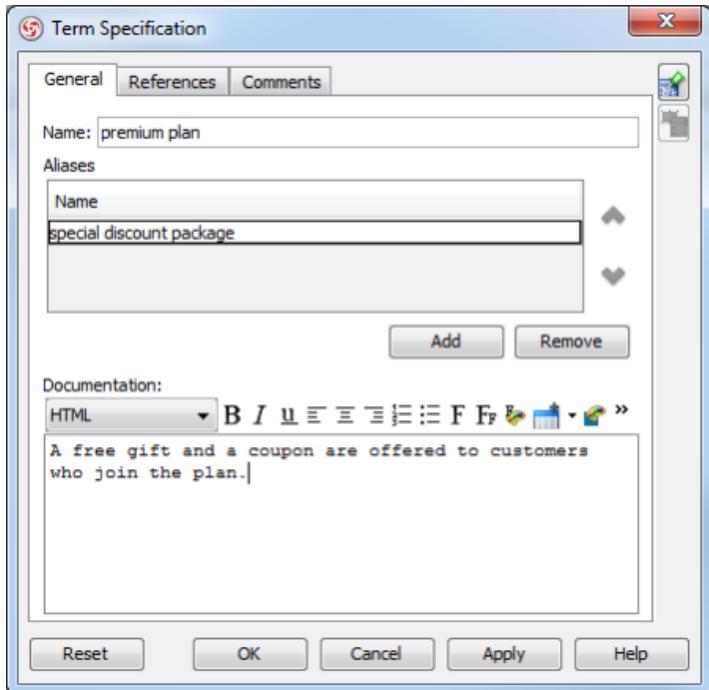
Click Open Specification... button

3. In **Term Specification** dialog box, 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 **Documentation**.



Type in text

5. Finally, click **OK** to confirm editing. The window will then return to **Glossary Grid**.

6. Moreover, you can insert as many new terms as you prefer. In Glossary Grid, click **Create Term (Insert)** to create another new term.

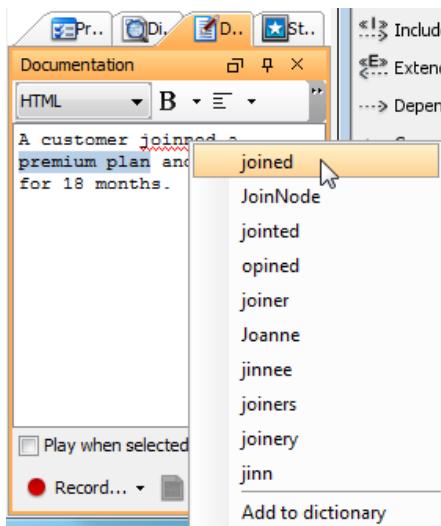
Glossary -> Glossary Grid		
Name	Aliases	Documentation
premium plan	special discount package	A free gift and a coupon are offered to customers who join the plan.

Click **Create Term (Insert)** button

Check spelling

When you type an incorrect word carelessly, documentation 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.



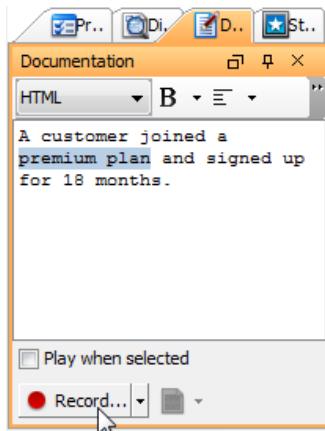
Select a correct word from the pop-up menu

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.

Document a shape in voice

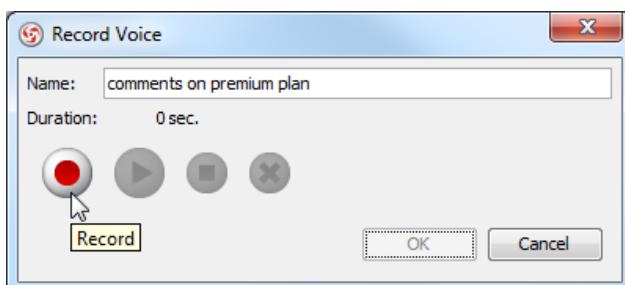
In addition to textual description, documentation pane also enables users to document for project data in verbal form.

Click **Record...** button for recording voice.



Click **Record...** button

In **Record Voice** dialog box, enter the name for the audio clip. Click **Record** to start recording while click **Stop** to terminate. Click **OK** if you want to save the voice recording; click **Cancel**, and vice versa.



Click **Record** in **Record Voice** dialog box

Name	Icon	Description
Record		Press it to start recording.
Play		Press it to play the voice you recorded.

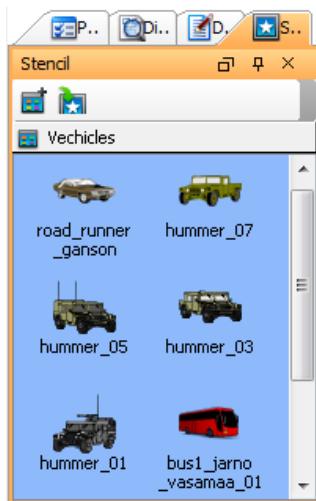
Stop  Press it to stop recording.

Clear  Press it to clear the voice you recorded.

*The description of icons on **Record Voice** dialog box*

Stencil pane

Stencil pane is a library of custom shapes or images that can be used on diagrams. With Stencil Pane, users can create custom shapes and display stencils by selecting stencils and drag them on diagram. Furthermore, stencils can be created in **Shape Editor**.



The **Stencil** pane

The toolbar

Name	Icon	Description
Add Stencil		Select a stencil to create a new shape. Different sorts of stencil are categorized into two folders: Computers and Shapes and a large amount of subfolders are subdivided into these two folders.
Import Stencil...		Import a stencil that created externally in Microsoft Visio to VP-UML.

The description of icons on Stencil pane

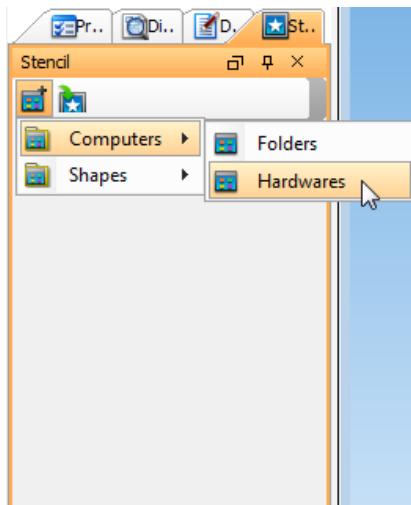
Close and open the Stencil pane

Stencil pane 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 > Stencil** from the main menu.

Create a stencil

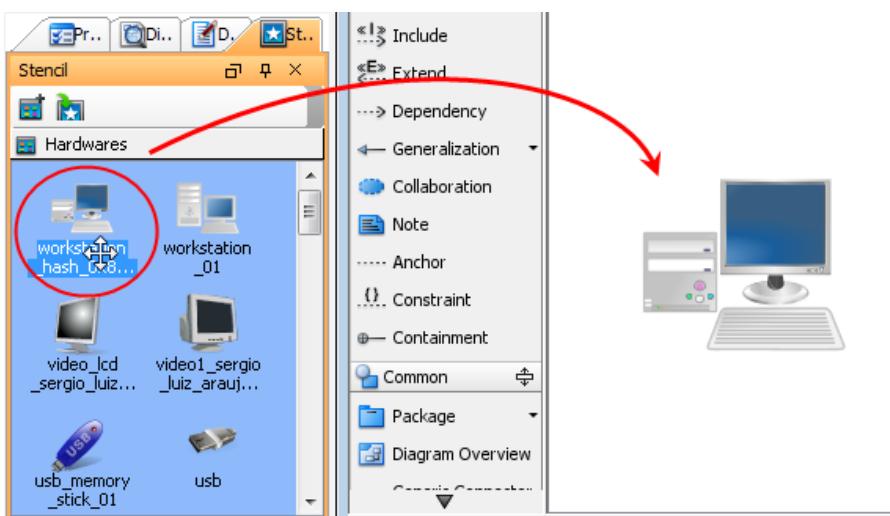
Instead of keeping the existing shapes, users can create a stencil to replace an existing one.

Click **Add Stencil** button on the top of stencil pane and select a subfolder from the pop-up menu.



Select **Hardwares** from the pop-up menu

When the subfolder is unfolded, press the preferred stencil and drag it on the diagram pane.

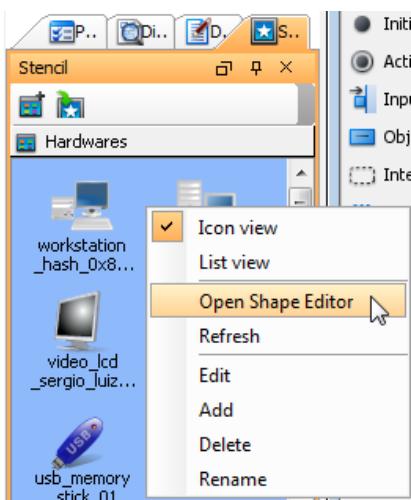


Drag a stencil on the diagram pane

Edit a stencil

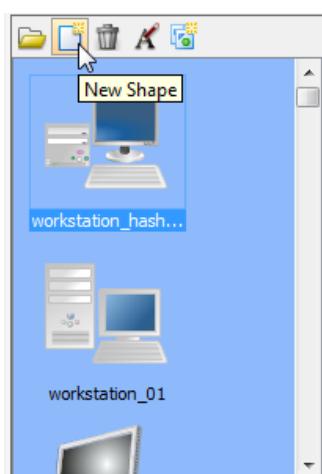
A stencil can be customized according to users' preference in **Shape Editor**.

Right click on the stencil pane's background and select **Open Shape Editor** from the pop-up menu.



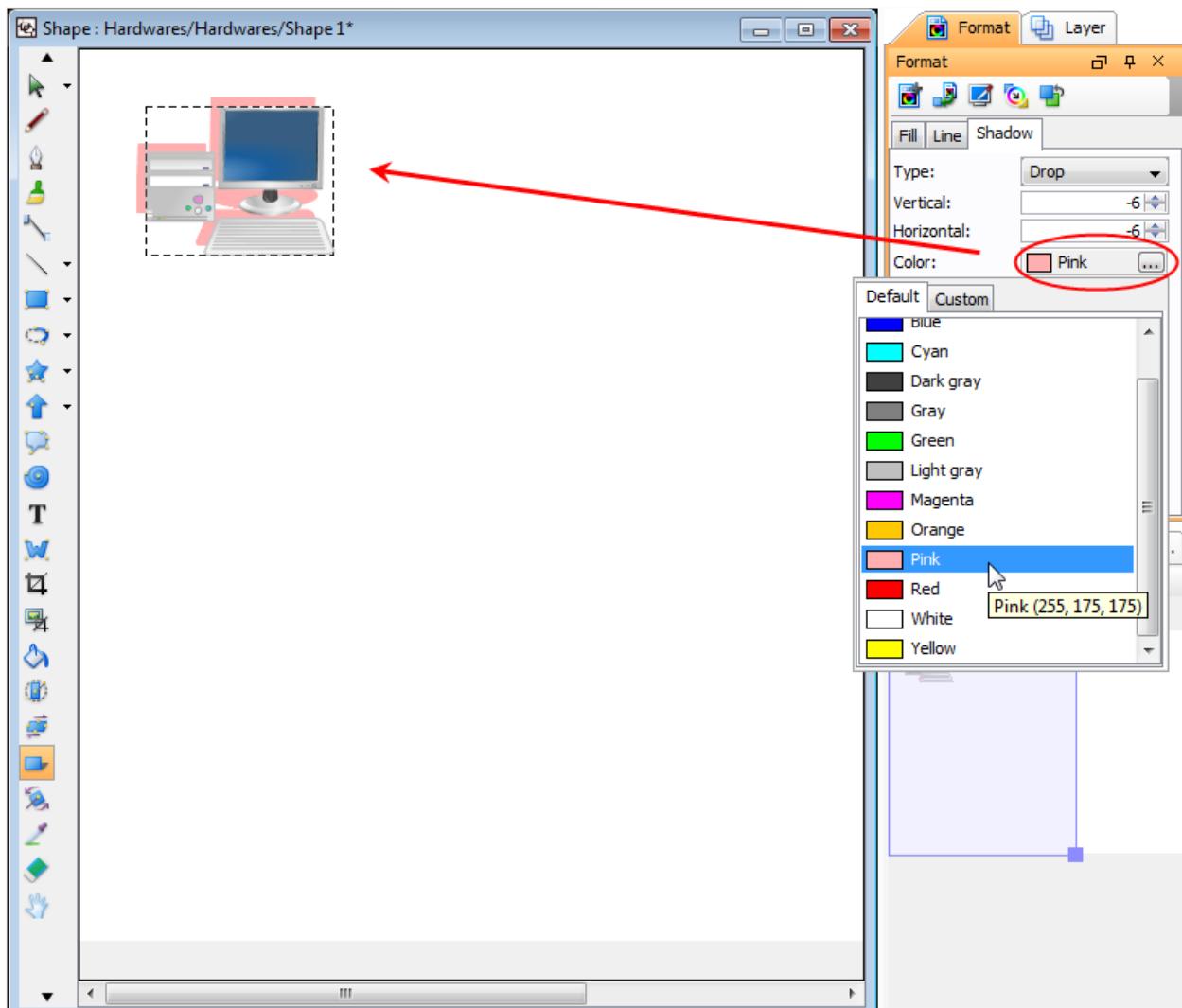
*Select Open Shape Editor
from the pop-up menu*

In **Shape Editor** dialog box, click **New Shape** button.



Click New Shape button

Press on the stencil you want to be edited and drag it on the diagram pane.
Insert any shape(s) from the diagram toolbar with your preference and edit the format for the shape(s) in **Format** tab.



Turn the shadow of stencil into pink

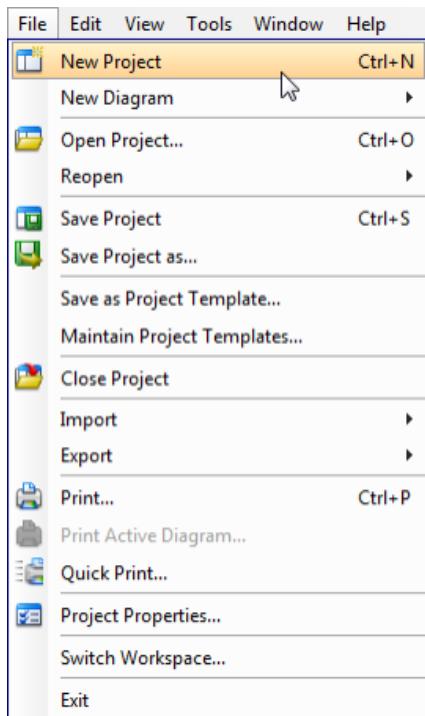
Import a stencil

The function of importing stencil enables you to import a stencil which is created in Microsoft Visio externally and send it to VP-UML by making use of the plug-in "VisioSendToVP".

Note that you are able to execute this function only after you have installed Microsoft Visio and have sent a stencil from Visio.

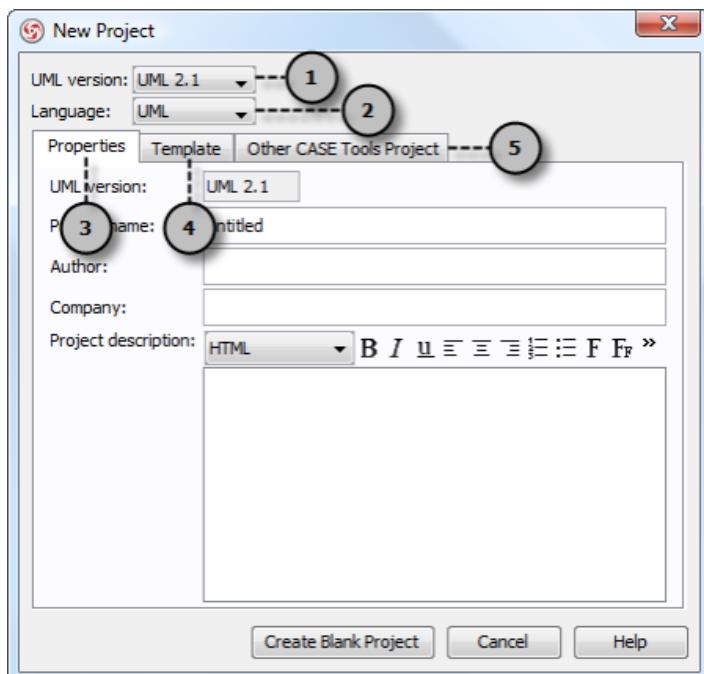
Creating project

Visual Paradigm for UML 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 menu **File > New Project**. The **New Project** dialog box appears. Click on **Create Blank Project** to create the project.



Create a new project

Overview of New Project dialog box

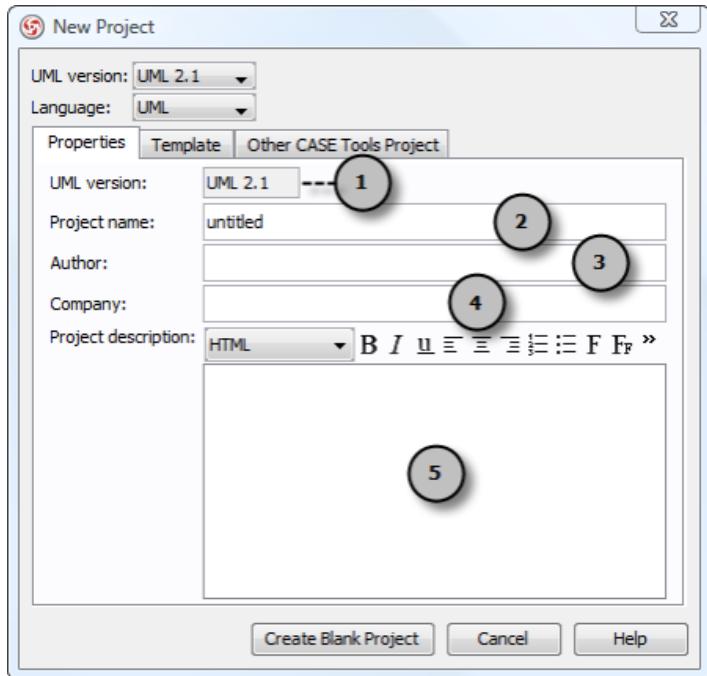


An overview of New Project dialog box

No.	Name	Description
1	UML version	Select the version of UML notation for the project. Normally you would select UML 2.1 so you can create UML diagrams of the latest standard. However if you want to create diagrams with older notation you should select UML 1.x.
2	Language	The Language combo box 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.
3	Properties	A page for specifying basic information of project like the project name, author, company and description.
4	Template	A page for you to create a project by selecting a template.

*Description of New Project dialog box***Properties**

To create a blank project, ensure the **Properties** page is being selected, and then click the **Create Blank Project** button.



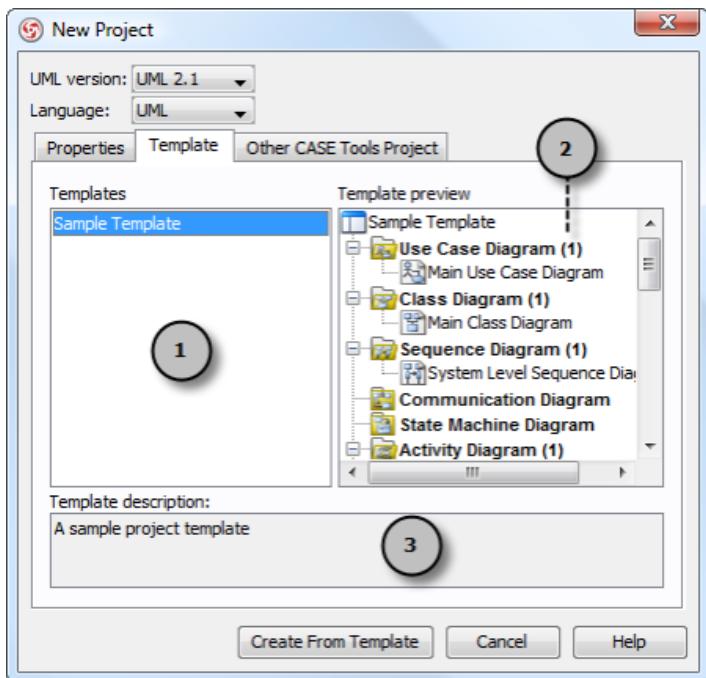
An overview of **Properties** page

No.	Name	Description
1	UML version	Select the version of UML notation for the project. Normally you would select UML 2.1 so you can create UML diagrams of the latest standard. However if you want to create diagrams with older notation you should select UML 1.x.
2	Project name	The name of project.
3	Author	The author of project.
4	Company	The company of project.
5	Project description	The project description. You can make use of the toolbar on top of the description pane to add formatted content.

*Description of Properties page***Template**

If you often create projects of similar structure, you can save the project as a template and use it for creating project later. For example, you can save a project with a use case diagram and a class diagram as template called "Static Modeling Template". When we create project from this template, the project will have the same structure as the template automatically.

To create project from template, select the **Template** page and select a template to use, then click the **Create From Template** button.



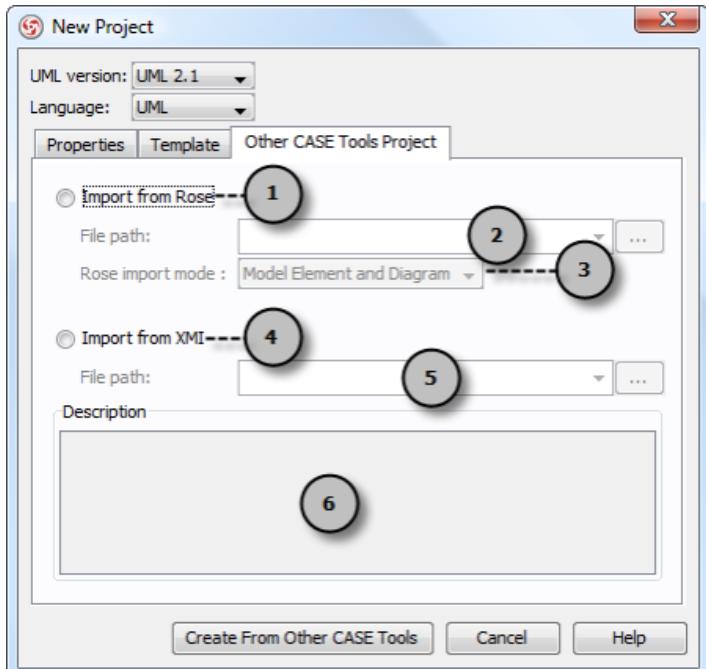
An overview of **Template** page

No.	Name	Description
1	Template list	A list of template that you have defined before.
2	Template preview	By selecting a template from the template list, you can read the structure in the Template preview pane.
3	Template description	A description of the template.

Description of **Template** page

Other CASE Tools Project

If you have legacy projects created with other CASE tools, you can import them to VP-UML. Select either to import from Rose or from XMI, specify the file path of source to import and click **Create From Other CASE Tools** at the bottom of dialog box.



An overview of **Other CASE Tools Project** page

No.	Name	Description

1 Import from Rose	Select this option if you want to create a new project by importing a Rational Rose project.
2 File path	The file path of Rational Rose (*.mdl) project.
3 Rose import mode	<p>The selection controls what to import. There are two modes:</p> <p>Model element and diagram -Import both model elements (e.g. use case, class...) and diagrams (e.g. use case diagram, class diagram...)</p> <p>Model element only - Import only model element but not diagram. In the new project you can find the imported model elements in Model Explorer.</p>
4 Import from XMI	Select this option if you want to create a new project by importing a XMI file.
5 File path	The file path of XMI (*.xmi) project.
6 Description	Description of chosen option.

Description of Other CASE Tools Project page

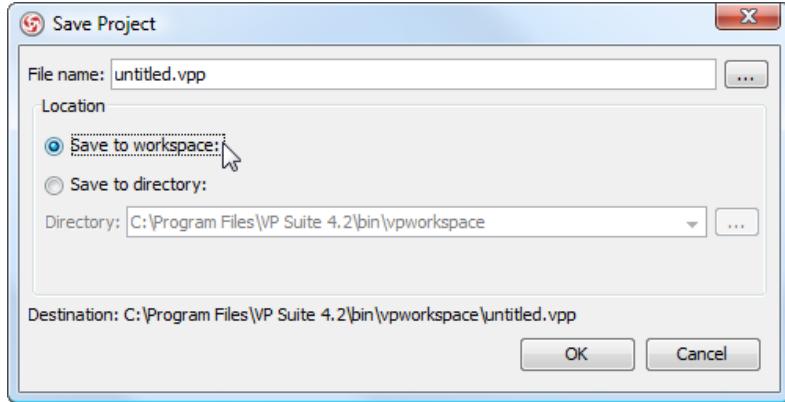
Saving project

VP-UML saves all project content to a single file, with file extension .vpp.

To save project, select **File > Save Project** from the main menu. If you are saving the first time, the **Save Project** dialog box will appear. You can save a project to workspace, or any location in the machine. For details, read the sections below.

Saving project to workspace

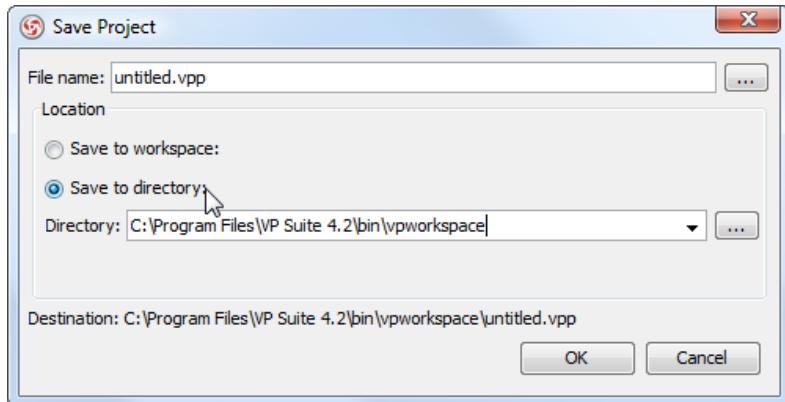
To save project to workspace, select **Save to workspace**. Enter the file name and click **OK** to save.



Save project to workspace

Saving project to specified directory

To save project to a specified directory, select **Save to directory**, and then specify in the **Directory** field the directory for saving project. Enter the file name and click **OK** to save.



Save project to a specified directory

Organizing diagrams by model explorer

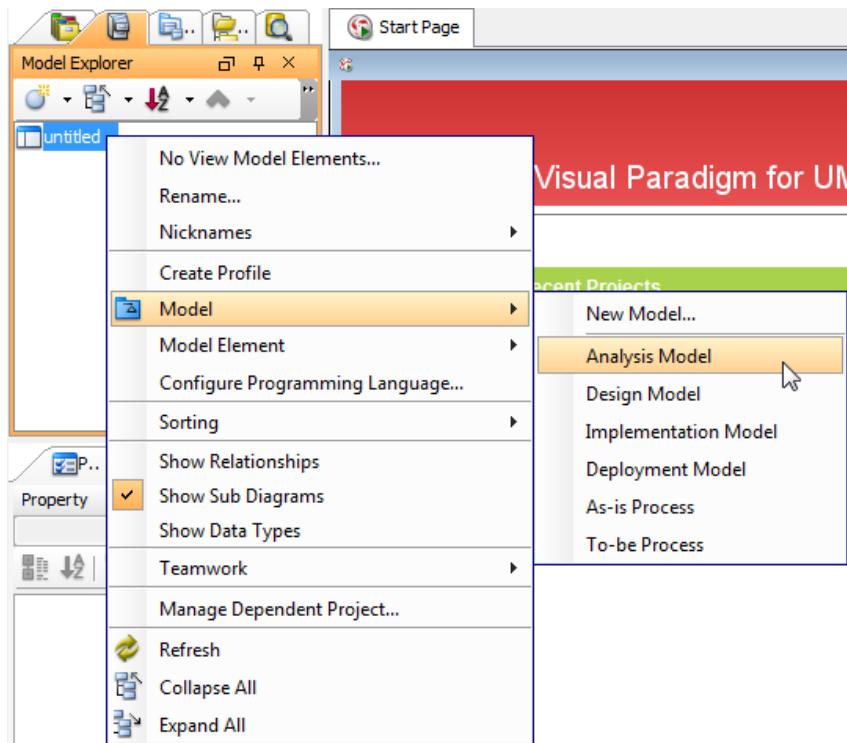
For small scale project, it would be easy to use Diagram Navigator to manage it. However, for middle to large scale project which has considerable numbers of diagrams and model elements, it would be better to use Model Explorer to organize the project.

VP-UML 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 Explorer 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.

Creating model

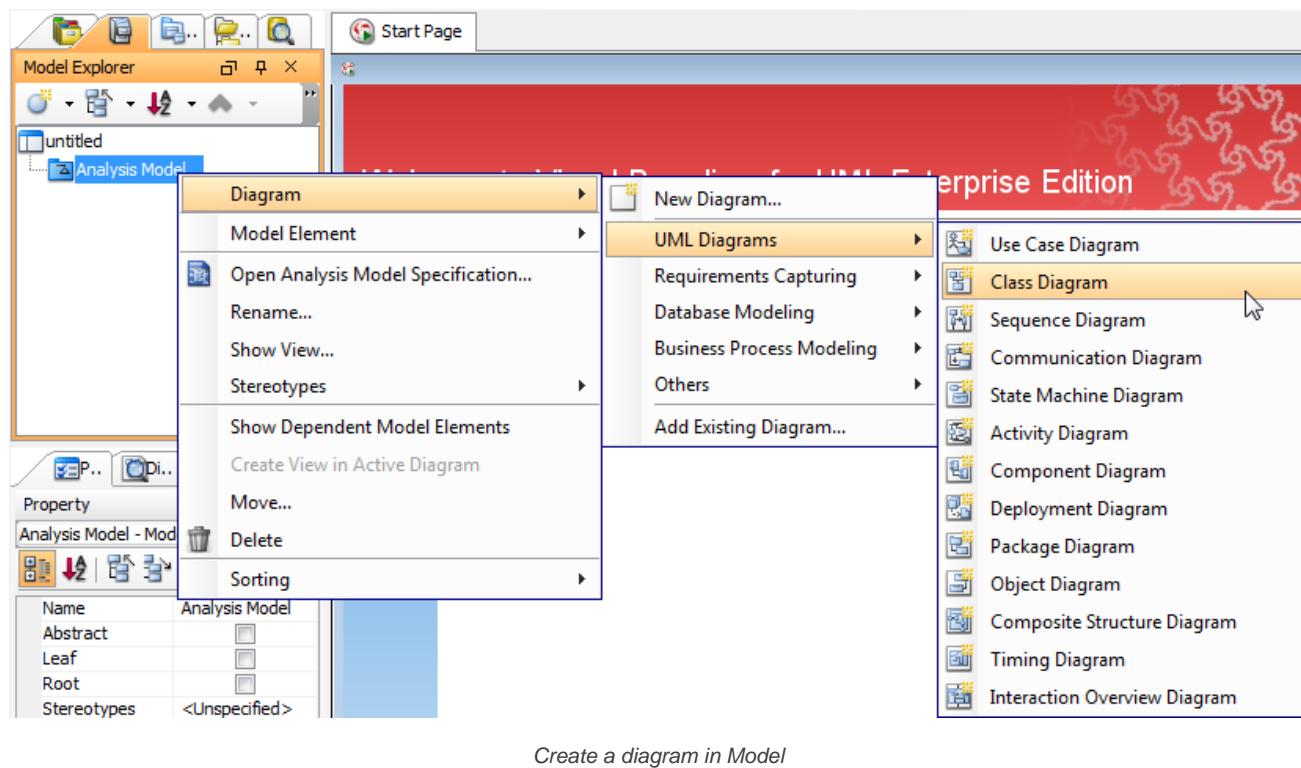
To create a Model, right-click on the project node in Model Explorer and select **Model** from the popup menu. You can either create a custom Model by selecting **New Model...**, or create a pre-defined Model (e.g. Analysis Model) by selecting it in the list.



Save project to a specified directory

Creating diagram in model

To create diagram in Model, right-click on the Model and select **Diagram** from the popup menu, and then select to either create a new diagram or add existing diagrams to the Model.

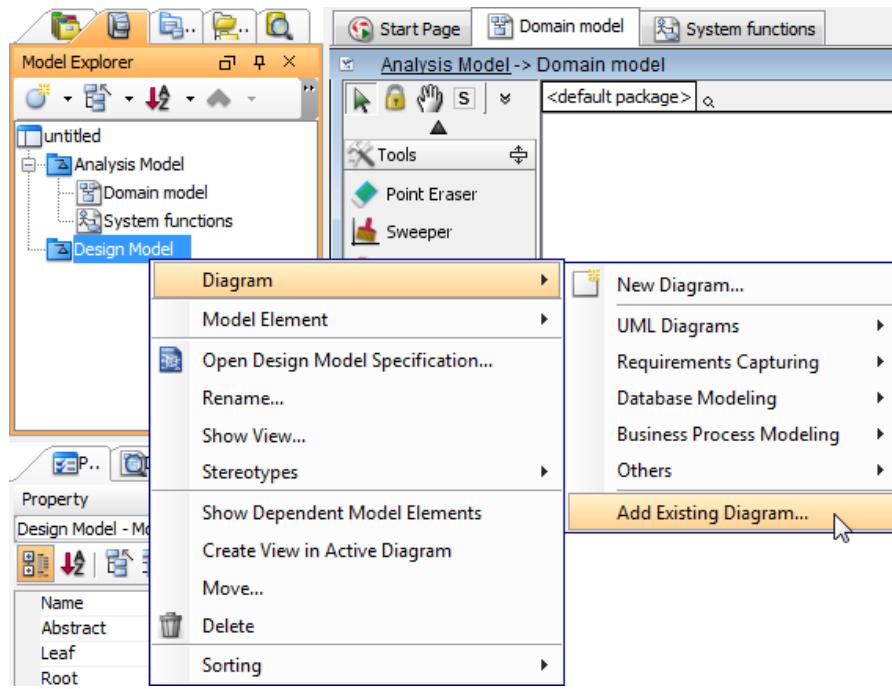


NOTE: When you create a shape, its model element will be put under the same model as diagram.

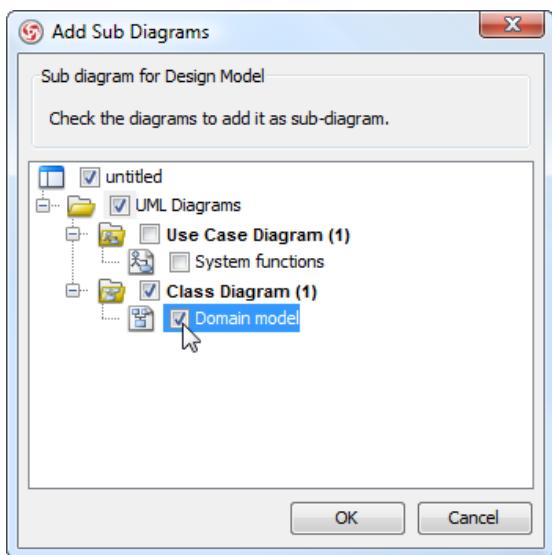
Moving diagrams between models

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 to transfer a diagram from one model to another.

To move diagram from one model to another, right-click on the target model in **Model Explorer** and select **Diagram > Add Existing Diagram...** from the popup menu.



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



Select diagrams to move

The selected diagrams will be moved to the target Model.

NOTE: Model elements that has master view on the diagram to move will move alongside with the diagram to the new model.

Project dependency

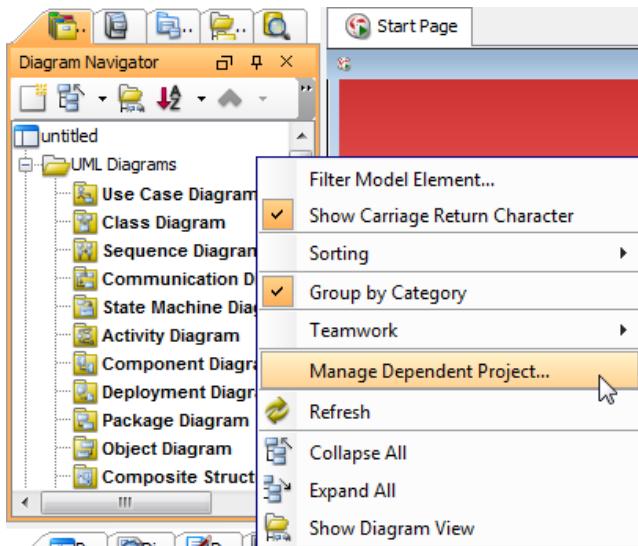
Project dependency enables you to reference to an another project, and use its model elements in developing our own project. You can organize our model elements in a more disciplined approach by having one Visual Paradigm project per library project. This also helps to "slim up" projects by breaking down a project one into smaller pieces. Other than this, you can reference to other projects, and create an overview project for them.

Adding dependent projects

To reference to another project:

1. Right-click on the background of any of the following panes and select **Manage Dependent Project...** from the popup menu.

- Diagram Navigator
- Model Explorer
- Class Repository
- Logical View
- ORM



To manage dependent project

2. In the **Manage Dependent Projects** dialog box, click **Add**.
3. In the **Open** dialog box, choose the Visual Paradigm project file (*.vpp) to reference to and click **Open**.

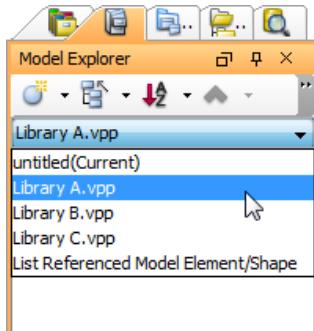
NOTE: You can only reference to a project that was saved under the same version of VP-UML you are opening. For example, if you are running VP-UML 7.1, you can only reference to a project saved by version 7.1 of VP-UML.

4. Click **Close**.

Using model elements in dependent projects

Once a project dependency has been established, you can develop your model using model elements in dependent project. To do this:

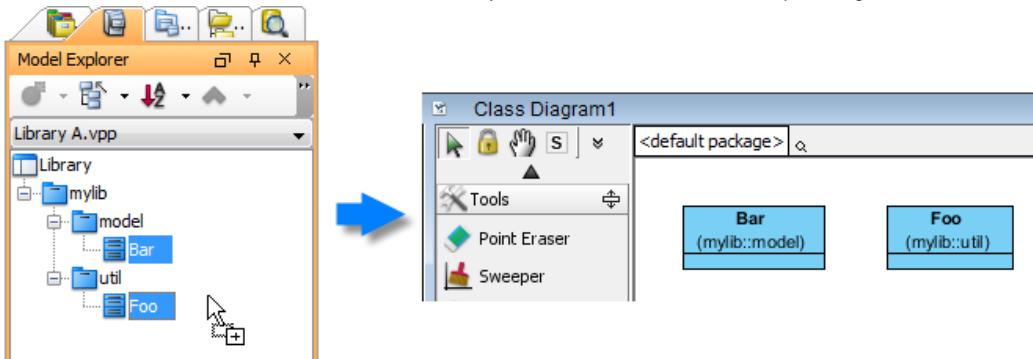
1. In **Model Explorer**, click on the drop down menu at the top of the pane and select the project that contains the model elements you want to use.



Selecting a dependent project

NOTE: By selecting the first selection (**Current**), model elements in the current project will be listed.

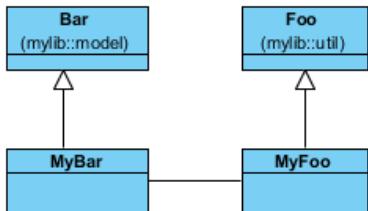
2. In the model element list, select the model element you want to use. Press and drop to diagram. This creates views from them.



Forming class diagram from model elements in dependent project

NOTE: Alternatively, you can right-click on the model elements and select **Create View in Active Diagram** from the popup menu.

3. Continue modeling with the referenced elements.



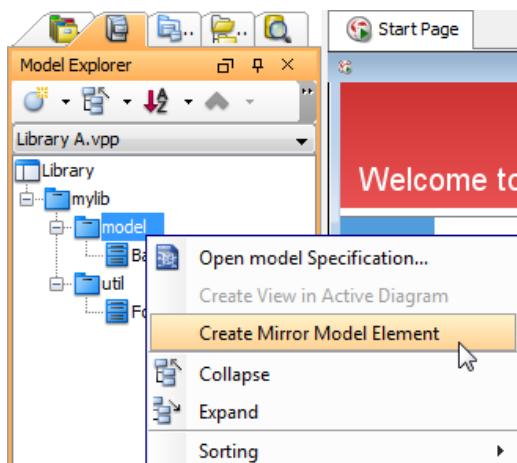
Modeling with model elements in dependent project

NOTE: Views for dependent project are read-only (i.e. non edit-able).

Mirroring model element

Due to views of dependent model elements are read-only, you cannot add shapes in it. This may be a problem when you want to use dependent 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 dependent model element. By mirroring, a dependent element is localized partially by keeping a mirrored copy in your project which echoes the element in dependent 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 dependent model element (e.g. package, business pools/lanes...), right click on the element in **Model Explorer** and select **Create Mirror Model Element** from popup menu. 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 can also right click on the shape and select **Convert to Mirror** from popup menu.

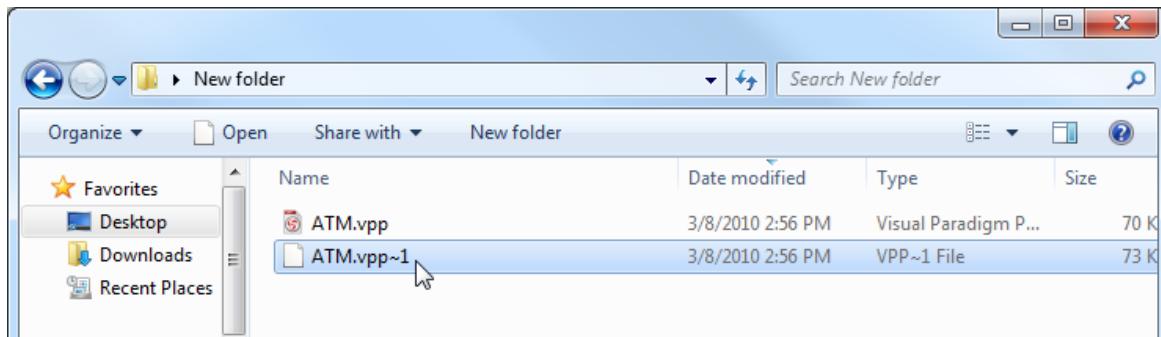


Mirror a dependent package

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.

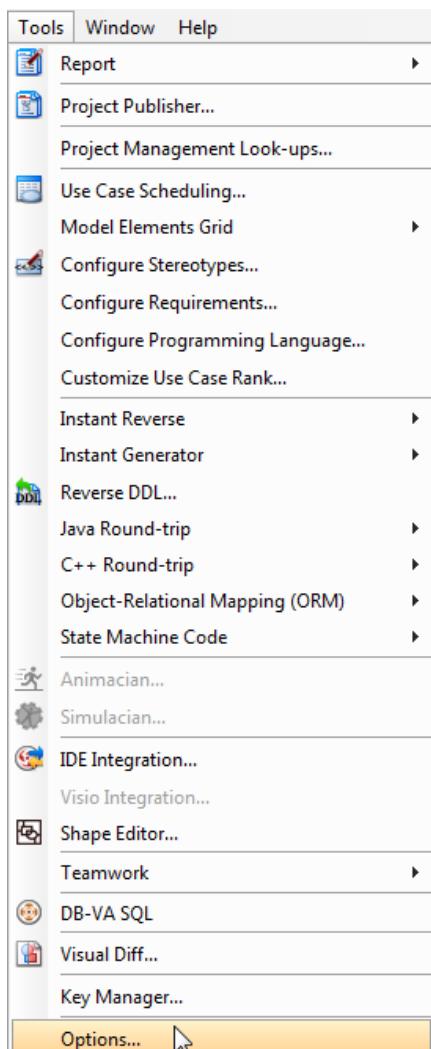
Backup is a default setting. After you save your project, it will be produced subsequently. The name of backup file is basically similar to the name of project file, but an extra ~ and a number are appended to it.



Backup copy is produced

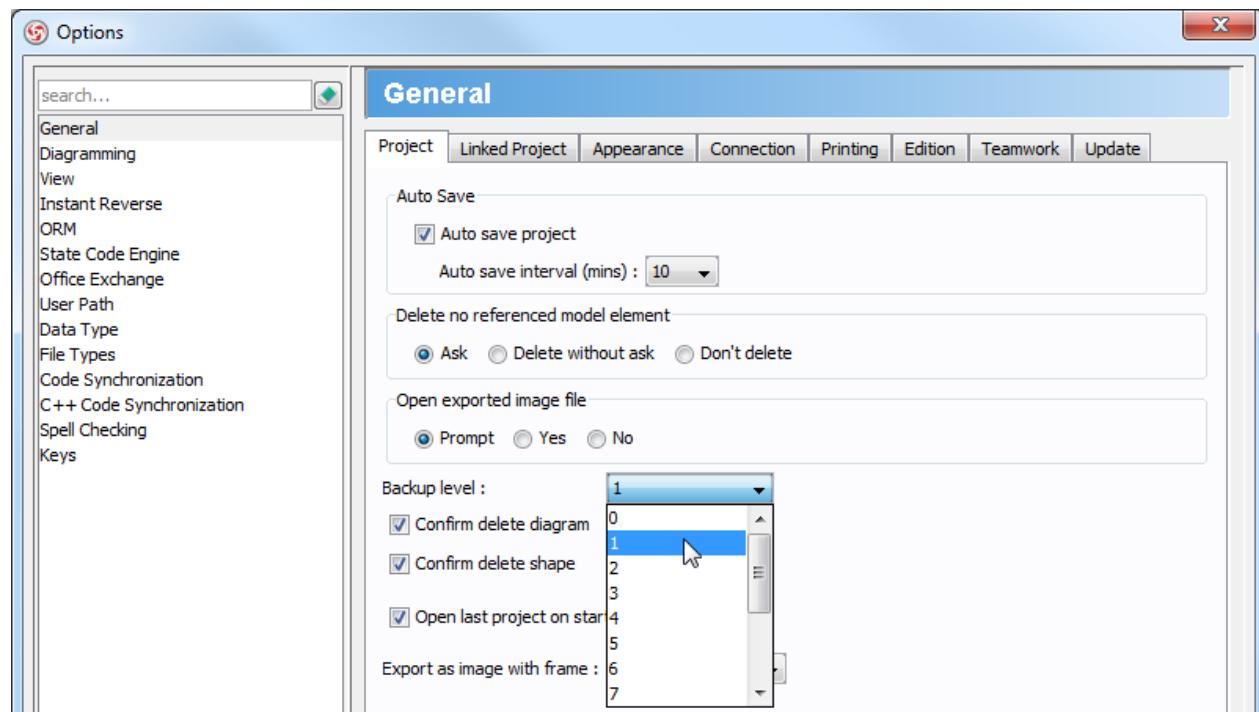
Setting the backup level

You can set the amount of backup copy for your own reference. Select **Tools > Options...** from the main menu.



*Select **Options...** from the main menu*

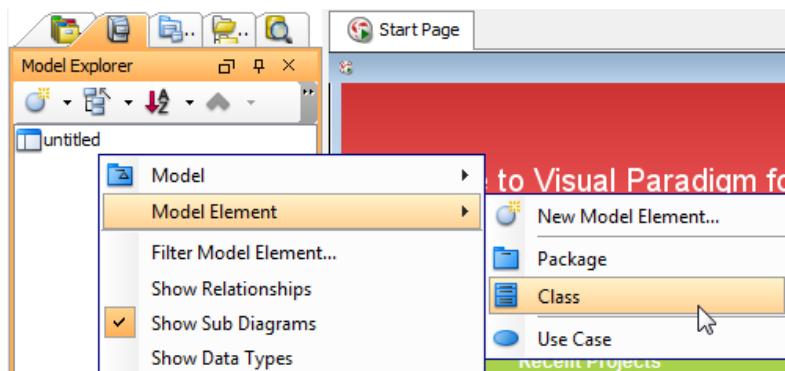
In **Options** dialog box, click **General** and select a number from the drop-down menu of **Backup level** under **Project** tab. The number of backup level represents the amount of backup copy that is produced after a project is saved.



Select the amount of backup level in **Options** dialog box

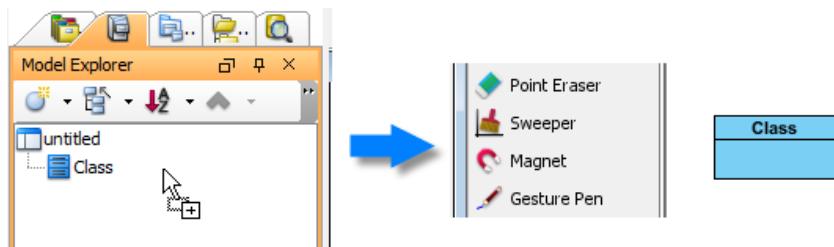
Model element and view

A model element is an elementary component of a model. It will be created when you create a shape on a diagram, or when we create one directly through the **Model Explorer**. An example of model element is class.



Creating a model element through Model Explorer

You can visualize an existing model element by dragging and dropping a model element from Model Explorer to diagram. We call the visualized form of model element a view, or a shape, depending on whether we want to emphasize the differences against model element, or we want to focus on diagramming operations.



Creating a view from an existing model element through drag and drop

When developing context-based diagrams, you will reuse a model element in different diagrams, resulted in creating 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 a view of a model element

If you want to open a diagram from a model element, right click on the model element in **Model Explorer** and select **Show View...** from the popup menu. Then, select the view to open in the **Show View** dialog box and click **Go to View** to open the diagram.

Master view and auxiliary view

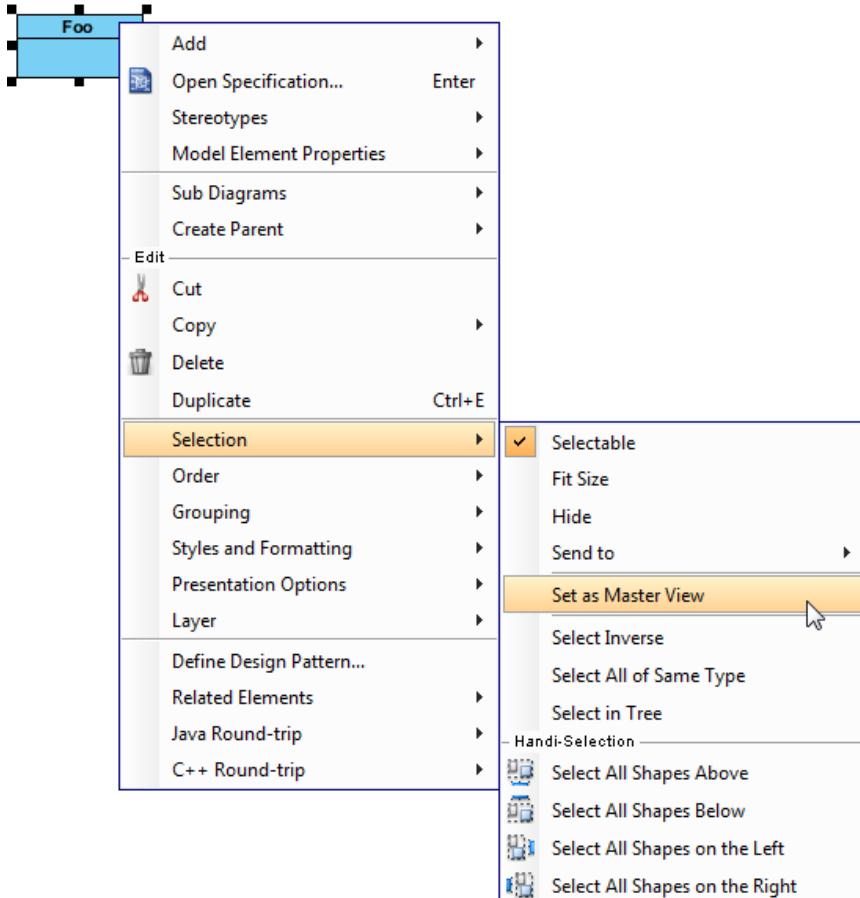
When you create a shape, it creates a model element as well as a view. We call the first view to be created from a model element a master view. For subsequent views created from the same model element, they are called auxiliary views. A model element can have unlimited number of auxiliary views. Yet, it can have at most one master view.

A master view differs from auxiliary view in that master view affects the parent assignment of the owned model element, while auxiliary view does not have such power. For example, if there is a package with a use case in it, and you try to drag the use case out of the package, the use case's parent will be changed from package to root. However, if the use case is an auxiliary view, moving it to another parent will have the package as its parent, remaining unchanged.

Converting an auxiliary view to master view

A model element can have multiple views. Yet, it can only have one master view. You can, however, convert an auxiliary view to become a master view. 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 when the conversion takes place.

To convert an auxiliary view to master view, right click on the auxiliary view and select **Selection > Set as Master View** from the popup menu.



Converting an auxiliary view to master

Creating diagrams

You can create diagrams in different ways:

- Using toolbar
- Using New Diagram dialog box
- Using popup menu of Diagram Navigator

To use toolbar to create:

Click on the icon on the toolbar.

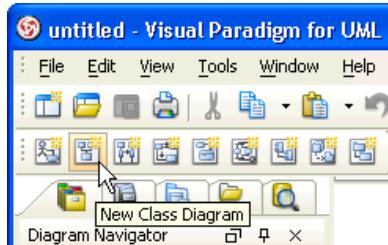


Figure 1-1 New Class Diagram icon on the toolbar

To use New Diagram dialog box to create:

1. Select **File > New Diagram > New Diagram...** from the main menu. The New Diagram dialog box is displayed.

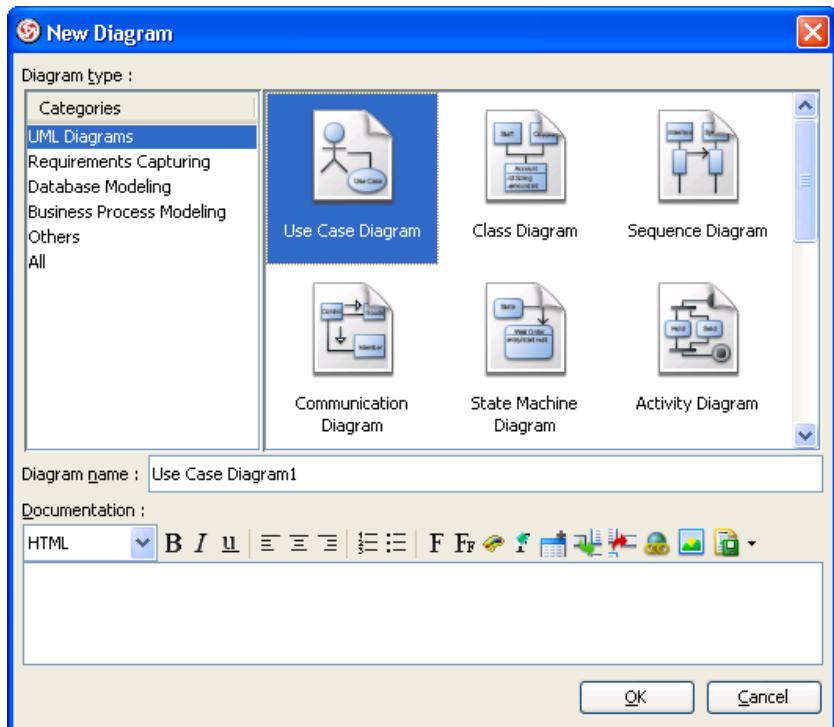


Figure 1-2 New Diagram dialog box

2. Then, select the category and select a diagram type in the category. You should also specify a diagram name. You may also specify the type of documentation.

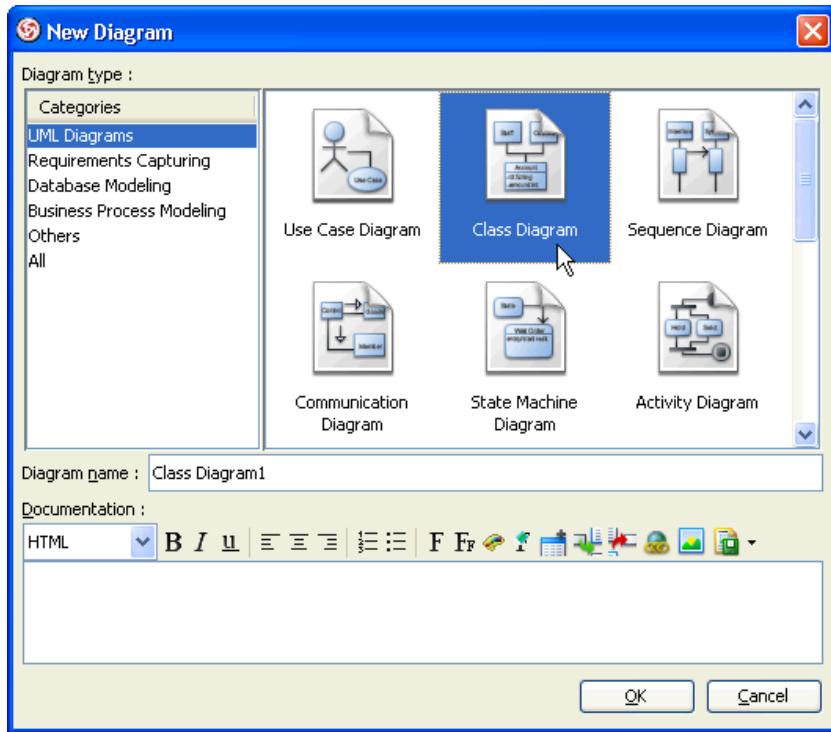


Figure 1-3 Select Class Diagram

To use the popup menu of Diagram Navigator to create:

Right click on the diagram type node in Diagram Navigator and select **New Class Diagram** in popup menu.

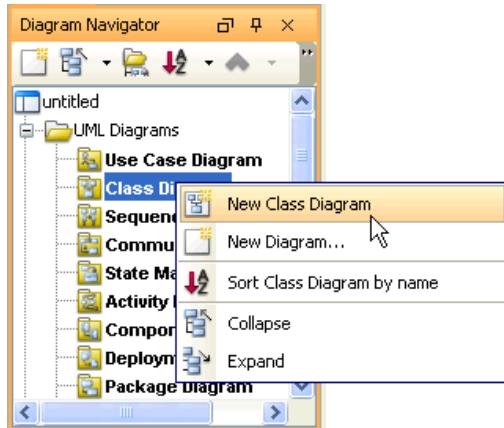


Figure 1-4 Select New Class Diagram from popup menu of Diagram Navigator

Drawing shapes

After creating a new diagram, you can create diagram elements using the diagram toolbar. In this section, we will introduce the techniques of how to draw shapes:

- Creating Shapes
- Creating Connectors
- Creating Self-Connection

Creating shapes

To create a shape, click on a diagram element button from the diagram toolbar and click on the diagram pane to create it. The element generated will have a default size.

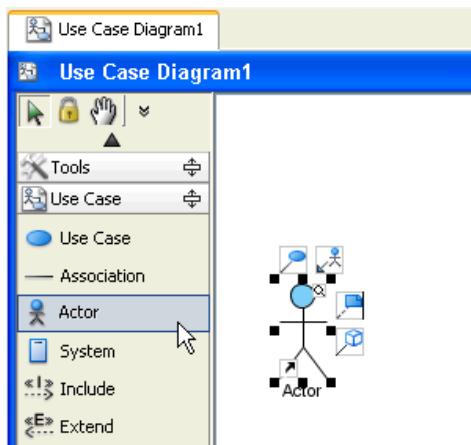


Figure 1-5 Click to Create Shapes

You can also drag a specific boundary before releasing the mouse to define a shape's initial size.

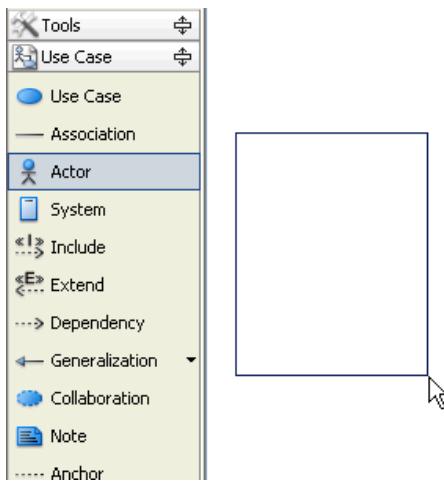


Figure 1-6 Create Shapes with specific size

Alternatively, you can also create a diagram element by dragging a diagram element button then dropping it on the diagram pane.

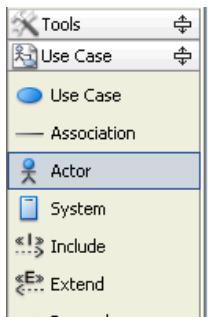


Figure 1-7 Drag and drop to Create Shapes

Apart from that, you can use the diagram popup menu to add a shape.

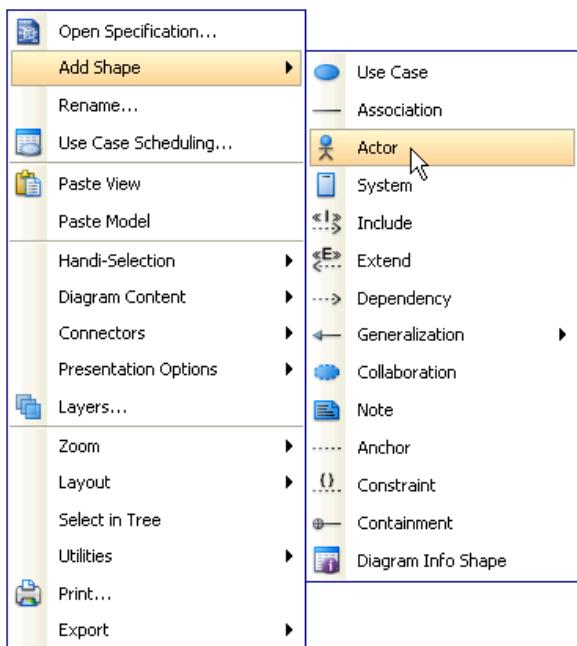


Figure 1-8 Create shapes using diagram popup menu

Creating connectors

To create a connector, select the desired connector from the diagram toolbar and click on the source shape. Drag the connector to the destination shape.

VP-UML provides continuous UML syntax checking. You will see a stop sign when you try to create an invalid connection, e.g. you cannot create a generalization relationship between an actor and a use case.

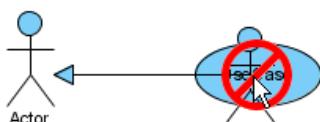


Figure 1-9 Try to create an invalid connection

If the connection is valid you will see a blue rounded rectangle surrounding the destination shape.

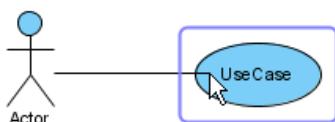


Figure 1-10 Try to create a valid connection

You may also use resource to create connectors.

Click on the Association resource of a shape and drag over the shape you want to connect to. If you release the mouse on an empty space, a shape will be created with the connector.

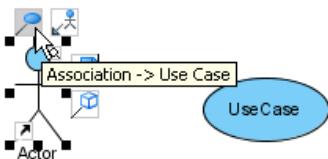


Figure 1-11 Click on resource

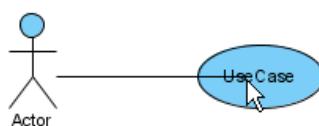


Figure 1-12 Drag over the shape

Creating self-Connection

Some of the shapes can have a connection to itself, for example Self-Association of a Class or Self-Link of an Object in a Communication Diagram. To create a self-connection, click on the connector button on the diagram toolbar and click once on the target object.

Alternatively, you can click on the **Self Association** resource.



Figure 1-13 Create Self-Connection

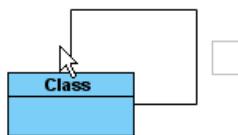


Figure 1-14 Self-Connection

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 no longer needs to go back and forth 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:

- Connection Resource
- Manipulation Resource
- Branching Resource

Connection resource

It is designed for creating elements and making connections.

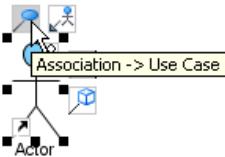


Figure 1-15 Create association with connecting a new or existing use case

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.

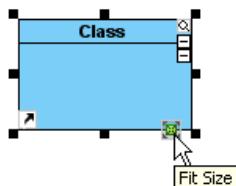


Figure 1-16 Fit size by manipulation resource

Branching resource

Branching Resource helps you to create decision structure in diagram.

Drawing freehand shapes

Freehand Shape is general graphic shapes. There are pen shapes, pencil shapes, and some regular shapes like circle, rectangle and arrow. You can use freehand shape for annotating diagram.

For example, you can use freehand shapes to represent some classes which come from other libraries.

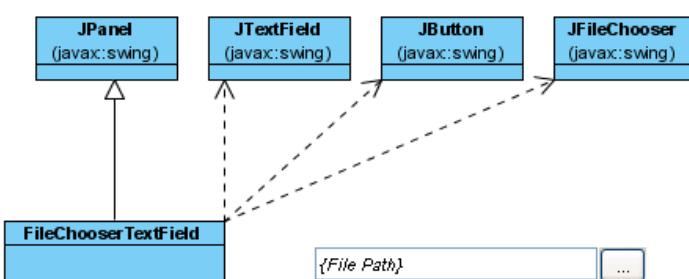


Figure 1-17 Classes come from javax.swing

Use Pencil to highlight the classes from javax.swing.

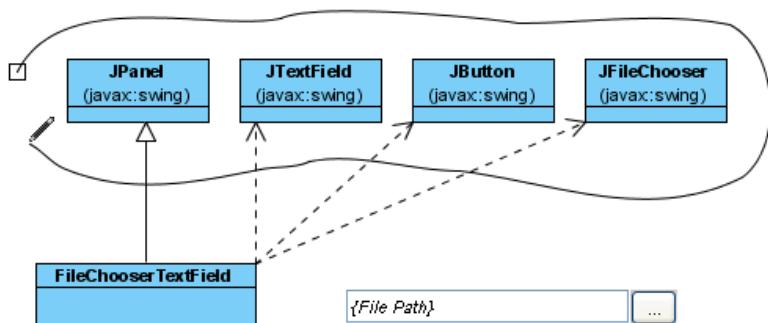
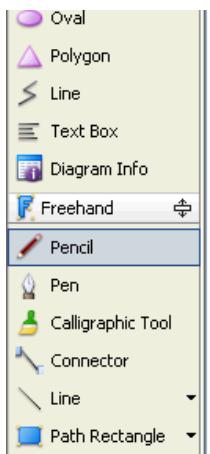


Figure 1-18 Pencil

Use Word Art to show text.

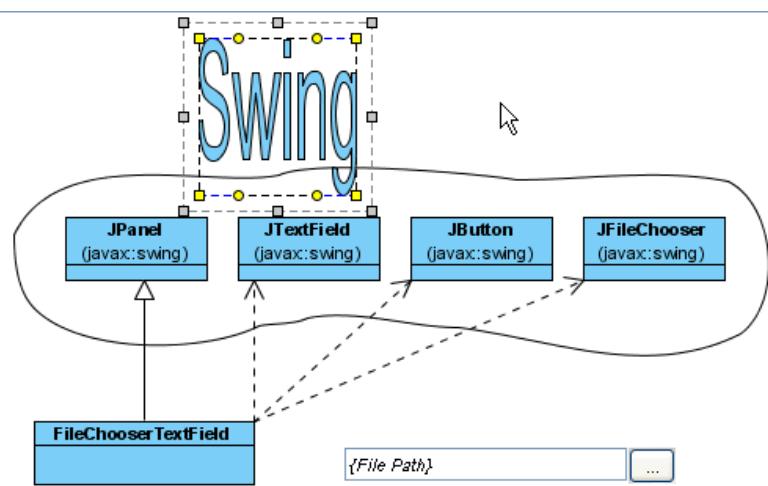


Figure 1-19 Word Art

Set freehand shapes style

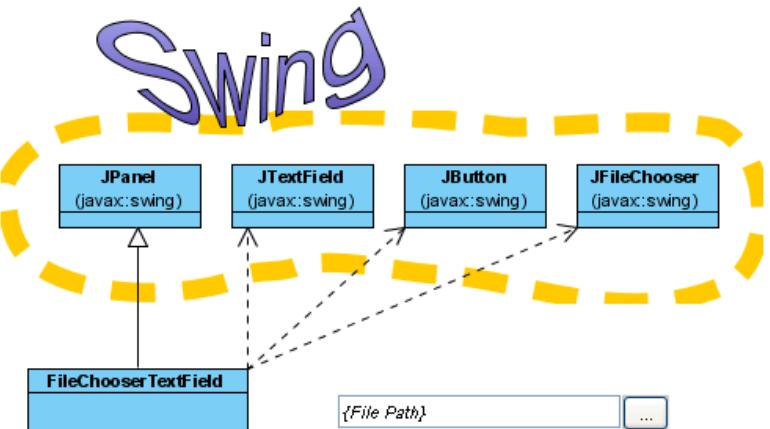


Figure 1-20 Styled freehand shapes

Changing package header

You can specify the parent package of any diagram through Package Header.

Package header when diagram created

When diagram create, the package header will be focused and let you to specify the parent package of the diagram. (Specify the package by entering the fully qualifier name of the package)

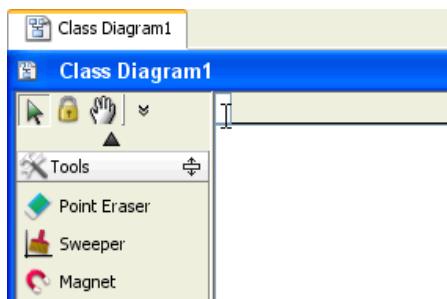


Figure 1-21 Specify parent package in package header

After you confirm the parent package. The diagram will be named same as the package.

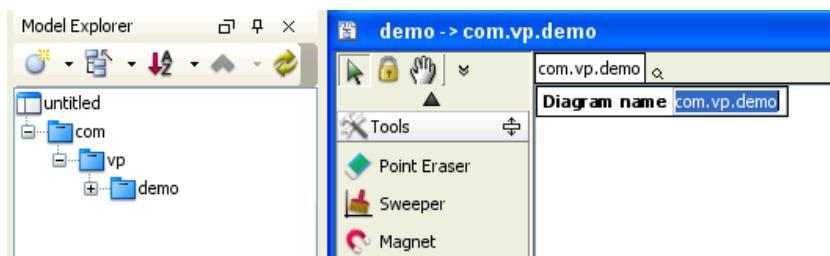


Figure 1-22 Diagram name will be same as fully qualify of parent package

You can modify the diagram name. (The parent package won't be changed to follow diagram name)

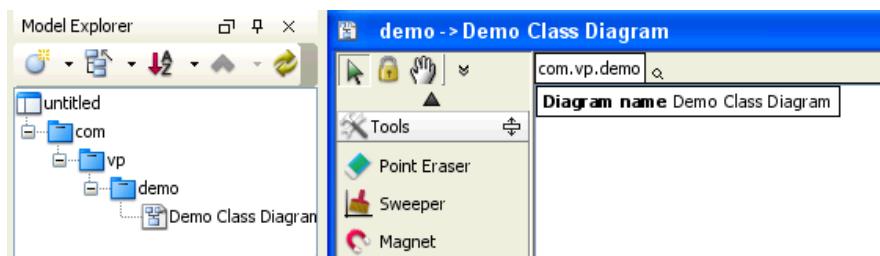


Figure 1-23 Diagram name can be edited to be different with parent package

You can open specification of parent package.

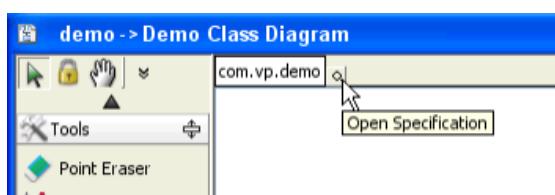


Figure 1-24 Open Specification

You also can change the parent package of the diagram by double click the package header.

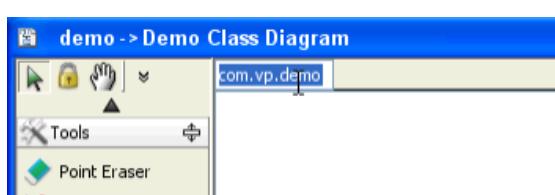


Figure 1-25 Double-click to change parent package

A new package will be created if the package name you entered does not exist in project. If the old parent package does not contain any elements, it will be deleted, so, the documentation (or other properties) or old parent package will be lost.

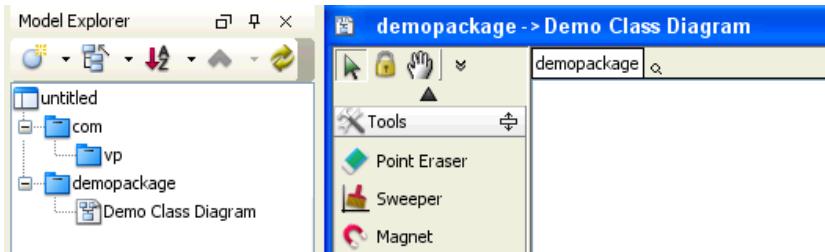


Figure 1-26 Old parent package may be deleted if new parent package is specified

You can show/hide the package header by diagram popup menu: **Presentation Options > Show Package Header**

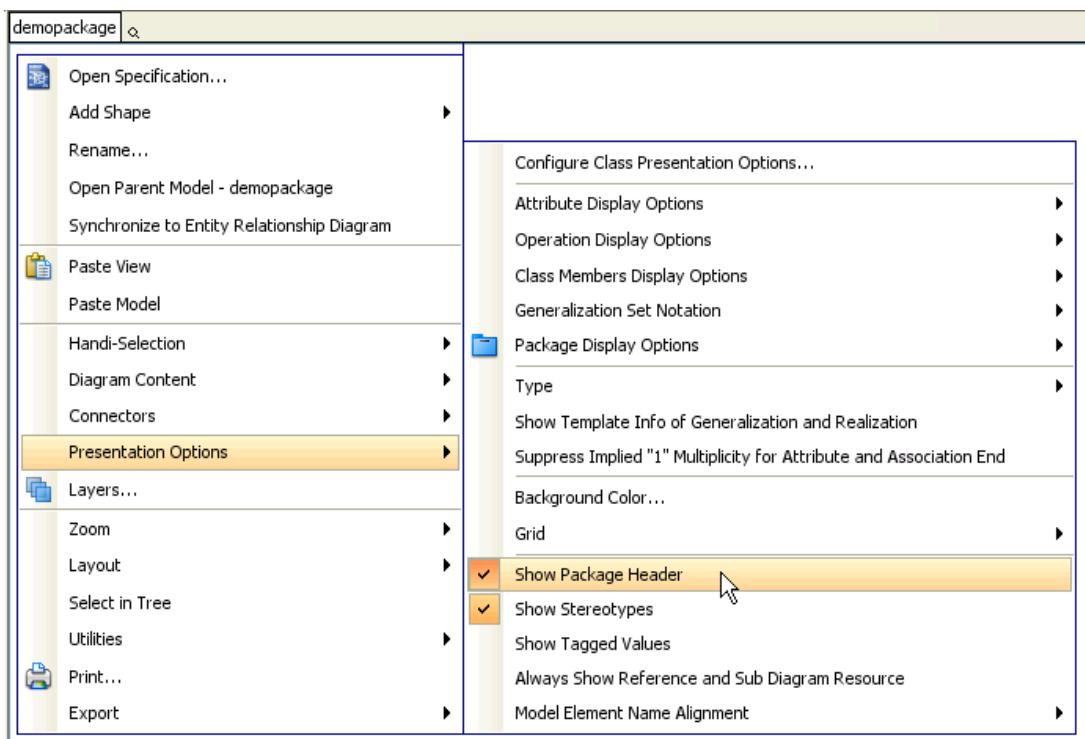


Figure 1-27 Show/hide package header

Justify shape name

In Visual Paradigm, shape name is aligned center horizontally, and top or middle vertically, depending on the characteristic of shapes. It is possible to realign the shape name, which is useful for language that is written from right to left, like Modern Hebrew.

Adjusting shape name's position

1.Right click on the diagram and select **Presentation Options > Model Element Name Alignment** from the popup menu. There you can choose the desired position to display the name.

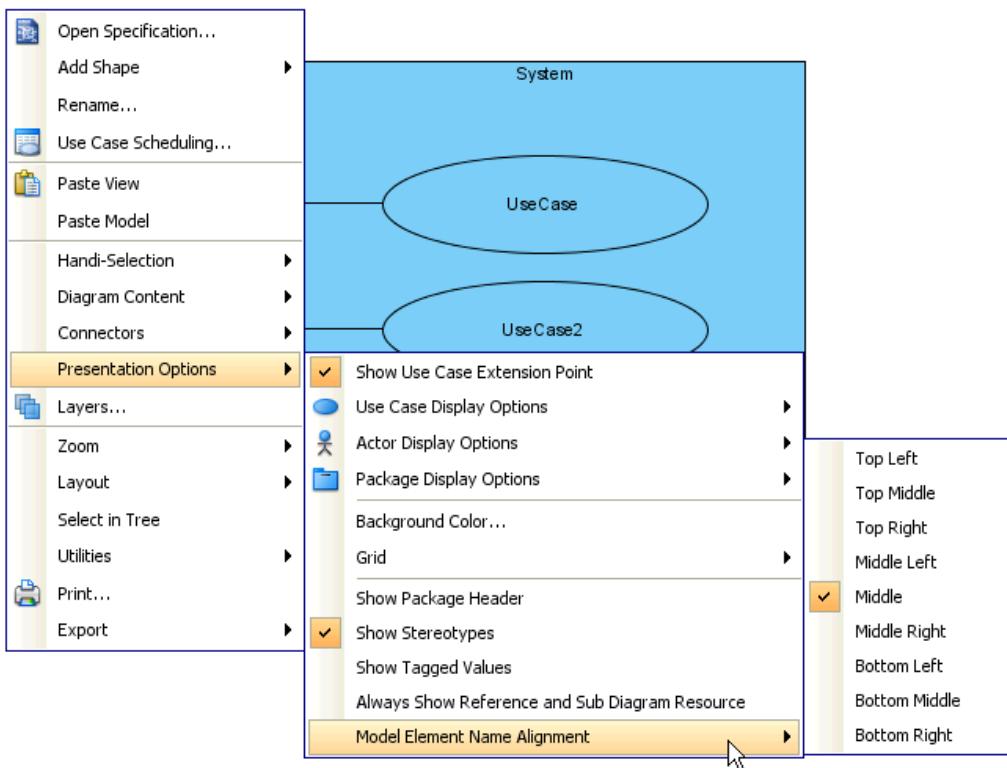


Figure 1-28 Select model element name alignment on diagram

2.This is the result of "Middle Right". All shapes which Model Element Name Alignment is Follow Diagram are now aligned middle right.

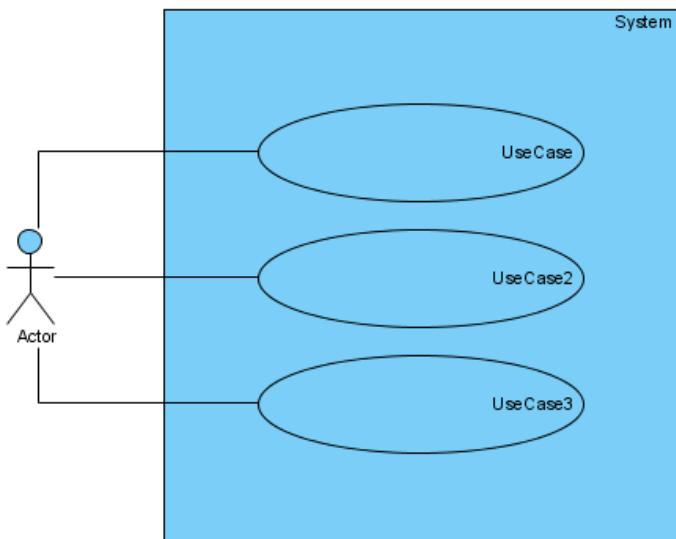


Figure 1-29 Align to Middle Right

Besides the diagram-wide setting, you can also set for specific shape. To do this:

- 1.Right click the shape and select **Presentation Options > Model Element Name Alignment** in the popup menu, and select the desired alignment option.

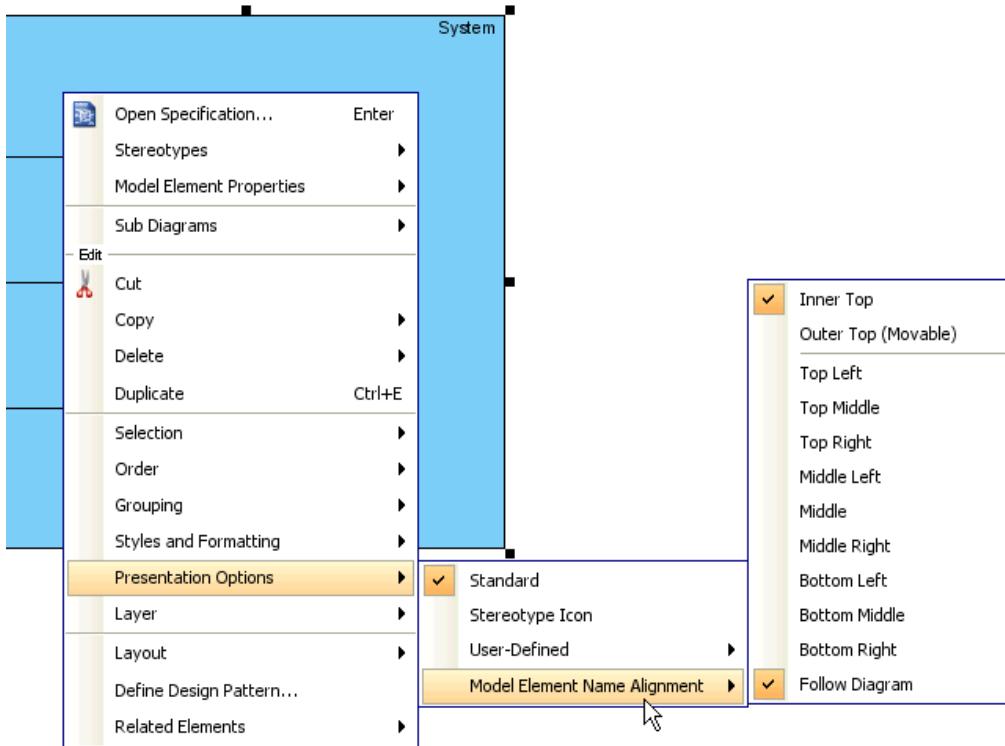


Figure 1-30 Select model element name alignment on shape

- 2.This is the result of setting System's name to align top middle.

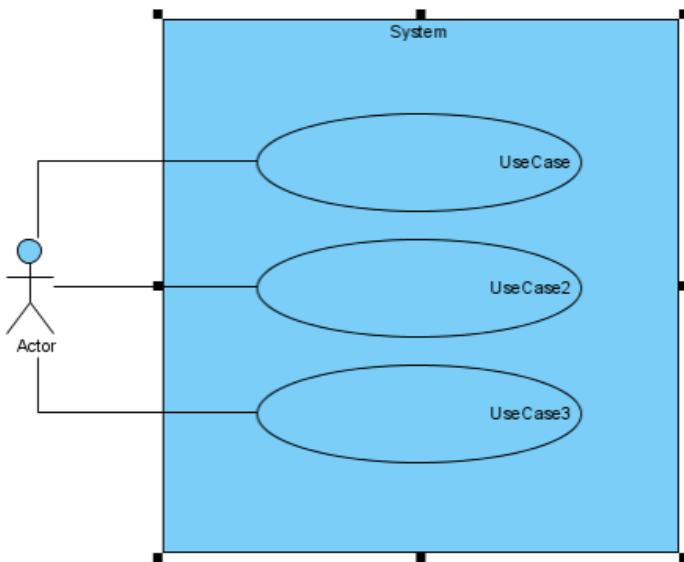


Figure 1-31 Selected shape's name is aligned to Top Middle

Besides setting of current diagram, you can also set for future diagrams.

- 1.Select **Tools > Options...** to open Options dialog box.
- 2.In the Options dialog box, open the **Diagramming** category, select the **Appearance** tab. You can select Model Element Name Alignment.

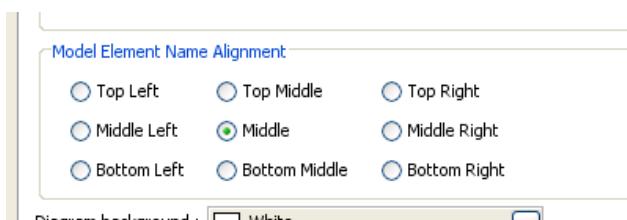


Figure 1-32 Define default Model Element Name Alignment in Options dialog

Exceptions

Some of the shapes are not able to justify name due to their characteristic. There are mainly two kinds of shapes. First, shapes with floating name label (freely movable) or with label not put inside the shape cannot be justified. Actor, Initial Pseudo Node and BP Start Events are examples of this kind of shape.



Figure 1-33 Floating name label

Container shapes have a limited scope of displaying names since their “bodies” are for containing other shapes, and are not available for positioning names. Package, State and System are example of this kind of shape.

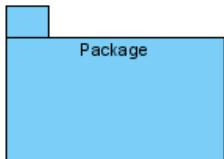


Figure 1-34 Container shape

Enable and disable minimum shape size

All shapes have their default minimum size. Users are not allowed to resize shapes to smaller than the minimum size, which help to ensure the shapes are compact yet clear enough to be seen on a diagram under normal conditions. The minimum size can be determined by fitting a shape's size.

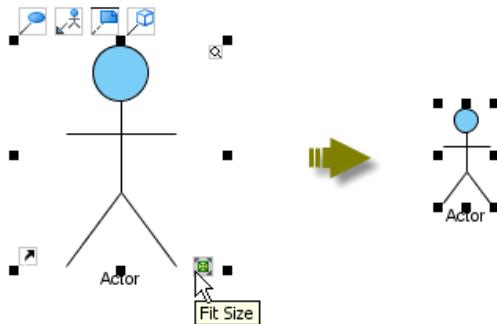


Figure 1-35 Minimum size can be determined by fitting shape size

Now, it is possible to disable such checking such that shapes can be resized to very small in size, ignoring the minimum size. For example:

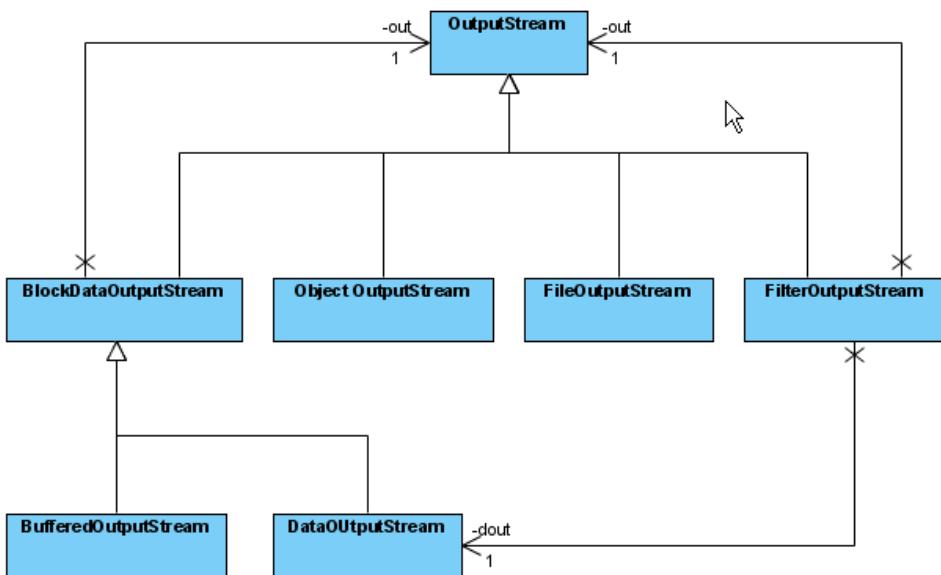


Figure 1-36 Example, shapes are fitted size

- 1.To disable the minimum size checking, select **Tools > Options...** to open **Options** dialog box.
- 2.In the Options dialog box, open the **Diagramming** category, select the **Appearance** tab and uncheck **Enable minimum size**. Click **OK** to confirm.



Figure 1-37 Disable minimum size checking in Option dialog

- 3.From Now on, you can resize the shapes to very small in size.

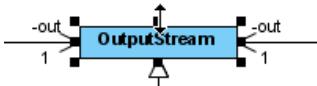


Figure 1-38 Resize to very small

- 4.And then select all the classes and select **Same Height**.

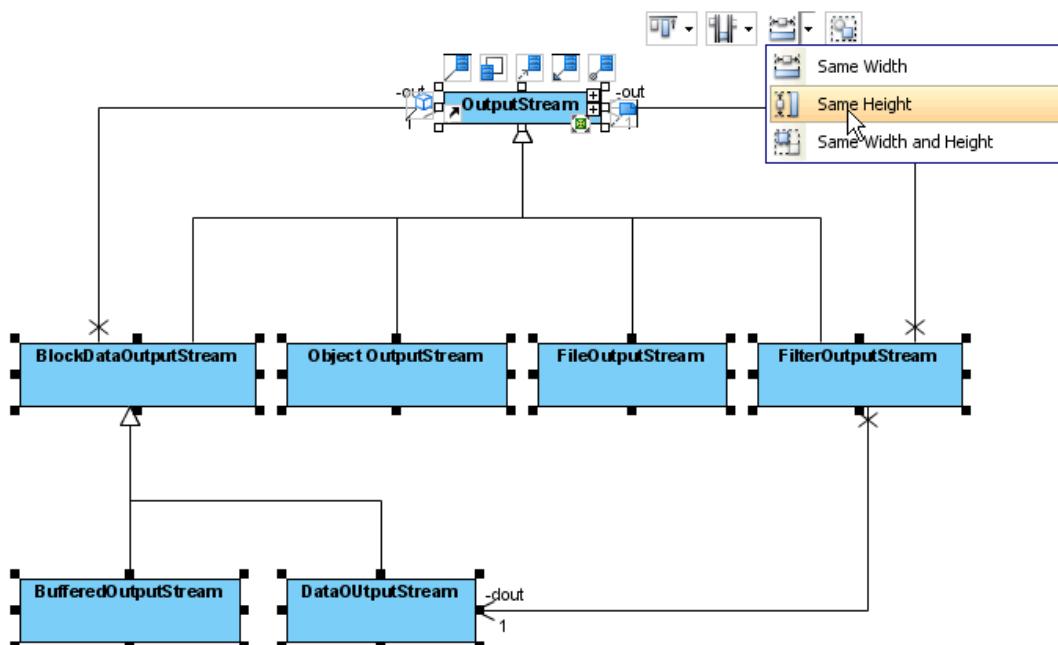


Figure 1-39 Same height as smallest size shape

- 5.This is the final result:

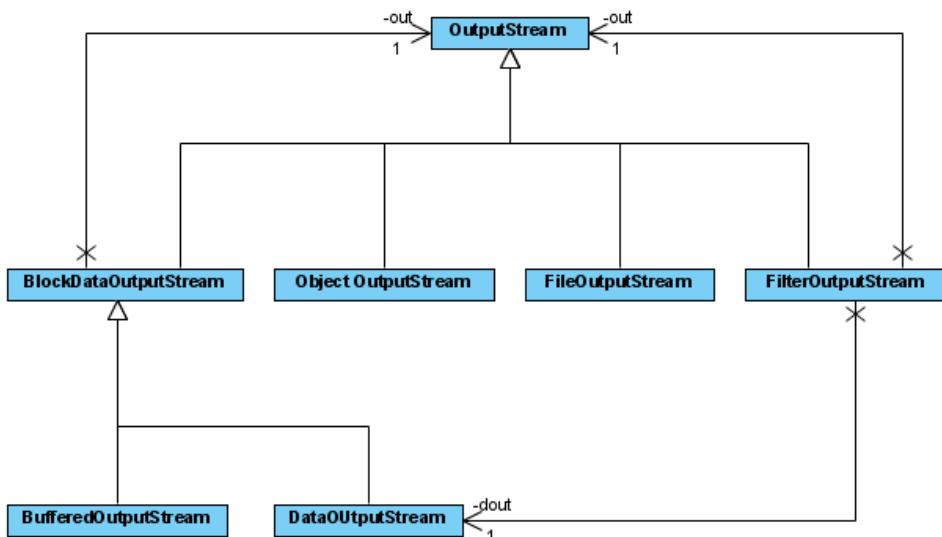


Figure 1-40 All shapes are small size

Undo and redo

When you create and edit a diagram, you may make mistakes like accidentally deleting a diagram element. You can use the Undo function to cancel the previous action. On the other hand, you may re-perform the action using the Redo action. The undo/redo feature in VP-UML is diagram based.

Undo

You can roll back undesirable changes by performing Undo. To undo an action, perform one of the following actions:

- Select **Edit > Undo** from main menu.
- Click on the **Undo** button  on toolbar.
- Press **Ctrl-Z**.

Redo

This feature is to re-perform actions that were just undone. To redo an action, perform one of the following actions:

- Select **Edit > Redo** from main menu.
- Click on the **Redo** button  on toolbar.
- Press **Ctrl-Y**.

Showing name for undo and redo action

Since Undo and Redo are easy and fast ways to cancel or restore the actions we've done in the project, but it's somehow inconvenient to rely on our own memories of the actions done.

You can know the Undo and Redo name from main menu

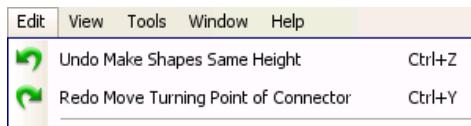


Figure 1-41 Menu shows Undo/Redo name

or tooltip shown on toolbar button

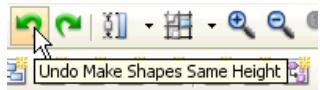
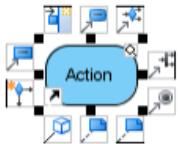


Figure 1-42 Toolbar button's tooltip shows Undo/Redo name

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.



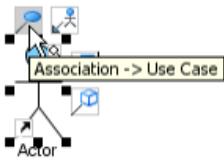
Resource centric interface on action

NOTE: To check or uncheck the options (including Resources, Group Resources, Extra Resources and Generic Resources Only) on resource centric interface, select **View > Resource Centric** from the main menu.

Creating shapes through resource centric interface

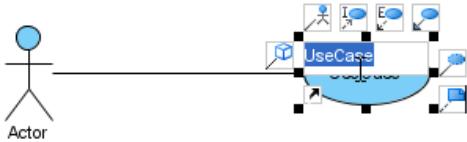
Instead of clicking shapes one by one from the diagram toolbar, you can create another shape from an existing shape on the diagram.

1. Move the mouse on a shape and select a resource icon out of resource centric interface.



Create a use case with resource centric interface

2. Drag the resource icon and release it until reaching to your preferred place.
3. As a result, another shape is created and linked with the previous shape.

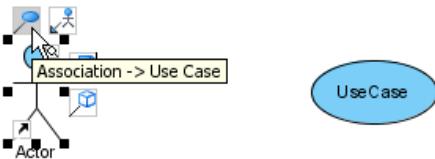


Release the mouse to create use case

Connecting shapes through resource centric interface

For relating two shapes together, you have to link a shape to another. With resource centric interface, you can not only link them up, but also select an appropriate relationship for them.

1. Select two shapes from the diagram toolbar and drag them on the diagram individually. Move the mouse on one shape and select a resource icon out of resource centric interface.



Create Use Case with Association

2. Drag the resource icon and release it until reaching to another shape.
3. As a result, two shapes are linked together.

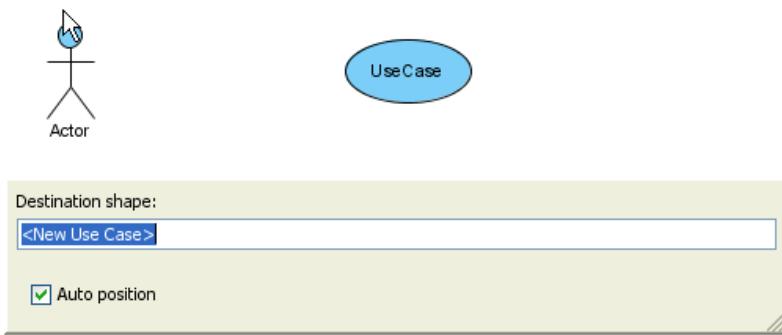


Release the mouse to create Association connecting with the Use Case

Using quick connect

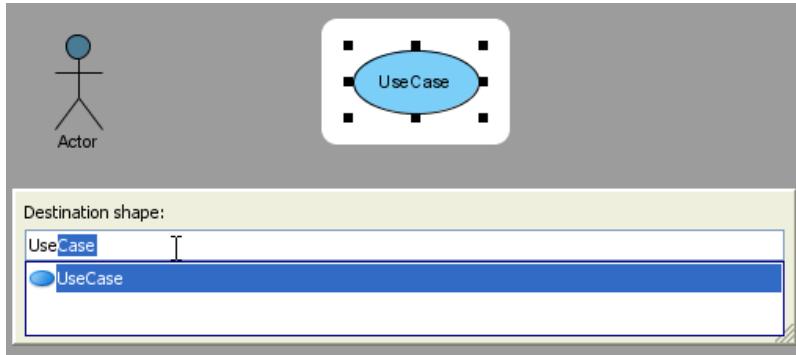
With quick connect, you are able to search the preferred shape and connect to it on the diagram accurately and quickly.

- After creating some shapes on diagram, move the mouse on a shape and click a resource icon out of resource centric interface. The **Quick Connect** dialog box will be shown subsequently.



Quick Connect dialog box

- 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 display.
- After you click a shape's name on the drop-down list, the shape will be spotlighted on the diagram immediately.



Use Case is spotlighted

- Confirm the selection, press **Enter**. The two shapes will be linked automatically.



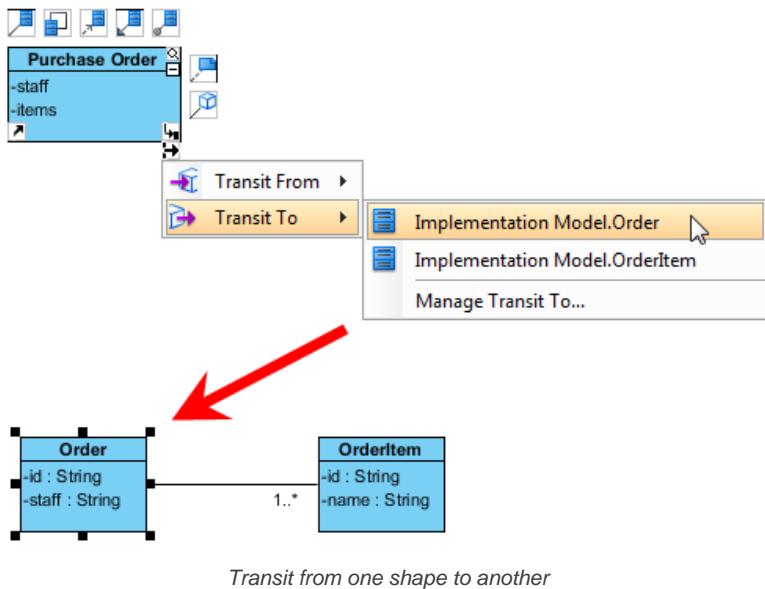
An actor is linked with a use case

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, the **Quick Connect** dialog box will not pop out even when you click the resource icon.

Managing transition of shapes

Model transiton 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 Transistor> 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.

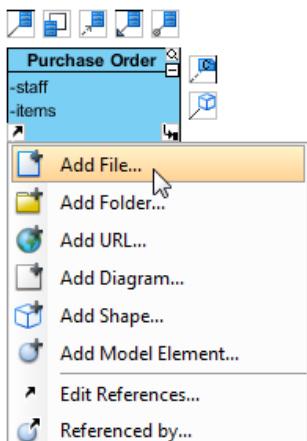


NOTE: Transition does apply not only on two shapes on the same diagram, but also two shapes in different diagrams.

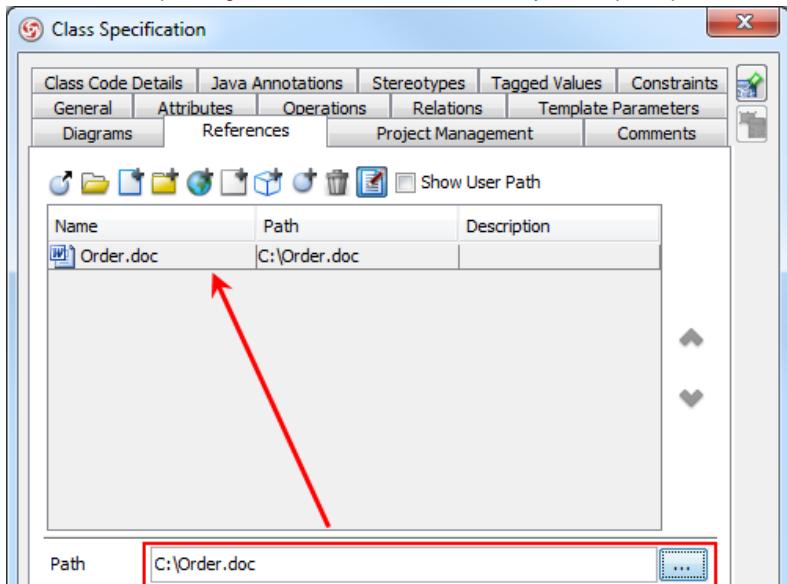
Adding and opening reference

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.

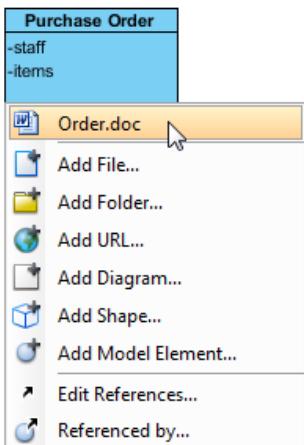


2. Insert the corresponding resource in **References** tab of your shape's specification dialog box.



Insert a file in Class Specification dialog box

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.

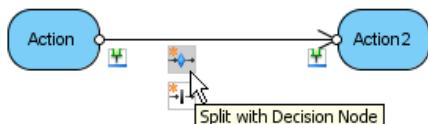


The newly created reference

Splitting connection by shape

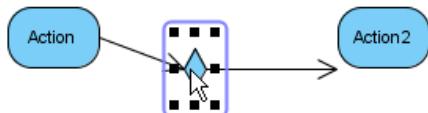
Resource centric supports some connectors (e.g. Control Flow in Activity Diagram) with splitting the connector by adding another shape.

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.

Creating structure

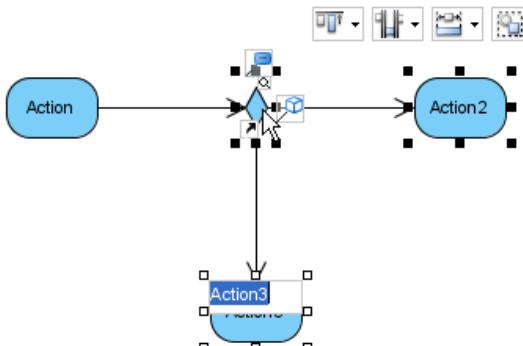
Resource centric interface supports some models (e.g. Action in Activity Diagram) with creating a structure of models. Instead of creating shapes one by one, branch resource icon helps speed up the shape creation process; you thereby can create several shapes simultaneously.

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.



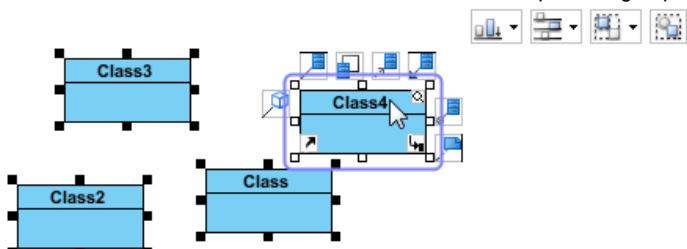
Create Decision Node and Actions

NOTE: There is an orange asterisk on the branch resource icon.

Group selection resource

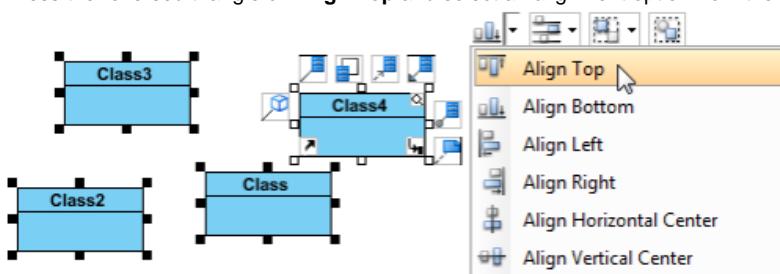
When a great amount of shapes are displayed on the diagram disorderly, resource centric interface can support alignment and grouping after selecting several shapes.

1. Select several shapes on diagram.
2. Move the mouse on the name of one of the 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

Tagged values

In UML 1.1, stereotypes and tagged values were used as string-based extensions that could be attached to UML model elements. In UML 2.x, stereotypes and tagged values will be defined in Profile, that can provide more structure and precision to the definition. VP-UML supports defined tagged values in stereotype, also support tagged values without stereotype.

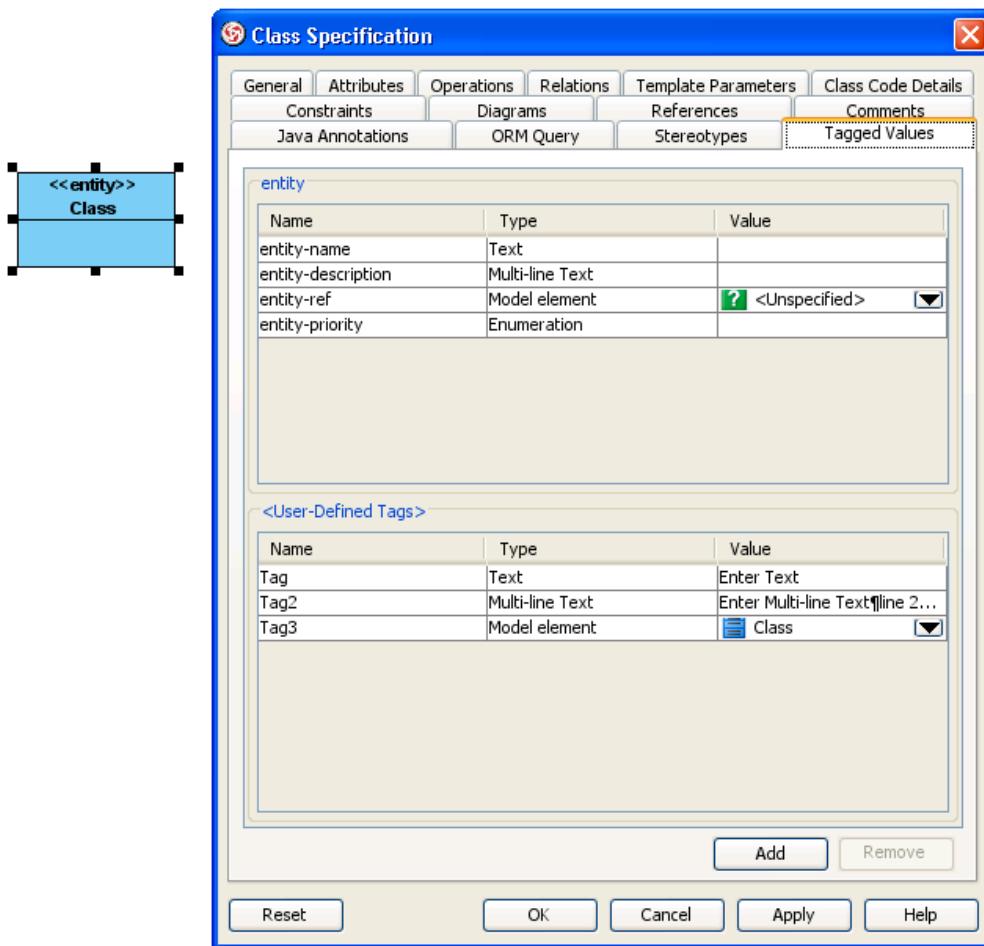


Figure 1-66 Stereotype-defined/user-defined tagged values are shown on Specification dialog of its owner model element

Adding user-defined tags

1. To define tagged values in model element. Open the specification of a model element.

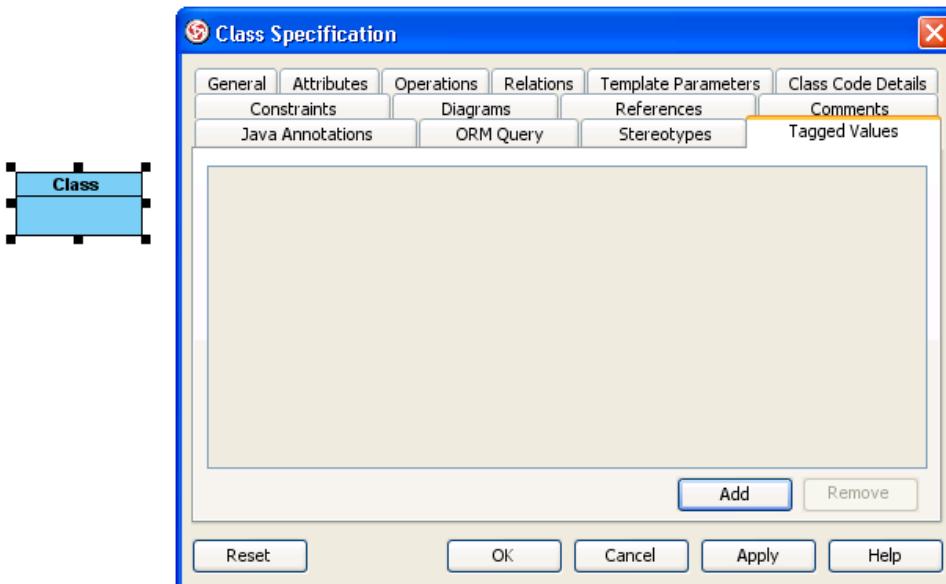


Figure 1-67 Open specification dialog of owner model element to create tagged value

2. Click on **Add** button, you can create tagged value with 3 types of value: **Text**, **Multi-line Text**, **Model Element**

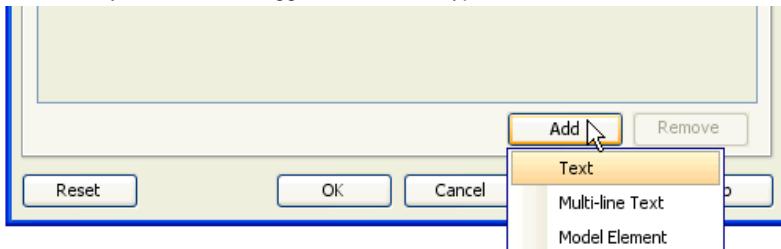


Figure 1-68 3 kinds of tagged value can be created

3. **Text** supports string-based value. You can enter the text on table directly.

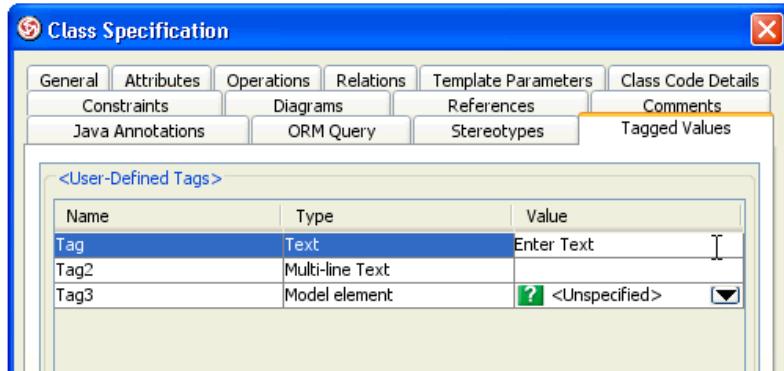


Figure 1-69 Text

4. **Multi-line Text** supports multi-line string. You can enter multi-line text after clicking ... button.

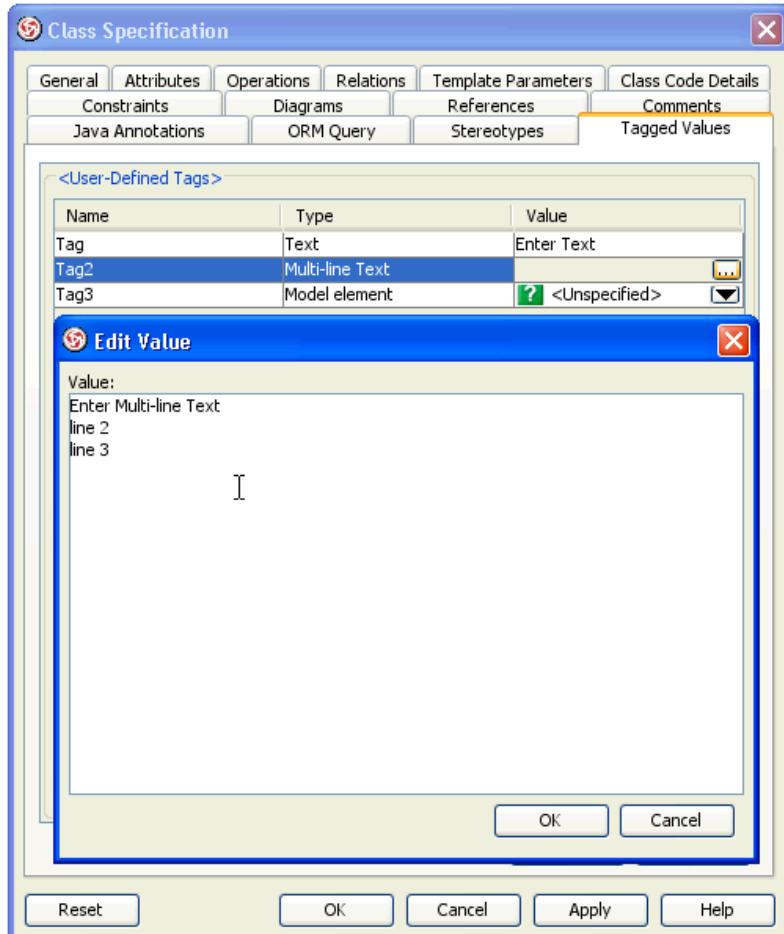


Figure 1-70 Multi-line Text

5. Model element supports reference of model element. You can select the referenced model after clicking the button.

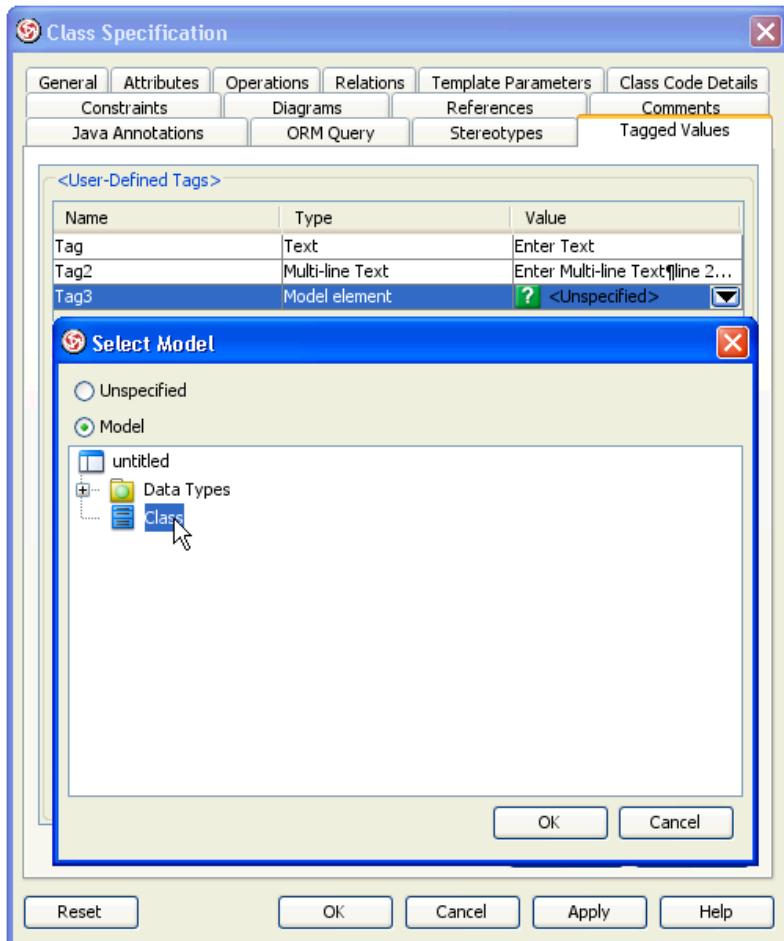


Figure 1-71 Model element

Editing tagged values

You can edit the user-defined tagged value's name, type and value.

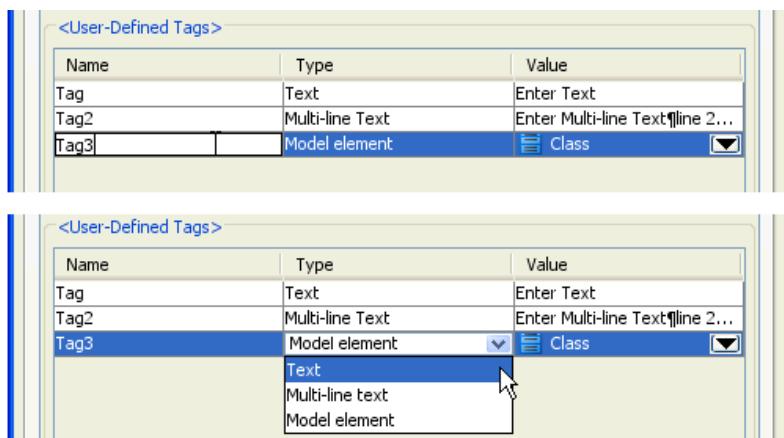


Figure 1-72 Can edit name, type and value on user-defined tagged value

But you can only edit the value of stereotype's tagged value. The name and type are not editable because they are defined in stereotype.

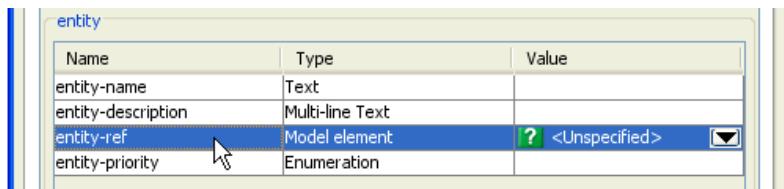


Figure 1-73 Only can edit value on stereotype-defined tagged value

Visualizing tagged values on diagram

Right click on the diagram you want to have the contained shapes' tagged value appear, and select **Presentation Options > Show Tagged Values** from popup menu.

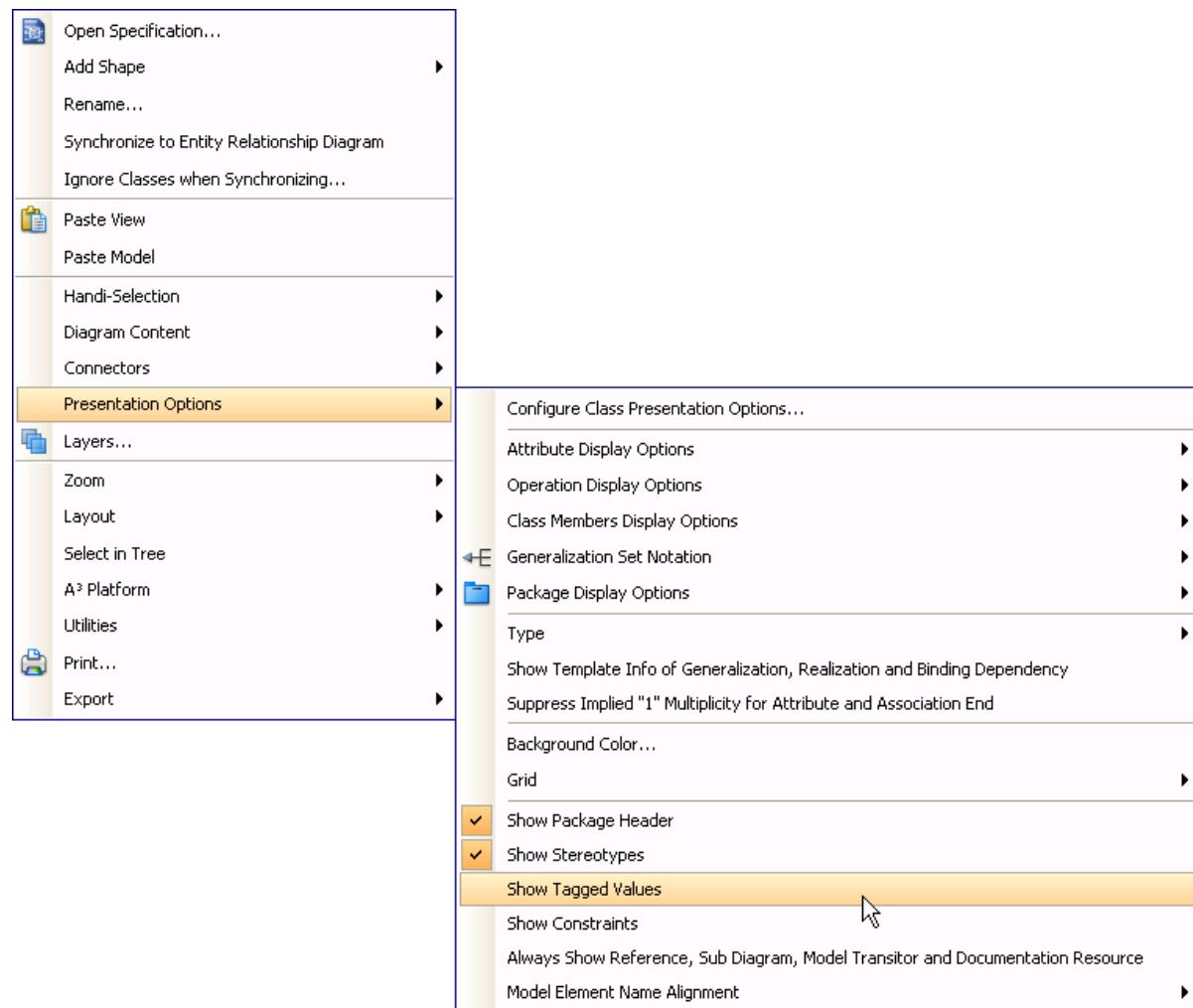


Figure 1-74 Show tagged values

You will see the tagged values, if defined, appear in shape.

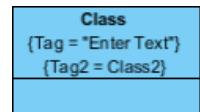


Figure 1-75 Tagged values appear in a class shape

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. It doesn't check your whole diagram automatically, but underlines wrong words with a curve line.

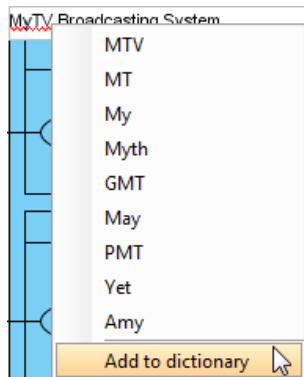
To correct a word

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.

To add 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.

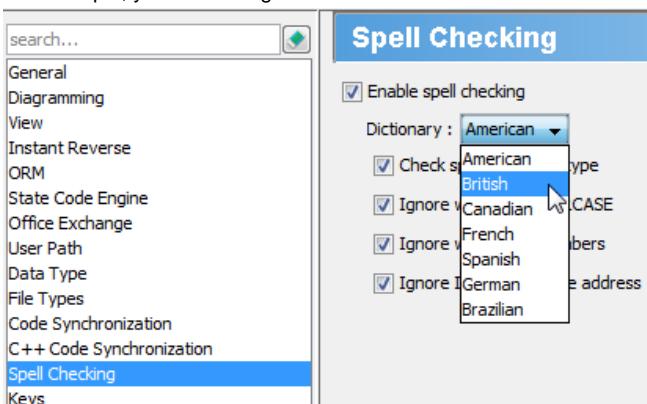


Adding a new word to dictionary

To change 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:

- Click **Tools > Options > Spell Checking**.
- For example, you can change from American to British.

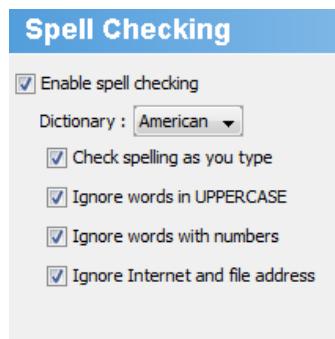


Changing the language of dictionary

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.



Checking other options of spell checking

Automatic diagram layout

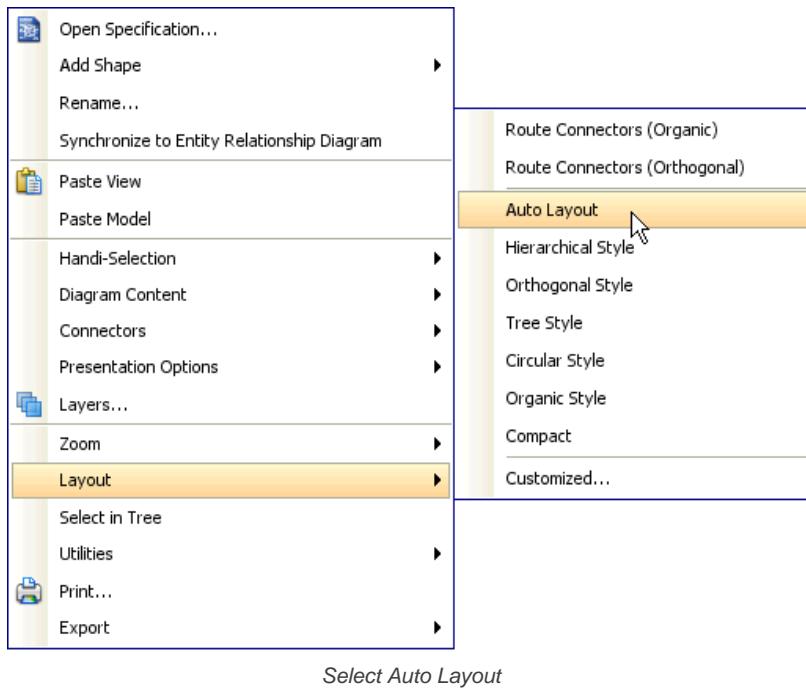
VP-UML provides a layout facility for arranging diagram elements in diagrams. Diagram elements, which are re-layouts, do not overlap; the relationship links, which are arranged, do not cross over each other. Different layout styles and configurable options are provided, which allows extremely flexible 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**, **Smart Organic Layout**.

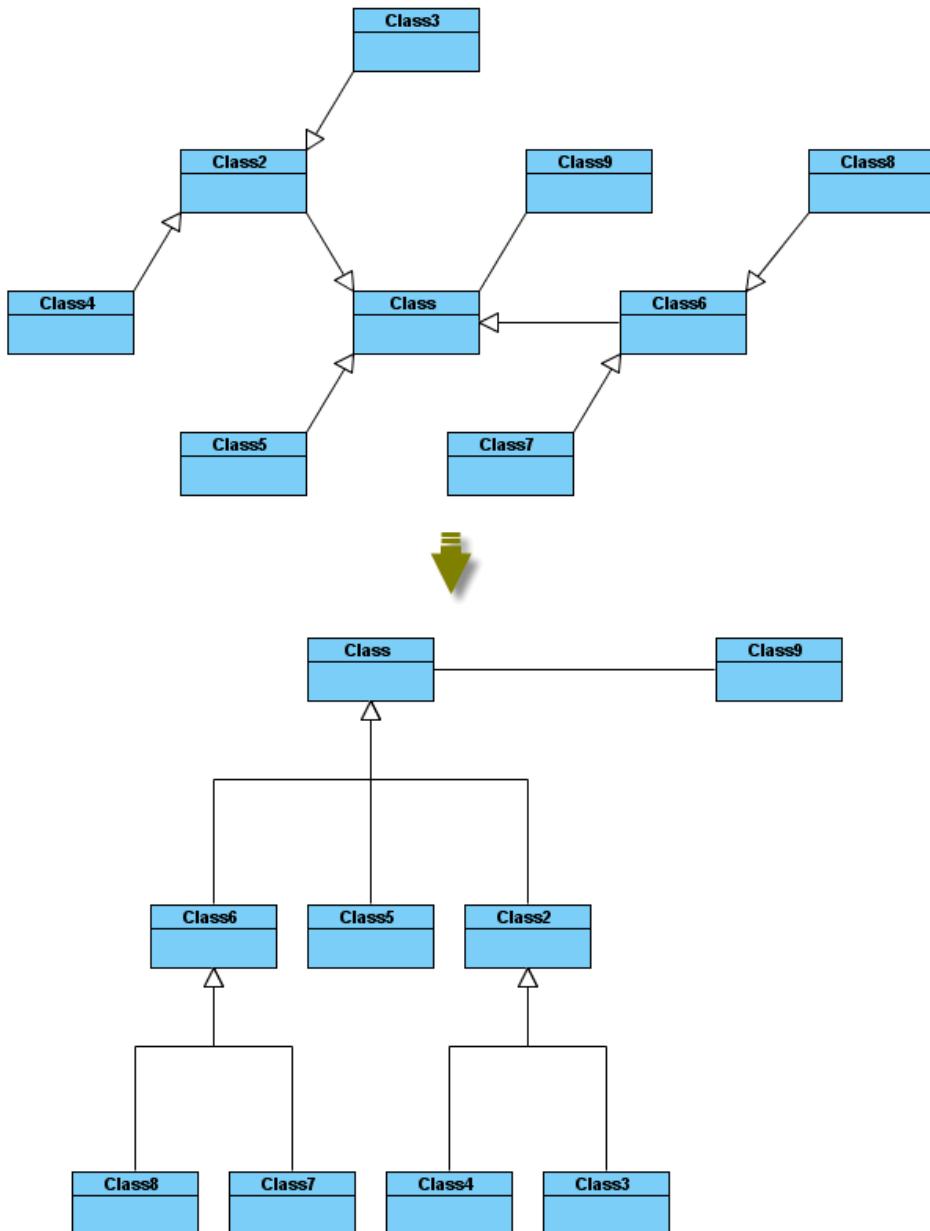
Auto layout

Selecting auto layout signifies that the most suitable layout are 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.



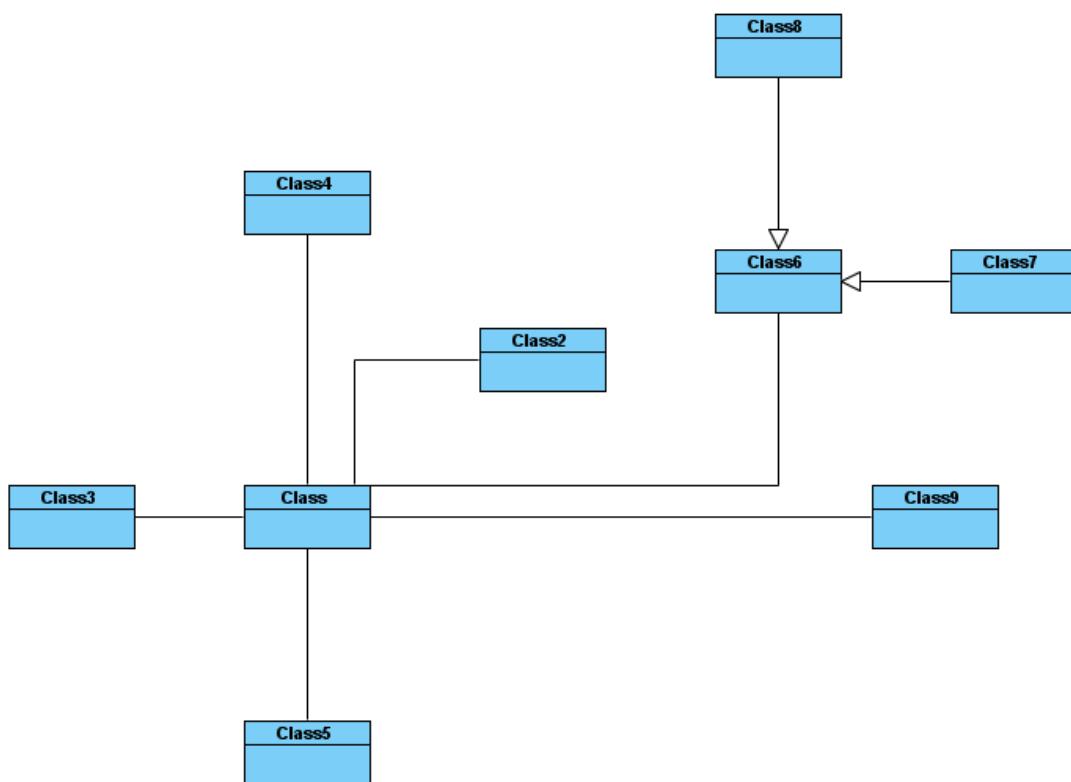
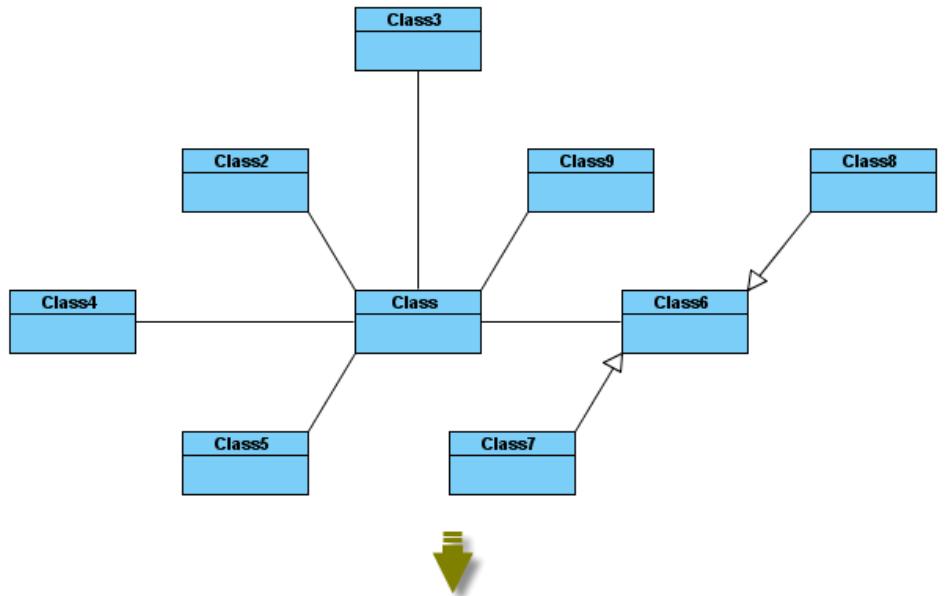
Select Auto Layout

Class diagram (Hierarchy base / factory class diagram)



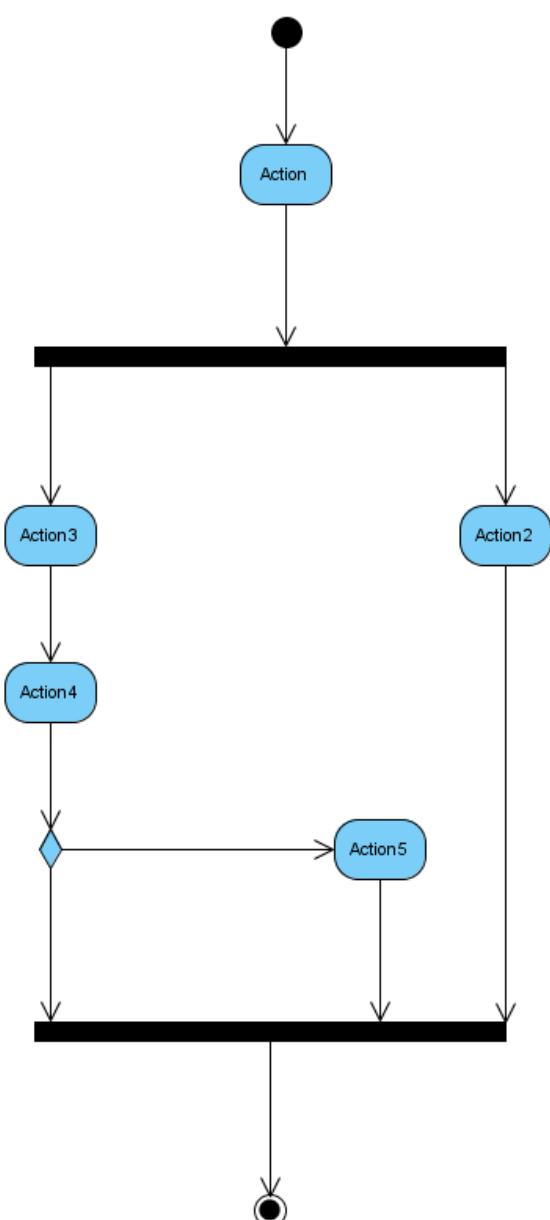
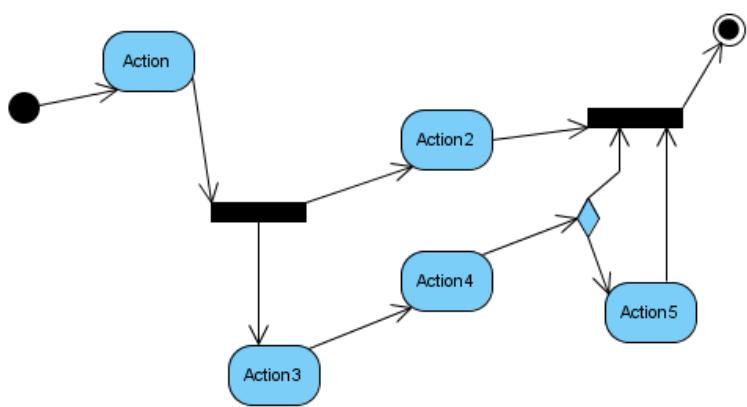
Hierarchy base (Factory class diagram)

Class diagram (Navigation base / mediator class diagram)



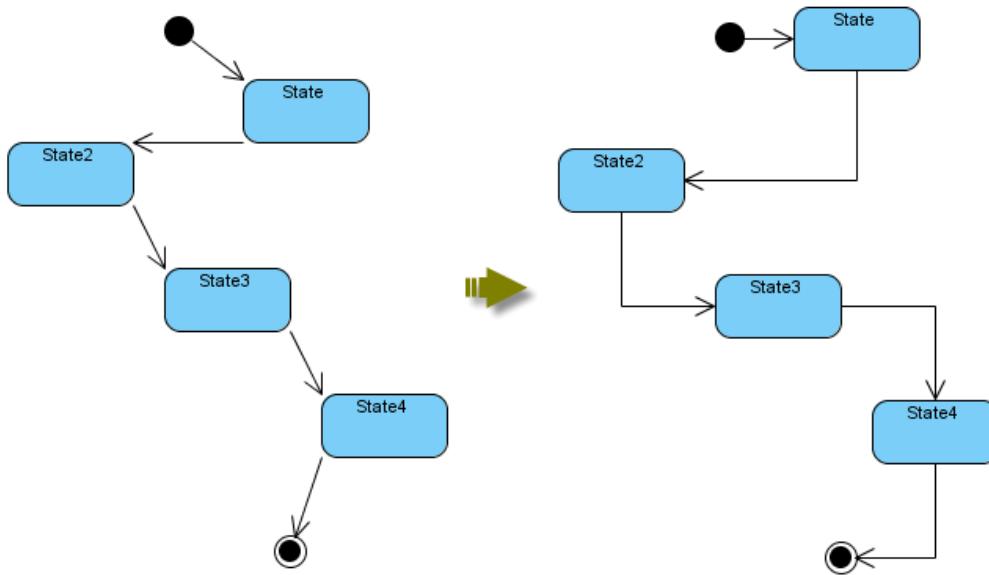
Navigation base (Mediator class diagram)

Activity diagram



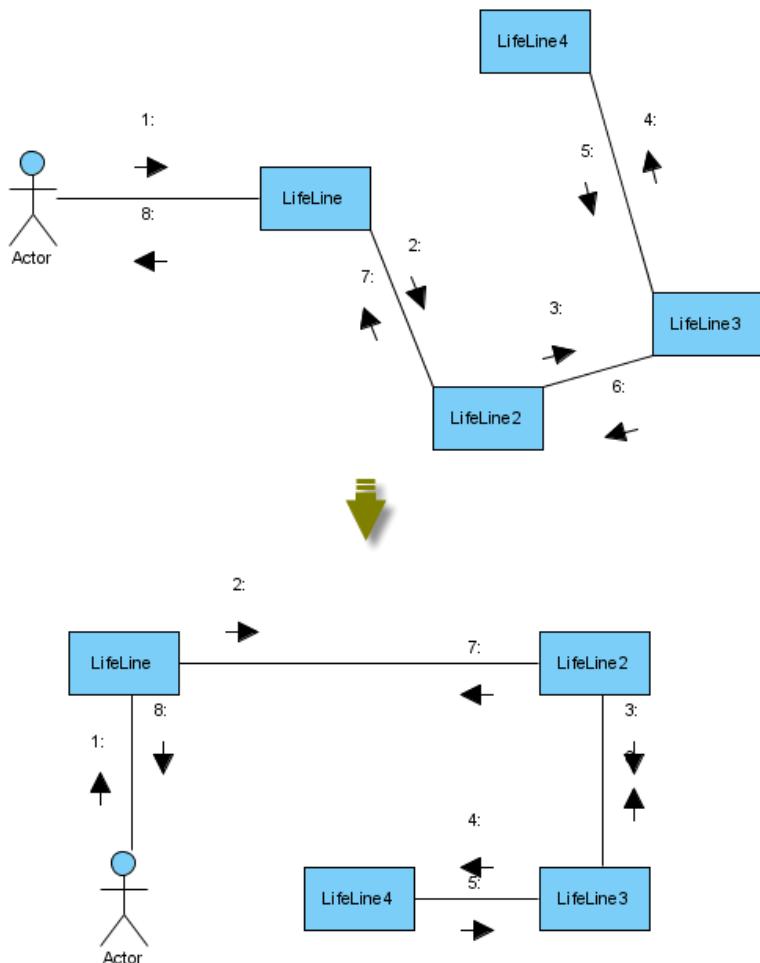
Auto Layout of activity diagram

State machine diagram



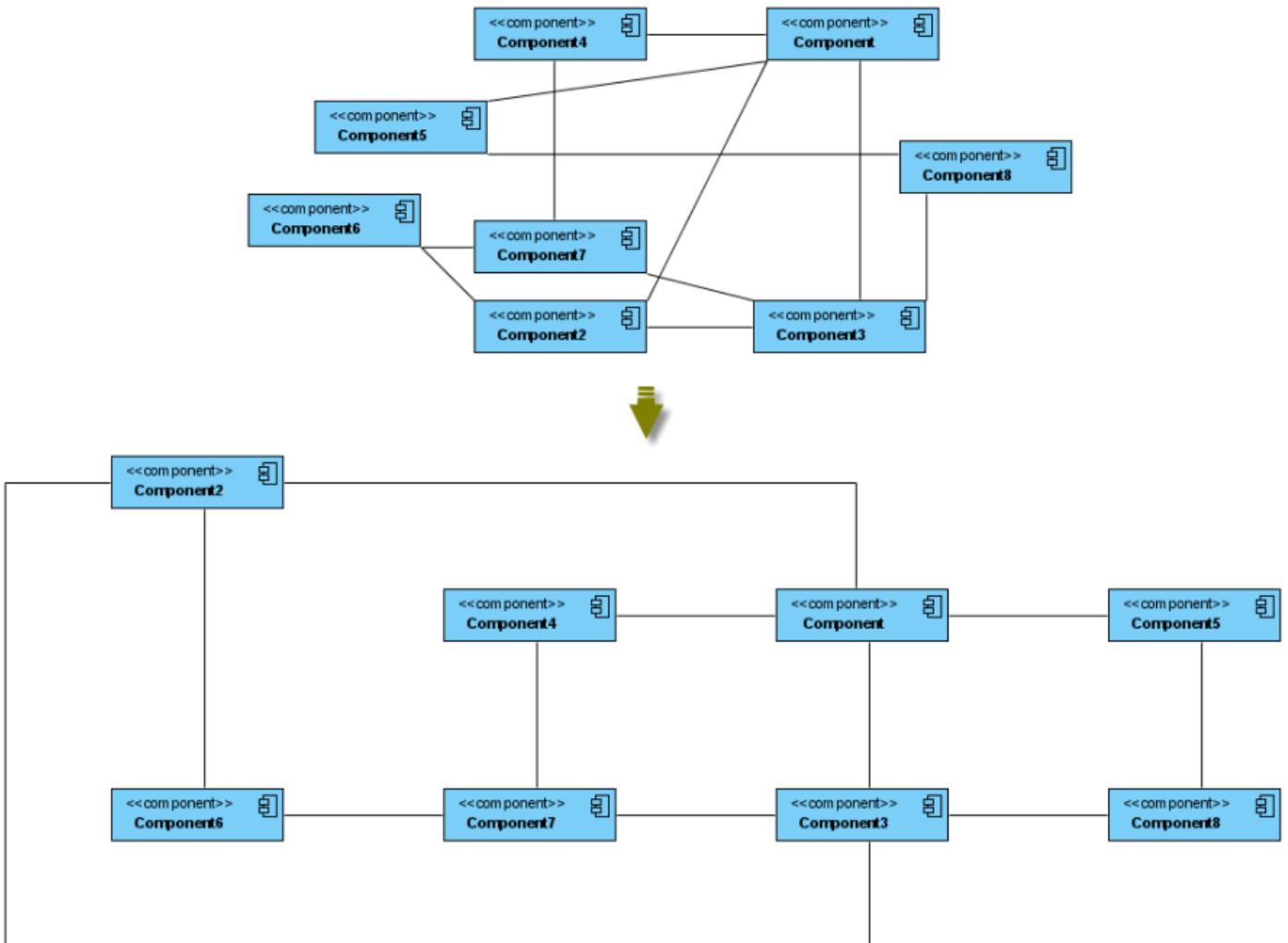
Auto layout of state machine diagram

Communication diagram



Auto layout of communication diagram

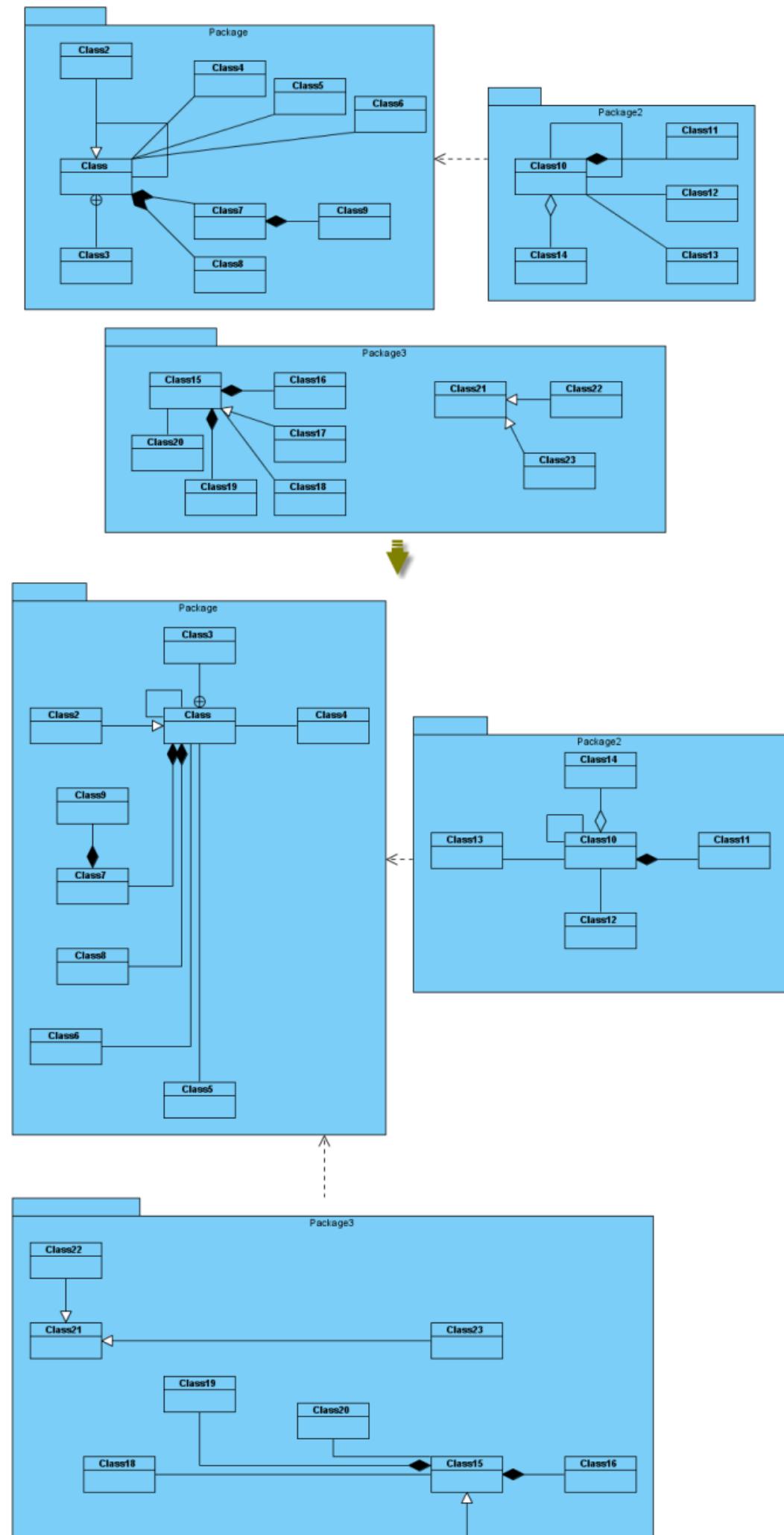
Other diagrams



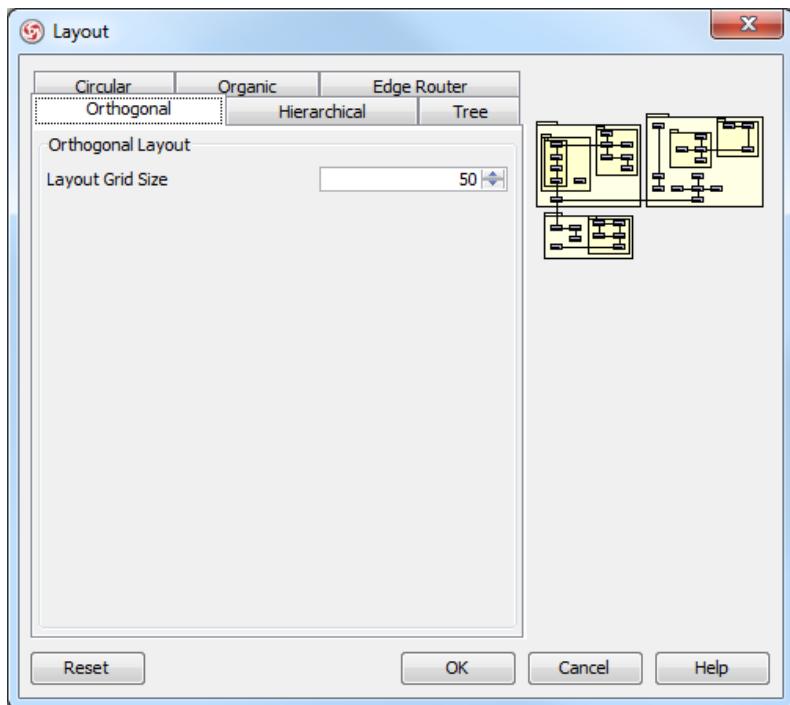
Auto layout of other diagrams

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 VP-UML, 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.



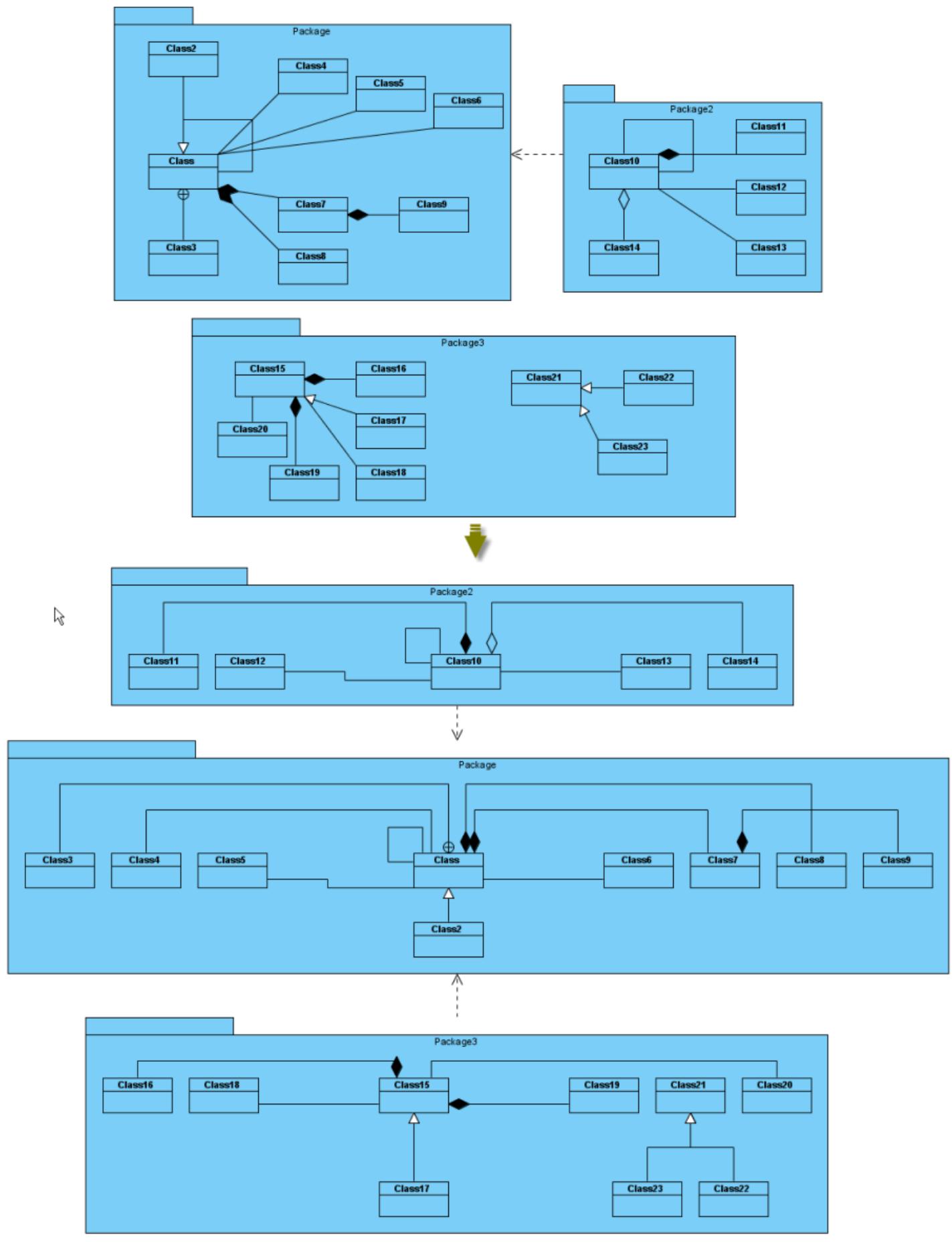
Layout Grid Size: the virtual grid size for layout. Each shape will be placed in accordance with its center point lays on a virtual grid point.



Orthogonal Layout setting

Hierarhic layout

Hierarhic 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.



Hierarhic Layout

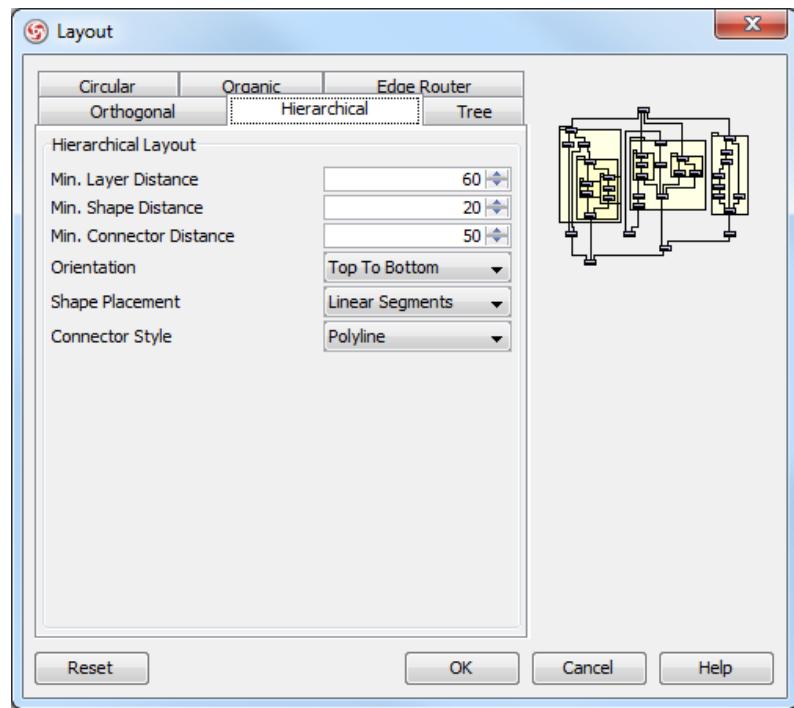
Min. Layer Distance: the minimal horizontal distance between the shapes.

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.

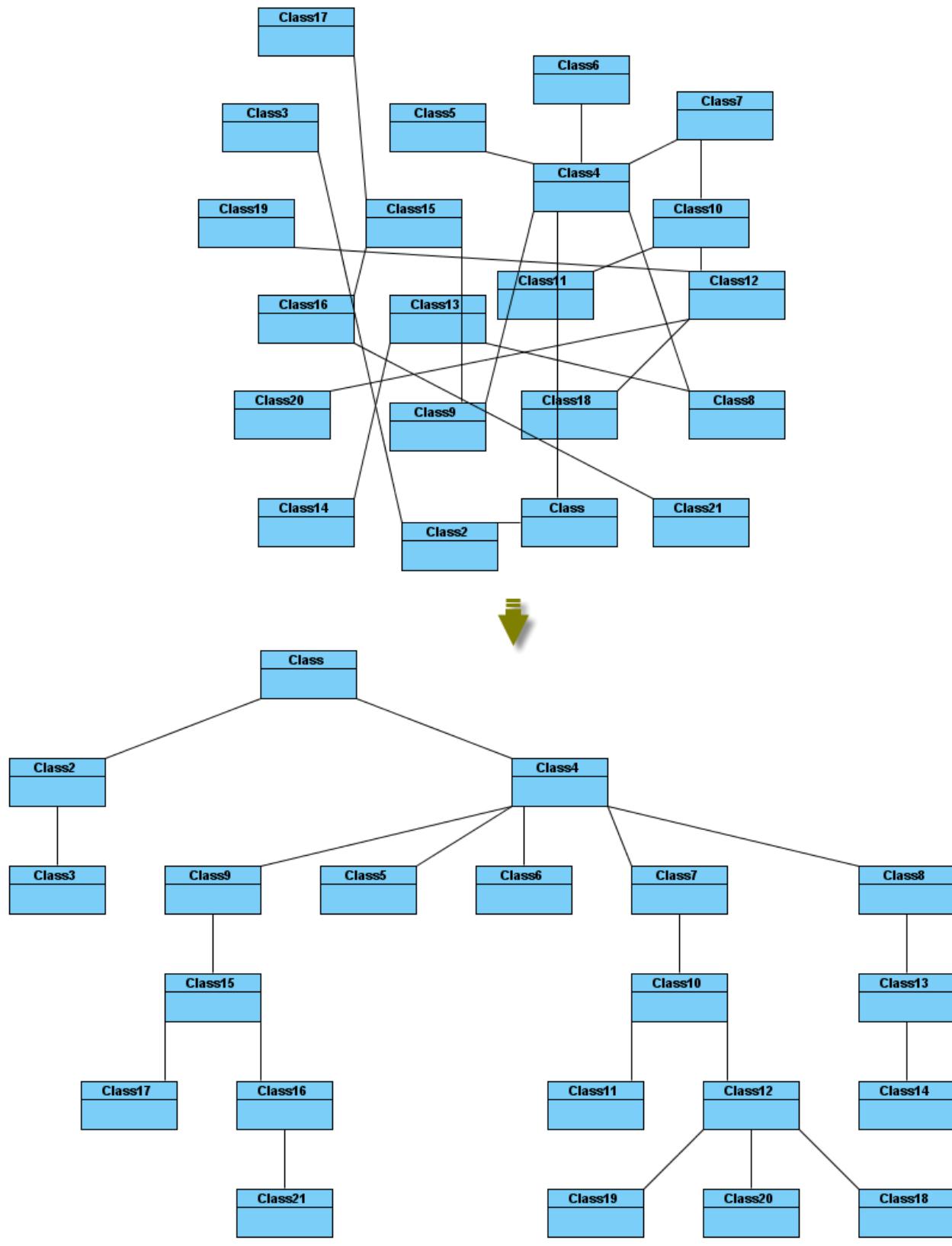
Shape Placement: affects the horizontal spacing between shapes, and the number of bends of the connectors -pendulum, linear segments, polyline, tree, simplex.



Hierarchical Layout setting

Directed tree layout

Directed Tree Layout, which is one of the tree layouts in VP-UML, 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

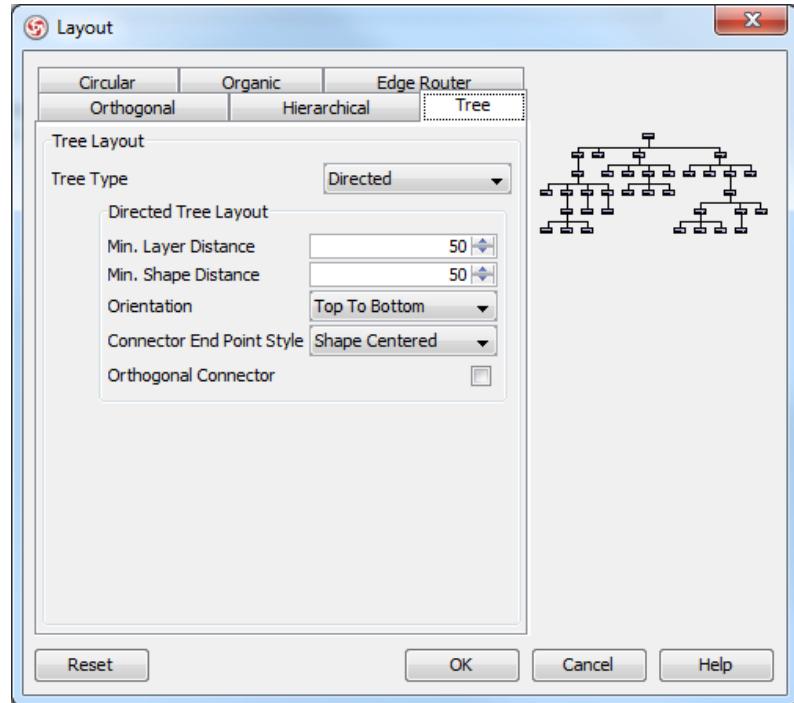
Min. Layer Distance: the minimal horizontal distance between the shapes.

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.

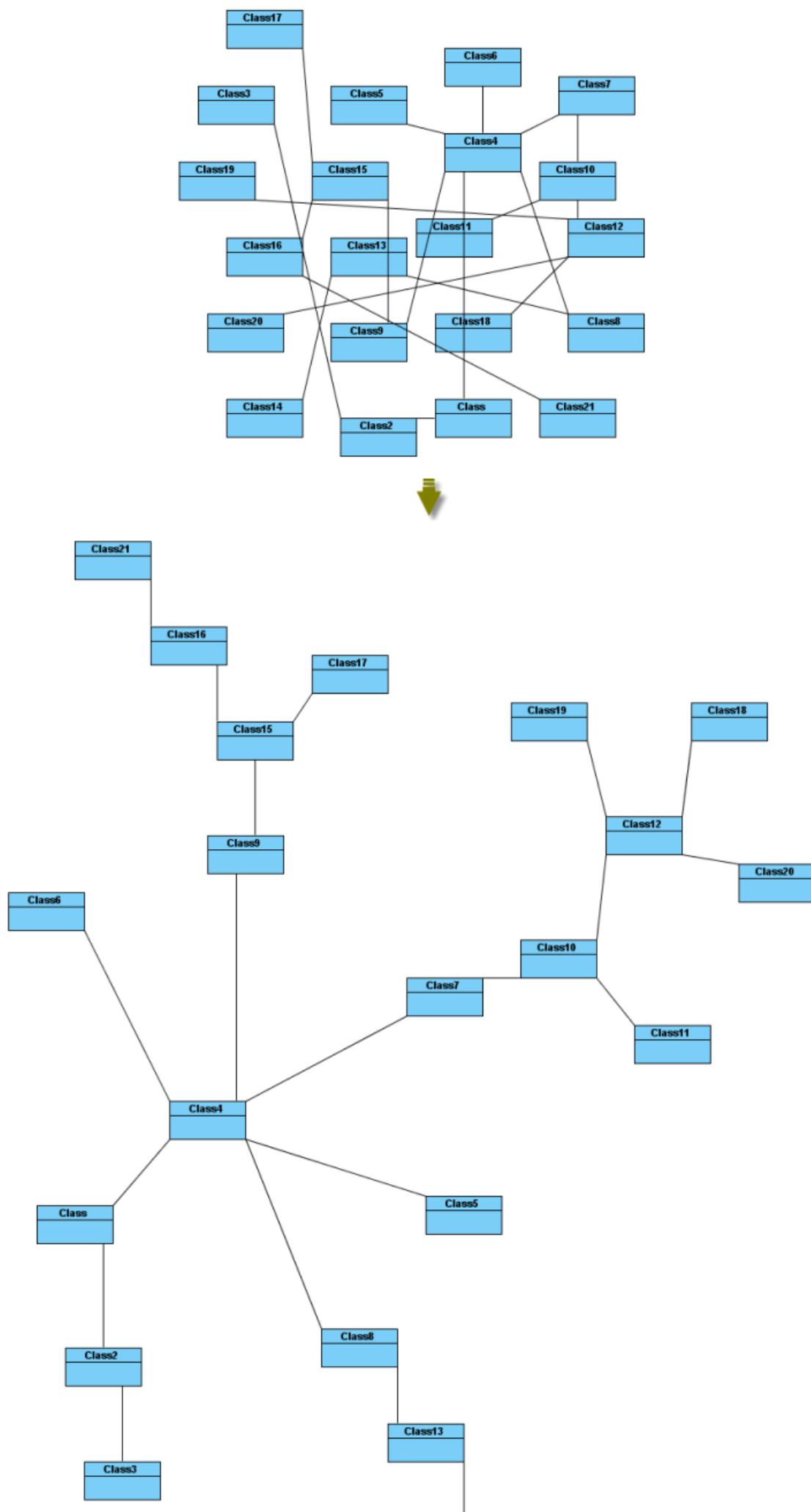
Orthogonal Connector: whether the connectors will be arranged in orthogonal.



Directed Tree Layout setting

Balloon tree layout

Balloon Tree Layout, which is one of the tree layouts in VP-UML, arranges shapes in a tree structure in a radial fashion. It is the best way for users to arrange large trees.

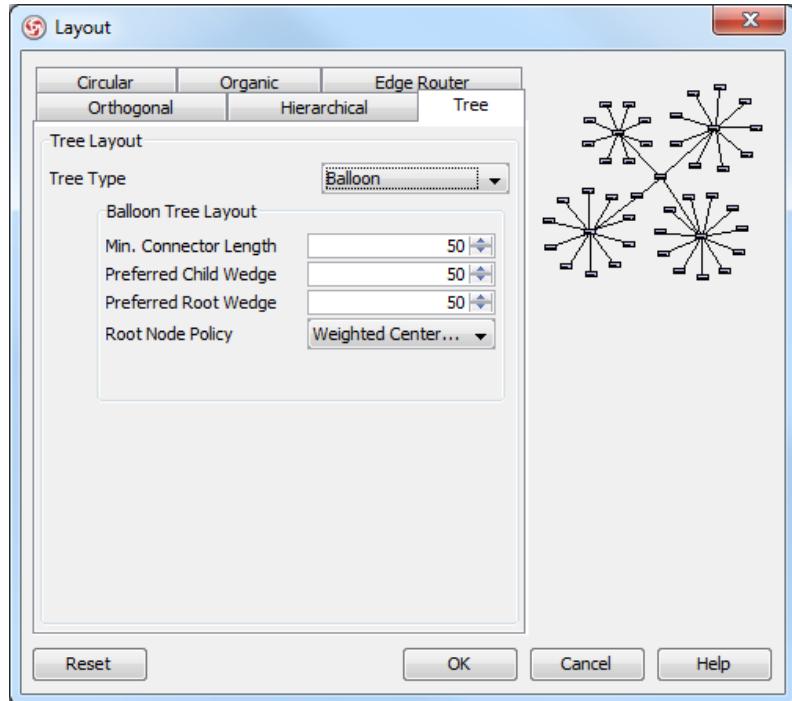


Min. Connector Length: the minimal distance between the connectors and shapes.

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.

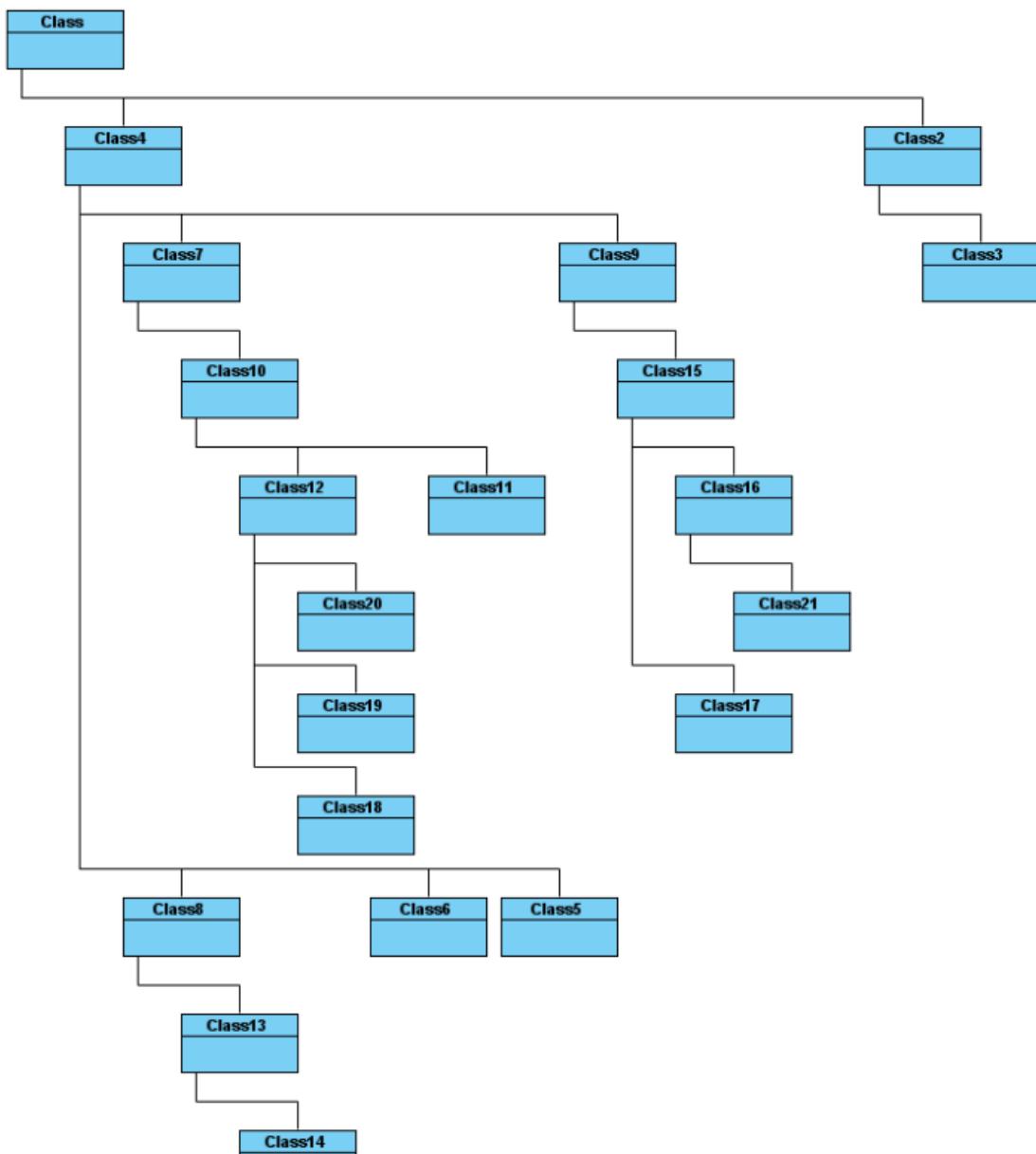
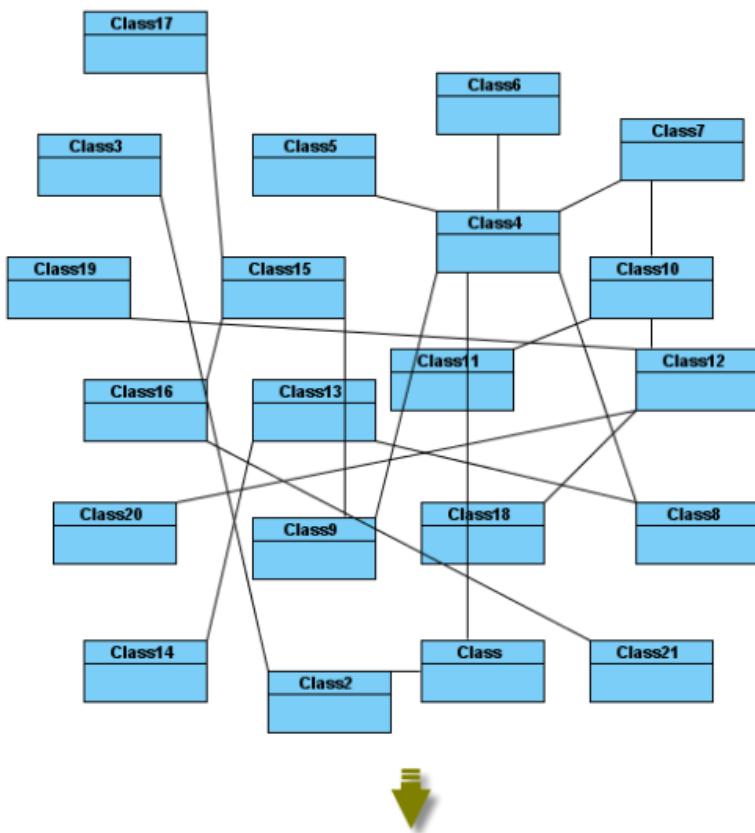
Root Node Policy: determines which node is chosen as the tree root node for layout – directed root, center root, and weighted center root.



Balloon Tree Layout setting

Compact tree layout

Compact Tree Layout, which is one of the tree layouts in VP-UML, arranges shapes in a tree structure. The aspect ratio (relation of tree width to tree height) of the resultant tree can be set.

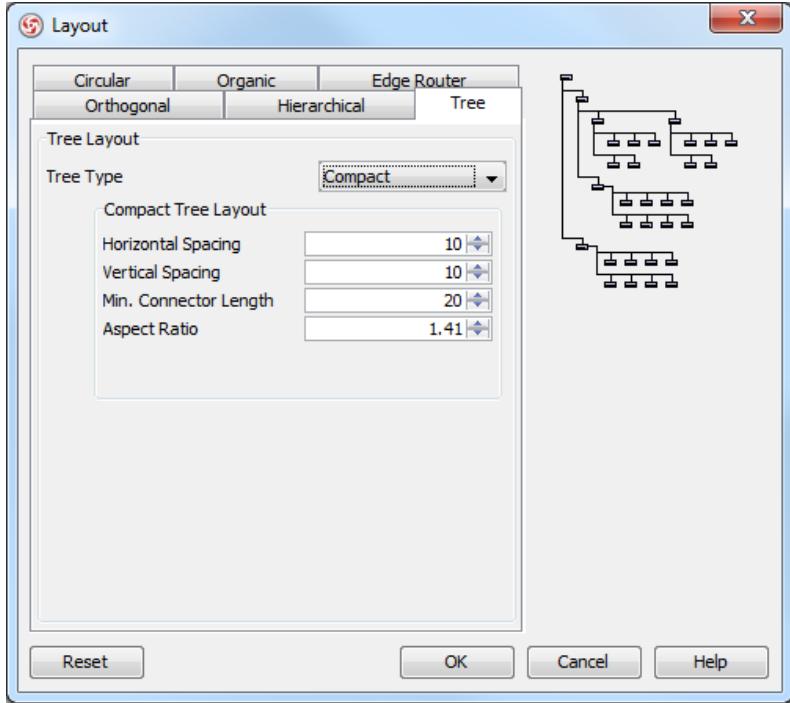


Horizontal Spacing: the horizontal spacing between the shapes.

Vertical Spacing: the vertical spacing between the shapes.

Min. Connector Length: the vertical distance of the connector segments.

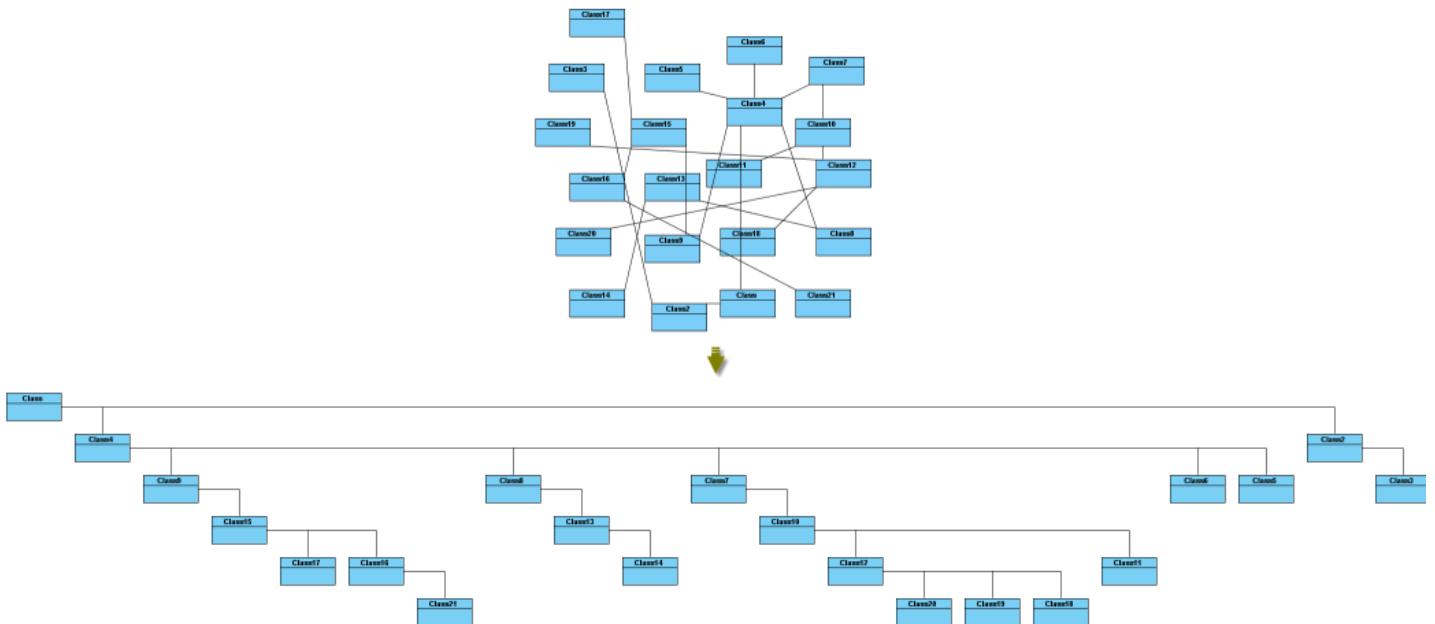
Aspect Ratio: the relation of the tree width to the tree height.



Compact Tree Layout setting

Horizontal-Vertical tree layout

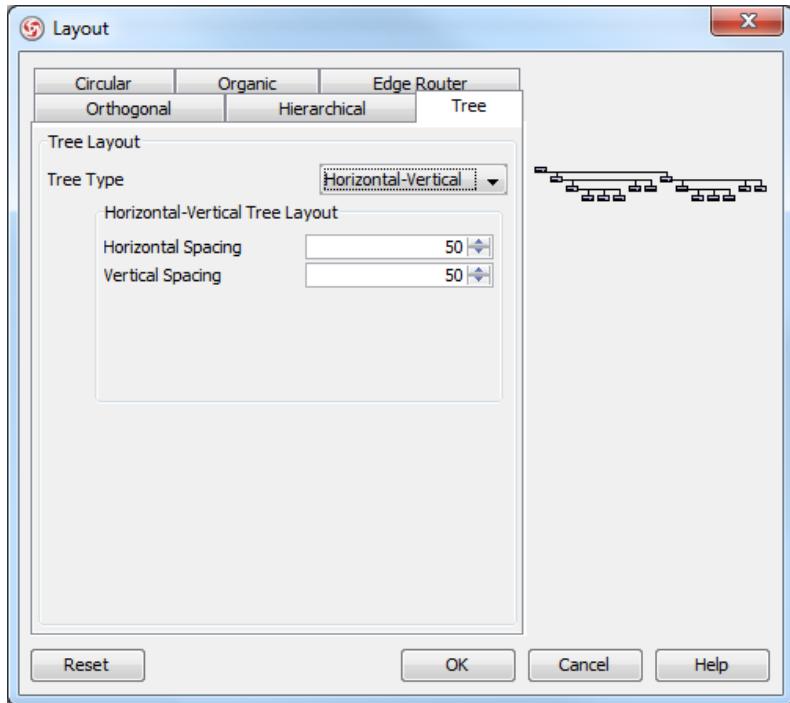
Horizontal-Vertical Tree Layout, which is one of the tree layouts in VP-UML, arranges shapes in a tree structure horizontally and vertically.



Horizontal-Vertical Tree Layout

Horizontal Spacing: the horizontal spacing between the shapes.

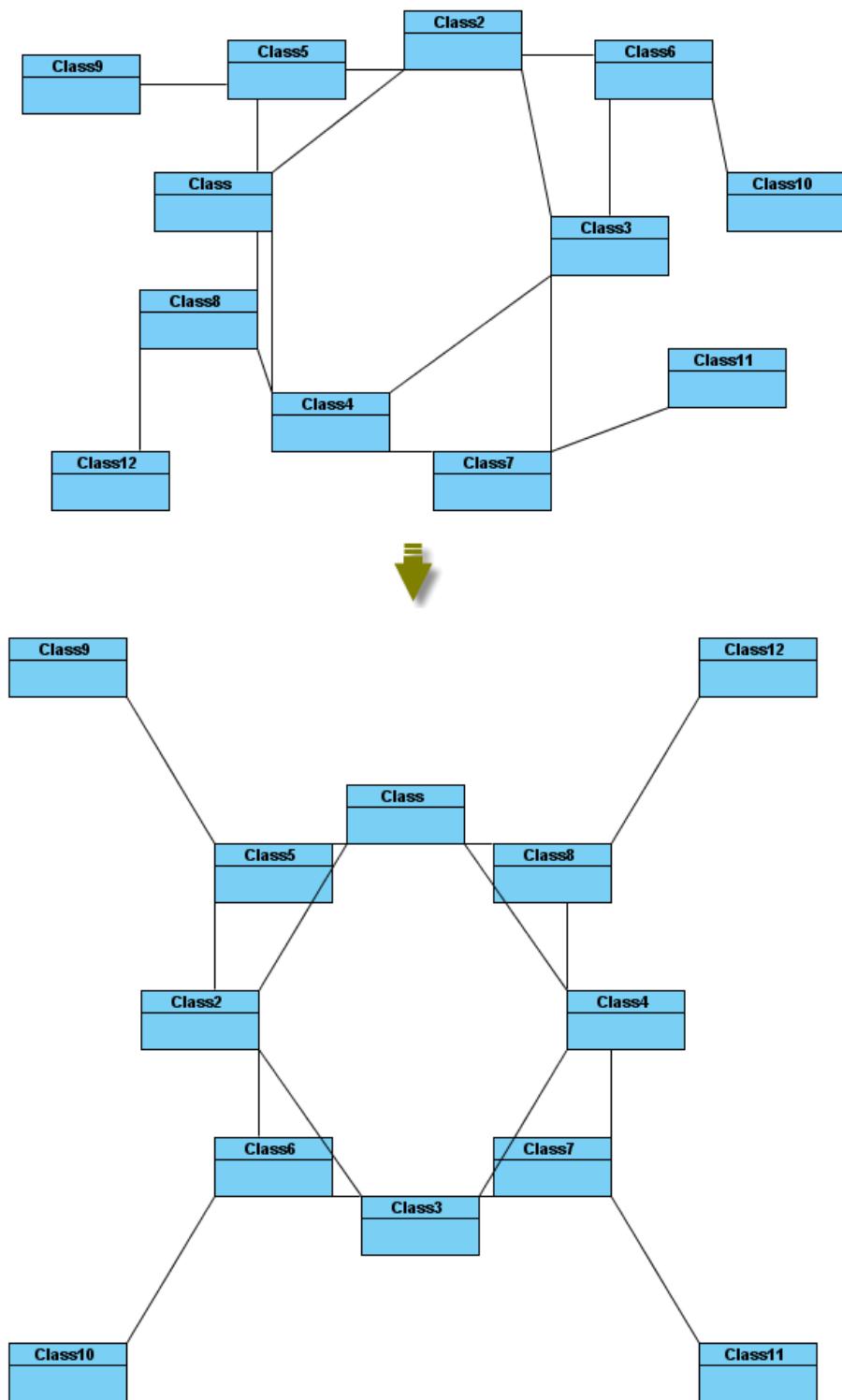
Vertical Spacing: the vertical spacing between the shapes.



Horizontal-Vertical Tree Layout setting

BBC compact circular layout

BBC Compact Circular Layout, which is one of the circular layouts in VP-UML, 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

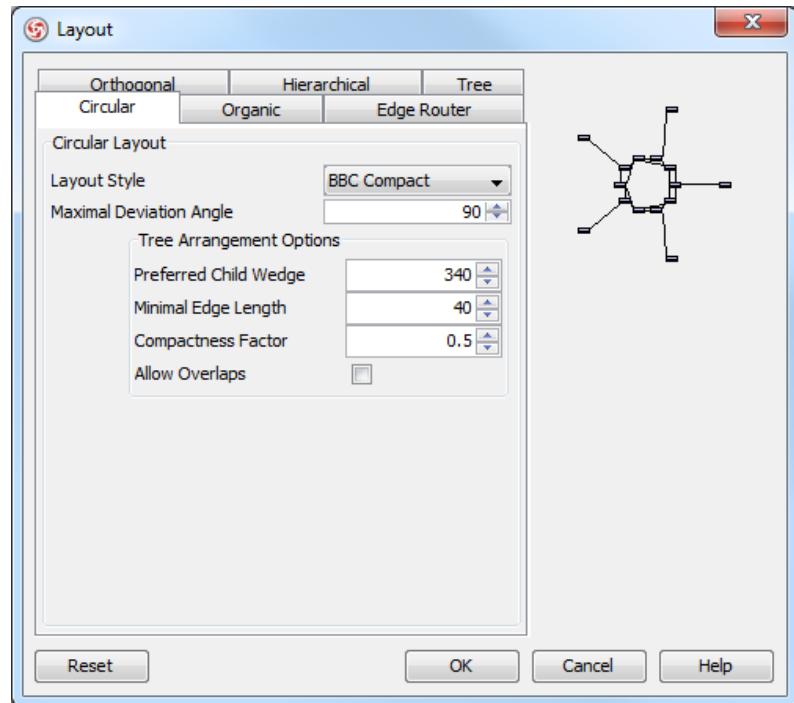
Maximal Deviation Angle: the maximal angle of deviation.

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.

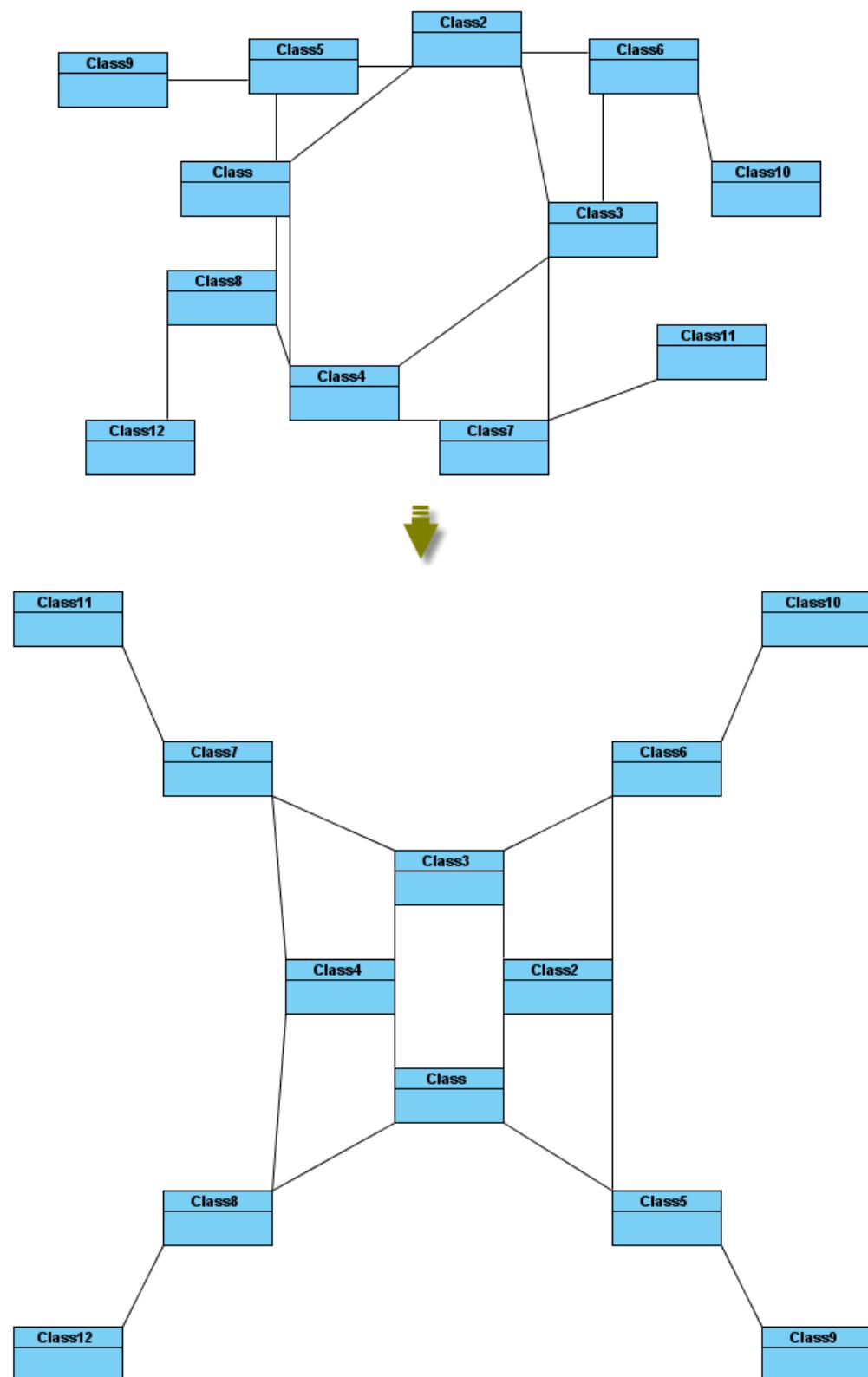
Allow Overlaps: whether the shape can be overlapped.



BBC Compact Circular Layout setting

BBC isolated circular layout

BBC Isolated Circular Layout, which is one of the circular layouts in VP-UML, 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

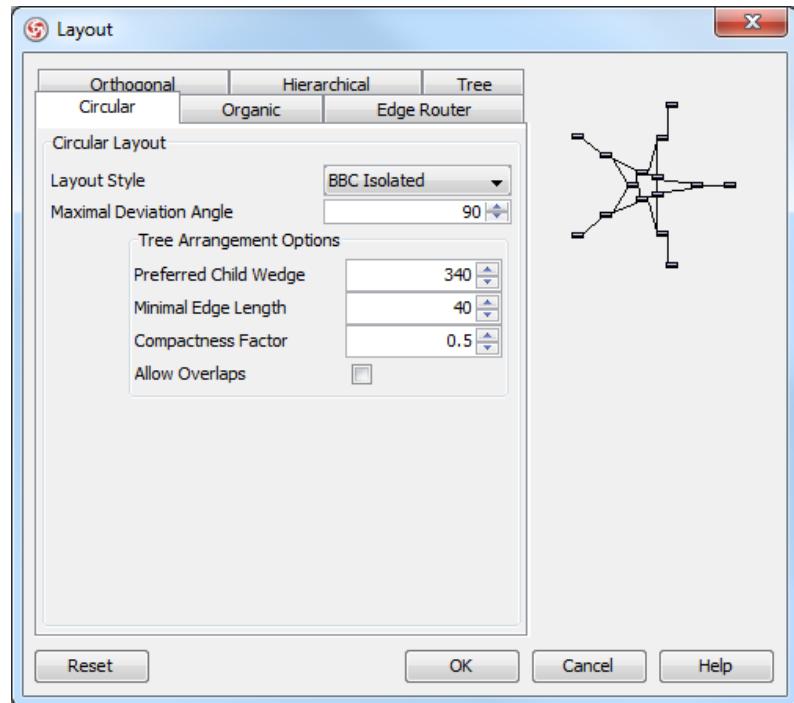
Maximal Deviation Angle: the maximal angle of deviation.

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.

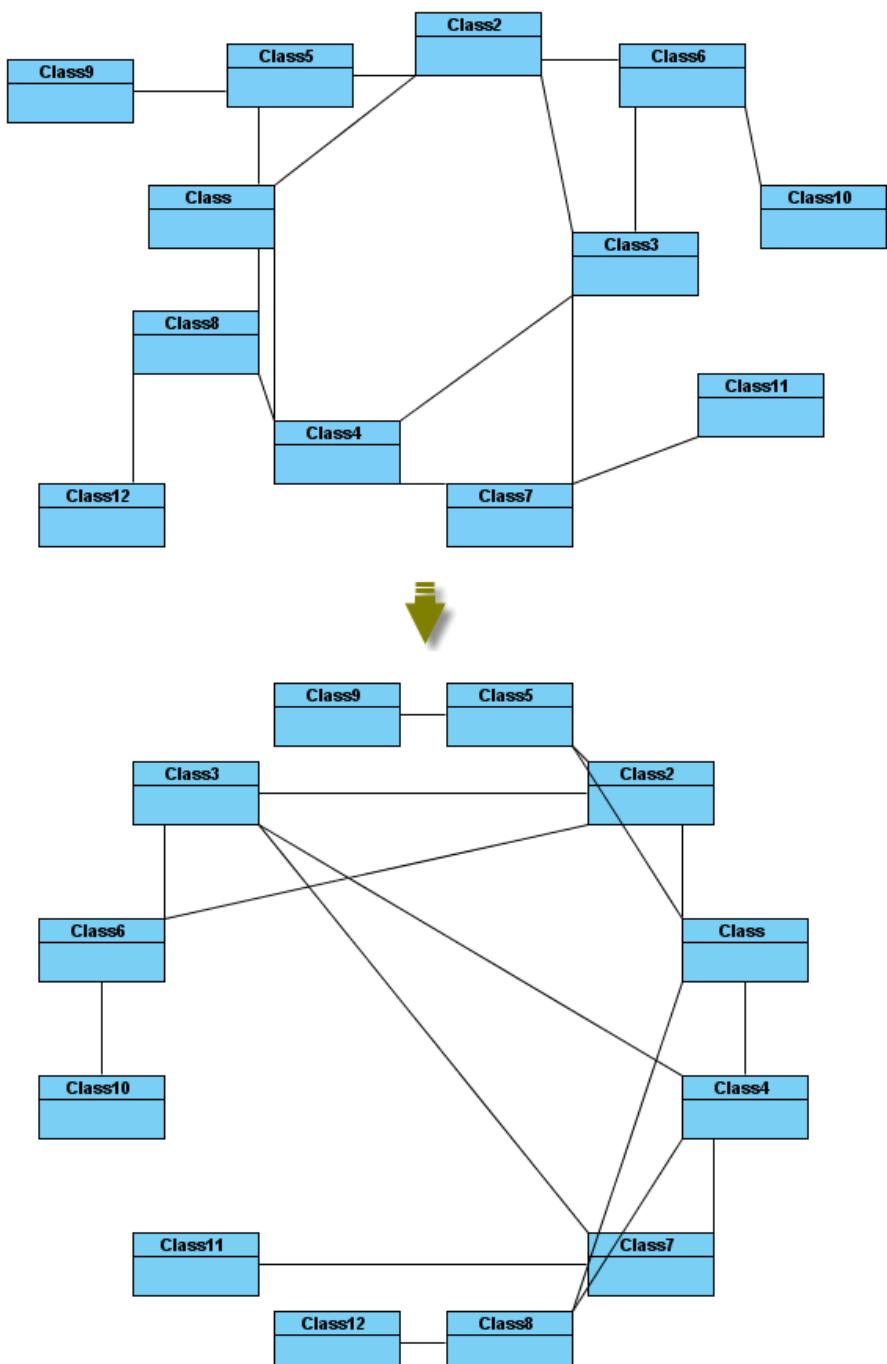
Allow Overlaps: whether the shape can be overlapped.



BBC Isolated Circular Layout setting

Single cycle circular layout

Single Cycle Layout, which is one of the circular layouts in VP-UML, arranges shapes in circular structure in single circle.

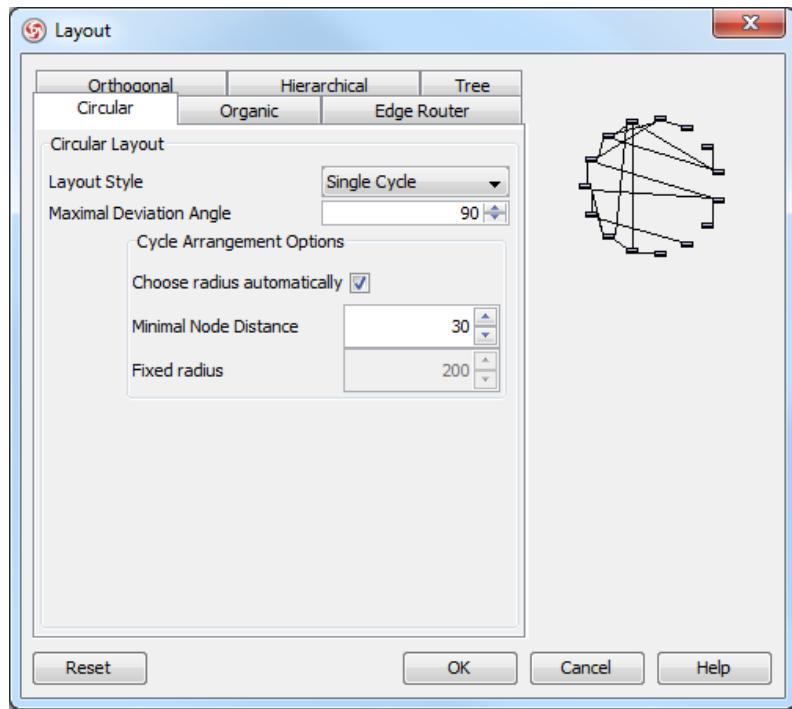


Single Cycle Circular Layout

Choose radius automatically: determine the radius of circular structure automatically or manually.

Minimal Node Distance: the minimal distance between the nodes.

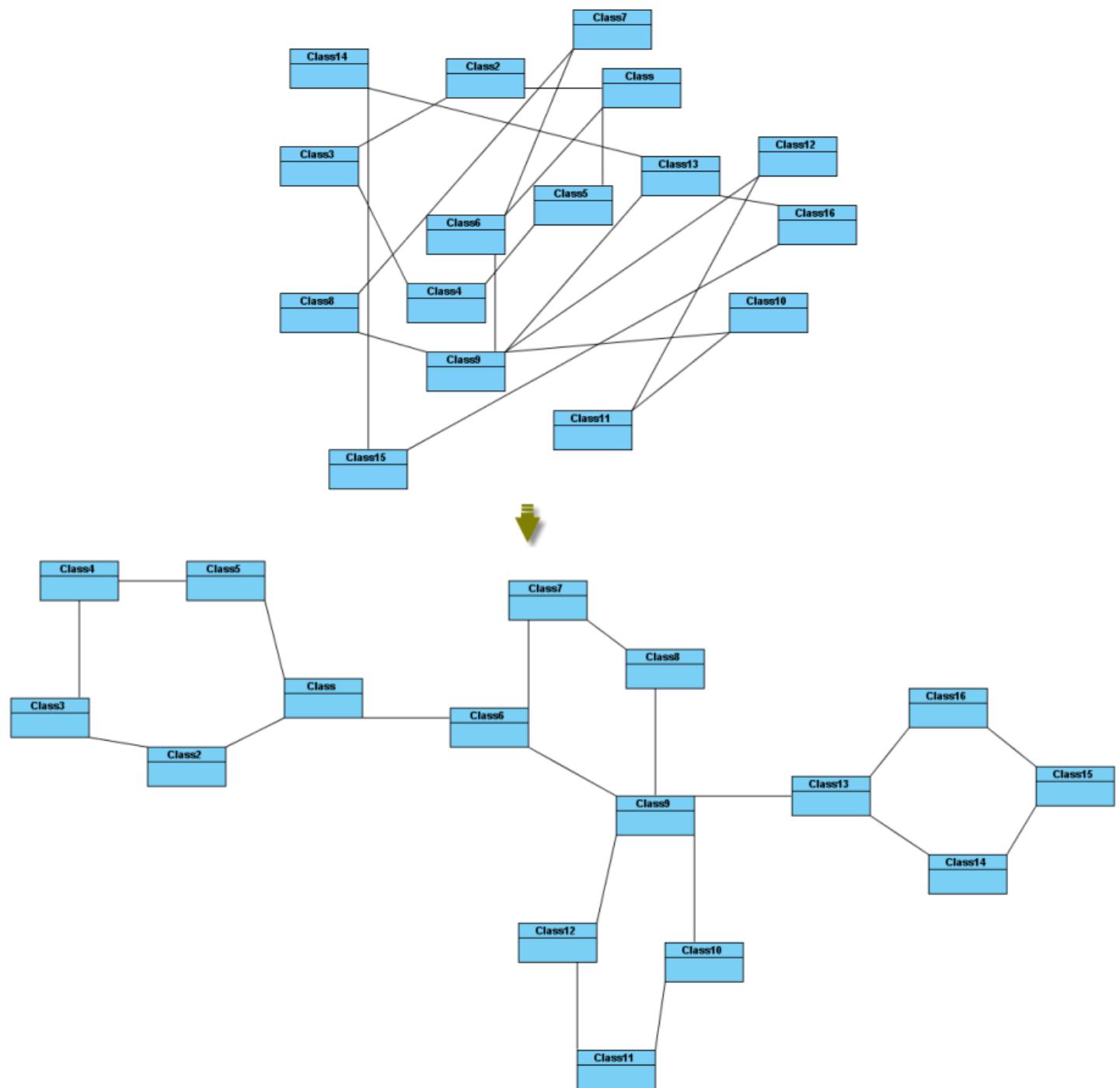
Fixed radius: the radius of circular structure.



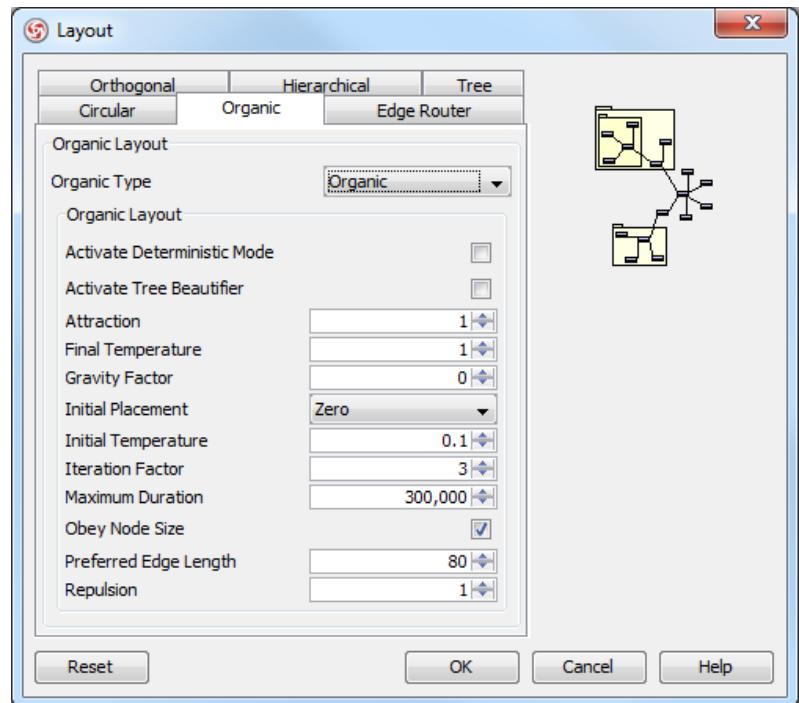
Single Cycle Circular Layout setting

Organic layout

Organic Layout, which is one of the organic layouts in VP-UML, arranges shapes in a star or ring structure. It is the best way for users to arrange the shapes that have highly connectivity relationship.



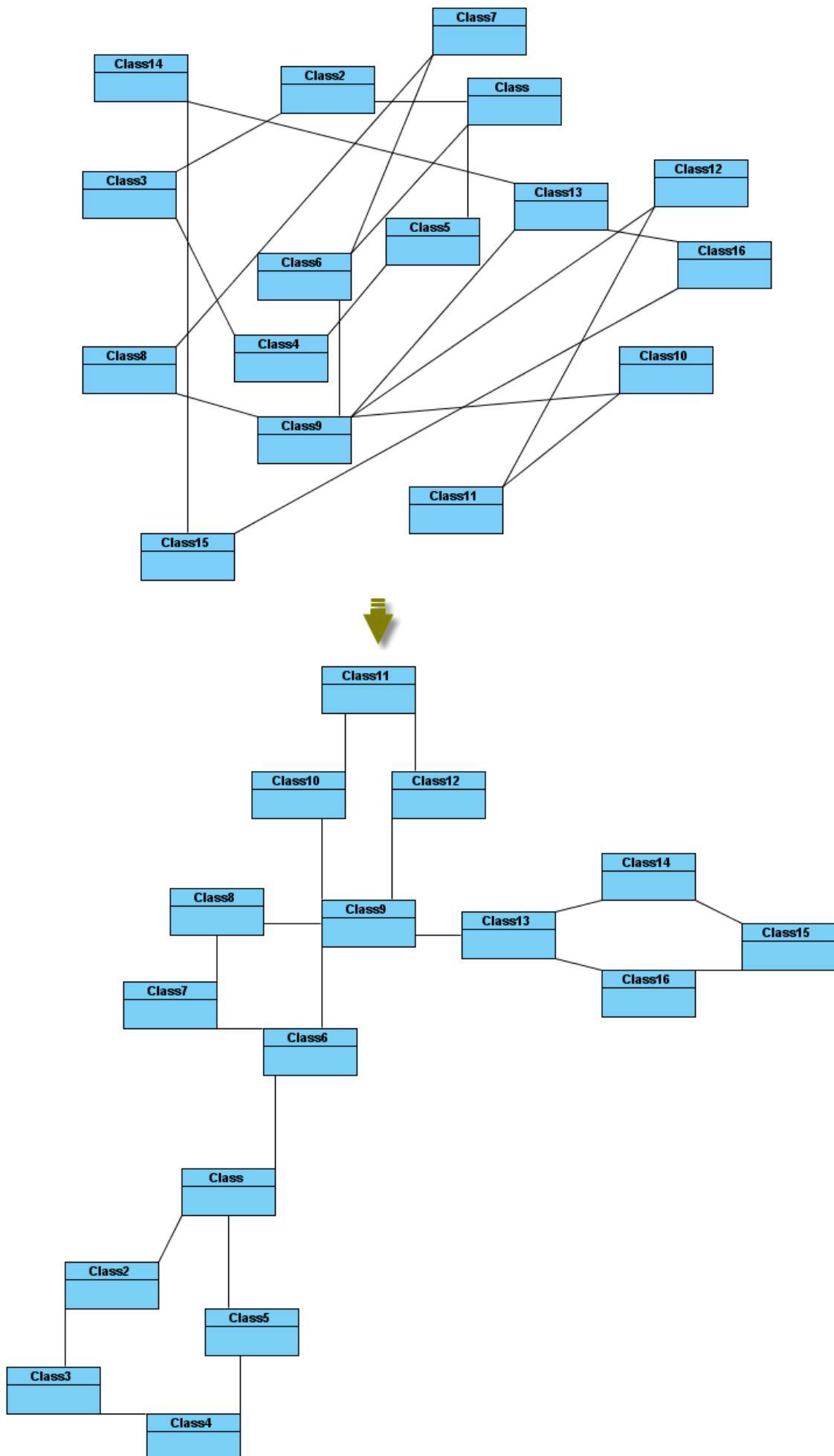
Activate Deterministic Mode: whether the layouter is in deterministic mode.
Activate Tree Beautifier: whether or not to activate the subtree beautifier.
Attraction: the degree of the attraction between shapes.
Final Temperature: the factor that affects the distance between shapes.
Gravity Factor: the 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, which is one of the organic layouts in VP-UML, 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.



Compactness: the factor that sets less/more compact layout.

Deterministic: whether the layouter is in deterministic mode.

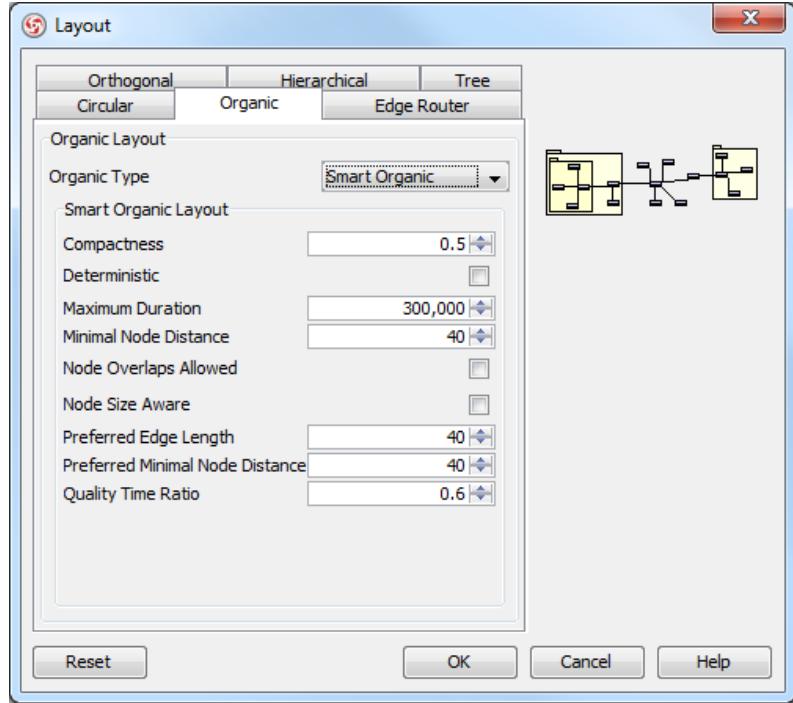
Minimal Node Distance: the minimal distance between nodes.

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.

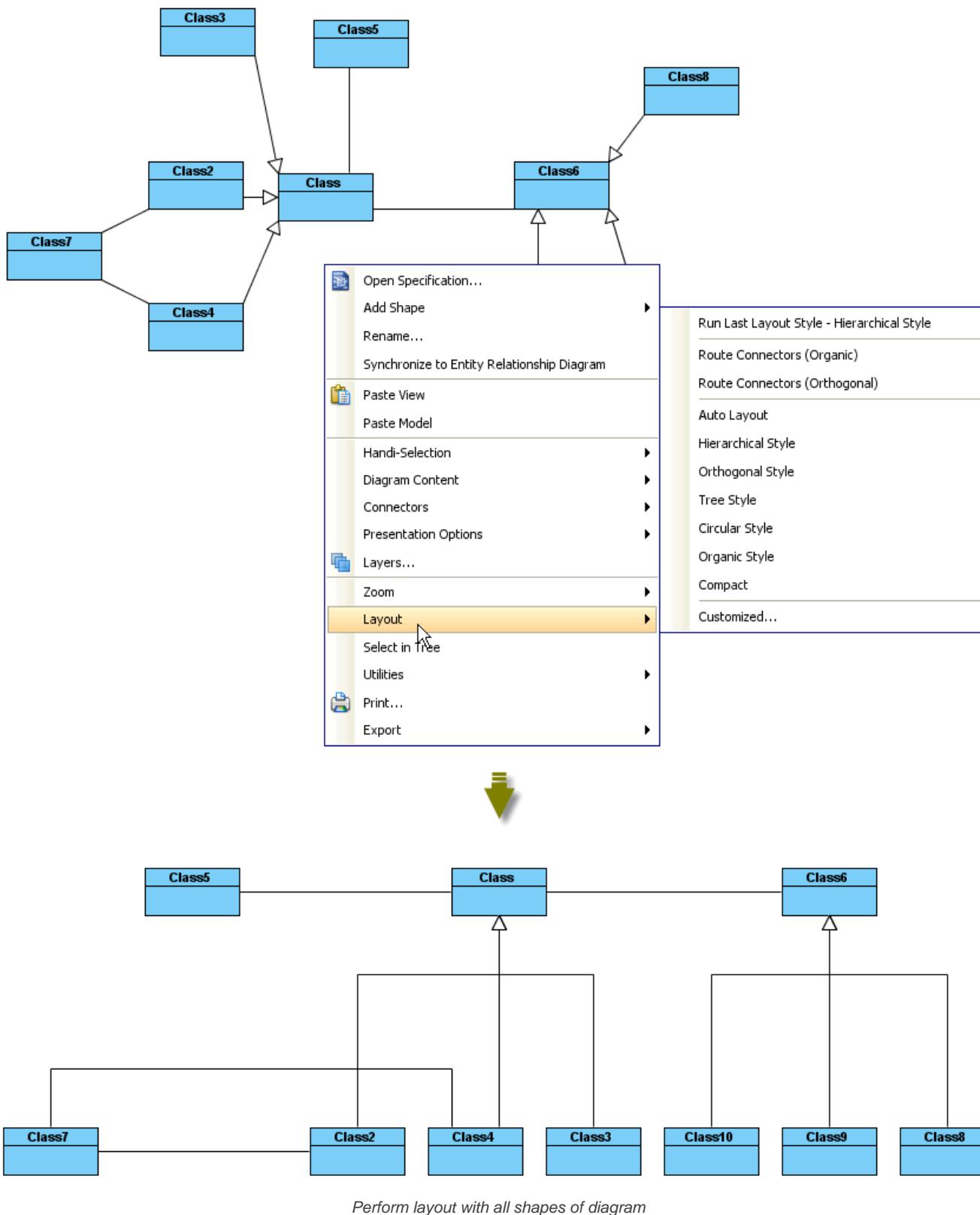
Quality Time Ratio: the ratio of the quality of layout to the producing time of layout.



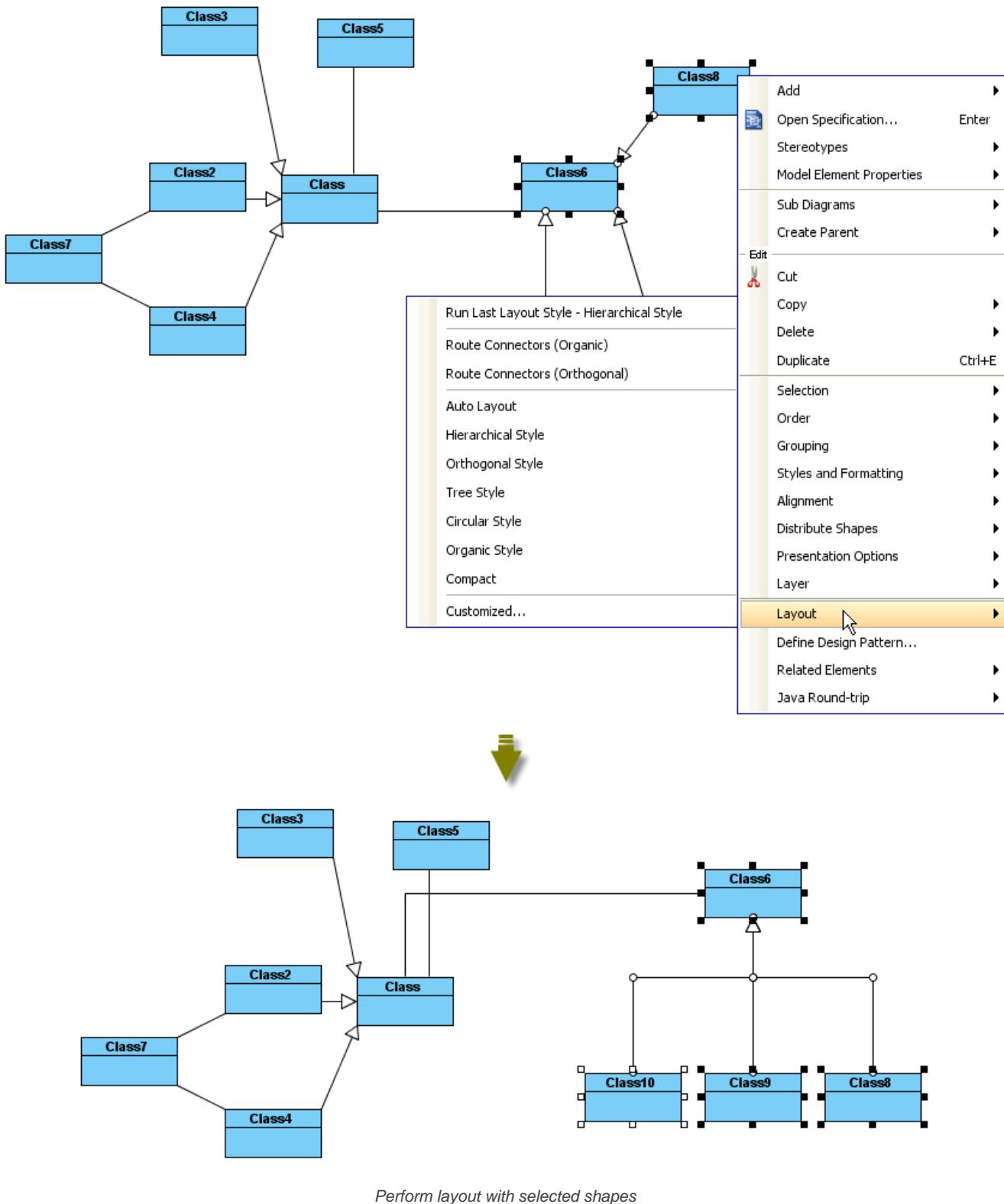
Organic Layout setting

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.



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).

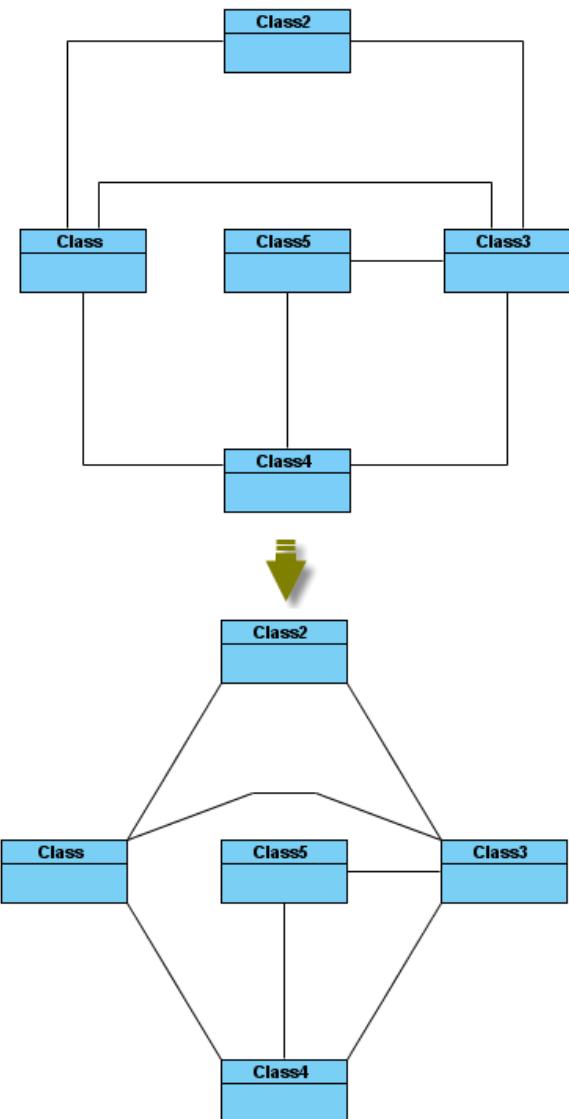


Automatic route connectors

There are 2 kinds of layouts which do not change the location of shapes, only change the connectors: **Organic Edge Route Layout** and **Orthogonal Edge Route Layout**

Organic edge route layout

Organic Edge Route Layout, which is one of the edge route layouts in VP-UML, arranges the connectors without affecting the location of shapes. It can ensure that the shapes will not overlap and keep a specific minimal distance.

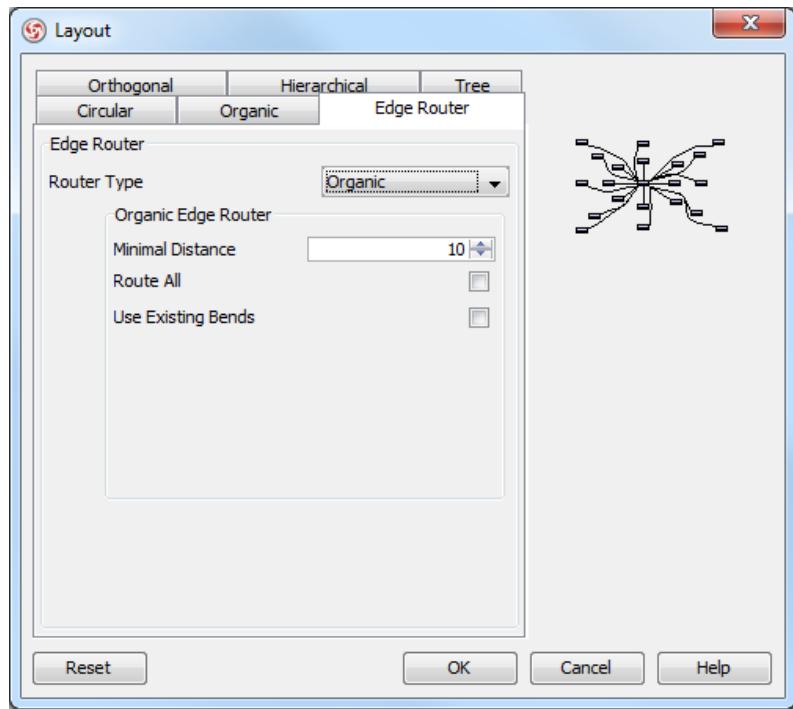


Organic Edge Route Layout

Minimal Distance: the minimal distance of the connectors.

Route All: whether all the connectors will be routed.

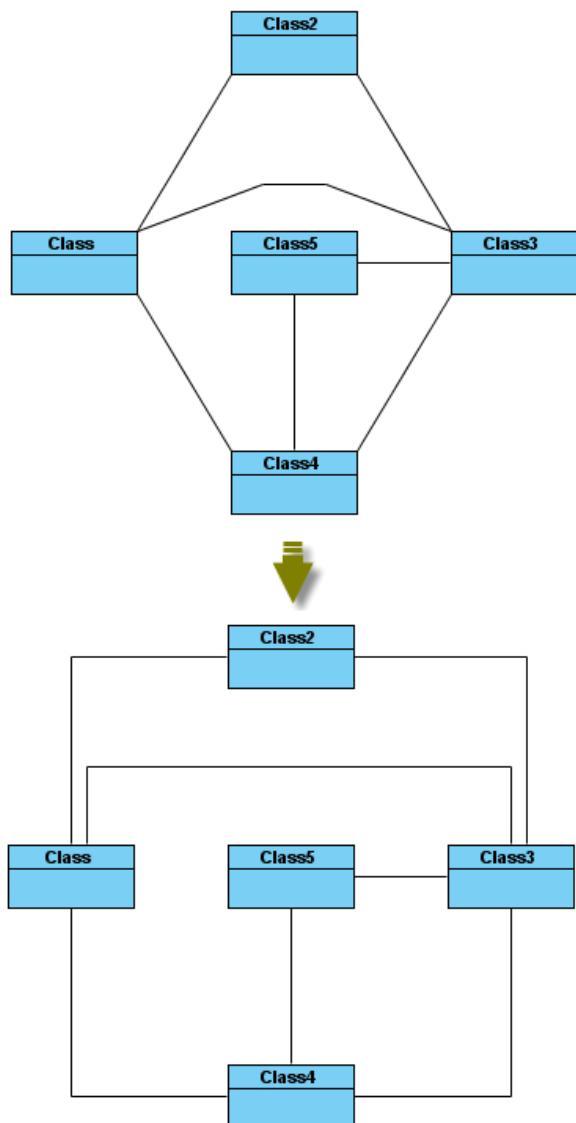
Use Existing Bends: whether using existing bends.



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.

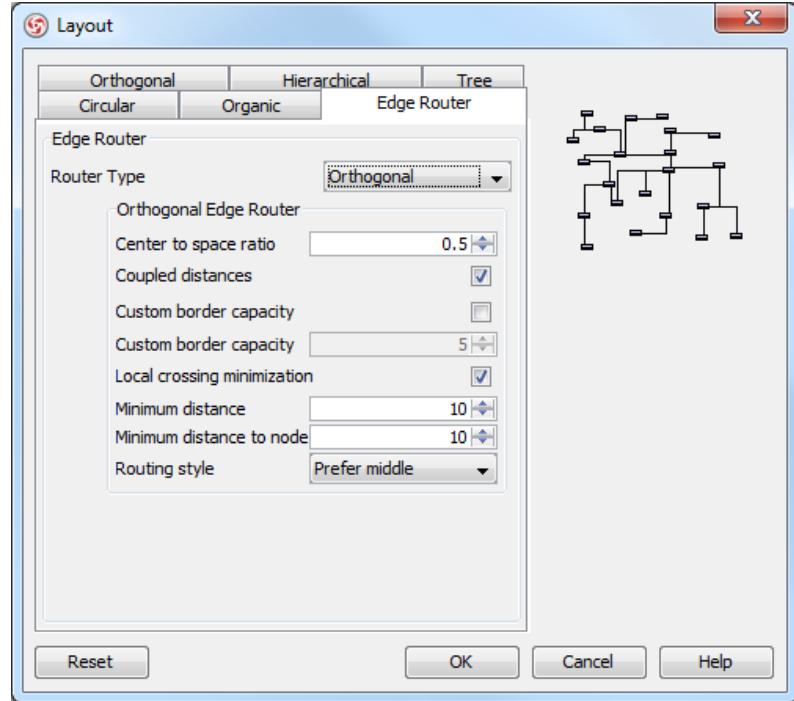
Local crossing minimization: whether the local crossing of connectors will be minimized.

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.



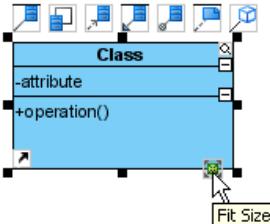
Fit shape size

In some cases, shapes are found oversized. For better presentation, you may need to resize them smaller. Fit size can help you 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**.

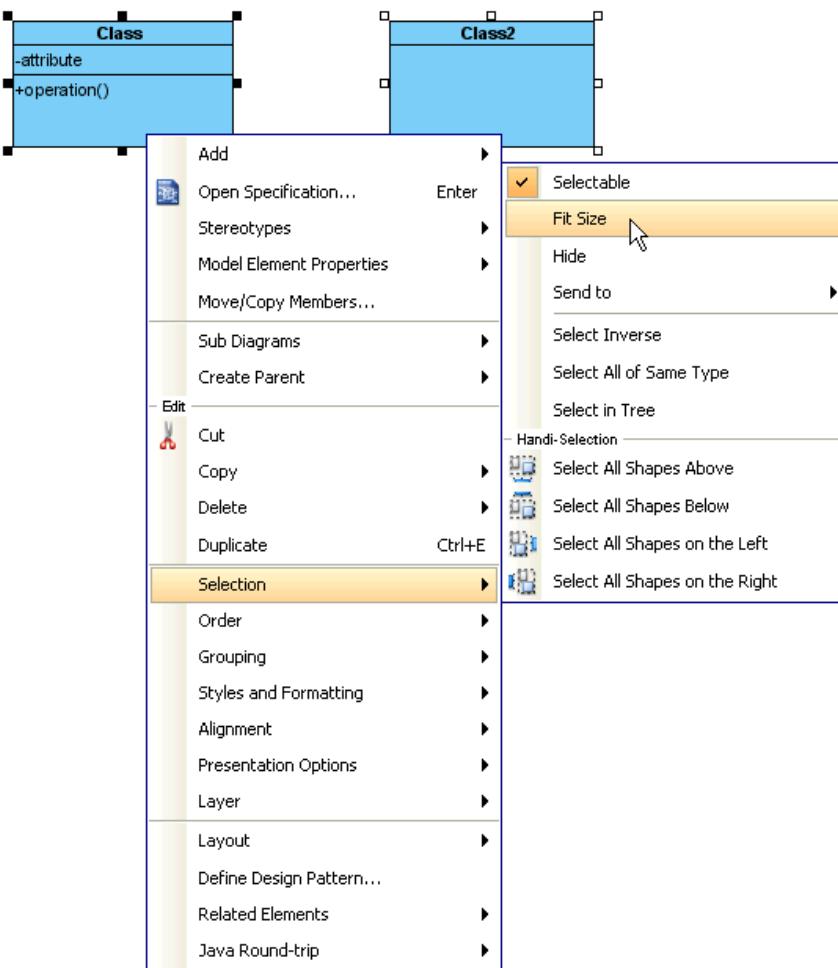
Fit selected shapes size

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, move the mouse on those shapes, right click on one of the selected shapes and then select **Selection > Fit Size** from the pop-up menu.



Fit size for several shapes from the pop-up menu

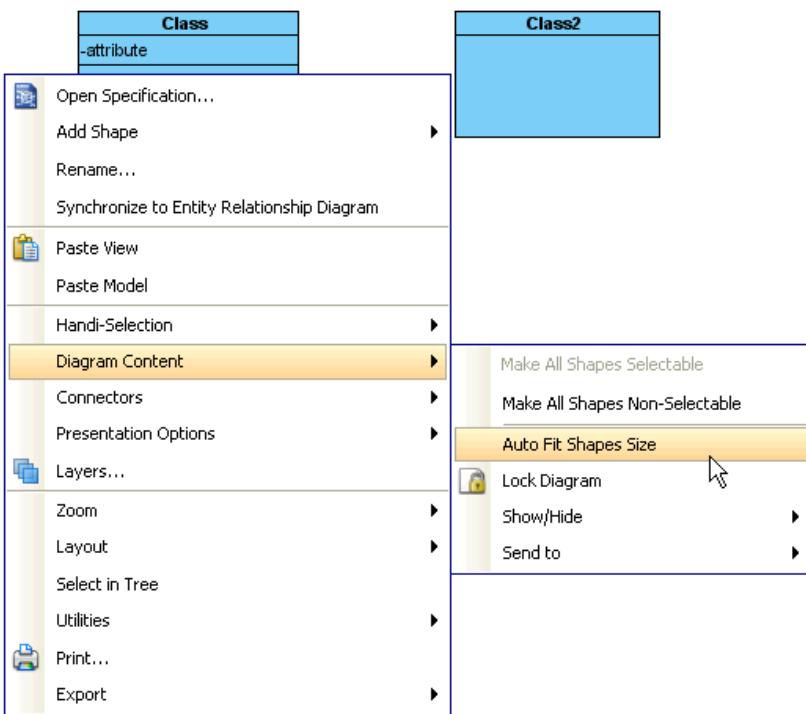
Each shape will be adjusted to its fit size in accordance with its content, instead of fixing all selected shapes into the exactly same size.



Shapes are fitted size

Check/uncheck automatic fit shape size mode

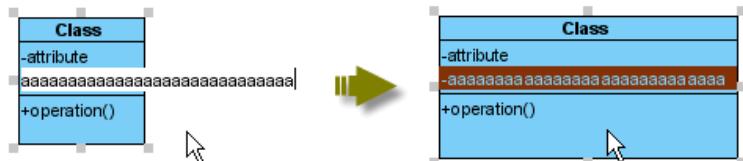
You can check/ uncheck the **Auto Fit Shapes Size** on diagram to make all the shapes on the diagram to be fitted size automatically. To do so, right click on the diagram's background, select **Diagram Content > Auto Fit Shapes Size** from the pop-up menu.



Check 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.

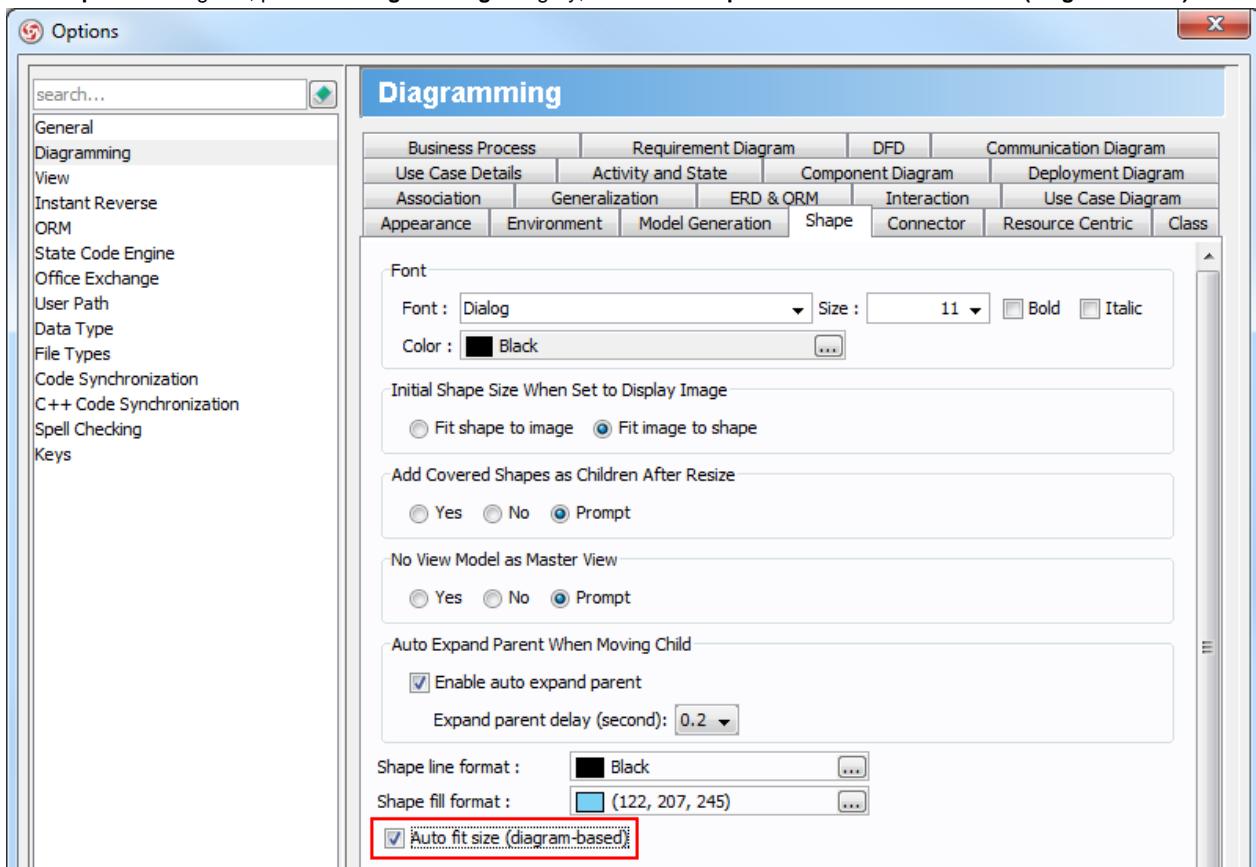


Class shape is resized automatically after new attribute added

You can also check **Auto Fit Size** for future usage.

1. Select **Tools > Options...** from the main menu to unfolded **Options** dialog box.

2. In the **Options** dialog box, press the **Diagramming** category, select the **Shape** tab and check **Auto fit size (diagram-based)**.



Check **Auto fit size (diagram-based)** in the **Options** dialog box

Diagram element selection

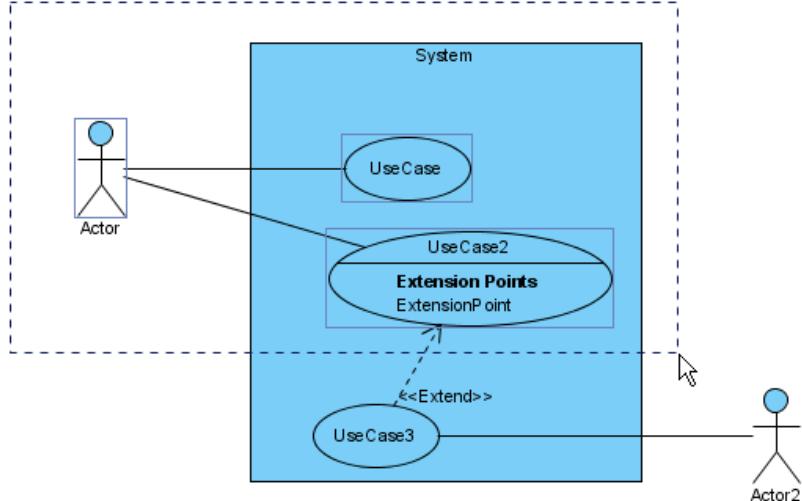
When a number of diagram elements need selecting simultaneously, diagram element selection supports this purpose. You can either click directly with hot keys or select a range of selection with the mouse. A specific type of diagram element(s) on a diagram can be selected as well.

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.

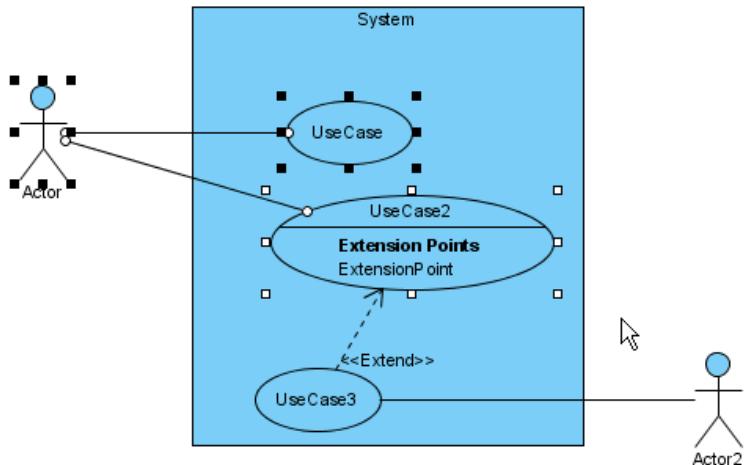
Selecting a range of shapes with the mouse

1. For selecting multiple shapes, drag them from corner to corner diagonally with the mouse. They will then be surrounded by a rectangle individually.



Select multiple shapes with the mouse

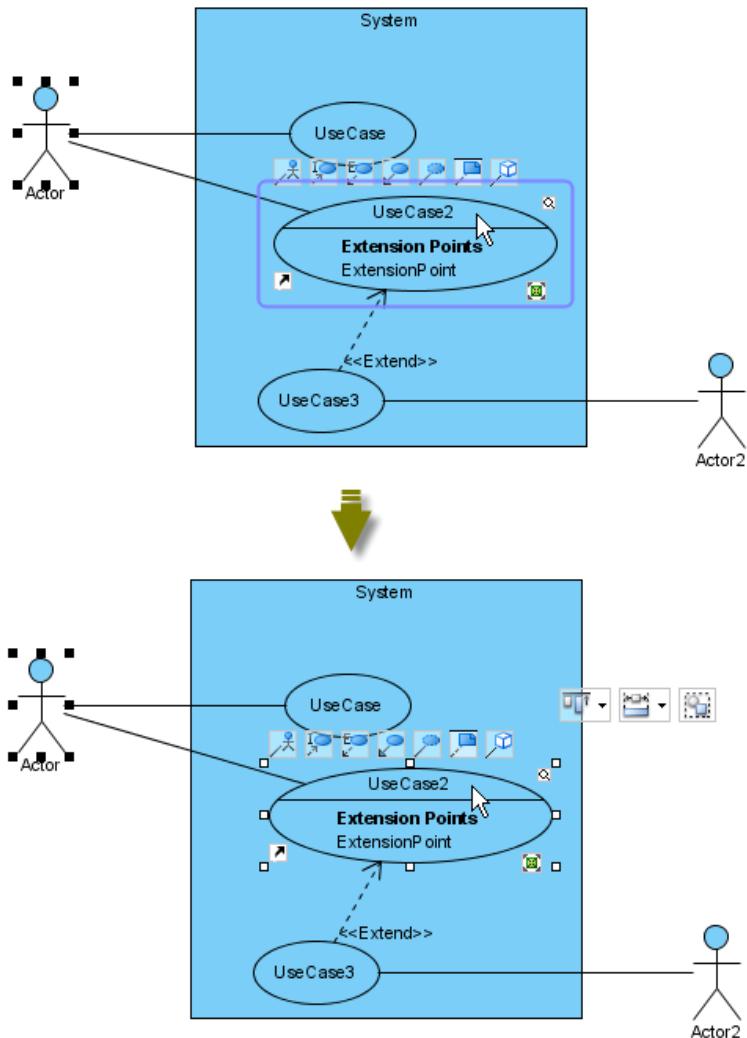
2. After releasing the mouse, those shapes will be selected.



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.

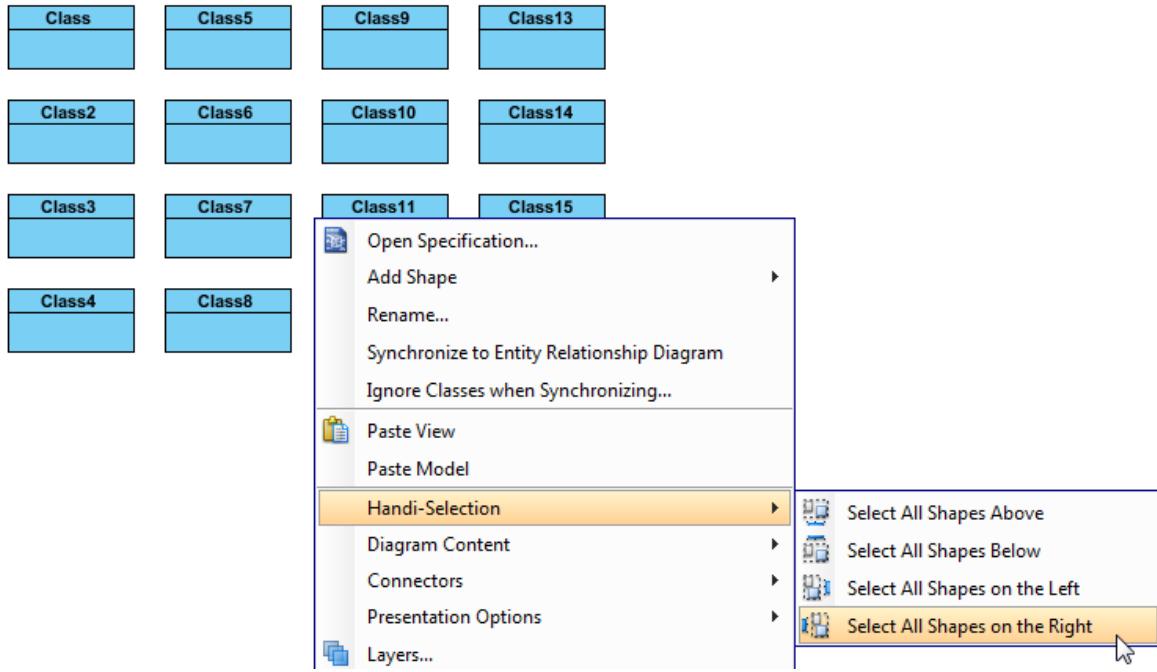


*Click shapes with pressing **Ctrl** or **Shift** key*

Handi-Selection

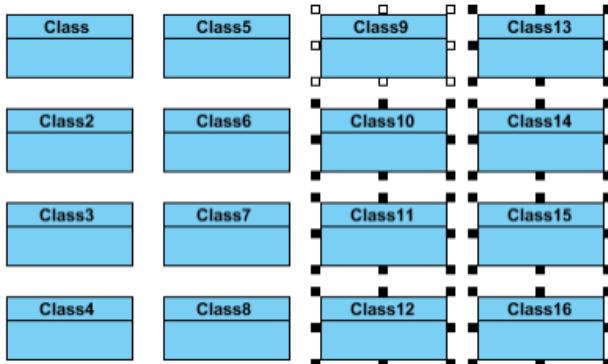
In some cases, the shapes are too complicated and more seriously, the whole diagram is extremely enormous that neither selecting a range of shapes with the mouse nor clicking shapes with pressing **Ctrl** or **Shift** key are the most suitable application. It is hard to drag the mouse on the large diagram, 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

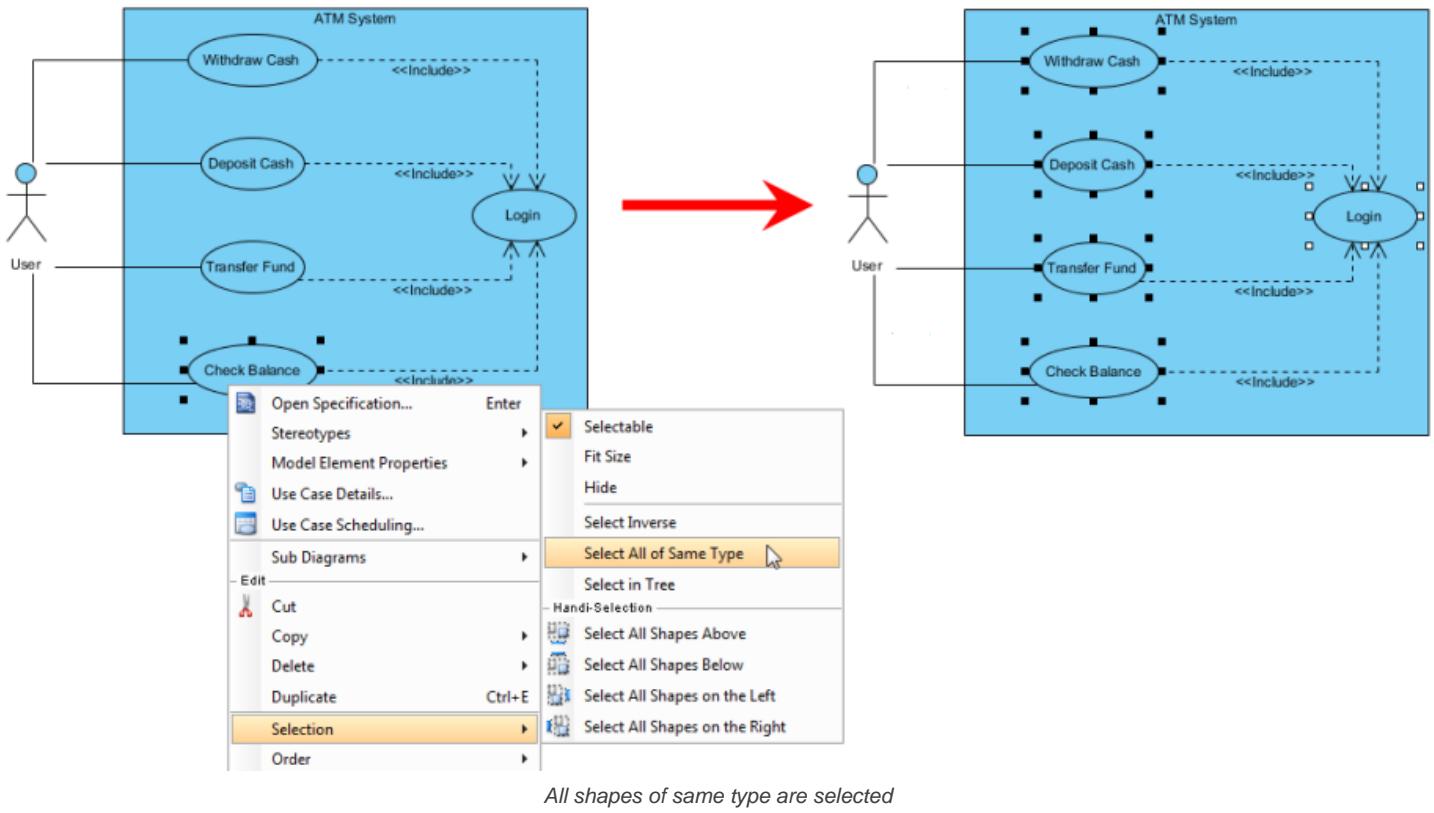
2. As a result, all the shapes of the particular scope will be selected.



All shapes on the right are selected

Selecting same type of shapes

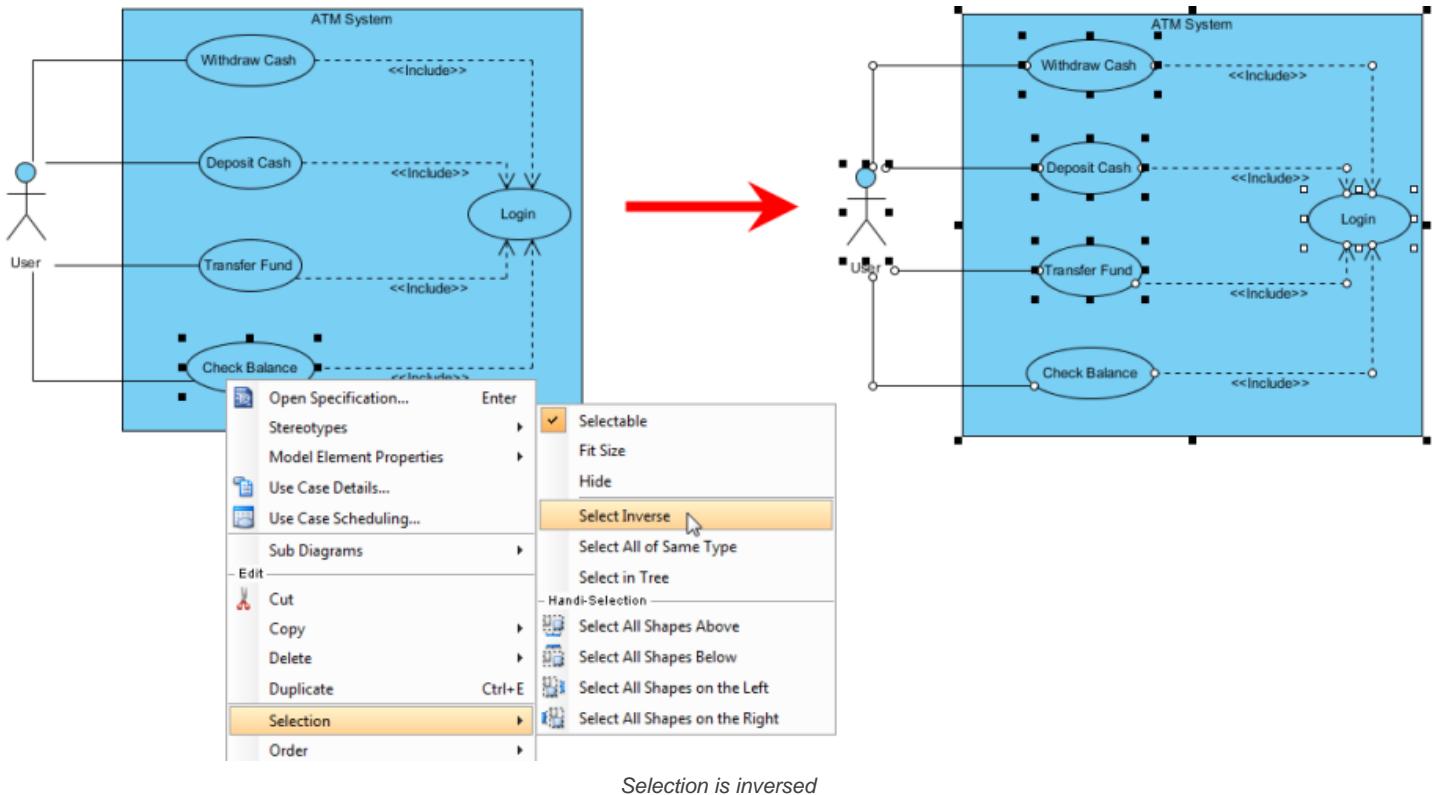
When you want to select a few shapes of same types on the diagram, right click on a shape and select **Selection > Select All of Same Type** from the pop-up menu. As a result, other shapes of same type as the shape you selected previously will be selected.



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

Copy and paste

Cut and paste

You can cut and paste the shapes between diagrams.

1. Right-click on selected shapes, select **Cut** from popup menu.

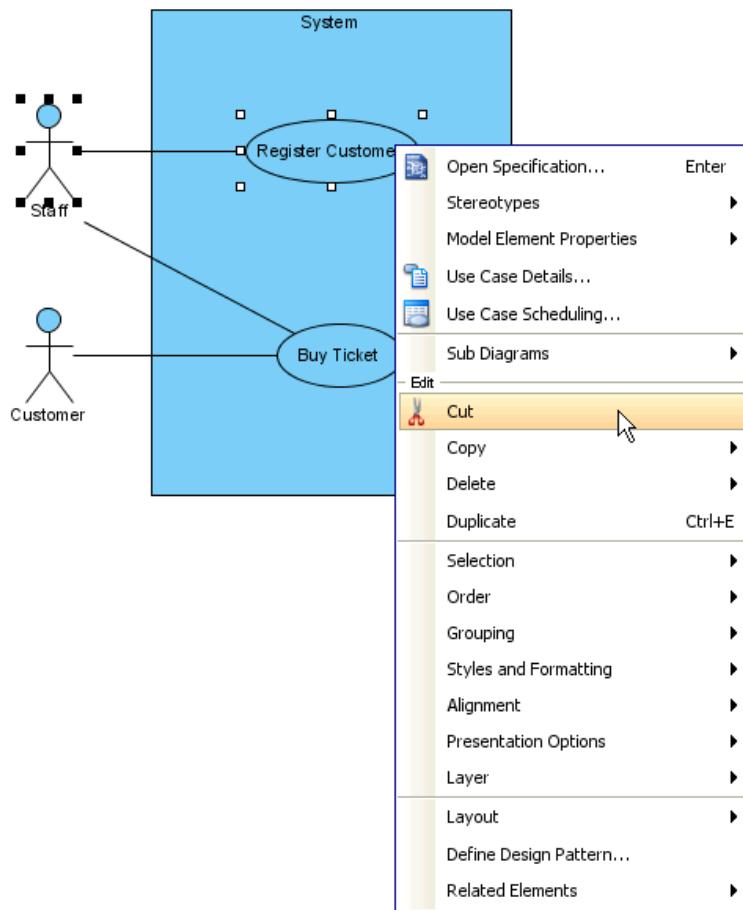


Figure 2-53 Cut selected shapes

2. On another diagram, right-click on empty area, select **Paste View** from popup menu.

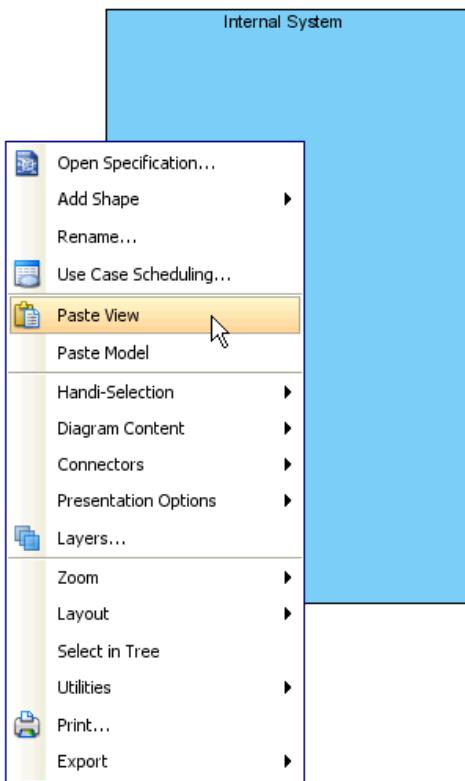


Figure 2-54 Paste the selected shapes on another diagram

3. Selected shapes are pasted.

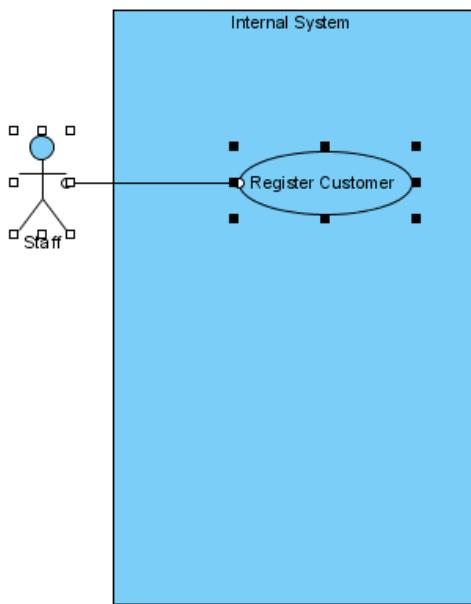


Figure 2-55 Shapes are pasted

4. You may found that the non-selected association is not shown on both diagrams, but the association is still exists, you can find that in open spec.

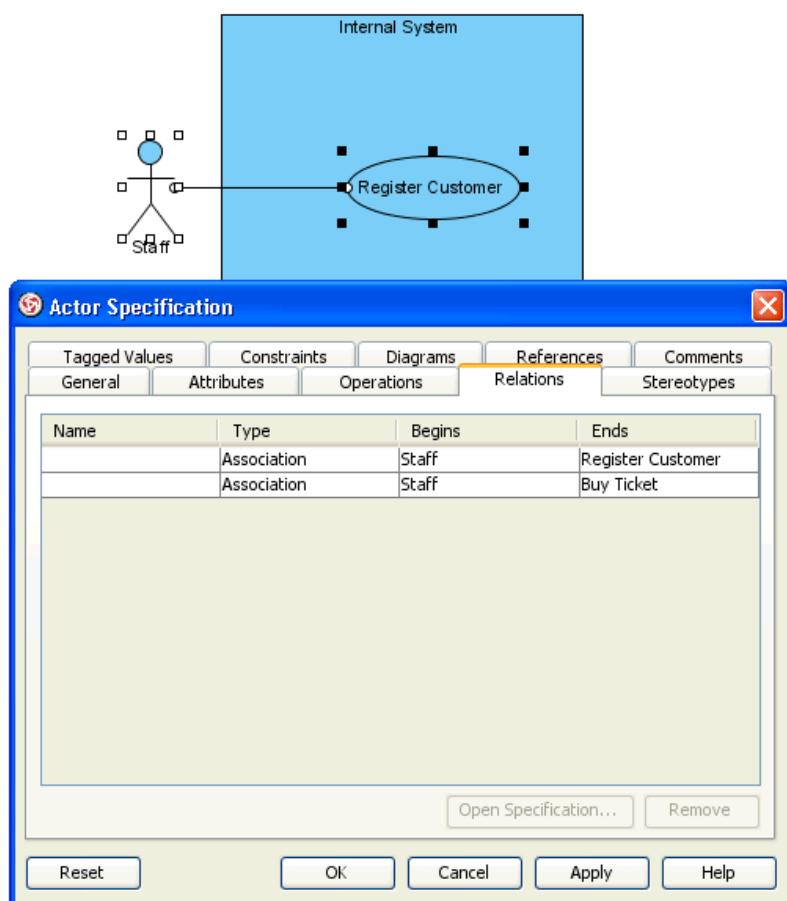


Figure 2-56 The non-selected association is still exists

Copy within VP-UML

Besides Cut and Paste, you can also Copy and Paste.

1. Right-click on the selected shapes, select **Copy > Copy within VP-UML EE**.

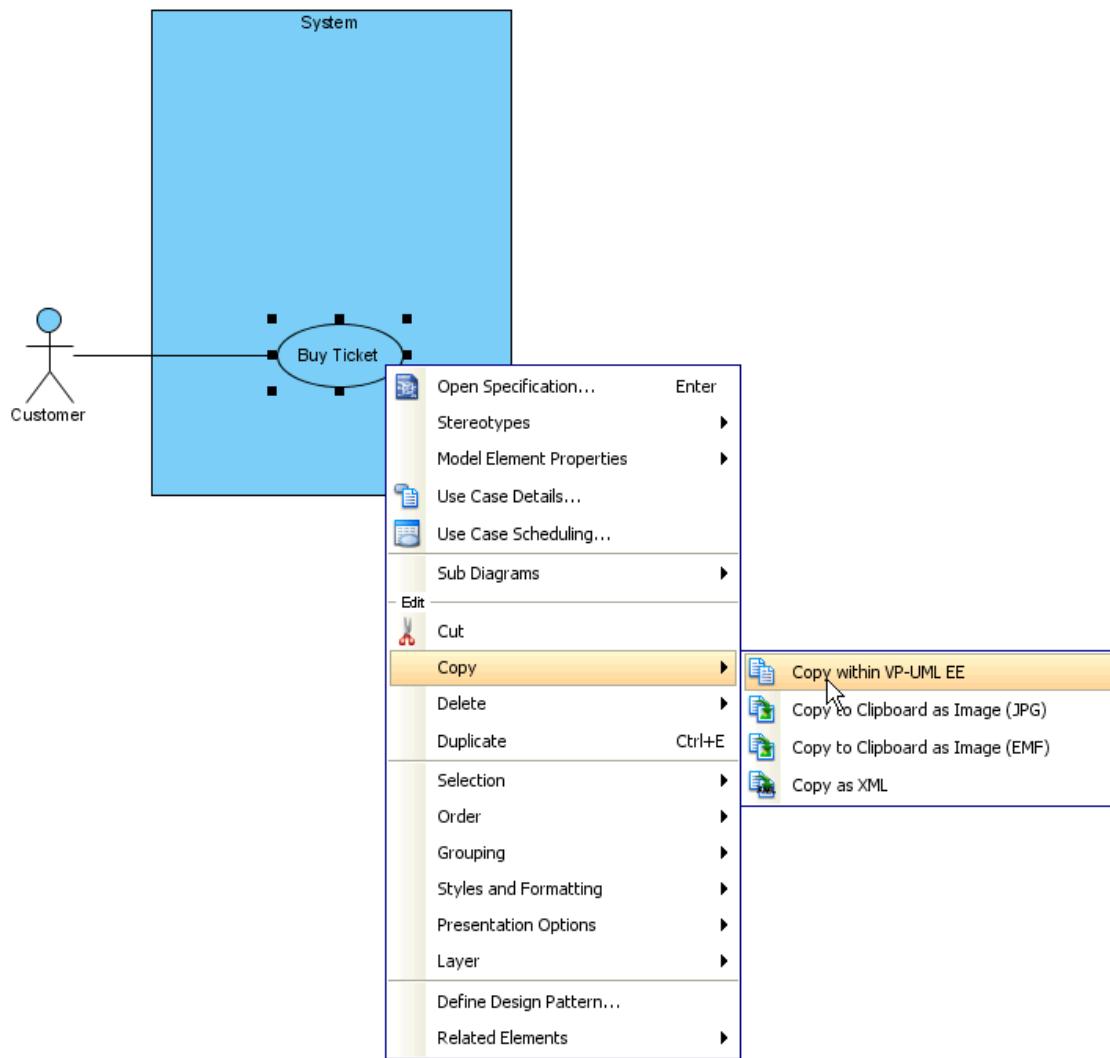


Figure 2-57 Copy selected shapes

2. On another diagram, right-click on empty area, select **Paste View** from popup menu.

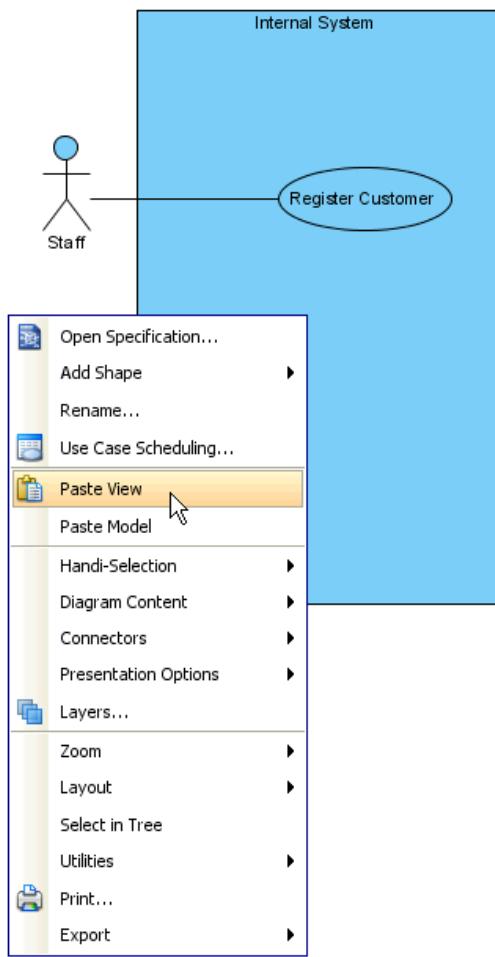


Figure 2-58 Paste the selected shapes

3. The shapes are pasted on diagram.

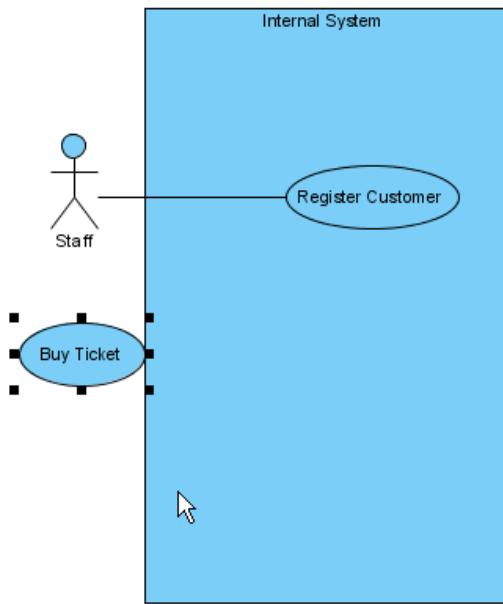


Figure 2-59 Shapes are pasted

4. You may found that, there is no Association shown between the *Staff* and *Buy Ticket*. To show the association, right-click on the shape. Select **Related Elements > Visualize Related Model Element....**

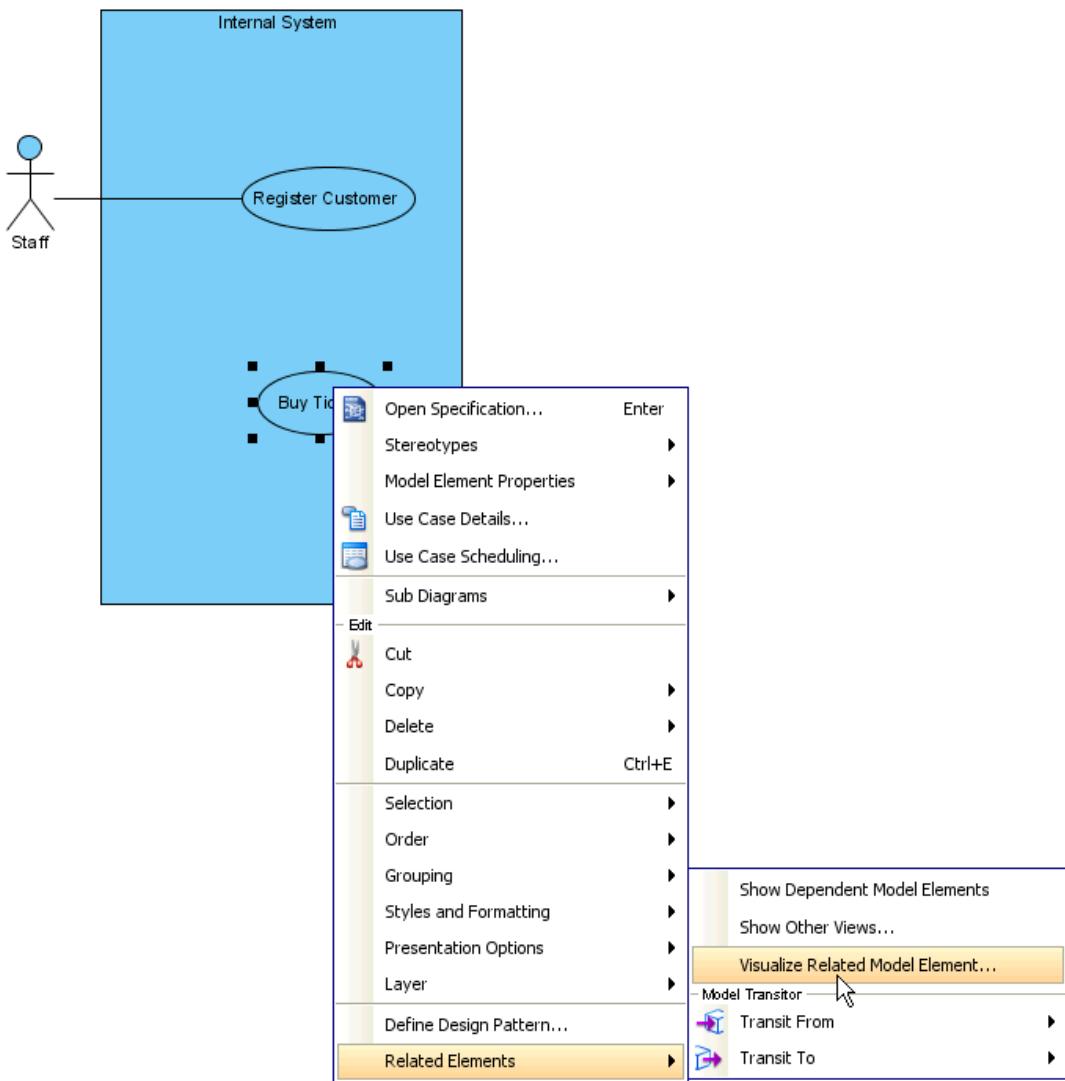


Figure 2-60 Right-click on the shape

5. Select the association with *Staff*, and click **Visualize**.

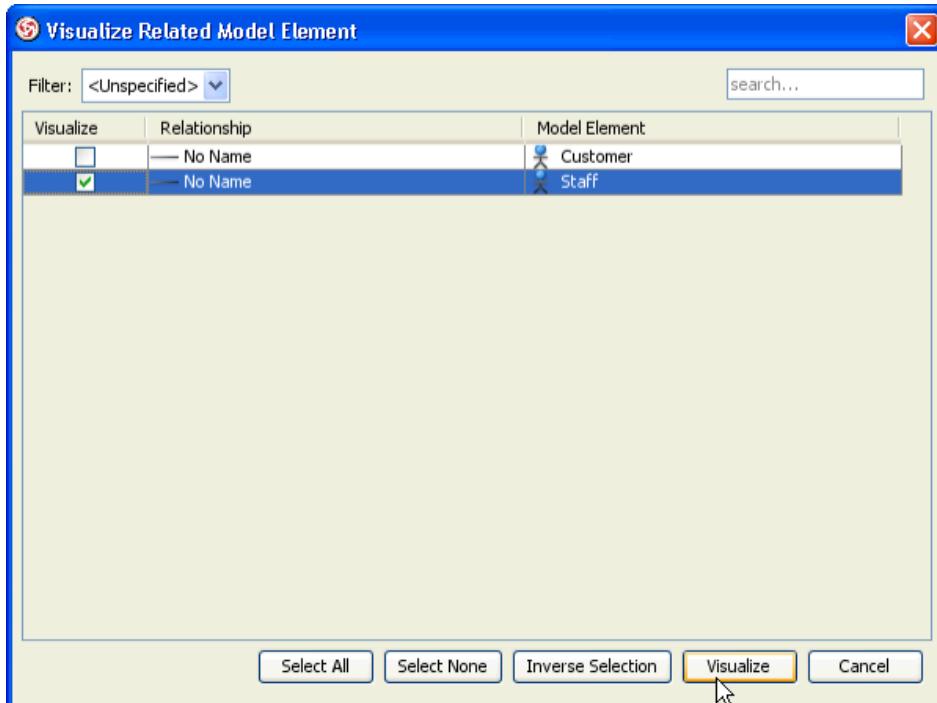


Figure 2-61 Visualize the association

6. The association is shown on the diagram.

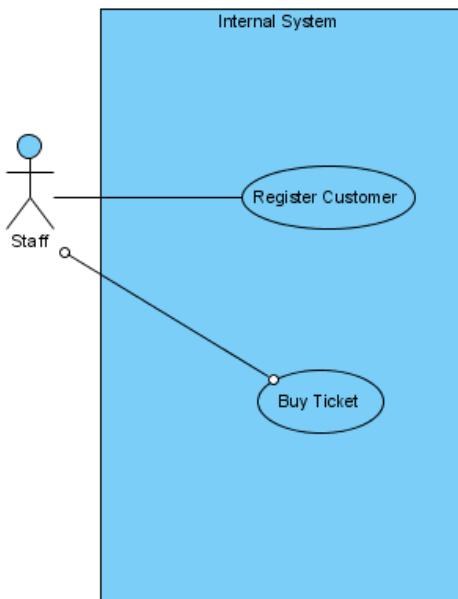


Figure 2-62 Association is shown

Copy to clipboard as image (JPG)

You can also copy the selected shapes as JPG image. Right-click on the selected shapes, select **Copy > Copy to Clipboard as Image (JPG)**.

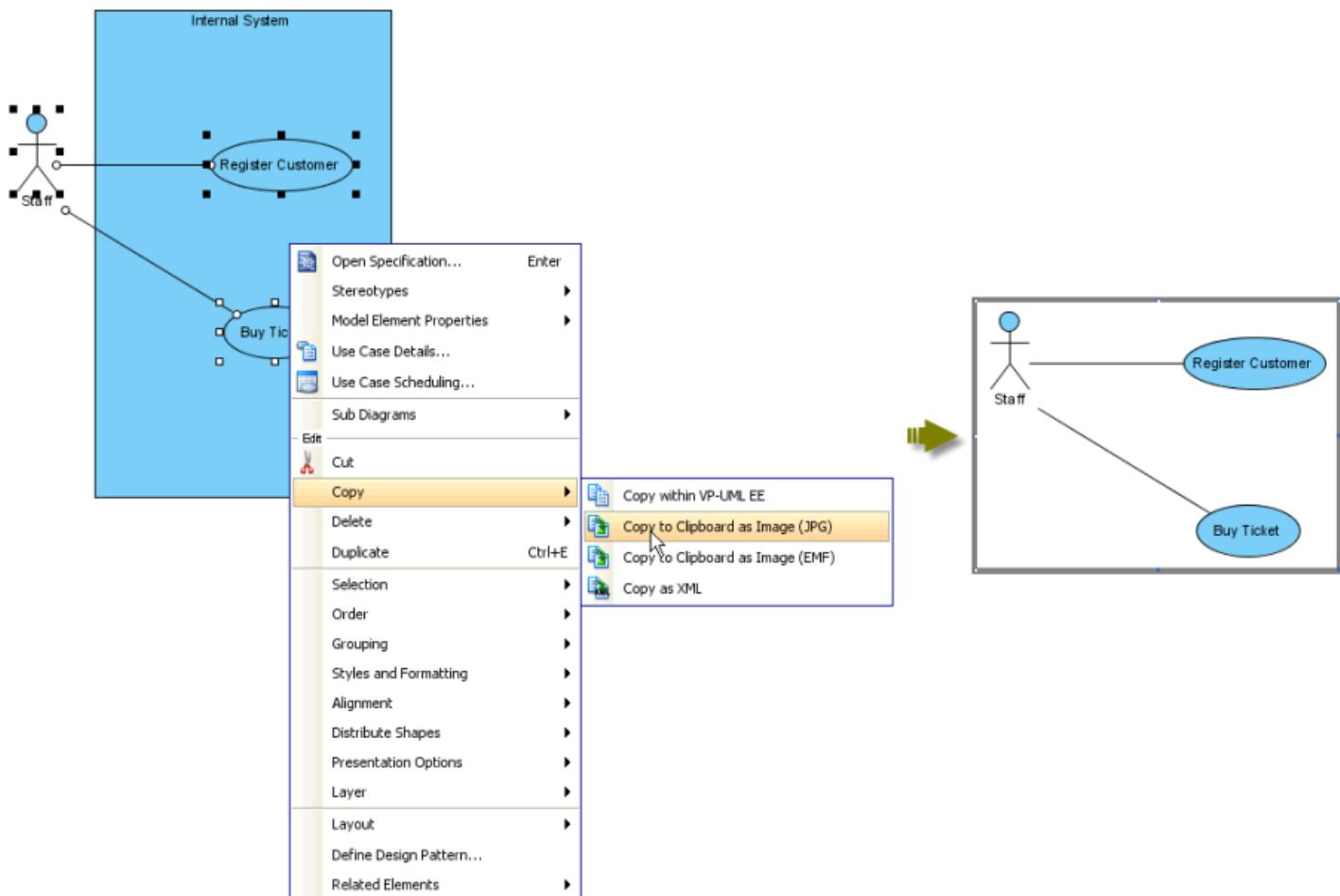


Figure 2-63 Copy to JPG image

Copy to clipboard as image (EMF)

You can also copy the selected shapes as EMF image. Right-click on the selected shapes, select **Copy > Copy to Clipboard as Image (EMF)**.

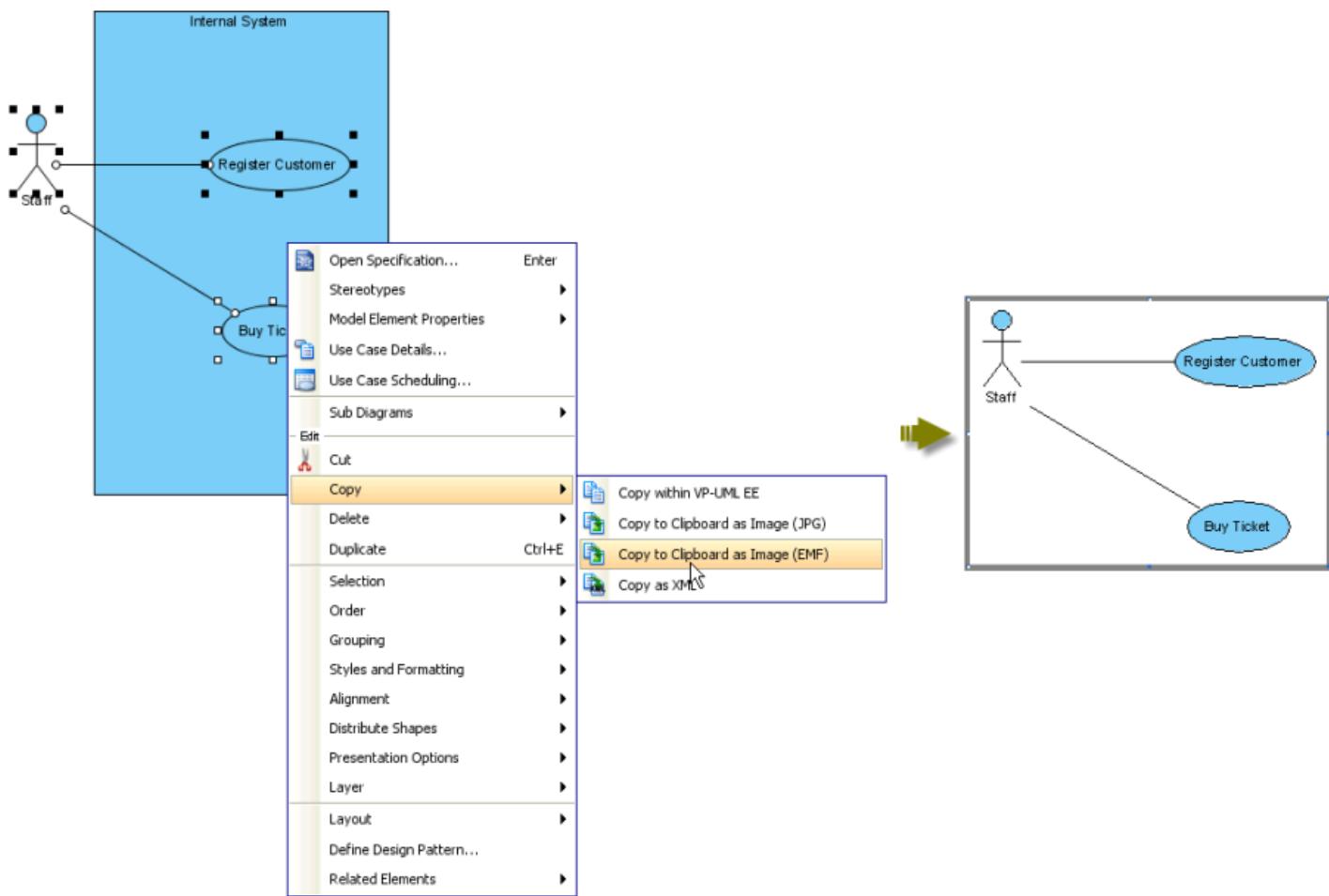


Figure 2-64 Copy to EMF image

Copy as XML

You can also copy the selected shapes as XML. Right-click on the selected shapes, select **Copy > Copy as XML**.

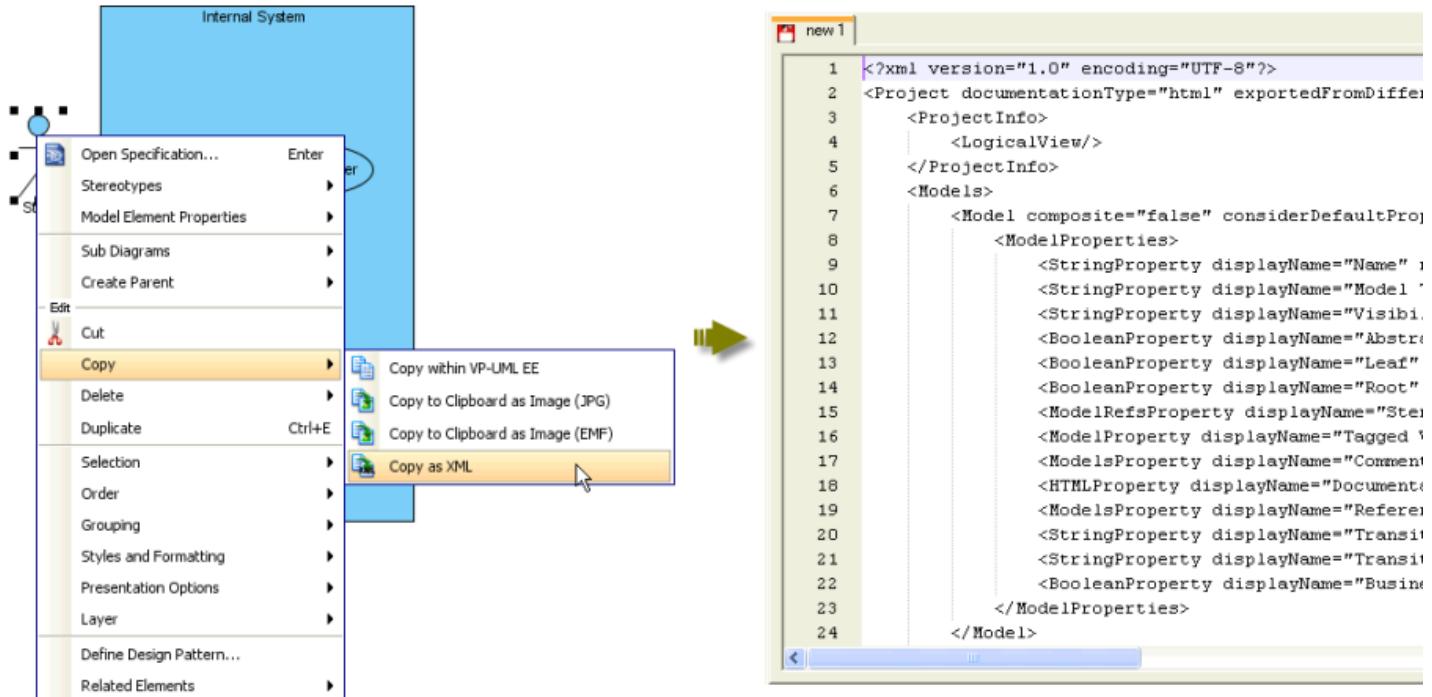


Figure 2-65 Copy to XML

Paste view

After you copy the model, you can paste in VP-UML.

1. Right-click on empty area on a diagram. Select **Paste View** from popup menu.

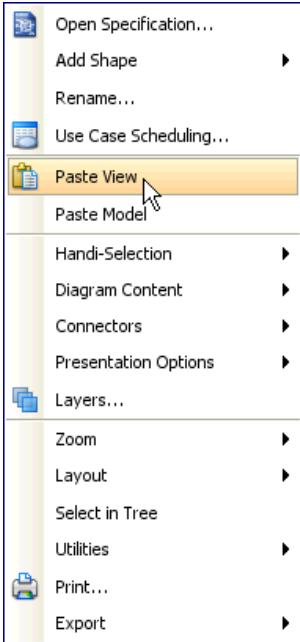


Figure 2-66 Paste View on diagram

2. The shape is pasted, no model will be created.

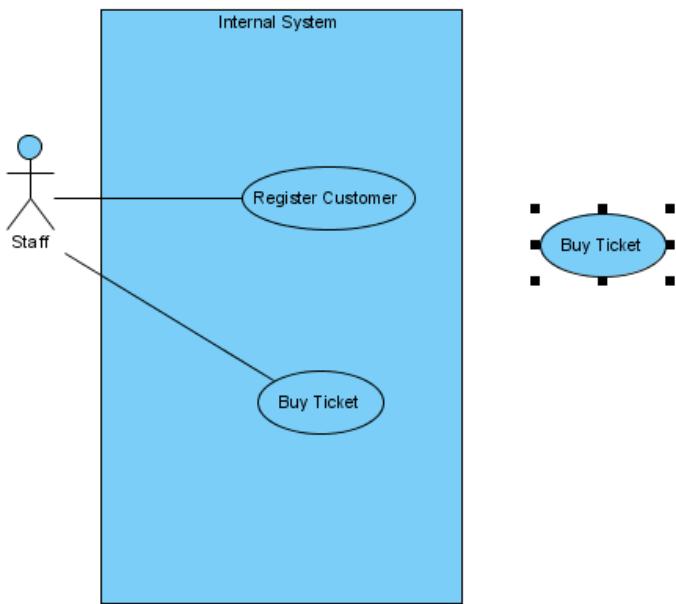


Figure 2-67 Shape is pasted

Paste model element

You can also paste with copying the model.

1. Right-click on empty area on a diagram, select **Paste Model** from popup menu.

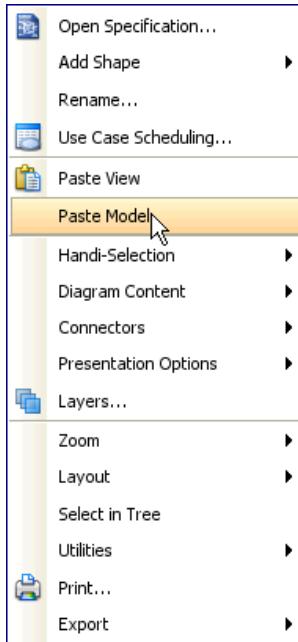


Figure 2-68 Paste Model on diagram

2. The model will be copied and pasted on the diagram. The new model will be named with appending a sequential number.

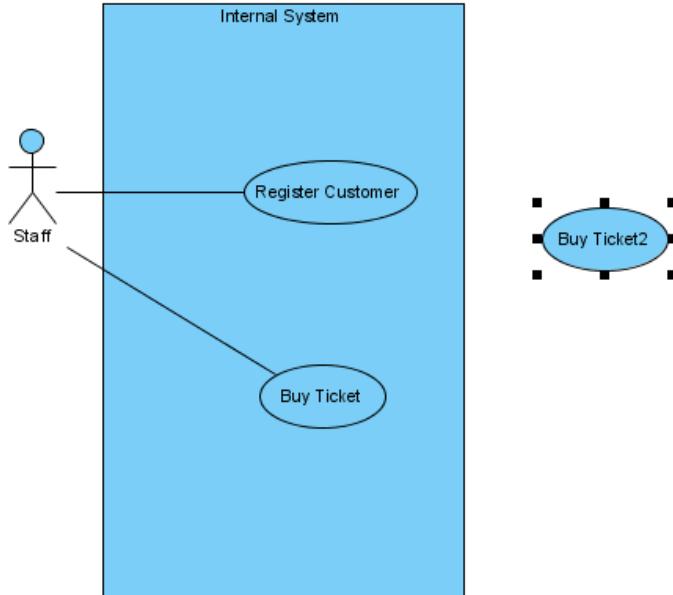


Figure 2-69 Model is pasted

Duplicate

You can duplicate a model on the diagram.

1. Right-click on the selected shapes, select Duplicate from popup menu.

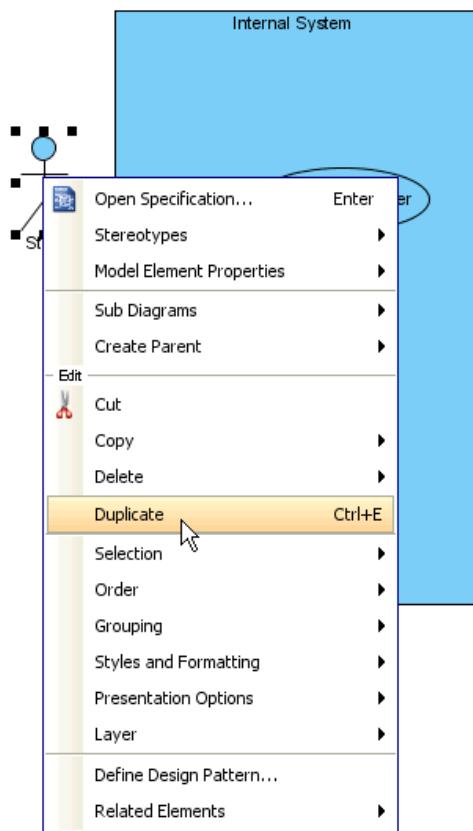


Figure 2-70 Duplicate the selected shapes

2. The model is copied and pasted on the diagram.

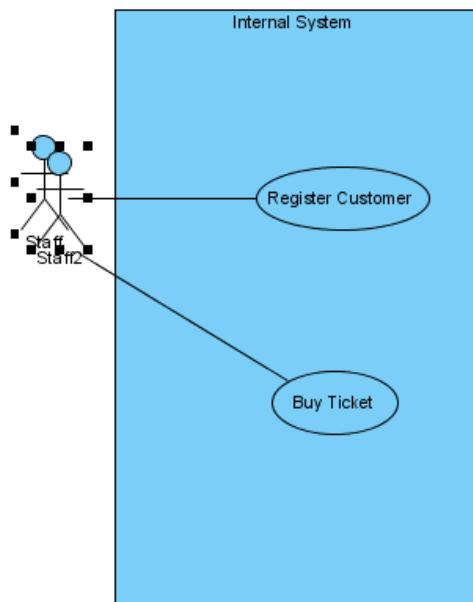


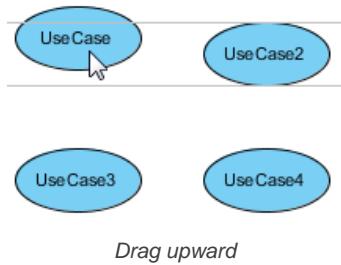
Figure 2-71 Model is duplicated

Alignment guide

Alignment is the adjustment of an object in relation with other objects. Therefore, the diagram alignment means adjusting a shape's position with another's (or others') in a straight line or in parallel lines. When you drag shapes, alignment guide will appear to let you position the dragging content in accordance with existing shape(s).

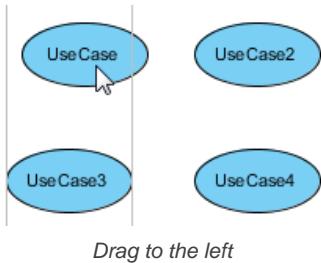
Show edges alignment guide

When you drag a shape upward or downward, two horizontal lines will reveal on the both edge of the shape as an offer of assistance for adjusting the shape's position.



Drag upward

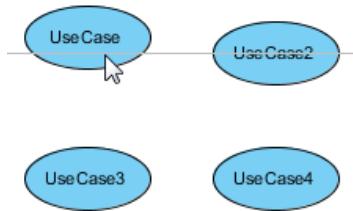
Similarly, when you drag a shape to the left or the right, two vertical lines will reveal on the both edge of the shape.



Drag to the left

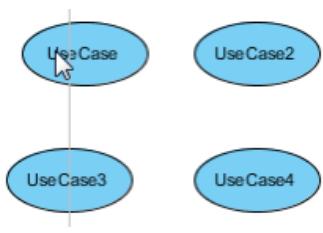
Show center alignment guide

When you drag a shape upward or downward, a horizontal line will reveal in the center of the shape as an offer of assistance for adjusting the shape's position.



Drag upward

Similarly, when you drag a shape to the left or the right, a vertical line will reveal in the center of the shape.



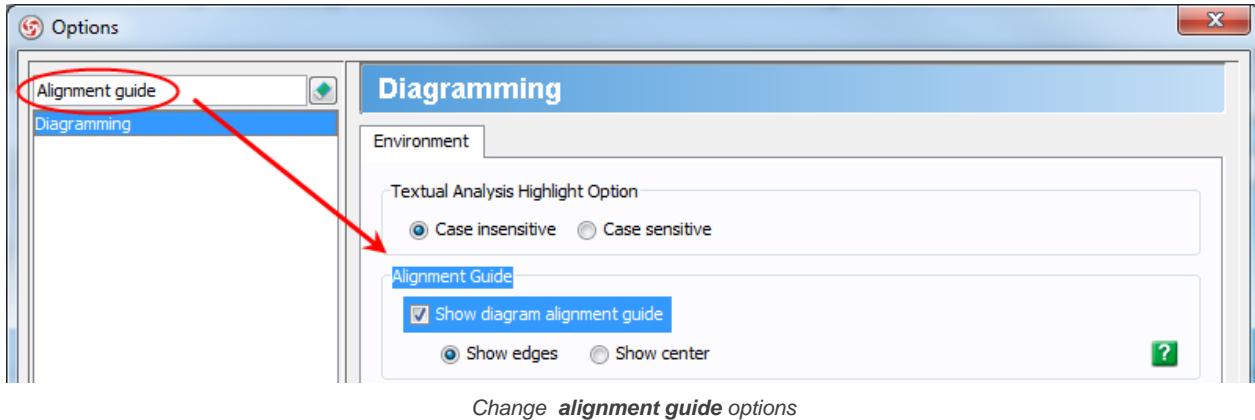
Drag to the left

Change alignment guide options

If you want the alignment guide to help you position when you drag shapes, check **Show diagram alignment guide**; on the contrary, uncheck it if you don't need a help. Two Choices are provided in alignment guide: **Show edges** and **Show center**. Choose **Show edges** if you want two horizontal lines to reveal on the both edge of the moving shape while choose **Show center** if you want a horizontal line to reveal in the center of the shape.

1. Select **Tool > Options...** from the main menu to unfold **Options** dialog box.

2. In shortcut, typing **Alignment guide** on top-left text field for searching. It will be shown that the option of **alignment guide** is on **Diagramming** category, **Environment** tab shortly. As you see, the option will be highlighted for your convenience.



Change **alignment guide** options

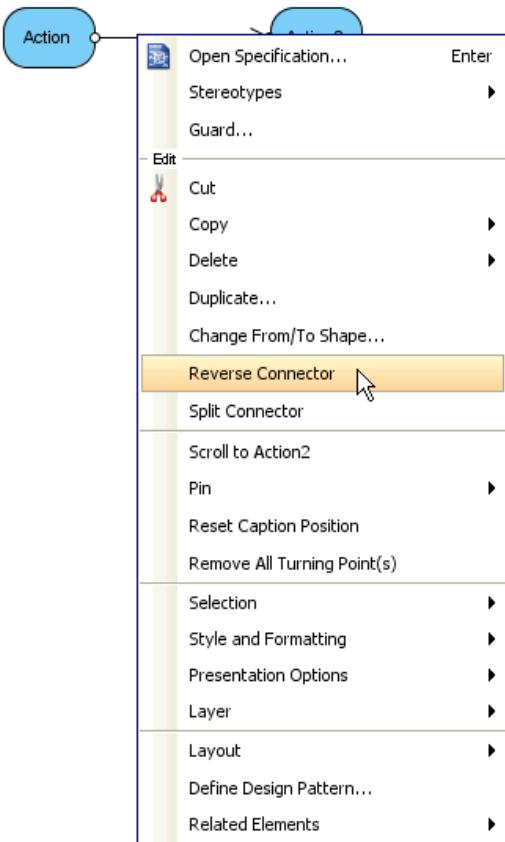
3. The option for **alignment guide** is checked as default. The next step you need to do is to choose either **Show edges** or **Show center** for your diagrams.
4. Uncheck **Show diagram alignment guide** if you do not want the assistance of alignment guide when dragging shapes.

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.

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: The function of reverse connector is not only for reverting the connector's direction, but also for repositing 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 those information.

NOTE: Connectors like create message in sequence diagram is not revertable

Visualize model element to diagram

VP-UML supports models and diagrams. A model may be shown on more than one diagram. A diagram may also show a model more than one time.

Drag model element from tree to diagram

To show a model on a diagram, you may drag the model from **Model Explorer** (or **Diagram Navigator**) into a diagram.

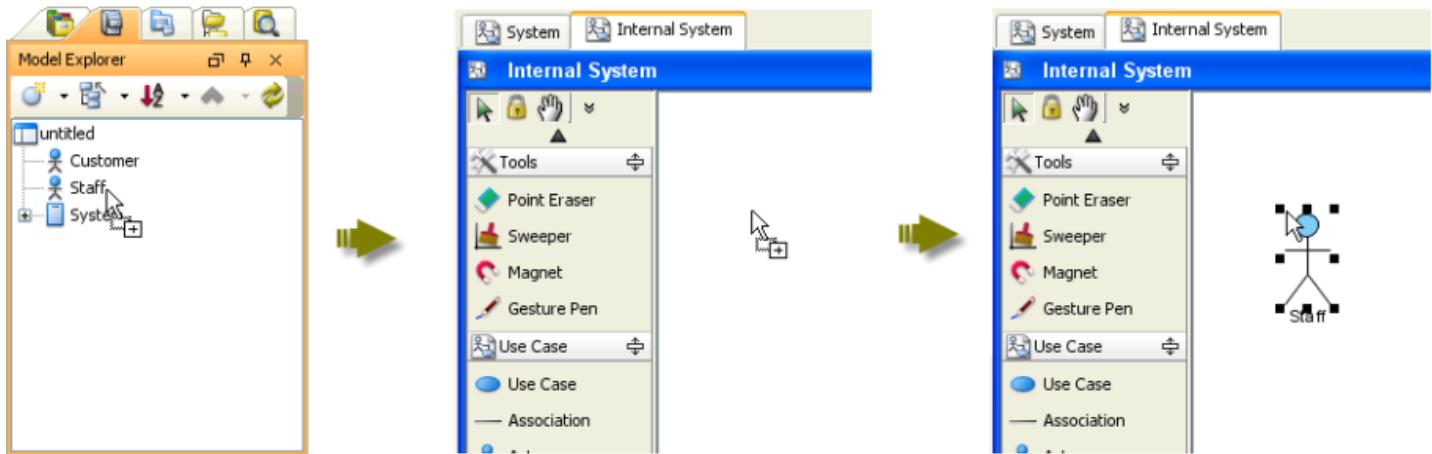


Figure 2-86 Drag and drop from tree to diagram to show a model on diagram

Visualize related model element

You may also want to show another models which related with the models on the diagram.

1. To do so, select the model on diagram, right-click and select **Related Elements > Visualize Related Model Element...** from popup menu.

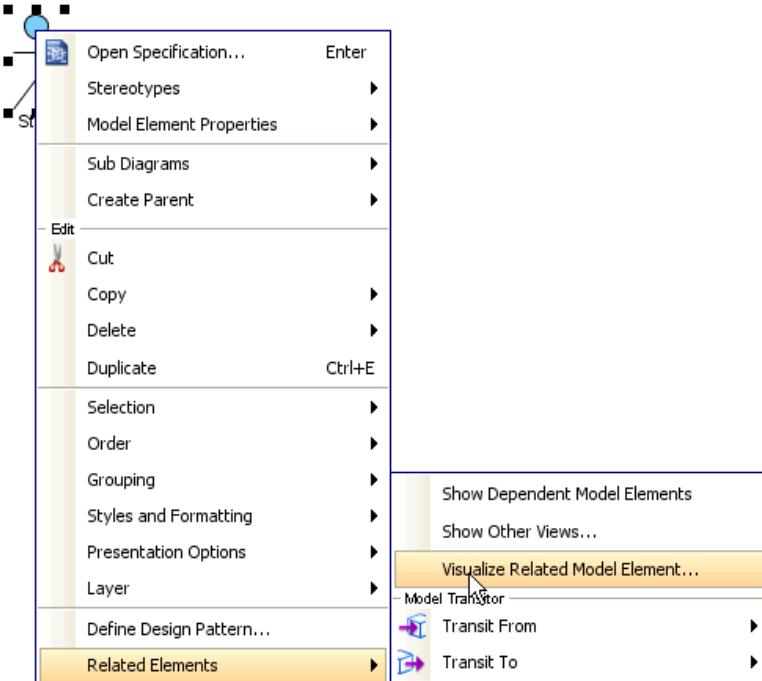


Figure 2-87 Visual Related Model Element

2. Select the related model elements you want to be shown on the diagram and select **Visualize**.

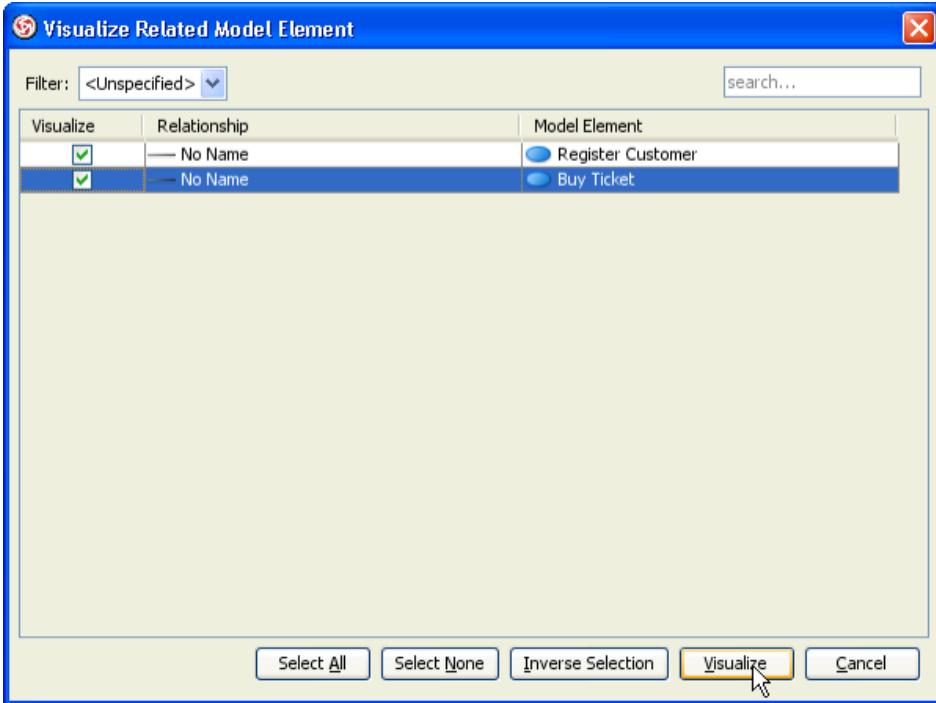


Figure 2-88 Visualize Related Model Element dialog box

3. The related models are shown on the diagram.

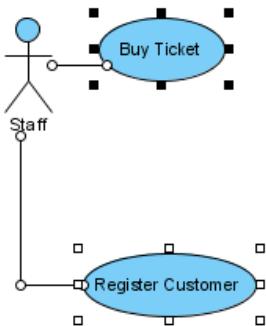


Figure 2-89 Related model elements are visualized

Model commenting

VP-UML supports comments on models.

Adding comment to model element

1. To add comment into model, right-click on model and select **Open Specification...** from popup menu.

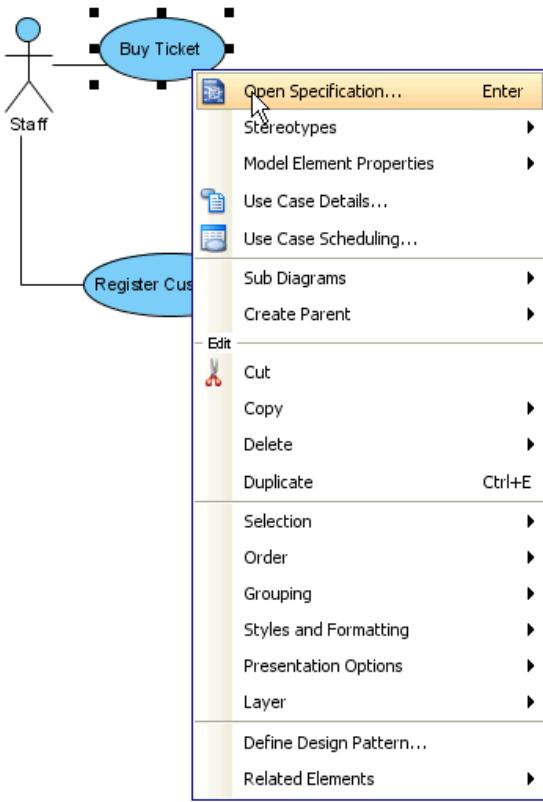


Figure 2-90 Open Specification

2. Select **Comments** tab.

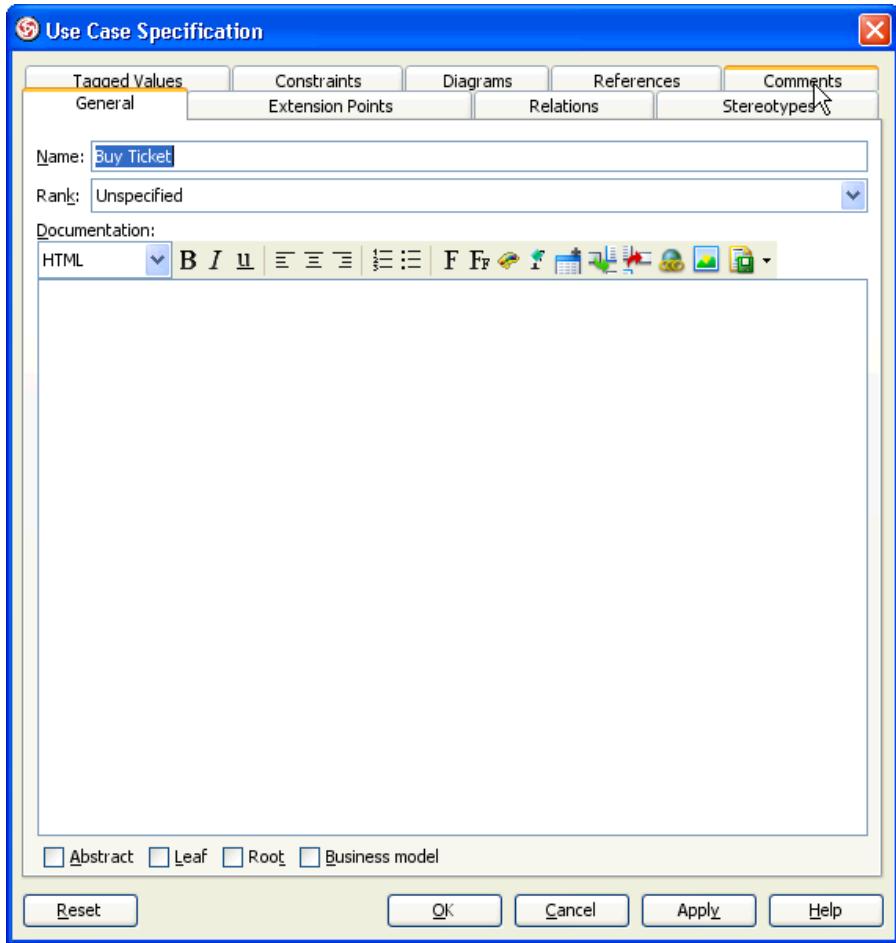


Figure 2-91 Comments tab

3. Select **Add** button to create a comment.

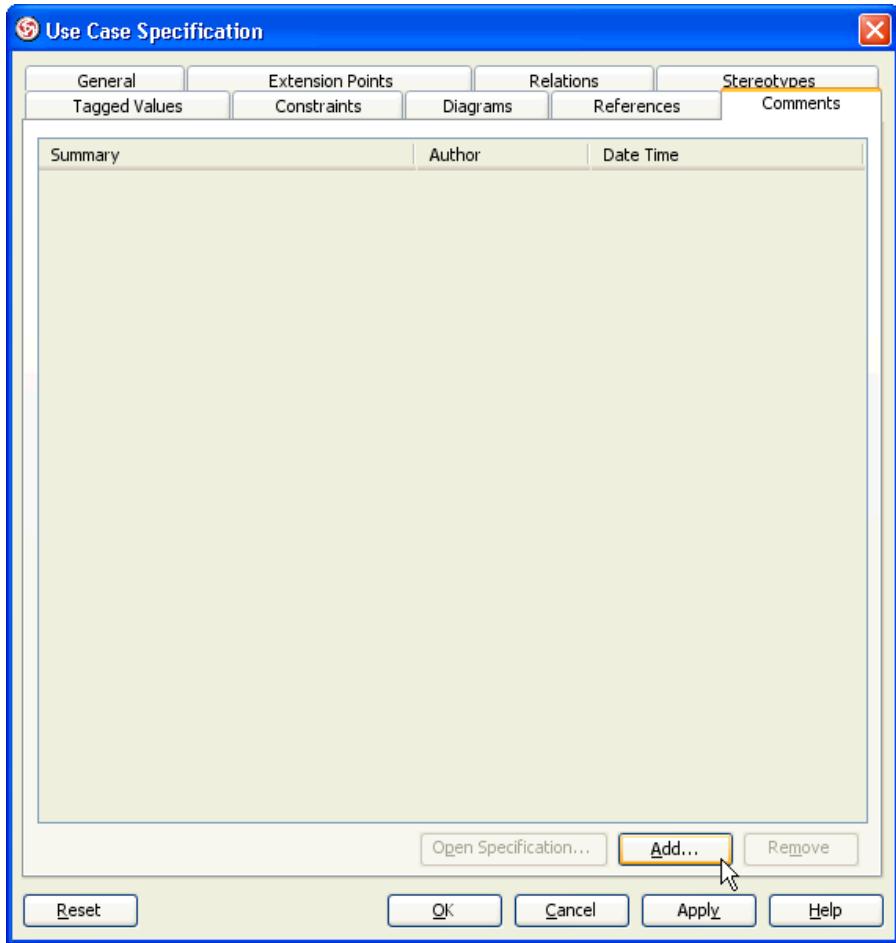


Figure 2-92 Add comment

4. Enter the name, author, etc... of the comment. And then select **OK** button to confirm adding the comment.

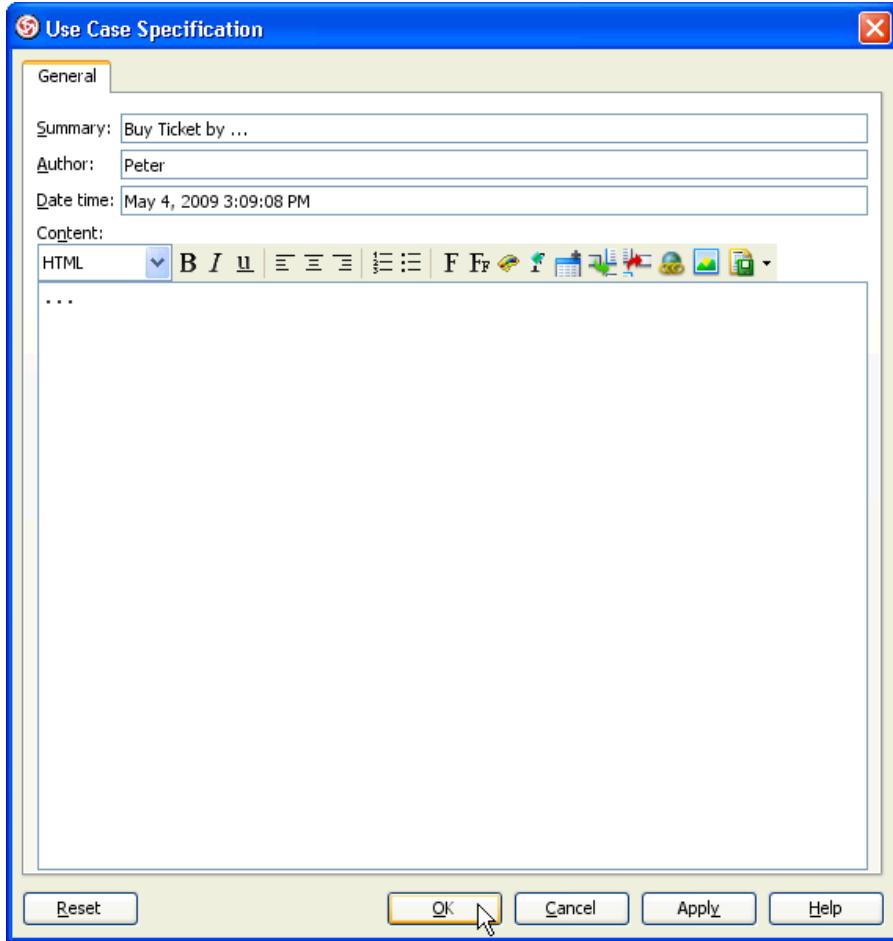


Figure 2-93 Contents of comment

5. Comment(s) is shown on **Comments** tab.

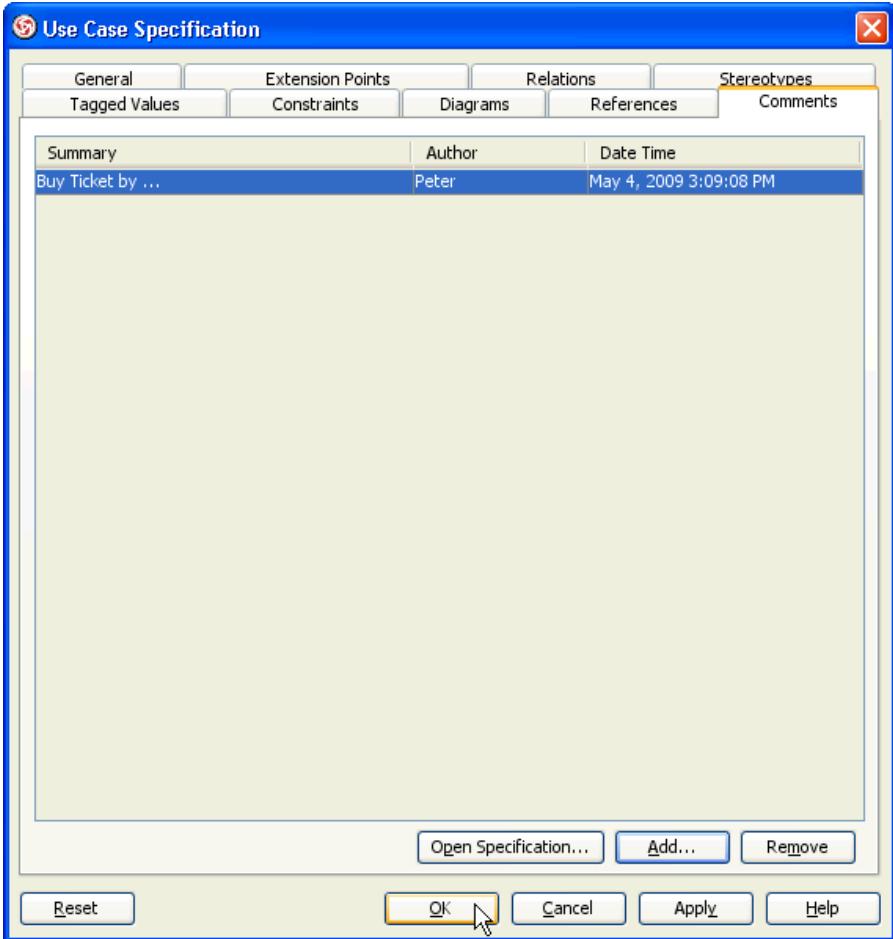


Figure 2-94 Comments shown

Managing comment of model element

1. To modify the comment, also on **Specification** dialog box, **Comments** tab, select the comment and select **Open Specification...** button.

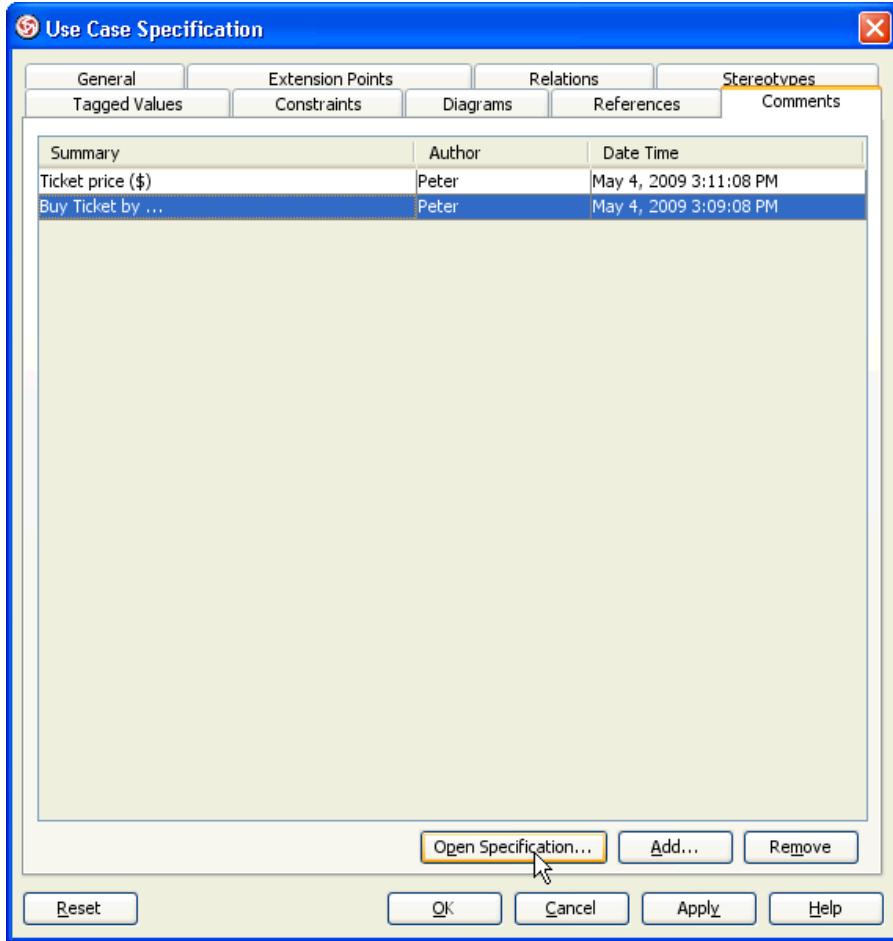


Figure 2-95 Open Specification of the comment

2. Modify the name, author, etc... of the comment and select **OK** to confirm changing.

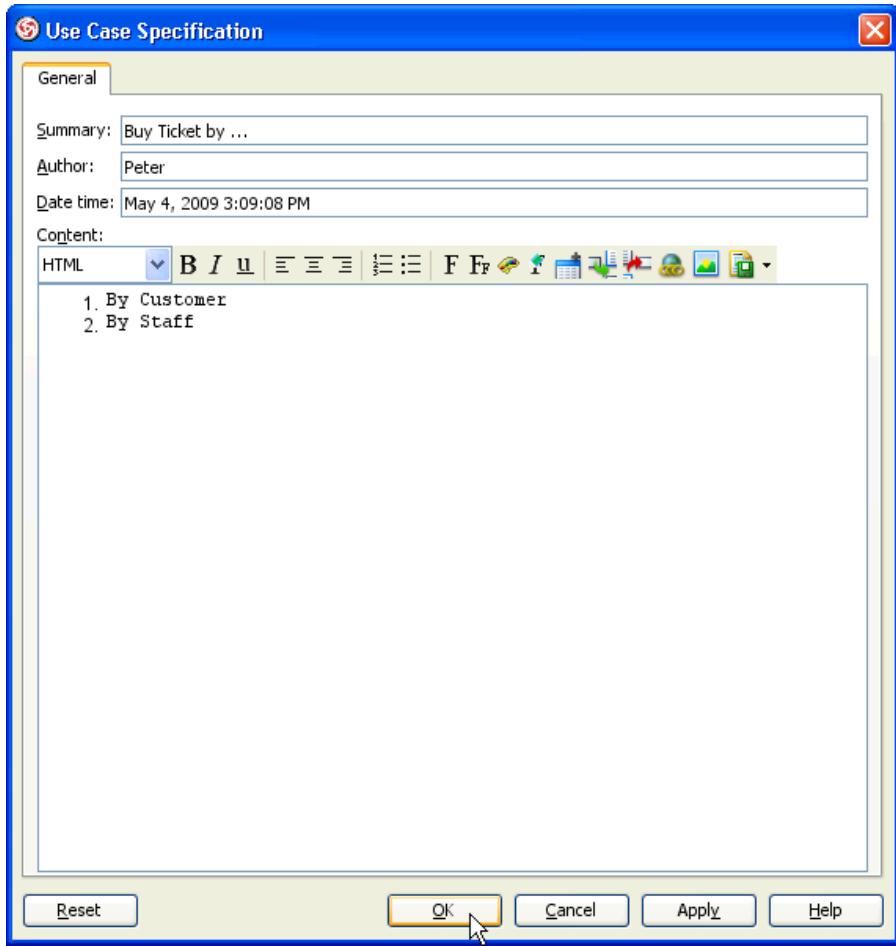


Figure 2-96 Modify contents of comment

Pinnable connector end

Connector can be pinned with its shape.

Temporary adjust connector end

1. You may want the connectors connect with shape at same point on the diagram. To do so, drag the connector's start/end point.

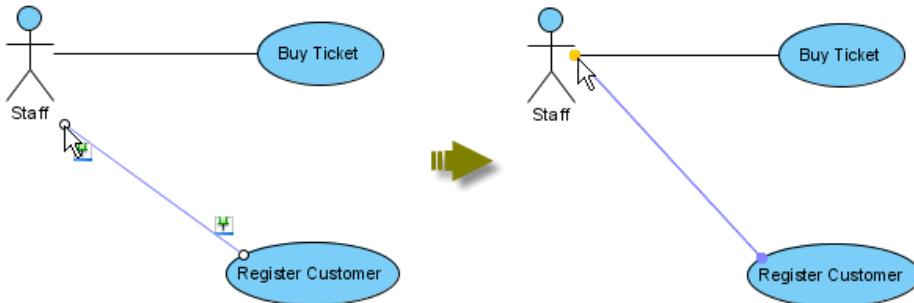


Figure 2-97 Drag connector point

2. The connector is temporary pinned. A description will be shown on diagram to describe how to pin the connector.

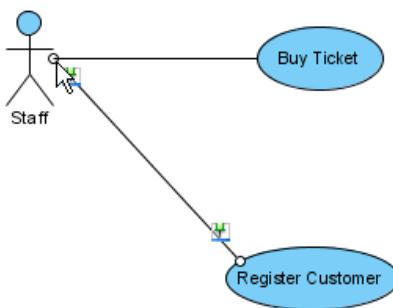


Figure 2-98 Connector is temporary pinned

3. But moving the from/to shapes of the connector, the connector will be unpinned.

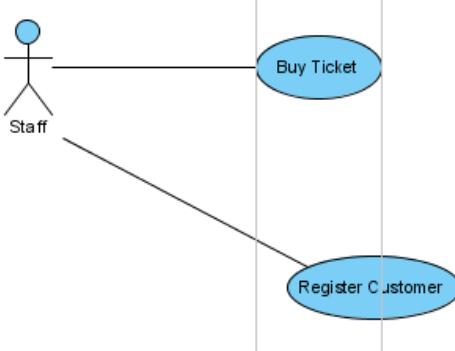


Figure 2-99 Connector is unpinned automatically

Pin connector end

1. To pin the connector, select the connector and select the **Pin** resource.

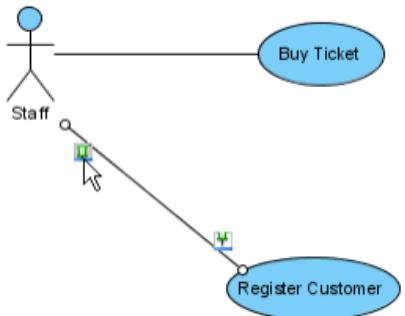


Figure 2-100 Pin the connector by selecting Pin resource

2. Drag the connector point again, no description of temporary pin will be shown this time.

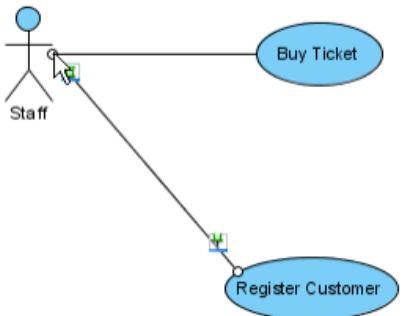


Figure 2-101 Drag connector point

3. Moving the shape, the connector won't be unpinned.

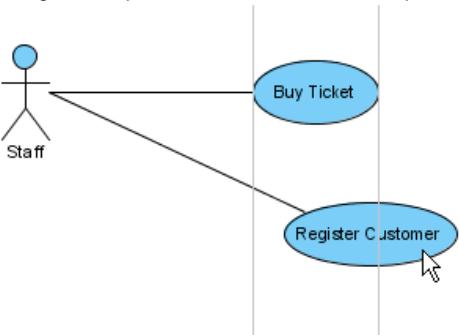


Figure 2-102 Connector is still pinned

Align and distribute diagram elements

This feature provides a facility to align selected diagram elements. You can align using toolbar, popup menu or group resource.

Aligning diagram elements

To align diagram elements, select the diagram elements on diagram.

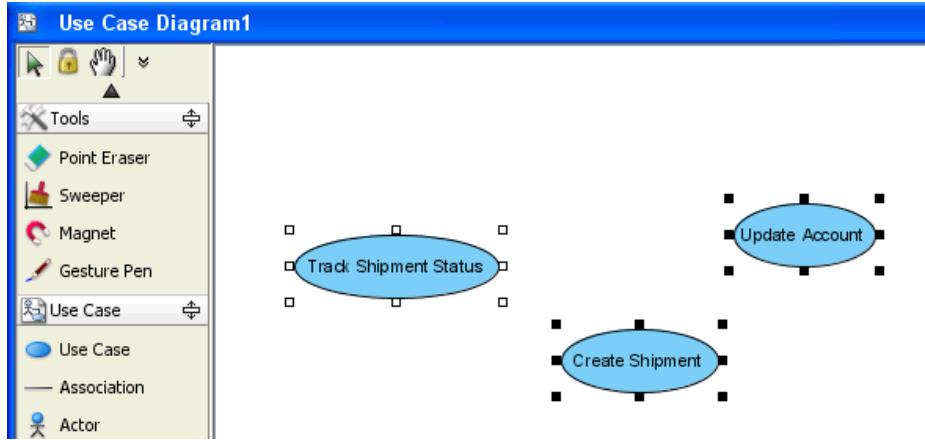


Figure 2-103 Select shapes

Using toolbar

1. Select alignment on toolbar.

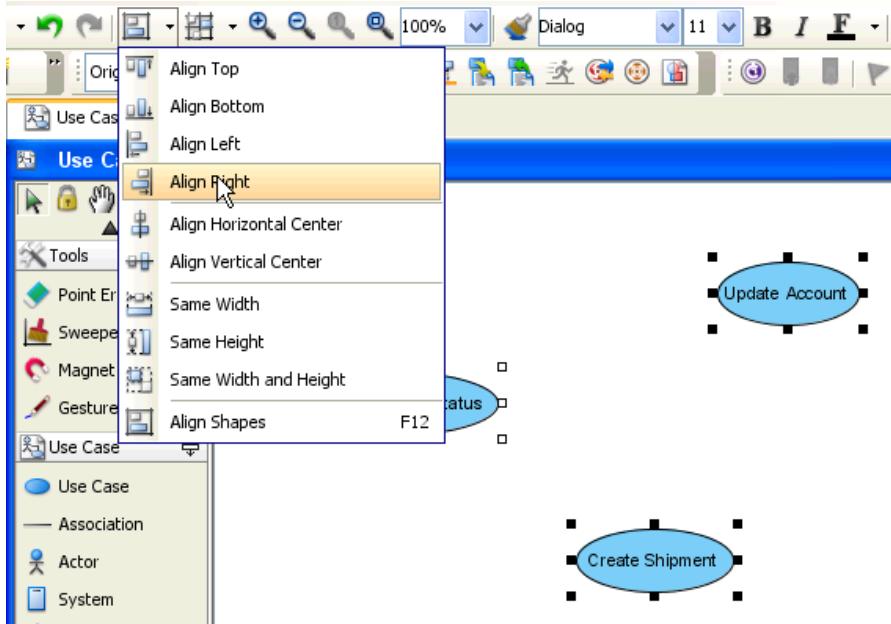


Figure 2-104 Align Right

2. The shapes are aligned to right. They are based on the shape with no-filled selector.

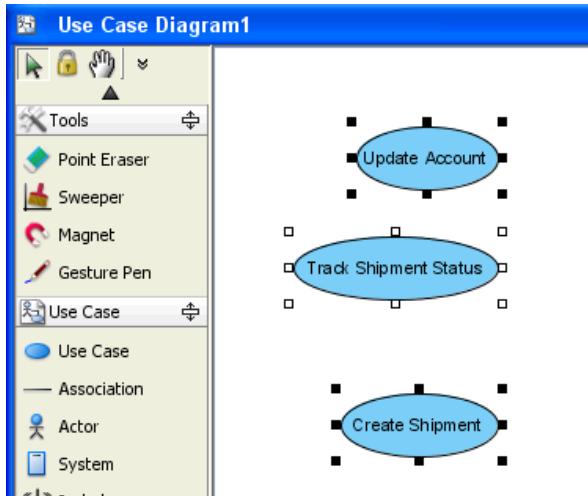


Figure 2-105 Shapes are aligned following the no-filled selector shape

3. To change the no-filled selector shape, you may press Ctrl key and click the shape 2 times to deselect and select the shape.

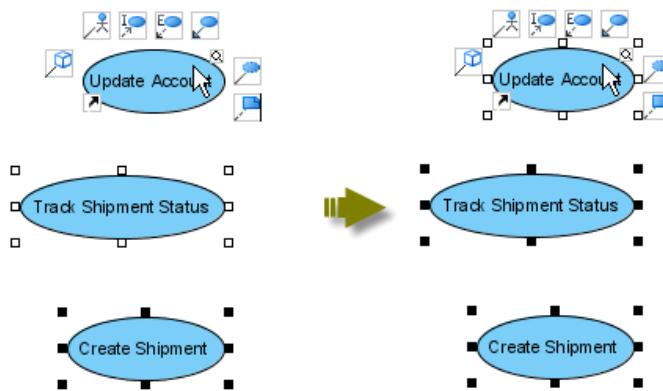


Figure 2-106 Change shape to be no-filled selector shape

Using grouping resources

1. Mouse over on any selected shape, Grouping Resources will be shown.

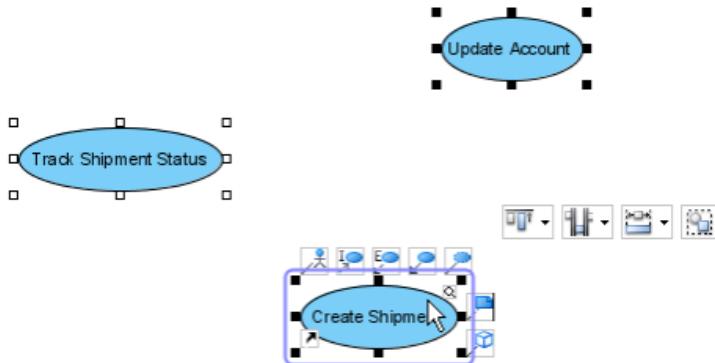


Figure 2-107 Grouping Resources shown on selected shape

2. Select alignment on the Grouping Resources.

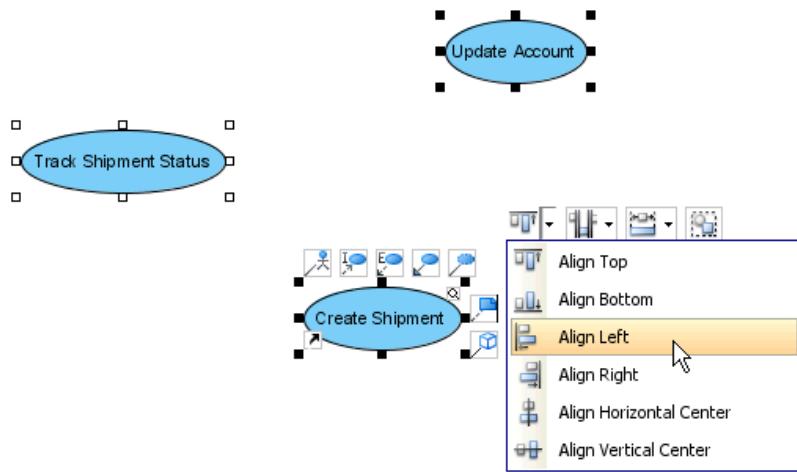


Figure 2-108 Select Align Left on resources

3. The selected shape will become no-filled selector shape and the alignment will follow the selected shape.

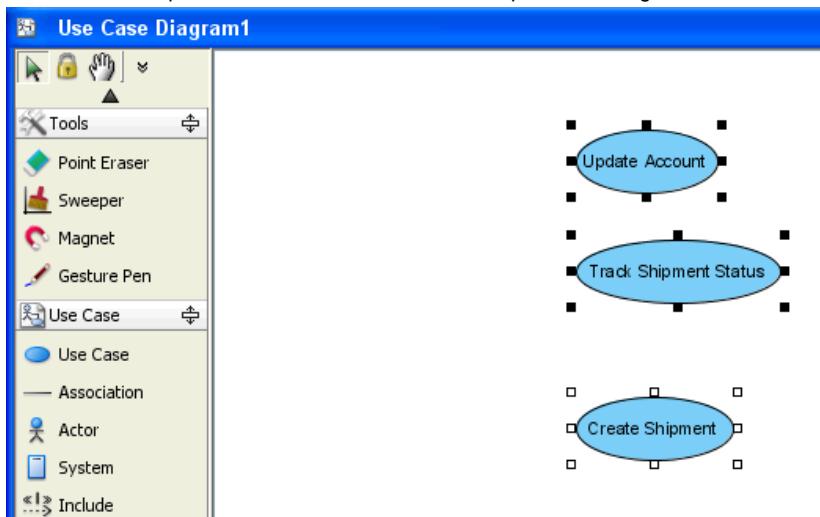


Figure 2-109 Shapes are aligned

Making diagram elements same width and height

Besides align the shapes, also can resize the shapes.

Using toolbar
Select **Same Width and Height** on toolbar.

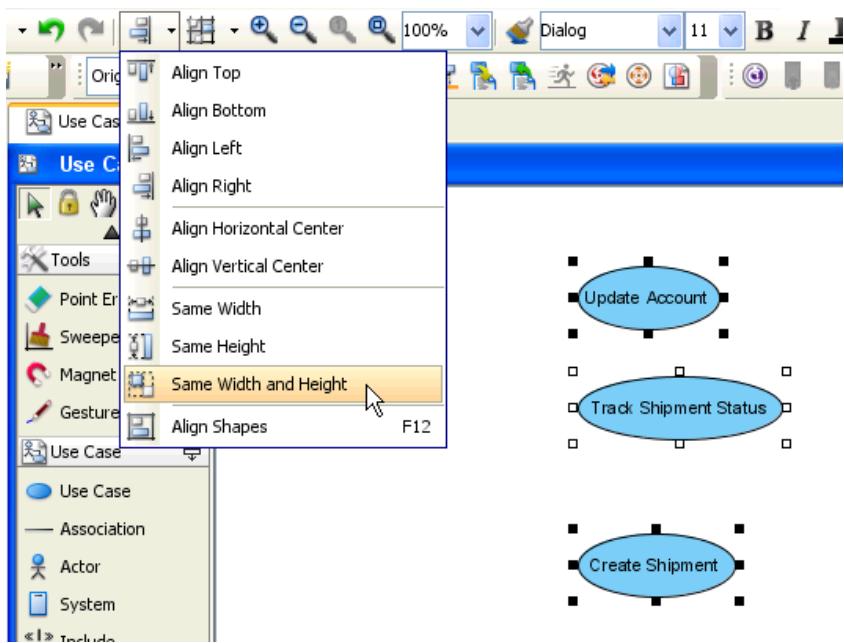


Figure 2-110 Resize by toolbar button

The shapes are resized.

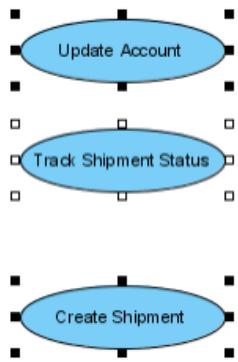


Figure 2-111 Shapes are resized

Using grouping resources

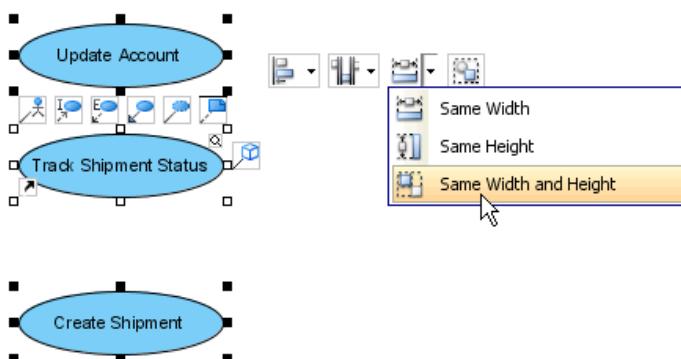


Figure 2-112 Resize by grouping resources

Using align shapes dialog box

To show Align Shapes Dialog, you may select Align Shapes on toolbar or select Edit > Align Shapes > Align Shapes... in menu.

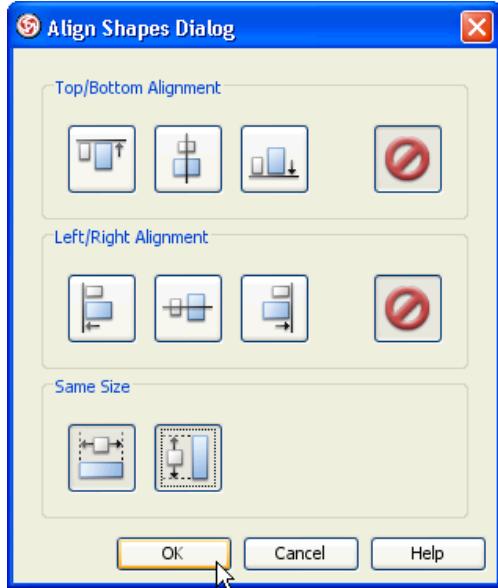


Figure 2-113 Select Same Width and Same Height on Align Shapes Dialog

Distribute diagram elements

Besides alignment and resize, you can distribute the diagram elements.

Using toolbar

Select Distribute Shapes Vertically.

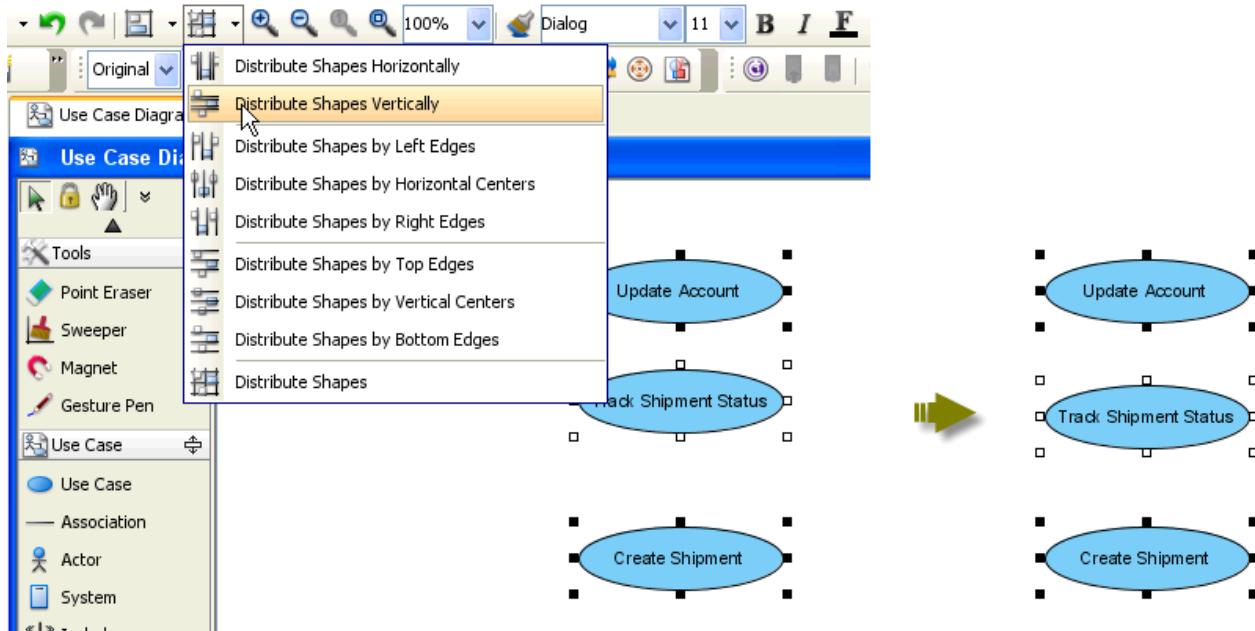


Figure 2-114 Shapes are located with same distance

Using grouping resources

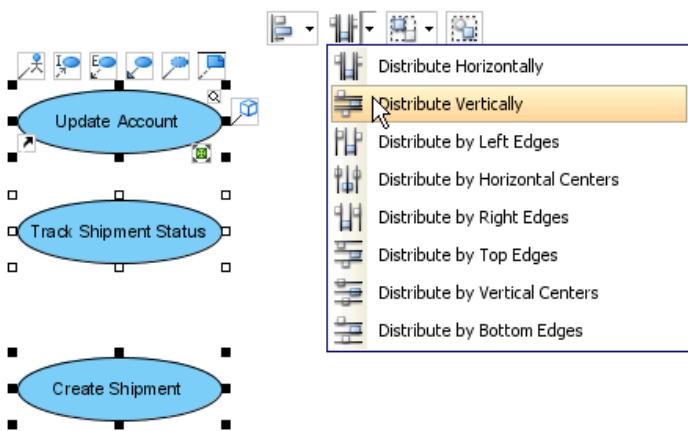


Figure 2-115 Distribute shapes in Grouping Resource

Annotate diagram elements with UML note shape

Creating UML note with resource centric

1. Move the mouse cursor over Anchor - > Note resource.



Figure 3-1 Mouse over resource

2. Drag resource to empty space on diagram pane.

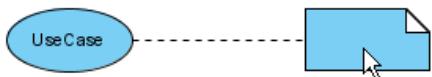


Figure 3-2 Dragging resource

3. Release the mouse, new connector and note are created.



Figure 3-3 Note created

Connecting UML note shape with anchor

1. Click Anchor in diagram toolbar.

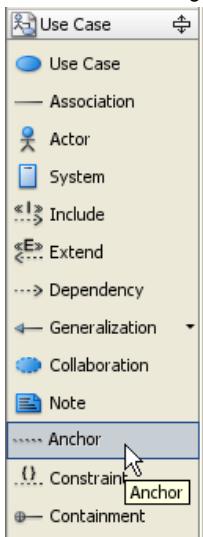


Figure 3-4 Diagram toolbar

- Move the mouse cursor over the shape to connect from.



Figure 3-5 Connecting from shape

- Drag to the note to connect to.



Figure 3-6 Conencting to note

- Release the mouse, an anchor is created to note.



Figure 3-7 Anchor created

Editing UML note content

- Double click the note to start editing its content.

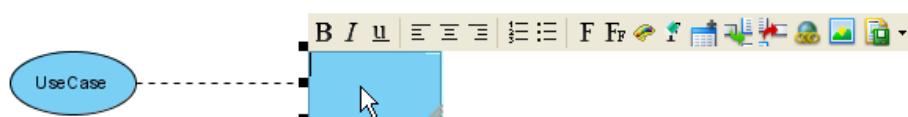


Figure 3-8 Start editing

- Drag the bottom right corner to resize it.



Figure 3-9 Resizing

3. Click on the diagram to confirm editing.

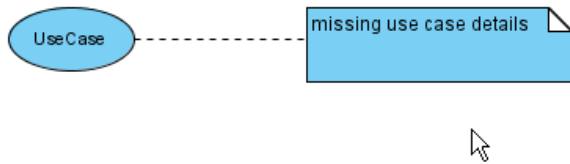


Figure 3-10 Confirm editing

Callout shape

Callout shape is a label that is used for explanation on your diagram elements. Inserting callout out shape aims to draw your customers' attention and give them additional remarks. You may find its function is similar to a photo caption or a comment. In fact, it is alike to a dialogue box that you can place it next to your diagram elements. Moreover, it does more than merely a photo caption or a comment.

To adjust the direction of callout shape pointer

1. Clicking **Callout** button in diagram toolbar and move it to the place you want to insert your remark.
2. You can adjust the direction of callout shape pointer, simply drag its end.

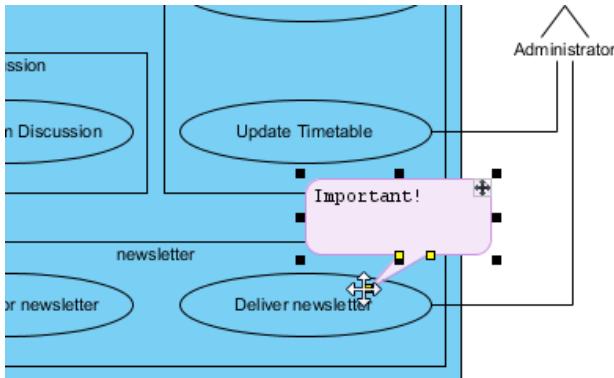


Figure 3-11 Adjusting the pointer

3. You may want to adjust the pointer to point to a more specific position, such as pointing to the name of diagram element, or a specific class member (attribute/operation) out of a class.

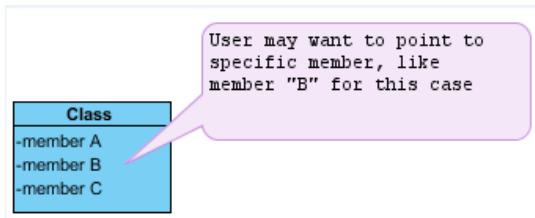


Figure 3-12 Adjusting the pointer to a specific class member out of a class

NOTE: You can not only adjust the pointer's direction, but also its length.

To reposition the shape

If you don't satisfy with the position of the callout shape that you have moved, you can drag the + icon that is located on the top right corner of the callout to reposition the shape in a straight and simple way.

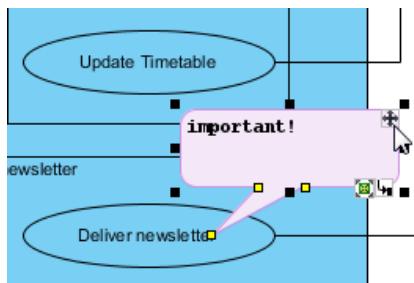


Figure 3-13 Repositioning the shape

NOTE: It only moves the callout shape and doesn't change the position of pointer by dragging the + icon. This helps you to remain the pointing and move the callout shape simultaneously.

To edit callout shape content

You can start editing by Double clicking on the callout shape.

To emphasize the content of callout shape

You may use any formatting buttons to insert formatting to the text in order emphasize your key points while you are editing the content, such as using bold type and changing the colour of the characters. This helps to enhance the visual effect of the text as well.

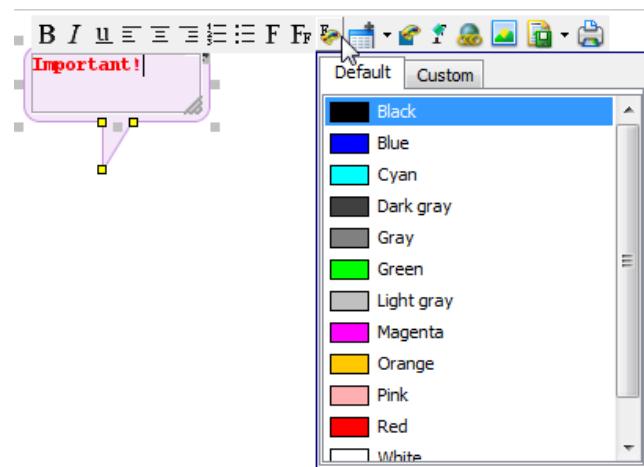
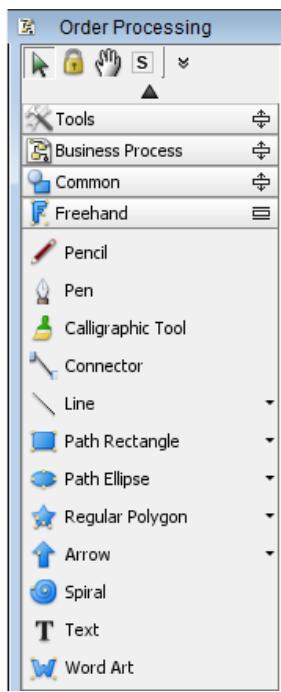


Figure 3-14 Formatting the text

Drawing freehand shapes

The application of freehand is mainly for annotation purpose. For instance, one of shape types in freehand can be used to emphasize how important is the diagram element.



Freehand toolbar interface

Summary of freehand shapes

Shape Type	Sample
Pencil	
Pen	
Calligraphic Tool	
Connector	
Line	
Label Line	
Path Rectangle	
Rectangle	

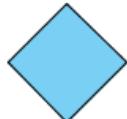
 Round Rectangle



 Round Rectangle 2



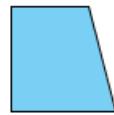
 Diamond



 Parallelogram



 Trapezoid



 Isosceles
Trapezoid



 Path Ellipse



 Ellipse



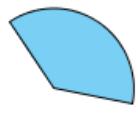
 Arc



 Chord



 Pie



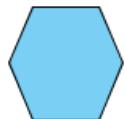
 Regular Polygon



 Isosceles Triangle



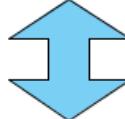
 Hexagon



 Arrow



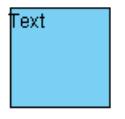
 Two Head Arrow



 Spiral



 Text



 Word Art



Summary of freehand shape

NOTE: Freehand can be switched on by clicking on diagram toolbar and select **Category > Freehand** from the pop-up menu.

Drawing free style path with pencil

1. Press on the empty space on diagram pane and drag to form the outline.



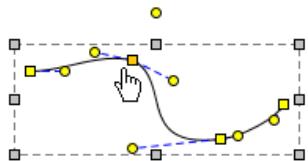
Drawing with pencil

2. Release the mouse and new freehand shape will then be created.

Activating the fine editing selector

Free 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

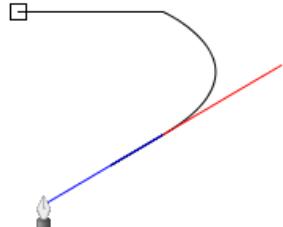
Drawing curve with pen

1. Click on empty space on diagram pane and drag it to create the first stroke.



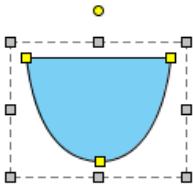
Straight line created

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

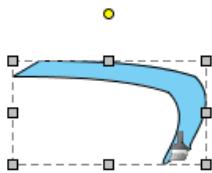
3. To confirm editing of the freehand shape, double click on diagram and a new freehand shape will be created. If the point returns to the starting point, it will form a closed path.



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

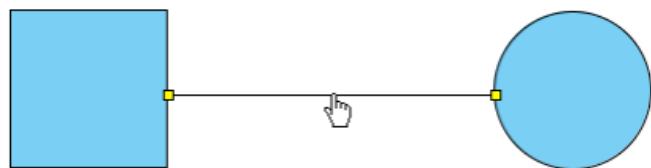
2. Release the mouse and a new connector will be created between them.



A line is created

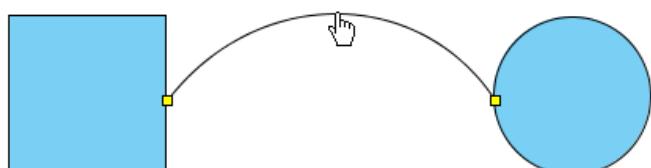
Bend a straight connector into a curve

1. Press on a straight connector.



Clicking on straight line

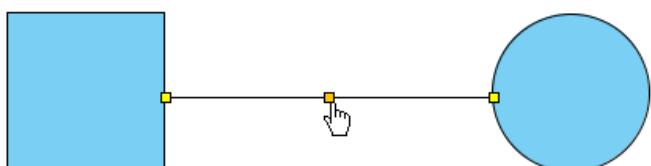
2. Bend it to your preferred direction and it will become a curve connector.



A curve connector

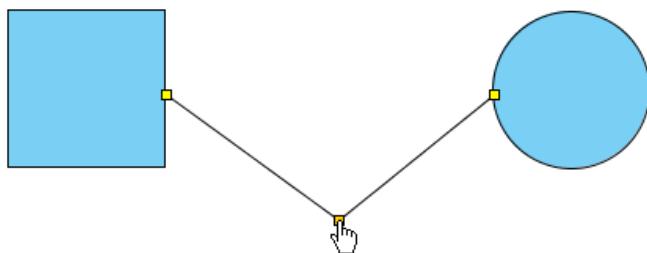
Split a straight connector

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

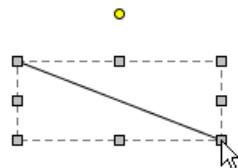
3. Drag on that point to split the line.



Moving mid point

Drawing straight and curved line

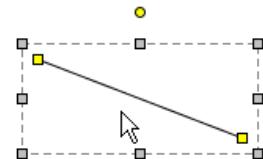
1. Press on the diagram pane and drag to form the outline.
2. Release the mouse button and a straight line will be created.



Line created

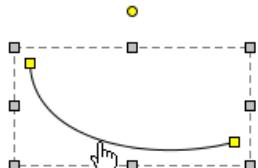
Bend a straight line into a curve

1. Select a straight line for a second to wait for the fine editing selectors popping out.



Showing fine editing selector

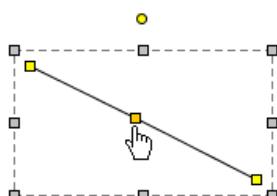
2. Press on the straight line. Drag it to bend into your preferred direction.



Dragging line as curve

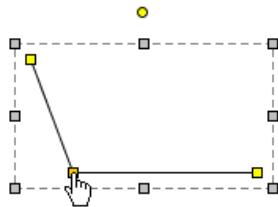
Split straight line

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

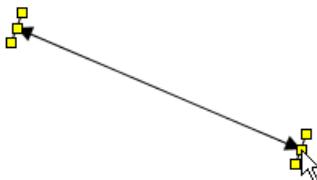
3. Drag on that point to split the line.



Moving mid point

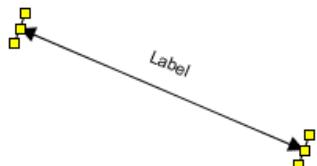
Drawing labelled line

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.

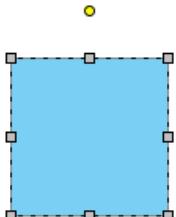


Label Line

5. You may drag the yellow selector to modify the line's outline.

Drawing rectangle

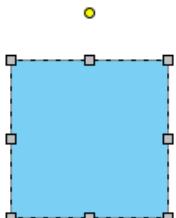
Click on the diagram to create a rectangle.



Freehand shape created

Drawing path rectangle

Click on the diagram to create a path rectangle.



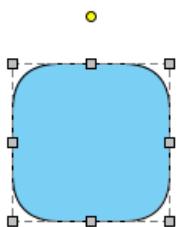
Freehand shape created

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 keep shape as a rectangle.

Drawing rounded rectangle

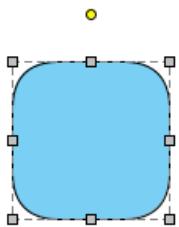
Click on the diagram to create a rounded rectangle.



Freehand shape created

Drawing rounded rectangle 2

Click on the diagram to create a rounded rectangle 2.



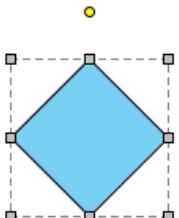
Freehand shape created

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

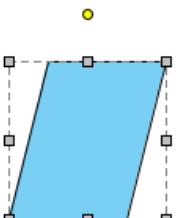
Click on the diagram to create a diamond shape.



Freehand shape created

Drawing parallelogram

Click on the diagram to create a parallelogram shape.

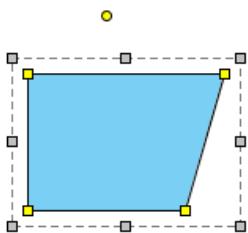


Freehand shape created

Drawing trapezoid

1. Click on the diagram to create a trapezoid shape.

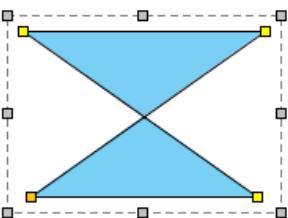
2. You can adjust the slope by dragging the fine editing selectors in yellow.



Other trapezoid outline

Drawing isosceles trapezoid

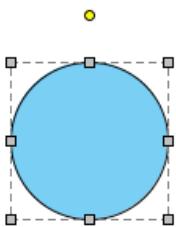
1. Click on the diagram to create an isosceles trapezoid shape.
2. You can reshape the Isosceles Trapezoid by dragging the fine editing selectors in yellow.



Other isosceles trapezoid outline

Drawing ellipse

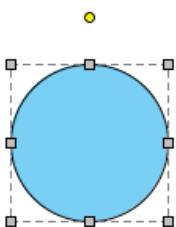
Click on the diagram to create an ellipse shape.



Freehand shape created

Drawing path ellipse

Click on the diagram to create a path ellipse shape.



Freehand shape created

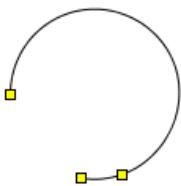
What's the difference between ellipse and path ellipse?

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.

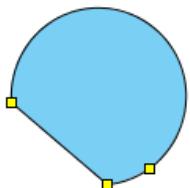
2. You can extend the line by dragging on the fine editing selectors in yellow.



Other arc outline

Drawing chord

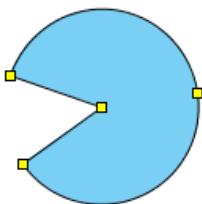
1. Click on the diagram to create a chord shape.
2. You can extend the arc of chord by dragging on the fine editing selectors in yellow.



Other Chord outline

Drawing pie

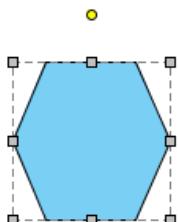
1. Click on the diagram to create a pie shape.
2. You can extend the arc of pie by dragging on the fine editing selectors in yellow.



Other pie outline

Drawing hexagon

Click on the diagram to create a hexagon shape.

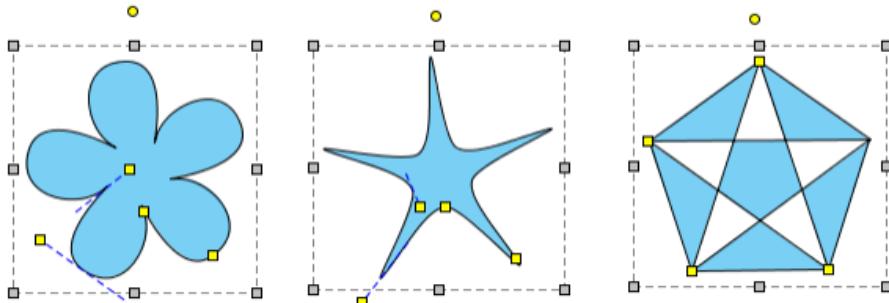


Freehand shape created

Drawing regular polygon

1. Click on the diagram to create a regular polygon shape.

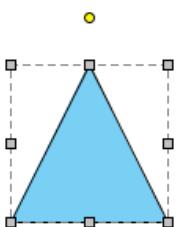
2. You can modify the outline of shape by dragging the fine editor selectors in yellow.



Other regular polygon outline

Drawing isosceles triangle

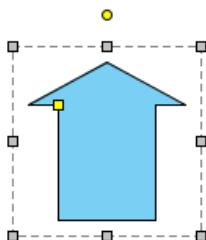
Click on the diagram to create an isosceles triangle shape.



Freehand shape created

Drawing single head arrow

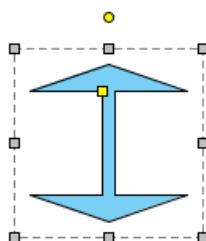
1. Click on the diagram to create an arrow shape.
2. You can reshape it by dragging the fine editing selectors in yellow.



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.

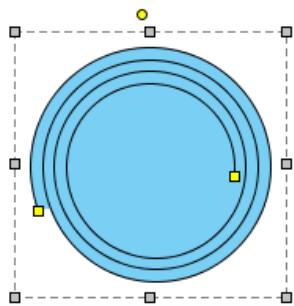


Other two head arrow outline

Drawing spiral

1. Click on the diagram to create a spiral shape.

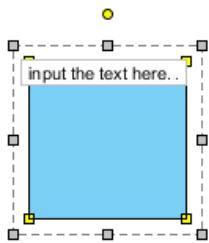
2. You can reshape it by dragging the fine editing selectors in yellow.



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

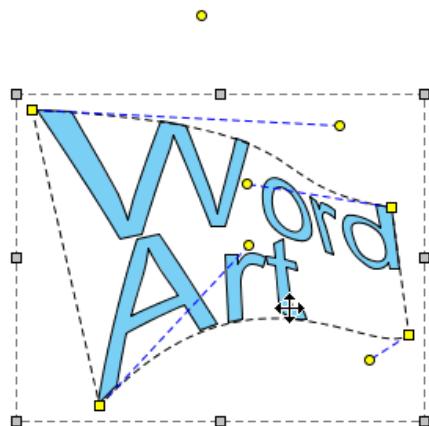
1. Click on the diagram to create a word art shape, and input the text. You can press **Enter** to insert line break.

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

Sweeper and Magnet

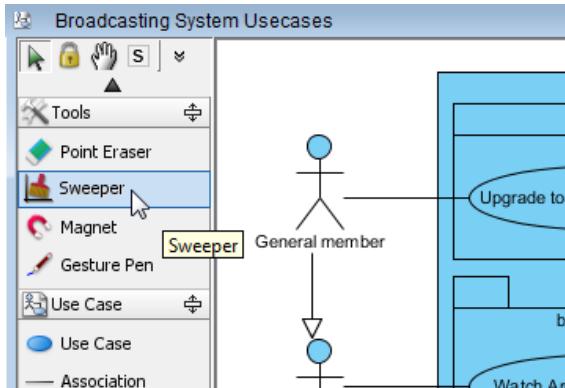
There is much truth in saying that you need to update your diagrams from time to time, here our utilities --- Sweeper and Magnet can help you to modify your diagrams easily. Why should you waste your time on layout? Sweeper can help you to increase more space between the diagram elements while Magnet can help you to diminish the space between the diagram elements. As a result, you can insert or delete diagram elements easily without worrying about the layout.

Sweeper

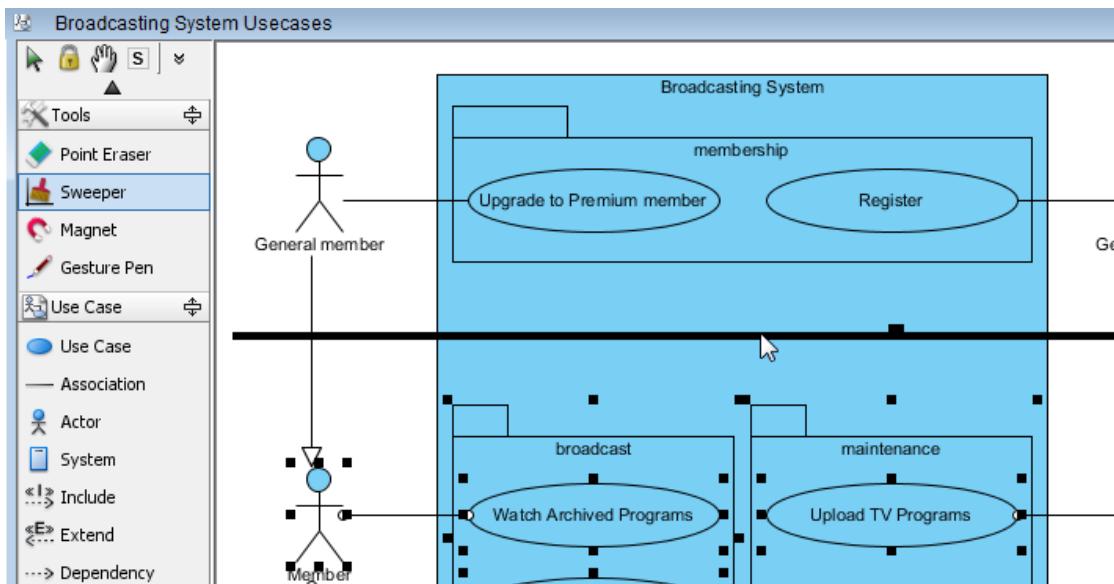
The Sweeper is one of the useful utilities for editing your diagrams. Using Sweeper to extend the space between the diagram elements makes you convenient to insert or move your diagram elements. When you have experienced of moving the diagram elements by yourself, you probably understand how hard it is to manage the space between the diagram elements one by one. Once you use Sweeper, you would never doubt how useful it is.

You can move or insert the diagram elements by following the simple steps below:

1. Click **Sweeper** button.
2. Point the cursor to the space where you would like to insert or move diagram elements.
3. Hold onto your mouse and move the line horizontally or vertically.



Clicking Sweeper button



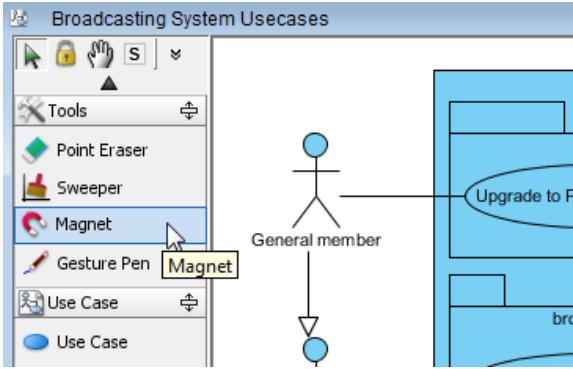
Moving down the diagram element horizontally

Magnet

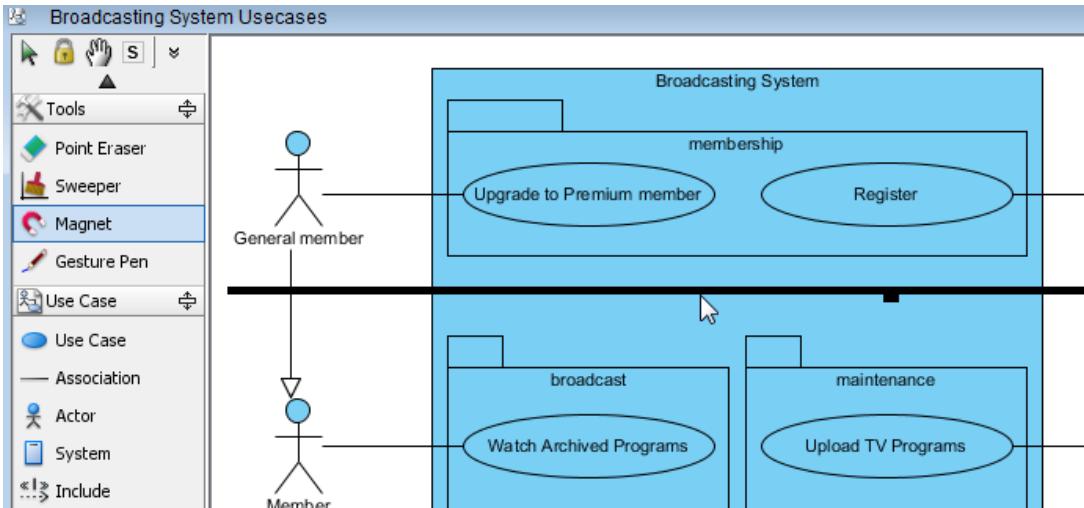
The magnet is another convenient utility for you to move and delete your diagram elements. When there are some diagram elements you would like to delete or move, you should try Magnet. Its function is to diminish the space between the diagram elements and make your diagrams much tidier for printing. To use Magnet, you can do as follows:

1. Click **Magnet** button.
2. Point the cursor to the space where you would like to delete or move diagram elements.

3. Hold onto your mouse and move the line horizontally or vertically.



Clicking Magnet button



Moving up the diagram element horizontally

Mouse gestures

A variety of shapes and model elements can be created by mouse gestures. For your convenience and quick creation, mouse gestures allow you to execute common commands and create UML models within all diagrams.

Drawing shapes

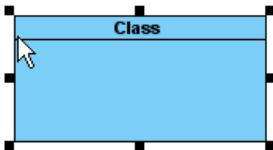
1. To start using a mouse gesture, press the right mouse button and drag it until finish drawing a shape.



Drawing clockwise rectangle

2. When the shape is done, release the mouse. After the shape is created, the action Description will be shown on top right corner of the diagram.

Create Class



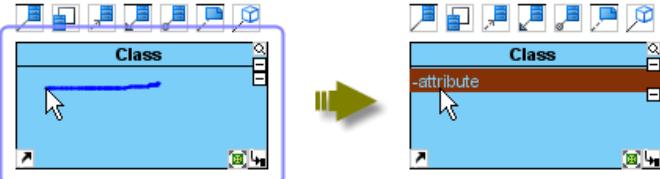
Class Created

Creating Class member

You can learn how to create attribute and operation within the class in the following sub-sections.

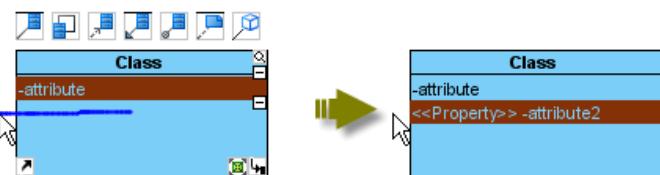
Creating an attribute

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

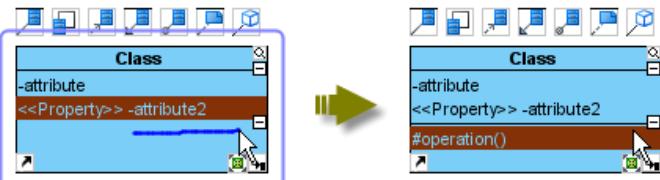
2. If you draw the line until outside the class, an attribute with <<Property>> stereotype will be created.



<<Property>> is created

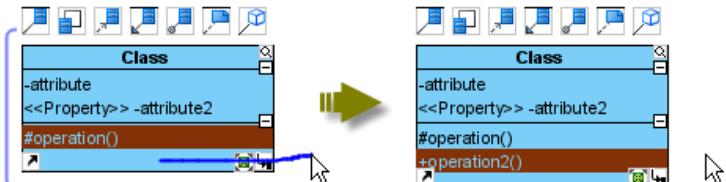
Creating an operation

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

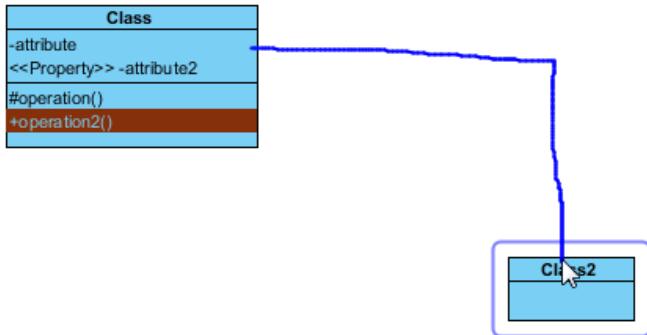
2. If draw the line until outside the class, a public operation will be created.



Public operation is created

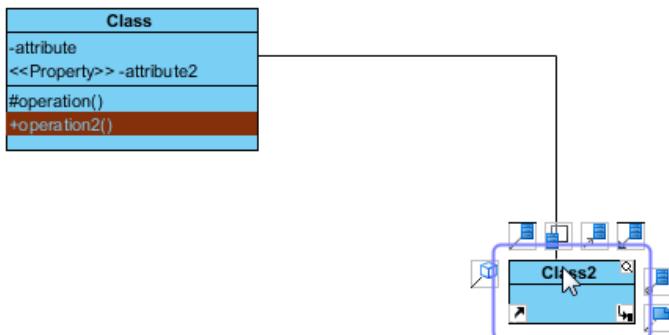
Connecting shapes

1. Draw a line from one shape to another. As a result, two shapes will be connected.



Drawing from a shape to another

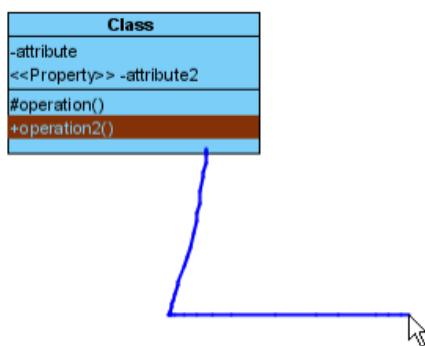
2. After mouse released, an association is created between the classes.



Association created

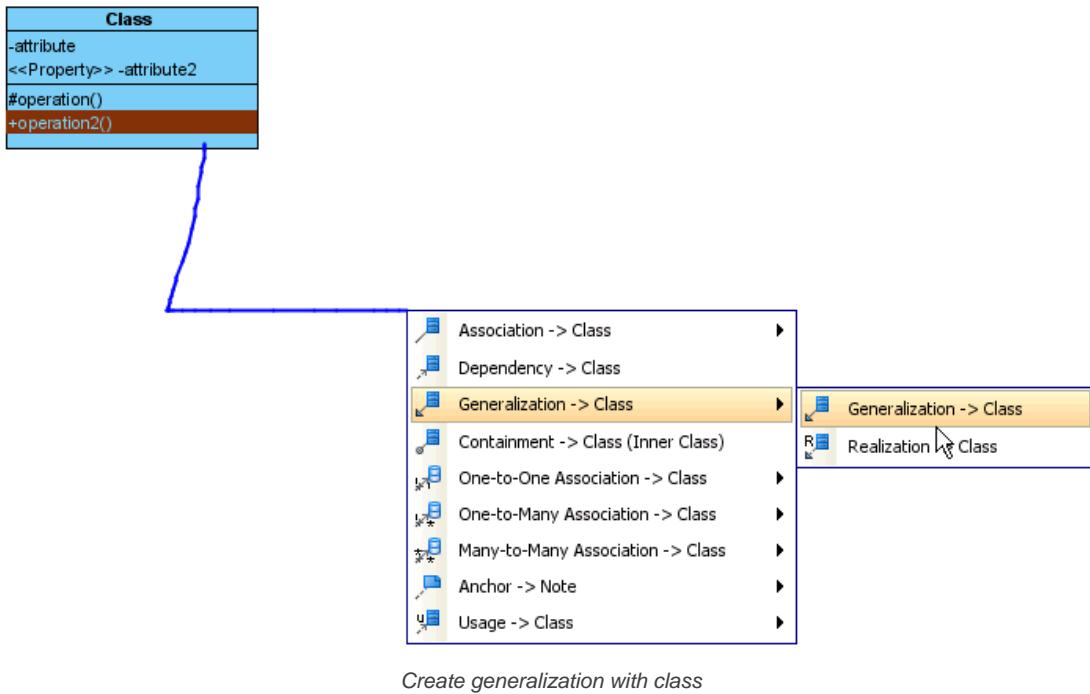
Creating a new shape

1. A new shape can also be created. To do so, draw a line from a shape to the place you preferred.

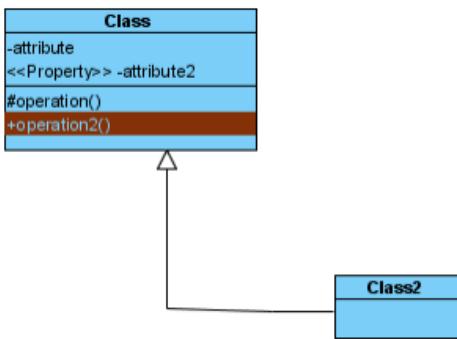


Drawing to empty area

2. After the mouse is released, a pop-up menu will be shown. You can select the type of connector and shape you would like to create from it.



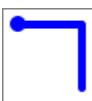
3. The two shaped with connector is created.



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/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 activity diagram

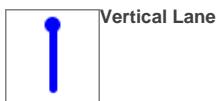
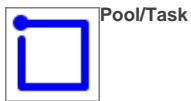
Activity diagram (UML 1.x)

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. Else if there is no Final State, a Final State will be created.)

The description of mouse gestures for activity diagram

Business process diagram

Icon	Description



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

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

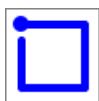
Data flow diagram

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

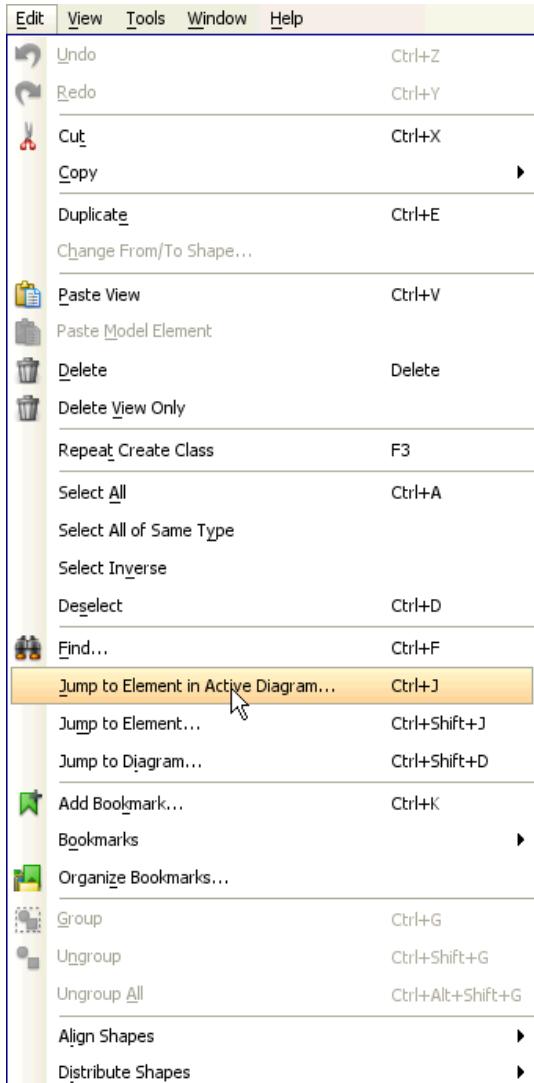
Jumping to shape

For searching a shape/shape 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

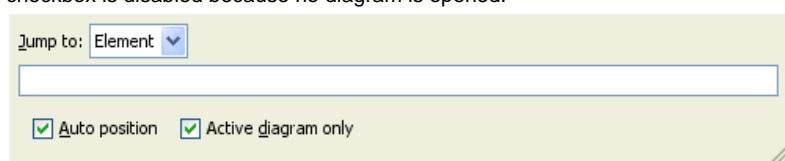
If there is an active diagram opened, you can jump to the model element in the active diagram.

1. You can select **Edit > Jump to Element in Active Diagram...** from the main menu or press **Ctrl+J** to unfold **Jump to** dialog box.



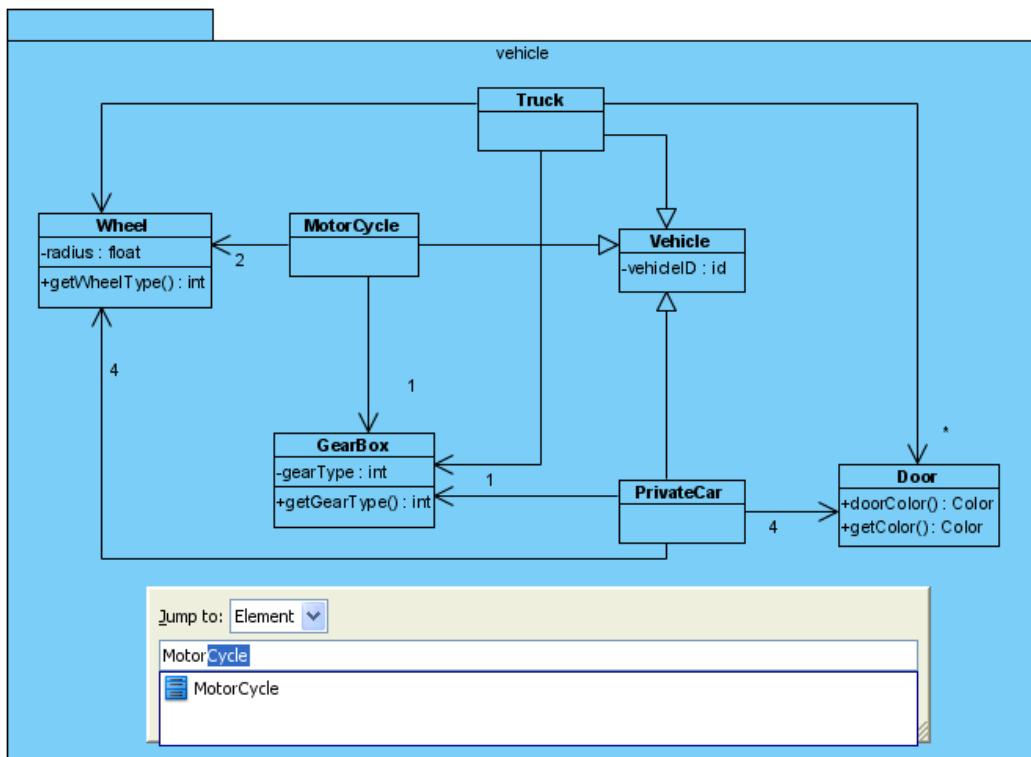
Select **Jump to Element in Active Diagram** from the main menu

2. Apart from jumping to element in active diagram, you can also jump to any element within the project. You may select **Edit > Jump to Element...** from the main menu 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.



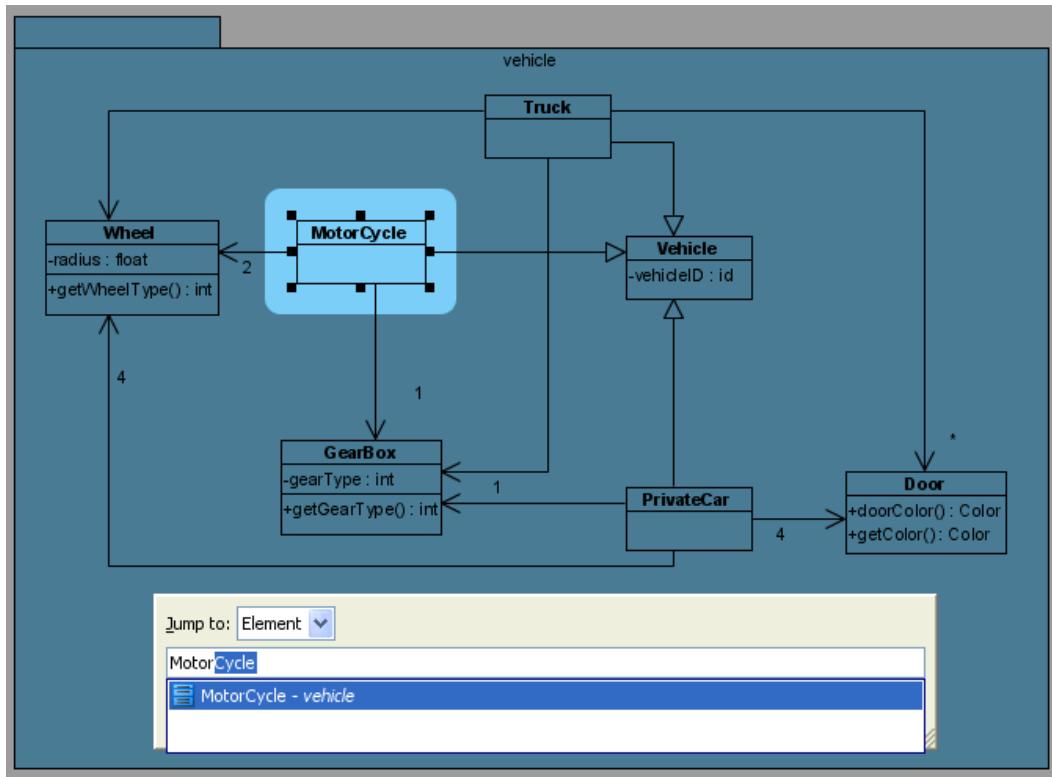
Jump to dialog box is shown

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 .



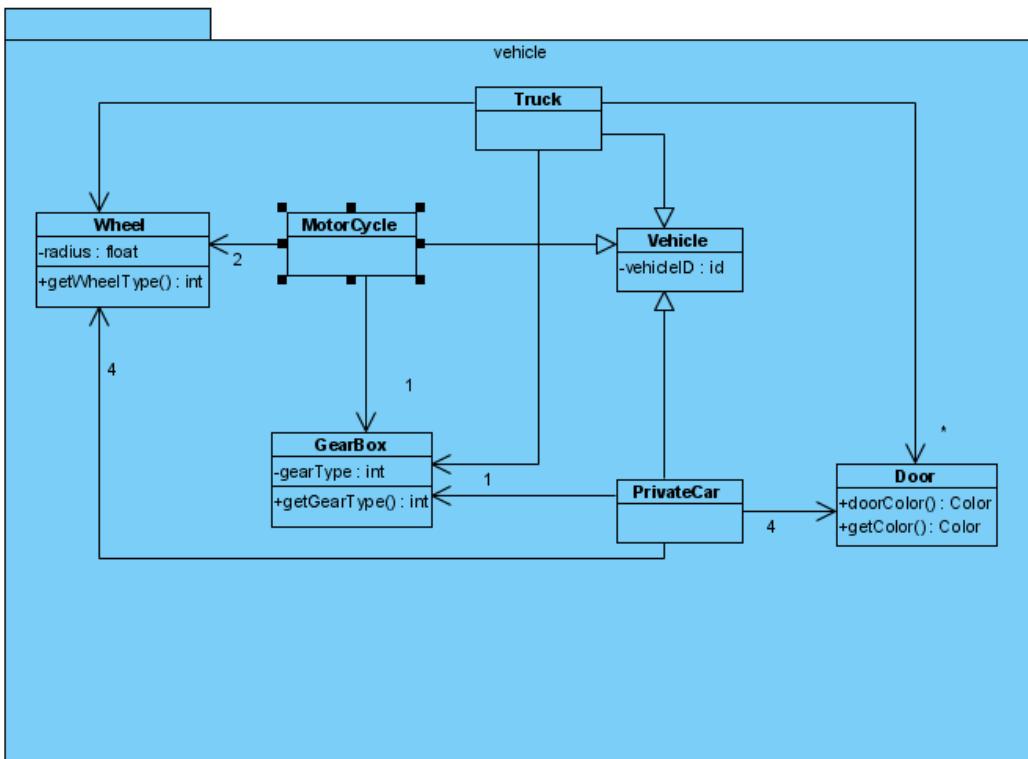
A list of possible shapes are shown

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

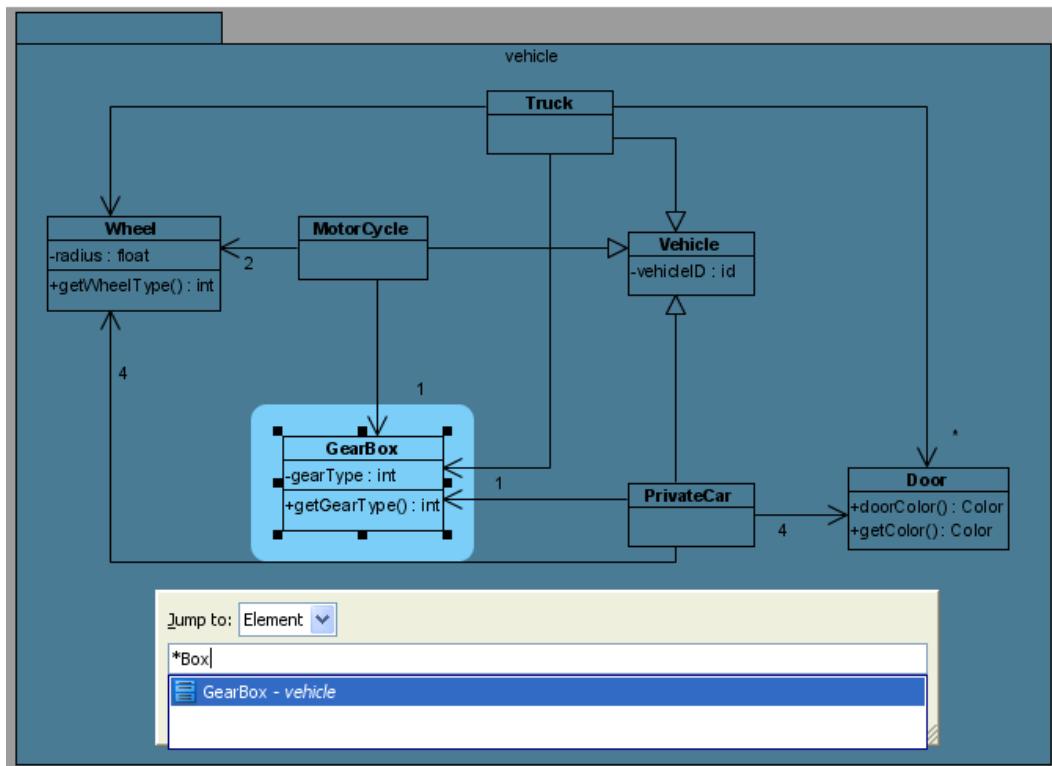
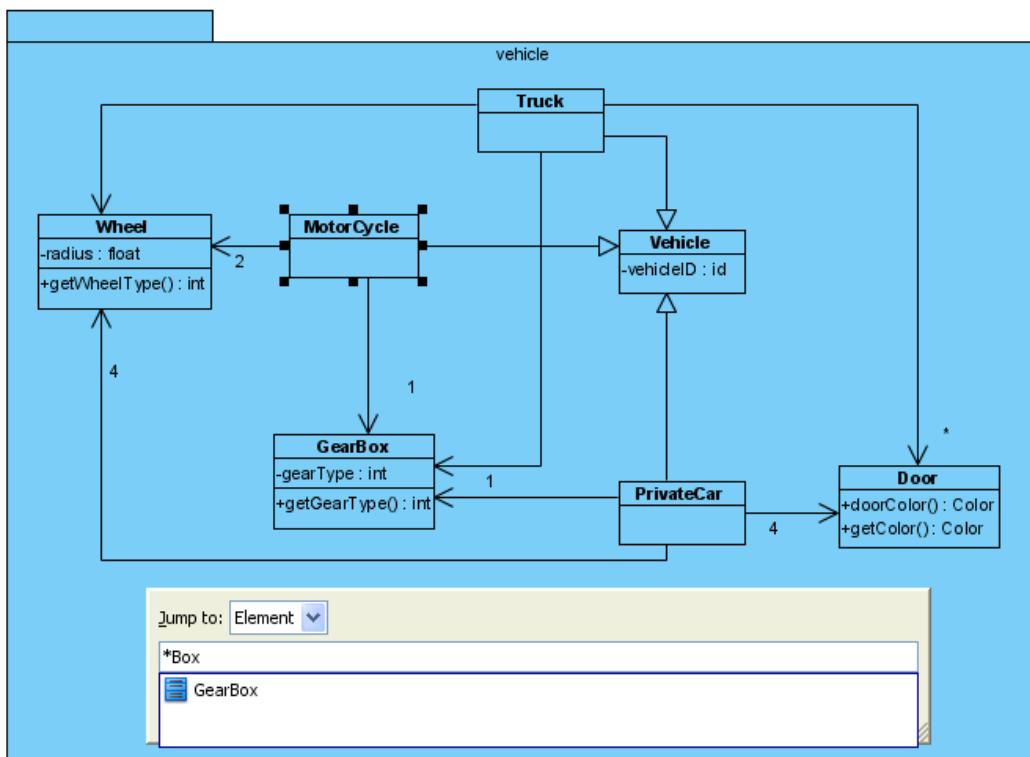
6. Press **Enter** to confirm jump to the model element. Finally, the **Jump to** dialog box will then be hidden and the model element will be selected on diagram.



shape is selected after confirm jump to

Filtering with wild card character

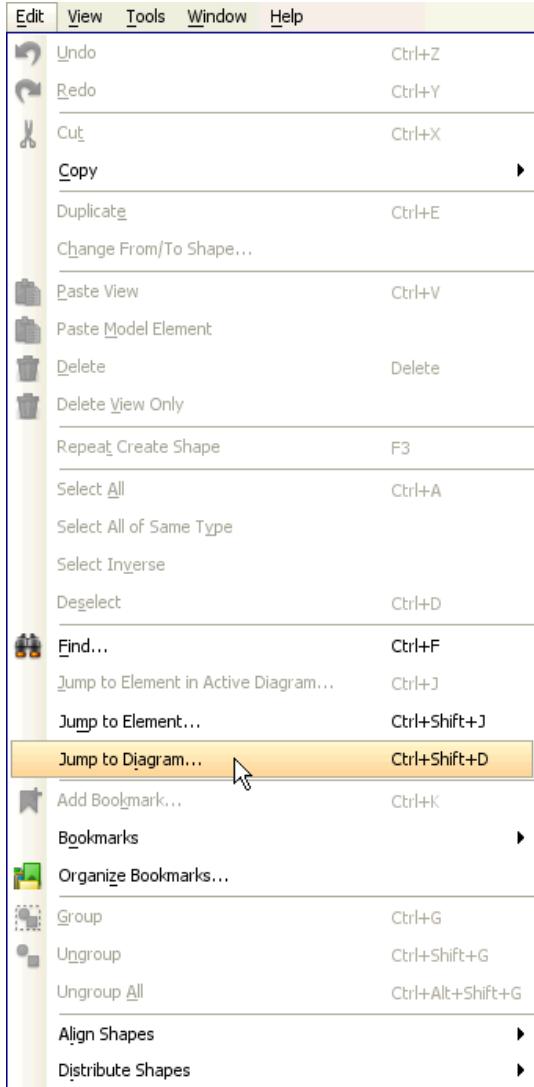
For quick search, you can type a word with * in the text field. The asterisk can substitute a character or a word when you don't remember the exact spelling. As a result, all names of shape that similar to the word you typed will be shown.



Entering name with *

[Jumping to diagram](#)

1. To do so, select **Edit > Jump to Diagram...** from the main menu or press **Ctrl+Shift+D** to show **Jump to** dialog box.



Select **Jump to Diagram...** from the main menu

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.



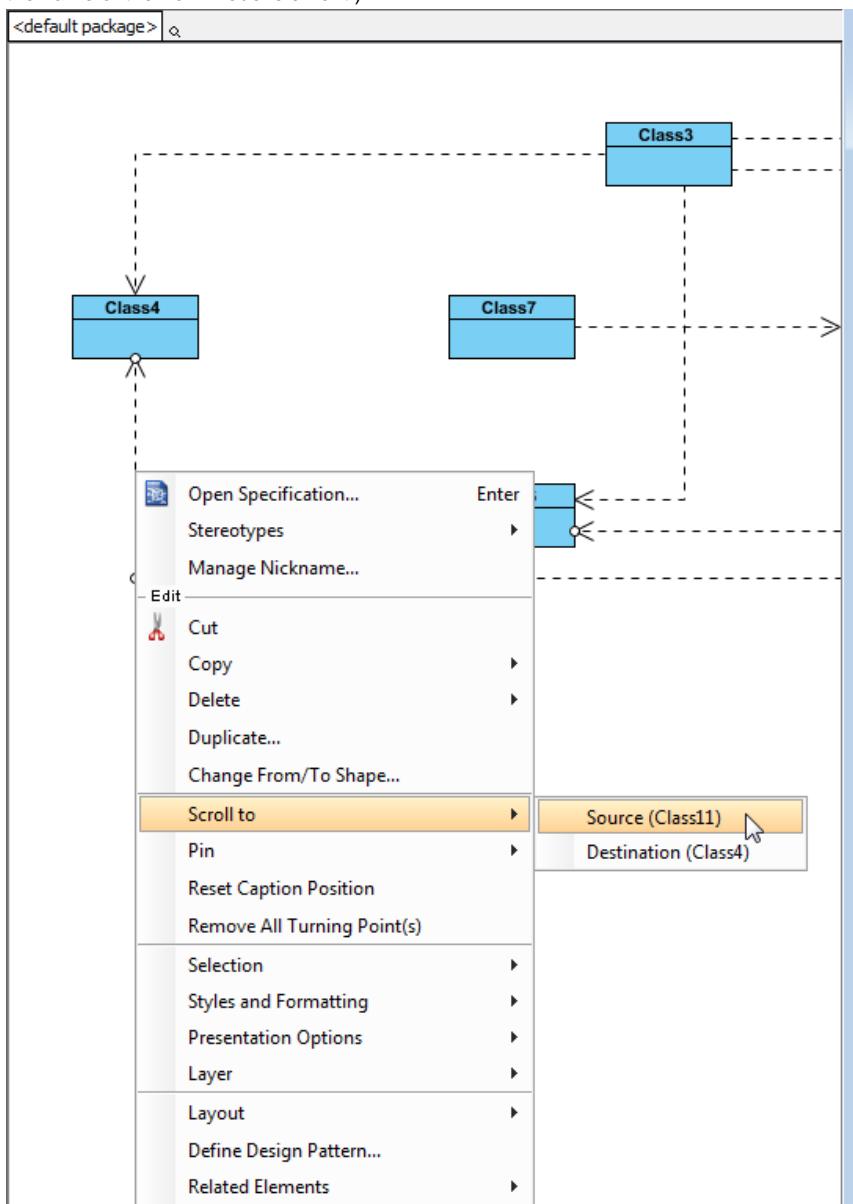
Select the diagram on list

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. VP-UML 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 .

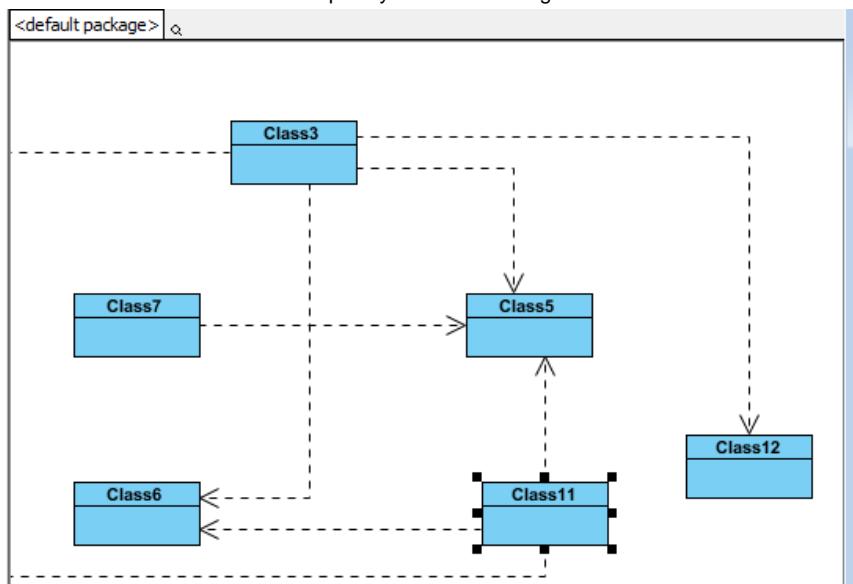
To achieve this:

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.



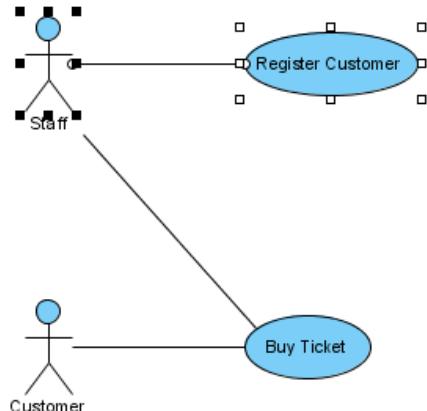
From model element is selected

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.

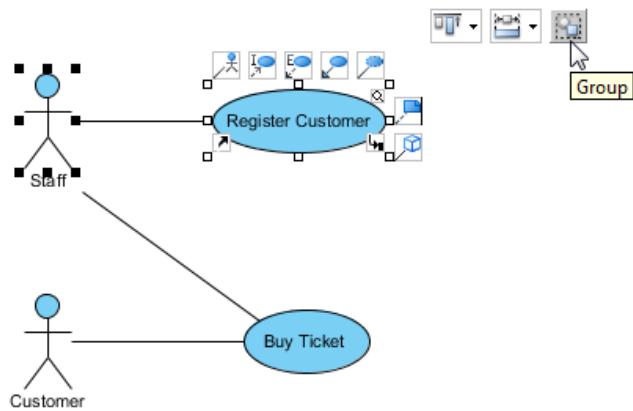
Grouping diagram elements

1. Select the shapes you want them to be grouped together.



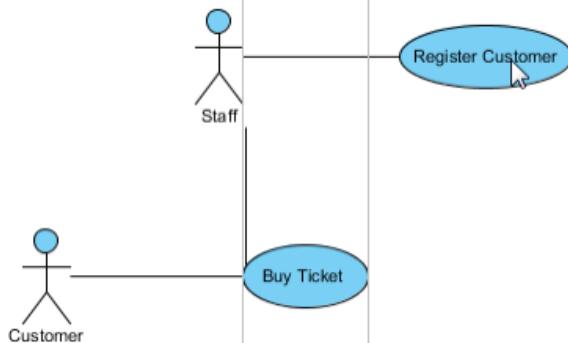
Select several shapes

2. Click resource icon **Group** to group the selected shapes.



Group through resource icon **Group**

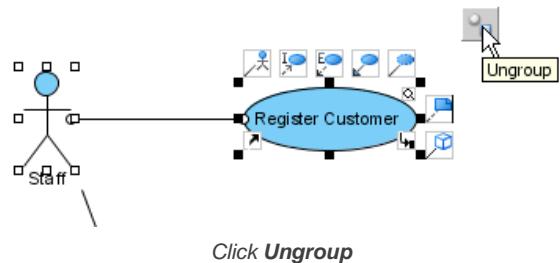
3. When you move a shape, other shapes within the same group will also be moved.



All shapes in group are moved together

Ungrouping diagram elements

To ungroup the shapes, click resource icon **Ungroup**.



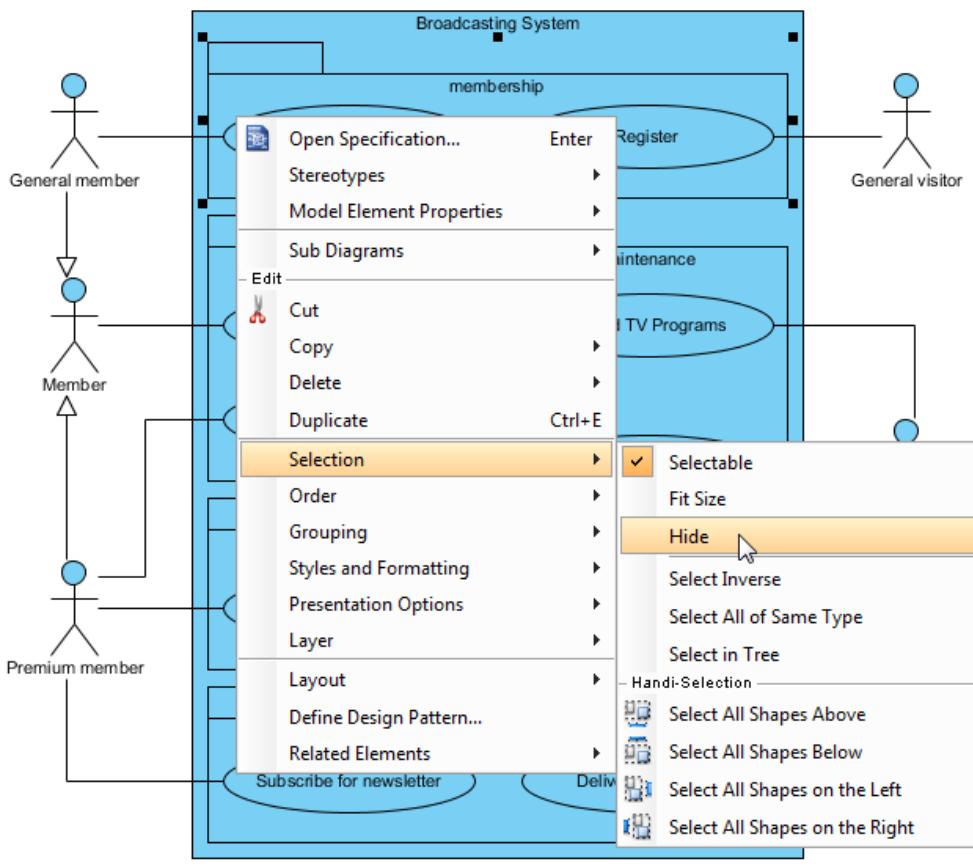
Click Ungroup

Show/hide diagram elements

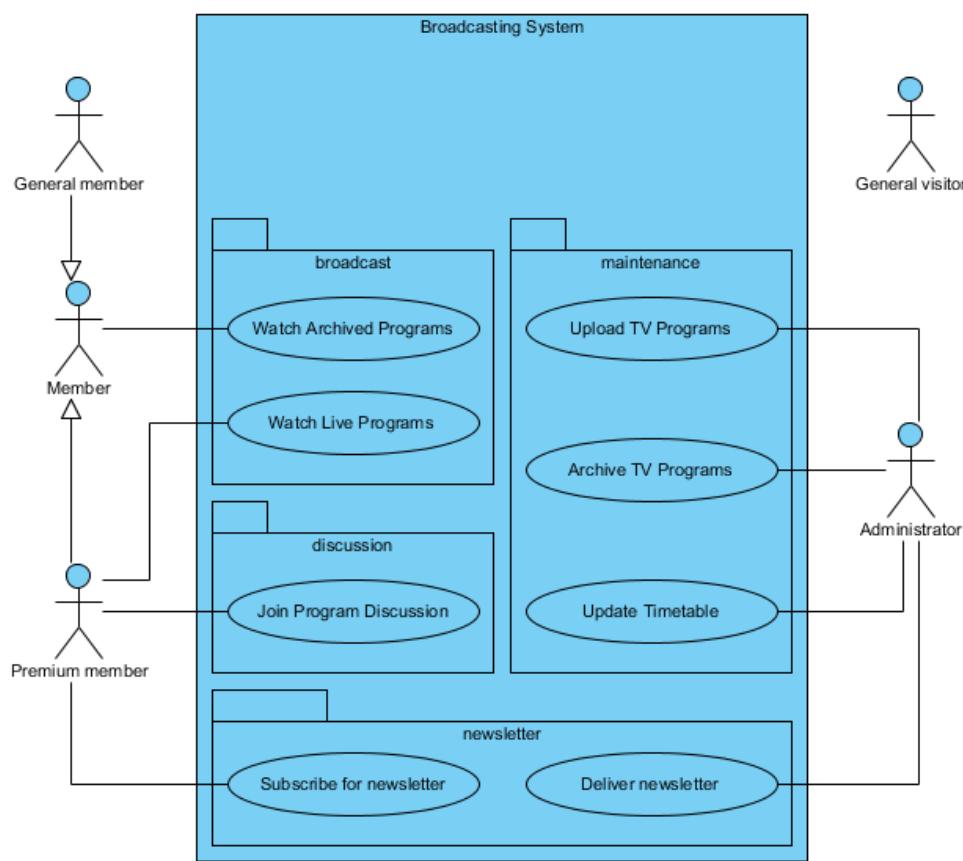
On a compact diagram, to hide or show some of the shapes is necessary. You can choose to hide away diagram elements on a diagram temporarily. For example, hiding away annotation shapes which record internal communications before printing out a diagram for customers' review. In VP-UML, hiding of diagram elements can be done per shape, per shape type or by stereotype.

Hide selected diagram element(s)

Right-click on the selected shape(s) and select **Selection > Hide** from the pop-up menu.



Apart from the selected shape, its related shape(s) (e.g. children and relationship) will also be hidden.



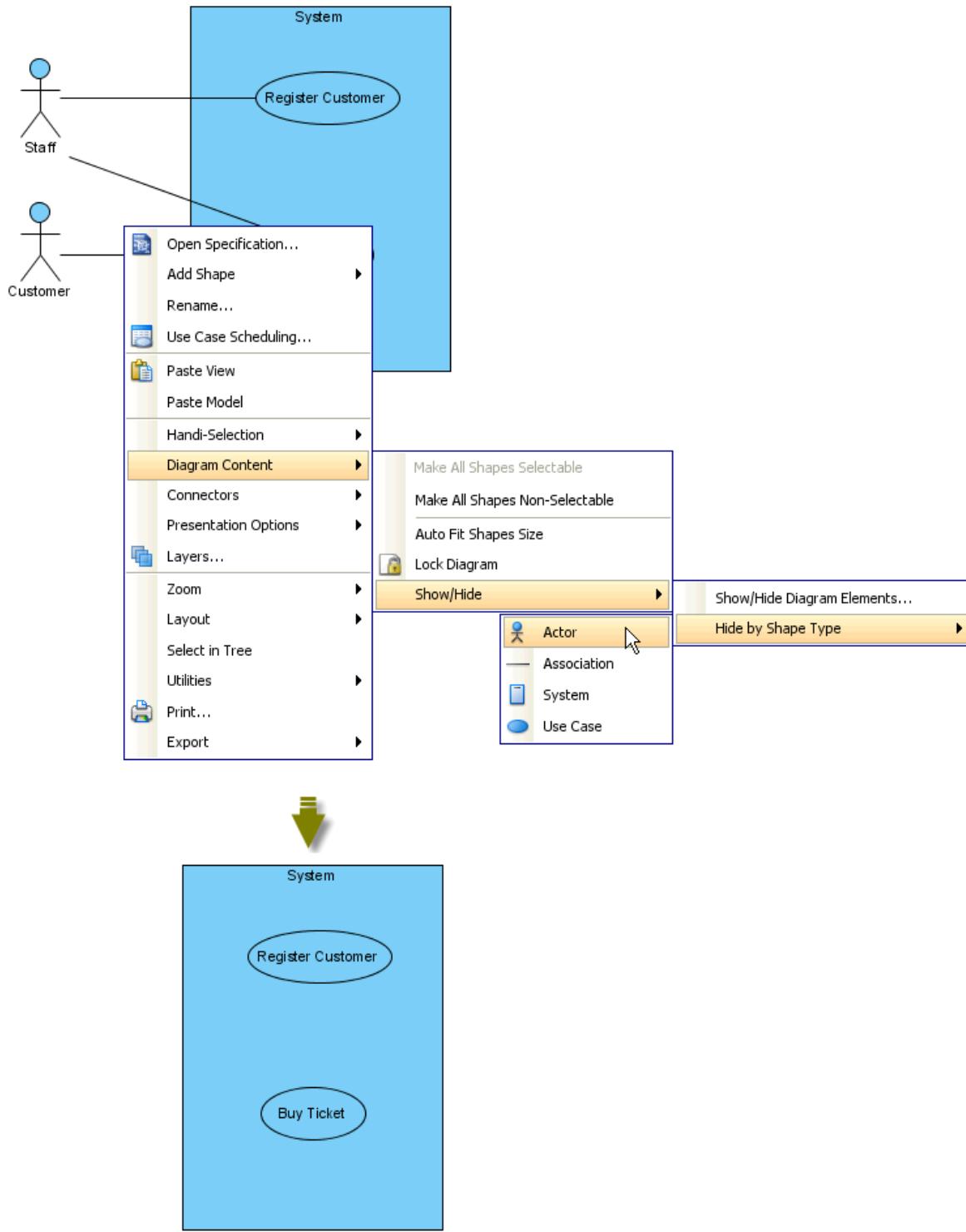
The selected shape and its related shapes are hidden

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.

Hide diagram elements by shape type

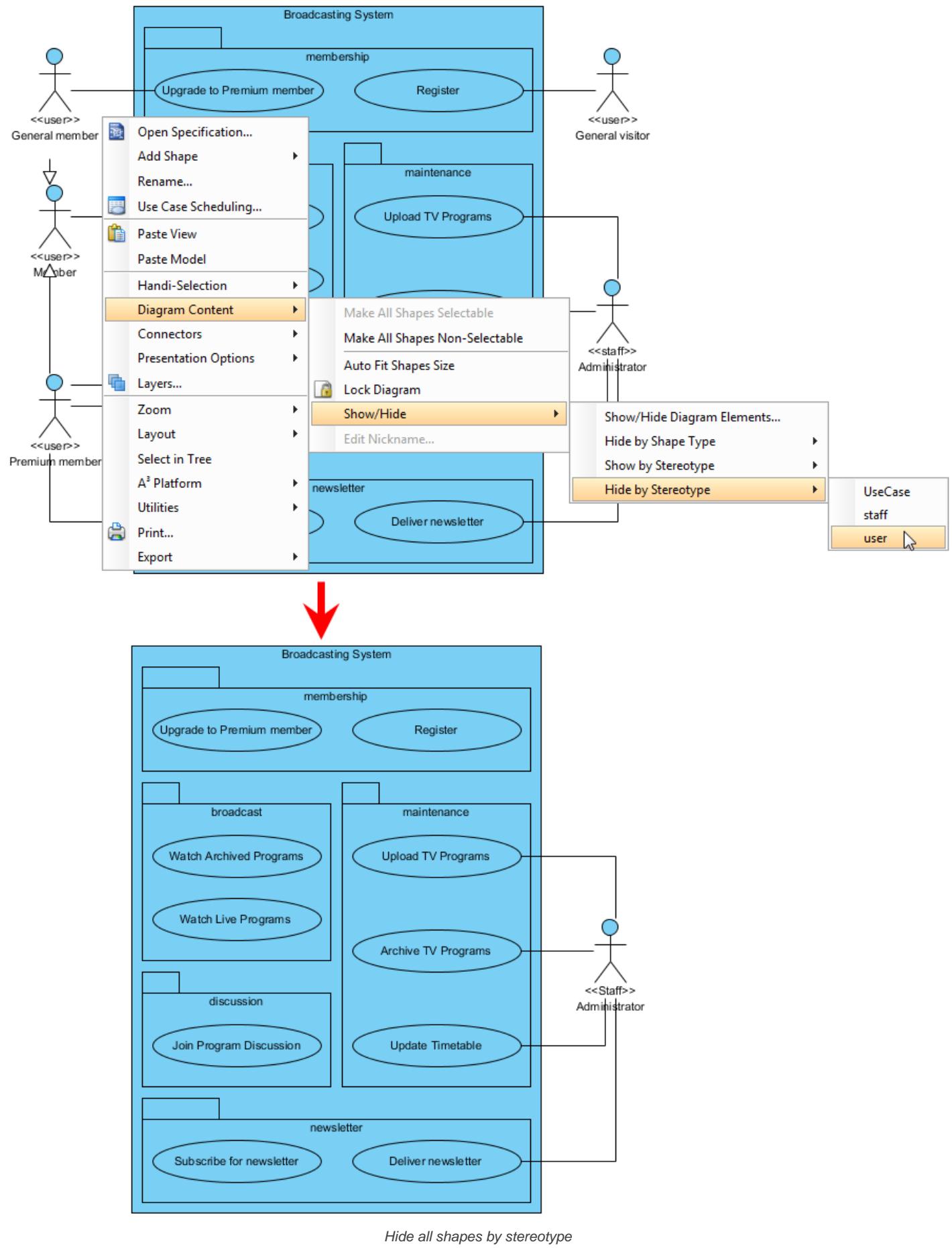
All the shapes with same type from the diagram can also be hidden. Right click on the diagram's background and select **Diagram Content > Show/Hide > Hide by Shape Type**, and then select the shape you want to be hidden from the pop-up menu. As a result, all shapes with the same type you selected will be hidden.



Hide all shapes by shape type

Hide 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.



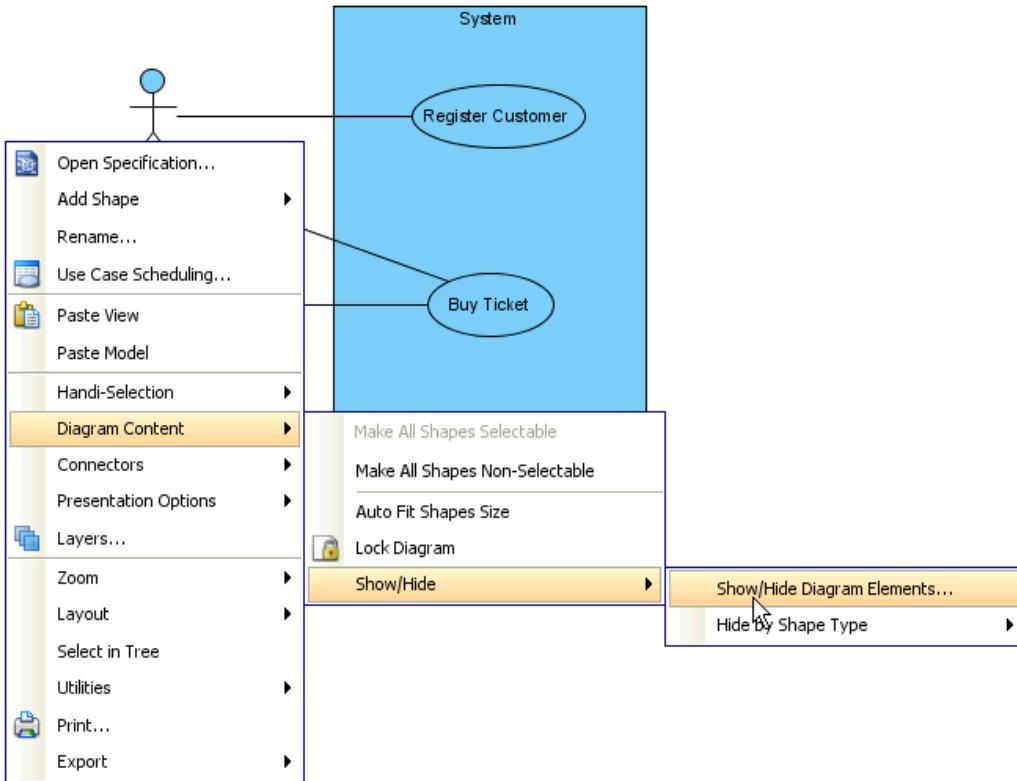
Show all hidden diagram elements

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.

Manage show/Hide of specific diagram element(s)

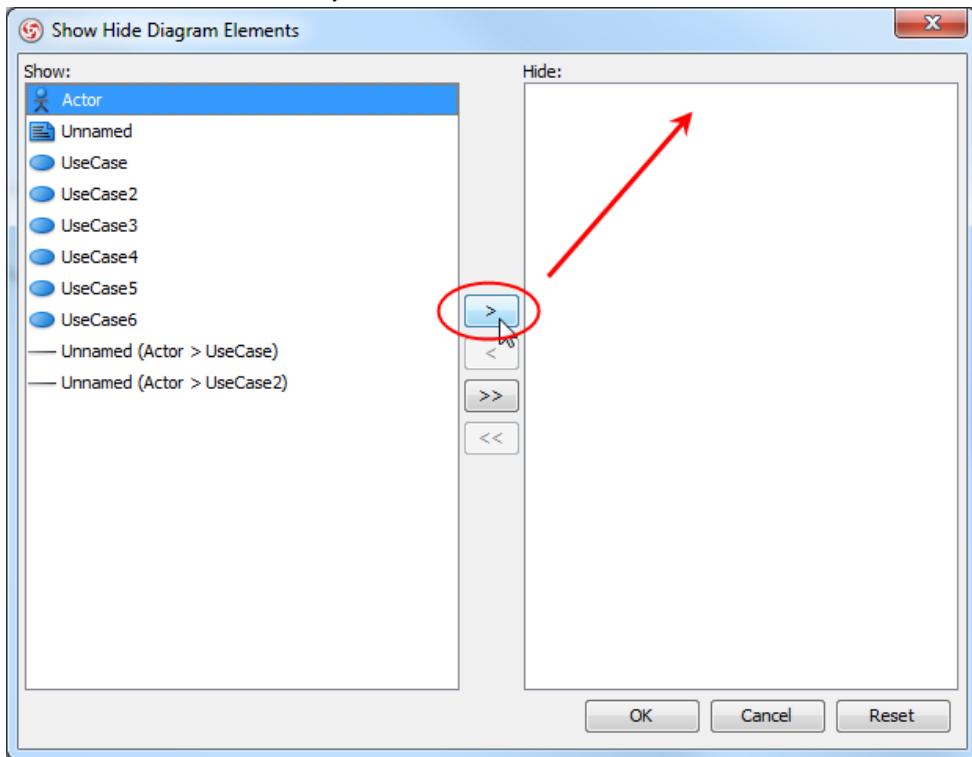
A better management of showing and hiding the shapes can be done in **Show Hide Diagram Elements** dialog box.

1. Right click on the diagram's background, select **Diagram Content > Show/Hide > Show/Hide Diagram Elements...** from the pop-up menu.



Select **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 versa. Finally, click **OK** to confirm.



Select a diagram element to be moved to **Hide** list

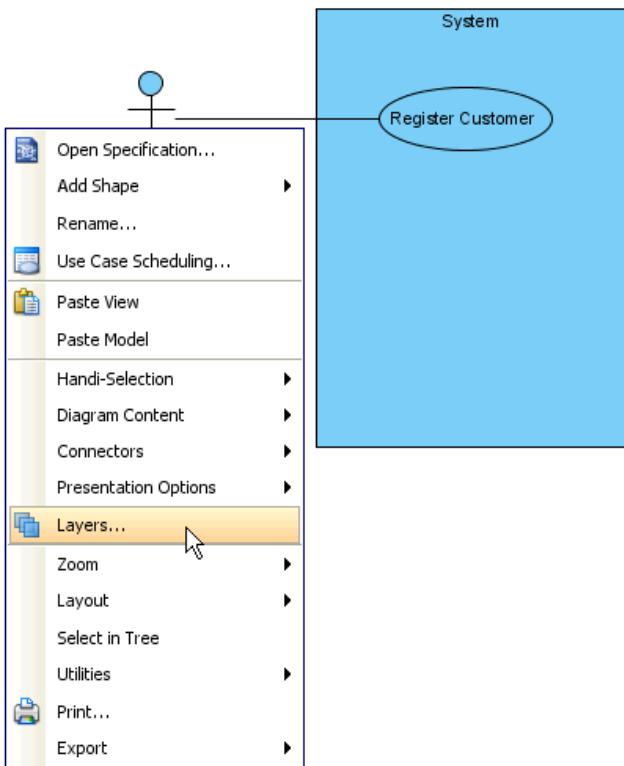
NOTE: If a shape is selected from **Show** list, the related shape(s) of the selected shape will be also removed to **Hide** list

Layer

Occasionally, you may experience deal with a number of shapes with difficulties. VP-UML 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.

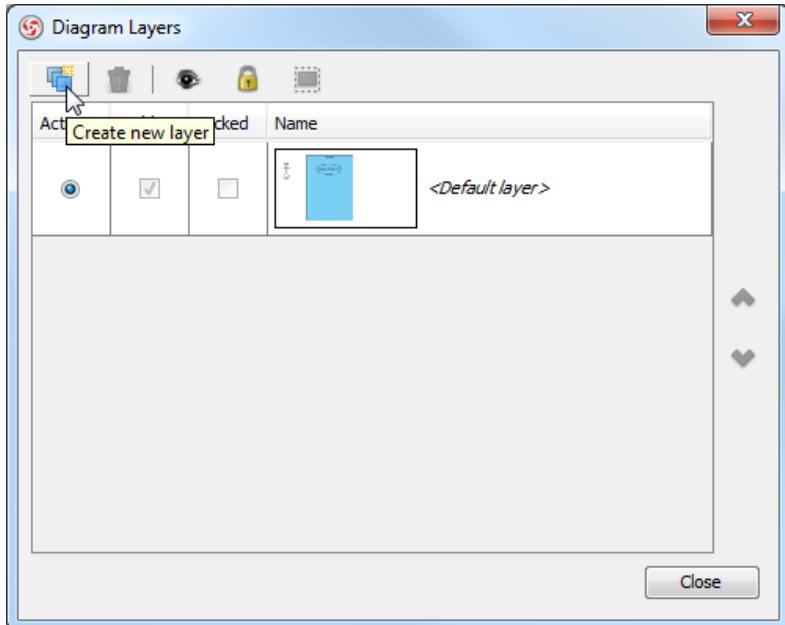
Creating a layer

1. Right click on the diagram background and select **Layers...** from the pop-up menu.



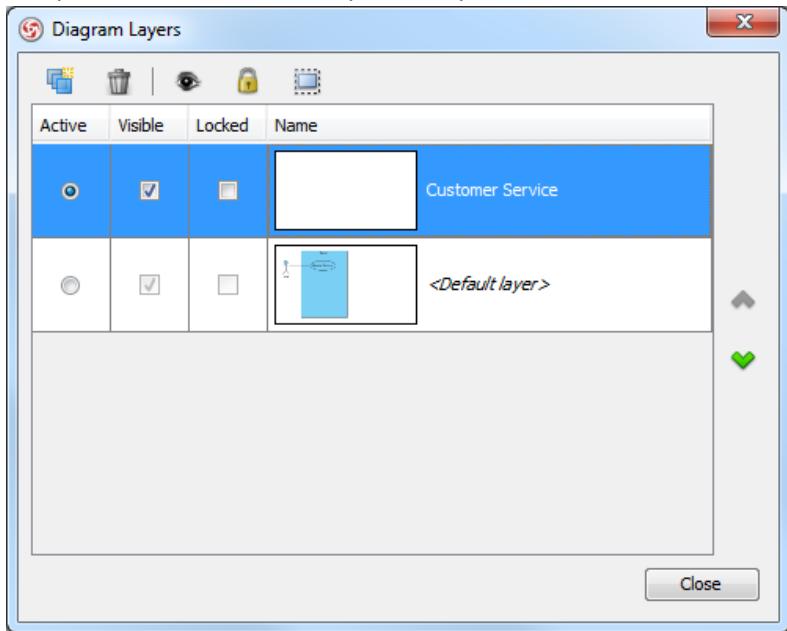
Select **Layers...** from the pop-up menu

2. In **Diagram Layers** dialog box, click **Create new layer** button to create a new layer.



Create a new layer in **Diagram Layers** dialog box

3. Finally, define the name for the newly created layer.

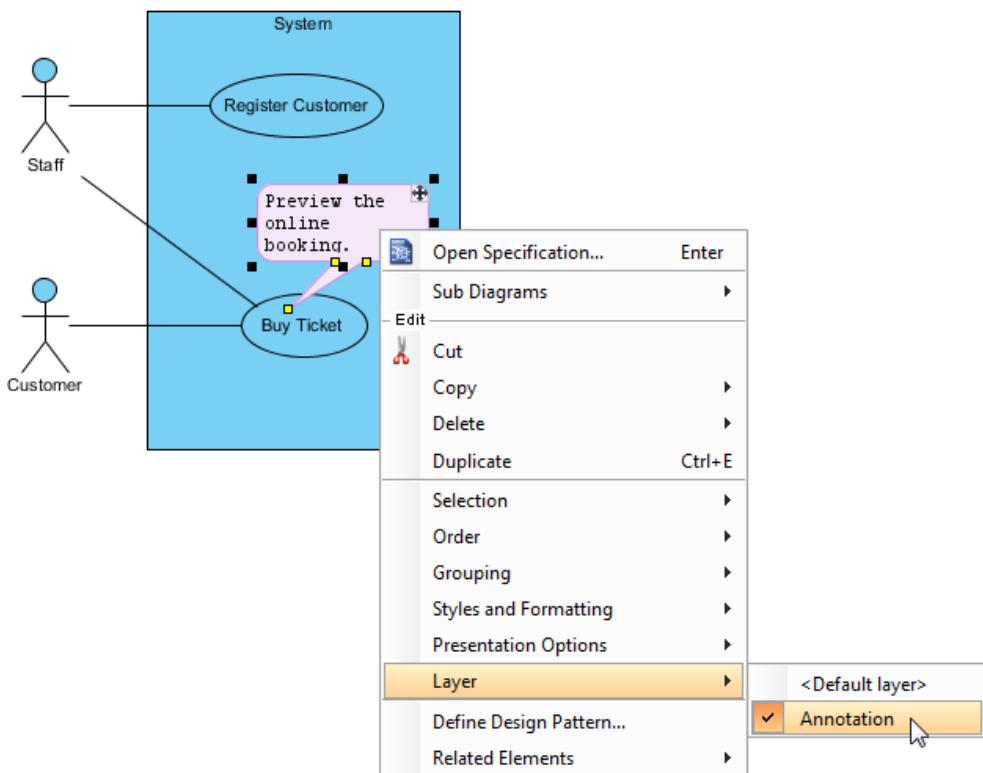


New layer will be an active layer

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 just like the steps of previous section. Right click on a shape and select **Layers**, and then select the new layer you have created.

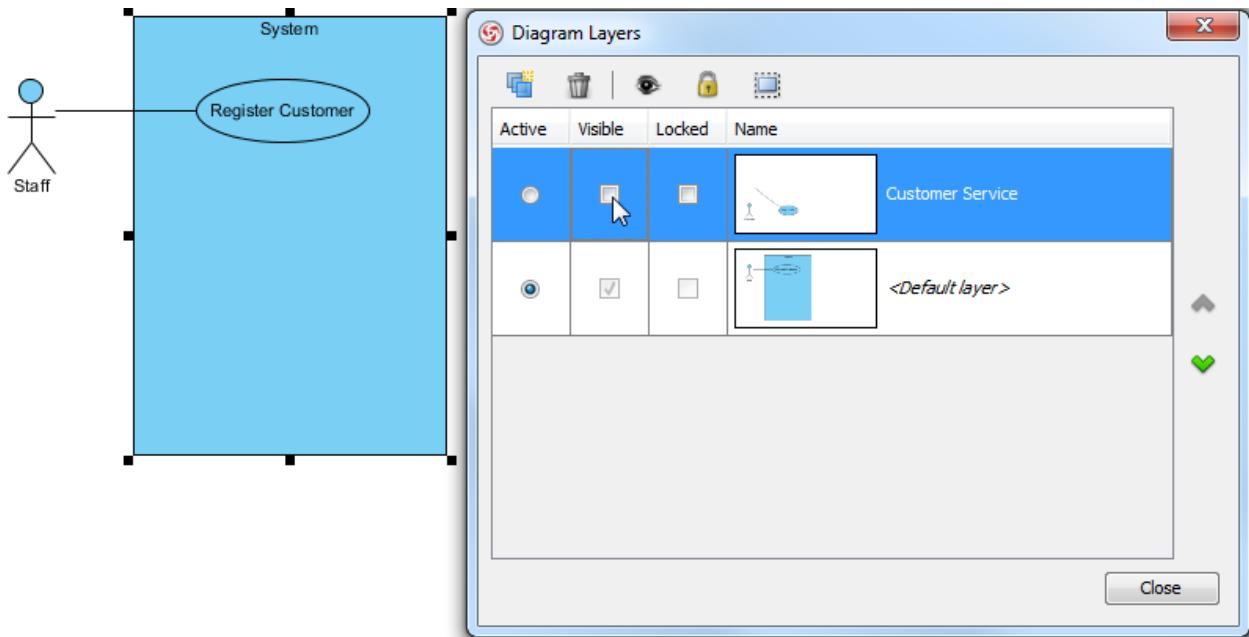


Send the shape to a new layer

As a result, the shape you selected will be sent to the new layer.

Hiding shapes on a layer

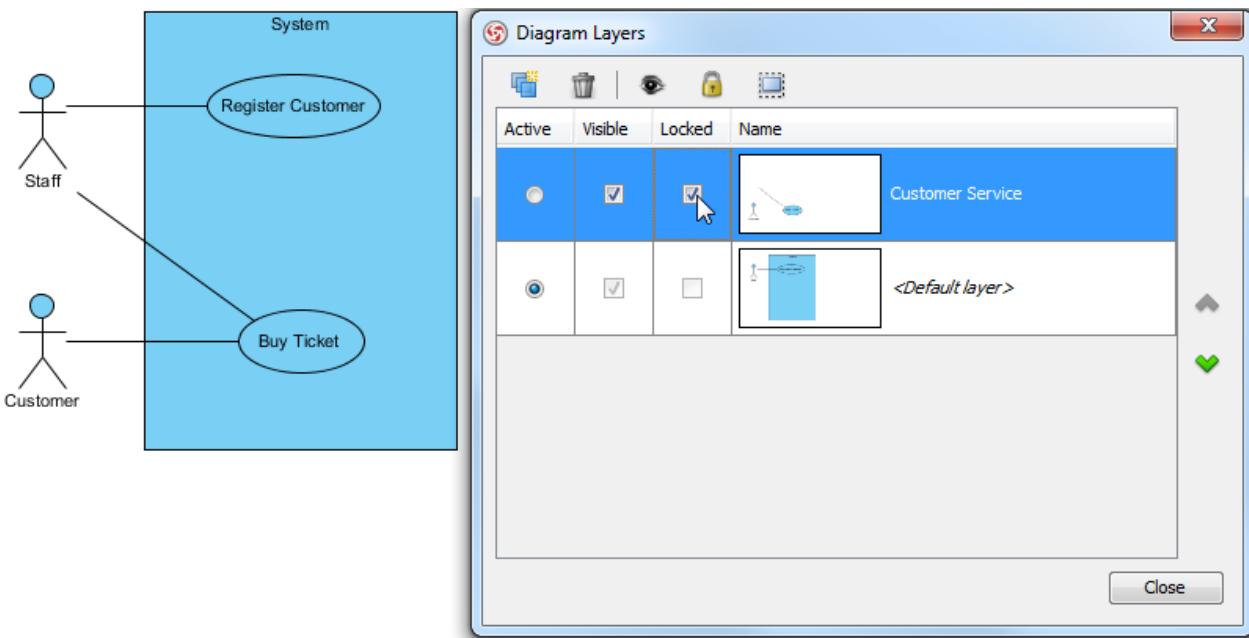
In **Diagram Layers** dialog box, you can make a layer invisible on diagram. To do so, uncheck **Visible** of the layer.



Shapes on the layers are invisible on diagram

Locking shapes on a layer

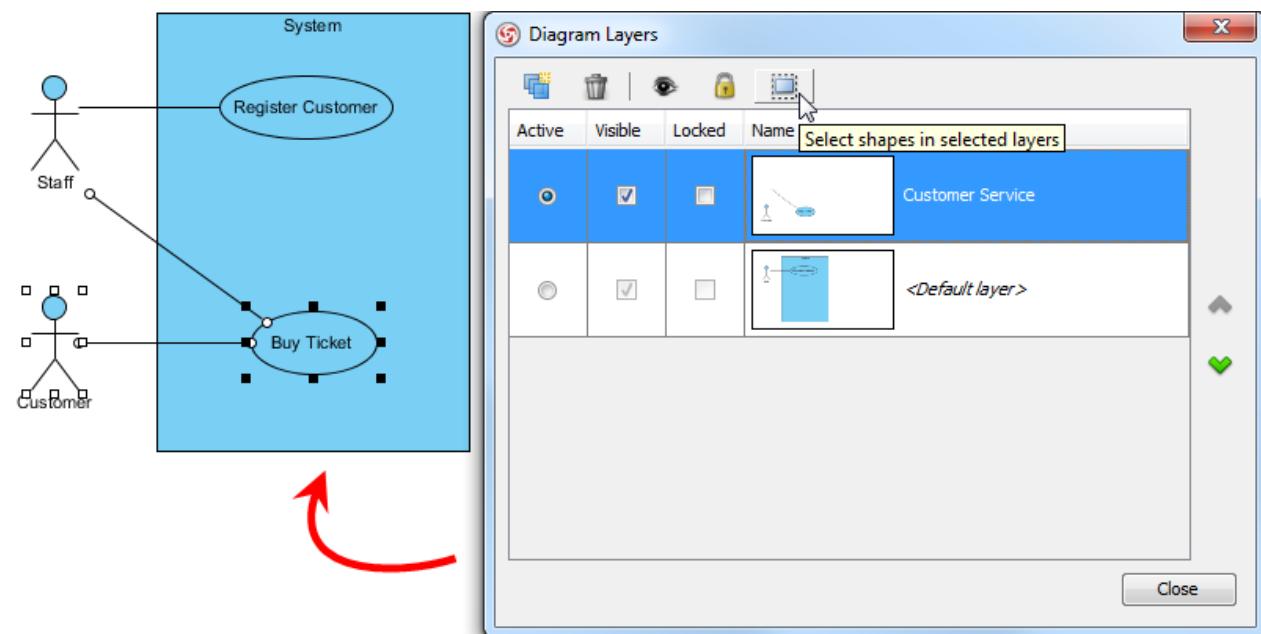
In **Diagram Layers** dialog box, check **Locked** to make all shapes in that layer immovable and unable for editing while uncheck **Locked** to make them movable and editable.



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 on **Diagram Layers** dialog box.



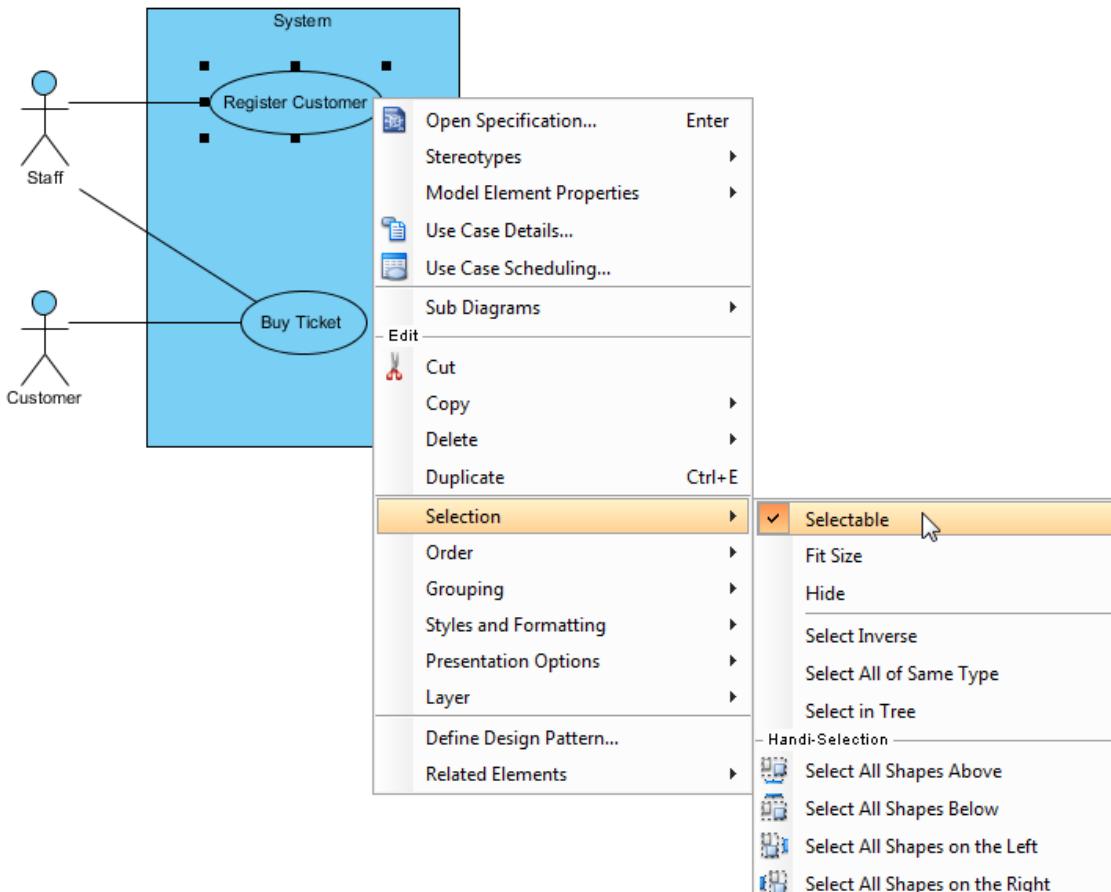
Shapes of the layer are selected on diagram

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.

Change the shape to non-selectable

1. Right click on the selected shapes, uncheck **Selection > Selectable** from the pop-up menu.

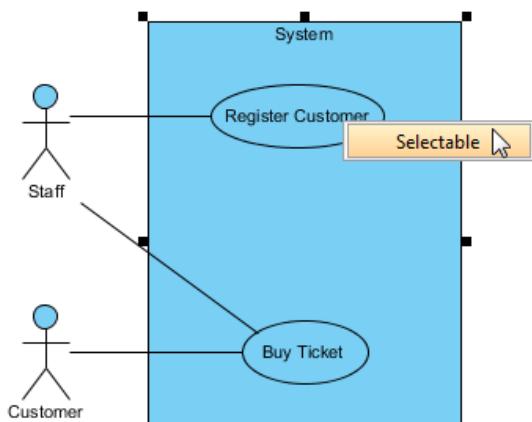


Selected shape to be non-selectable

2. After the shape is non-selectable, click the shape cannot make it to be selected. Neither will the shape be selected by mouse dragging on diagram.
3. Therefore, the non-selectable shape(s) will not be moved when other shapes are moved.

Change the shape to selectable again

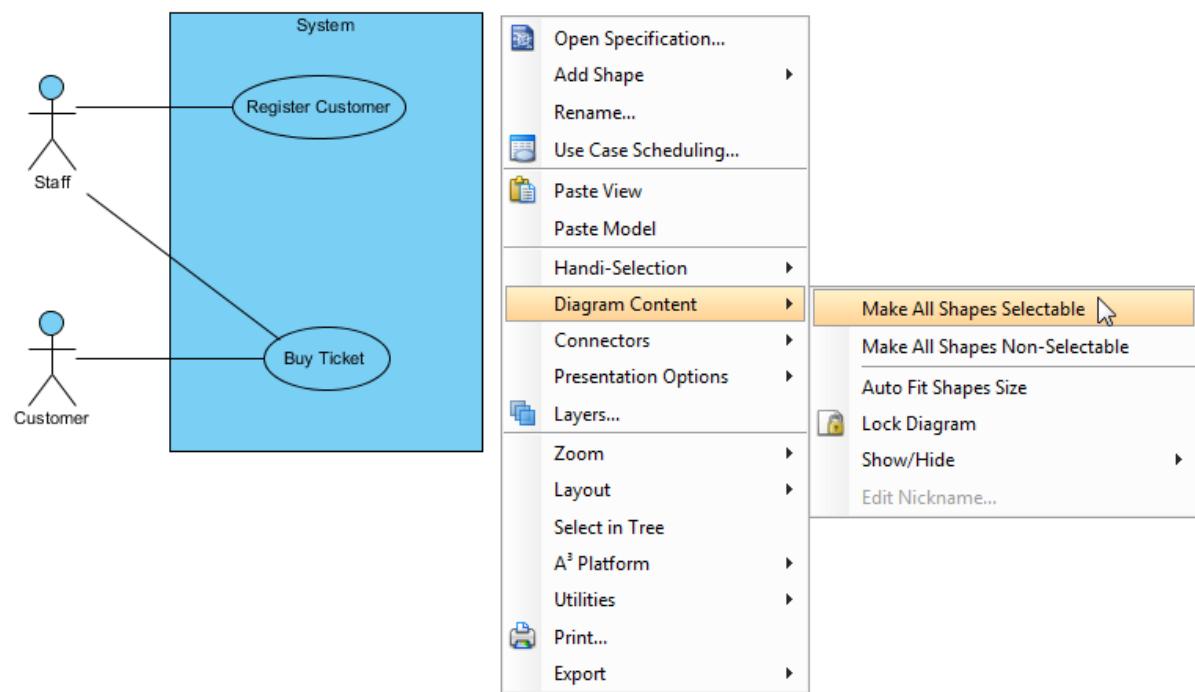
To make the shape selectable again, right click on the non-selectable shape and select **Selectable** from the pop-up menu.



Select Selectable

Set 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.



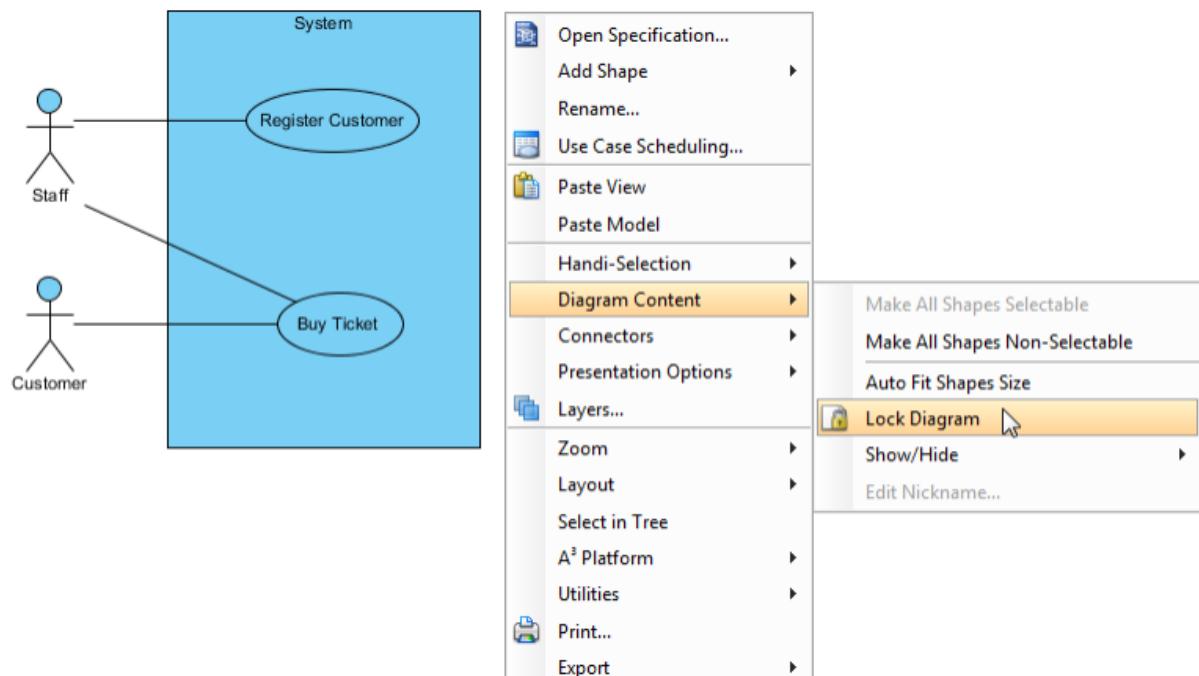
Select **Make All Shapes Selectable**

Locking diagram

For reviewing the diagram, the application of lock diagram is the straight way. Diagram can be locked to be read-only. After it is locked, any shapes on the diagram cannot be modified, inserted or deleted.

Locking the diagram

To lock the diagram, right click on the diagram background and select **Diagram Content > Lock Diagram** from the pop-up menu.



Lock the diagram

After it is locked, the diagram toolbar is disabled subsequently. As a result, no shapes can be selected and the diagram becomes read-only. As you can see the **Diagram Locked** dialog box is shown at the bottom left of the diagram, however, close this dialog box will not unlock the diagram.

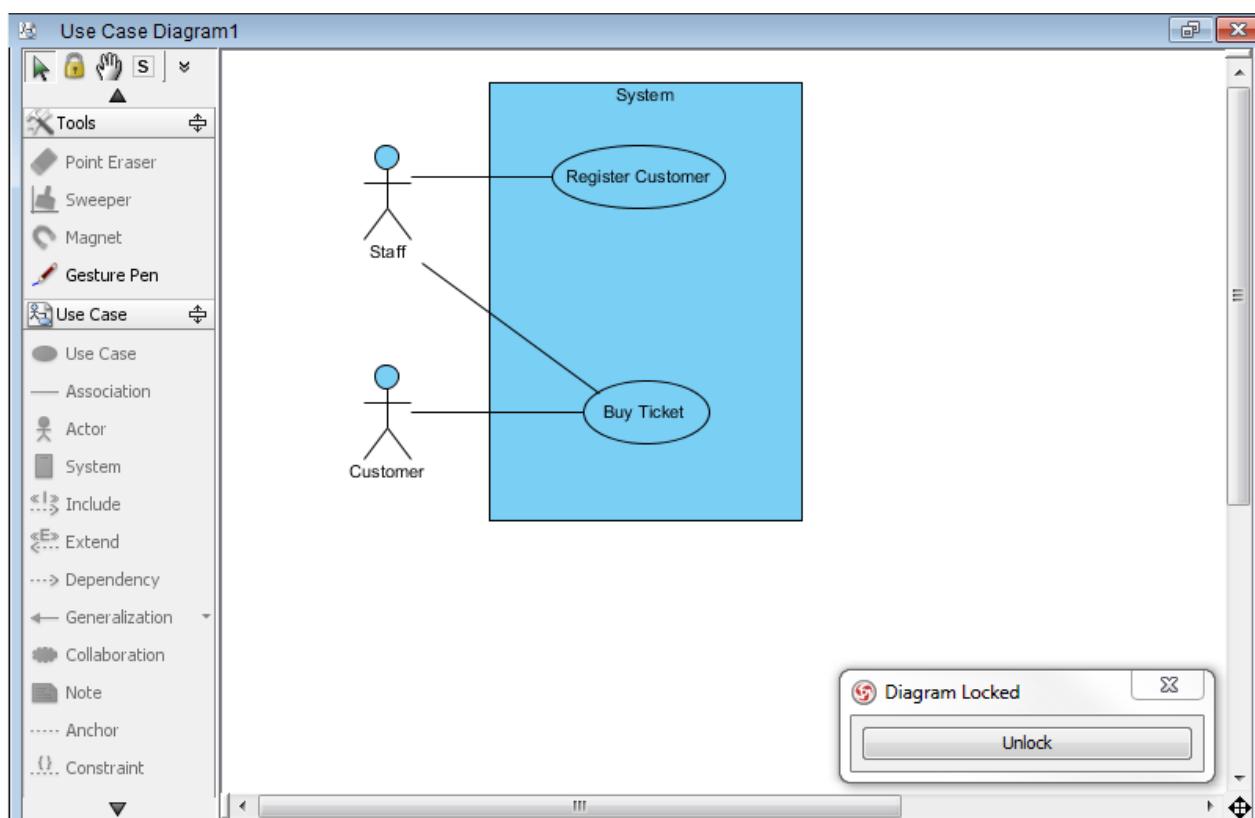
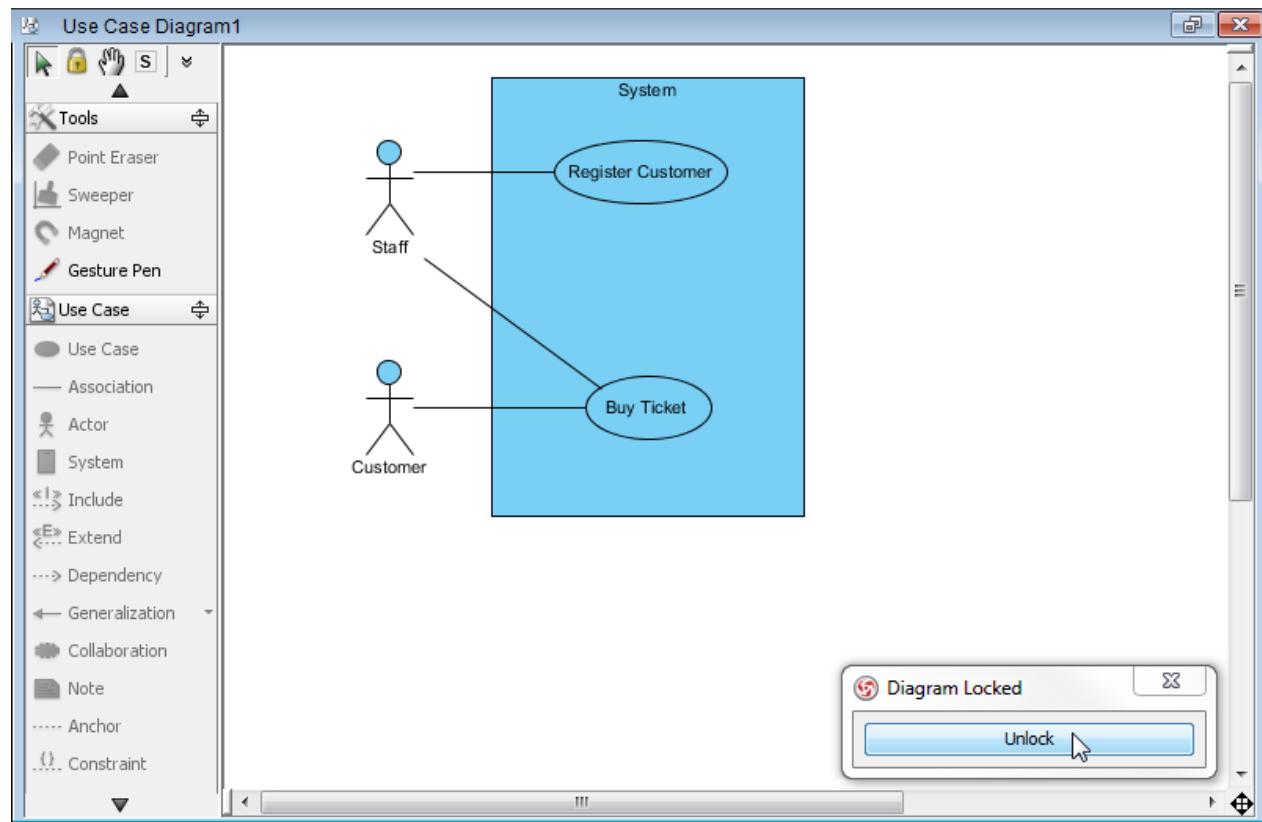


Diagram is locked

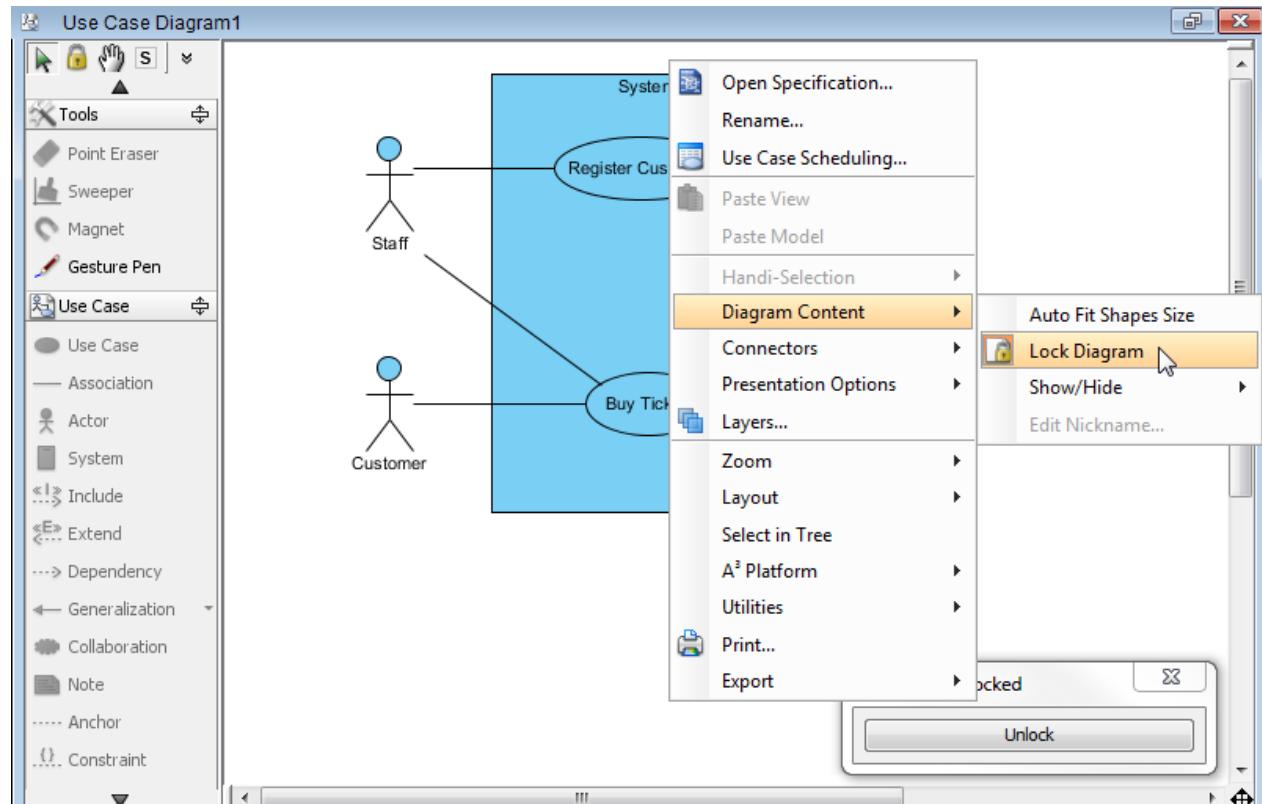
Unlocking the diagram

To unlock the diagram, you should click **Unlock** button on the **Diagram Locked** dialog box.



Click **Unlock** in the **Diagram Locked** dialog box

Alternatively, you can right click on the diagram background and select **Diagram Content > Lock Diagram** from the pop-up menu.



Unlock the diagram from the pop-up menu

Showing model element in multiple diagrams (Context base modeling)

Models can be shown on more than one diagram. To show models on different diagram. You may drag and drop the model from tree to diagram, or copy and paste the model view. For example, you have a diagram contains some classes may be reused on other diagram.

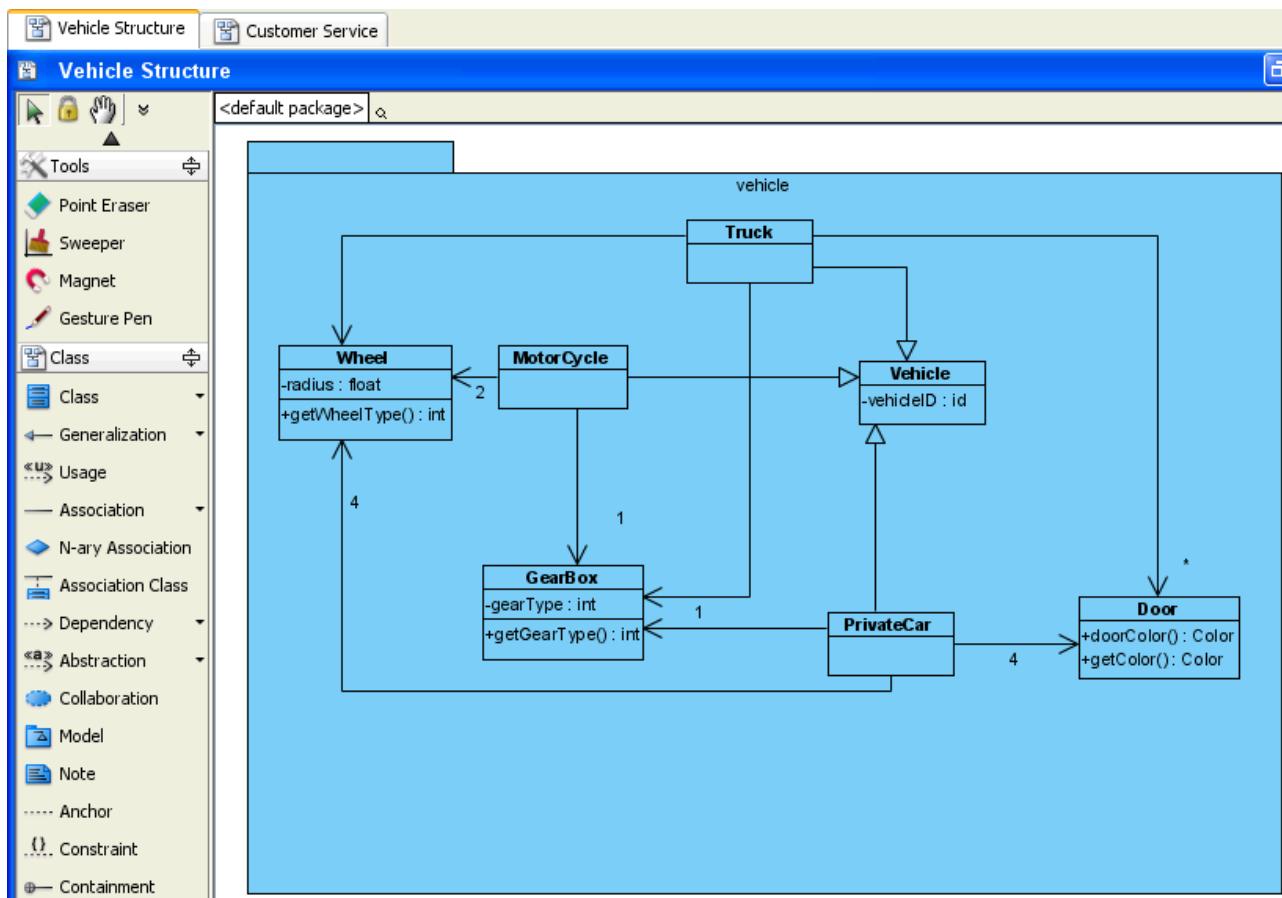


Figure 4-63 Classes may be reused

And have another diagram that will reuse the classes.

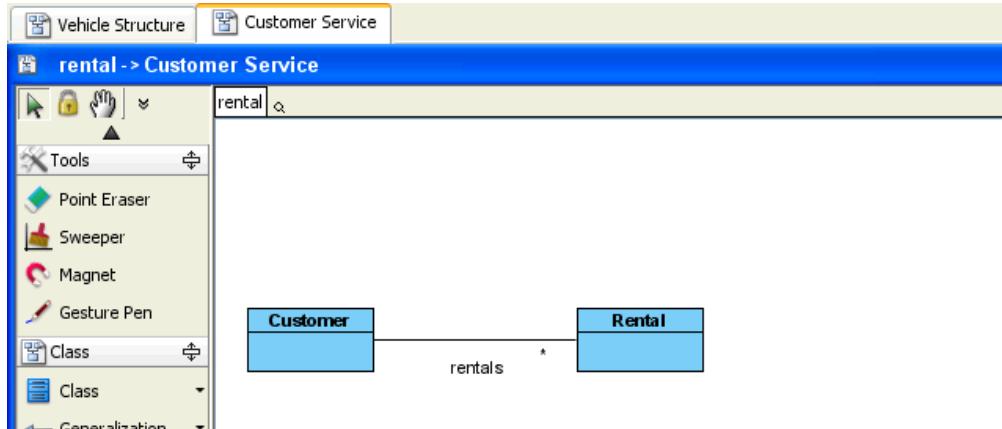


Figure 4-64 Diagram will reuse classes

Reusing model elements

You may drag and drop the models from tree to the diagram.

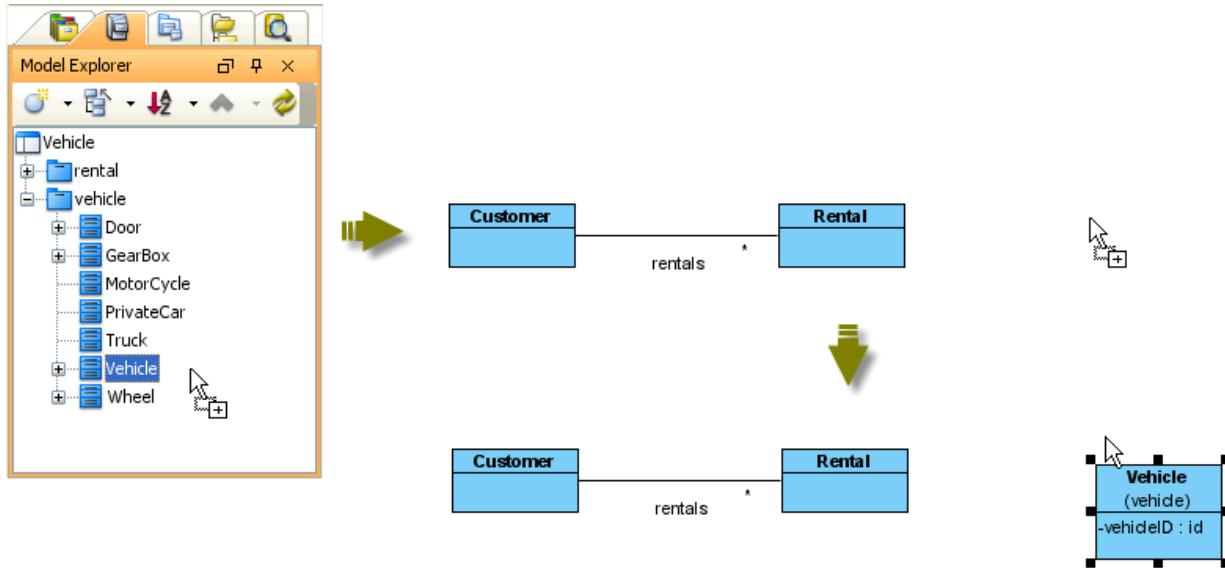


Figure 4-65 Model shown on diagram after drag and drop

You may also copy the model first.

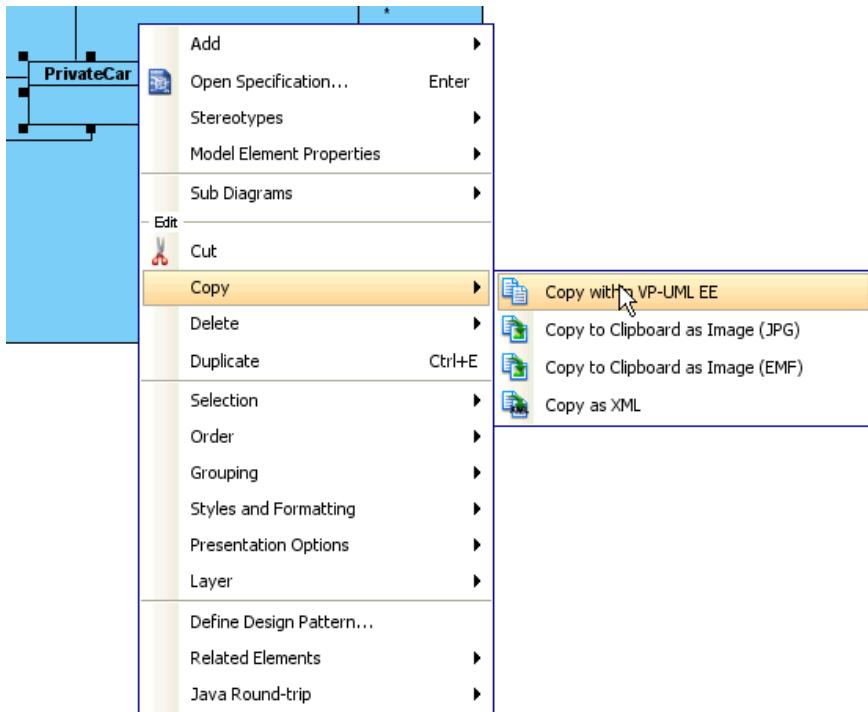


Figure 4-66 Copy the model

And then **Paste View** on the diagram. (**Paste Model** will create a new model.)

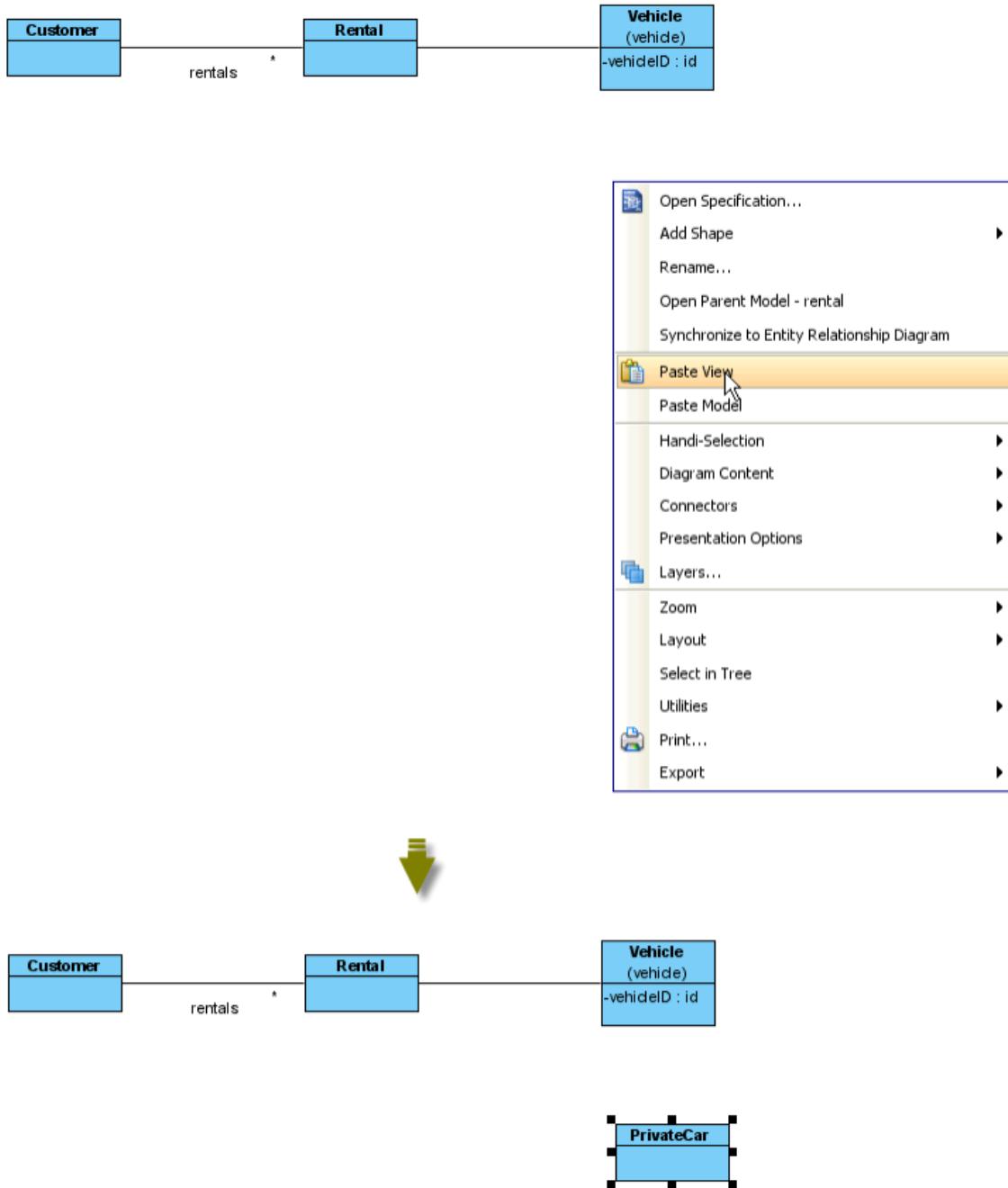


Figure 4-67 Paste view

The relationship between the models may not be shown on diagram, to show the relationship, right-click on the model, select **Related Elements > Visualize Related Model Element...** from popup menu.

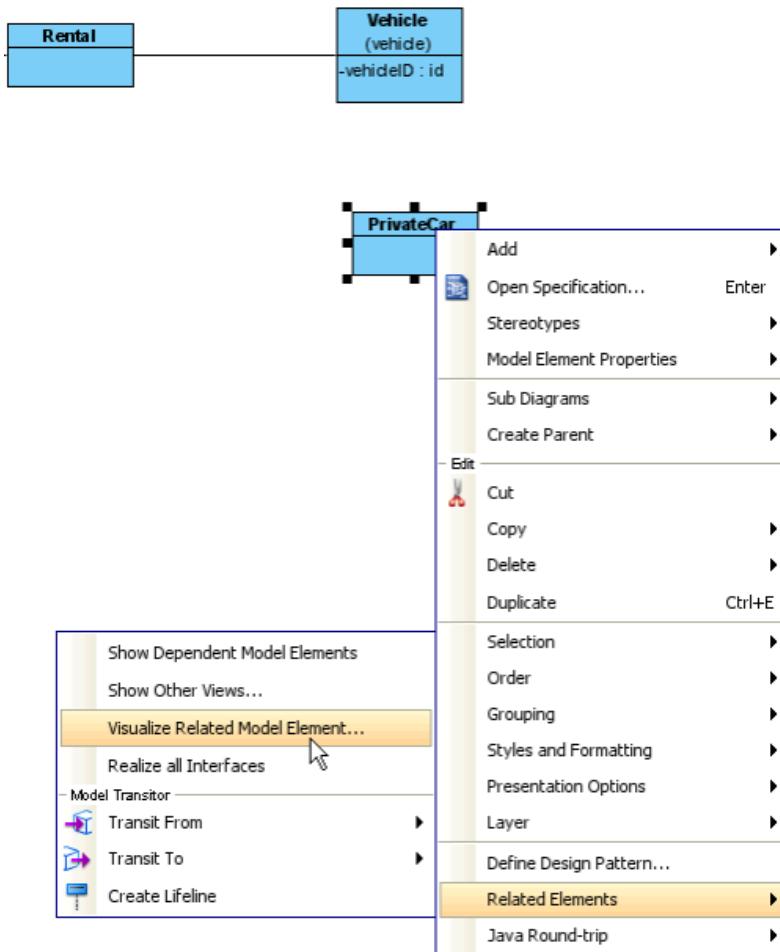


Figure 4-68 Visualize Related Model Element

On **Visualize Related Model Element** dialog box, select the relationships between the models, and select **Visualize** button.

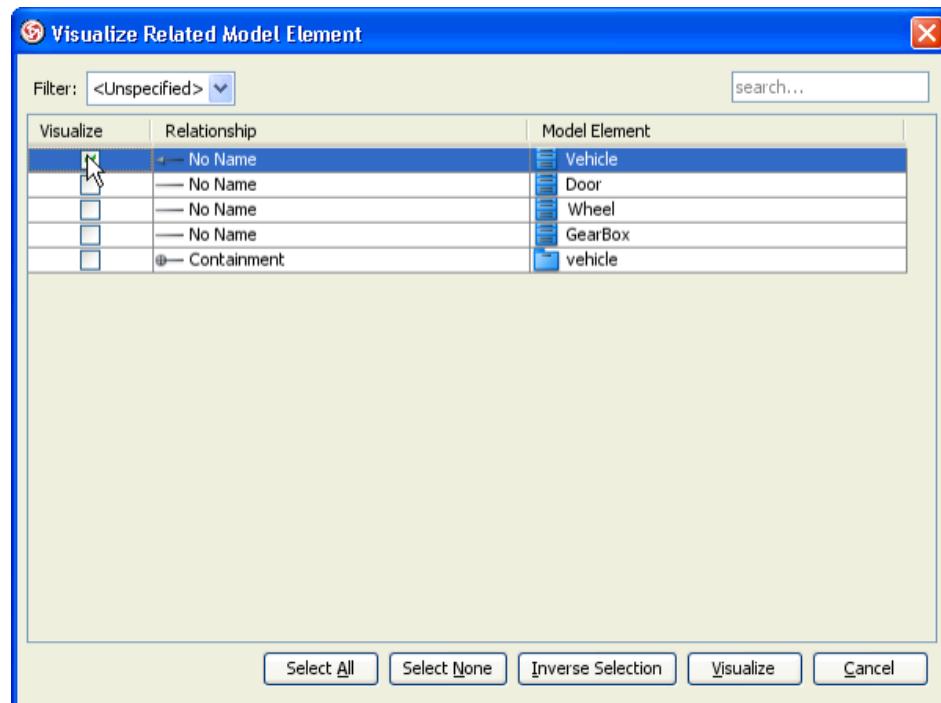


Figure 4-69 Visualize the relationship

The relationship is shown on diagram.

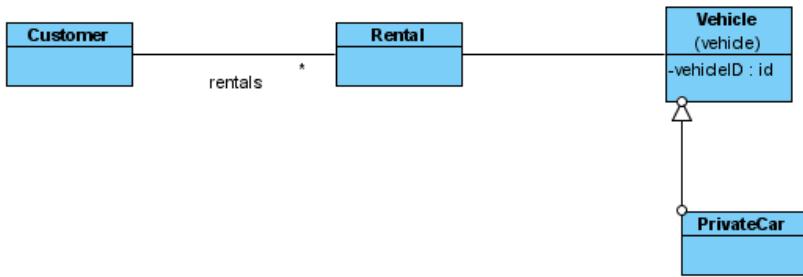


Figure 4-70 Relationship is visualized

Open other views

You can know the model is shown on which diagram(s). To do so, right-click on the model, select **Related Elements > Show Other Views...** from popup menu.

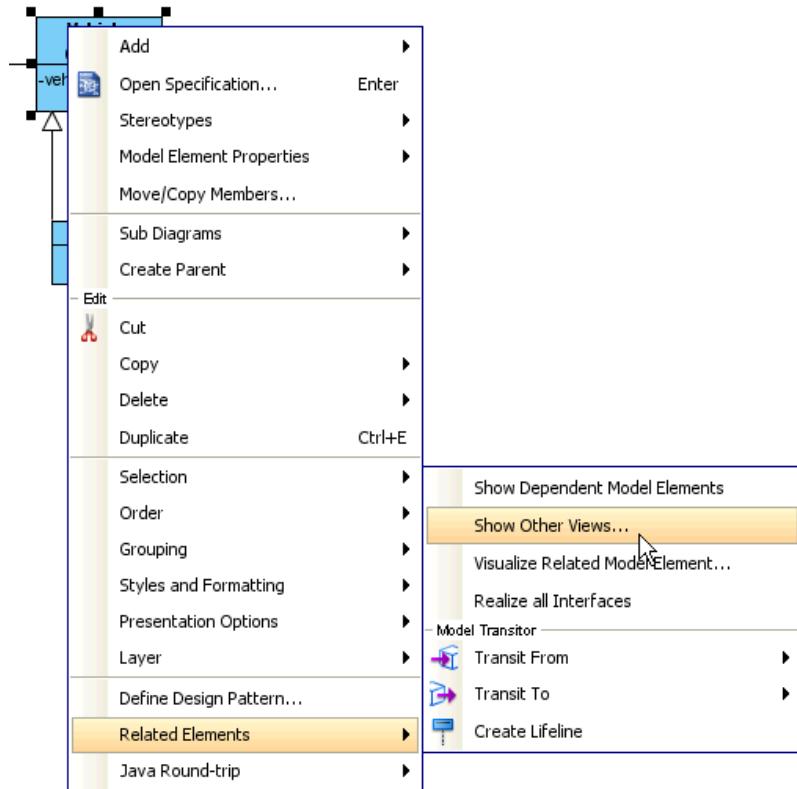


Figure 4-71 Go to other view

The diagrams, that shows this model (besides the current diagram), are listed and shown as preview. You can select the diagram and select **Go to View** button to open the diagram.

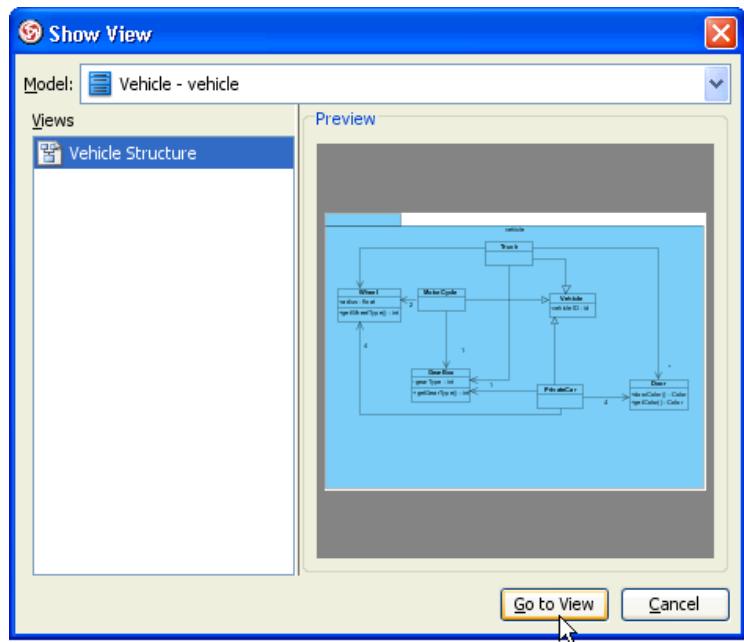


Figure 4-72 Go to other view

The diagram is opened and the model is selected on the diagram.

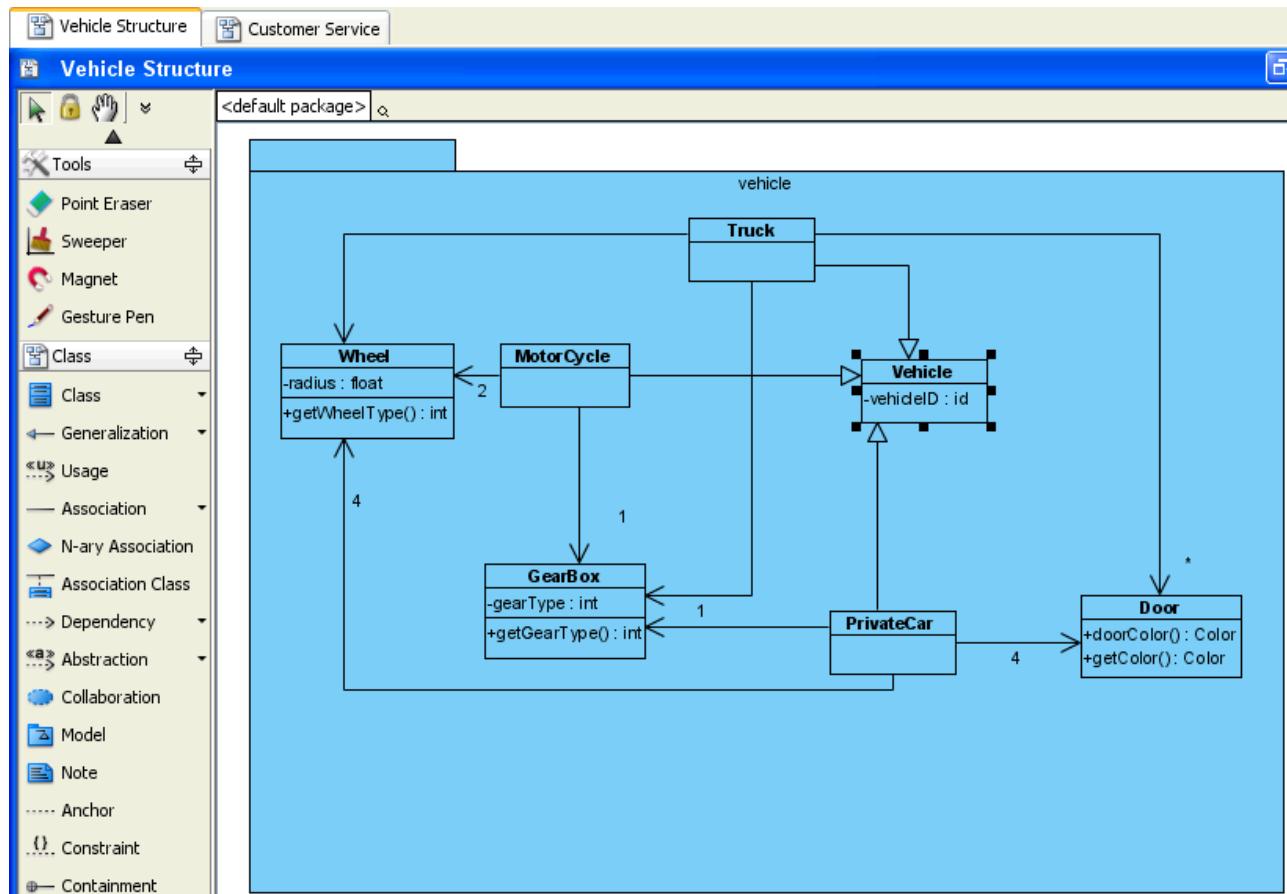


Figure 4-73 Diagram is opened with selecting the model

Enforcing master view between model element and shape

If a model is shown on different diagram. You can specify which shape is the master view of the model. To do so, right-click on the shape, select Selection > Set as Master View from popup menu. (If the shape you right-clicked already is the master view, the Set as Master View won't be shown on popup menu)

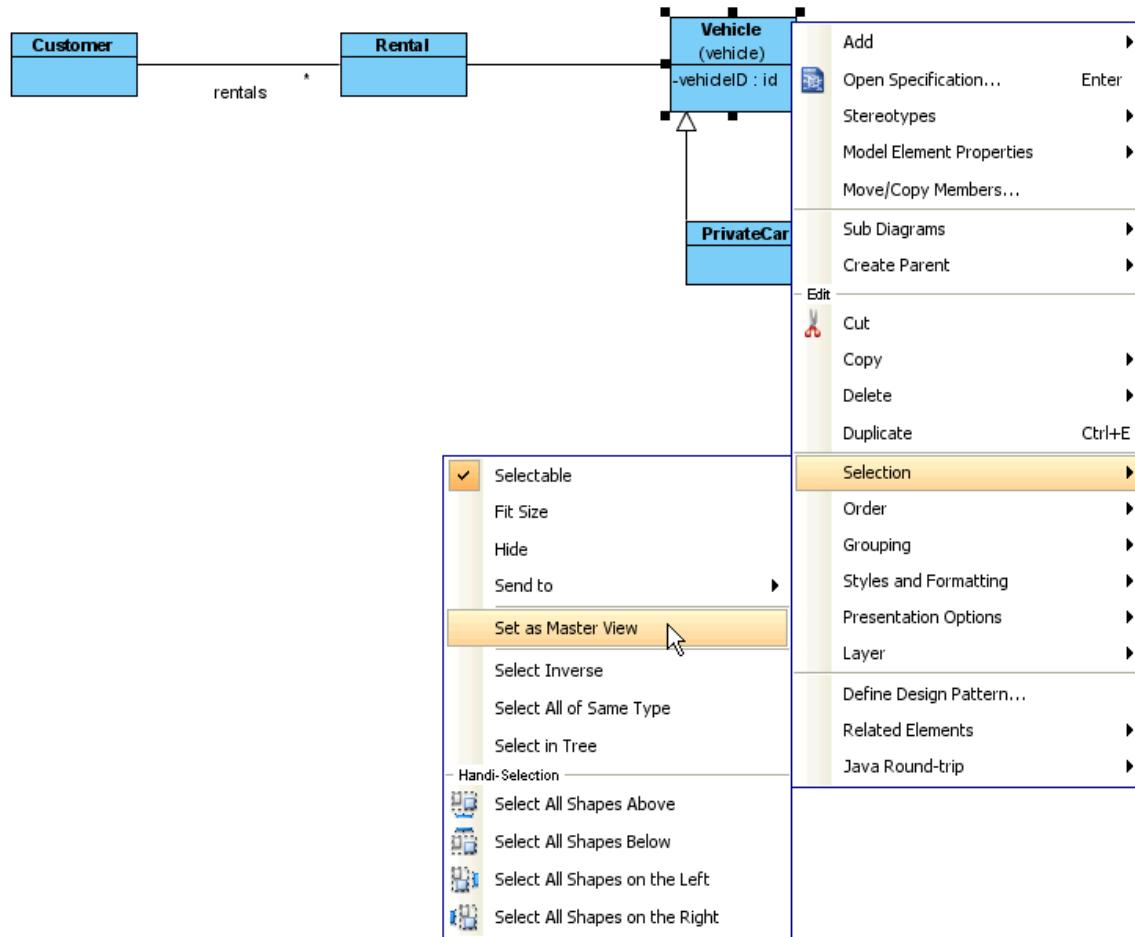


Figure 4-74 Set the shape as Master View of the model

Using overview diagram

Overview Diagram can show a set on diagrams as preview, and shows the relationships between the diagrams.

Creating overview diagram

To create Overview Diagram, you may select **File > New Diagram > Others > Overview Diagram** from main menu.

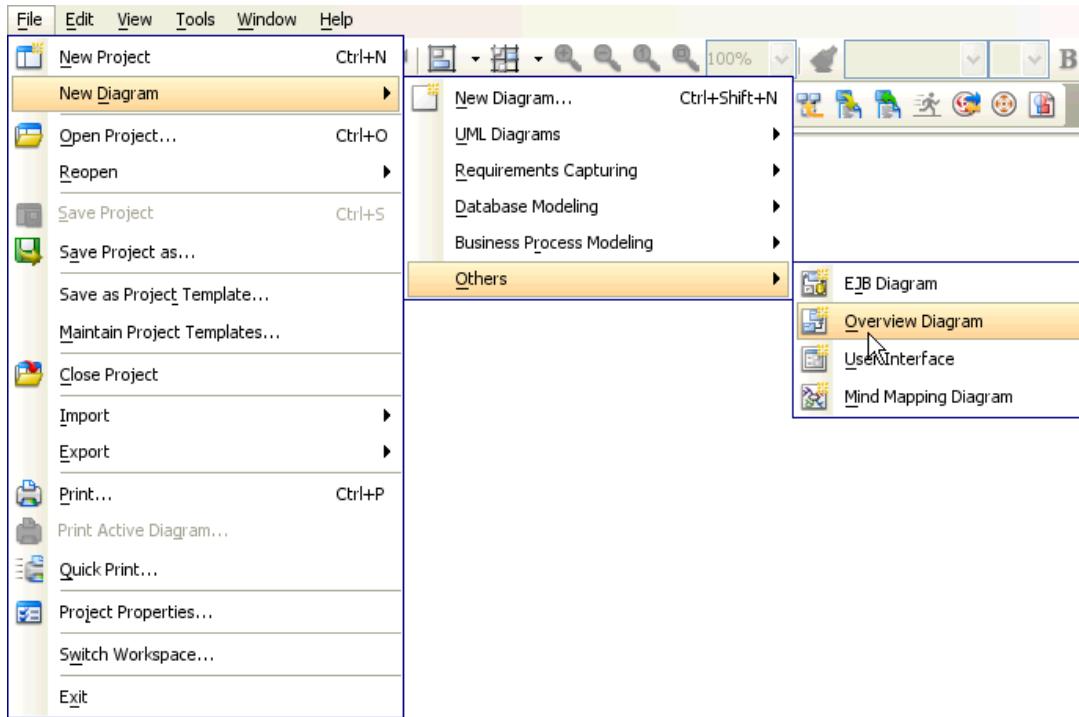


Figure 4-75 Create Overview Diagram

The overview diagram is created.

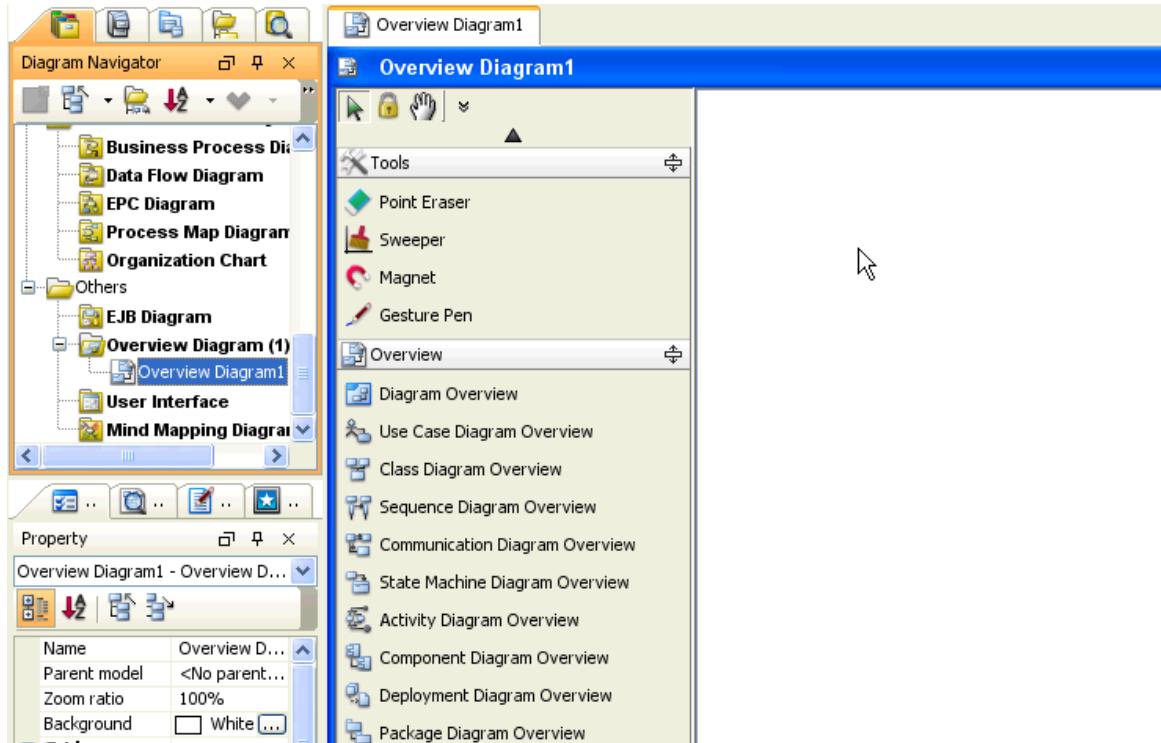


Figure 4-76 Overview Diagram is created

Creating use case diagram

You can create diagram from overview diagram.

To create an Use Case Diagram from overview diagram, select **Use Case Diagram Overview** from platelet.

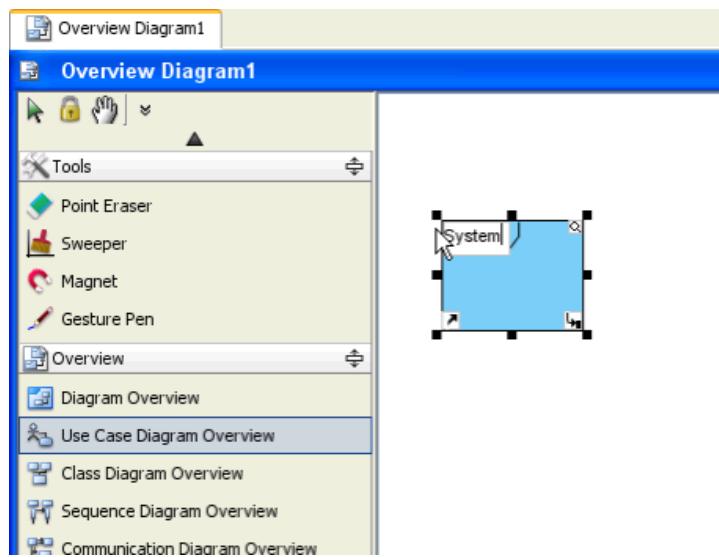


Figure 4-77 You may name the use case diagram

The use case diagram will be auto opened to let you draw the content.

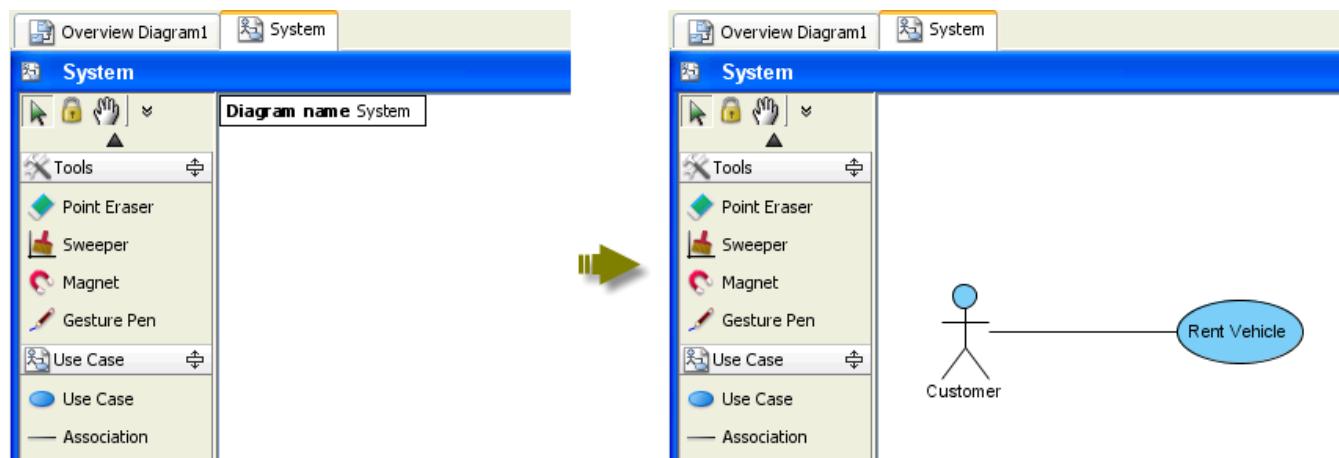


Figure 4-78 Draw the use case diagram

Switch back to overview diagram, you can see the preview of the use case diagram is shown.

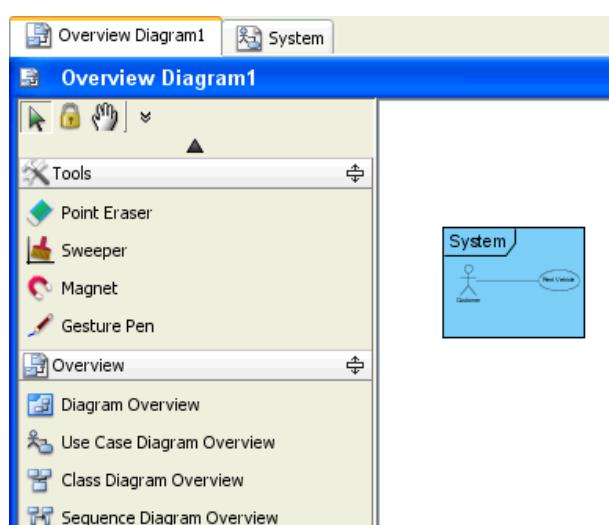


Figure 4-79 Overview diagram shows the use case diagram

Showing existing diagram

Besides creating a new diagram from overview diagram, you can also show an existing diagram on overview diagram.
Assume you have a sequence diagram.

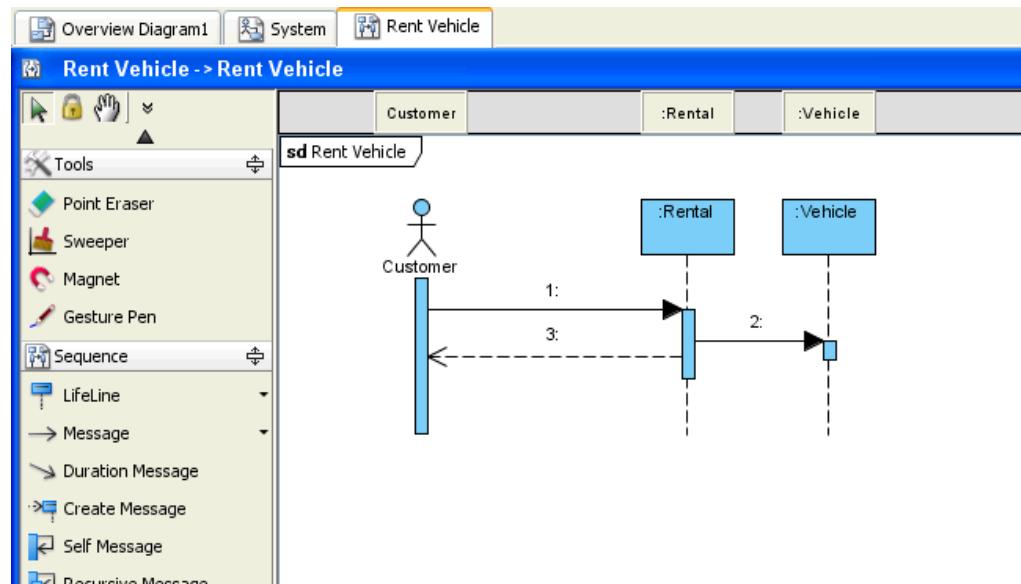


Figure 4-80 Sequence diagram is already exists

1. On overview diagram, create a **Diagram Overview**.

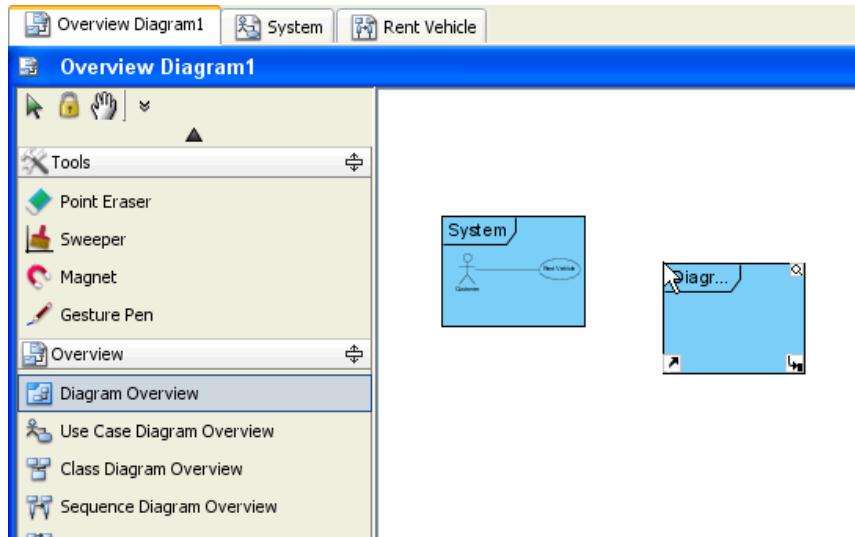


Figure 4-81 Create Diagram Overview

2. Right-click on the diagram overview and select **Associate to Diagram...** from popup menu.

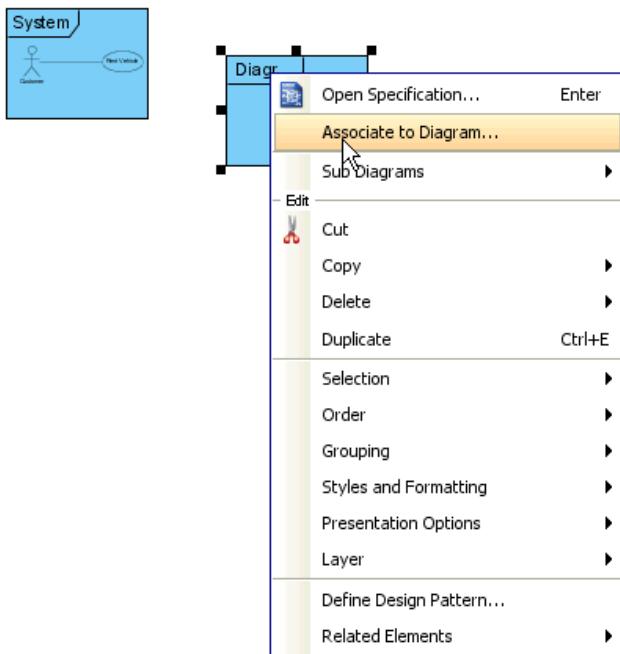


Figure 4-82 Associate to diagram

3. Select the sequence diagram and select **OK** button.

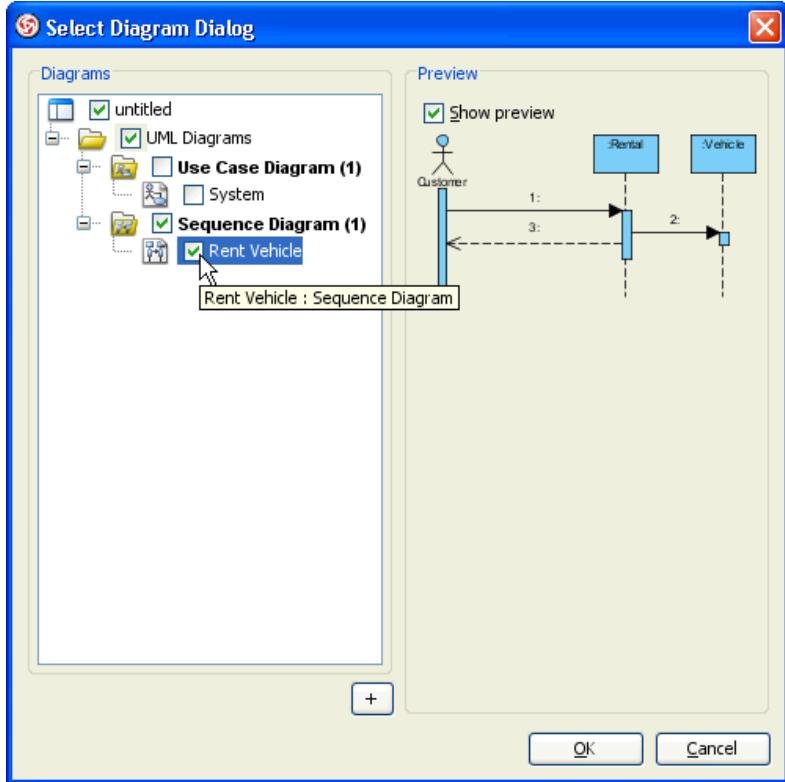


Figure 4-83 Select existing diagram

4. The diagram overview will show the sequence diagram. You also can resize the diagram overview to preview the diagram clearly. And add relationship between the diagram overview.

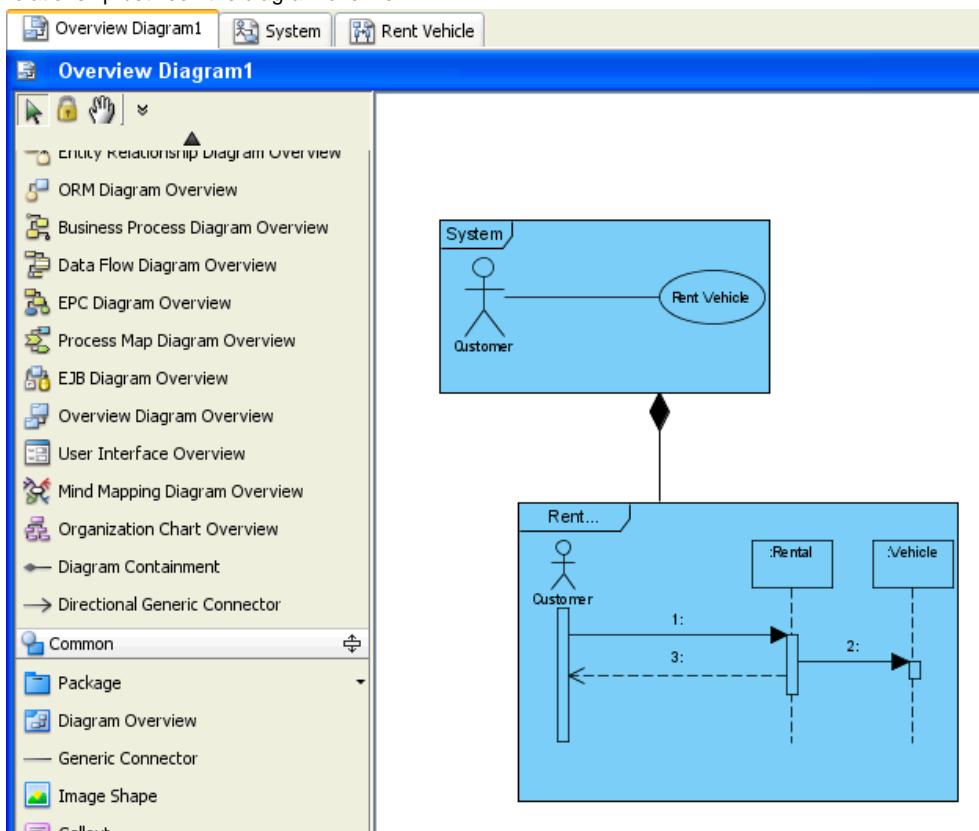


Figure 4-84 Diagrams are shown

Reference to external resources

Reference to files

1. Move the mouse cursor over the shape to add reference, and click on the **References** resource.

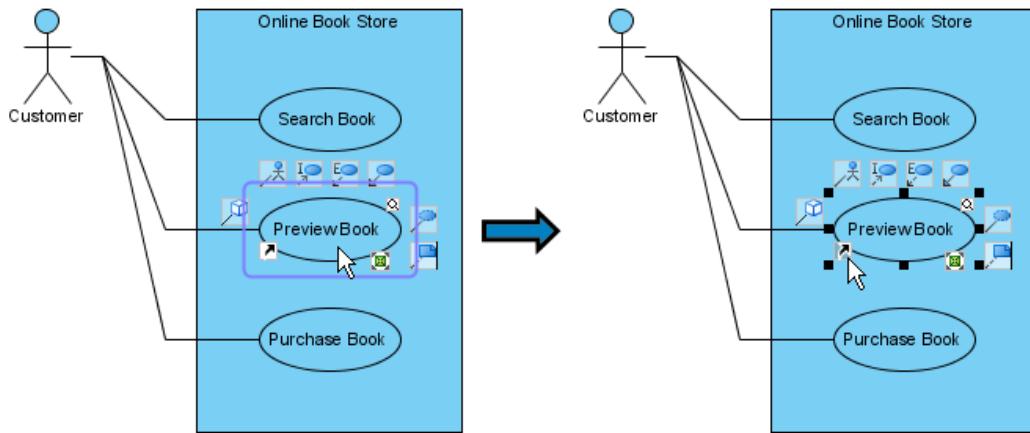


Figure 5-1 Mouse over references resource

2. Click on **Edit References...** in the popup menu.

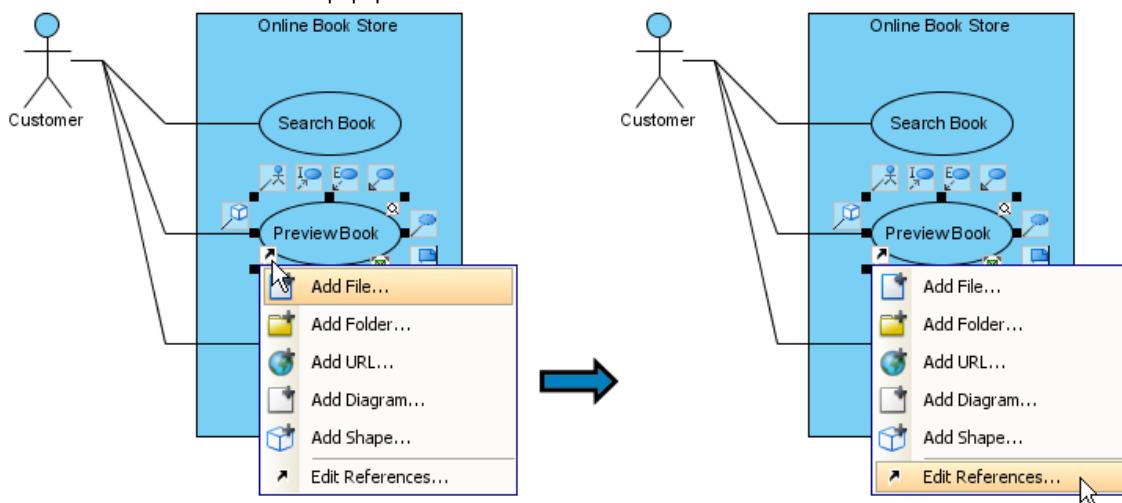


Figure 5-2 Edit references from resource

3. Click on the **Add File...** button.

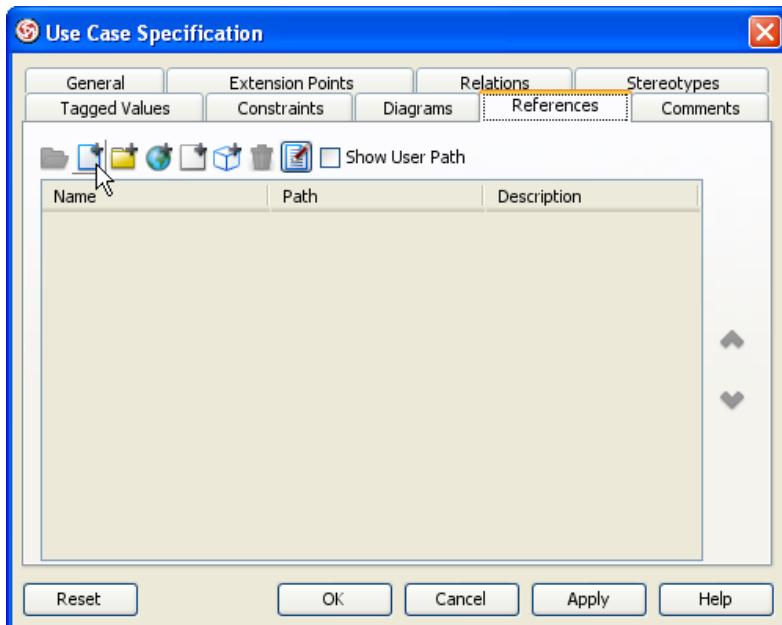


Figure 5-3 Adding file reference

4. Enter the referenced file path.

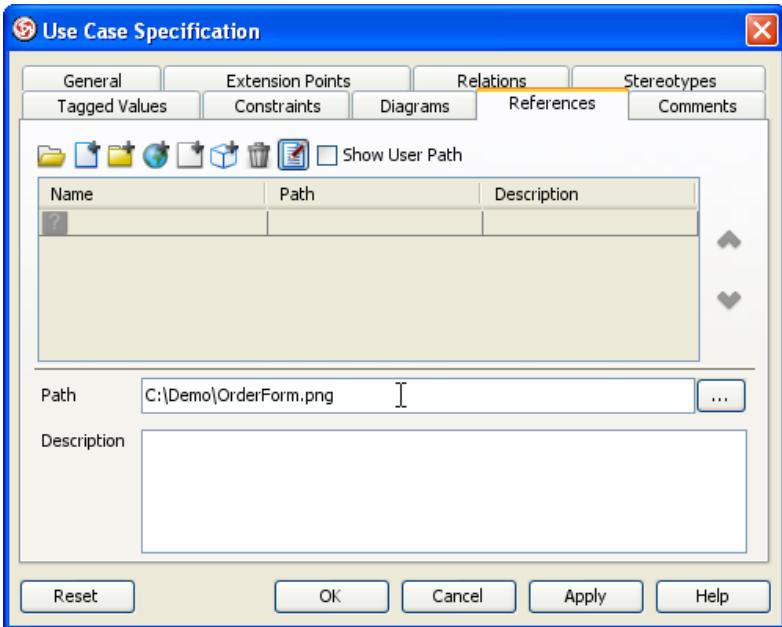


Figure 5-4 Inputting file reference path

5. Input description of the file.

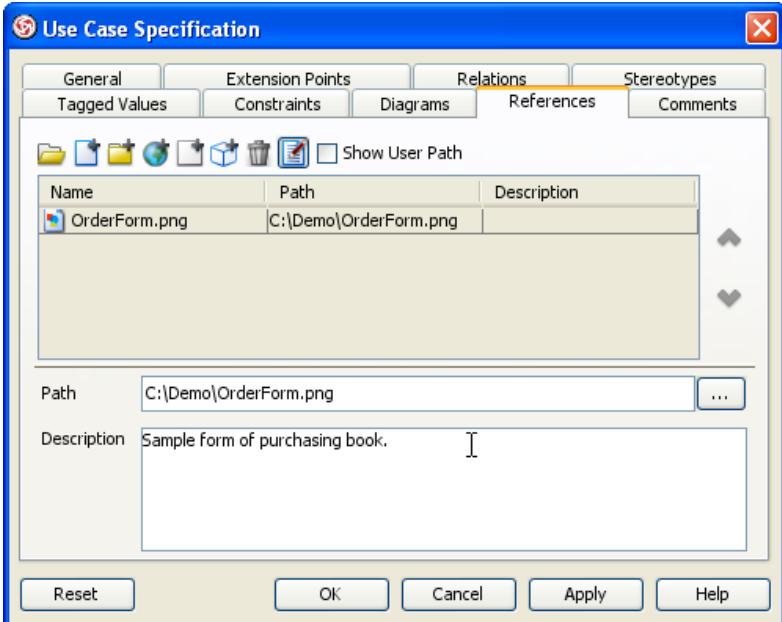


Figure 5-5 Inputting file reference description

6. Click **Apply** to confirm the reference creation.

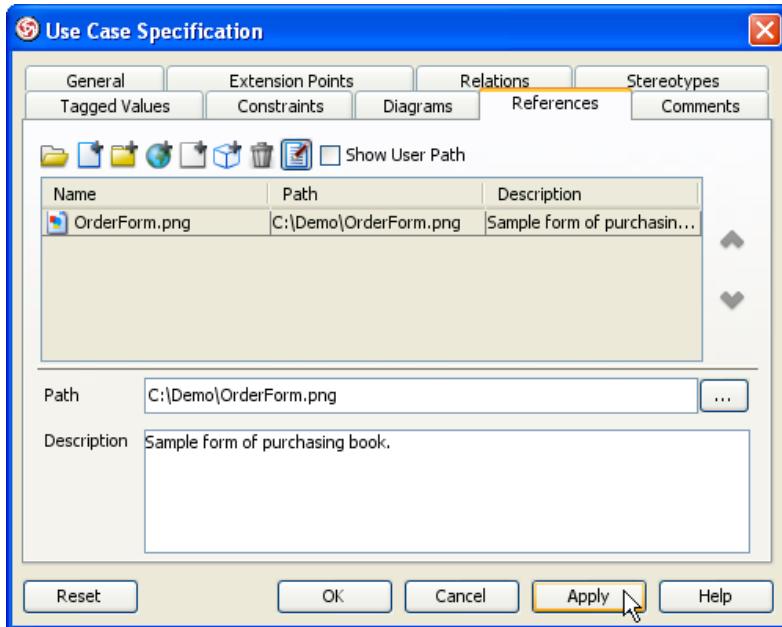


Figure 5-6 Apply file reference

7. Select the file reference to be opened.

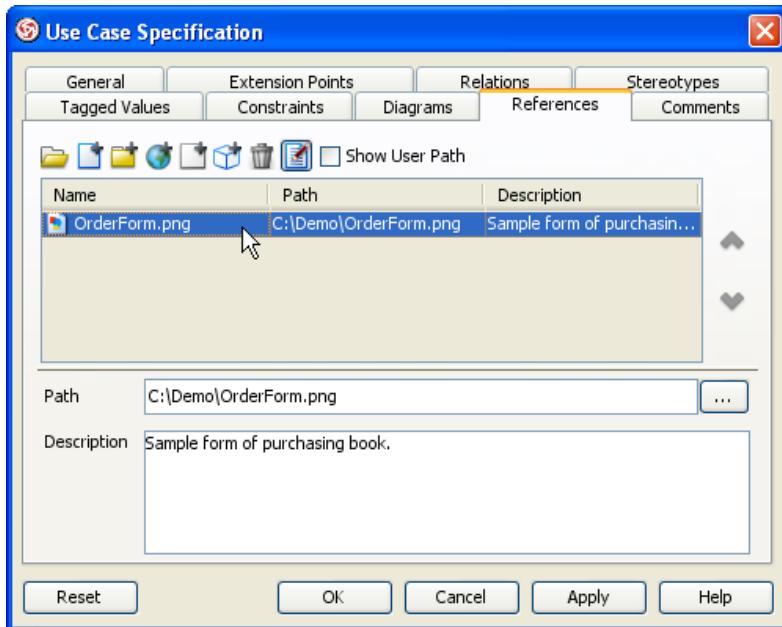


Figure 5-7 Select file reference

8. Click on the **Open...** button to open the selected file reference.

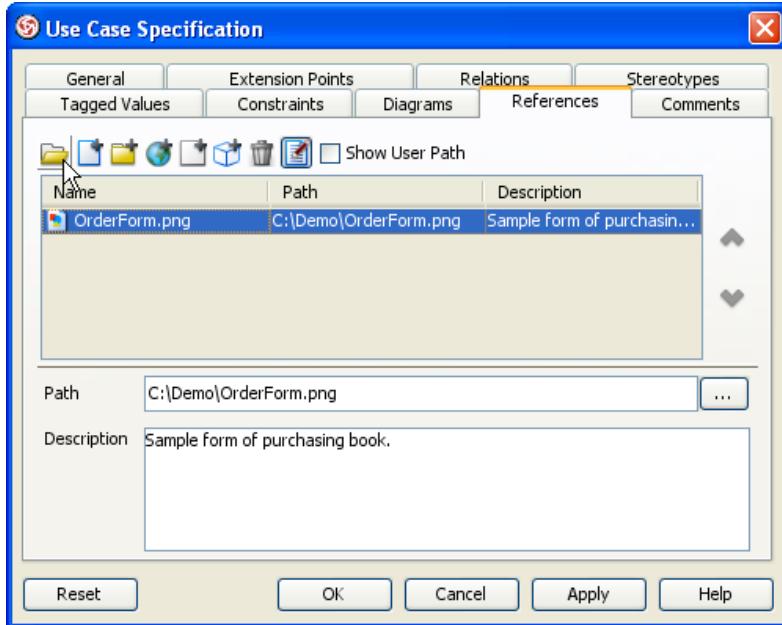


Figure 5-8 Open file reference

Reference to folder

Adding a folder reference

1. Move the mouse cursor over the shape to add reference, and click on the **References** resource.

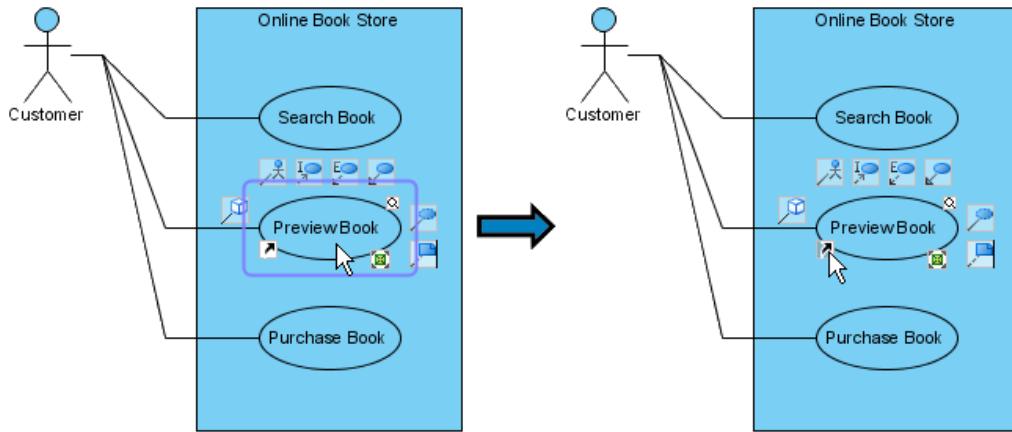


Figure 5-9 Mouse over references resource

2. Click on **Add Folder...** in the popup menu.

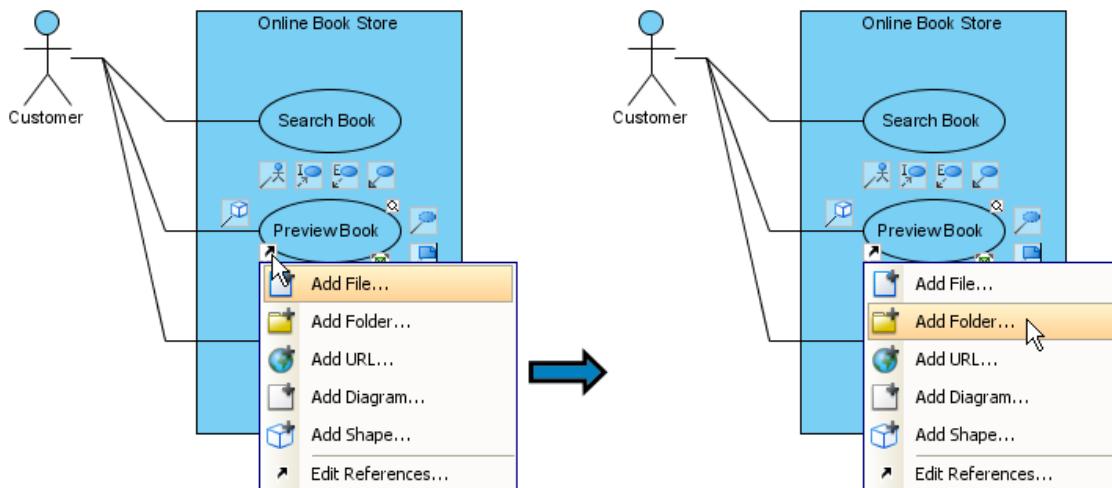


Figure 5-10 Add folder from resource

3. Click on the ... button for Path.

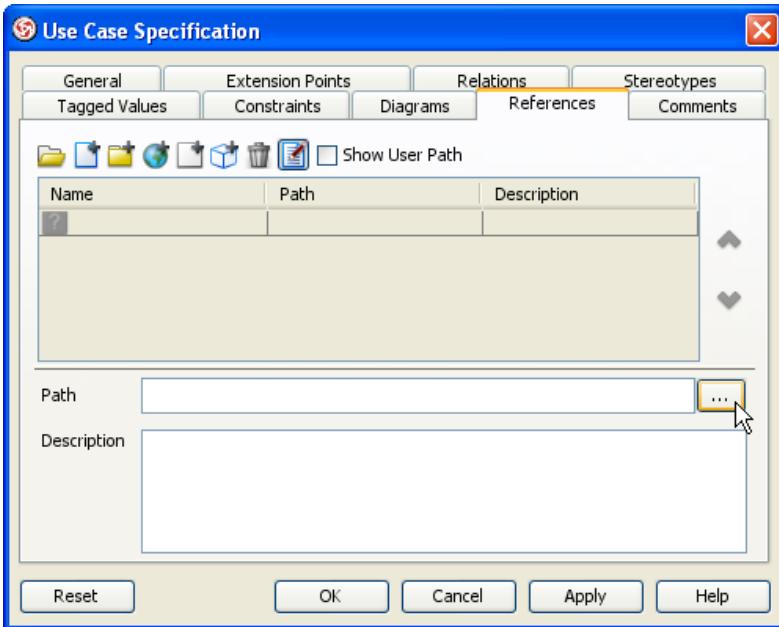


Figure 5-11 Browse folder

4. Select folder to be referenced.

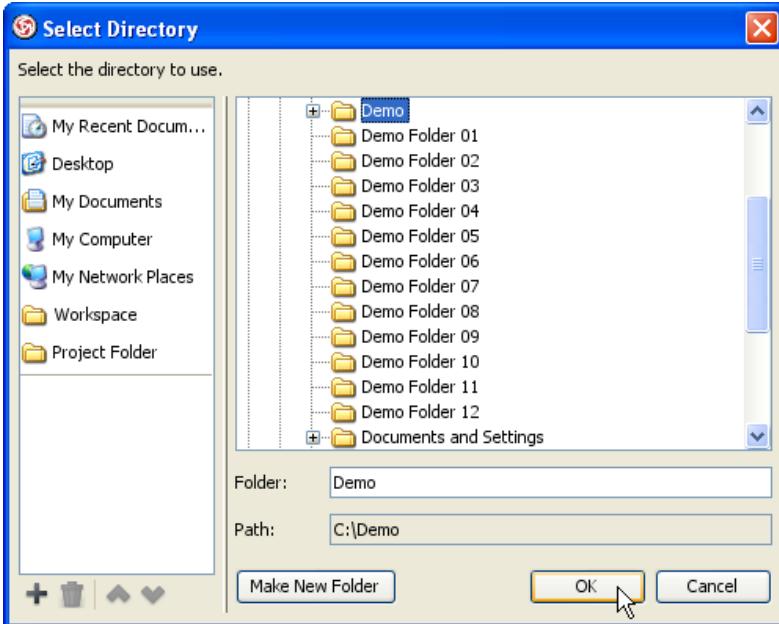


Figure 5-12 Select reference folder

5. Input description of the folder.

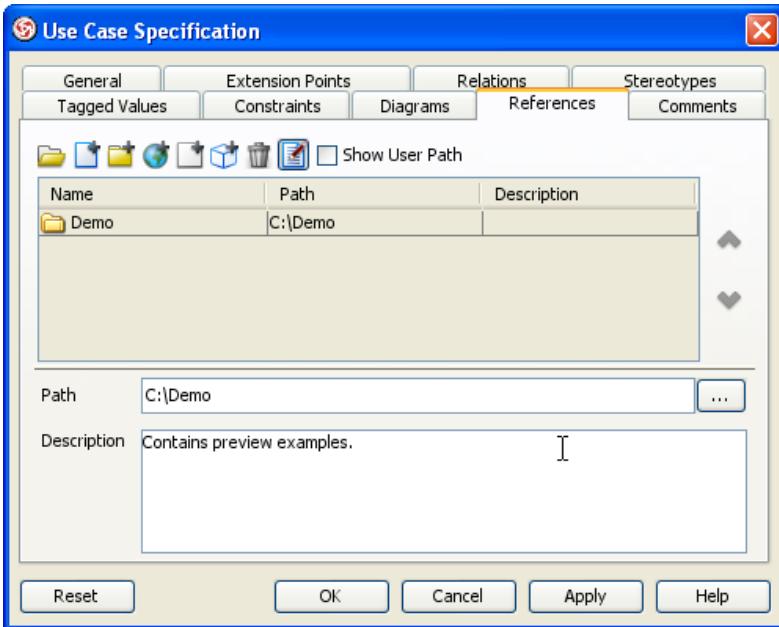


Figure 5-13 Inputting folder reference description

6. Click on the **Apply** button to confirm the creation of reference.

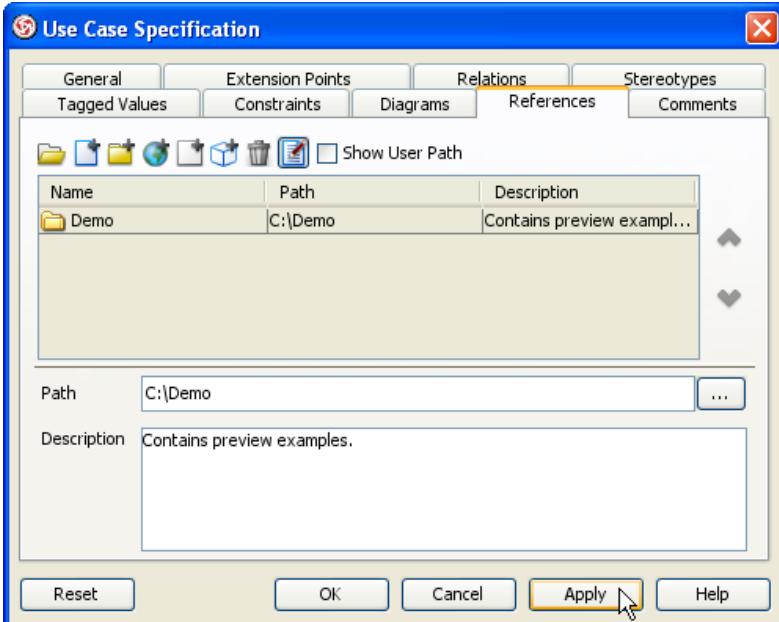


Figure 5-14 Apply folder reference

Opening a folder reference

- Move the mouse cursor over the shape to open reference, and click on the **References** resource.

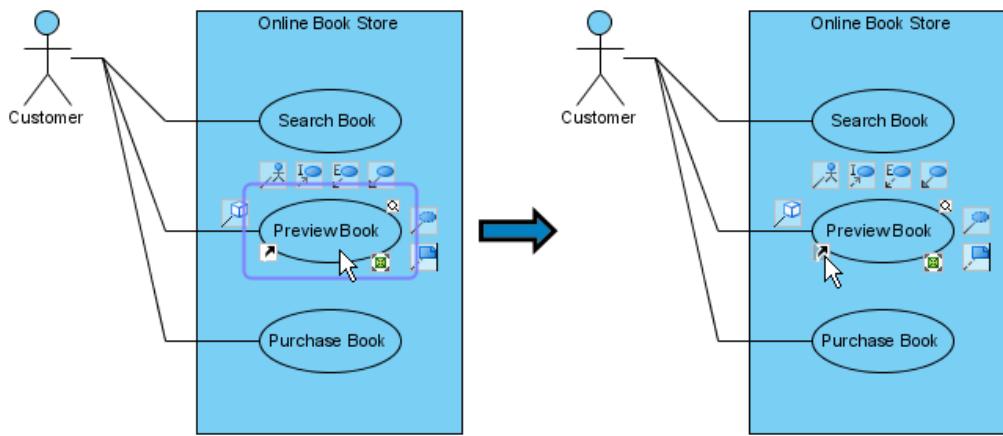


Figure 5-15 Mouse over references resource

- Select the folder reference to open.

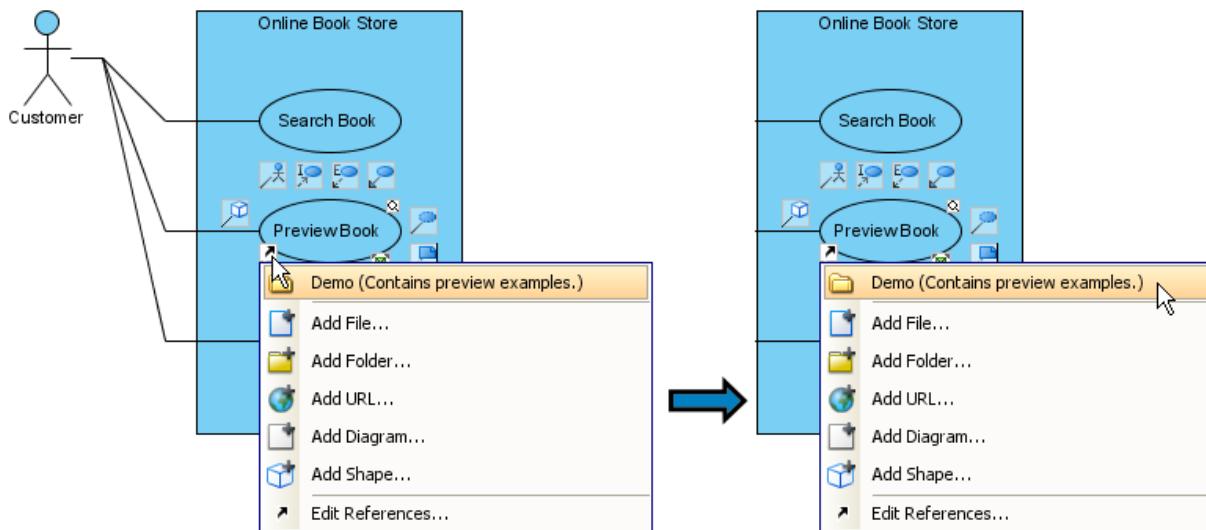


Figure 5-16 Open folder reference

Reference to URL

- Right click on shape and select **Open Specification...** from the popup menu.

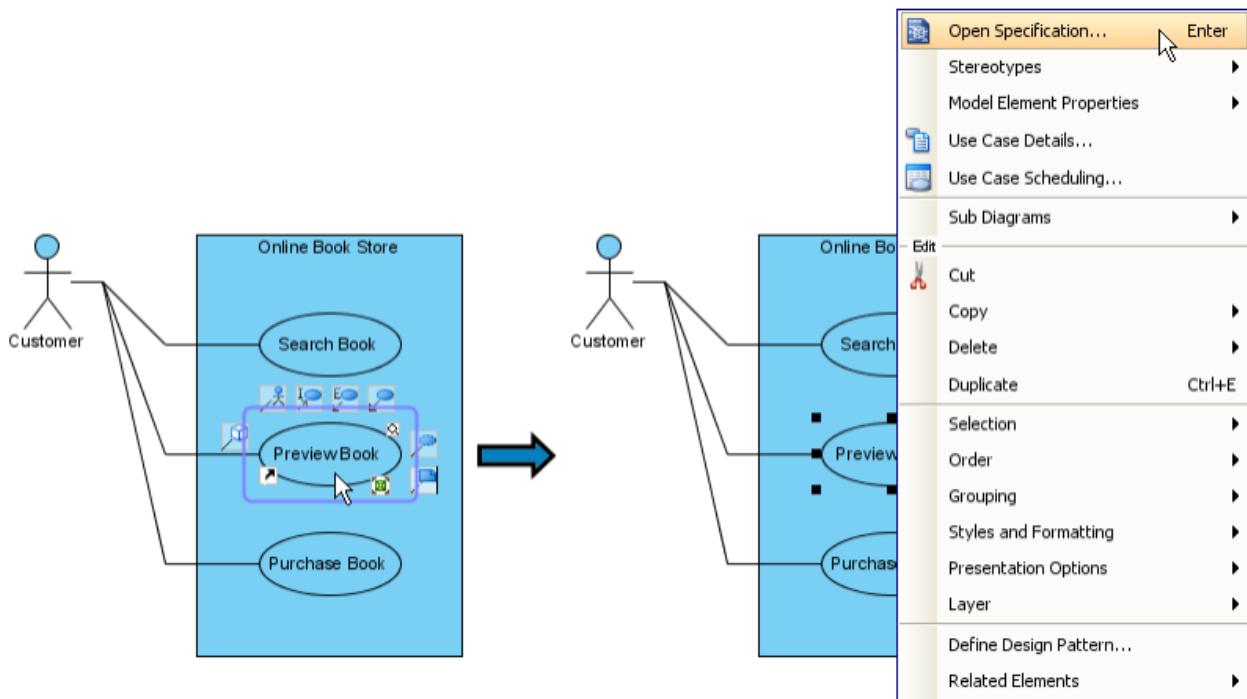


Figure 5-17 Show popup of shape

- Open the **References** tab.

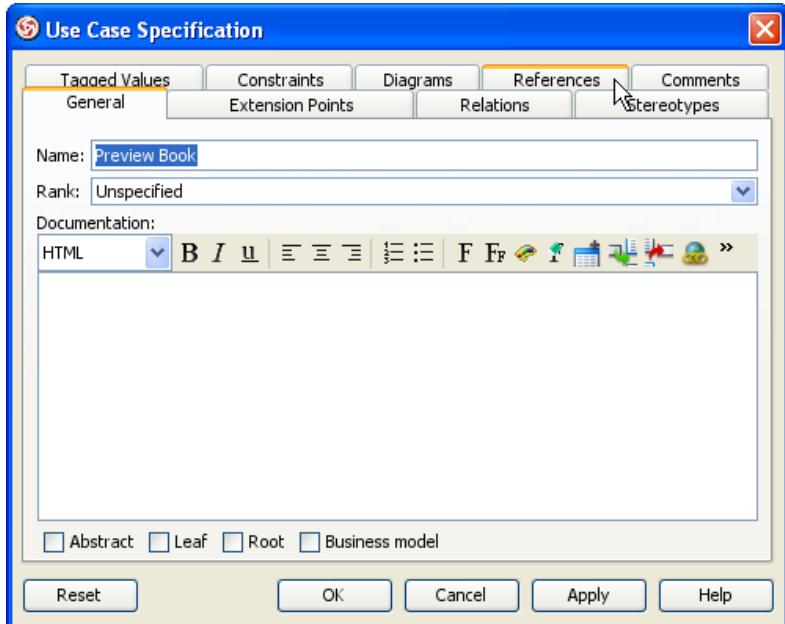


Figure 5-18 Switch to references tab

3. Click on the **Add URL...** button.

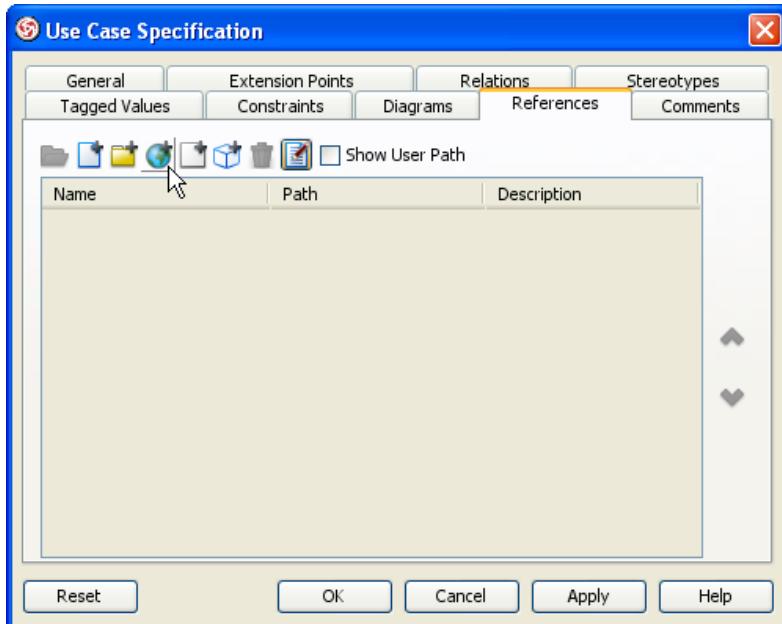


Figure 5-19 Adding URL reference

4. Input the URL path.

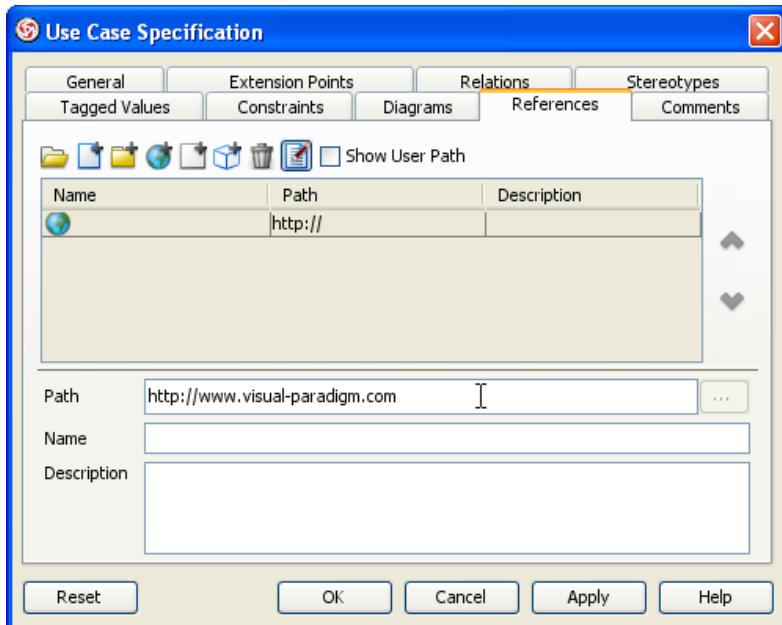


Figure 5-20 Inputting URL reference path

5. Input the URL name.

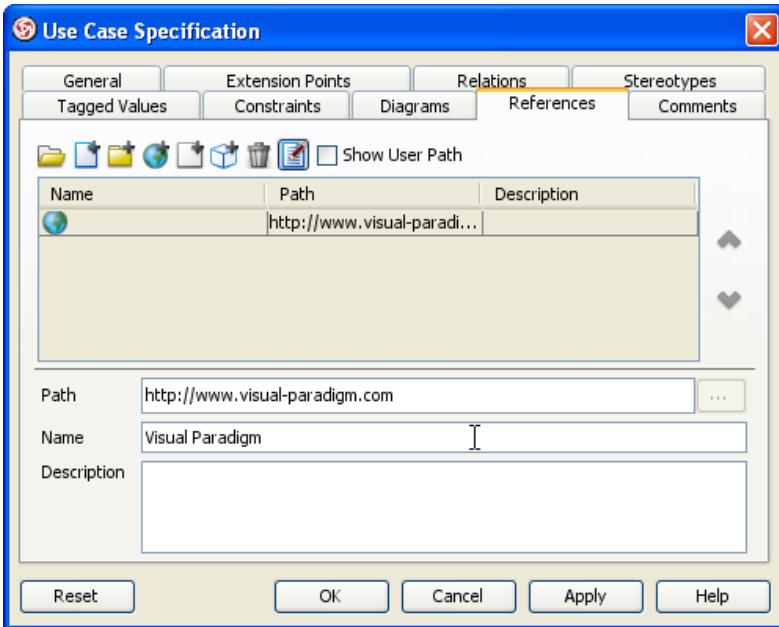


Figure 5-21 Inputting URL reference name

6. Input the URL description.

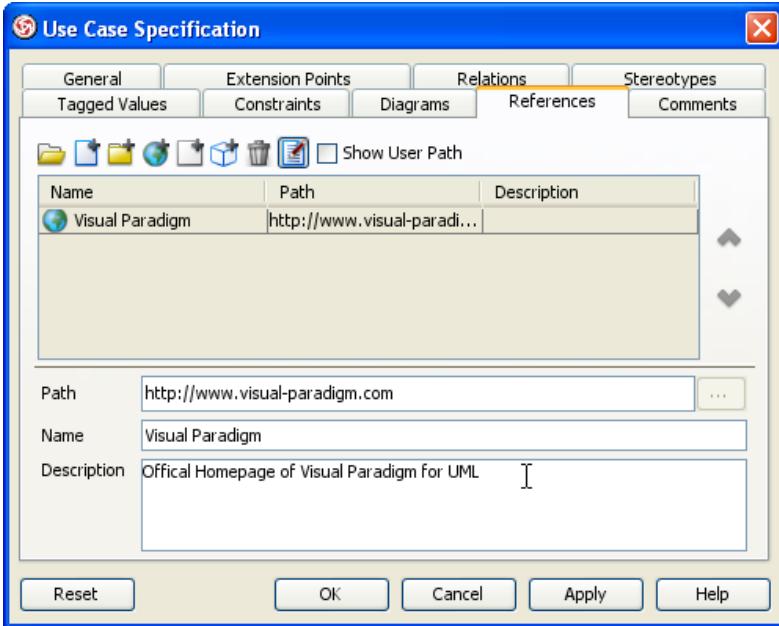


Figure 5-22 Inputting URL reference description

7. Click **Apply** to confirm reference creation.

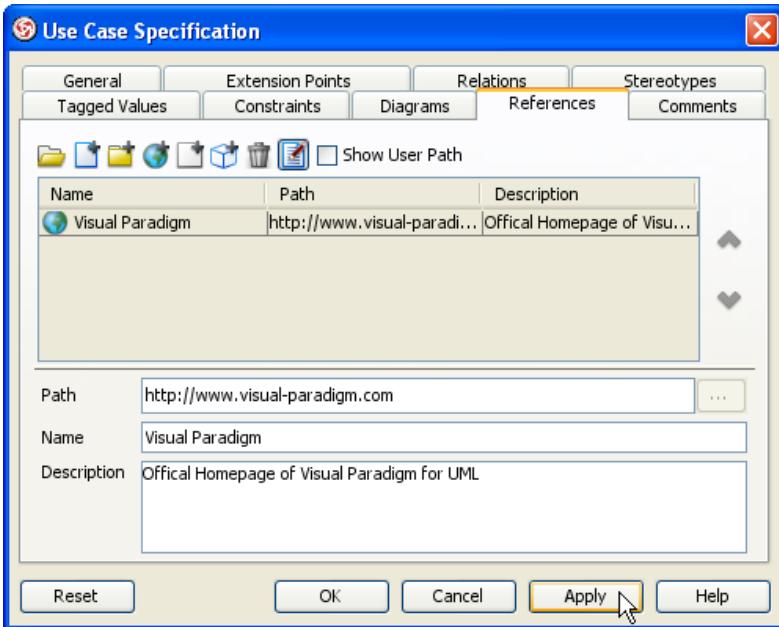


Figure 5-23 Apply URL reference

8. Click on the **Details** button suspend the details pane.

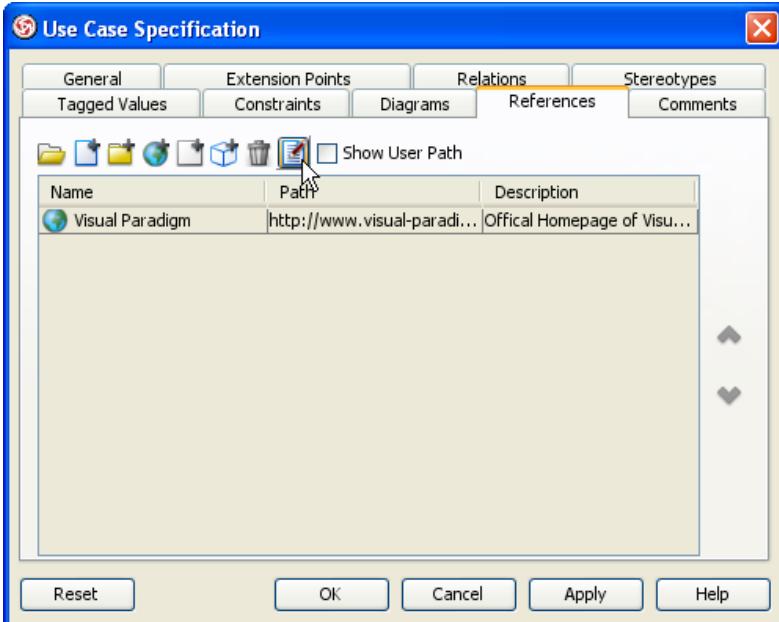


Figure 5-24 Hide reference details

9. Select the URL reference to be opened.

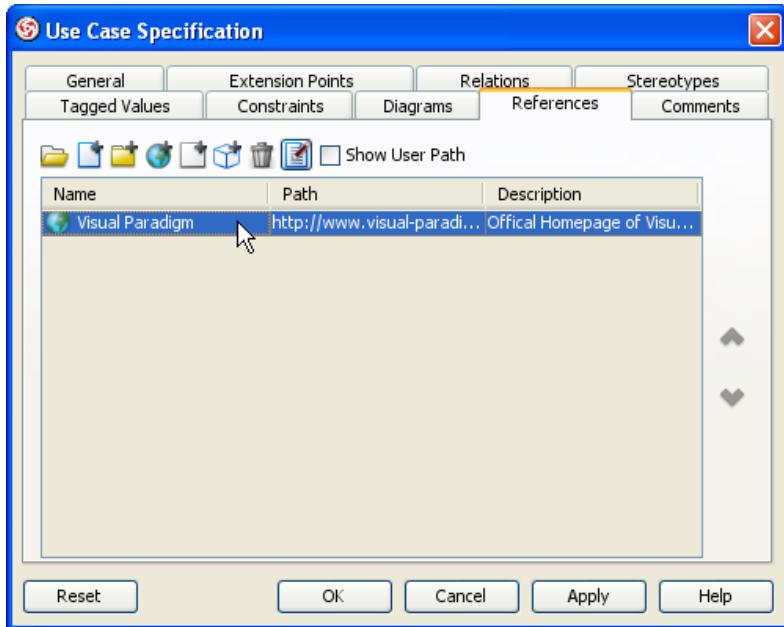


Figure 5-25 Select URL reference

10. Click on the **Open** button to open reference.

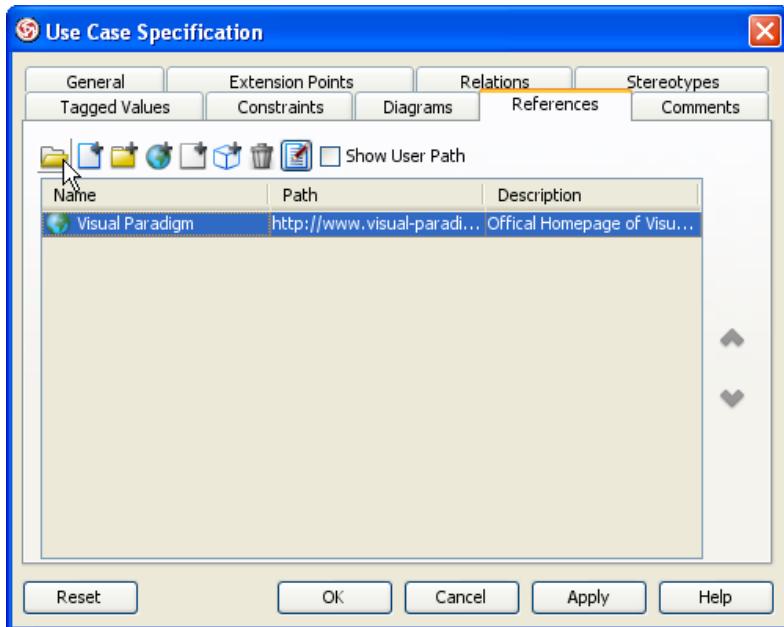


Figure 5-26 Open URL reference

Reference to diagrams and shapes

Reference to diagrams

1. Mouse over shape and click on References resource.

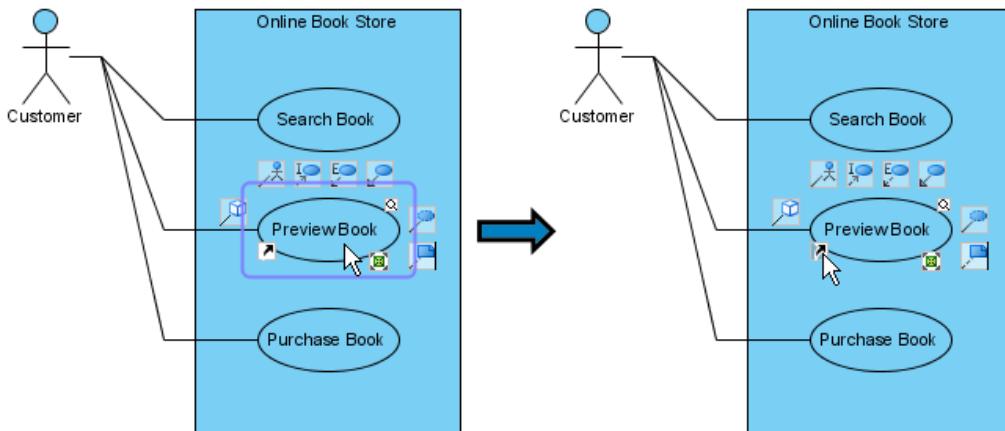


Figure 5-27 Mouse over references resource

2. Click on Edit References... menu item.

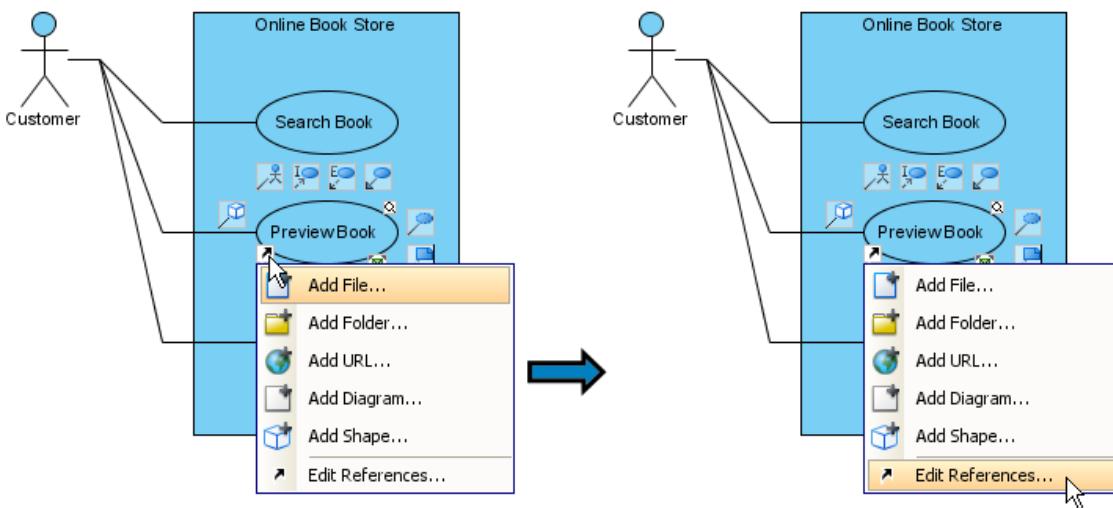


Figure 5-28 Edit references from resource

3. Press Add Diagram... button.

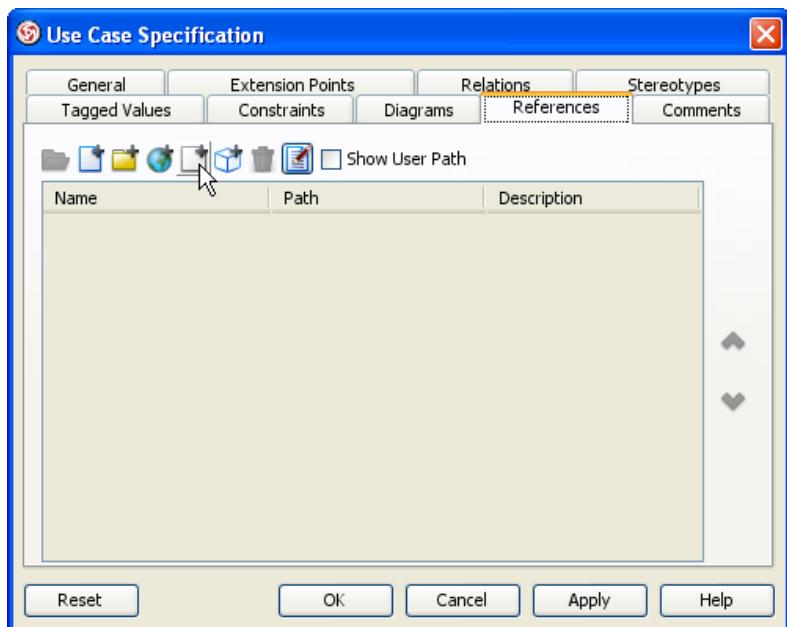


Figure 5-29 Adding diagram reference

4. Select reference diagram.

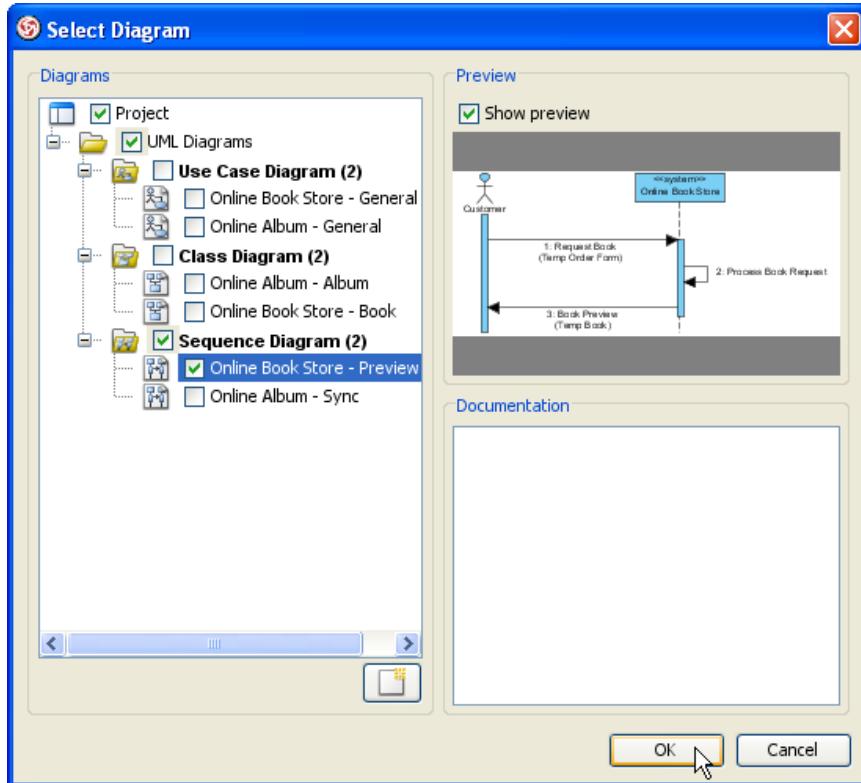


Figure 5-30 Selecting reference diagram

5. Input description of the diagram.

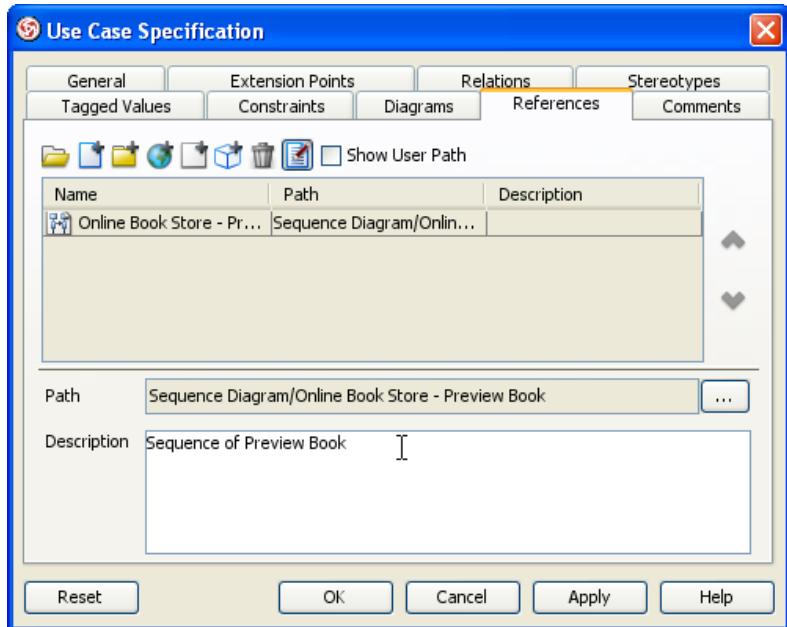


Figure 5-31 Inputting diagram reference description

6. Press Apply button to confirm the reference creation.

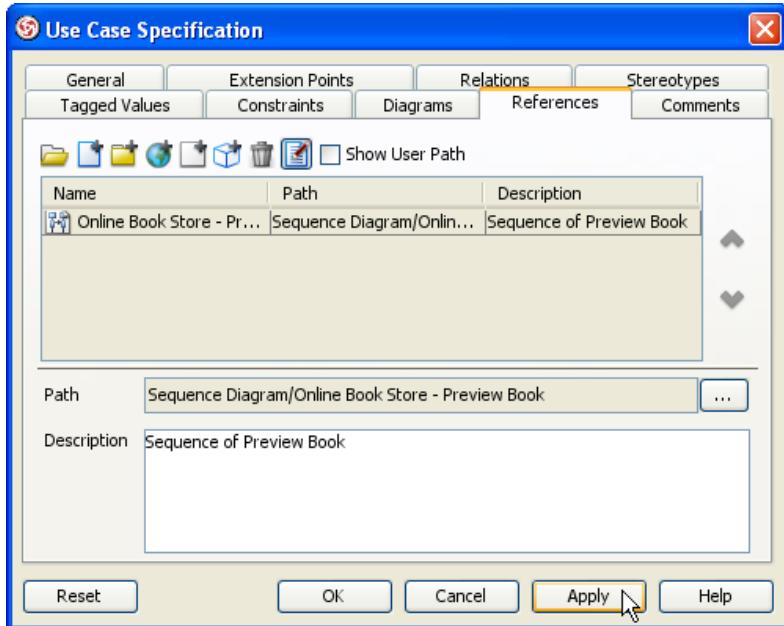


Figure 5-32 Apply diagram reference

7. Select the diagram reference to be opened.

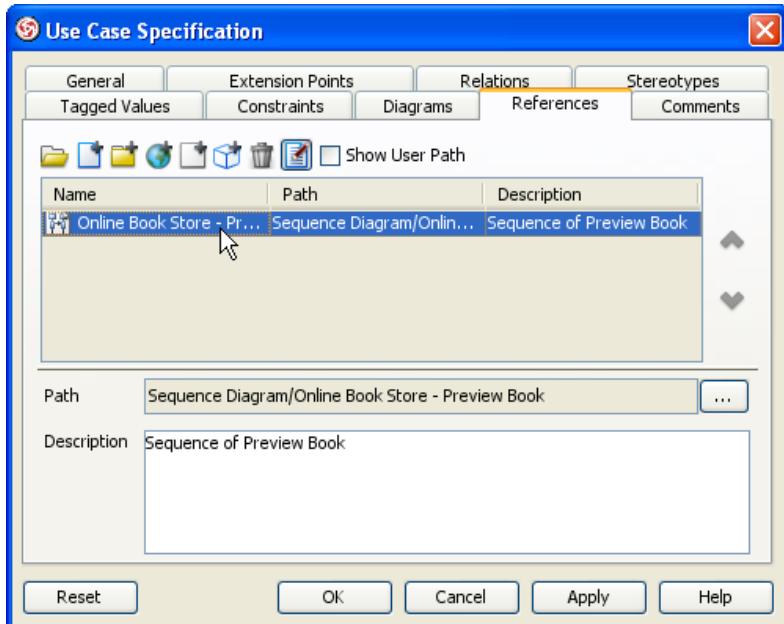


Figure 5-33 Select diagram reference

8. Press Open... button.

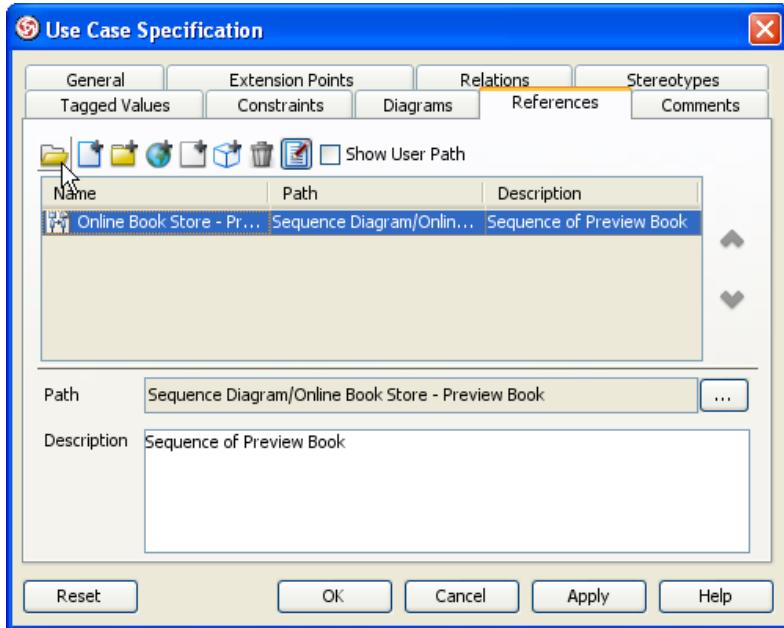


Figure 5-34 Open diagram reference

Reference to shapes

1. Mouse over shape and click on References resource.

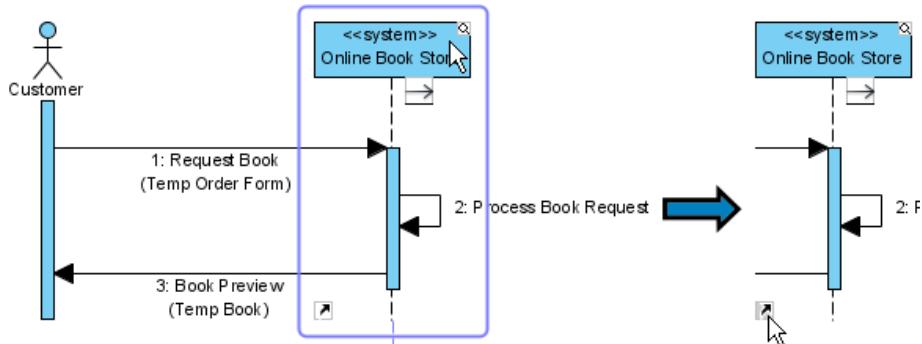


Figure 5-35 Mouse over references resource

2. Click on Add Shape... menu item.

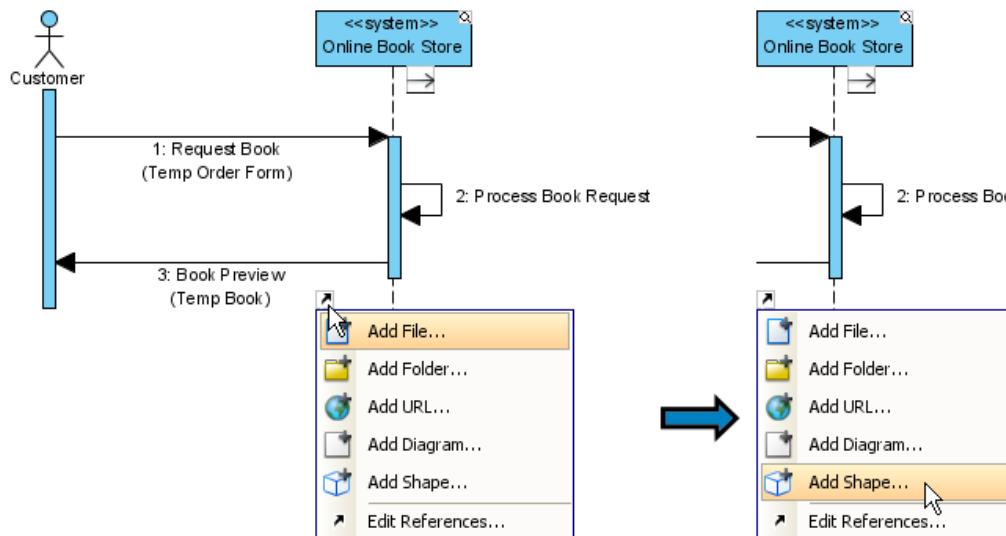


Figure 5-36 Add shape from resource

3. Select shape to be referenced.

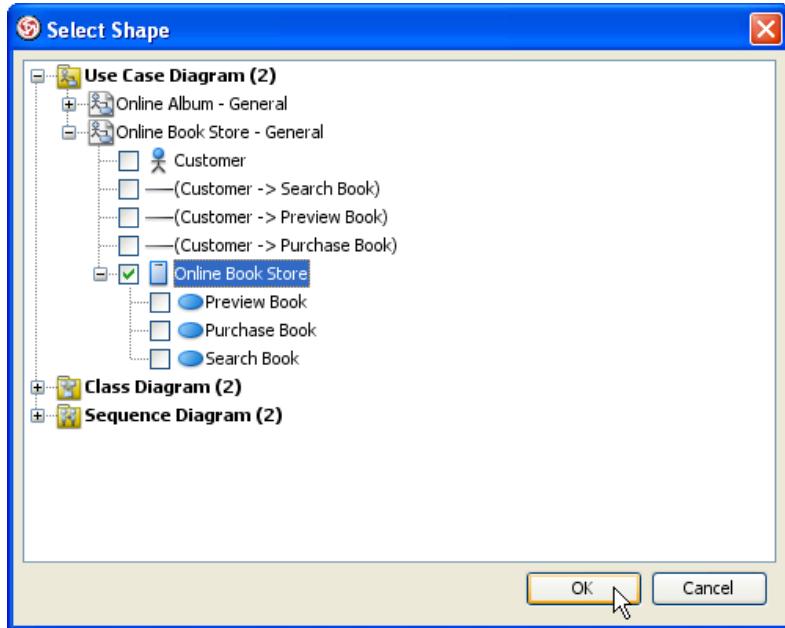


Figure 5-37 Select reference shape

4. Input description of the shape.

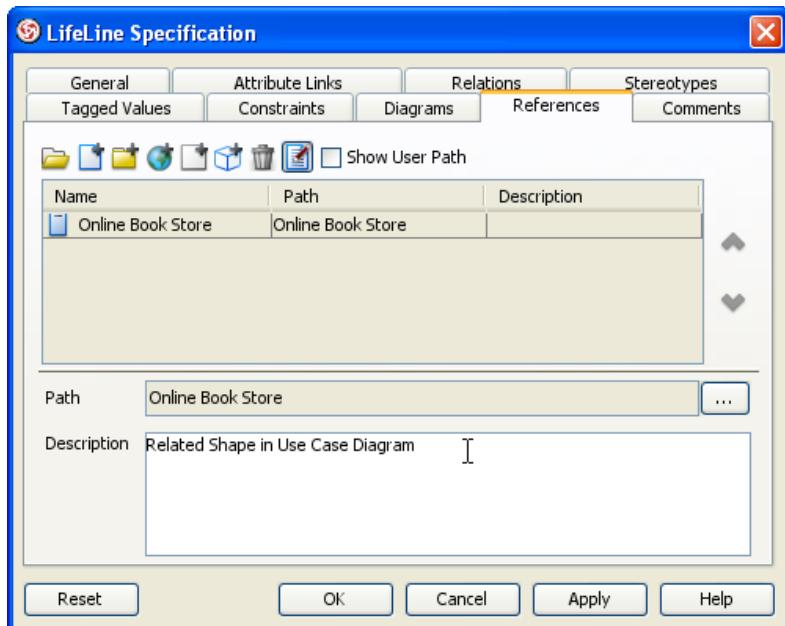


Figure 5-38 Inputting shape reference description

5. Press Apply button to confirm the reference creation.

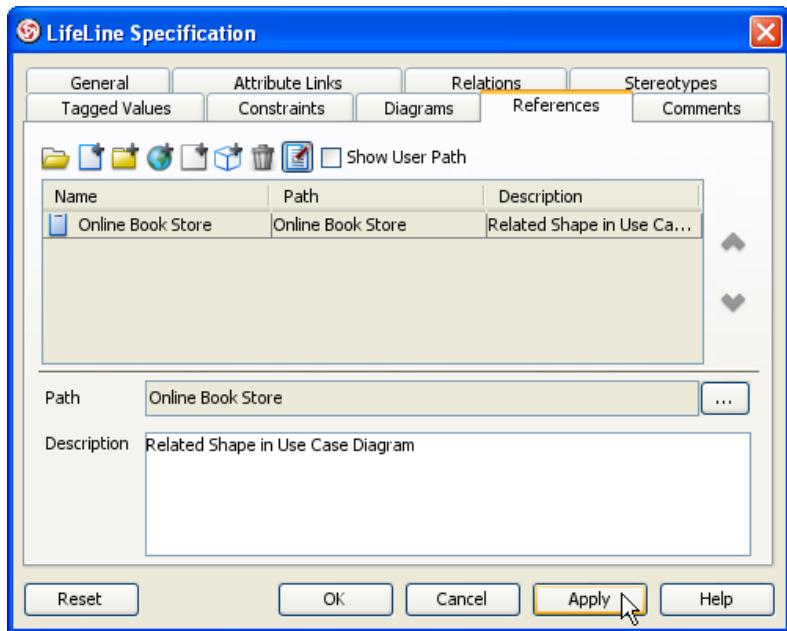


Figure 5-39 Apply folder reference

6. Mouse over shape and click on References resource.

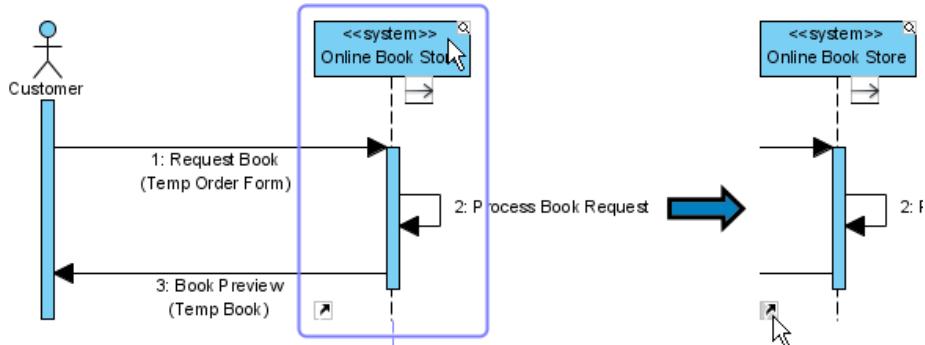


Figure 5-40 Mouse over references resource

7. Click on shape reference menu item.

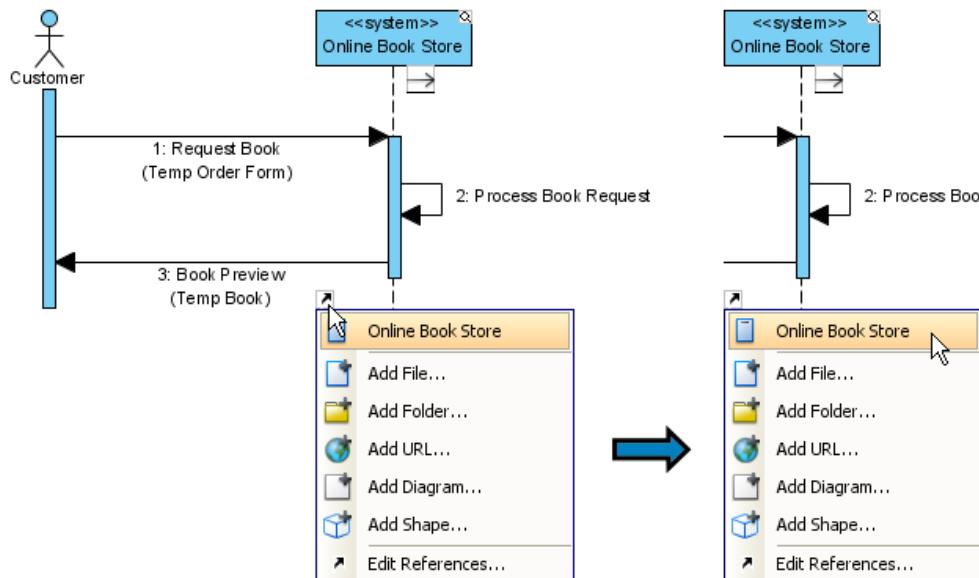


Figure 5-41 Open shape reference

Managing references

Edit references

1. Mouse over shape and click on References resource.

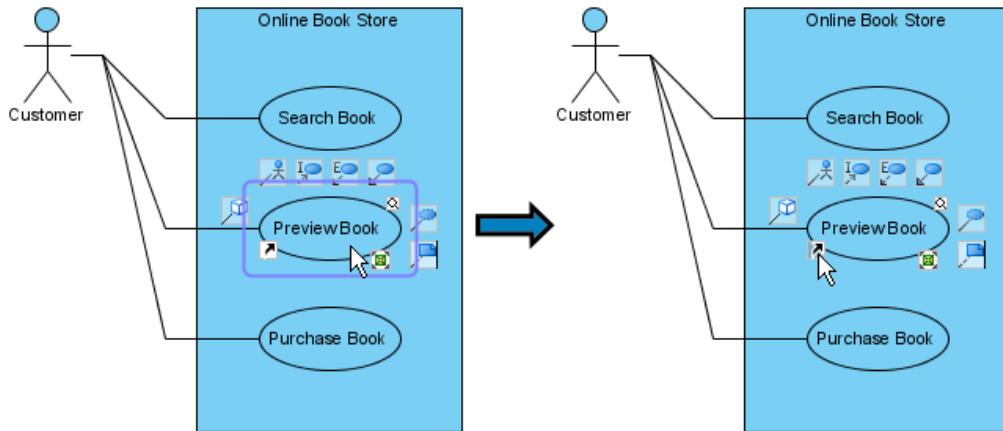


Figure 5-42 Mouse over references resource

2. Click on Edit References... menu item.

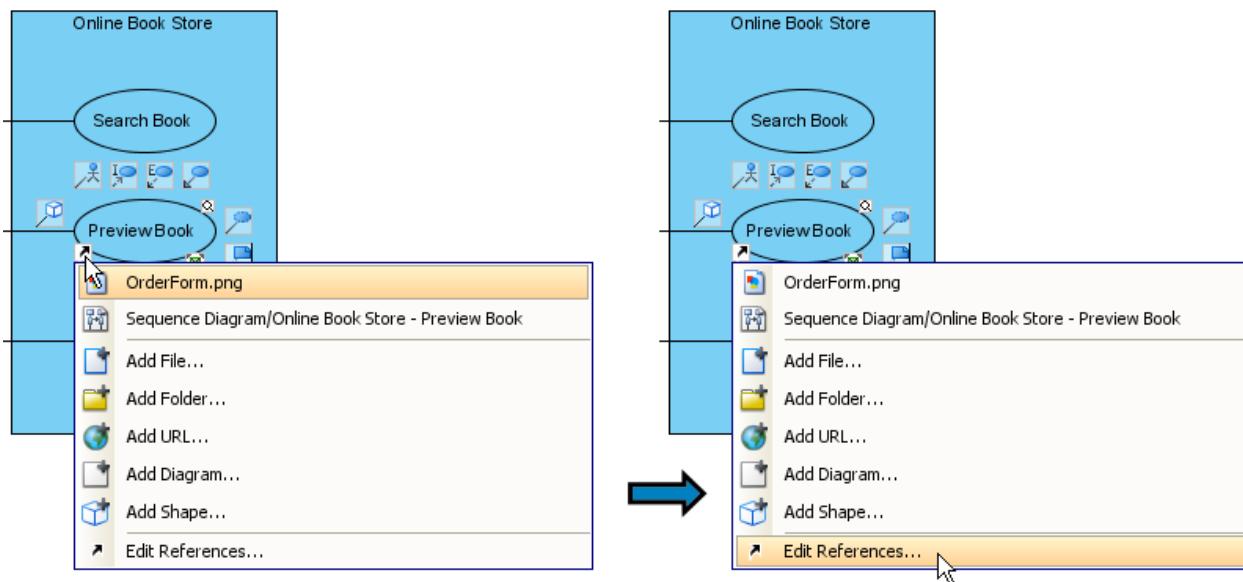


Figure 5-43 Edit references from resource

3. Double-click description of file reference.

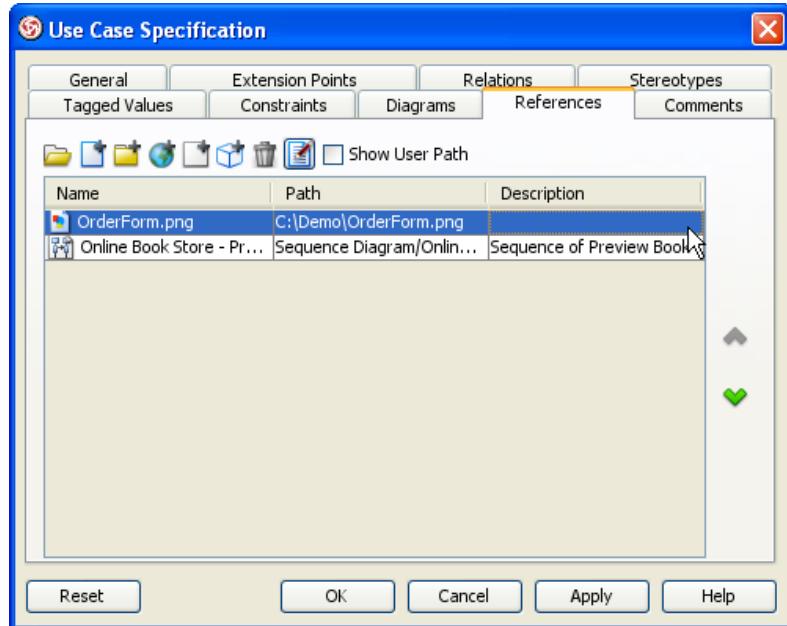


Figure 5-44 Double click file reference description

4. Input description of reference file.

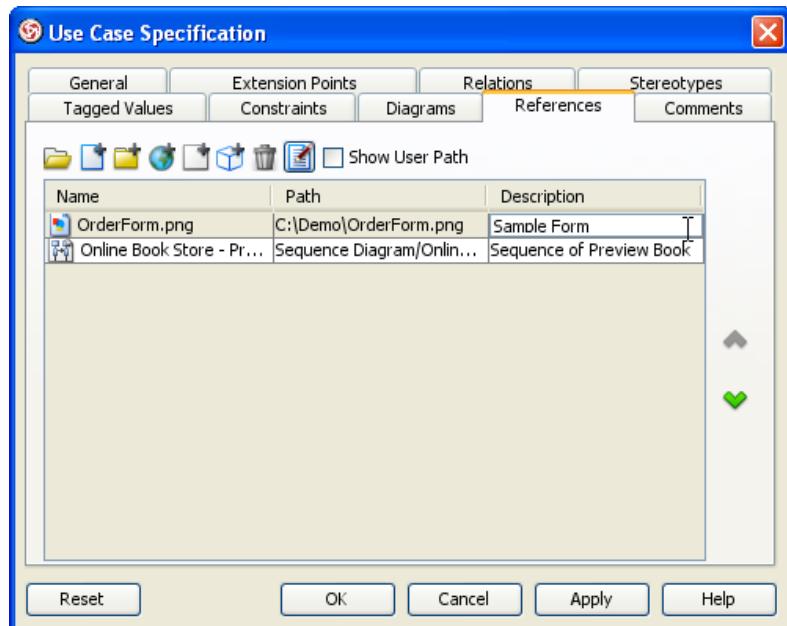


Figure 5-45 Inputting file reference description

5. Press Enter to confirm description editing.

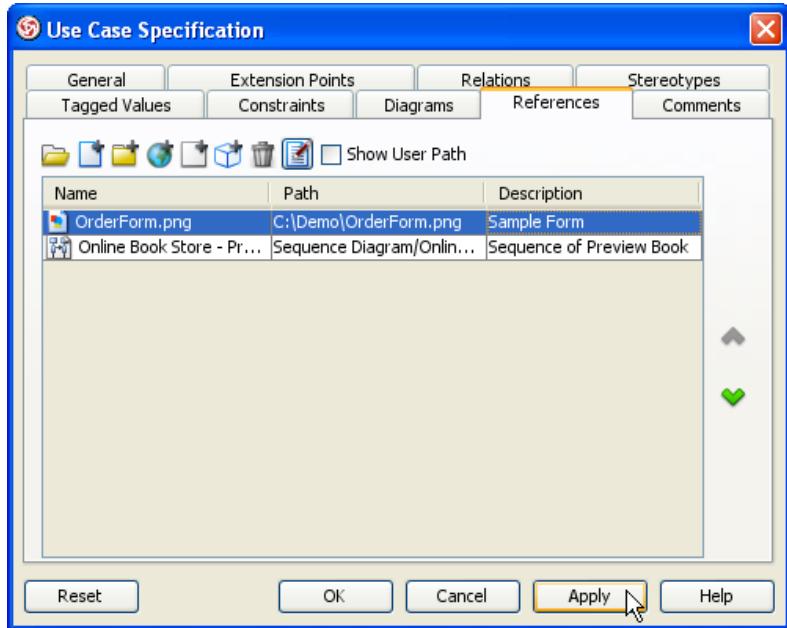


Figure 5-46 Confirming description editing

Add reference

1. Mouse over shape and click on References resource.

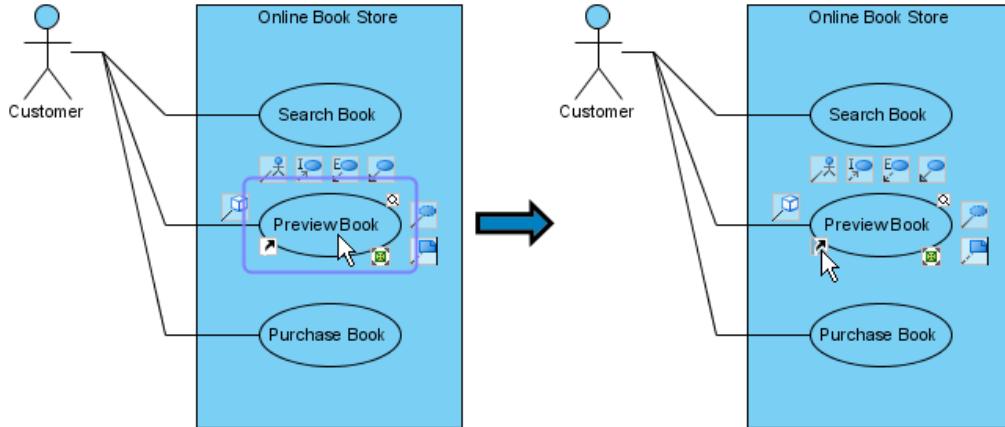


Figure 5-47 Mouse over references resource

2. Click on Add URL... menu item.

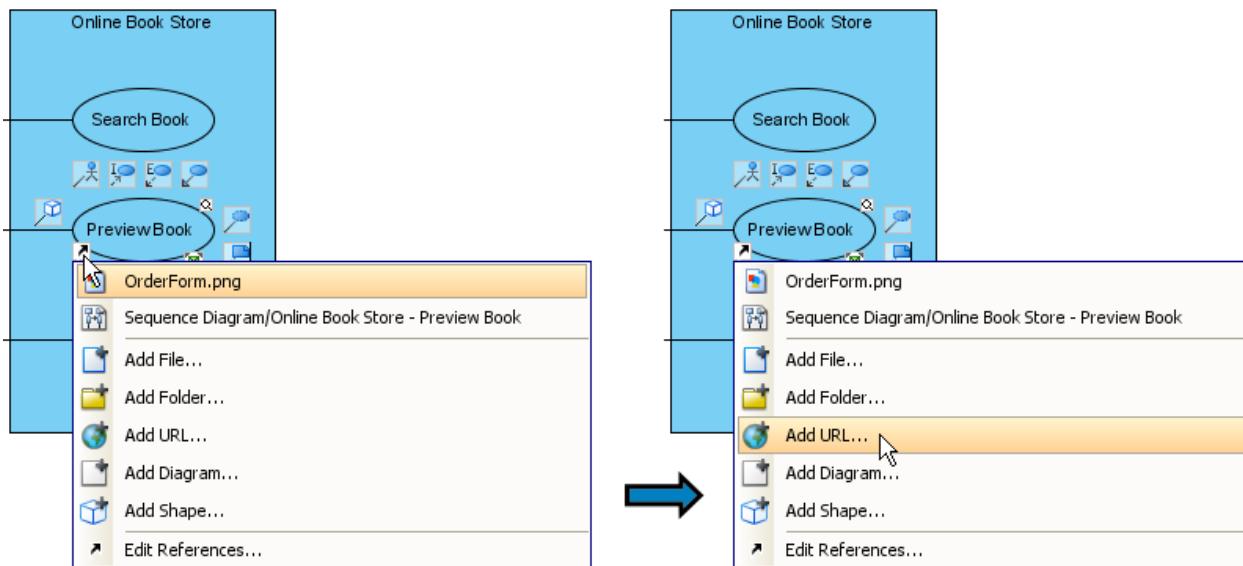


Figure 5-48 Add URL from resource

3. Input path of the URL.

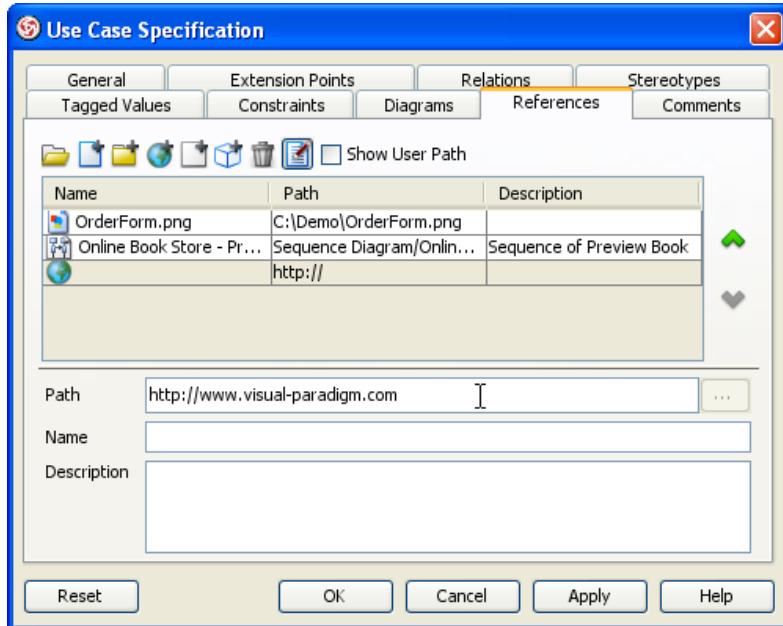


Figure 5-49 Inputting URL reference path

4. Press Apply button to confirm the reference creation.

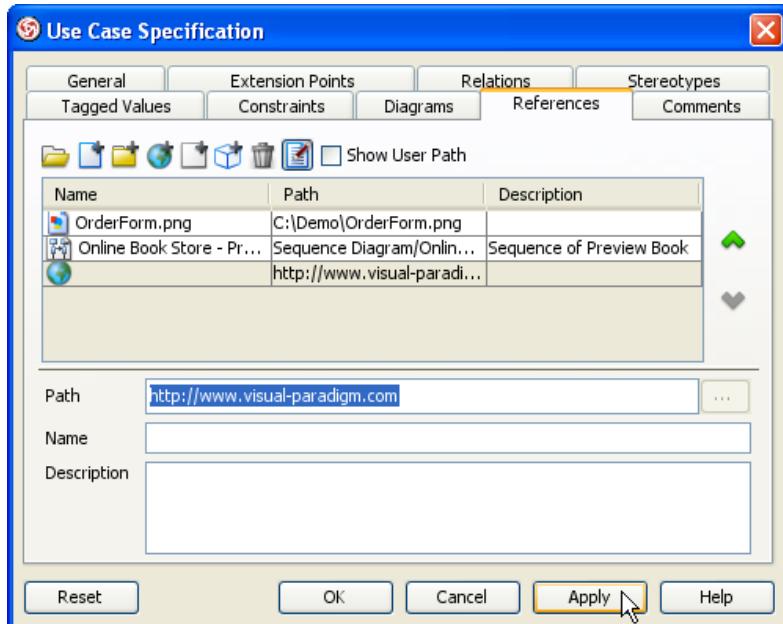


Figure 5-50 Apply URL reference

Remove reference

1. Mouse over shape and click on References resource.

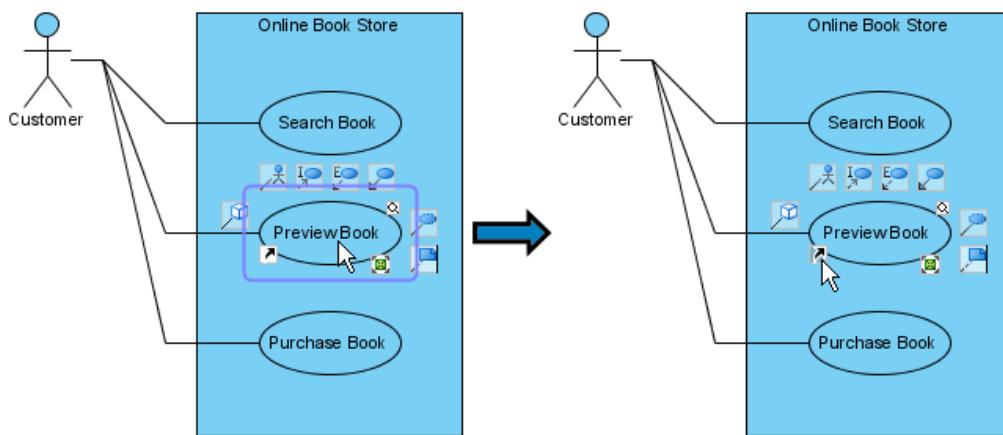


Figure 5-51 Mouse over references resource

2. Click on Edit References... menu item.

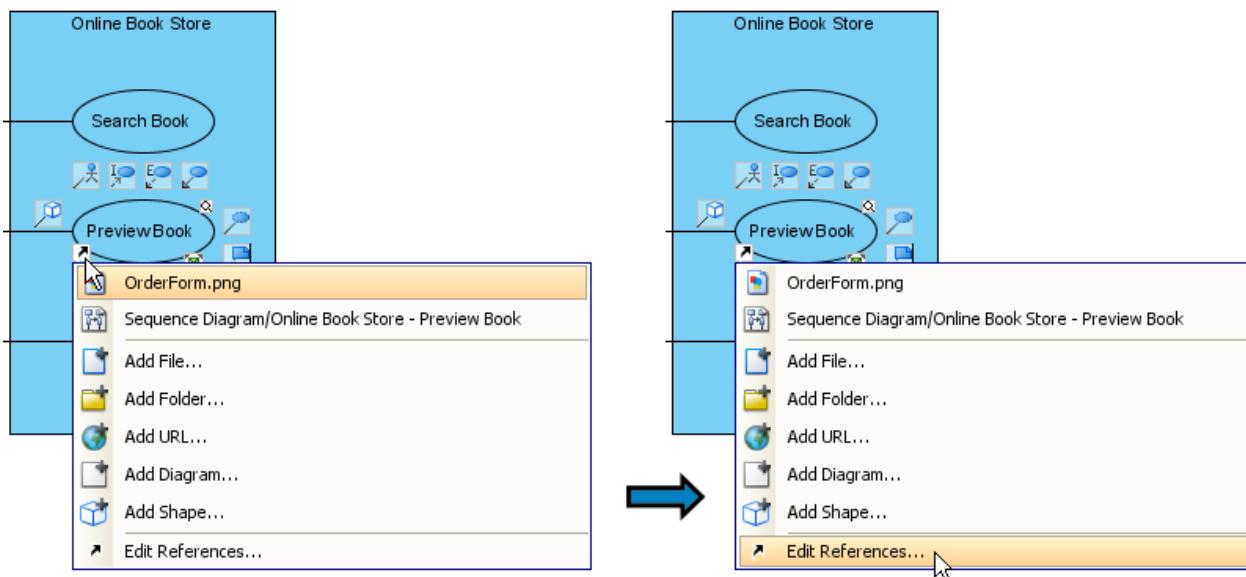


Figure 5-52 Edit references from resource

3. Select the diagram reference to be removed.

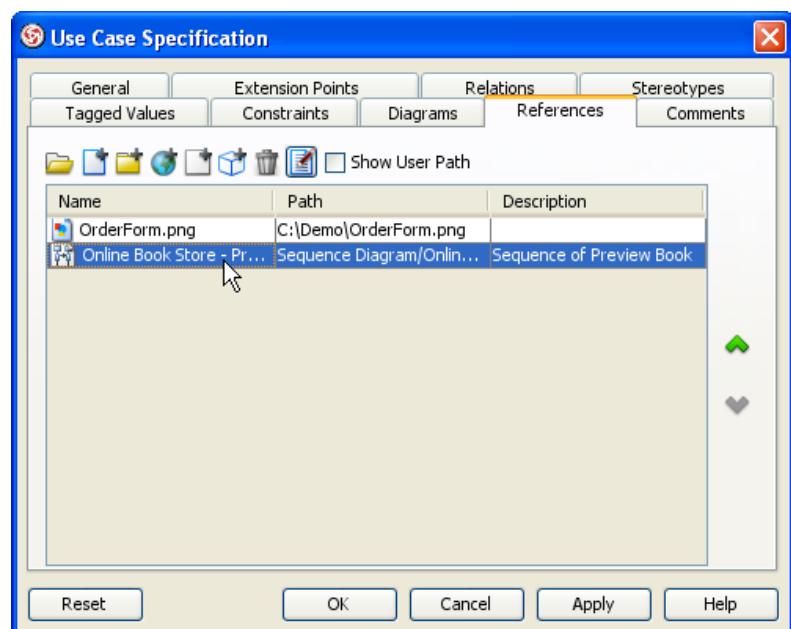


Figure 5-53 Select diagram reference

4. Press Remove button.

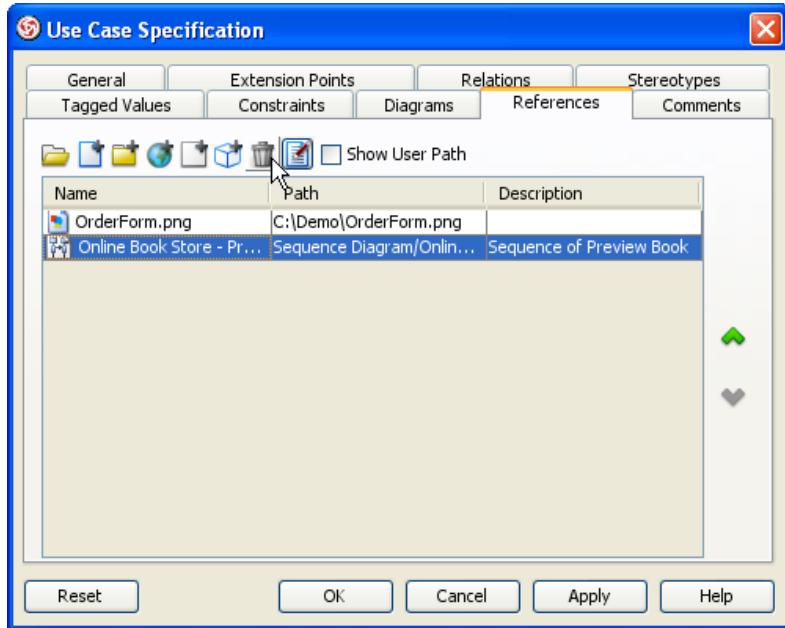


Figure 5-54 Remove diagram reference

5. Press Yes to confirm removing diagram reference.

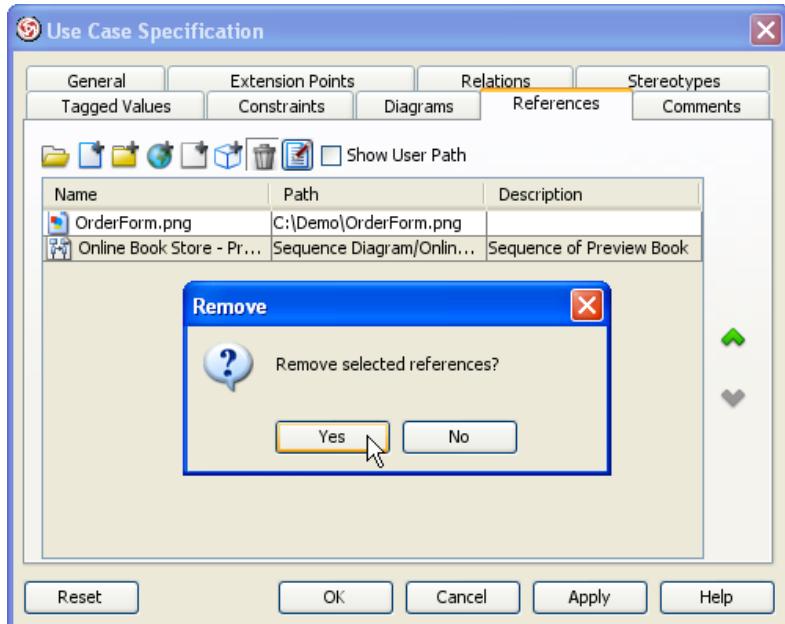


Figure 5-55 Confirm removing diagram reference

6. Press Apply button to confirm the reference removal.

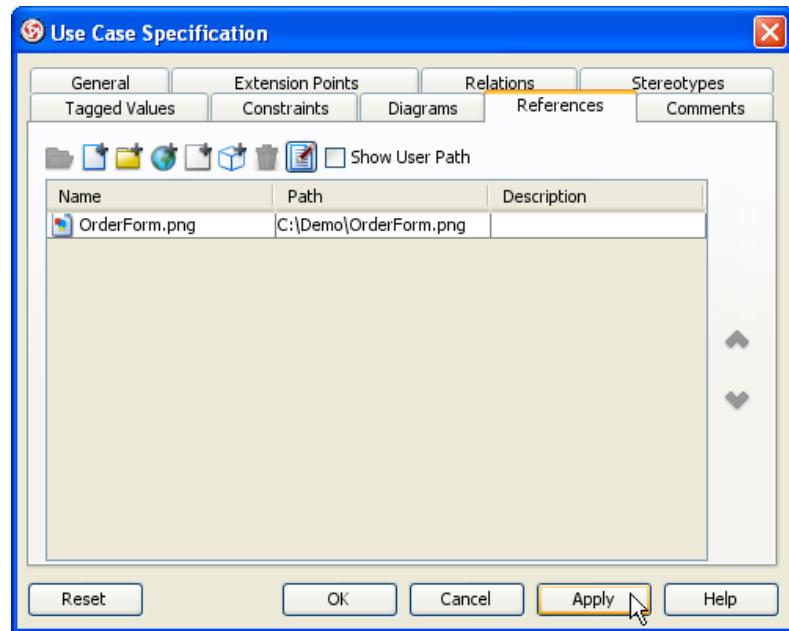


Figure 5-56 Apply diagram reference removal

Elaborate model element with sub diagram

Creating sub diagram

1. Mouse over shape and click on Sub Diagrams resource.

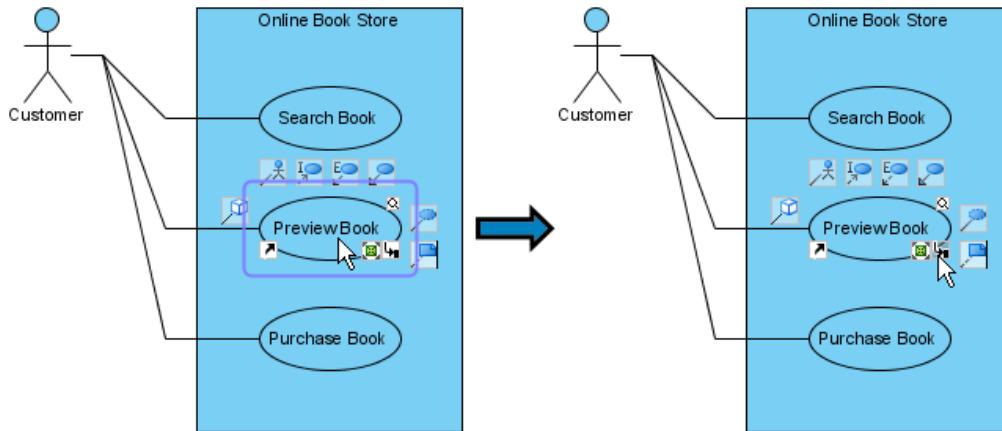


Figure 5-57 Mouse over sub-diagrams resource

2. Click on Add > Sequence Diagram menu item.

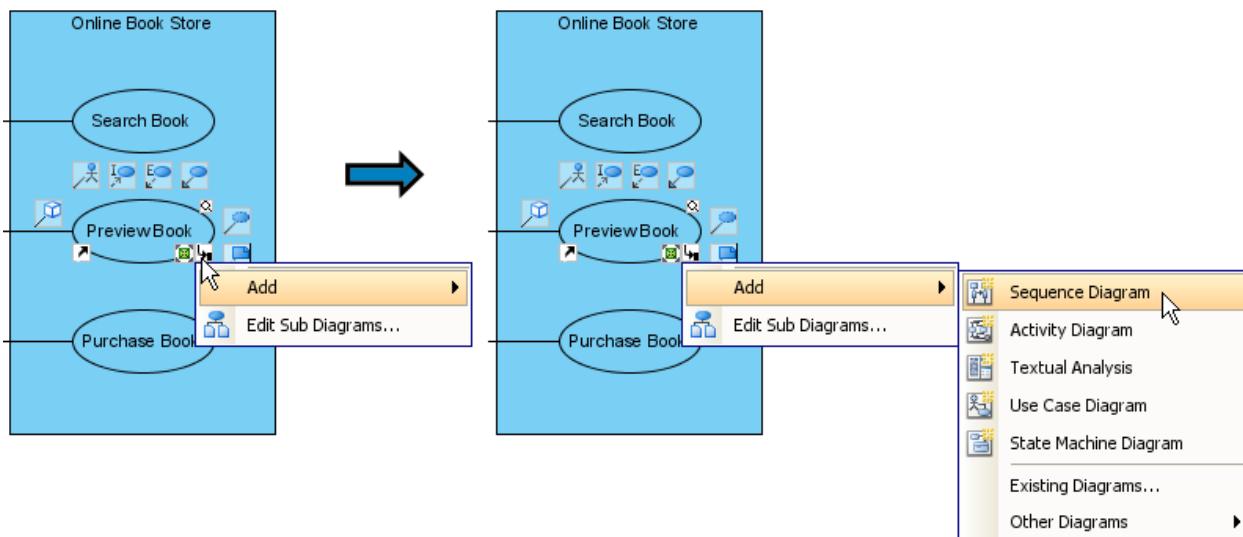


Figure 5-58 Add sub diagram from resource

Adding existing diagram as sub diagram

1. Right click on shape.

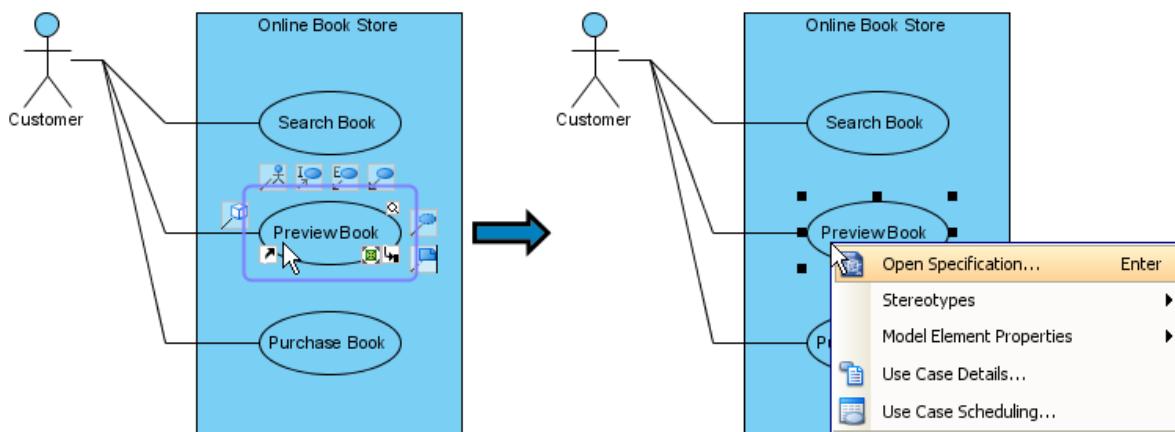


Figure 5-59 Show popup of shape

2. Click on Sub Diagrams > Add Existing Diagrams... menu item.

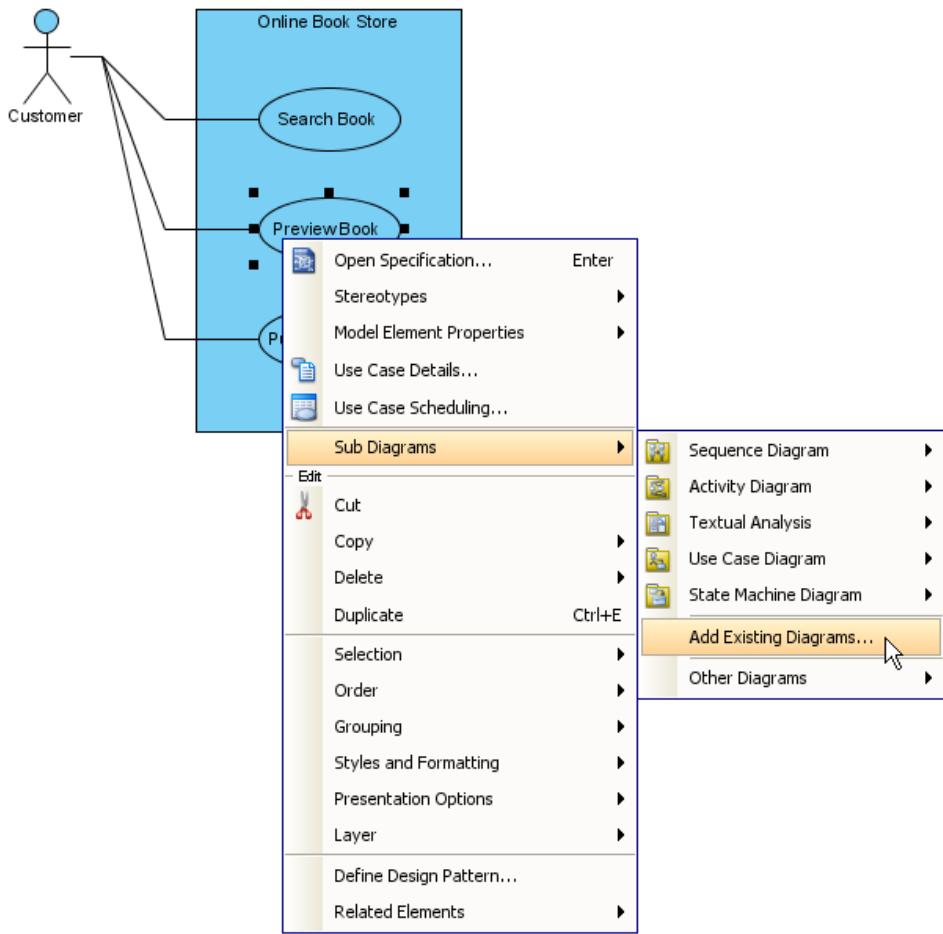


Figure 5-60 Adding existing diagram as sub diagram

3. Select sub diagram.

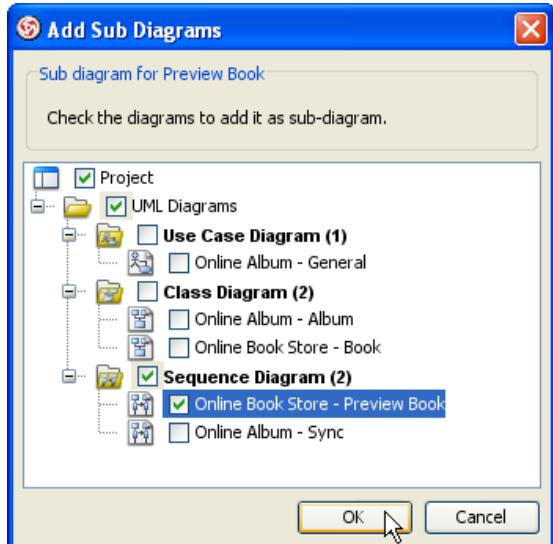


Figure 5-61 Selecting sub diagram

Remove sub diagram

1. Mouse over shape and click on Sub Diagrams resource.

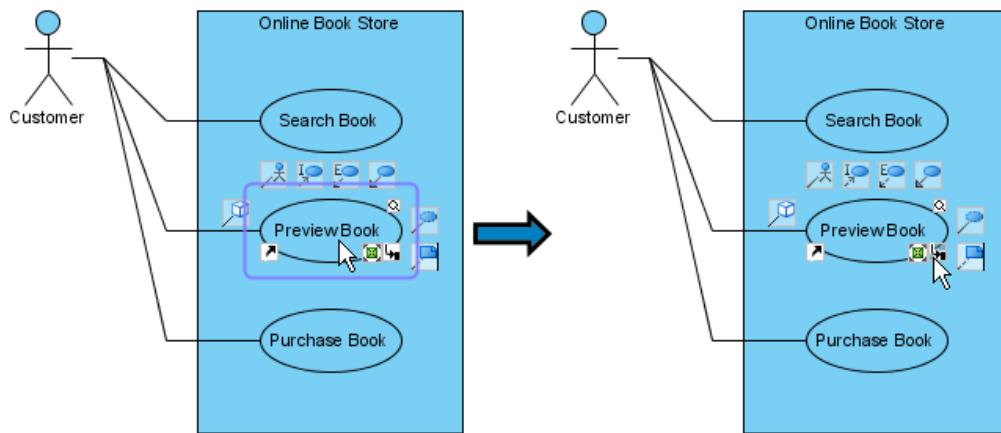


Figure 5-62 Mouse over sub diagrams resource

2. Click on Edit Sub Diagrams... menu item.

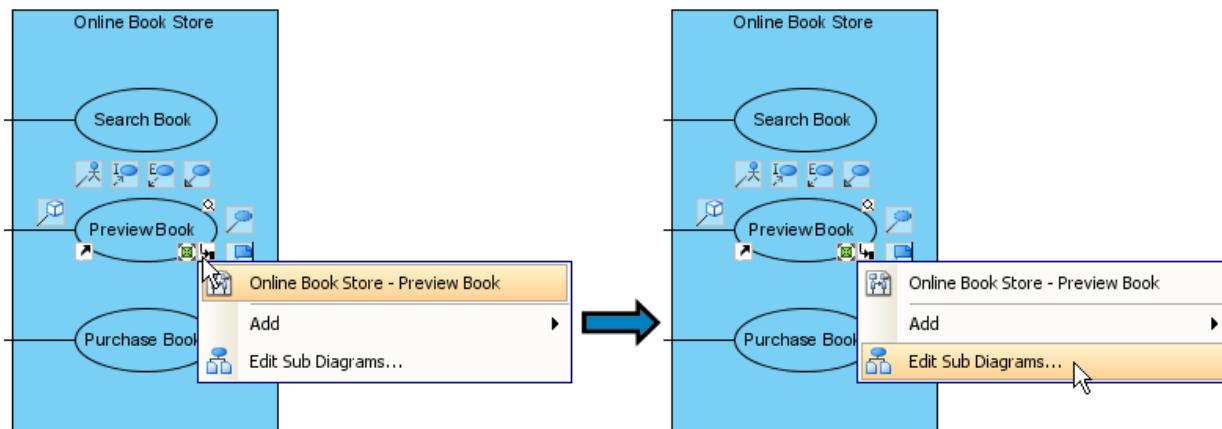


Figure 5-63 Edit sub diagrams from resource

3. Select sub diagram to be removed.

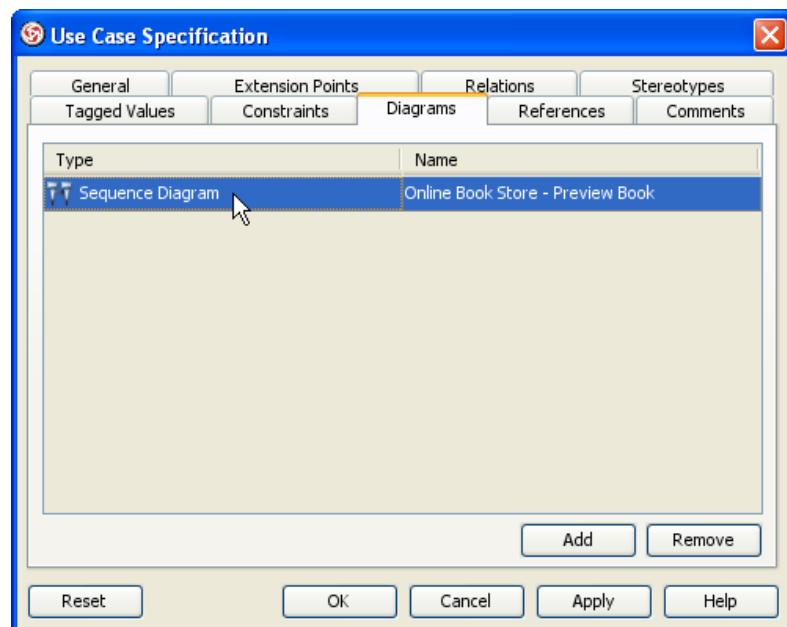


Figure 5-64 Select sub diagram

4. Press Remove button.

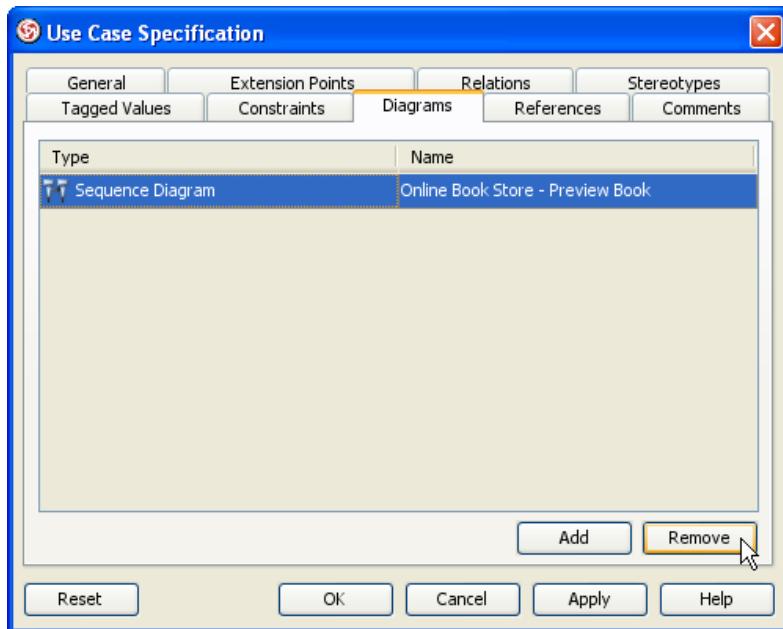


Figure 5-65 Remove sub diagram

5. Press Yes to confirm removing sub diagram.

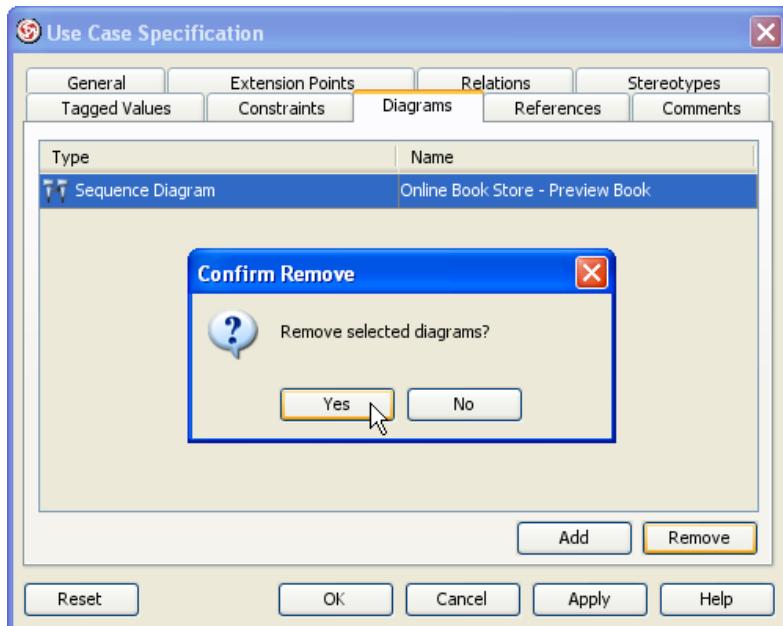


Figure 5-66 Confirm removing sub diagram

6. Press Apply button to confirm the sub diagram removal.

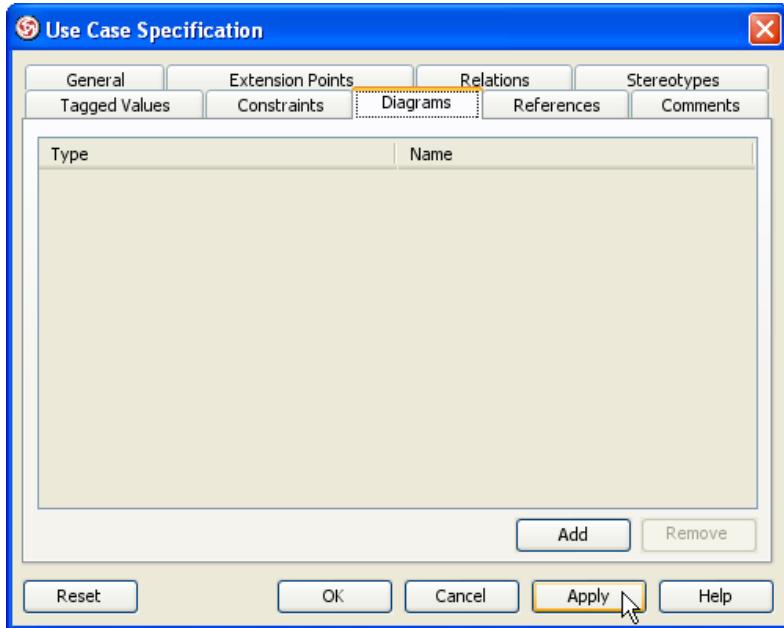


Figure 5-67 Apply sub diagram removal

Sub diagram and it's diagram element

Adding sub diagram will add all model element as child.

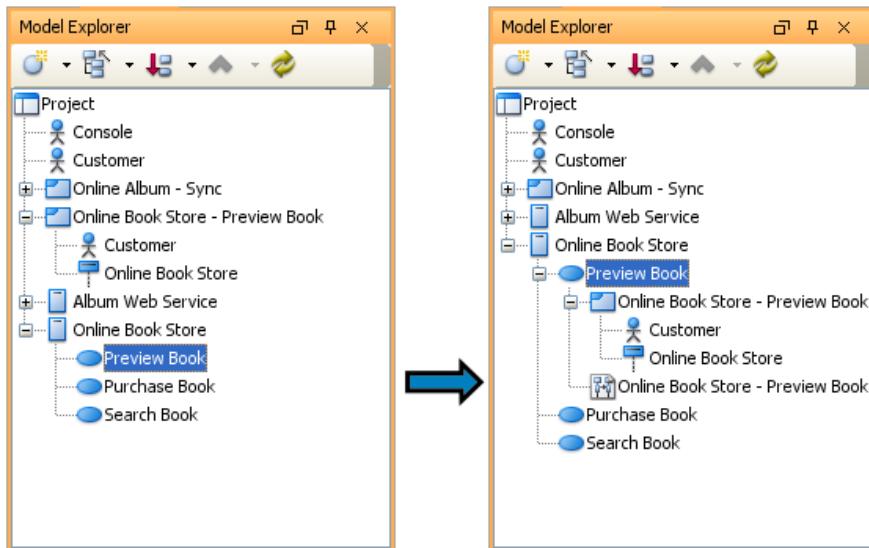


Figure 5-68 Different after added sub diagram

Showing sub diagrams and reference indicators

1. Right click on diagram and click on Presentation Options > Always Show Reference and Sub Diagram Resource menu item.

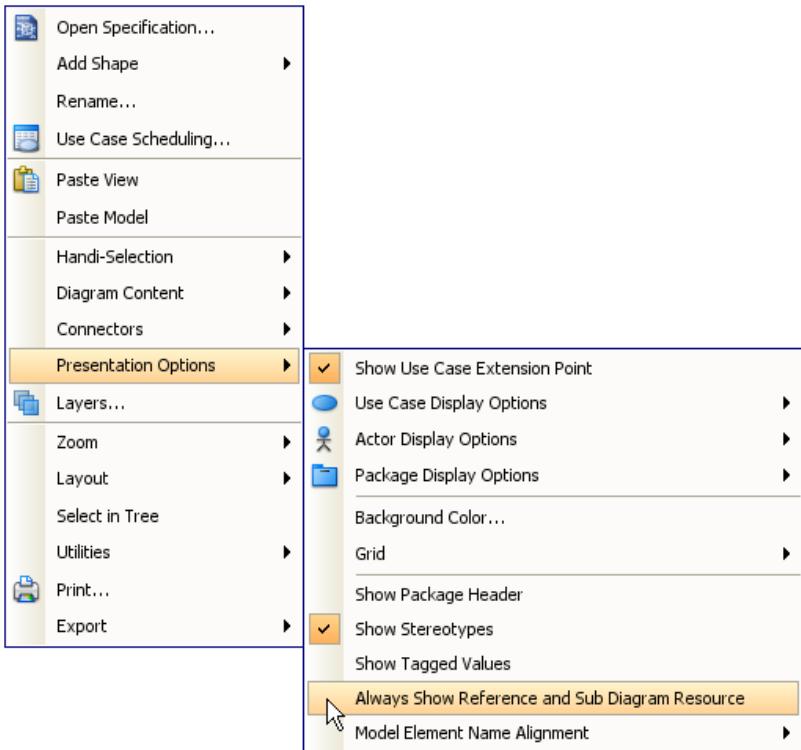


Figure 5-69 Show popup of diagram

2. References resource and Sub Diagrams resources are shown.

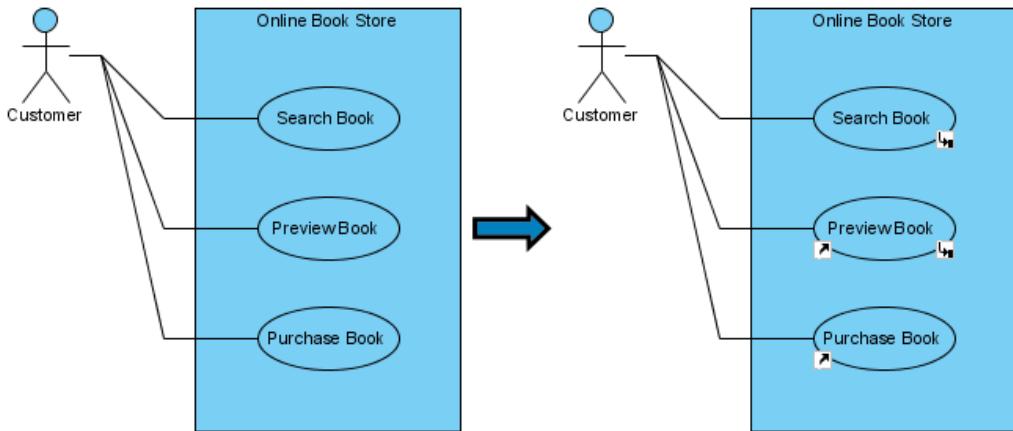


Figure 5-70 Show references resource and sub diagrams resource

Changing diagram elements styles

You can change the diagram element's style in the **Format** dialog. To open the Format dialog, right click on the shape and select **Styles and Formatting... from the popup menu.**

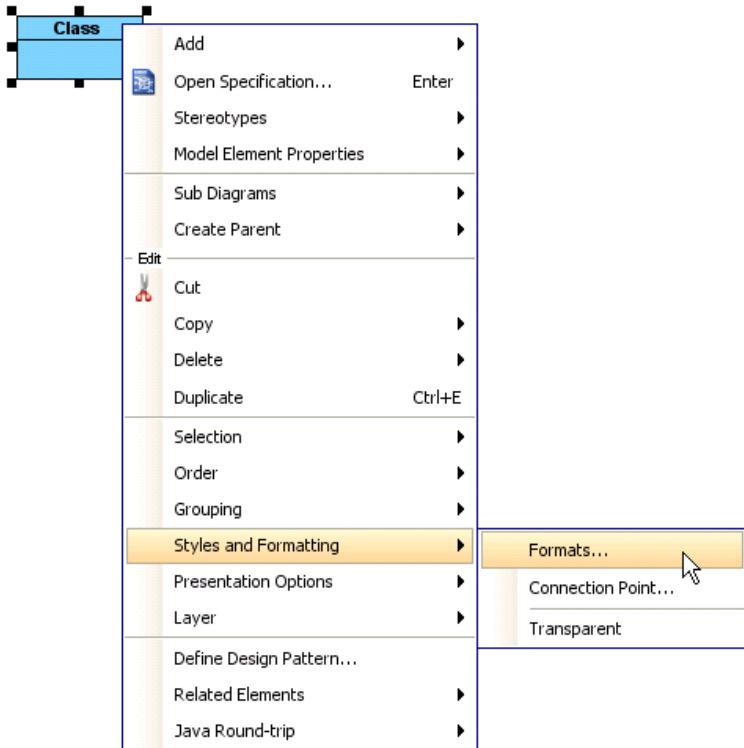


Figure 6-1 Open Format dialog from popup menu

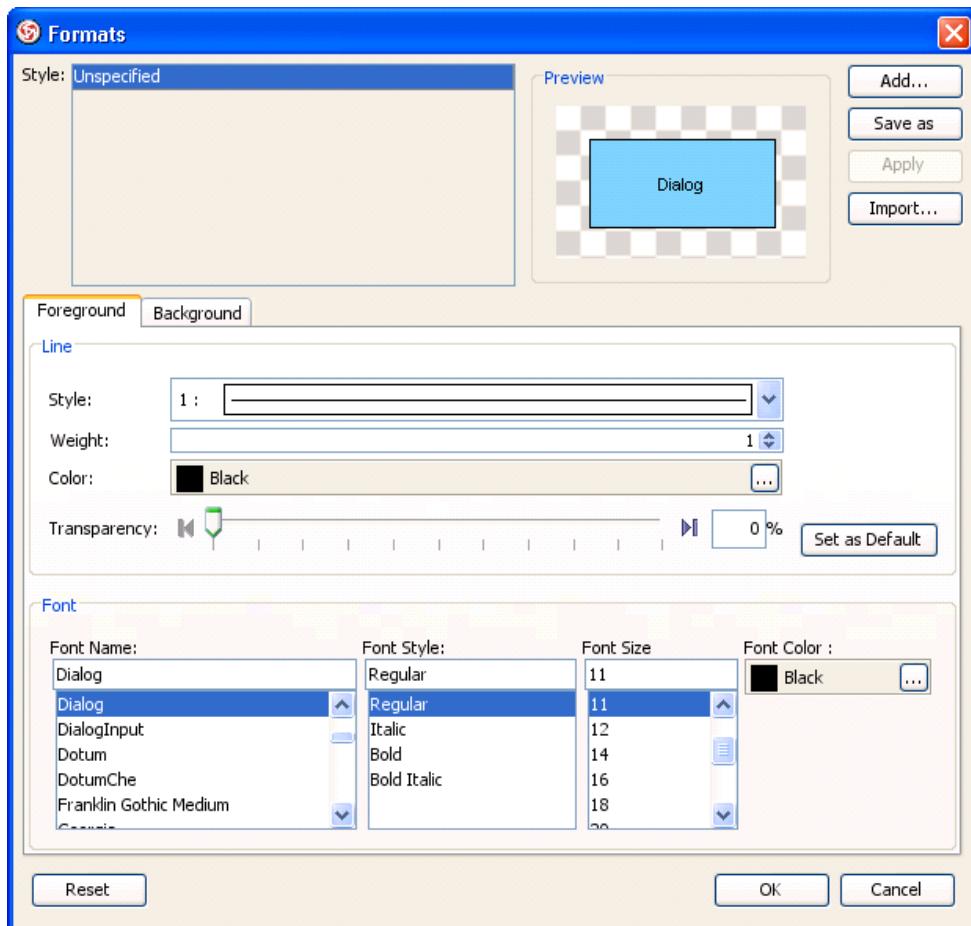


Figure 6-2 Format dialog

You can change the following settings from the Format dialog:

- Changing Shapes Foreground Style

- Changing Shapes Font Style
- Changing Shapes Background Style

Changing shapes foreground style

In the **Format** dialog, you can change the foreground style in the **Font** section.

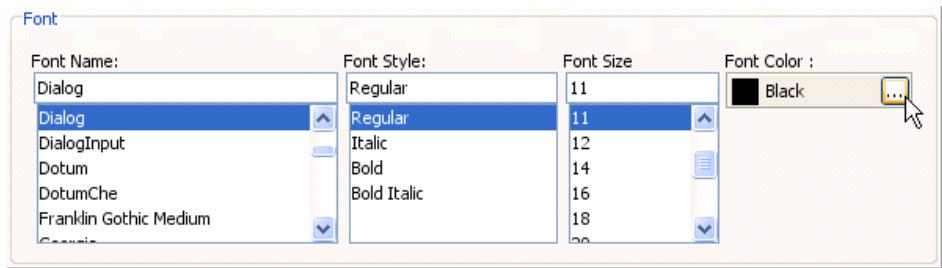


Figure 6-3 Font section

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).



Figure 6-4 Color pane

If you want to specify a custom color, you can switch to the **Custom** pane

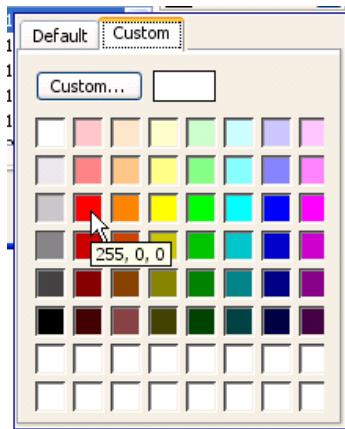


Figure 6-5 Custom color pane

After changing the font color, the preview will update automatically.



Figure 6-6 Change font color preview

Changing shapes font style

In the Font section, you can also change the font family, style and size.

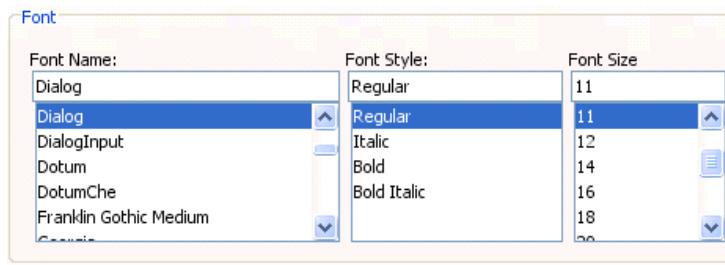


Figure 6-7 Changing font style

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.

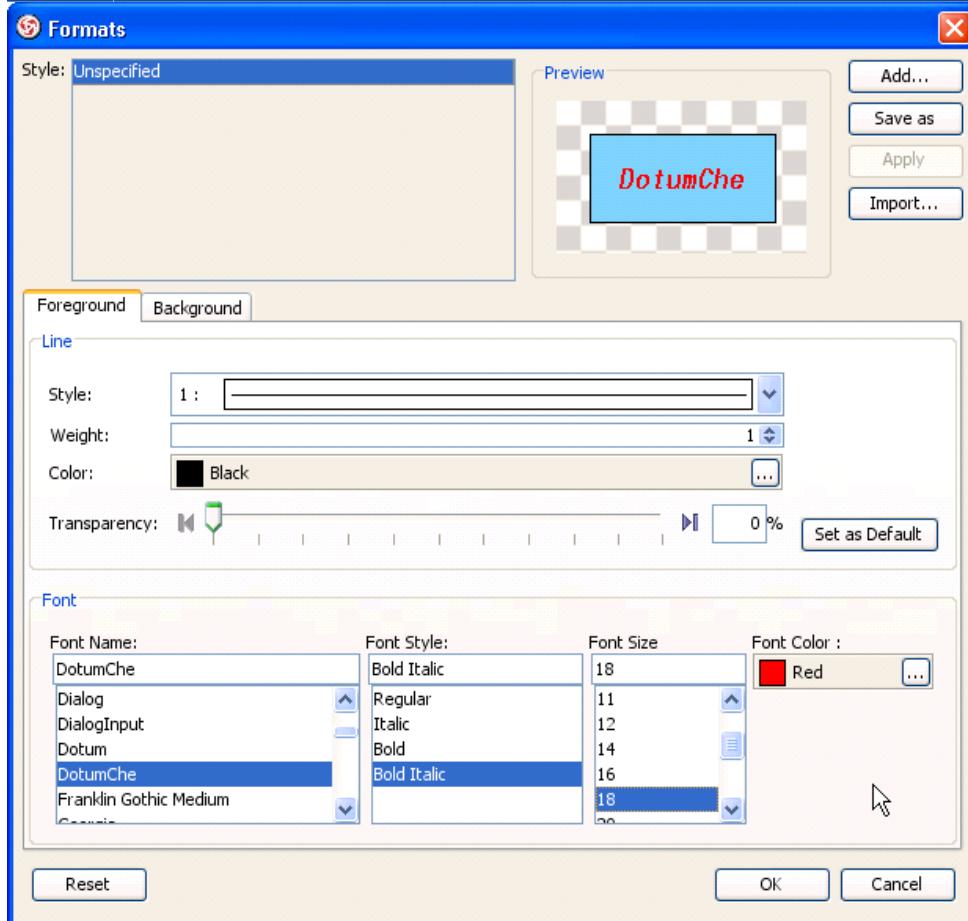


Figure 6-8 Change font style

The Preview pane displays the selected font format.

Changing shapes background style

You can click on the **Background** tab to custom the background style.

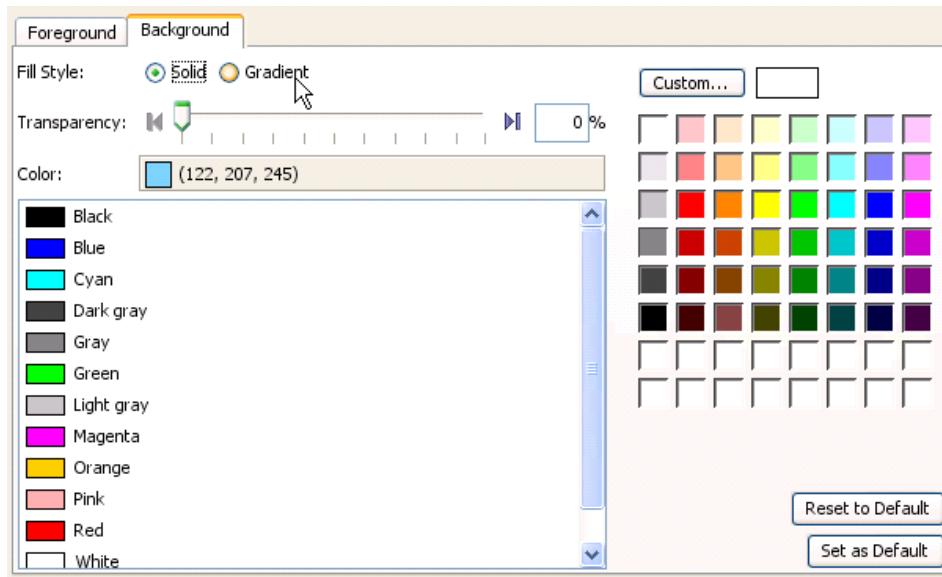


Figure 6-9 Changing Background style

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	To save the current fill color as the default fill color for new shapes, click the Set as Default button.
Reset to Default	To 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.

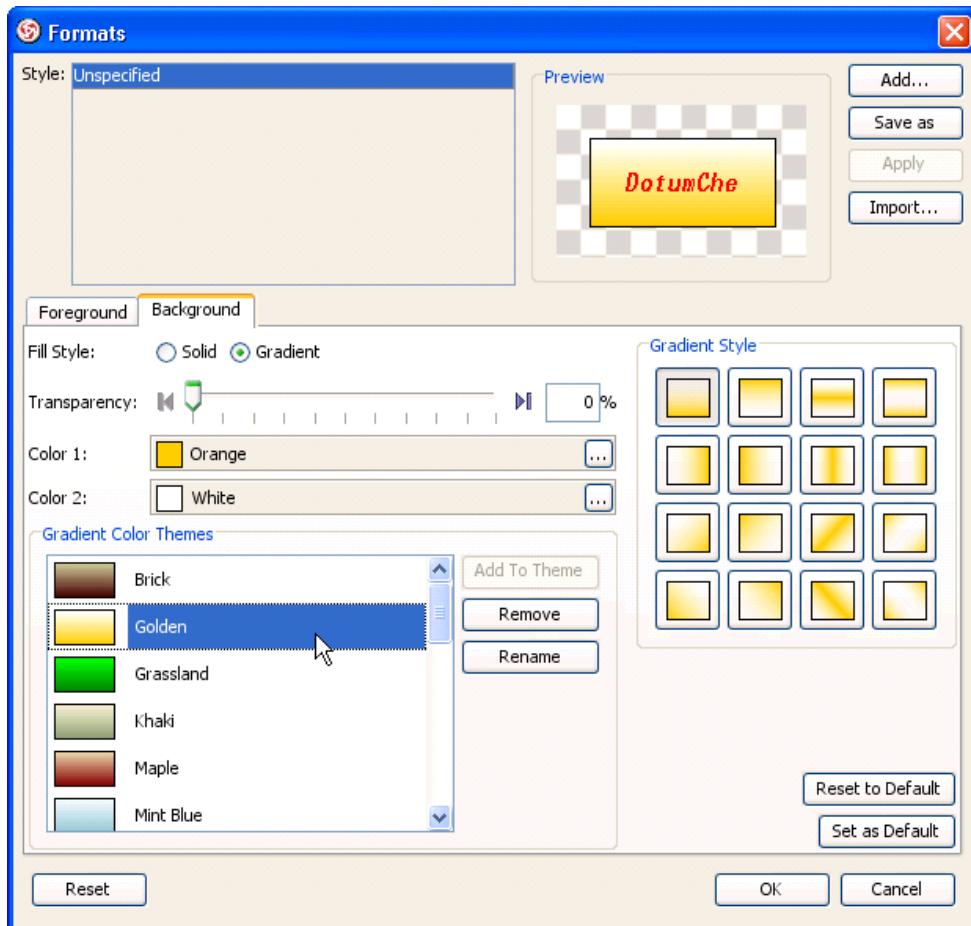


Figure 6-10 Select Gradient Fill style

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 to 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 buttons in the Gradient Style pane. To select a gradient style to use click on one of the styles.

After you done, click **OK** to apply to the selected shape.



Figure 6-11 Change shape background style result

Changing connector line style

To change the connector line style via **Property table**:

1. Select the connector.
2. Find the **Foreground** row in the Property table.

3. Click the ... button.

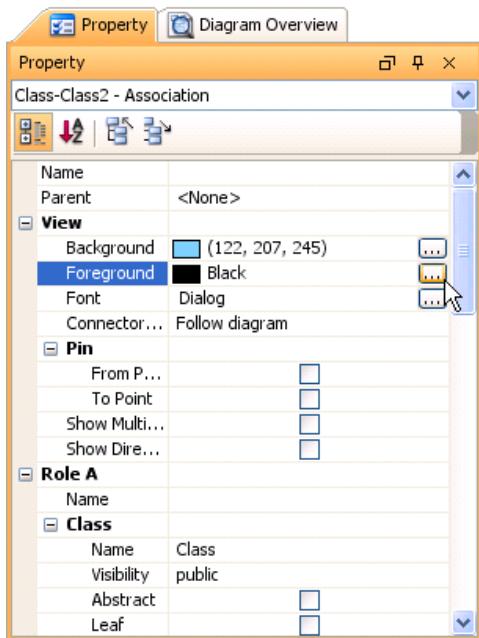


Figure 6-12 Change connector line style on properties table

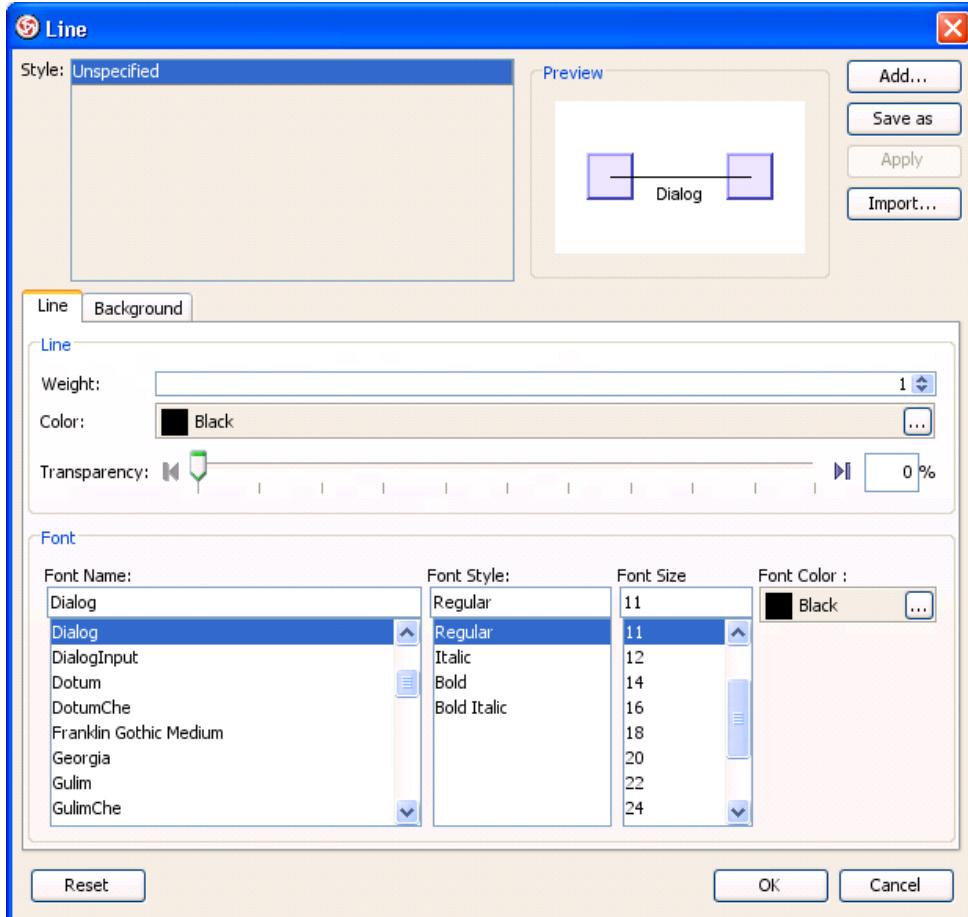


Figure 6-13 Line style dialog

You can format the line style in the line section. It allows you to adjust weight (thickness), color and transparency.

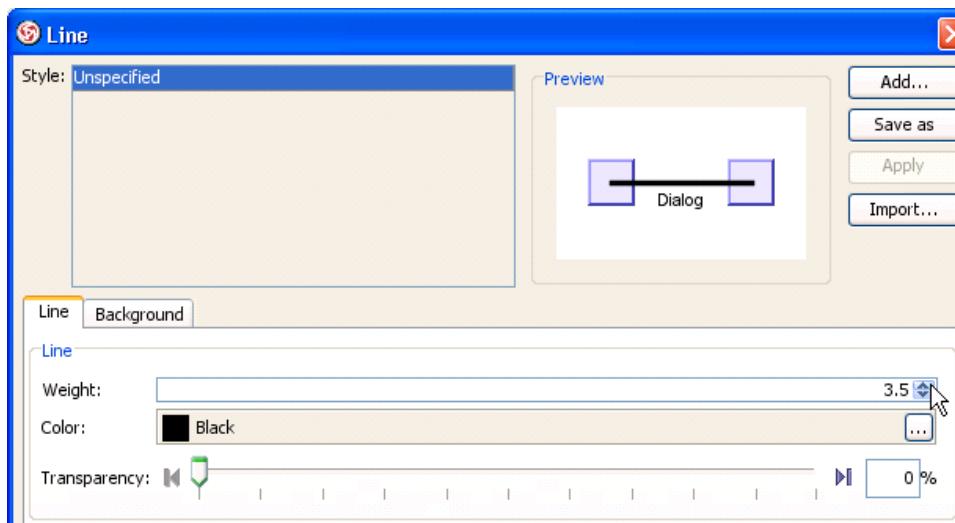


Figure 6-14 Line section

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 connector background style

Upon selecting the **Background** tab, it allows you to specify the background fill style of the line.

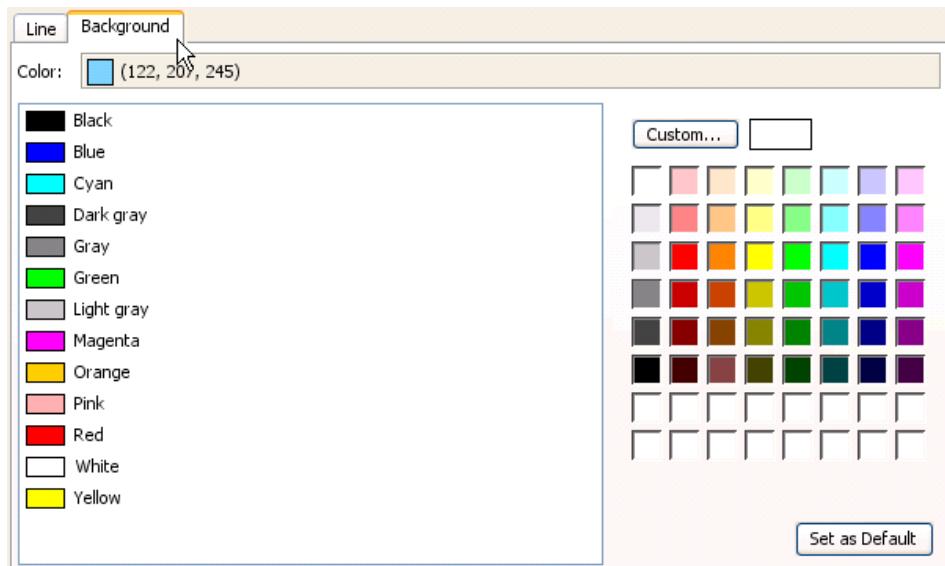


Figure 6-15 Change connector background style

Click OK to apply the settings.



Figure 6-16 Change connector style result

Manage and apply styles

VP-UML allows you to define your own style and apply to other shapes by simple steps.

Adding style

To open the Style dialog, Select **View > Styles...** from the main menu.

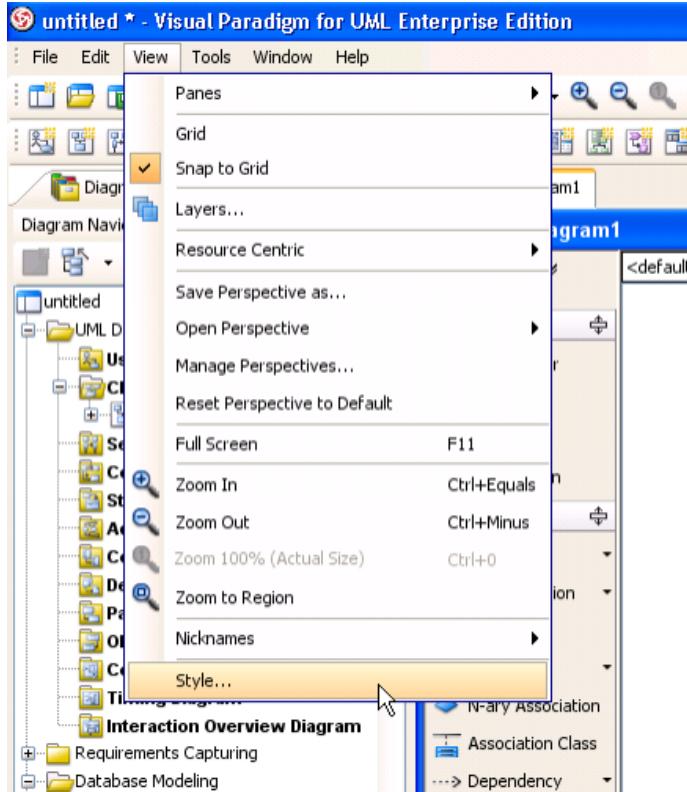


Figure 6-17 Open Style dialog

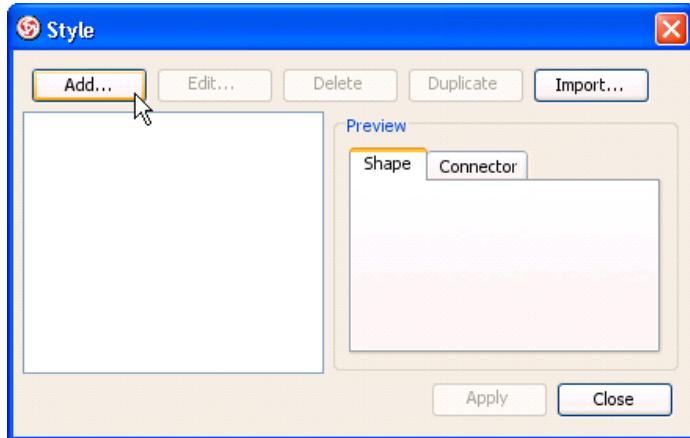


Figure 6-18 Style dialog

In the Style dialog, click **Add...** to create and edit a new style.

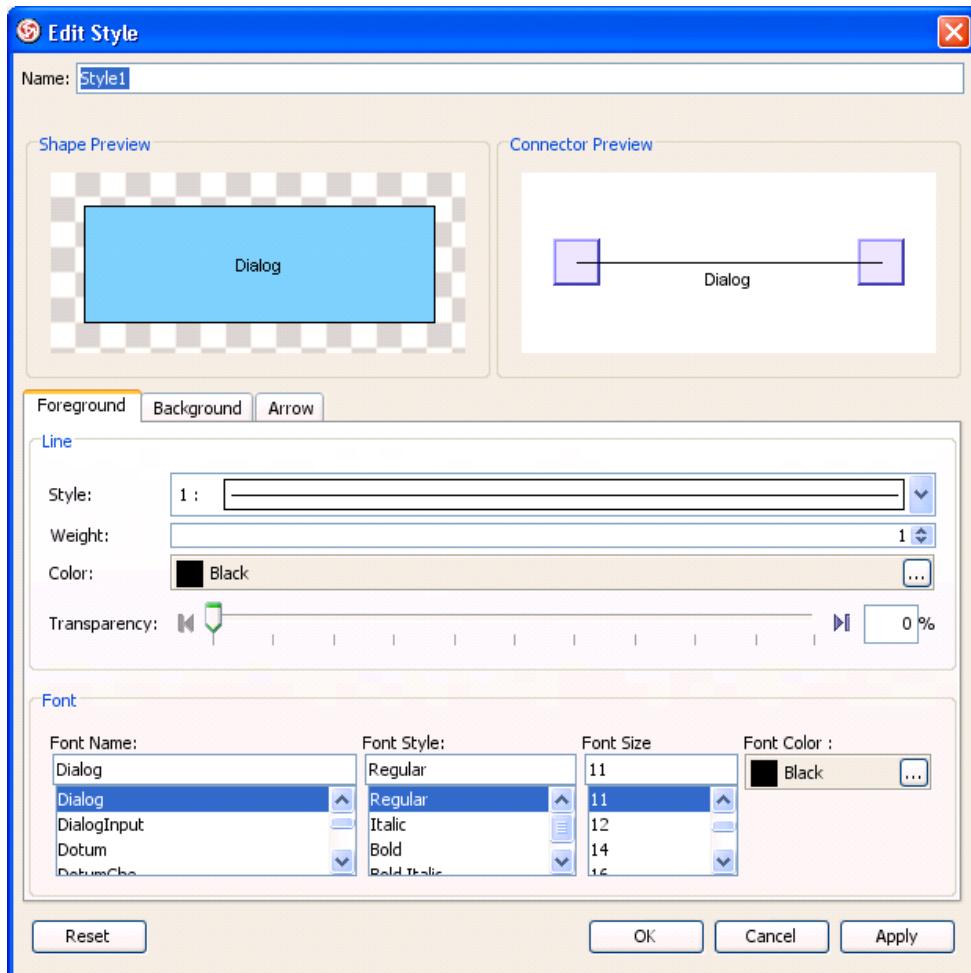


Figure 6-19 Edit Style dialog

In the **Edit Style** dialog, you can change the followings setting of the style:

- Name

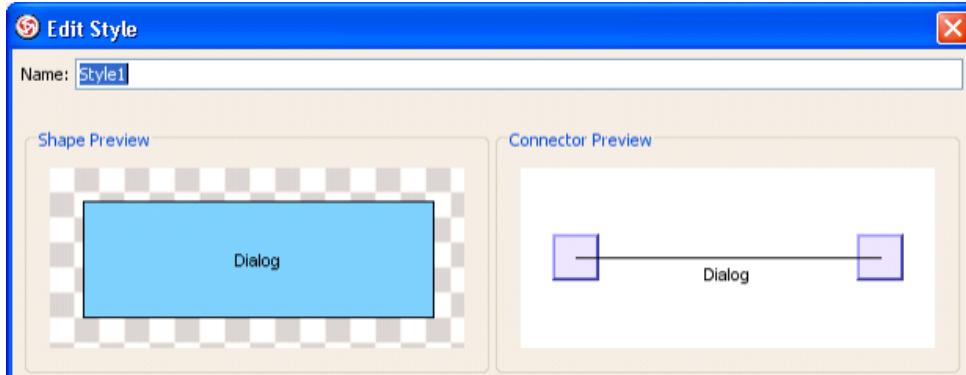


Figure 6-20 edit name

- Foreground Line Style

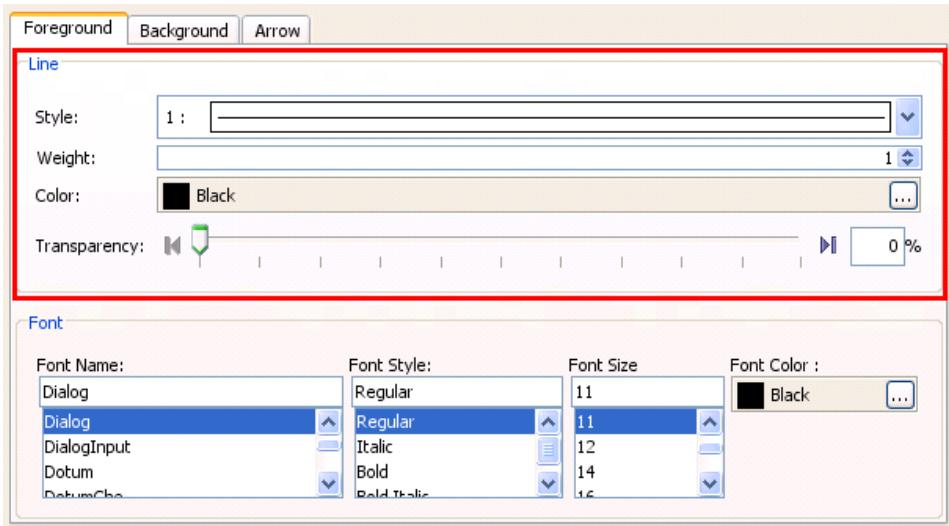


Figure 6-21 edit line style

- Font Style

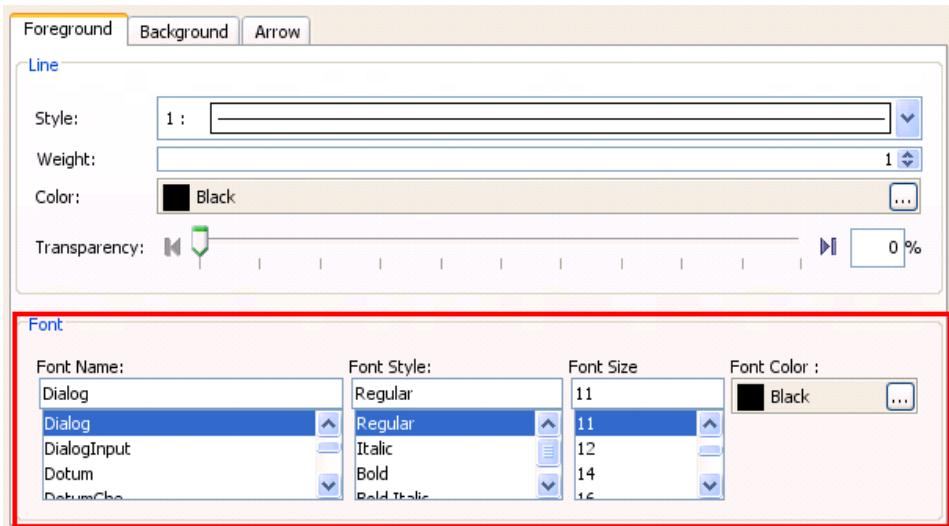


Figure 6-22 edit font style

- Background Style

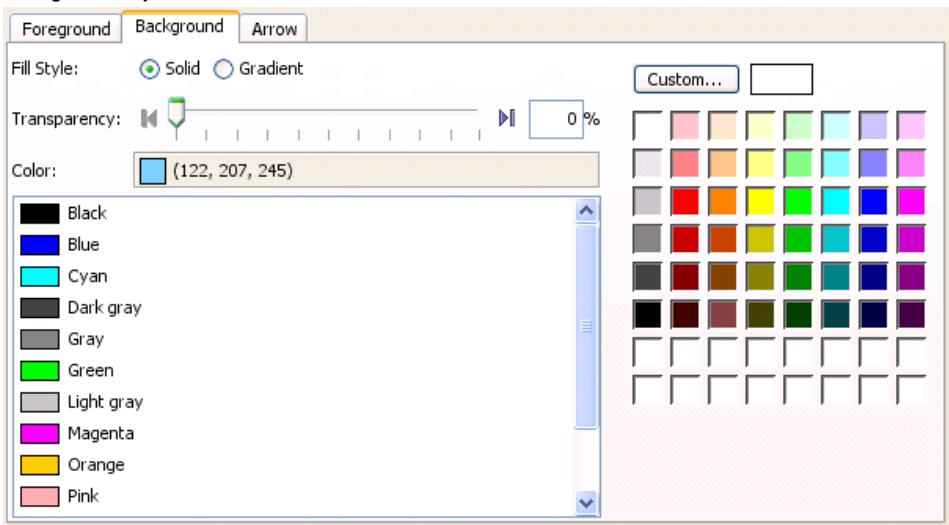


Figure 6-23 edit background style

- Arrow Style

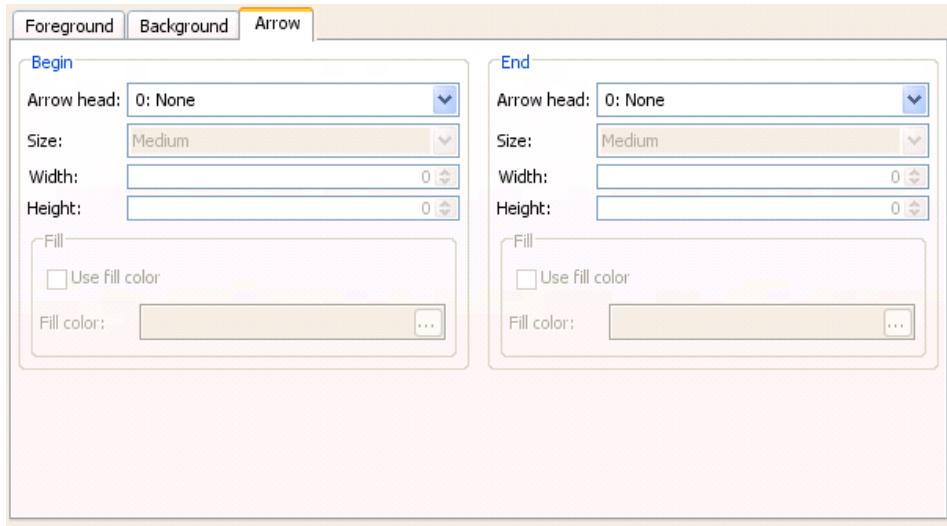


Figure 6-24 edit arrow style

After change the settings, select **OK** to add the style.

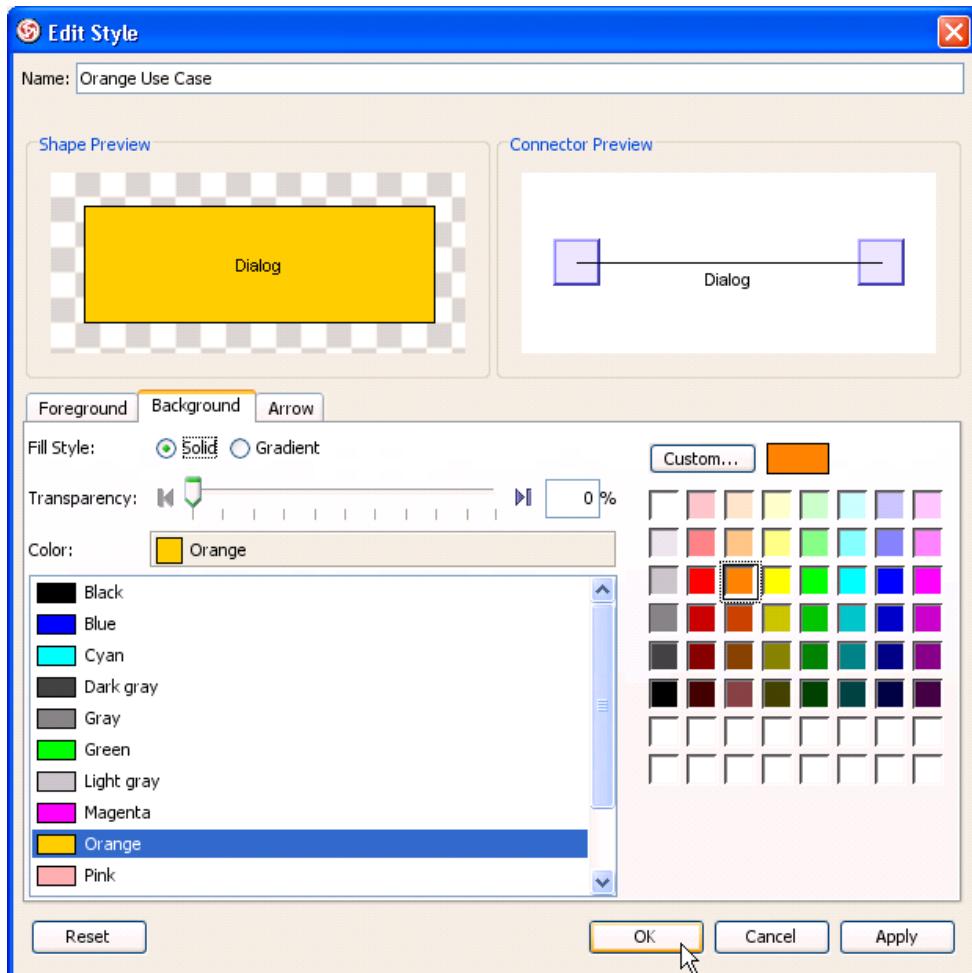


Figure 6-25 style settings

The style was added to the project.

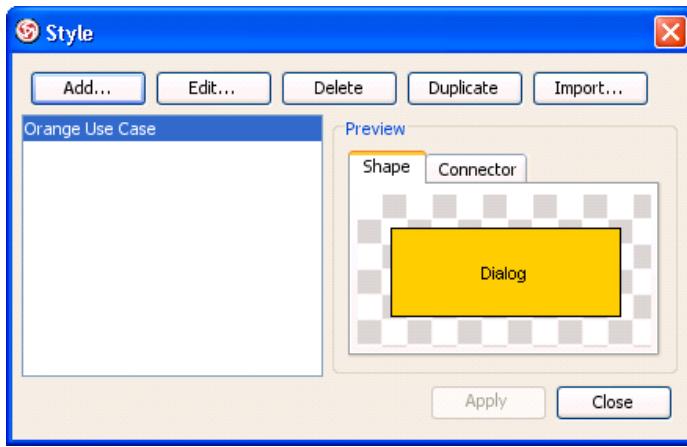


Figure 6-26 style added

Applying style

Upon keeping the Style dialog open, create a new use case diagram together with a new use case.

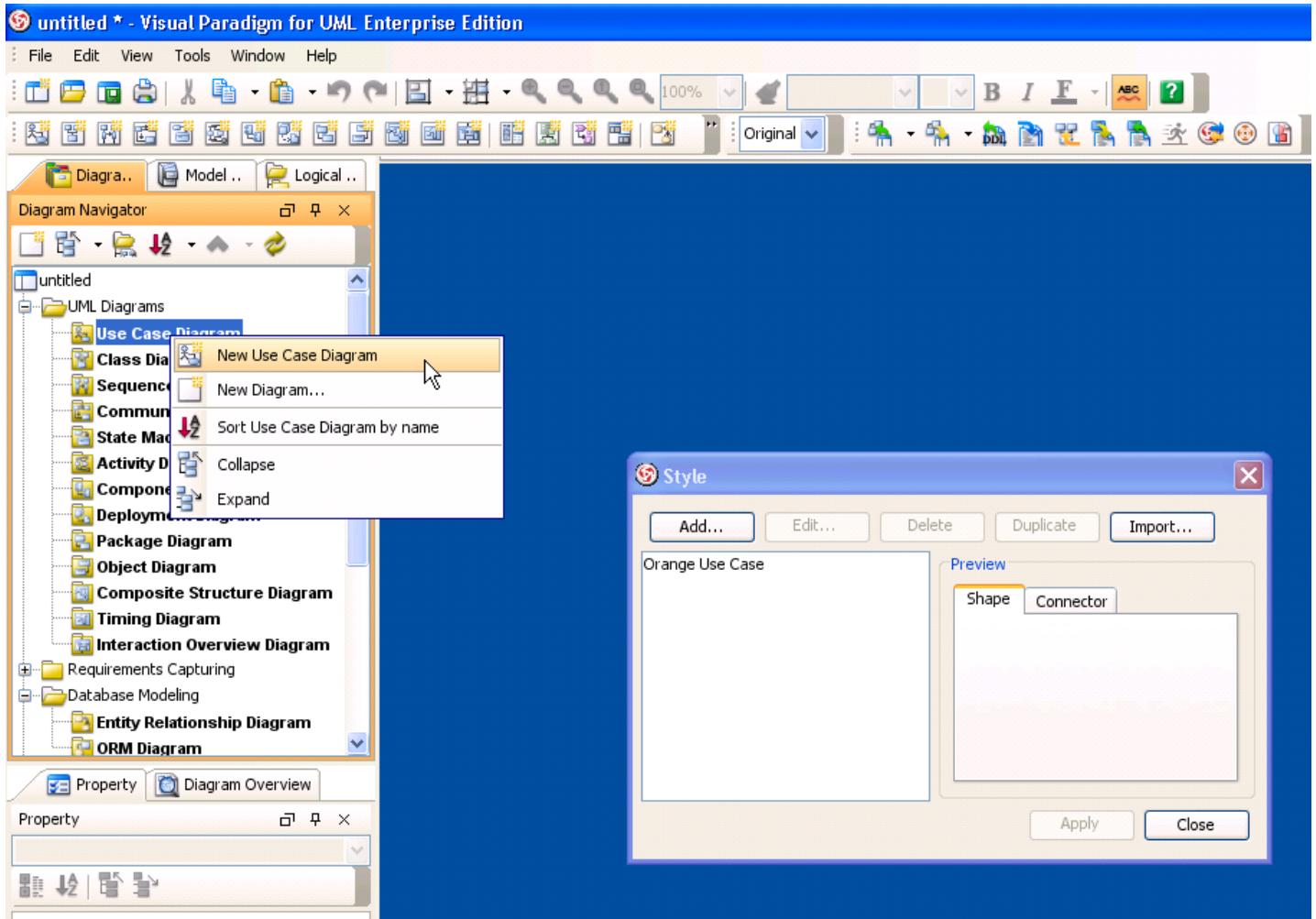


Figure 6-27 New use case diagram

Select the shape, and click **Apply** in the **Style** dialog.

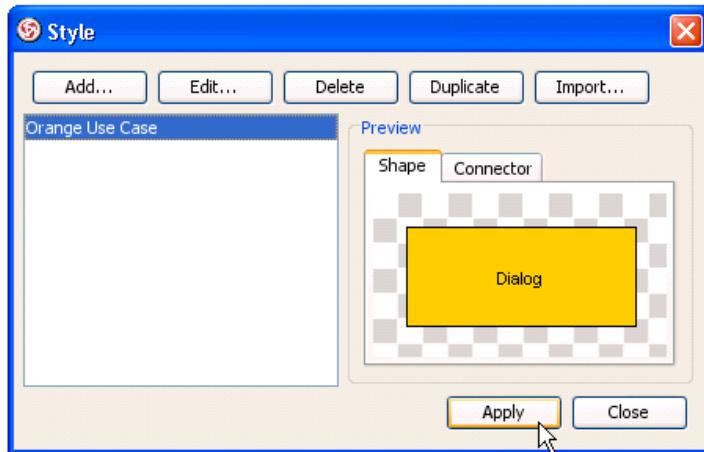


Figure 6-28 Apply style to shape

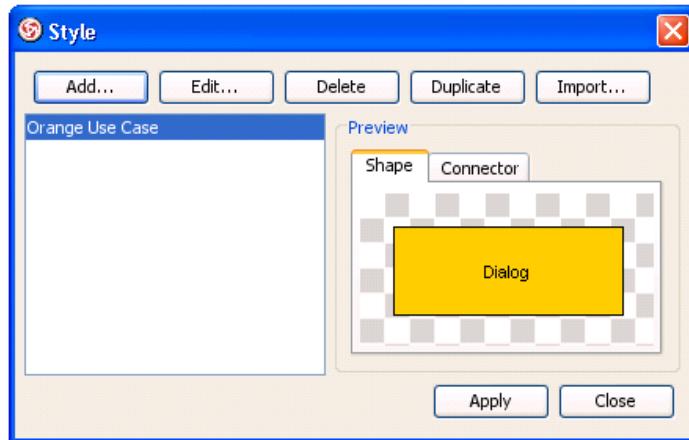


Figure 6-29 Apply style result

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, VP-UML provides five connector styles to help you handle the connectors, namely **Rectilinear**, **Oblique**, **Curve**, **Round Oblique** and **Round Rectilinear**.

Setting connector line style

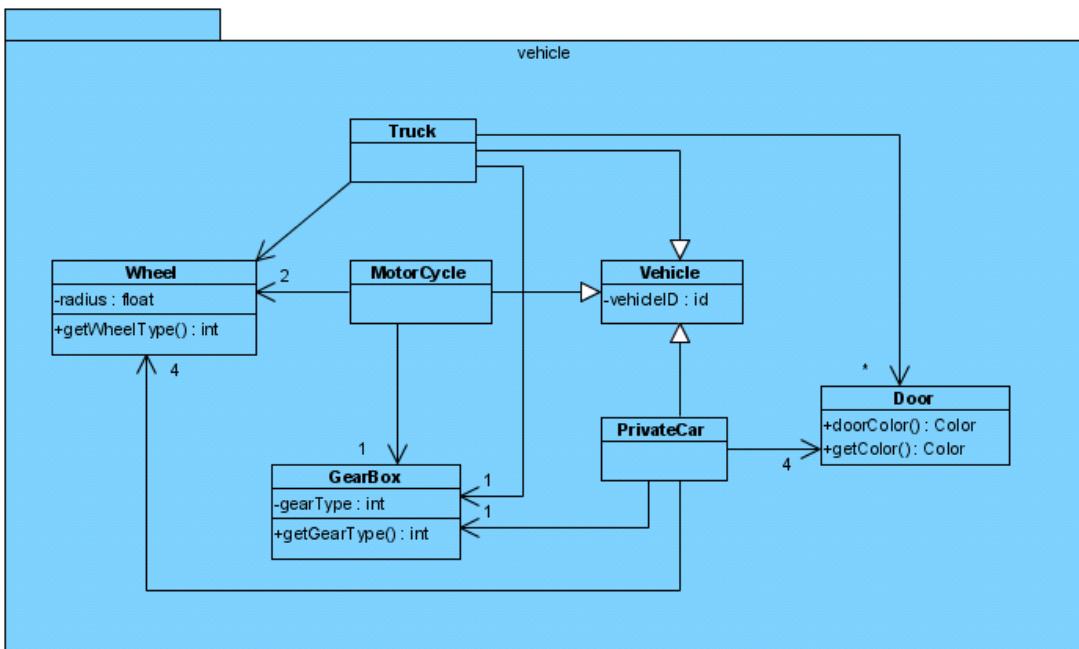


Figure 6-30 Sample class diagram

To change the line style, select the connector and select **Style and Formatting > Connector Style > Rectilinear** from the popup menu.

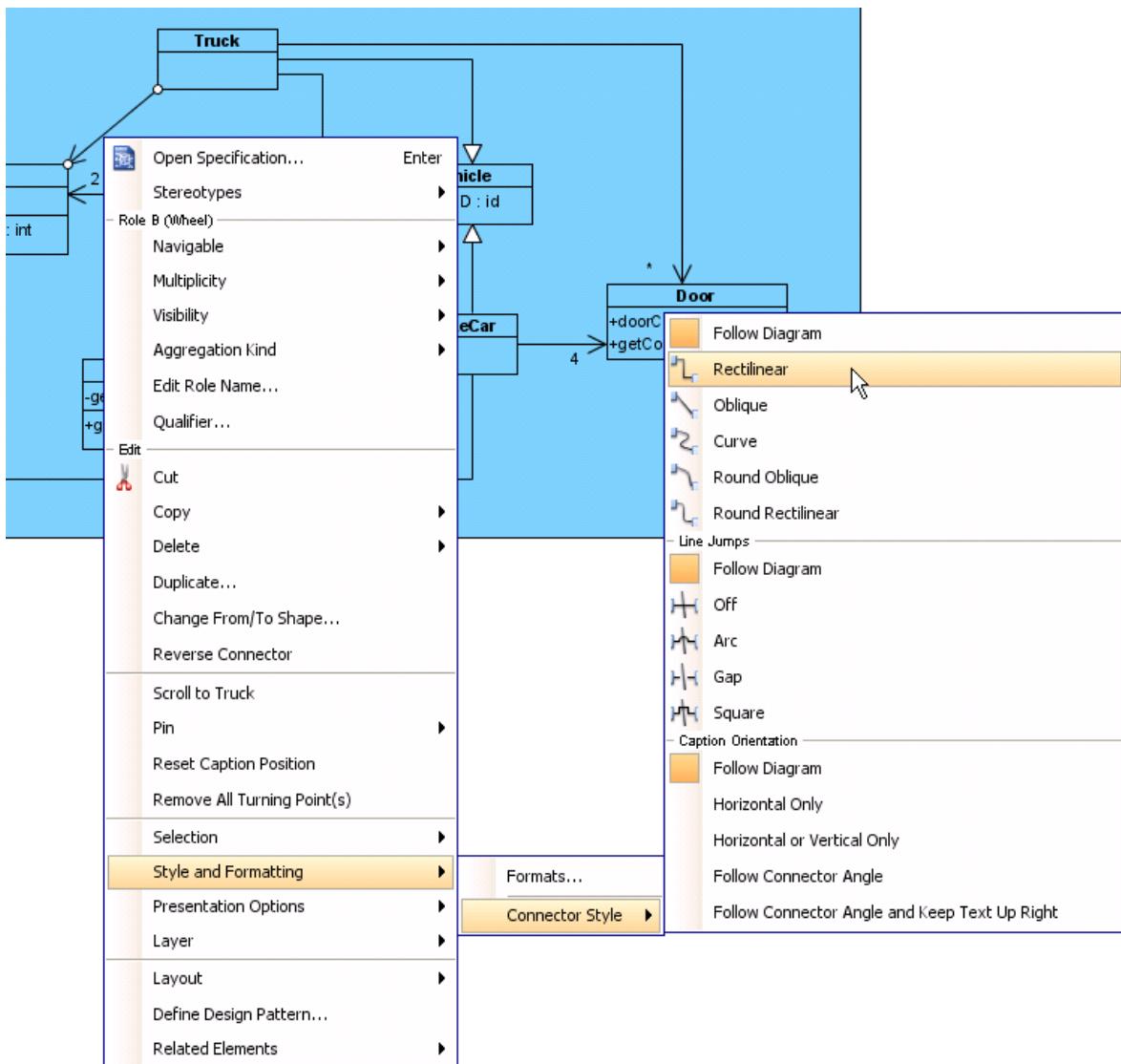


Figure 6-31 Change line style

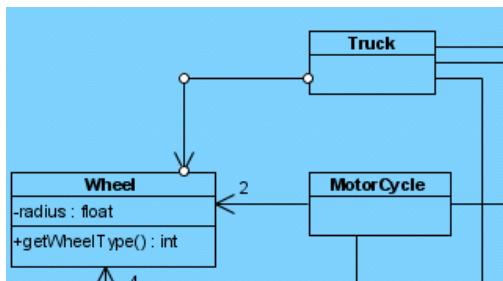
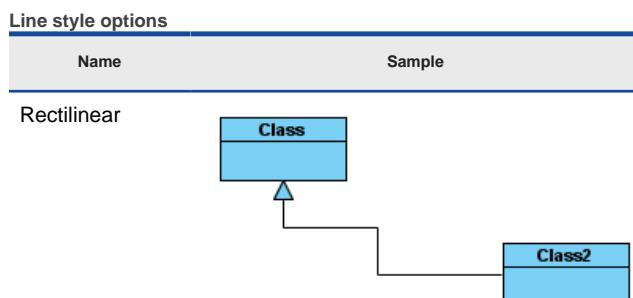
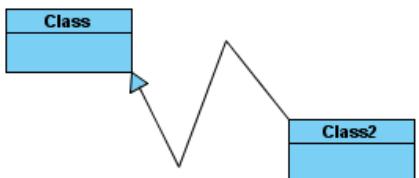


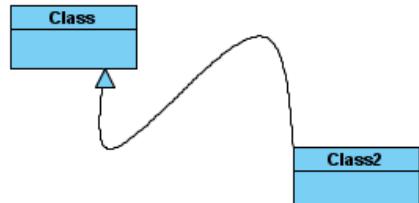
Figure 6-32 Rectilinear style



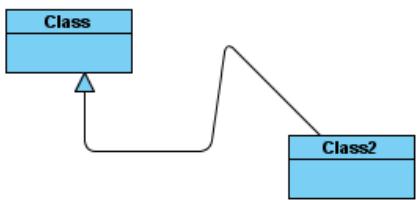
Oblique



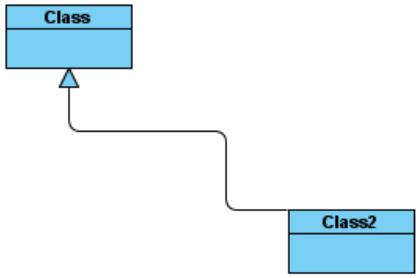
Curve



Round Oblique



Round
Rectilinear



Setting diagram base line style

Beside the 5 style mentioned above, there also have **Follow Diagram** feature, you don't need to set it one by one if you want to change all connectors in the diagrams which defined as **Follow Diagram**.

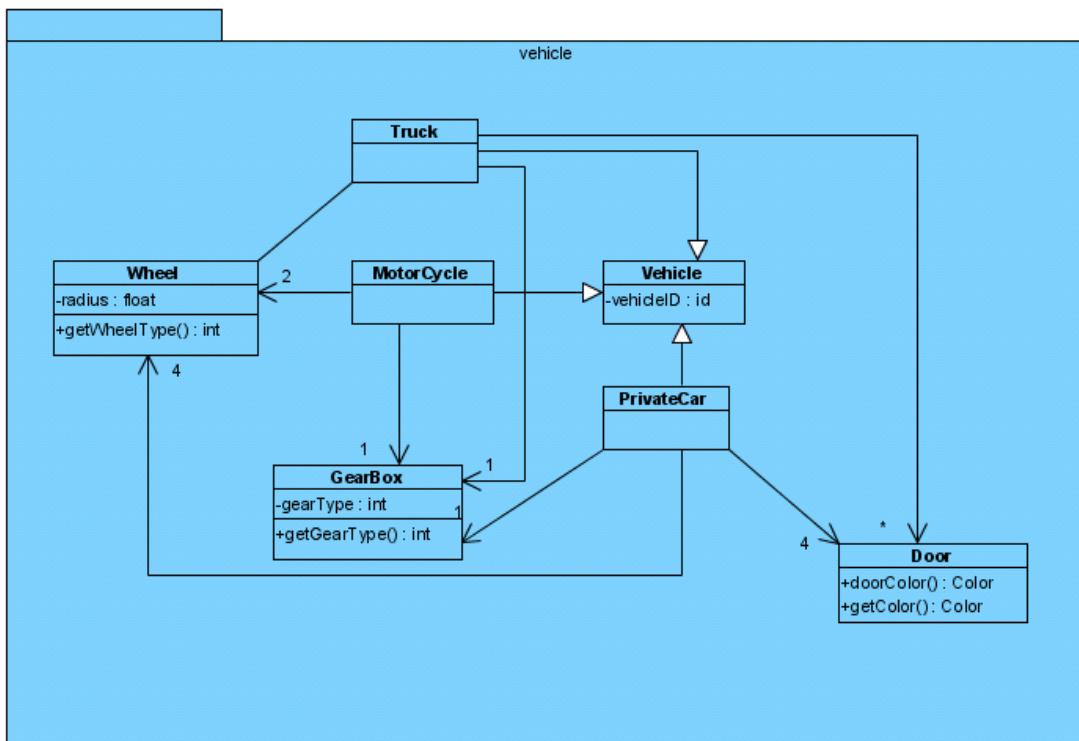


Figure 6-33 Sample class diagram 2

To change the diagram line style, right click on the diagram, and select **Connectors > Curve** from the popup menu.

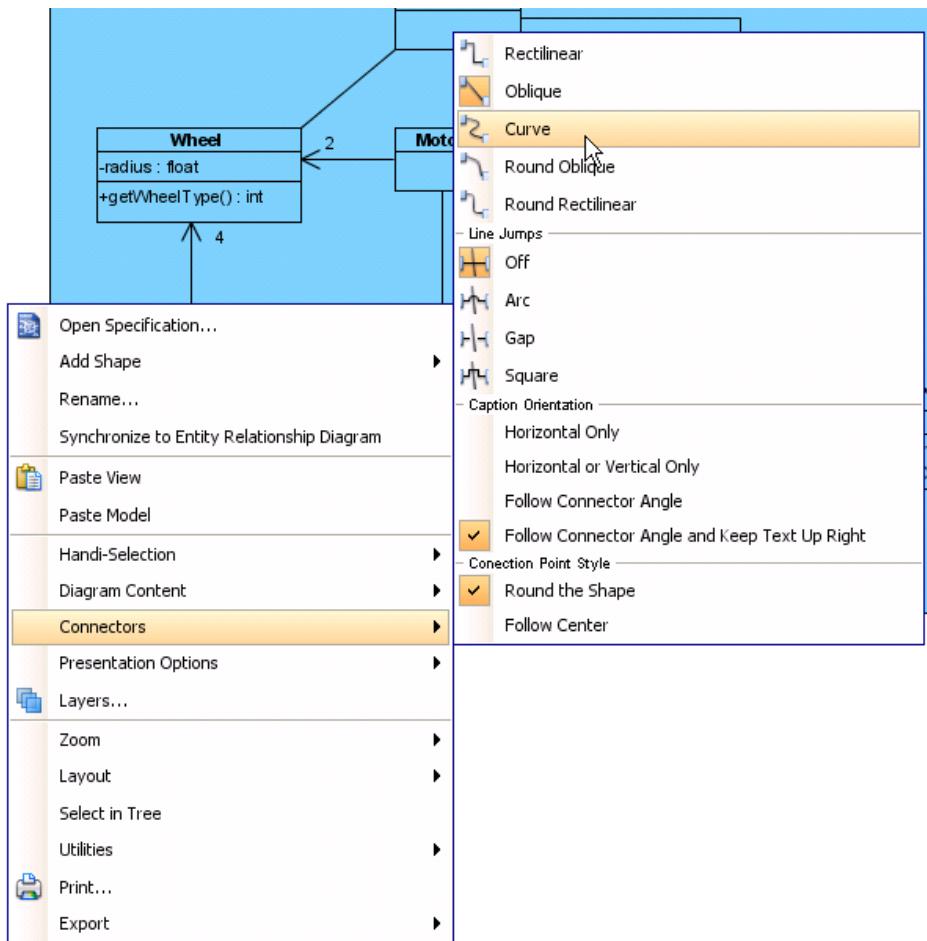


Figure 6-34 Change diagram line style

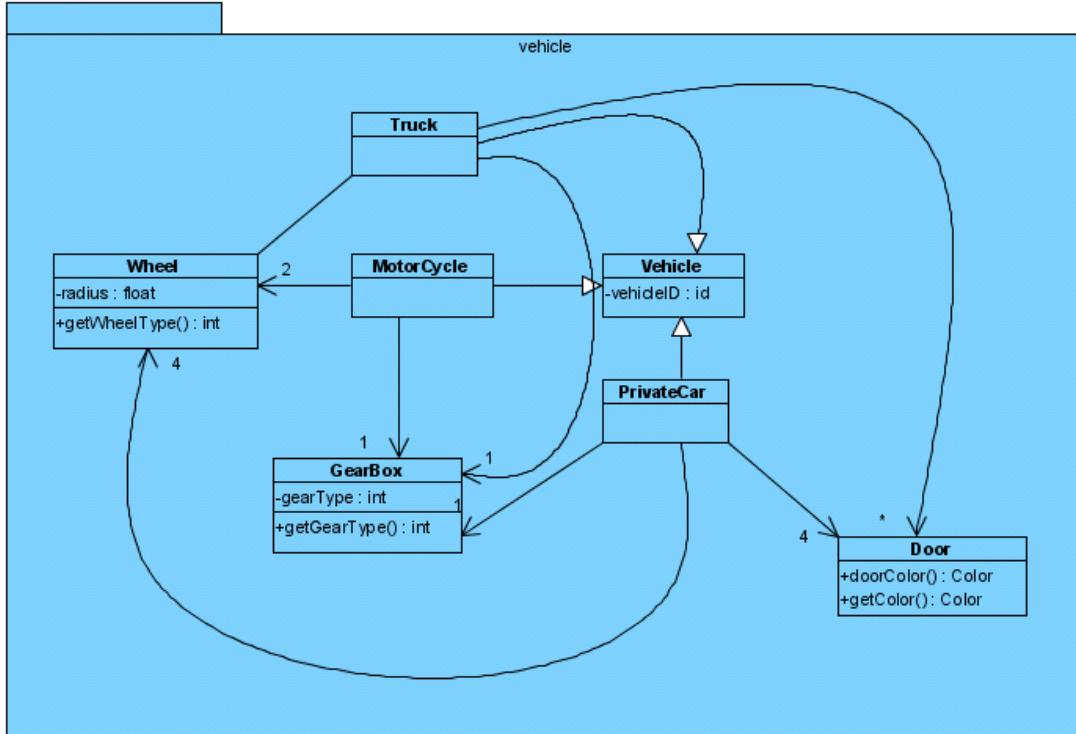


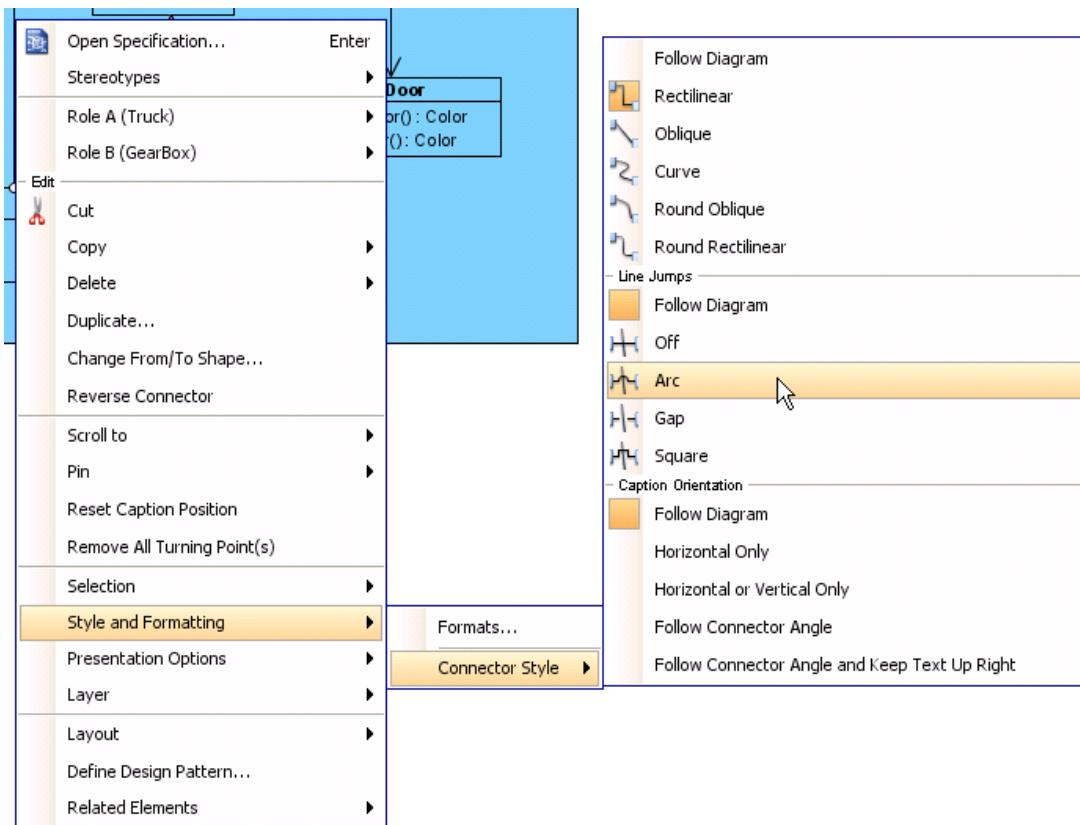
Figure 6-35 Curve line style

Setting line jumps options

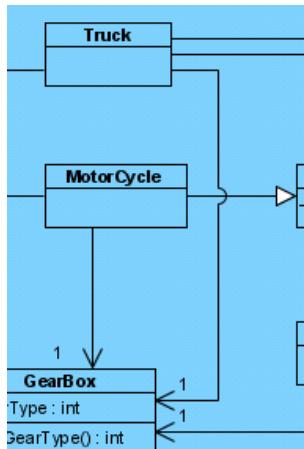
Occasionally, as more diagram elements on your diagram, more miscellaneous connectors are overlapped with each other. It is impossible to explicate on those intersections which connector you indicate. The advantage of Line Jumps is making one of the two connectors different to another to indicate that which connector links with which diagram element clearly.

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 one out of four options under **Line Jumps**.



Changing Arc line jumps

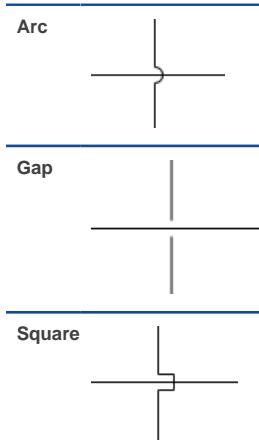


Arc sample

NOTE: The line jumps options are available only when two connectors are overlapped.

Line jump options

Name	Sample
Off	

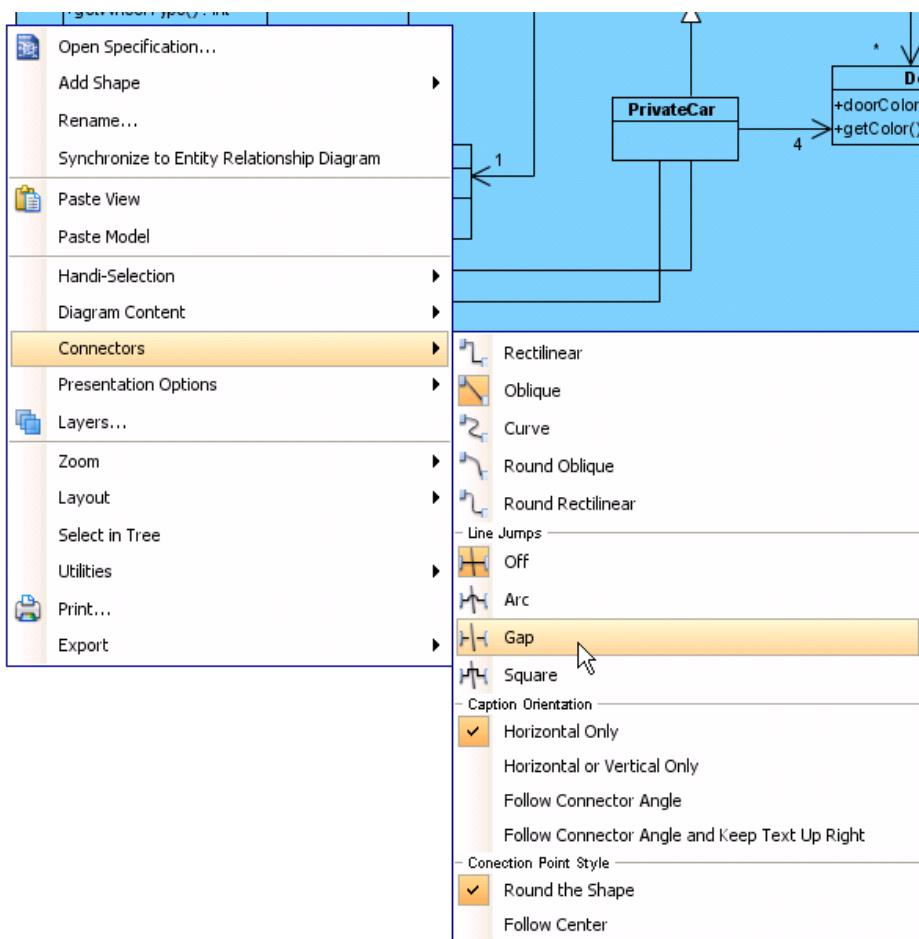


The description of 4 line jump options

Setting diagram base line jumps options

In addition to the 4 options mentioned above, **Follow Diagram** is another choice for altering the connectors. The prime feature of **Follow Diagram** is, instead of setting it one by one, all connectors in the diagrams can be changed simultaneously instead of setting one by one.

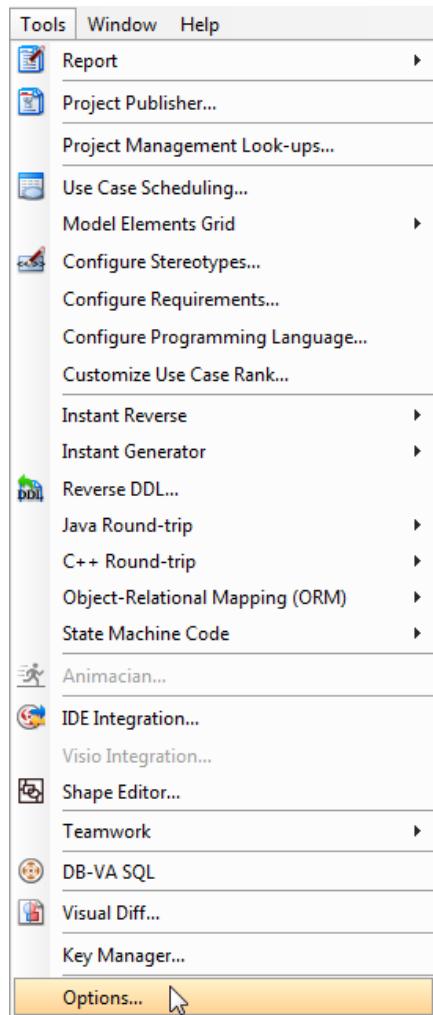
Right Click on the diagram background, select **Connectors** and select one out of four options under **Line Jumps** from the popup menu.



*Selecting **Gap** from the popup menu*

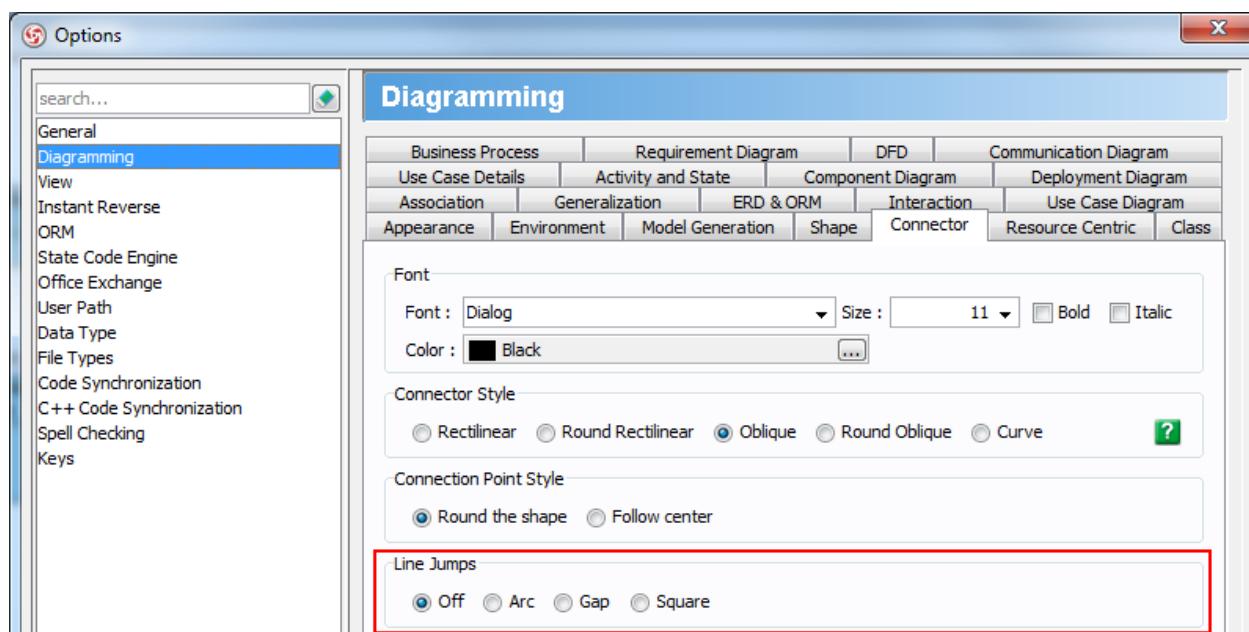
Setting Line Jump for New Project

select **Tools > Options** from the main menu.



Clicking **Options...** in popup menu

In the **Options** dialog box, open the **Diagramming** page, switch to the **Connector** tab and select the **Line Jumps** style, or select **Off** to disable it. At last, Click **OK** to confirm the changes.

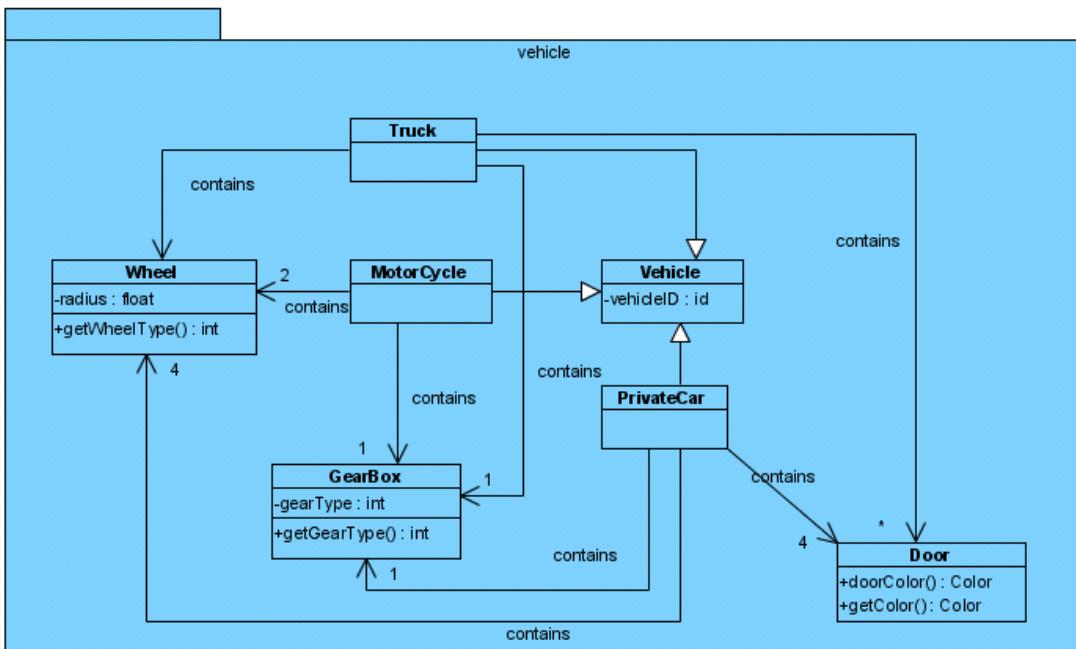


Selecting **Off** in **Options** dialog box

Setting connector caption orientation

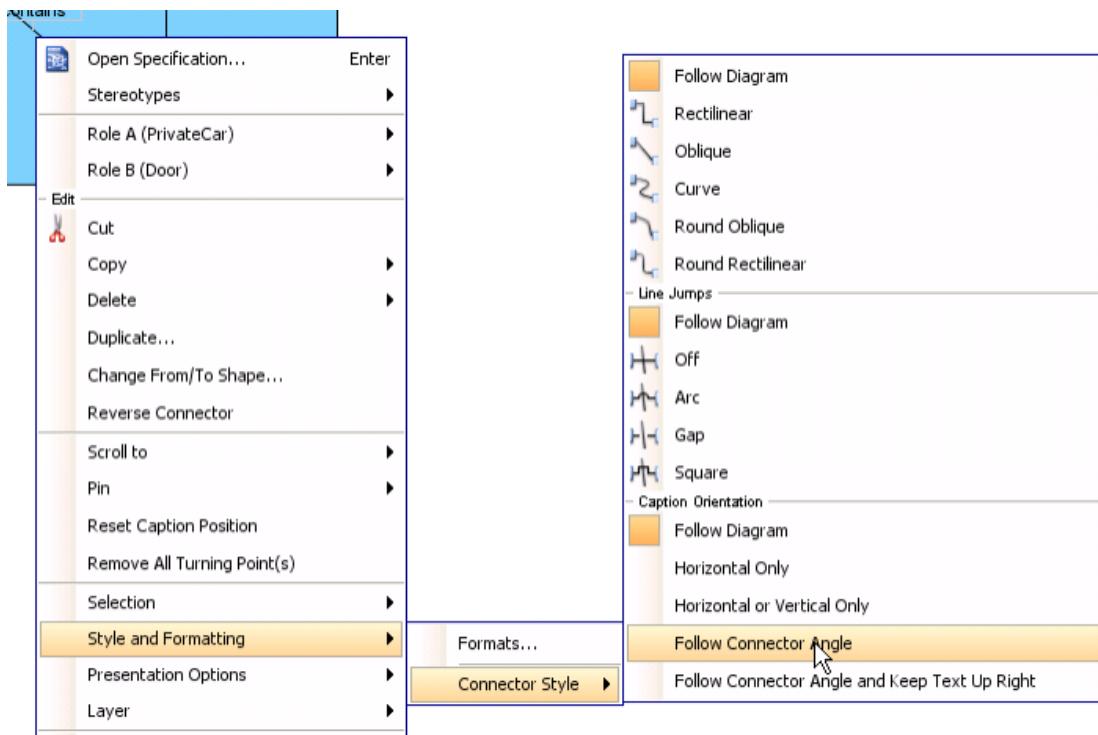
Visual Paradigm supports a number of ways of aligning connector caption, which apply on different modeling preferences. By default, the caption of a connector is aligned horizontal only, but you also can customize it to **Follow Diagram**, **Horizontal Only**, **Horizontal or Vertical Only**, **Follow 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

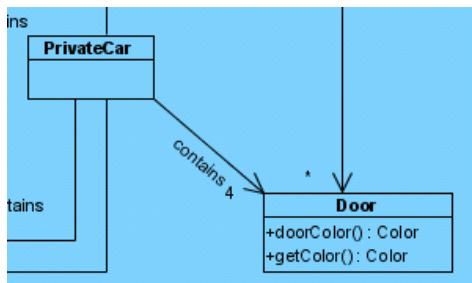


Sample class diagram 1

To customize the caption orientation option of a specify 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

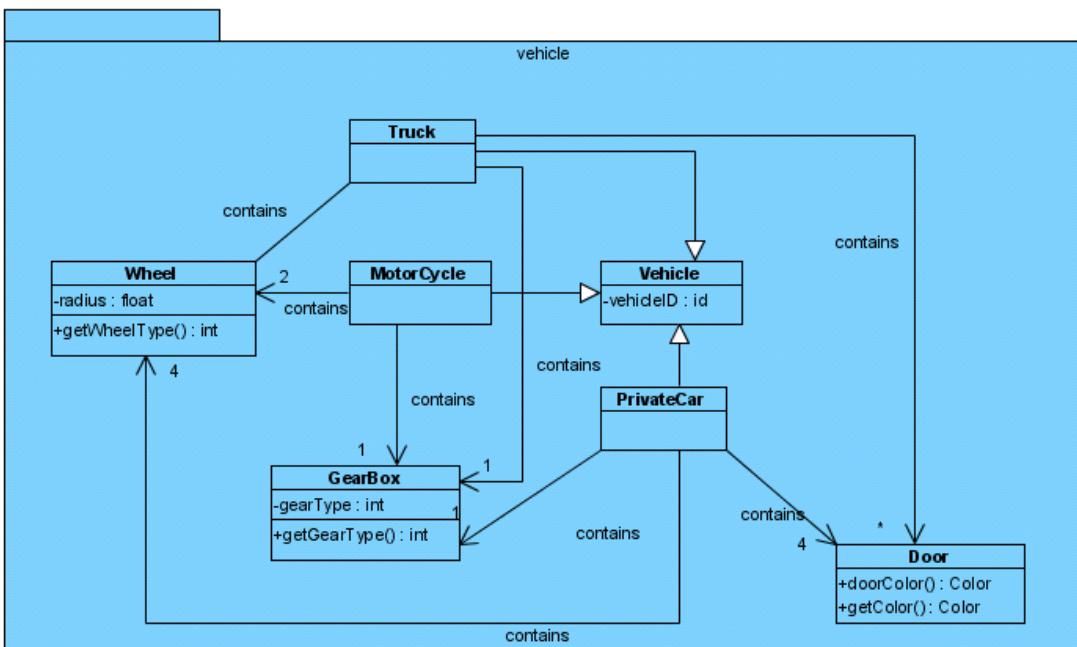
Caption orientation options

Name	Sample	Description
Horizontal Only	<pre> classDiagram class Class class Class2 Class --> Class2 "caption" </pre>	The caption of connector is aligned horizontally.
Horizontal or Vertical Only	<pre> classDiagram class Class class Class2 Class --> Class2 "caption" </pre>	The caption of connector is aligned either horizontally or vertically, according to the connector.
Follow Connector Angle	<pre> classDiagram class Class class Class2 Class --> Class2 "caption" </pre>	The caption of connector is aligned the diagonal angles of both shapes.
Follow Connector Angle and Keep Text Up Right	<pre> classDiagram class Class class Class2 Class --> Class2 "caption" </pre>	The caption of connector is aligned the diagonal angles of both shapes, but keeps top right.

The description of 4 caption orientation options

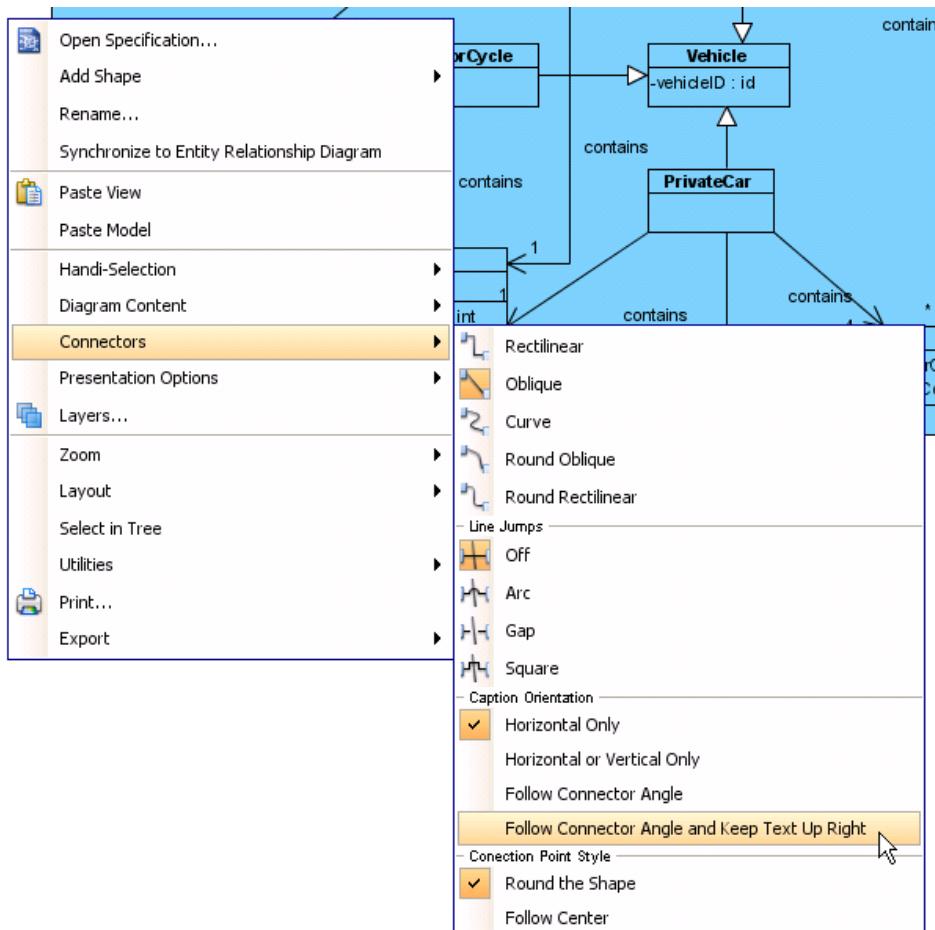
Setting diagram base connector caption direction

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.



Sample class diagram 2

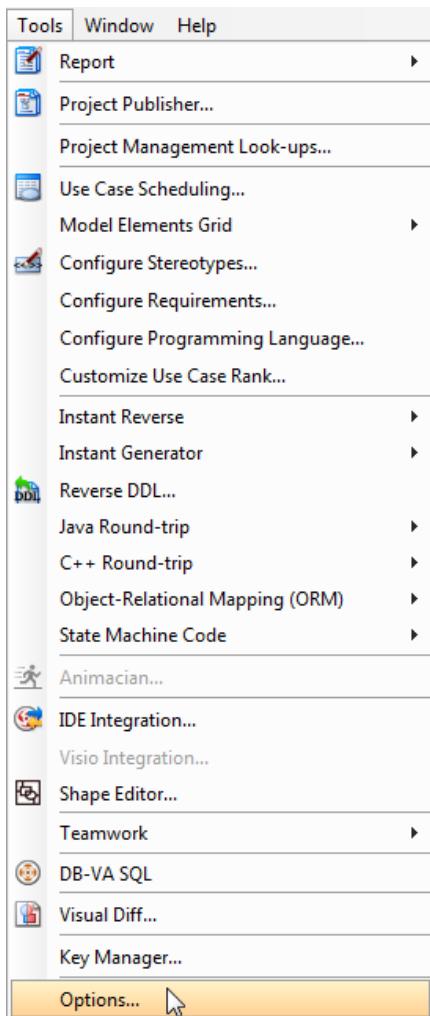
Right Click on the diagram background, select **Connectors** and select one out of four options under **Caption Orientation** from the popup menu.



Change caption orientation by diagram popup menu

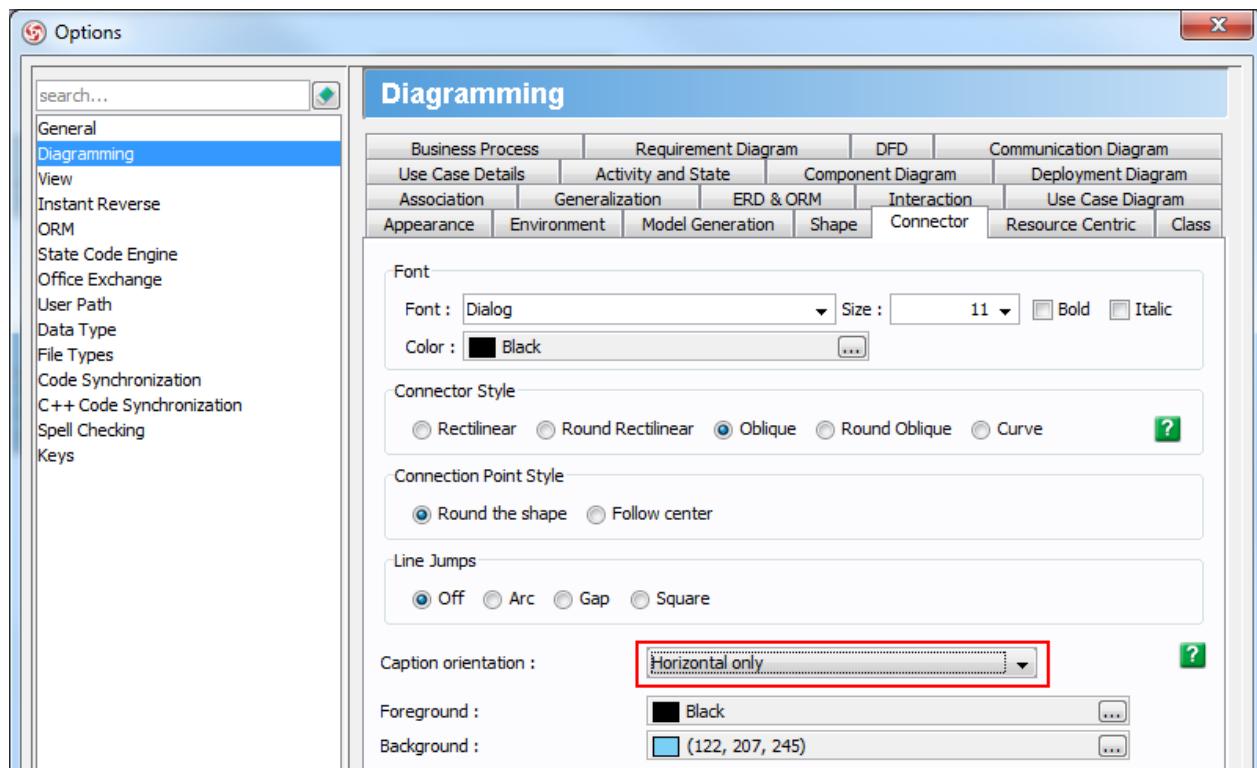
Workspace wide

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 **Tools > Options** from the main menu.



Clicking **Options...** in popup menu

In the Options dialog box, 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.



Selecting **Horizontal only** in **Options** dialog box

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.

To copy format to another shape

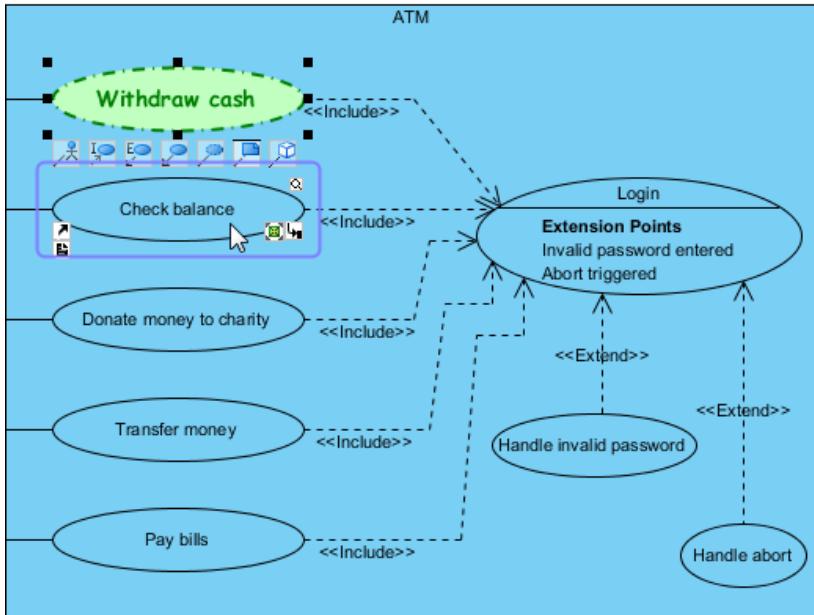
If you want another shape to have exactly the same formatting properties as the previous one you have done, you can simply:

1. Click on the shape that you want its format to be cloned.
2. Click on the **Format Copier** button in toolbar.



Clicking Format Copier

3. Click the shape you want to format.



Cloning the format property from one shape to another

NOTE: You can copy formatting to another type(s) of shape.

To copy format to multiple shapes

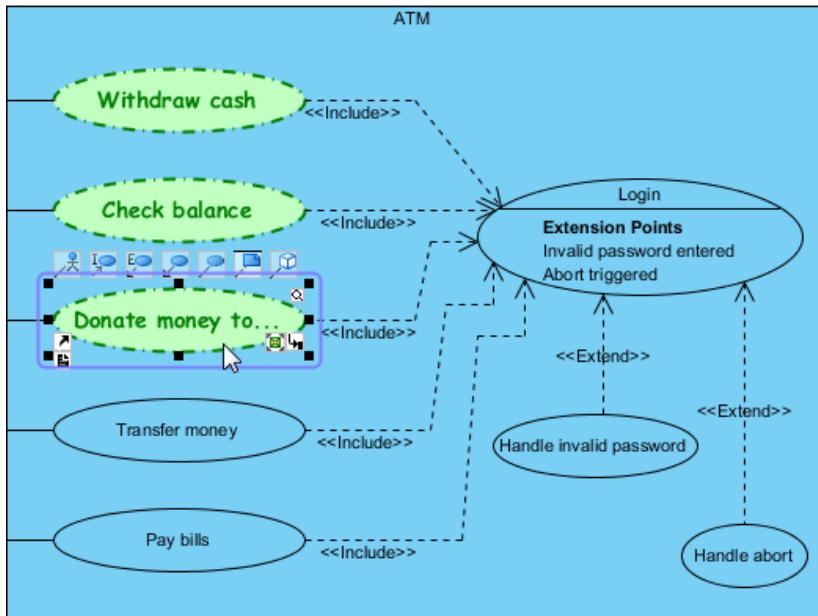
If you want the format properties of your previous shape to be cloned to more than one shapes, you should:

1. Click on the shape that you want its format to be cloned.
2. Double click the **Format Copier** button on toolbar.



Double Clicking Format Copier

3. Click the shape you want to format.



Cloning 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 Format copier once again.

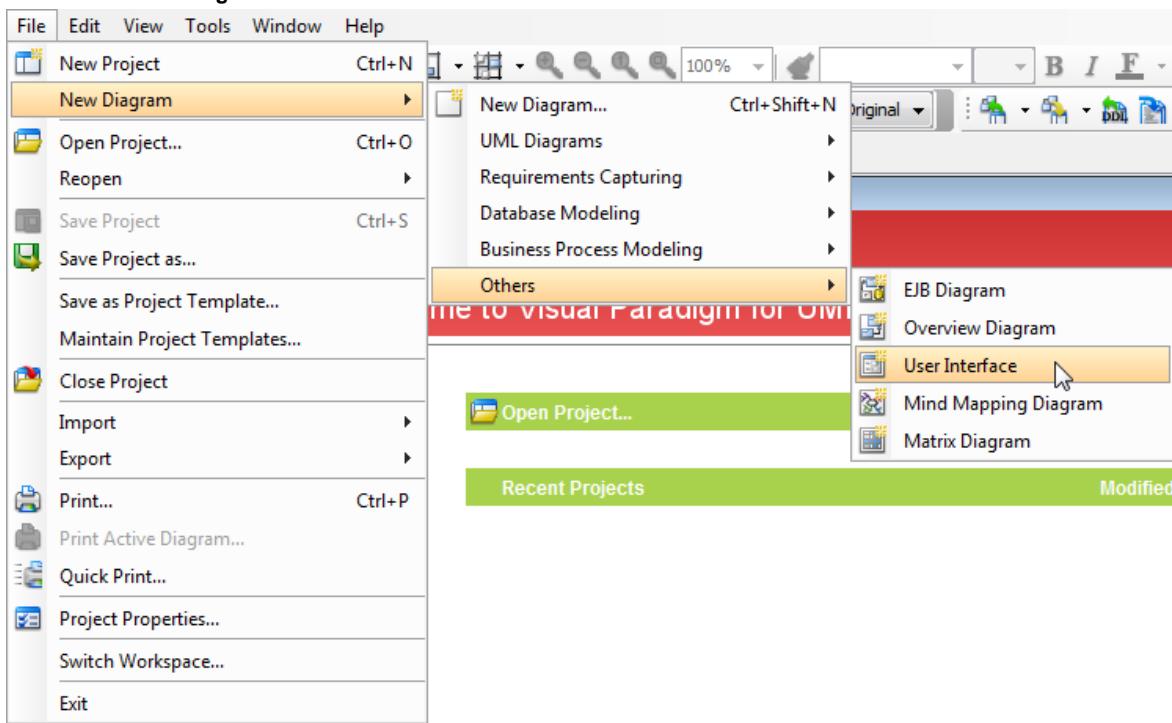
NOTE: You can only copy format to shapes within the same diagram.

Creating user interface diagram

You can design the graphical user interface of your application or system by using user interface diagram. The diagram toolbar of User Interface provides you with common GUI components like frame, text fields, labels, buttons, etc. You can select appropriate components and drag them to diagram to construct the GUI. You can also change the appearance of user interface by modifying GUI components' properties.

To create a user interface diagram:

1. Select **File > New Diagram > Others > User Interface** from the main menu.

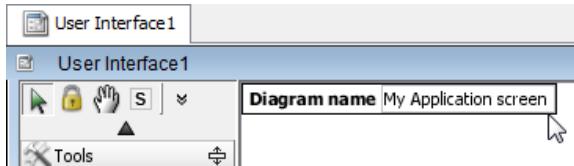


Create a User Interface diagram through the main menu

NOTE: Alternatively, you can create a User Interface diagram with the steps below:

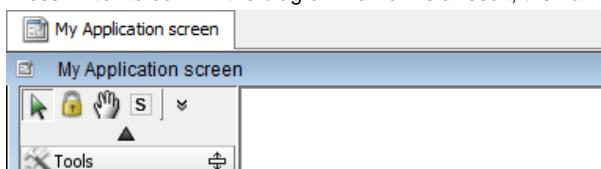
- Right click on **User Interface** in **Diagram Navigator** and select **New User Interface** from the popup menu.
- Click on the **New User Interface** button in toolbar
- Click on **New User Interface** in Start Page

2. Name your screen design by filling in the diagram name.



Enter the diagram name

3. Press **Enter** to confirm the diagram name. As a result, the name for User Interface diagram is created.



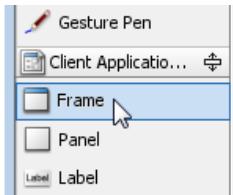
A User Interface diagram is created

Frame

A Frame is a top-level window with a title and a border. The size of the frame includes any area designated for the border. User can add components on a frame.

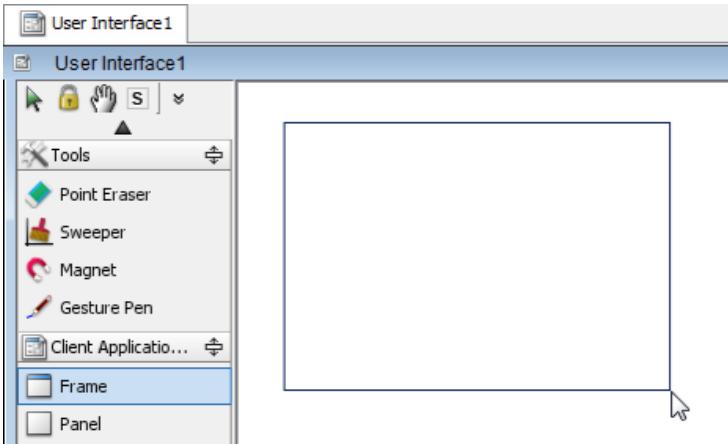
Creating a frame

1. Select the **Frame** tool from the diagram toolbar.



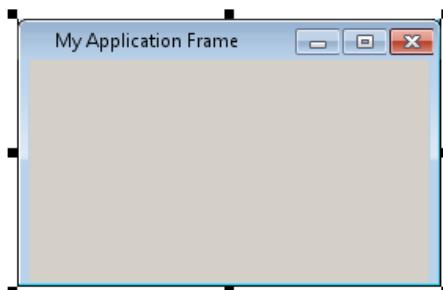
Selecting **Frame**
from diagram toolbar

2. Press on diagram to set the position of the frame. Drag diagonally from the starting point to expand the frame.



Drag diagonally to expand the frame

3. Release the mouse button to confirm the size of frame.
4. Enter the frame title and press **Enter** to confirm.



A frame is created

Frame properties

The appearance of frame can be changed by editing its properties. To configure a frame, right click on the Frame and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a frame.

Property	Description
Title	Title refers to the text that appear at the top of frame, which used to be the caption of application or the function of opening frame.
Icon	Icon is the tiny image that appear at the top left of frame. Users are required to provide a valid image file as icon.
Iconifiable	The Iconifiable state controls whether the minimize button is shown or not. When Iconifiable is set, the minimize button is shown, otherwise hidden.
Maximizable	The Maximizable state controls whether the maximize button is shown or not. When Maximizable is set, the maximize button is shown, otherwise hidden.
Closable	The Closable state controls whether the close button is shown or not. When Closable is set, the close button is shown, otherwise hidden.

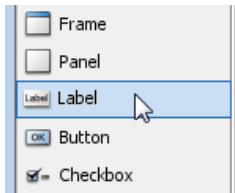
Description of frame properties

Label

A label is a text component that appear on a screen. The text "User", "Password", "Address" appear on a registration form are examples of labels.

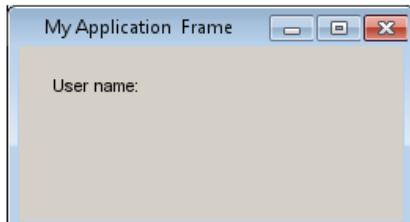
Creating a label

1. Select the **Label** tool from the diagram toolbar.



*Selecting **Label**
from diagram toolbar*

2. Click on a container (e.g. a Frame) or diagram background to create a label.
3. Enter the label caption and press **Enter** to confirm the caption of label.



A label is created

Label properties

The appearance of label can be changed by editing its properties. To configure a label, right click on the label and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a label.

Property	Description
Caption	Caption is the text appear in a label.
Mnemonic	Mnemonic is a key which enables users to select a label by simultaneously pressing the Alt key and the mnemonic key on the keyboard.

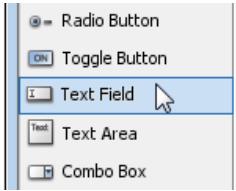
Description of label properties

Text field

Text Field is a component that allows the editing of a single line of text. Fields in a registration form are typical examples of text fields.

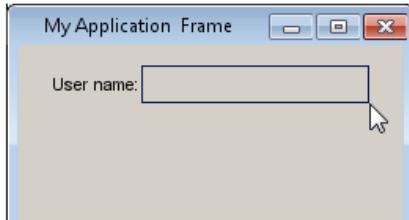
Creating a text field

1. Select the **Text Field** tool from the diagram toolbar.



Selecting **Text Field**
from diagram toolbar

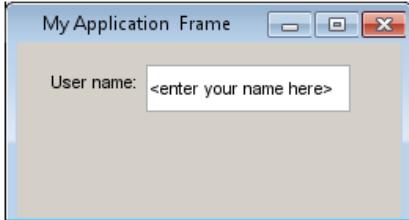
2. Press to set the position of the text field. Drag diagonally from the starting point to expand the text field.



Drag diagonally to expand the text field

3. Release the mouse button to confirm the size of text field.

4. Enter the text field text and press **Enter** to confirm the title of text field.



A text field is created

Text field properties

The appearance of text field can be changed by editing its properties. To configure a text field, right click on the text field and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a text field.

Property	Description
Text	Text refers to the text that appear in the text field.

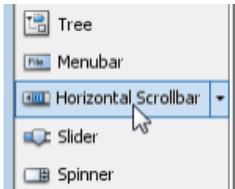
Description of text field properties

Scrollbar

A scrollbar is a bar with a knob which appear at the bottom or right of a container, such as a frame. The user positions the knob in the scrollbar to determine the contents of the viewing area. The program typically adjusts the display so that the end of the scrollbar represents the end of the displayable contents, or 100% of the contents. The start of the scrollbar is the beginning of the displayable contents, or 0%. The position of the knob within those bounds then translates to the corresponding percentage of the displayable contents.

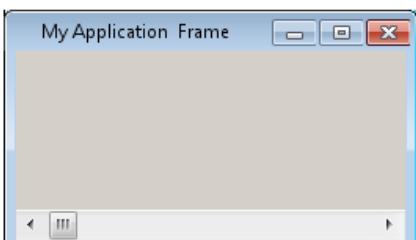
Creating a scrollbar

1. Select the **Horizontal/Vertical Scrollbar** tool from the diagram toolbar. You can change between a horizontal or a vertical scrollbar tool by clicking on the tiny reverted triangle, and selecting the preferred type of scrollbar from the popup menu.



Selecting **Horizontal Scrollbar** from diagram toolbar

2. Click on a container (e.g. a Frame) or diagram background to create a scrollbar. If you create inside a container, and if it has no scrollbar with same orientation exists, the scrollbar will be docked to the bottom or the right of the container, depending on the scrollbar orientation.



Scrollbar is created

Change orientation with resource-Centric interface

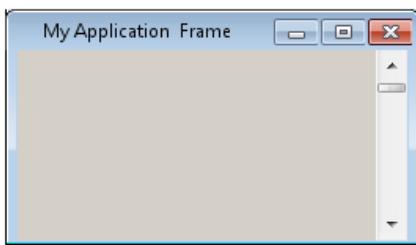
A scrollbar can be oriented horizontally or vertically, which determines the contents of the viewing area when being scrolled. User can change the orientation of a scrollbar easily through the resource-centric interface. When a scrollbar is put inside a container, changing orientation will automatically dock to the bottom or the right of container. To change scrollbar orientation:

1. Move the mouse cursor over the scrollbar.
2. Click on the resource icon **Switch Orientation**.



Click on the resource icon for switching **Scrollbar Orientation**

3. The orientation of scrollbar then be switched.



Scrollbar Orientation switched from horizontal to vertical

Automatic sticking scrollbar to frame

In a normal User Interface design, scrollbars, no matter horizontal or vertical, are placed at the bottom or on the right of a container component. VP-UML follows this practice. When creating a scrollbar in a container component, or when moving a scrollbar into a container component, the scrollbar will be docked to the border of component automatically.

Scrollbar properties

The appearance of scrollbar can be changed by editing its properties. To configure a scrollbar, right click on the scrollbar and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a scrollbar.

Property	Description
Orientation	Indicates if the scroll bar is vertical or horizontal.
Block increment	Amount the value changes when the scrollbar track is clicked on either side of the knob.
Unit increment	Amount the value changes when the end arrows of the scrollbar are clicked.
Minimum	The minimum value of scrollbar.
Maximum	The maximum value of scrollbar.
Value	Value which controls the location of the scroll bar knob.

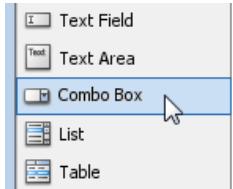
Description of scrollbar properties

Combo box

Combo box is a component that combines a button and a drop-down list. The user can select a value from the drop-down list, which appears at the user's request.

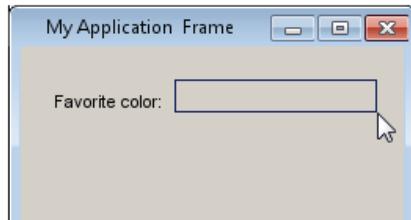
Creating a combo box

1. Select the **Combo Box** tool from the diagram toolbar.



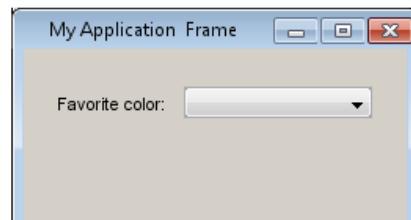
Selecting **Combo Box**
from diagram toolbar

2. Click to set the position of the combo box. Drag diagonally from the starting point to expand the combo box.



Drag diagonally to expand the Combo box

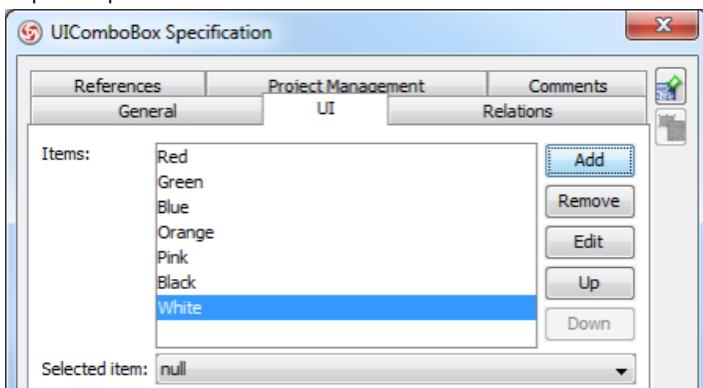
3. Release the mouse button to confirm the size of combo box.



Combo box is created

Editing combo box values

1. Right click on the combo box and select **Open Specification...** from the popup menu.
2. In the **UI** tab of the **UIComboBox Specification** dialog box, click **Add**.
3. Enter the value for of item in **Input** dialog box and click **OK** to confirm.
4. Repeat step 2 and 3 to create all the combo box items.



Combo box items are created

5. Click **OK** to confirm editing.

Combo box properties

The appearance of combo box can be changed by editing its properties. To configure a combo box, right click on the combo box and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a combo box.

Property	Description
Items	The items that can be selected by users.
Selected item	The default selected item.

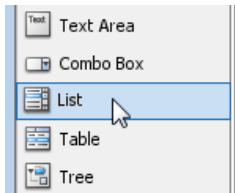
Description of combo box properties

List

List is a component that lists all available selection in rows, and allow the selection of values in it.

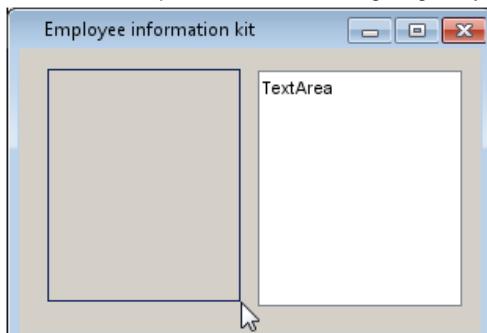
Creating a list

1. Select the **List** tool from the diagram toolbar.



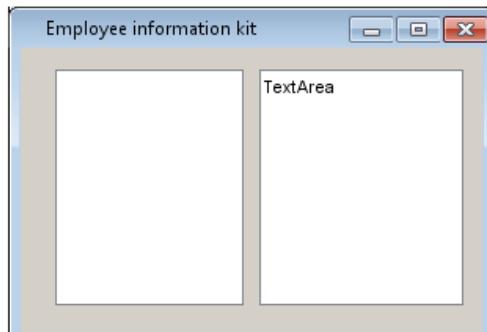
Selecting List from diagram toolbar

2. Press to set the position of the list. Drag diagonally from the starting point to expand the list.



Drag diagonally to expand the List

3. Release the mouse button to confirm the size of list.

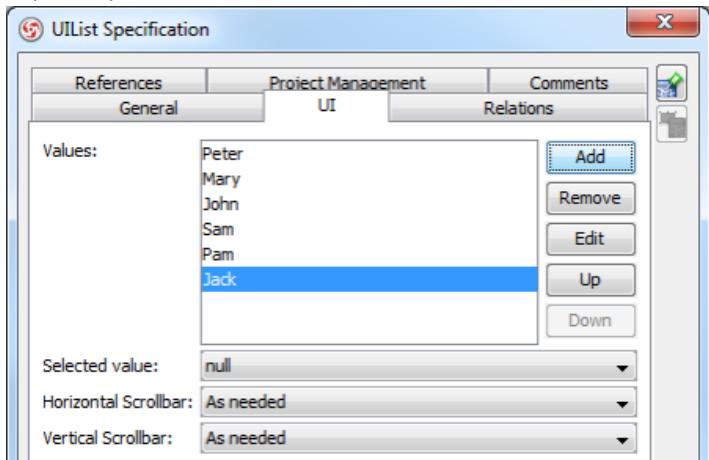


List is created

Editing list values

1. Right click on the list and select **Open Specification...** from the popup menu.
2. In the **UI** tab of the **List** specification dialog box, click **Add**.
3. Enter the value for of item in **Input** dialog box and click **OK** to confirm.

4. Repeat step 2 and 3 to create all the list items.



List items are created

5. Click **OK** to confirm editing.

List properties

The appearance of list can be changed by editing its properties. To configure a list, right click on the list and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a list.

Property	Description
Values	The items that can be selected by user.
Selected value	The default selected item in the list.
Horizontal scroll bar	Determines when a horizontal scrollbar will appear in a list.
Vertical scroll bar	Determines when a vertical scrollbar will appear in a list.

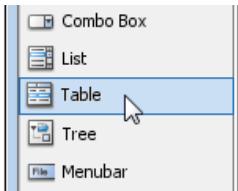
Description of list properties

Table

Table is used to display and edit regular two-dimensional tables of cells.

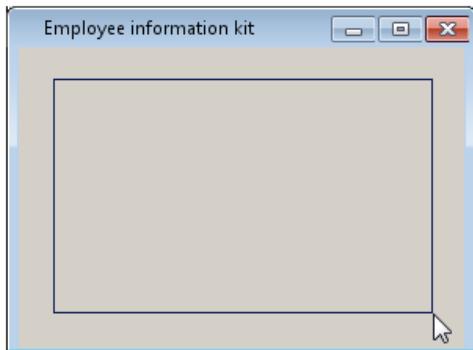
Creating a table

1. Select the **Table** tool from the diagram toolbar.



*Selecting **Table**
from diagram toolbar*

2. Press to set the position of the table. Drag diagonally from the starting point to expand the table.



Drag diagonally to expand the Table

3. Release the mouse button to confirm the size of table.

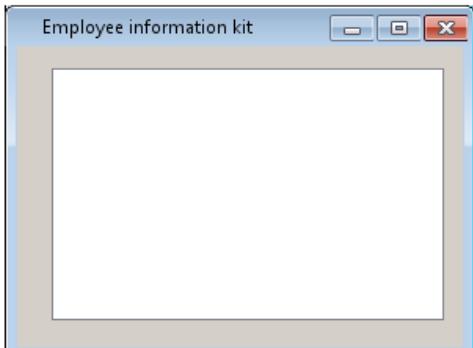


Table is created

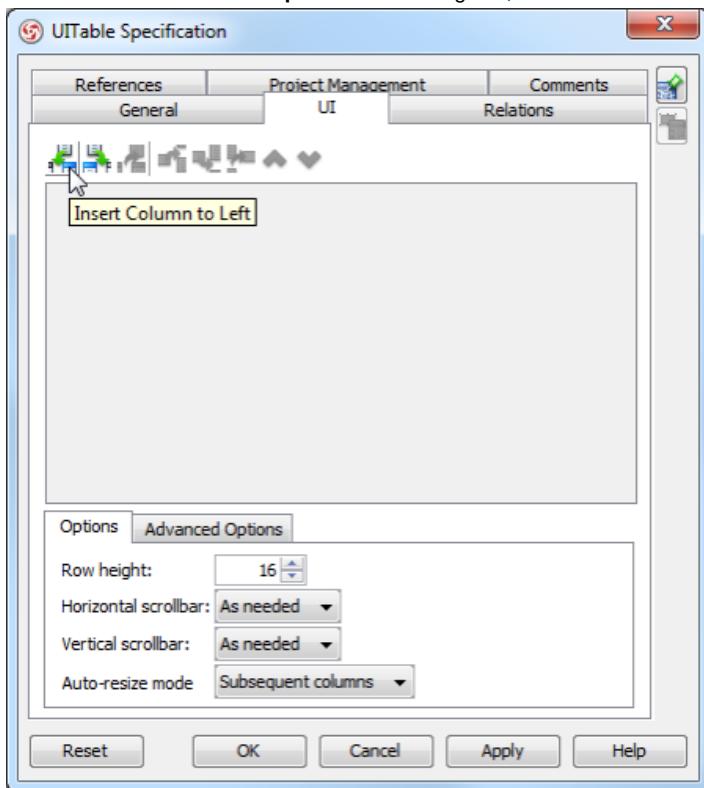
Editing table contents

Basic setup

To insert columns and rows into a table:

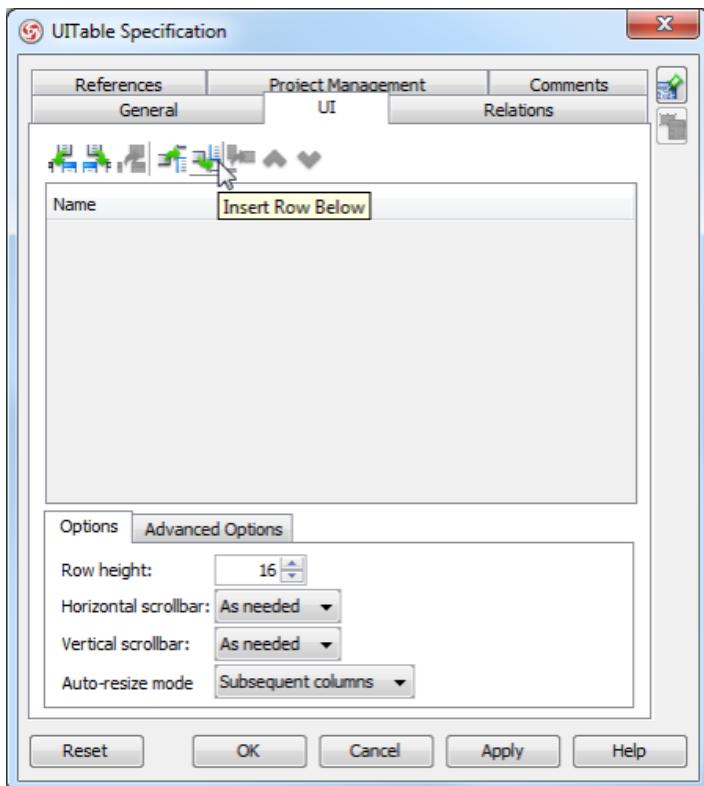
1. Right click on the table and select **Open Specification...** from the popup menu.

2. In the **UI** tab of the **UITable Specification** dialog box, click either **Insert Column to Left** or **Insert Column to Right** to insert the first column.



Insert a column into table

3. Enter the column name in **Input** dialog box and click **OK** to confirm.
4. Repeat step 2 and 3 to create all columns.
5. Click **Insert Row Below** to insert a row. Note that a row can be inserted only when there is a column exist.



Insert a row into table

6. A row is created with empty cell(s). Fill in the cell(s) if necessary. Cell can be edited by double clicking or by pressing the **F2** key.
7. Repeat step 5 and 6 to create all rows.

8. Click **OK** to confirm editing.

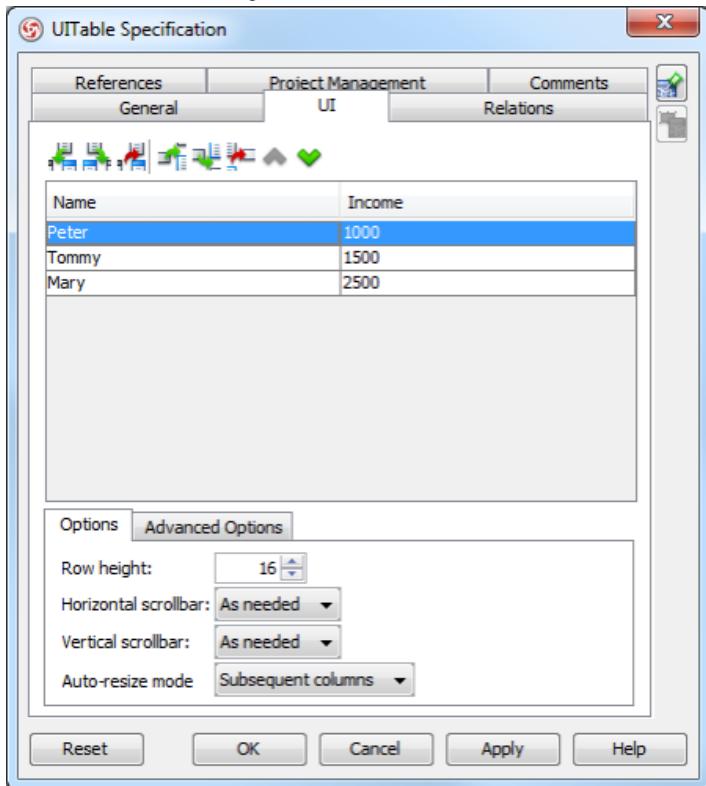
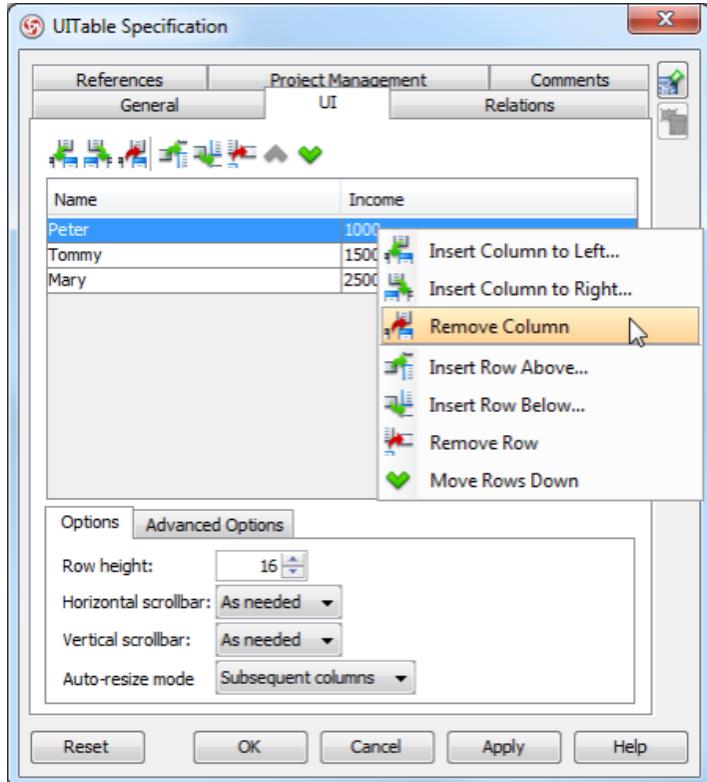


Table with data

Removing a column

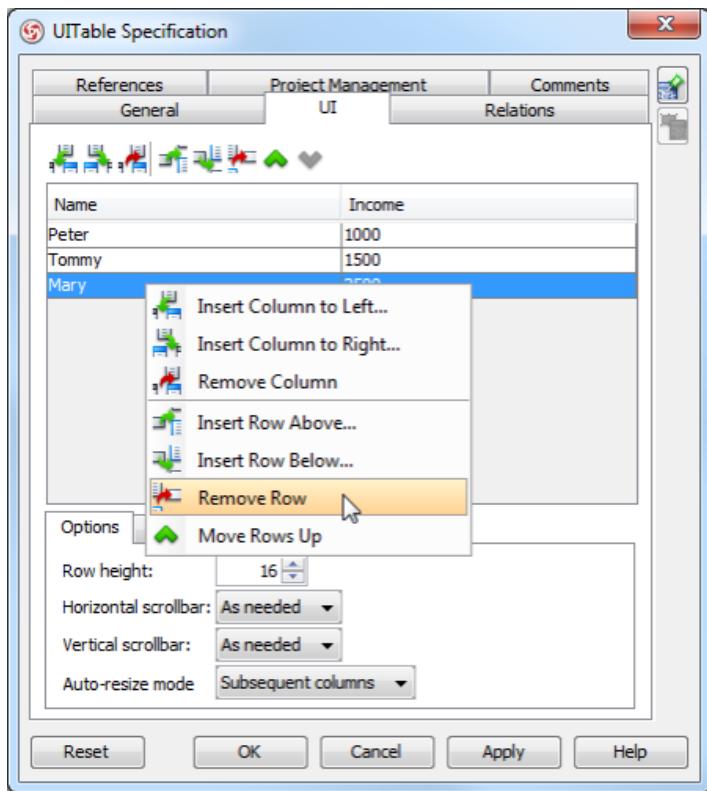
To remove a column in a table, open **UITable Specification** dialog box, open the **UI** tab, right click on a cell of a column and select **Remove Column** in the popup menu.



To remove a column in table

Removing a row

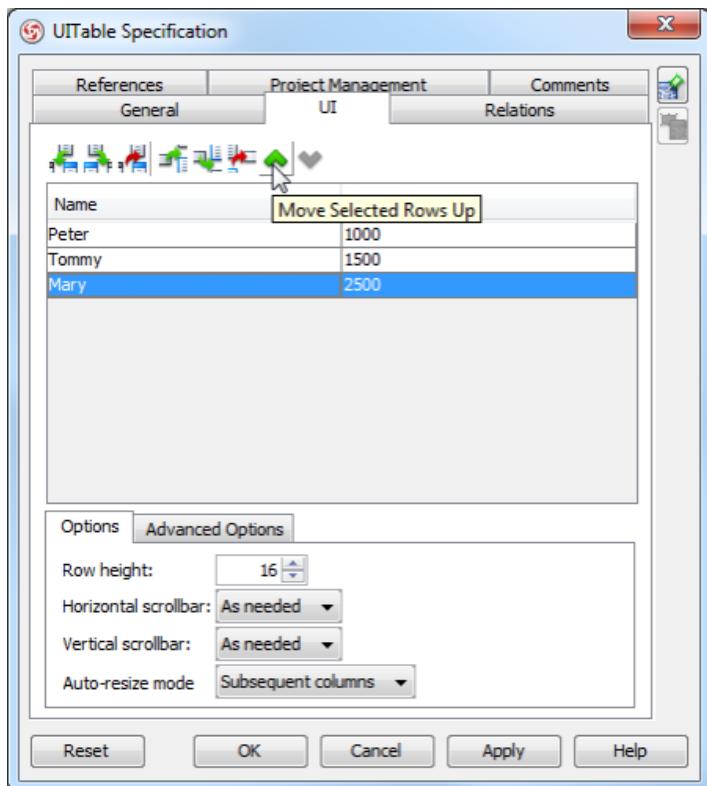
To remove a row in a table, open **UITable Specification** dialog box, open the **UI** tab, right click on a row and select **Remove Row** in the popup menu.



To remove a row in table

Reordering rows

To reorder rows, open **UITable Specification** dialog box, open the **UI** tab, select the row(s) to reorder, and click on the **Move Selected Rows Up** or **Move Selected Rows Down** button.



Moving a row in table upwards

Table properties

The appearance of table can be changed by editing its properties. To configure a table, right click on the table and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a table.

Property	Description
Row height	The height of rows in table.
	Determines when a horizontal scrollbar will appear in a table.
	Determines when a vertical scrollbar will appear in a table.
	Determines how the widths of columns will be affected when other columns are being resized. Off: Disable auto resizing Next column: When a column is being resized, all columns on the right and left of margin are updated Subsequent columns: When a column is being resized, all columns on the right are resized at the same time Last column : When a column is being resized, width of the right-most column are updated All columns: When a column is being resized, widths of all columns are changed
	Determines the color of grid in table.
	Determines the visibility of horizontal lines in table.
	Determines the visibility of vertical lines in table.

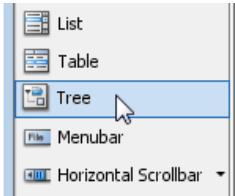
Description of table properties

Tree

A Tree is a component that displays a set of hierarchical data as an outline

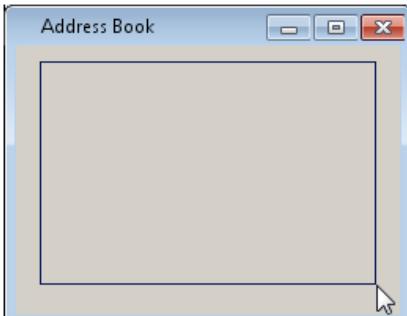
Creating a tree

1. Select the **Tree** tool from the diagram toolbar.



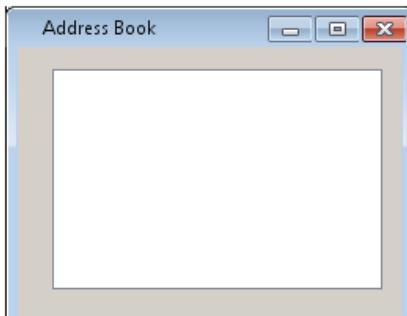
Selecting Tree from diagram toolbar

2. Press to set the position of the tree. Drag diagonally from the starting point to expand the tree.



Drag diagonally to expand the Tree

3. Release the mouse button to confirm the size of tree.



Tree is created

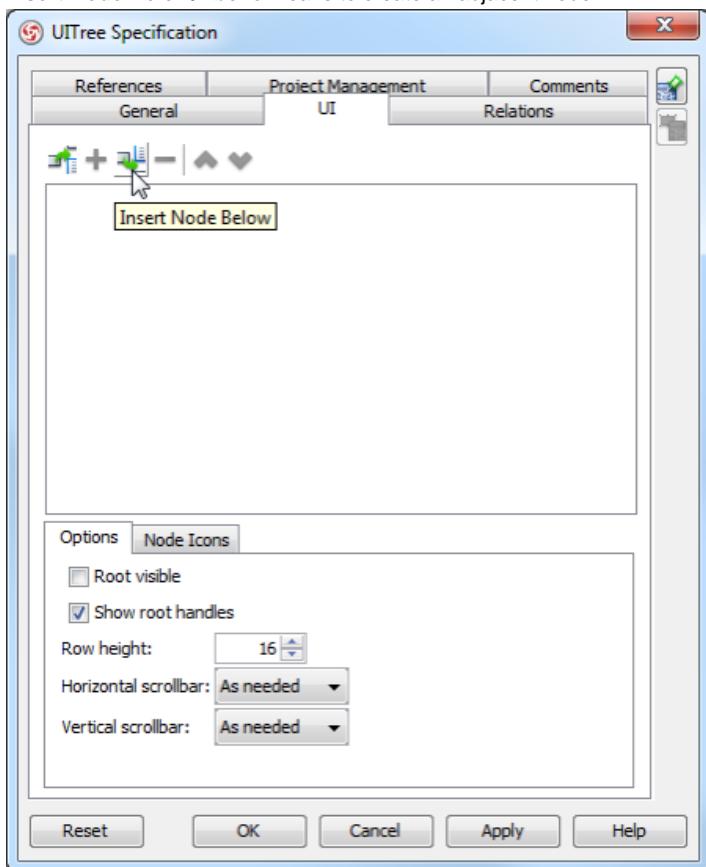
Editing tree contents

Basic setup

To insert columns and rows into a tree:

1. Right click on the tree and select **Open Specification...** from the popup menu.

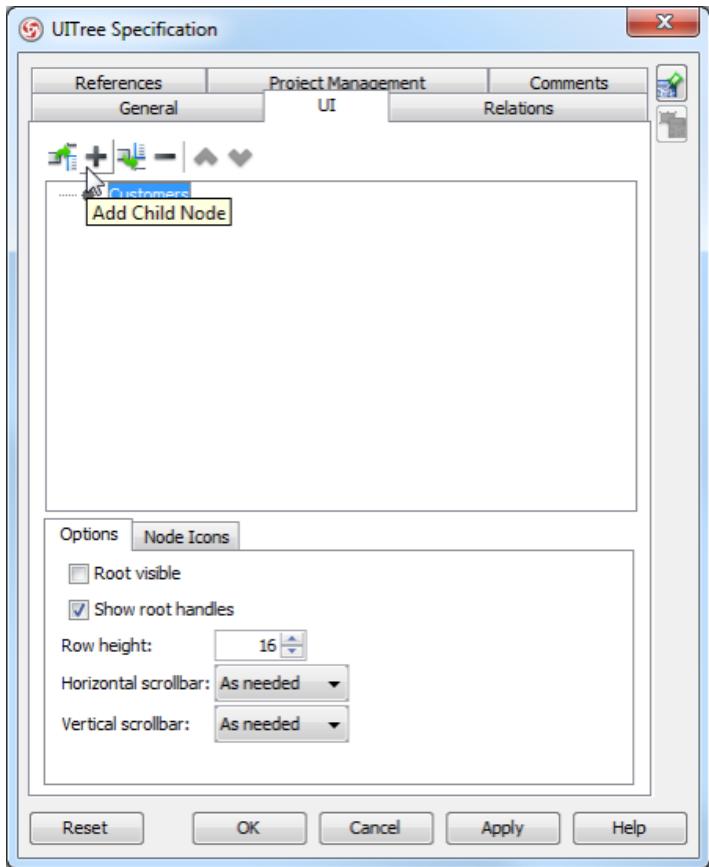
2. In the **UI** tab of the **UITree Specification** dialog box, click either **Insert Node Below** or **Insert Node Above** to insert the first column. Note that **Insert Node Below/Above** means to create an adjacent node.



Insert a node into tree

3. Enter the node name in **Input** dialog box and click **OK** to confirm.
4. To add a child node, select an existing node.

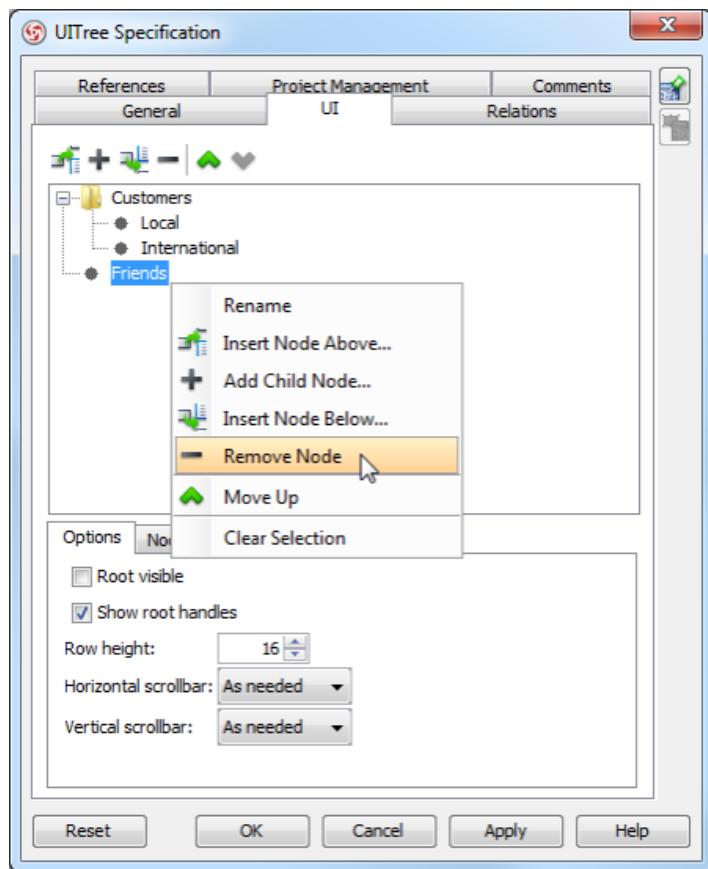
5. Click Add Child Node.



6. Enter the node name in **Input** dialog box and click **OK** to confirm.
7. Repeat step 2 to 6 to create all nodes.
8. Click **OK** to confirm editing.

Removing a node

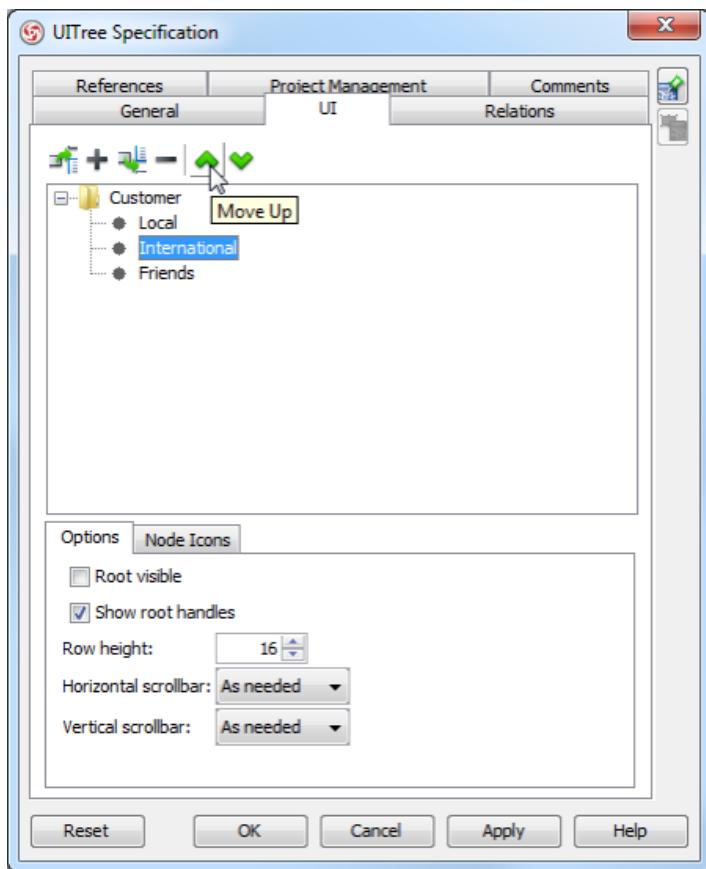
To remove a node in a tree, open the specification dialog box of the tree, open the **UI** tab, right click on the node(s) to remove and select **Remove Node** in the popup menu.



To remove a node in tree

Reordering nodes

To reorder nodes, open the specification dialog box of the tree, open the **UI** tab, select the node(s) to reorder, and click on the **Move Up** or **Move Down** button.



Moving a node in tree upwards

Tree properties

The appearance of tree can be changed by editing its properties. To configure a tree, right click on the tree and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a tree.

Property	Description
Root visible	Determines whether the root of tree will appear.
Show root handles	Determines whether the expand/collapse button for root will appear.
Row height	The height of nodes in tree.
Horizontal scroll bar	Determines when a horizontal scrollbar will appear in a tree.
Vertical scroll bar	Determines when a vertical scrollbar will appear in a tree.
Node icons	Set the image icon to appear for a node when at different state. Default node icon: Icon for all nodes when default icon for collapsed and expanded nodes are not set. Default collapsed icon: Icon for collapsed nodes. It overwrites the default node icon setting. Default expanded icon: Icon for expanded nodes. It overwrites the default node icon setting.

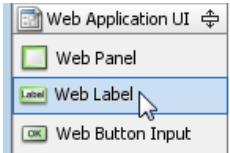
Description of tree properties

Web Label

A web label is a text component that appear on a screen. The text "User", "Password", "Address" appear on a registration form are examples of labels.

Creating a label

1. Select the **Web Label** tool from the diagram toolbar.



Selecting **Web Label**
from diagram toolbar

2. Click on a container (e.g. a Web Panel) or diagram background to create a web label.
3. Enter the label caption and press **Enter** to confirm the caption of label.



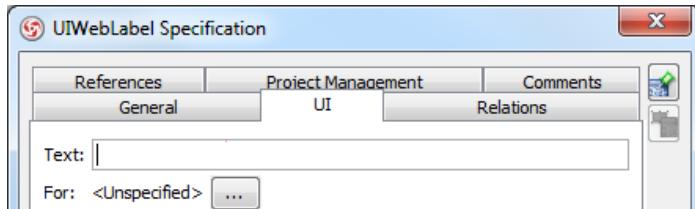
User name:



A label is created

Label properties

The appearance of web label can be changed by editing its properties. To configure a web label, right click on the label and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a label.



UIWebLabel Specification's UI tab

Property	Description
Text	Text is the text appear in a label.
For	Specifies which form element a label is bound to.

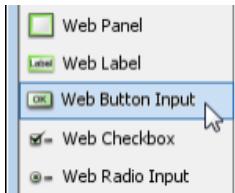
Description of web label properties

Web Button Input

A web button is a component on a web form which triggers certain action when being clicked. The button "Validate" that appear on a registration form is an example of web button.

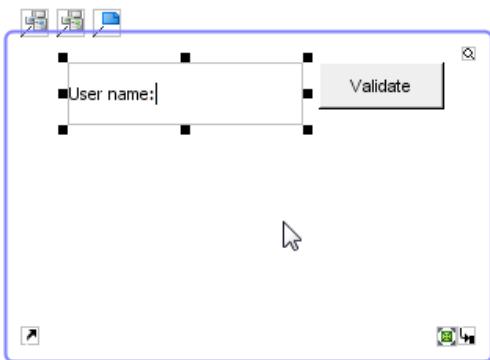
Creating a web button

1. Select the **Web Button Input** tool from the diagram toolbar.



Selecting **Web Button Input**
Input from diagram toolbar

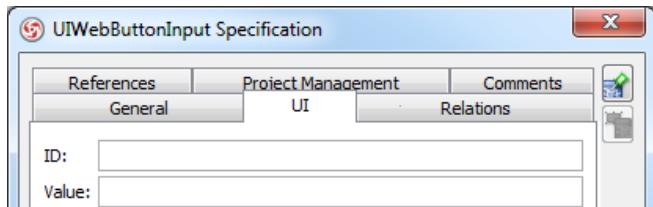
2. Click on a container (e.g. a web panel) or diagram background to create a button.
3. Enter the button caption and press **Enter** to confirm the caption of button. This creates the button.



A button is created

Web button input properties

The appearance of web button can be changed by editing its properties. To configure a button, right click on the button and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a button.



UIWebButtonInput Specification's UI tab

Property	Description
ID	ID is the ID of web button.
Value	Value is the text appear on a web button input.

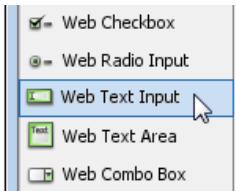
Description of web button properties

Web Text Input

Web Text Input is a component that allows the editing of a single line of text. Editable text fields in a registration form are typical examples of text inputs.

Creating a web text input

1. Select the **Web Text Input** tool from the diagram toolbar.



Selecting **Web Text Input**
from diagram toolbar

2. Press to set the position of the text input. Drag diagonally from the starting point to expand the text input.

User name: Validate

Nickname:

Drag diagonally to expand the text input

3. Release the mouse button to confirm the size of text input.

4. Enter the text input text and press **Enter** to confirm the title of text input.

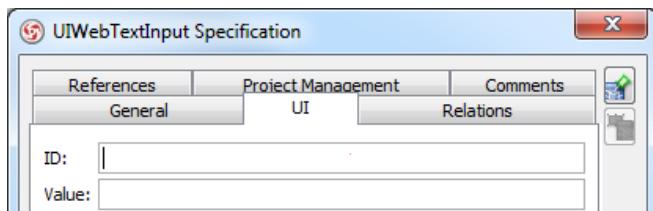
User name: Validate

Nickname:

A text input is created

Text input properties

The appearance of text input can be changed by editing its properties. To configure a text input, right click on the text input and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a text input.



UIWebTextInput Specification's UI tab

Property	Description
ID	ID refers to the ID of text input on a form.
Value	Value refers to the text that appear in the text input.

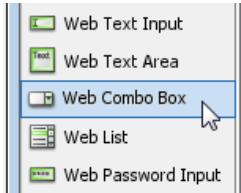
Description of web text input properties

Web Combo Box

Web Combo box is a component that combines a button and a drop-down list. User can select a value from the drop-down list to make a decision.

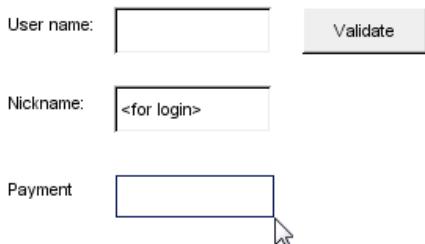
Creating a web combo box

1. Select the **Web Combo Box** tool from the diagram toolbar.



Selecting **Web Combo Box** from diagram toolbar

2. Press to set the position of the combo box. Drag diagonally from the starting point to expand the combo box.



Drag diagonally to expand the Combo box

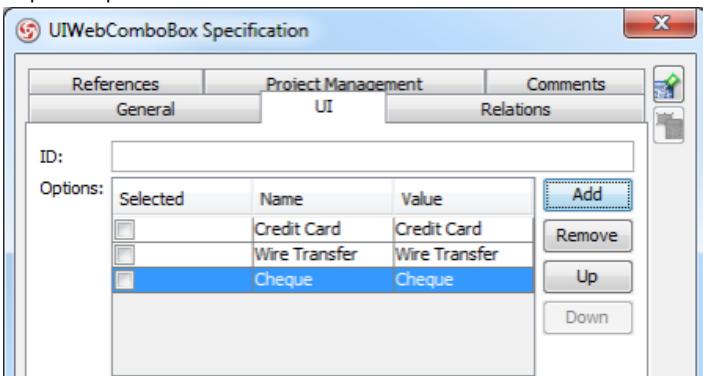
3. Release the mouse button to confirm the size of combo box.



Combo box is created

Editing combo box values

1. Right click on the combo box and select **Open Specification...** from the popup menu.
2. In the **UI** tab of the **UIWebComboBox Specification** dialog box, click **Add**.
3. Enter the value for of item and click **OK** to confirm.
4. Repeat step 2 and 3 to create all the combo box items.

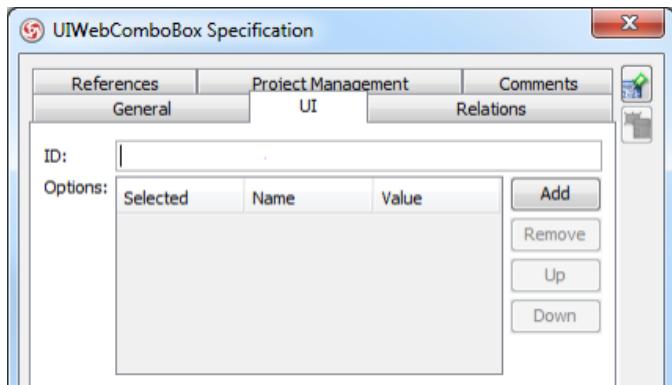


Combo box items are created

5. Click **OK** to confirm editing.

Combo box properties

The appearance of web combo box can be changed by editing its properties. To configure a combo box, right click on the combo box and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a combo box.



UIWebComboBox Specification's UI tab

Property	Description
ID	ID of the combo box in web form.
Options	The items that can be selected by users. You can optionally mark the default item's Selected to be true.

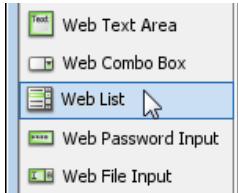
Description of web combo box properties

Web List

List is a component that lists all available selection in rows, and allow the selection of values in it.

Creating a list

1. Select the **Web List** tool from the diagram toolbar.



Selecting **Web List**
from diagram toolbar

2. Press to set the position of the list. Drag diagonally from the starting point to expand the list.

User name: Validate

Nickname: <for login>

Payment

Interest

Drag diagonally to expand the List

3. Release the mouse button to confirm the size of list.

User name: Validate

Nickname: <for login>

Payment

Interest

List is created

Editing list values

1. Right click on the list and select **Open Specification...** from the popup menu.
2. In the **UI** tab of the **UIWebList Specification** dialog box, click **Add**.
3. Enter the value for of item in **Input** dialog box and click **OK** to confirm.
4. Repeat step 2 and 3 to create all the list items.

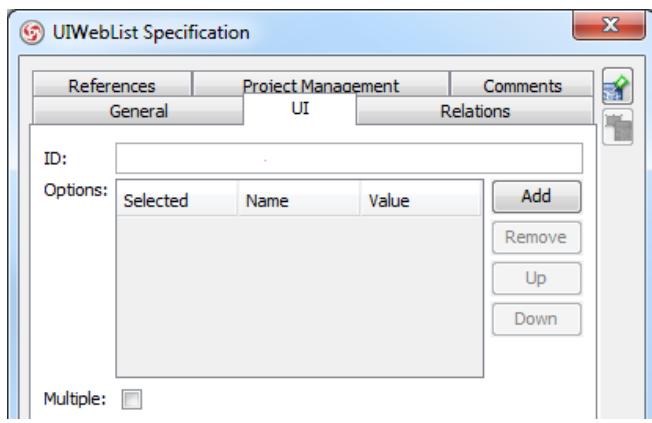
5. Click **OK** to confirm editing.

User name:	<input type="text"/>	<input type="button" value="Validate"/>
Nickname:	<input type="text" value="<for login>"/>	
Payment	<input type="button" value=""/>	
Interest	Sports Finance Gardening Car	

List with values

List properties

The appearance of web list can be changed by editing its properties. To configure a list, right click on the list and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a list.



UIWebList Specification's UI tab

Property	Description
ID	ID of list in form.
Options	The items that can be selected by users. You can optionally mark the default item's Selected to be true.
Multiple	Determine whether the list supports single or multiple item selection.

Description of web list properties

Web Password Input

Web Password Input is a component that allows the entering of password. Text entered will be shown as asterisks, which act as a mask of password.

Creating a web password input

1. Select the **Web Password Input** tool from the diagram toolbar.



Selecting **Web Password Input** from diagram toolbar

2. Press to set the position of the password input. Drag diagonally from the starting point to expand the password input.

A screenshot of a form creation interface. It includes fields for 'User name' (text box), 'Nickname' (text box containing '<for login>'), 'Password' (text box with a blue border and a vertical resize handle on the right), 'Payment' (button with a dropdown arrow), and 'Interest' (list box with items: Sports, Finance, Gardening, Car). A mouse cursor is shown dragging from the bottom-left corner of the 'Password' text box towards the top-right, indicating the expansion of its size.

Drag diagonally to expand the password input

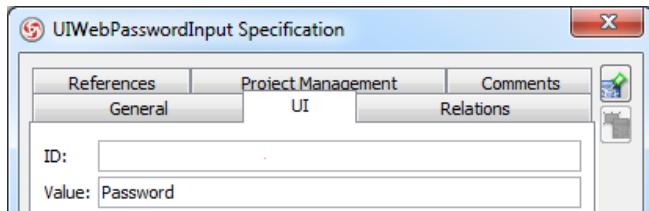
3. Release the mouse button to confirm the size of password input and press **Enter** to confirm the title of password input.

A screenshot of the same form after step 3. The 'Password' field now contains six asterisks ('*****') and has a larger rectangular shape. The other fields ('User name', 'Nickname', 'Payment', 'Interest') remain unchanged.

A password input is created

Password input properties

The appearance of password input can be changed by editing its properties. To configure a password input, right click on the password input and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a password input.



UIWebPasswordInput Specification's UI tab

Property	Description
ID	ID refers to the ID of password input on a web form.

ID ID refers to the ID of password input on a web form.

Value Value refers to the underlying password of the password input.

Description of web password properties

Web File Input

Web File Input is a component that involve a text field which represent file field, and a **Browse** button for locating the file to select.

Creating a web file input

1. Select the **Web File Input** tool from the diagram toolbar.



Selecting **Web File Input**
from diagram toolbar

2. Press to set the position of the file input. Drag diagonally from the starting point to expand the file input.

User name: Validate

Nickname:

Password

Payment

Interest

Logo

Drag diagonally to expand the file input

3. Release the mouse button to confirm the size of file input and press **Enter** to confirm creation.

User name: Validate

Nickname:

Password

Payment

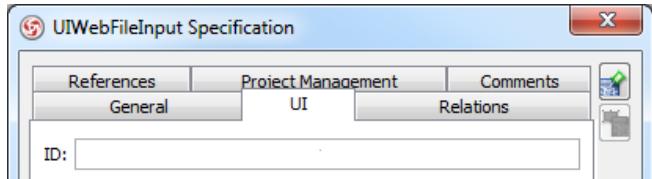
Interest

Logo

A file input is created

File input properties

The appearance of web file input can be changed by editing its properties. To configure a file input, right click on the file input and select **Open Specification...** from the popup menu. The **UI** tab is where user can configure a file input.



UIWebFileInput Specification's UI tab

Property	Description
ID	ID refers to the ID of file input on a web form.

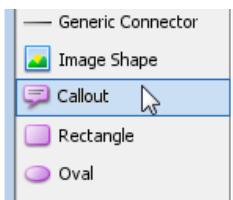
Description of web file input properties

Annotating the UI design with callout shape

Annotation is an extra information, such as comments, notes, explanation, or other type of external mark that describes a model. It is an effective way to edit and review work in a work group environment. Designers often need to be able to jot down information which should not be part of the model itself. In this situation, annotation helps. For instance, annotation can be added to a User Interface design to specify validation checking.

To annotate a User Interface design:

1. Select **Callout** from the diagram toolbar. Note that the **Callout** is under the **Common** category.



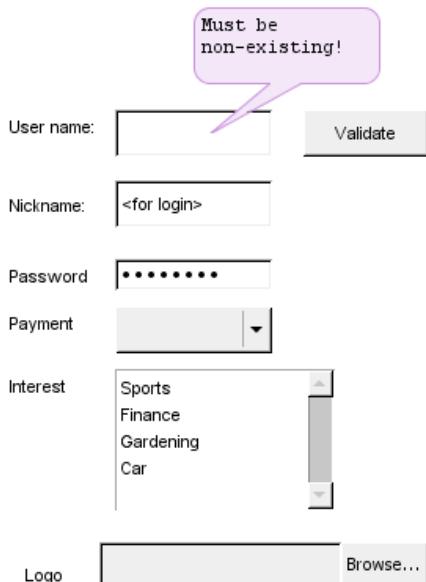
Selecting **Callout**
from diagram toolbar

2. Click on the diagram to create a Callout shape. Enter the content of annotation. Formatings can be applied through the buttons at the bar above the callout shape.



Enter the content of Callout shape

3. We need to attach the pointer of Callout shape to the User Interface component that need to be annotated. Drag on the end of Callout shape's pointer towards the User Interface component.
4. Release the mouse button to confirm the pointer position when reached the component.
5. Click on the diagram background to confirm editing.



Using Callout shape to annotate a text field

6. Repeat the above steps to annotate the whole design with Callout shapes.

The diagram illustrates a user interface form with various input fields and a validation button. Annotations are used to provide feedback for each field:

- User name:** A text input field with a placeholder. A callout bubble points to it with the text "Must be non-existing!".
- Nickname:** A text input field containing placeholder text. A callout bubble is positioned near the right edge of the field.
- Password:** A password input field showing masked text. A callout bubble points to it with the text "At least 8 chars long."
- Payment:** A dropdown menu with a visible list of options: Sports, Finance, Gardening, and Car. A callout bubble is attached to the bottom of the dropdown menu.
- Interest:** A dropdown menu with a visible list of options: Sports, Finance, Gardening, and Car. A callout bubble is attached to the bottom of the dropdown menu.
- Logo:** A file upload input field with a "Browse..." button. A callout bubble is attached to the right side of the input field.

User Interface diagram with annotations

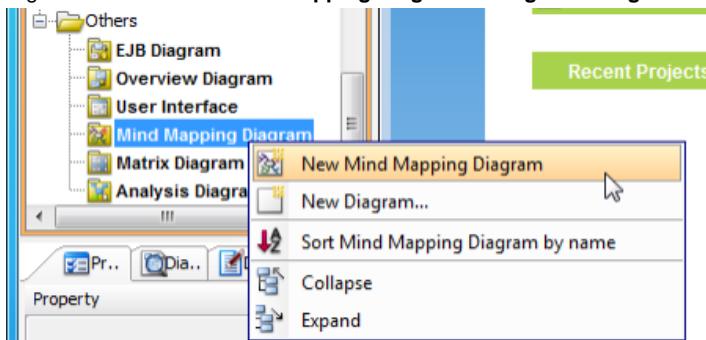
Drawing mind mapping diagram

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

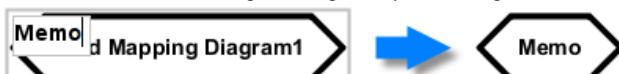
To create a mind mapping diagram:

1. Right click on the node **Mind Mapping Diagram** in **Diagram Navigator** and select **New Mind Mapping Diagram** in popup menu.



To create a mind mapping diagram

2. This creates a mind mapping diagram with a central idea node appear 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.

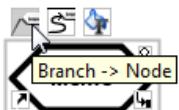


Naming central idea node

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**.



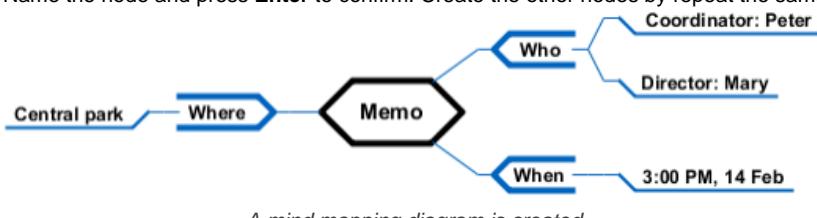
To create a node

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 repeat the same steps.



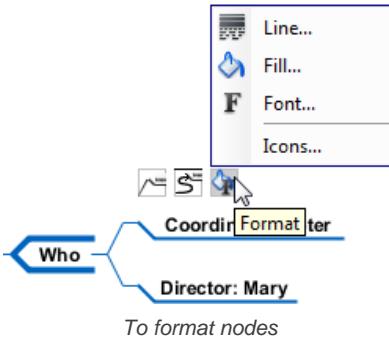
A mind mapping diagram is created

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

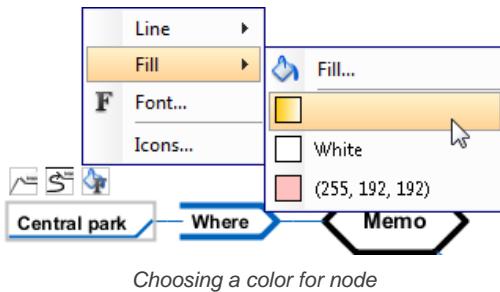
1. Select the node(s) that you want to format. Multiple node selection can be made by a range selection or by pressing the **Ctrl** key and select the nodes subsequently.
2. Move the mouse pointer to a node within the selection. Click on the **Format** resource icon.



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)

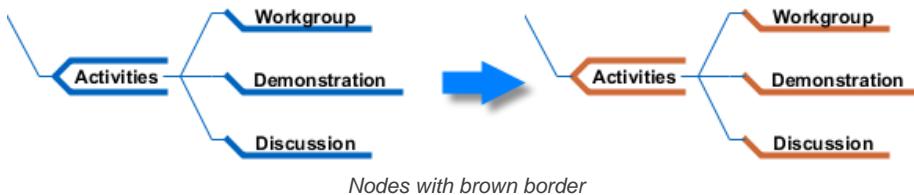
Once you have confirmed your selection, your choice will be memorized. When you want to apply the settings on other nodes, you can select the new nodes, re-open the same popup menu, and select the setting through the popup menu.



Choosing a color for node

Line

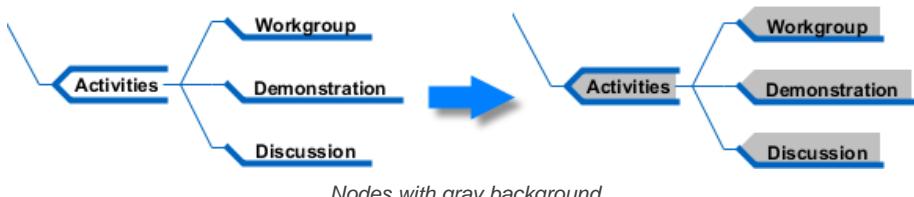
Line settings control the appearance of border around node(s). You can adjust the style (e.g. dash, solid), the weight, which is the thickness of line, the color and the level of transparency.



Nodes with brown border

Fill

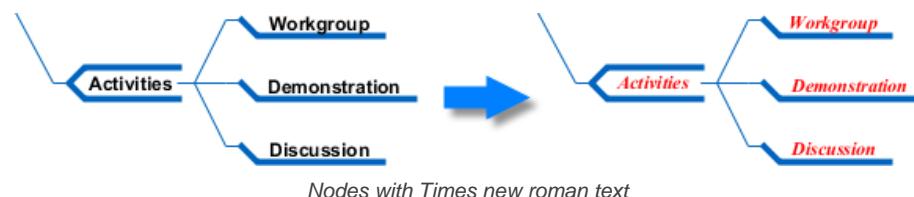
Fill settings control the background color of node(s). You can apply solid and gradient colors, as well as to control the transparency.



Nodes with gray background

Font

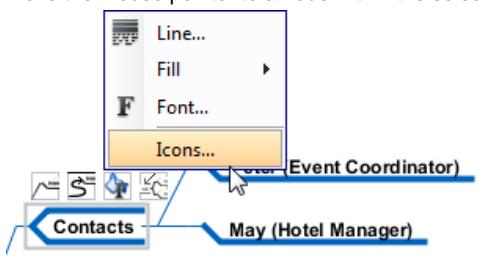
Font settings control the font style, size, type and color of text appear on a node.



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 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.



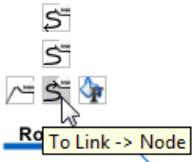
A node with icons

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 be 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 are related, and the relationship is meaningful, 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.

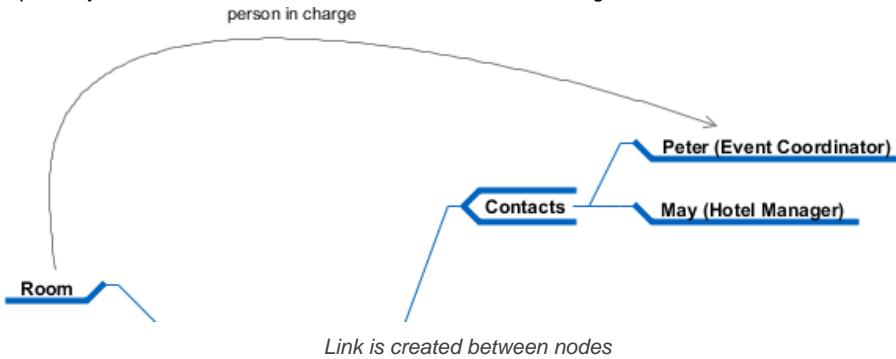


To link to another node

2. Drag to the target node and release the mouse button.

NOTE: Unlike the usage of 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.

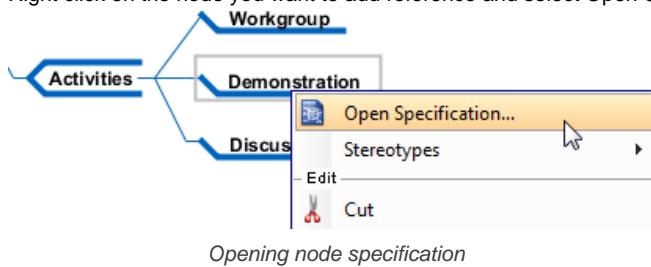


Reference to resources

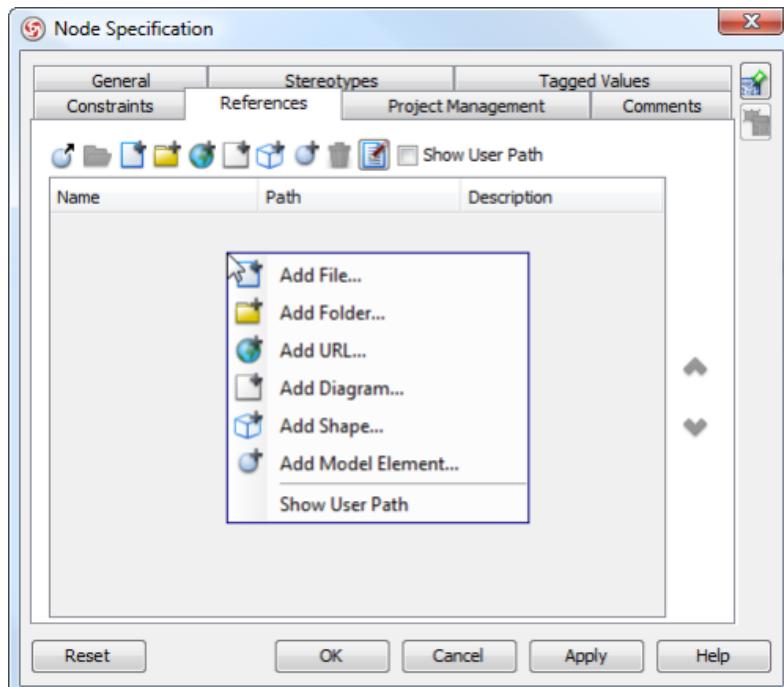
You can add references to node, to reference to both internal and external resources such as a shape, a diagram, a file, a URL, etc. For example, to make a node *Prepare Agenda* link to a document of agenda template. This makes a mind map more informative by providing additional information from a mind 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.



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 popup menu.



Going to 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.

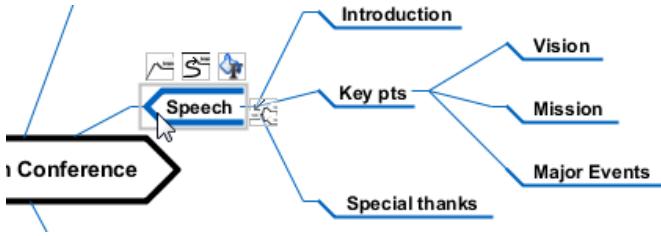
Once a reference has been added, you can open it from the **References** tab by right clicking on it and selecting **Open...** from the popup menu.

Opening a referenced file

Relocating a branch

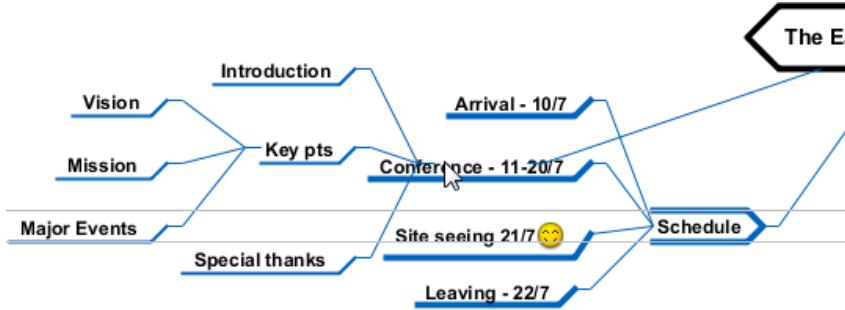
In case a branch of nodes is mis-positioned, you can reposition it to under another node through drag and drop. Here are the steps:

1. Press on the pointer end of the first node of a branch that you want to reposition.



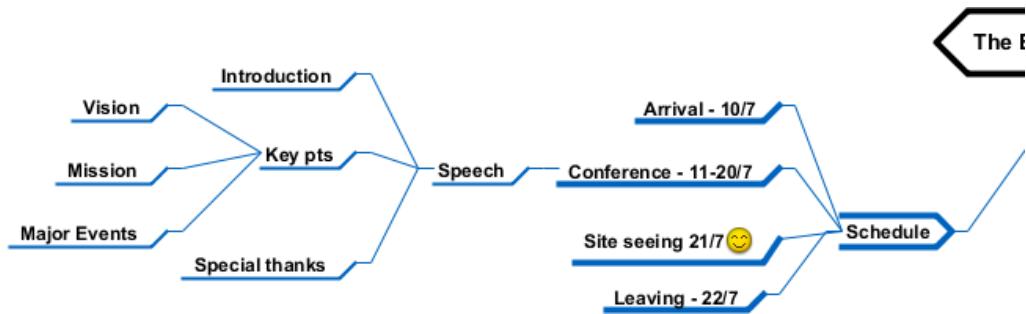
Pressing on the pointer end of a node

2. Drag to node that you want to move the branch to.



Dragging over the target node

3. Release the mouse button.



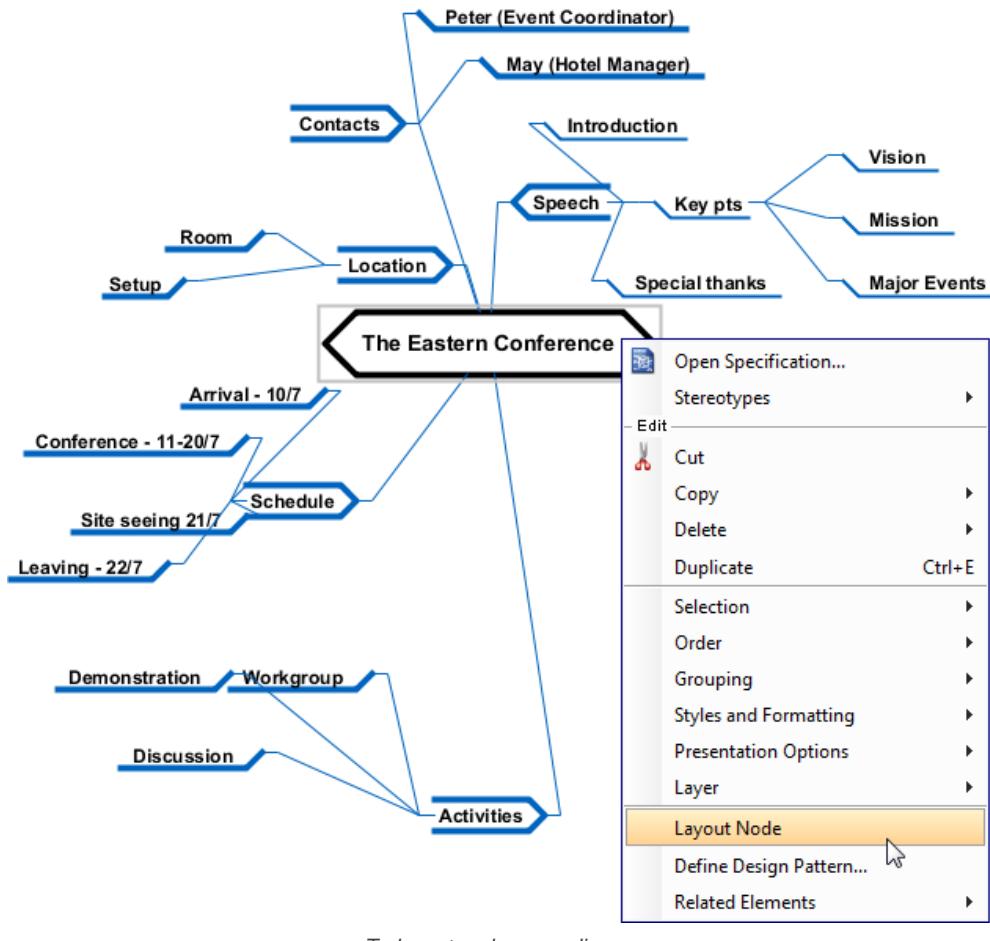
Mouse button released

Layout diagram

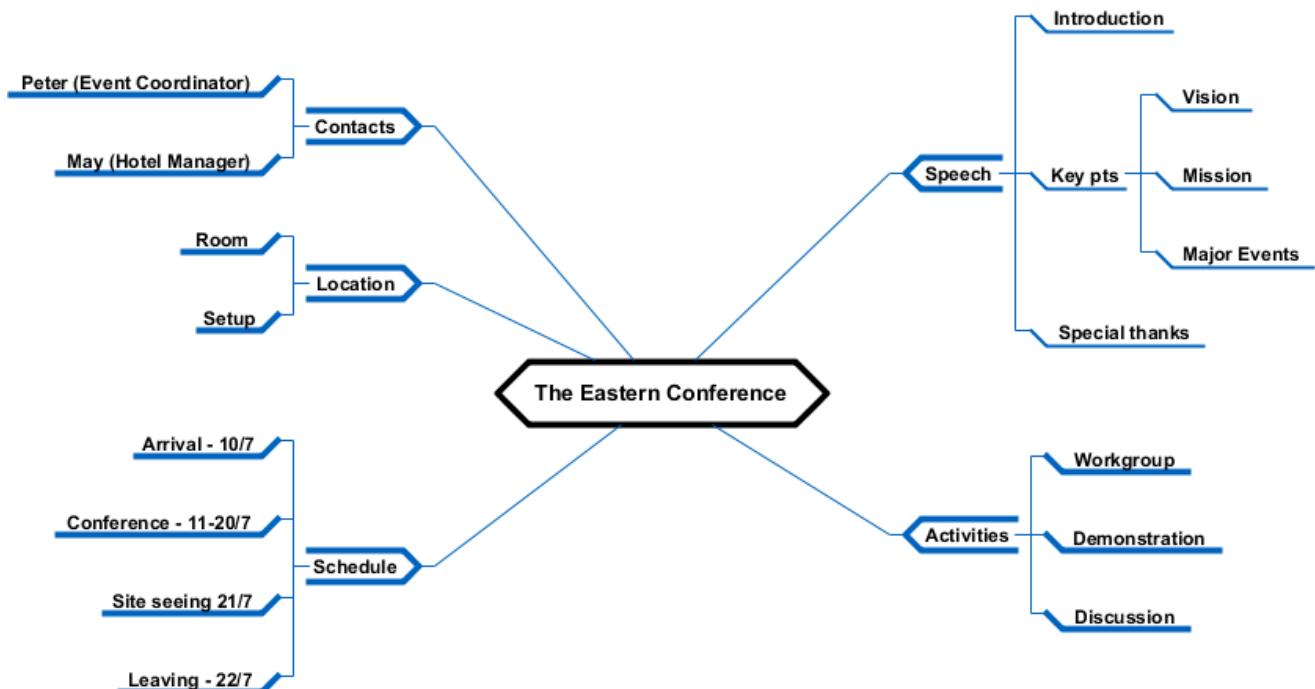
In a mind mapping diagram, ideas are stretching out across, which leads difficulties in tracing nodes with different ideas due to the unorganized nodes. It will be time consuming to rearrange the idea nodes manually. This also affects our brainstorming procedure by caring the tidiness of diagram. By 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 drew. 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.



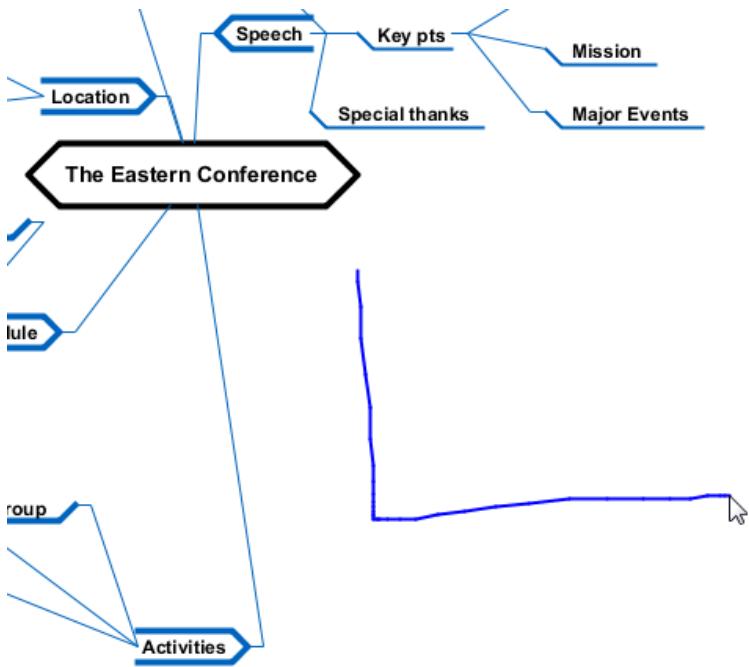
To layout nodes on a diagram



Result of diagram based layout - all nodes are layout-ed

Using resource-centric interface

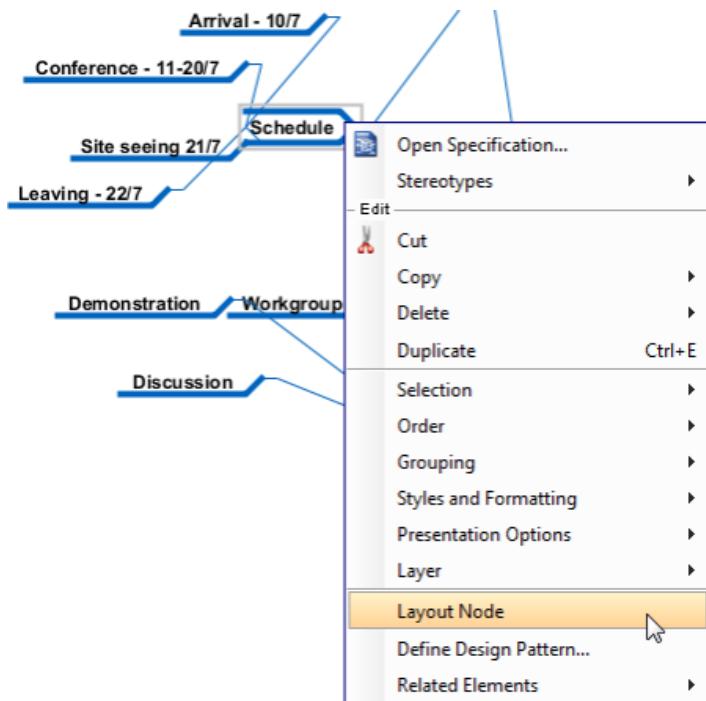
Mouse gesture enables you to layout shapes on a diagram through the movement of mouse. To perform a layout with mouse gesture, right press on the diagram background, sketch a "L" like gesture path and release the mouse button to execute layout.



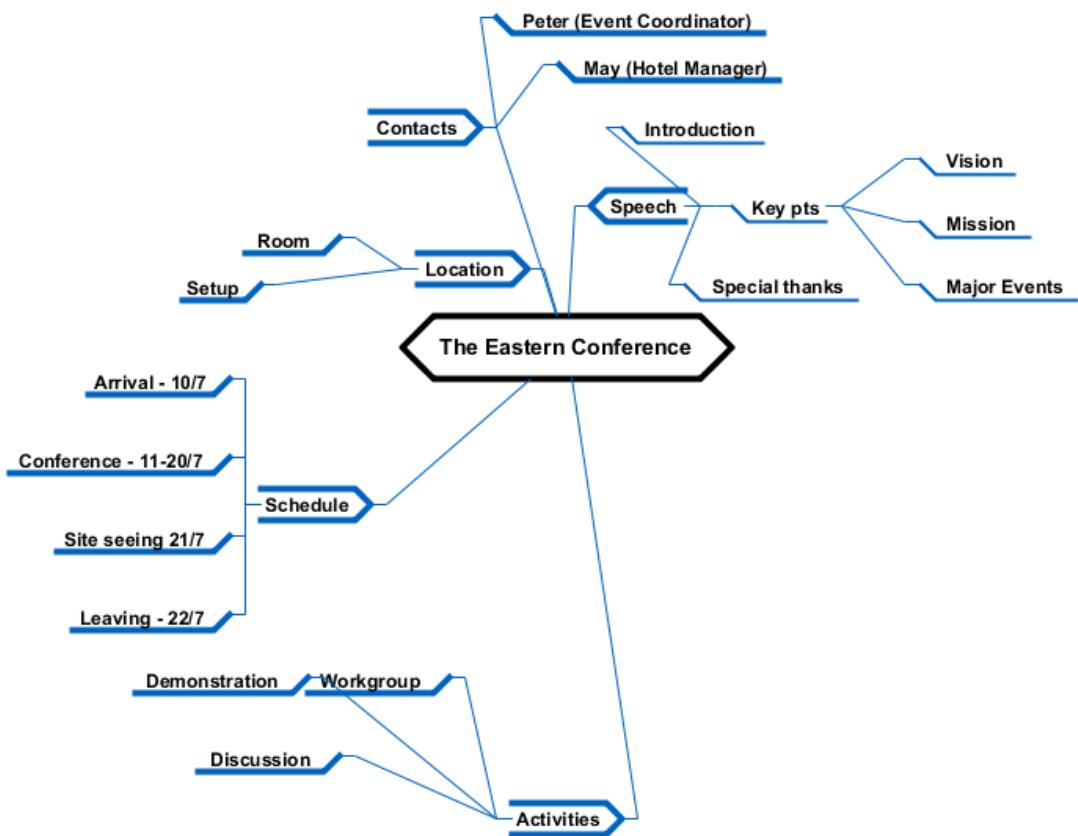
Layout diagram with mouse gesture

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

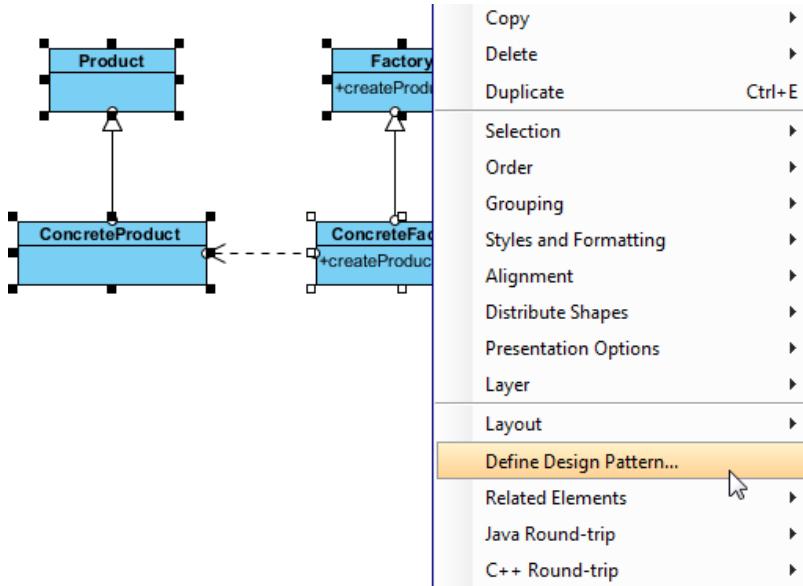
Defining design pattern

In VP-UML, 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.

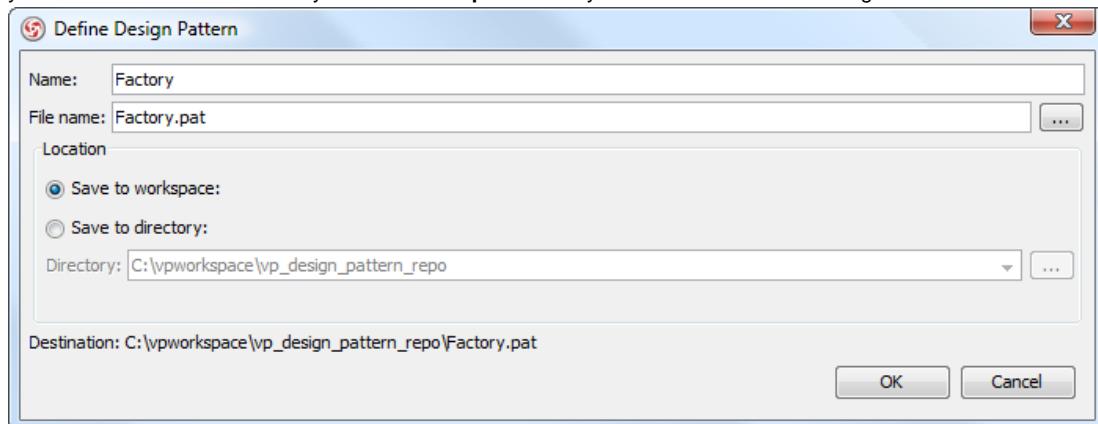
Defining design pattern

1. In the diagram where the pattern was drawn, 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

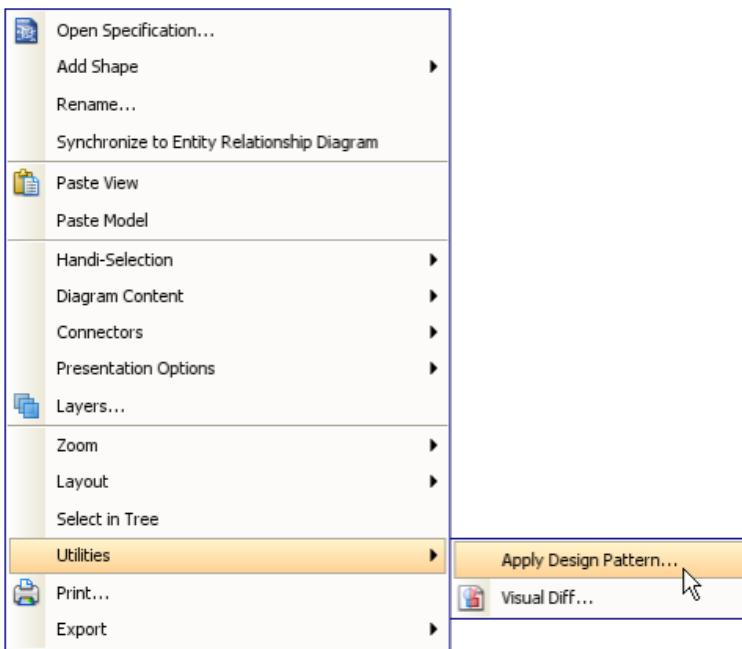
3. A design pattern is needed to save as a file. In the **Define Design Pattern** dialog, specify the name and file name for the pattern, with **.pat** as extension. You can save the pattern file to workspace for ease of sharing with other projects that will be opened in current workspace. Besides, you 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

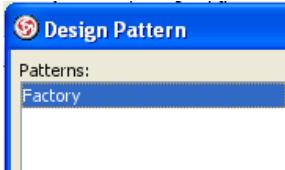
Applying design pattern

1. Create a class diagram
2. Right click on diagram, select **Utilities > Apply Design Pattern...** from the popup menu.



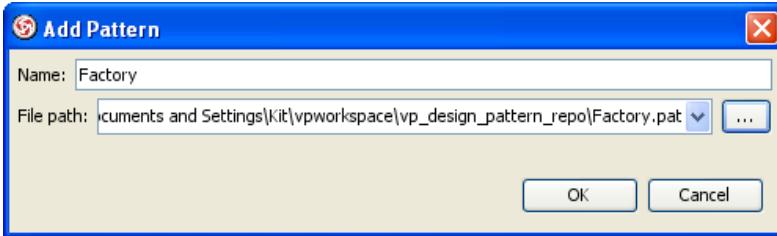
Apply design pattern

3. In **Design Pattern** dialog, you can see a list of defined patterns, select *Factory* from the list.



Select pattern

If you have a .pat file, click **Add** button to import into the list.



Add pattern

4. On the right hand side of the dialog, you can see the image of the pattern. Fill in the name of classes and operations on the bottom of dialog.

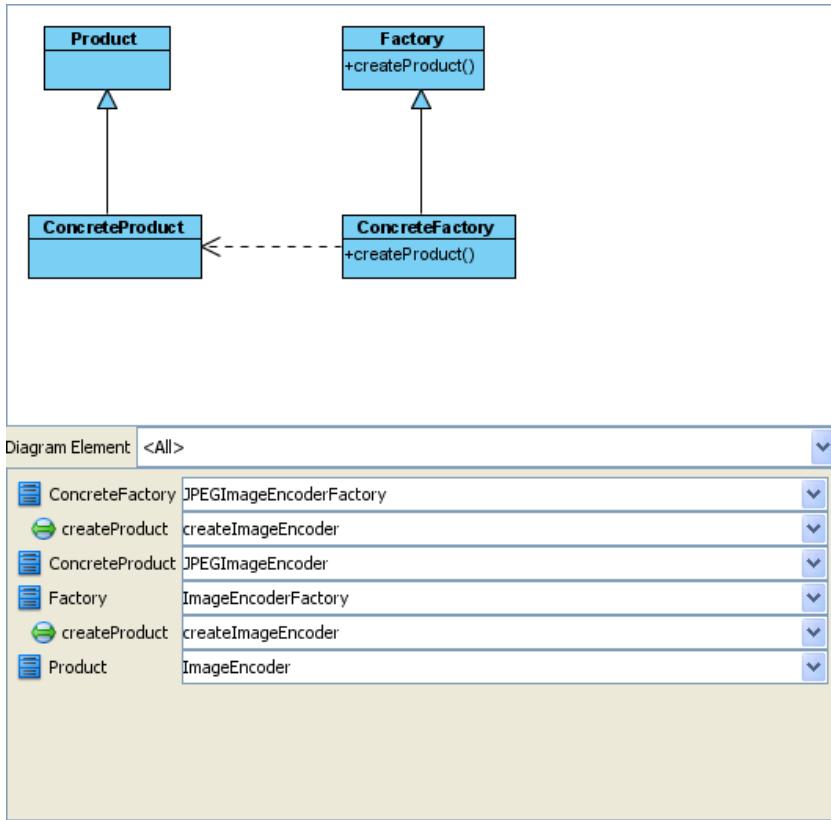
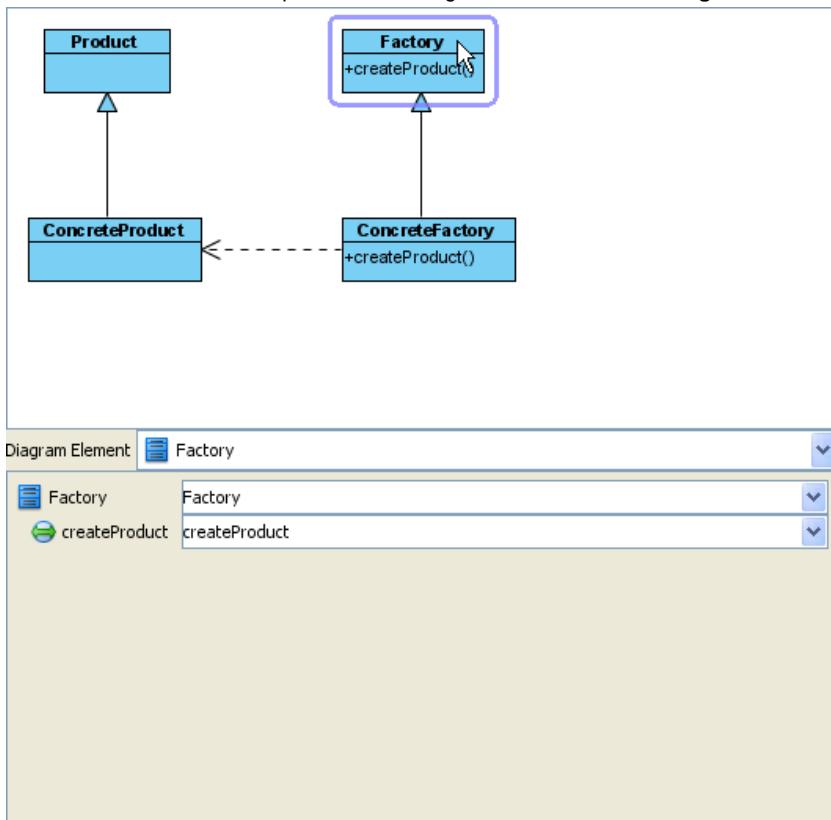


Diagram Element <All>

ConcreteFactory	JPEGImageEncoderFactory
createProduct	createImageEncoder
ConcreteProduct	JPEGImageEncoder
Factory	ImageEncoderFactory
createProduct	createImageEncoder
Product	ImageEncoder

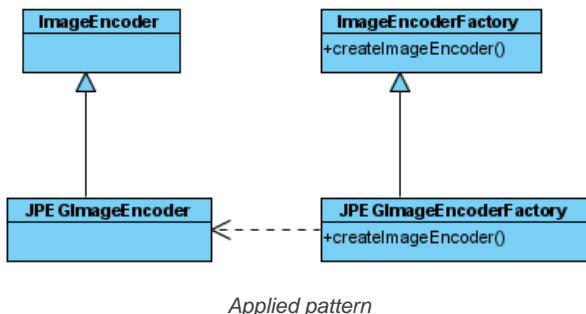
Fill in values

You can also click on the shape or select a diagram element from the **Diagram Element** combo box to filter the list.



Filter pattern element

5. Finally, click **OK**. The pattern will be applied to the diagram.

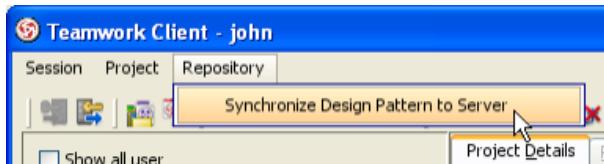


Synchronize design pattern with teamwork server

Teamwork server can synchronize design patterns defined and saved to workspace, you can then share the design patterns with your team members. This feature is available to Visual Paradigm Teamwork Server, Subversion, CVS and Perforce.

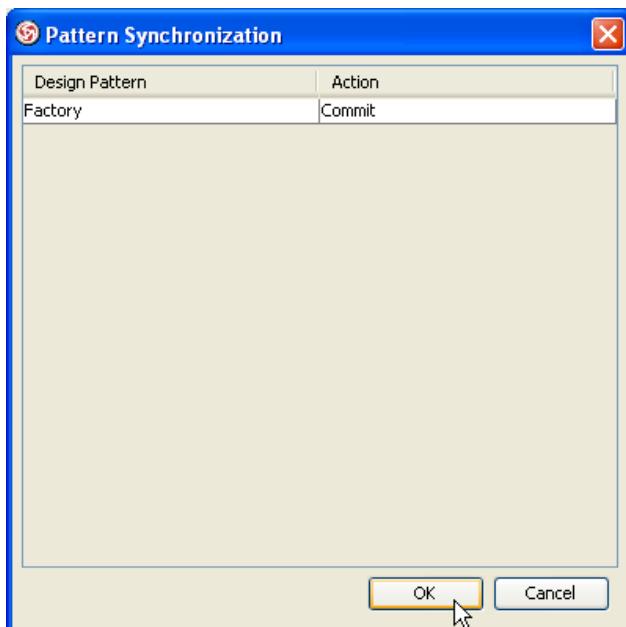
Synchronize local design pattern to server

1. Open **Teamwork Client** from Toolbar, or **Tools > Teamwork > Open Teamwork Client...** from the main menu.
2. Login to the teamwork server.
3. From the **Teamwork Client** dialog, select **Repository > Synchronize Design Pattern to Server** from the menu.



Synchronize Design Pattern to Server

4. You'll see **Pattern Synchronization** dialog, verify the design patterns and actions and click **OK** to continue. The pattern will be committed to teamwork server.



Commit pattern

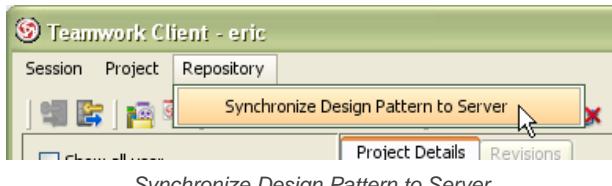
Synchronize design pattern from server

1. Start VP-UML with another user, using a different workspace.
2. Create a class diagram.
3. Right click on diagram, select **Utilities > Apply Design Pattern...** from the popup menu.
4. From **Design Pattern** dialog, you'll find the patterns list is empty because design patterns didn't exists in workspace.



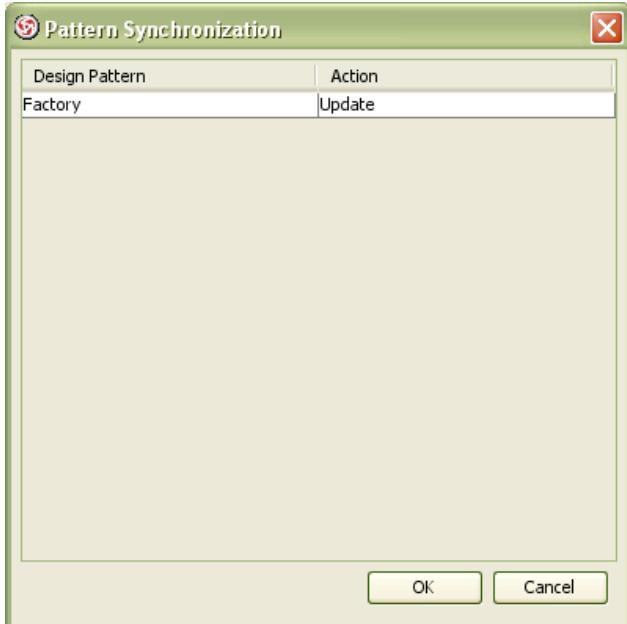
Apply design pattern

5. Open **Teamwork Client** and select **Repository > Synchronize Design Pattern to Server**.



Synchronize Design Pattern to Server

6. You'll see a **Pattern Synchronization** dialog showing that patterns are available for update from teamwork server. Click **OK** and the pattern will be updated from teamwork server to workspace.



Update pattern

7. Apply design pattern again, the design pattern is available in the list. You can now select the pattern and apply to your project.



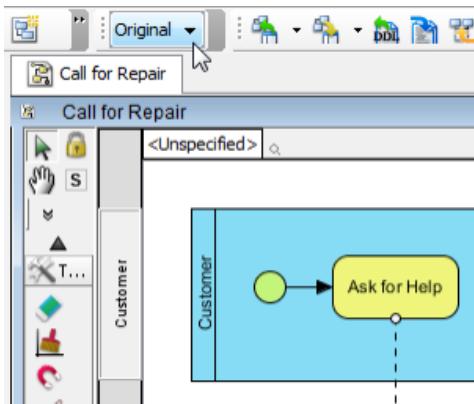
Apply design pattern again

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 VP-UML project content. While we have applied certain language in naming and documenting 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.

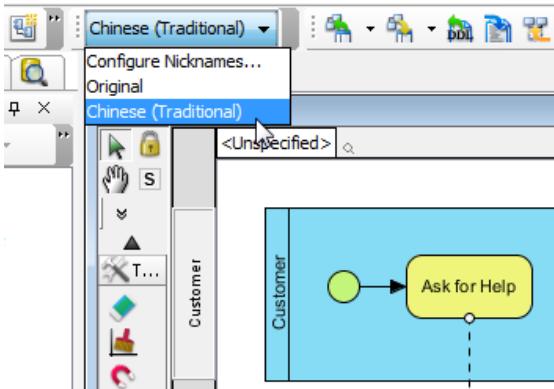
One model element can have one **Original** name and multiple nicknames, and the same for documentation. With nickname, you can define and view different names without affecting the original name of model elements. You can disable the effect of nickname anytime by switching to **Original** nickname. 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.

The following screenshot shows the **Business Process Diagram** in original name:



Original name

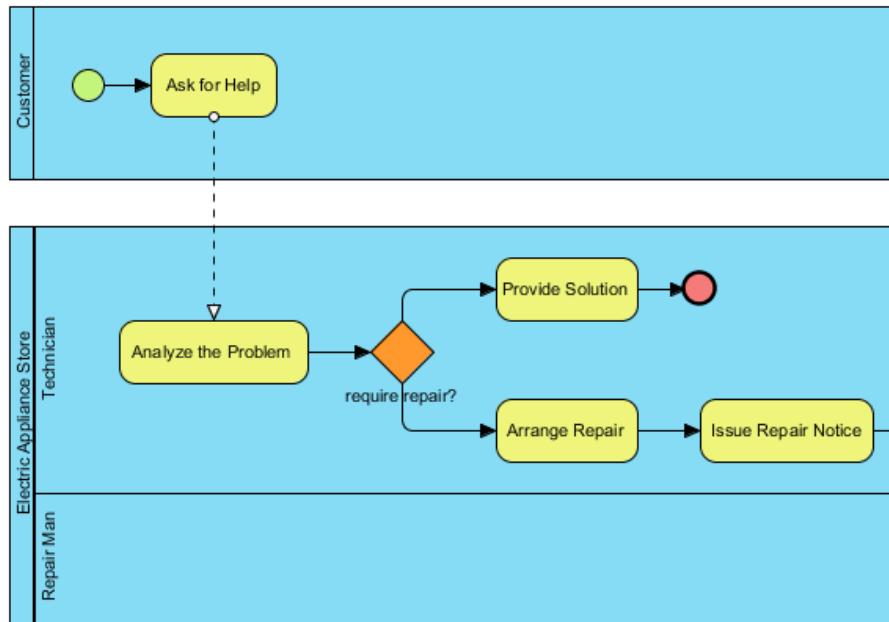
After you define a nickname and rename model element, switching nickname will refresh the diagram with the selected nickname.



Selecting a nickname

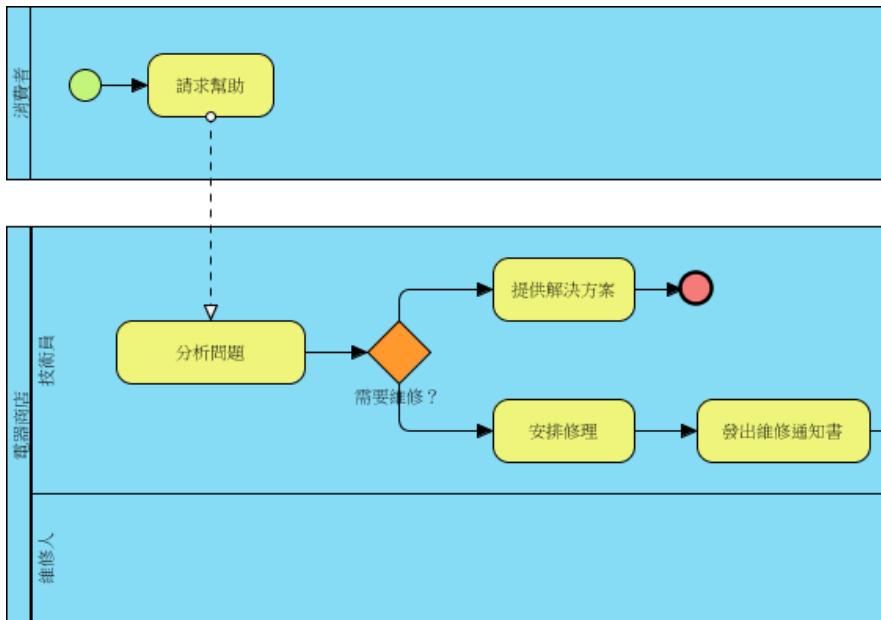
Multi-national team

If you are working in a team and your members using different languages, you can define model elements name and documentation in multiple languages. Each member can choose their own language for modeling or view diagrams. The following example demonstrate the **Business Process Diagram** in English and French:



Original version of a business process

You can create a *French nickname* and rename the model elements:

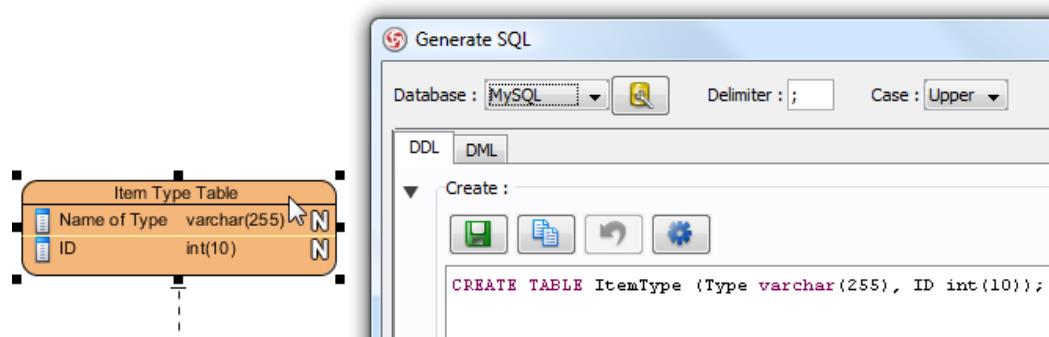


Chinese version of a business process

Now you can switch between English (Original) and Traditional Chinese anytime, or even creating more nicknames.

Increase readability of entity relationship diagram

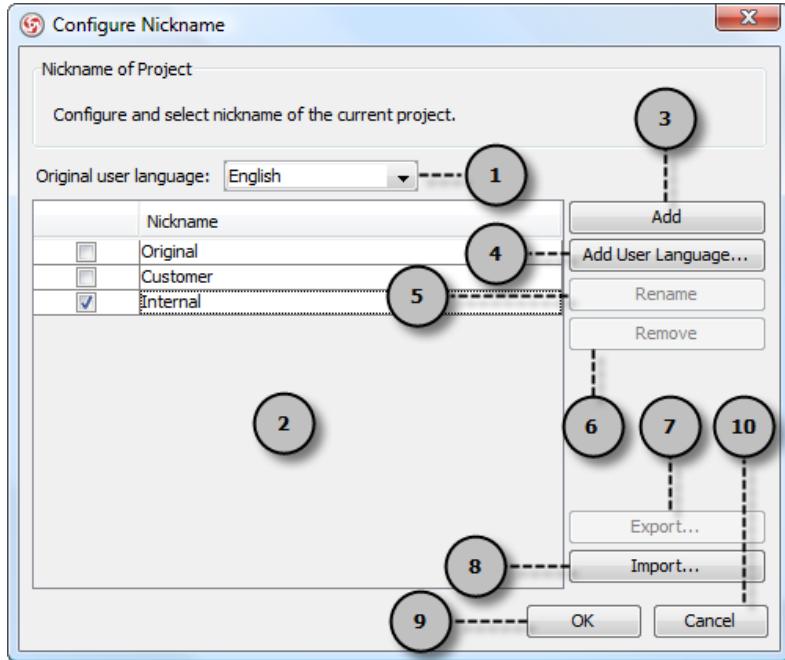
The name of **Entity** will be used to generate SQL, but Database Management System (DBMS) has many constraints on the name of Entity, Column, etc, and each DBMS are different . These constraints include the length of the name, reserved keywords, special characters, etc. They restricted the database 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 display **ERD** in nickname but generate SQL in original name:



Configure nickname

You can add a nickname through the nickname drop down menu that appears in toolbar, or through the appropriate menu under the View menu in menu bar. By adding a nickname, you can start editing the names and documentations of model elements under the new nickname.

Overview of Configure Nickname dialog box



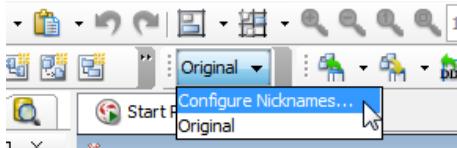
An overview of Configure Nickname dialog box

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 documentation of model elements that are named differently in nickname.
8	Import	Click to import the XML exported from Configure Nickname dialog box.
9	OK	Click to apply the nickname configuration and close this dialog box.
10	Cancel	Click to close the dialog box without applying the changes.

Description of Overview dialog box

Adding nickname

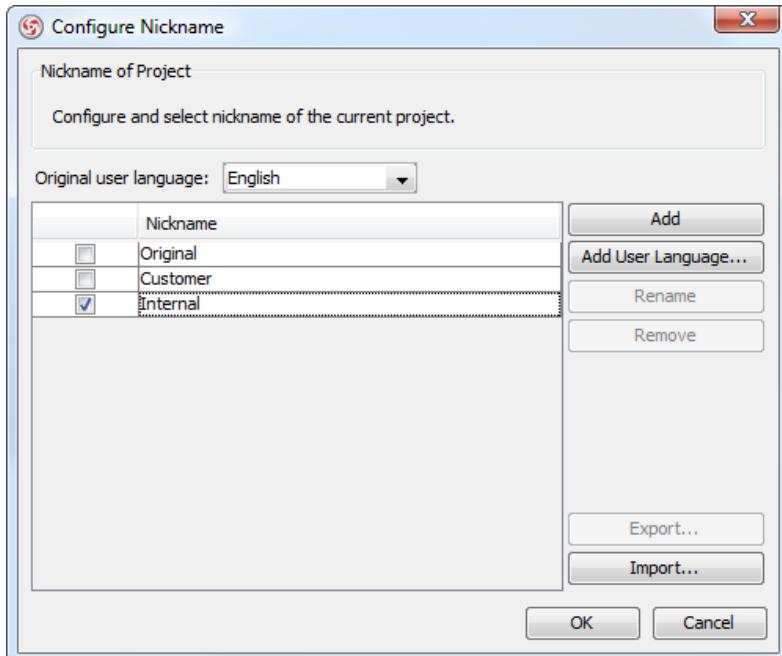
1. Click on the drop down menu with caption **Original**, and select **Configure Nicknames...** from the popup menu.



Configure Nicknames

NOTE: This is If you do not see this drop down menu, it could be due to the toolbar is being turned off. You can show it by right clicking on the empty space on the toolbar and selecting **Nickname** from popup menu, or simply, access the same function by selecting **View > Nicknames > Configure Nicknames...** from the main menu.

2. The current working copy is by default in **Original** nickname, with English as user language. Click **Add** in the **Configure Nickname** dialog box.
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** dialog box.



Two nicknames are added

Renaming nickname

1. Click on the drop down menu with caption **Original**, and select **Configure Nicknames...** from the popup menu.

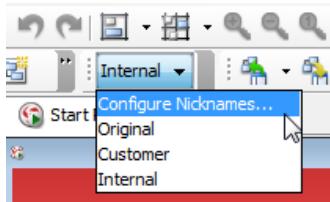


Figure 10-8 Configure Nicknames

NOTE: This is If you do not see this drop down menu, it could be due to the toolbar is being turned off. You can show it by right clicking on the empty space on the toolbar and selecting **Nickname** from popup menu, or simply, access the same function by selecting **View > Nicknames > Configure Nicknames...** from the main menu.

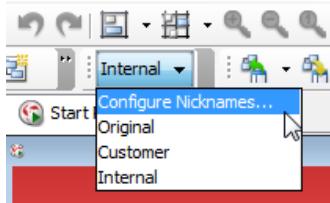
2. In the **Configure Nickname** dialog box, select the nickname to rename. Click **Rename**.

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** dialog box.

Removing nickname

1. Click on the drop down menu with caption **Original**, and select **Configure Nicknames...** from the popup menu.



Configure Nicknames

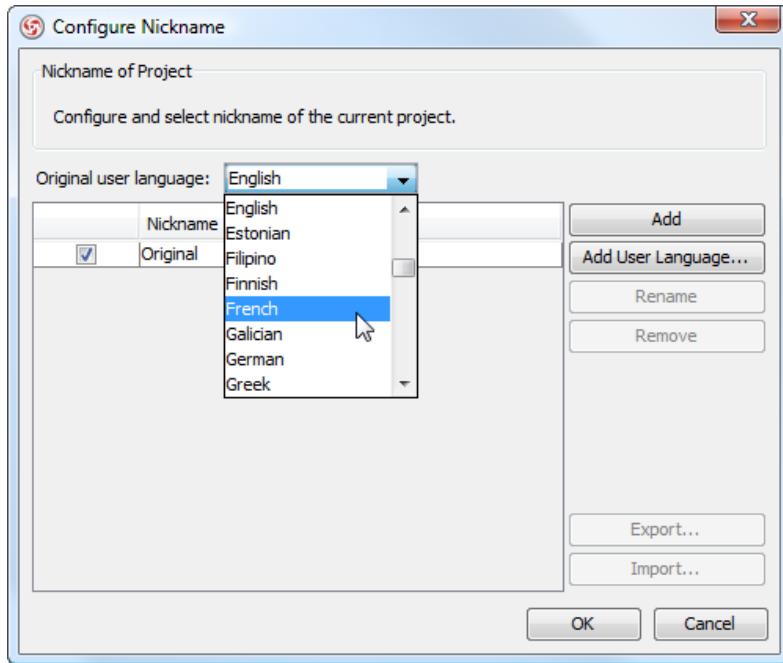
NOTE: This is If you do not see this drop down menu, it could be due to the toolbar is being turned off. You can show it by right clicking on the empty space on the toolbar and selecting **Nickname** from popup menu, or simply, access the same function by selecting **View > Nicknames > Configure Nicknames...** from the main menu.

2. In the **Configure Nickname** dialog box, 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.

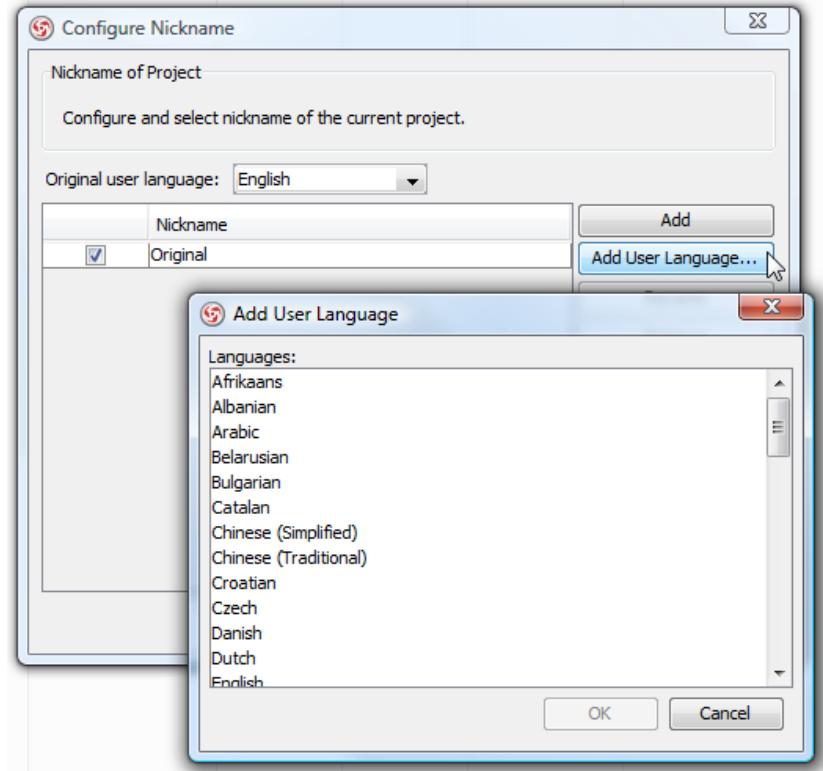
Specifying user language for the nickname

There must be an original nickname in every project, namely *Original*, representing 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.



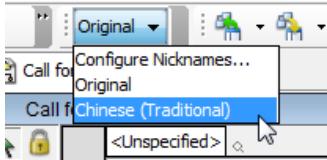
Add a user language

Using nickname

Once you have defined a nickname, you can start updating your model by entering the new names and documentations of model elements.

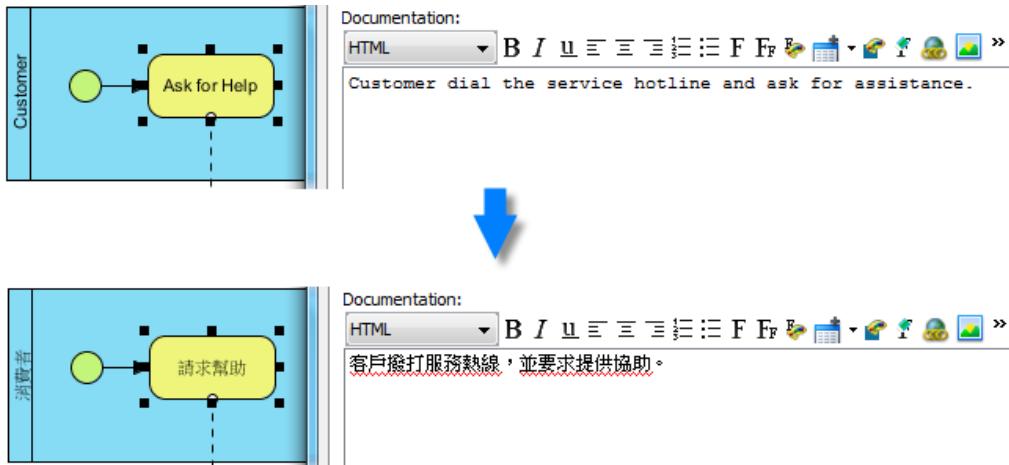
Start updating elements' nickname

- From the application toolbar, select the nickname you need to work with.



Selecting a nickname to work with

- Start renaming model elements and updating their documentation. The changes you make will only be applied to the selected nickname.

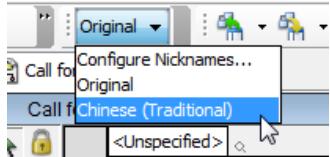


Updating the name and documentation of elements under a nickname

Switching between nicknames

The names and documentations 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 documentations of model elements will be updated to show the definition under the new nickname.

To switch between nicknames, select the nickname to switch to from the drop down menu of nickname in application toolbar.



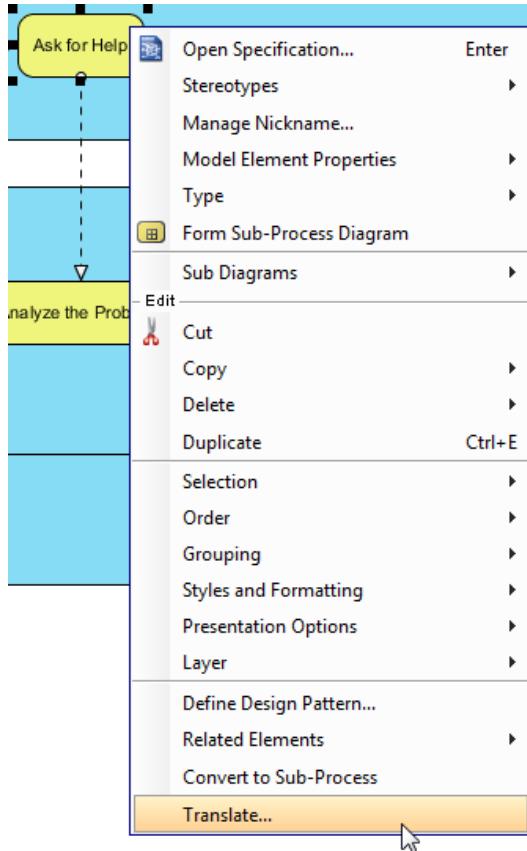
Switching nickname

Translation

Further to the definition of nickname, you also can make use of the translate function to translate your work into another language. This is particularly useful when you want to add a new nickname for another different language, without entering all names and documentations from the beginning.

To perform a translation:

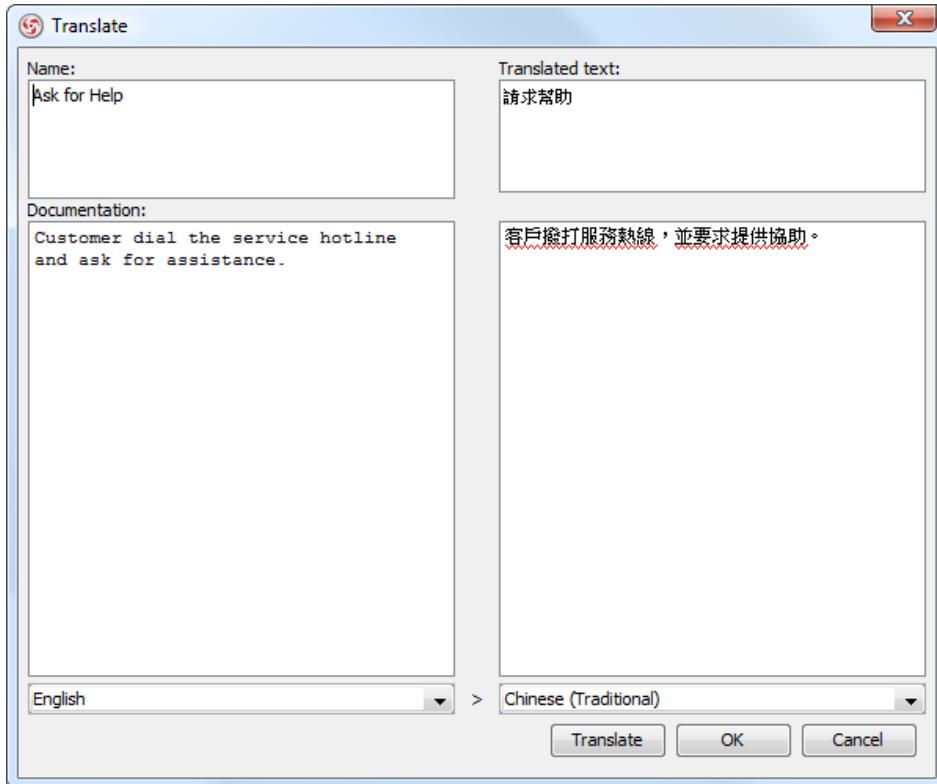
1. Right click on the diagram's background or the shape you want to perform translation and select **Translate...** from the popup menu.



To perform translation on a shape

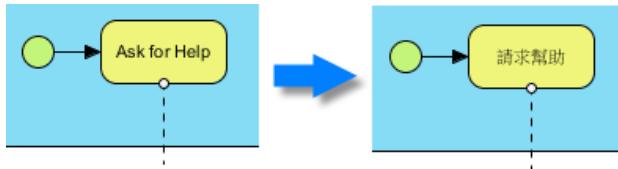
NOTE: Make sure you are working under a nickname in order to see the Translate menu.

2. In the **Translate** dialog box, the left hand side displays the name and documentation of the chosen shape in the language set for original nickname, which is English by default, while the right hand side displays name and documentation in the language of the active nickname.



Translation outcome

3. Review and make necessary correction to the translated outcome. You can change to translate to another language by selecting a different language from the drop down menu at the bottom right corner of dialog box, then click the Translate button. Click **OK** to proceed with translation.



A translated business task

What is visual diff?

There are times that we want to compare two diagrams. For example, to compare an ERD of conceptual model with an ERD of physical model, to compare a domain class diagram with a class diagram ready for implementation. VP-UML lets you compare differences between diagrams to trace the changes between them.

Diagram comparison

By using the Visual Diff tool, users can compare two diagrams, know their differences by reading the result of comparison. Changes such as modification of properties (e.g. name) and addition/removal of containing models, etc can be found easily.

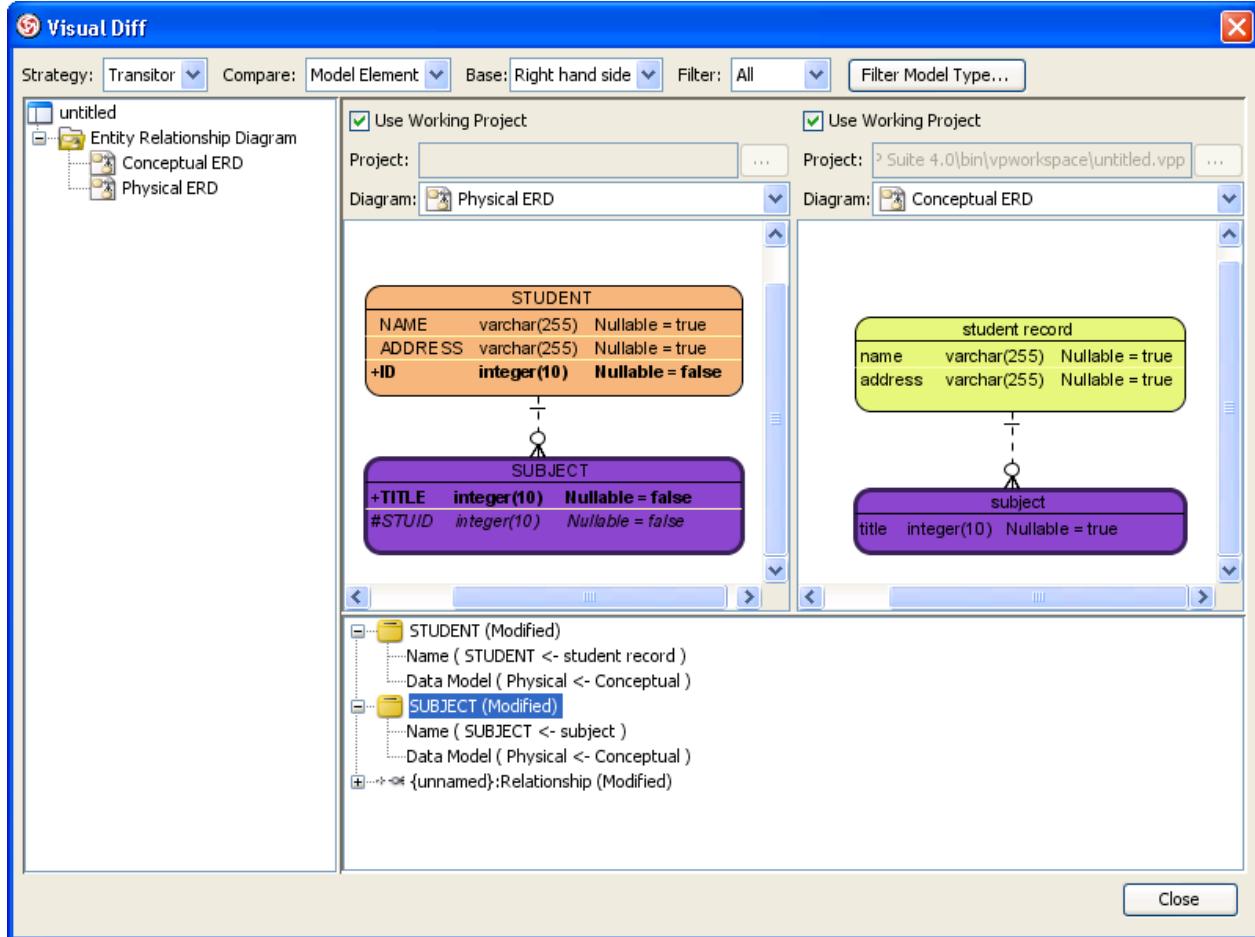


Figure 11-1 The Visual Diff tool

Various comparison strategies

A comparison strategy determines how two diagrams will be compared. Each strategy work best for a special purpose. You can select the appropriate strategy that suits your need. Here are 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 useful when visualizing 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. Typical examples are to compare databases and class models.
Transitor	Shapes will be matched base on their transition established by Model Transitor. This way of comparison is useful when visualizing differences between Models.

Table 11-1 Description of comparison strategies

Compare view only, model element only, or both

Comparison can cover view, model element, or both of them, which affects the result to be displayed. By comparing view, differences in view settings such as the coordinate of shapes will be considered as changes. By comparing model element, differences in model element level like their names are considered as changes.

The following screenshot displays both view and model element differences.

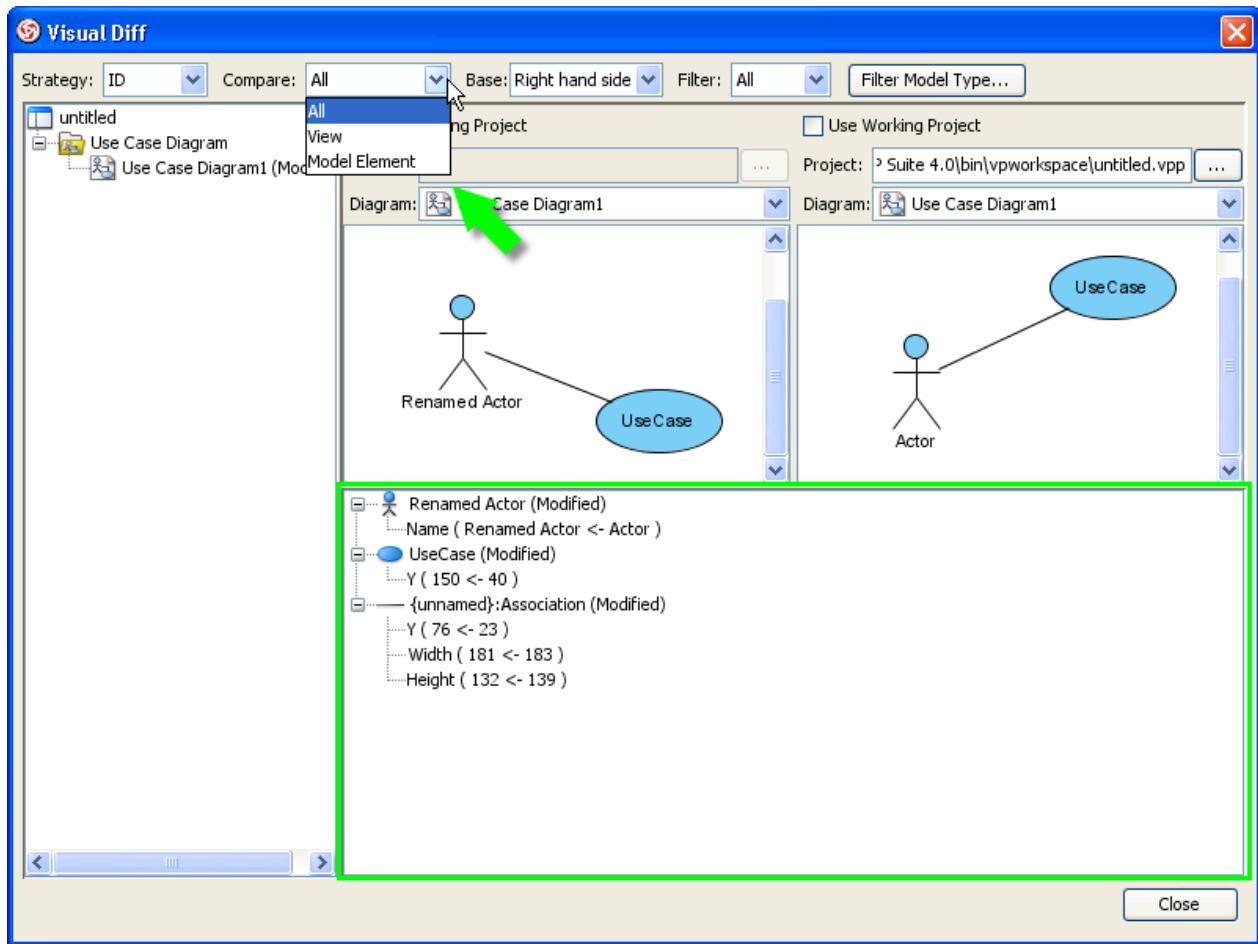


Figure 11-2 Compare both View and Model Element

Below is the result when selected only to compare **View**. Differences in coordinates and shapes' width and height are therefore considered as changed.

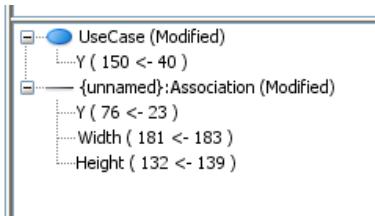


Figure 11-3 Compare View

Below is the result when selected only to compare **Model Element**. Rename of model element is therefore considered as a change.

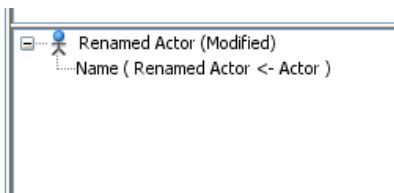


Figure 11-4 Compare Model Element

Compare left to right, or the other way round

Comparisons are made between two diagrams, which are put at the left and right hand side in the **Visual Diff** dialog box 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 exist on the right hand side, the result pane will show that the shape is newly added. However, it is unknown to VP-UML whether the shape should be said as created or removed. Therefore, user can switch the base from right to left. By doing so, the absent of shape on the left hand side will result in a report of deleted shape instead of an addition of shape.

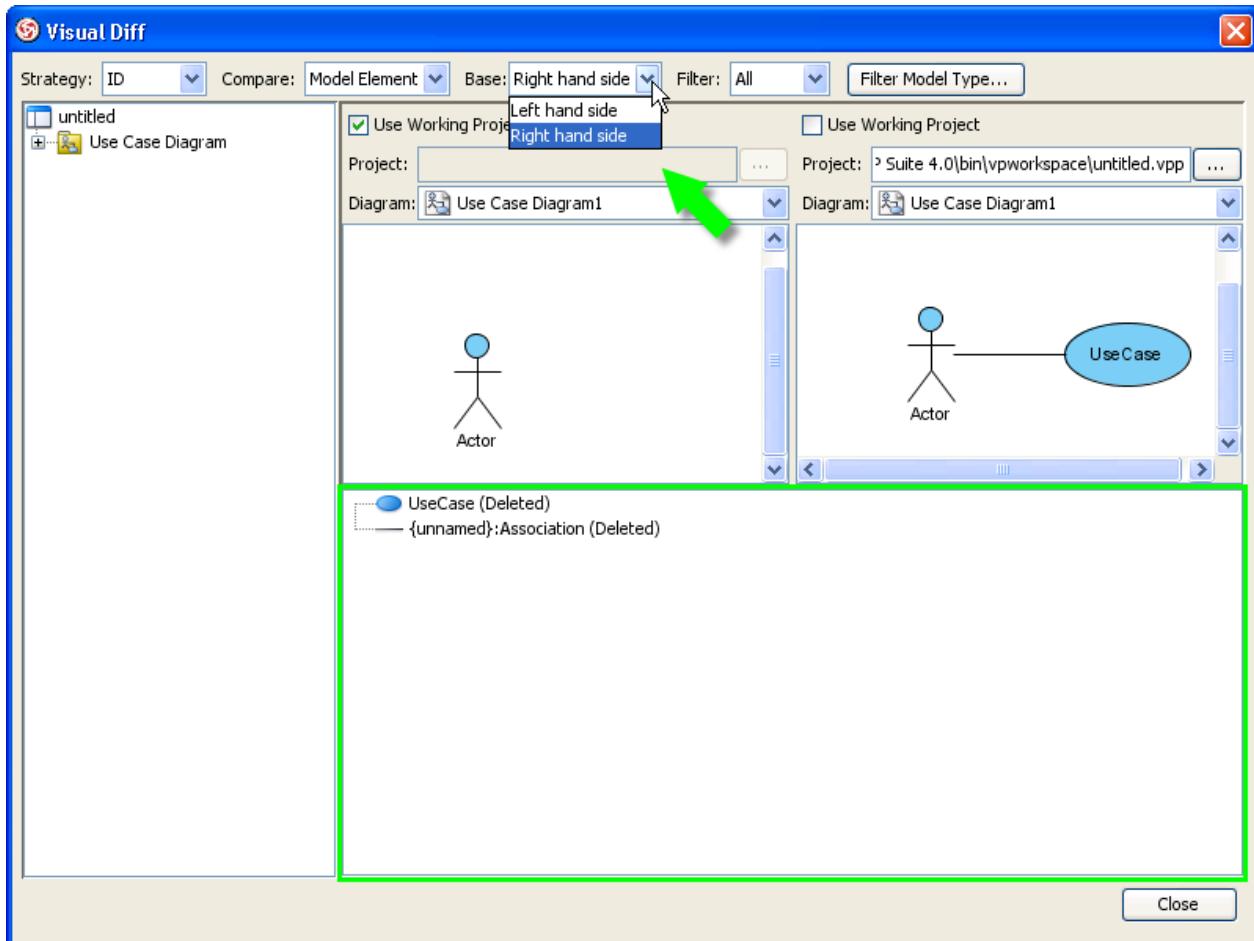


Figure 11-5 Comparing diagrams with right hand side as base

Below is the result when the base is switched from right hand side to left hand side. Deletion of model is said to be addition (see the text **New**).

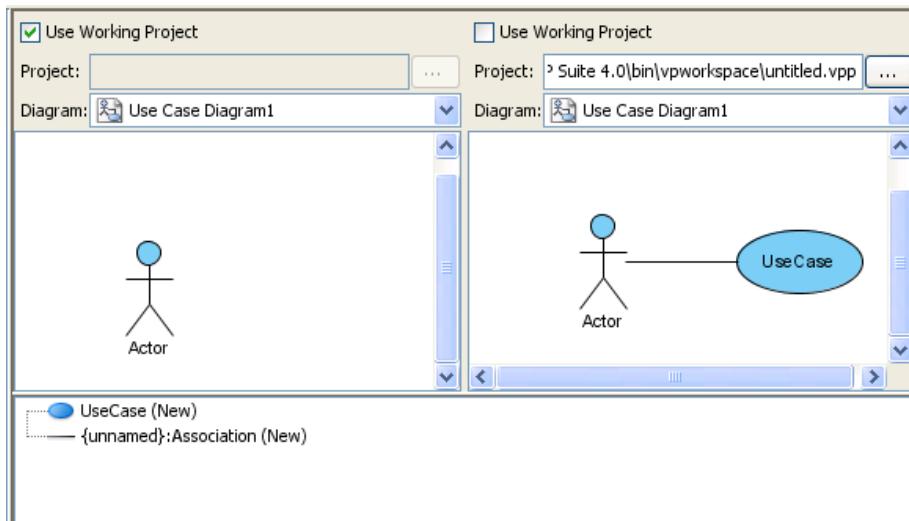


Figure 11-6 Comparing diagrams with left hand side as base

Comparing as-is and to-be business process diagram

Here are two Business Process Diagrams, one for modeling the As-is Process, and the other one is for modeling the To-be Process. We will make use of Visual Diff to find the differences between them.

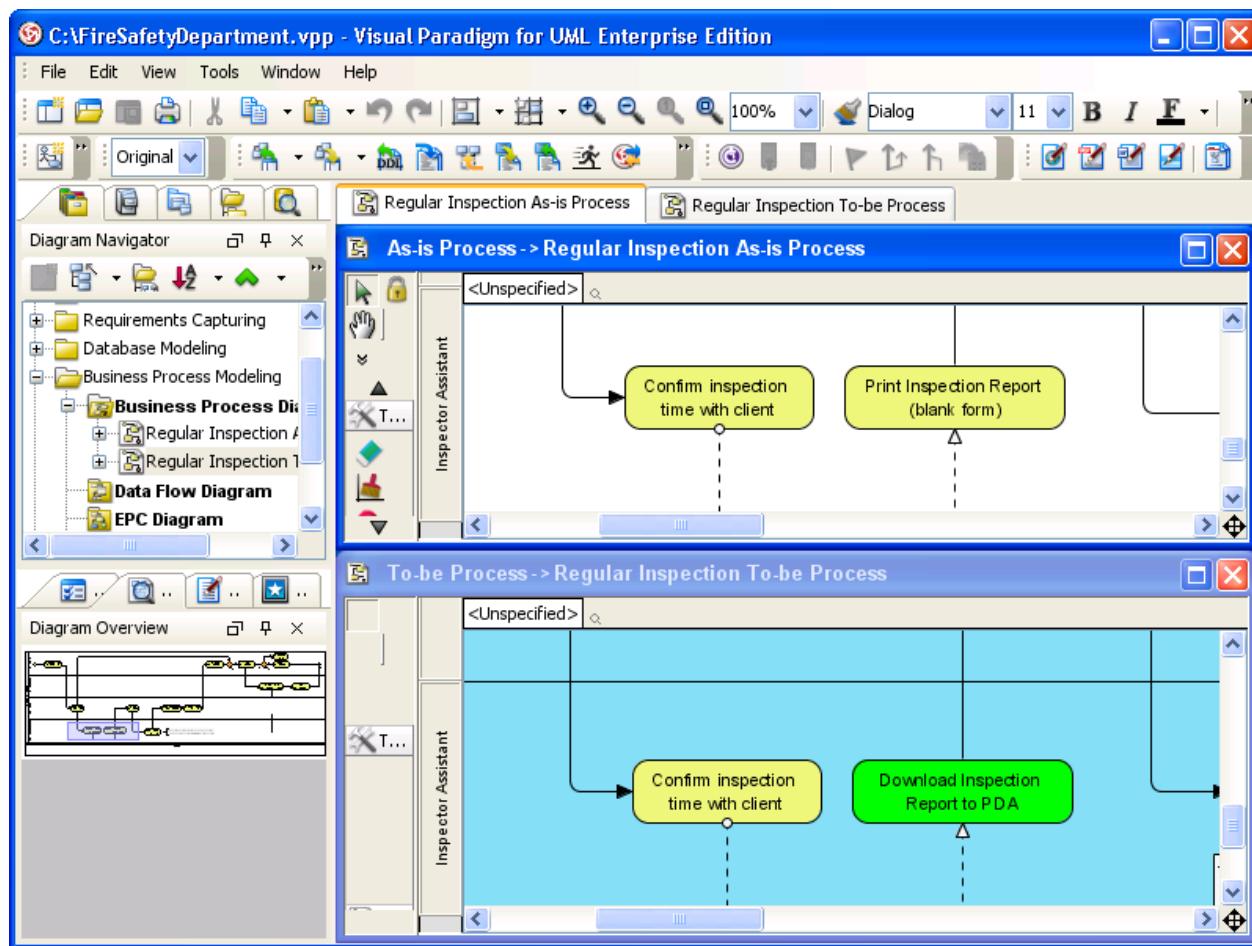


Figure 11-7 BPDs for As-Is Process and To-Be Process

- From the diagram of As-Is Process, select **Tools > Visual Diff...** from the main menu.

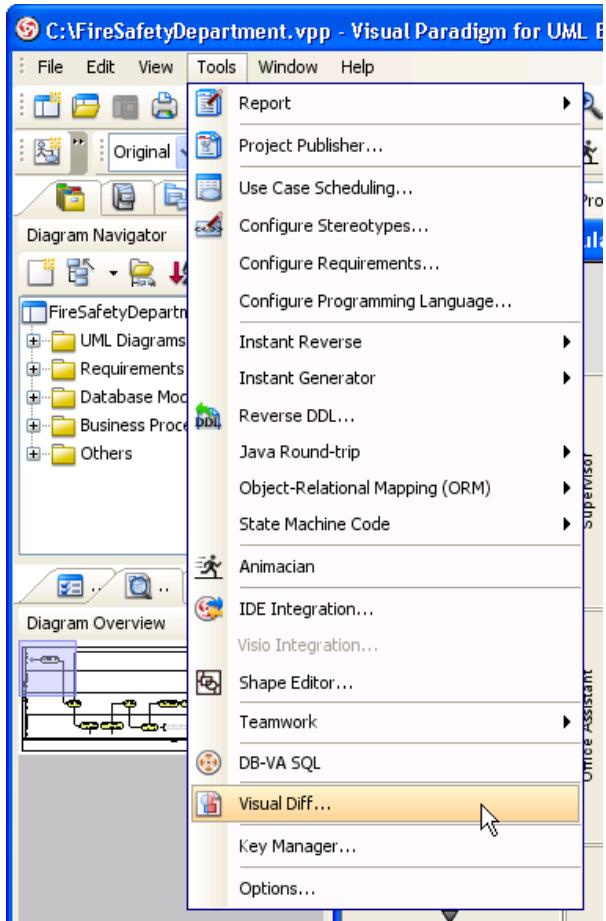


Figure 11-8 Launching Visual Diff through the main menu

NOTE: Besides starting through the main menu, you can start Visual Diff through the ways below:

- Right-click on diagram background and selecting **Utilities > Visual Diff...** from the popup menu.
- Click on the **Visual Diff** button on the **Tools** toolbar.

This starts **Visual Diff**.

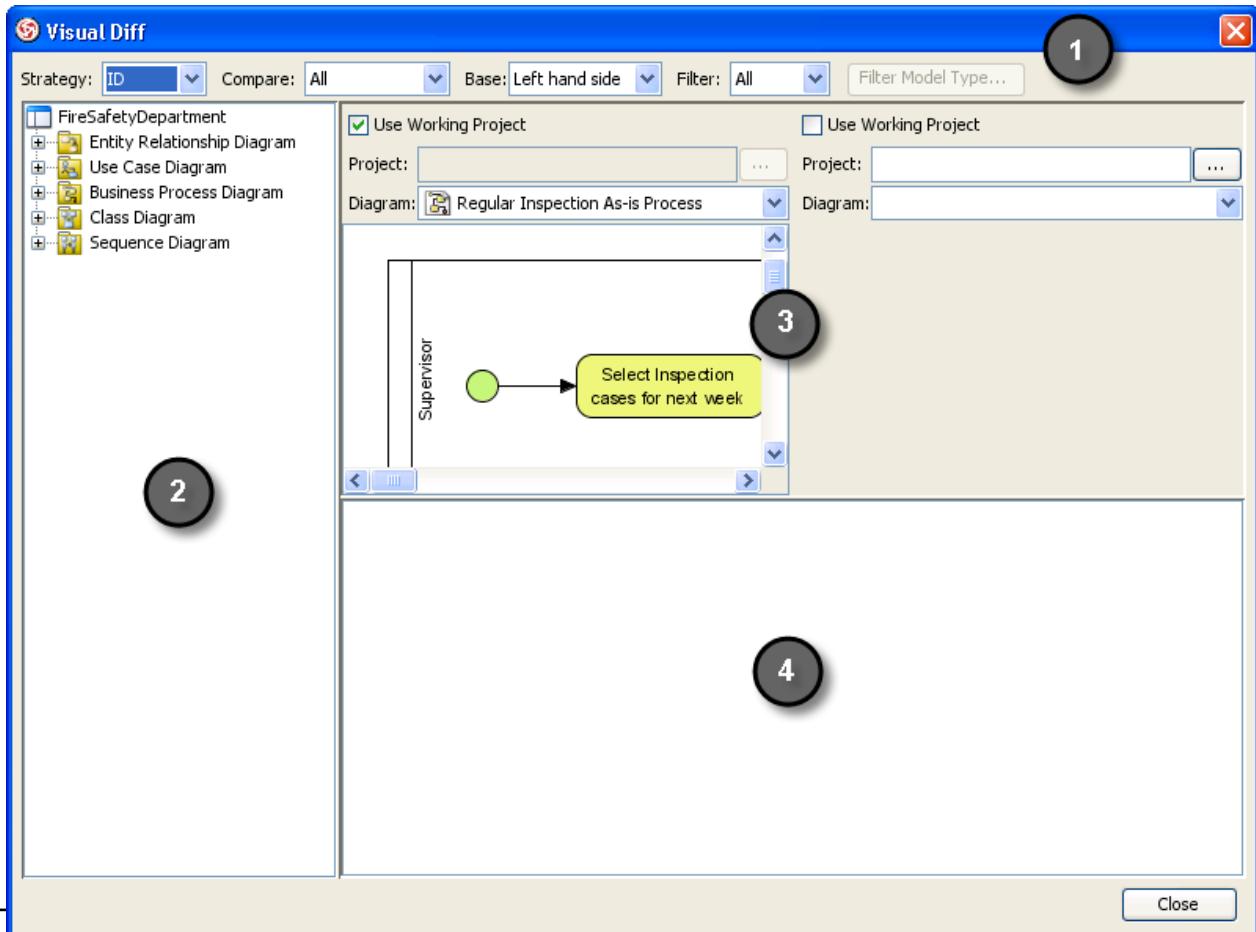


Figure 11-9 The Visual Diff dialog box

2. The left hand side is showing the currently opening diagram. Let's keep it unchanged. Now, select **Use Working Project** for the right hand side so that we can select a diagram in the same project to compare with.

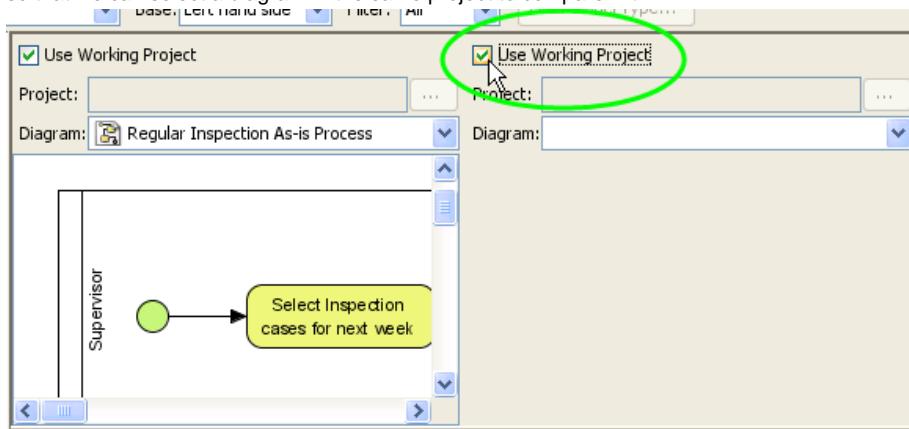


Figure 11-10 Select Use Working Project

NOTE: To compare with diagram in another project, uncheck **Use Working Project** and select the project file in the Project field.

3. Select the *To-be Process* to compare with.

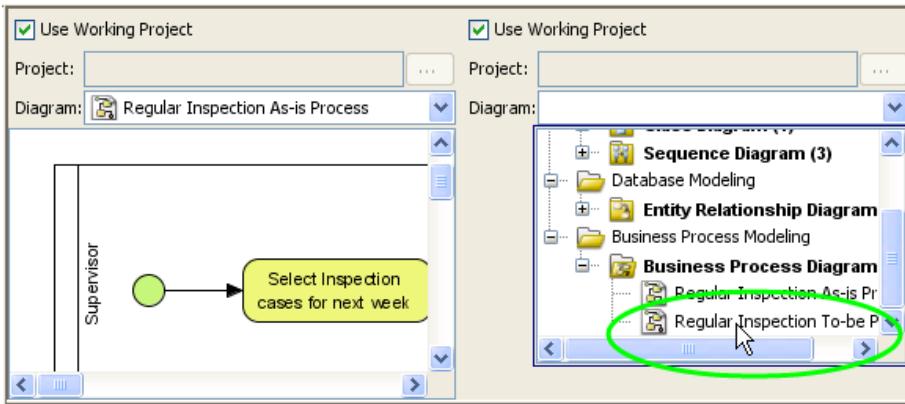


Figure 11-11 Select diagram to compare

Visual Diff is updated to show the two diagrams side by side. The result pane at the bottom is updated, too. However, we need to configure **Visual Diff** in order to compare in the way we want.

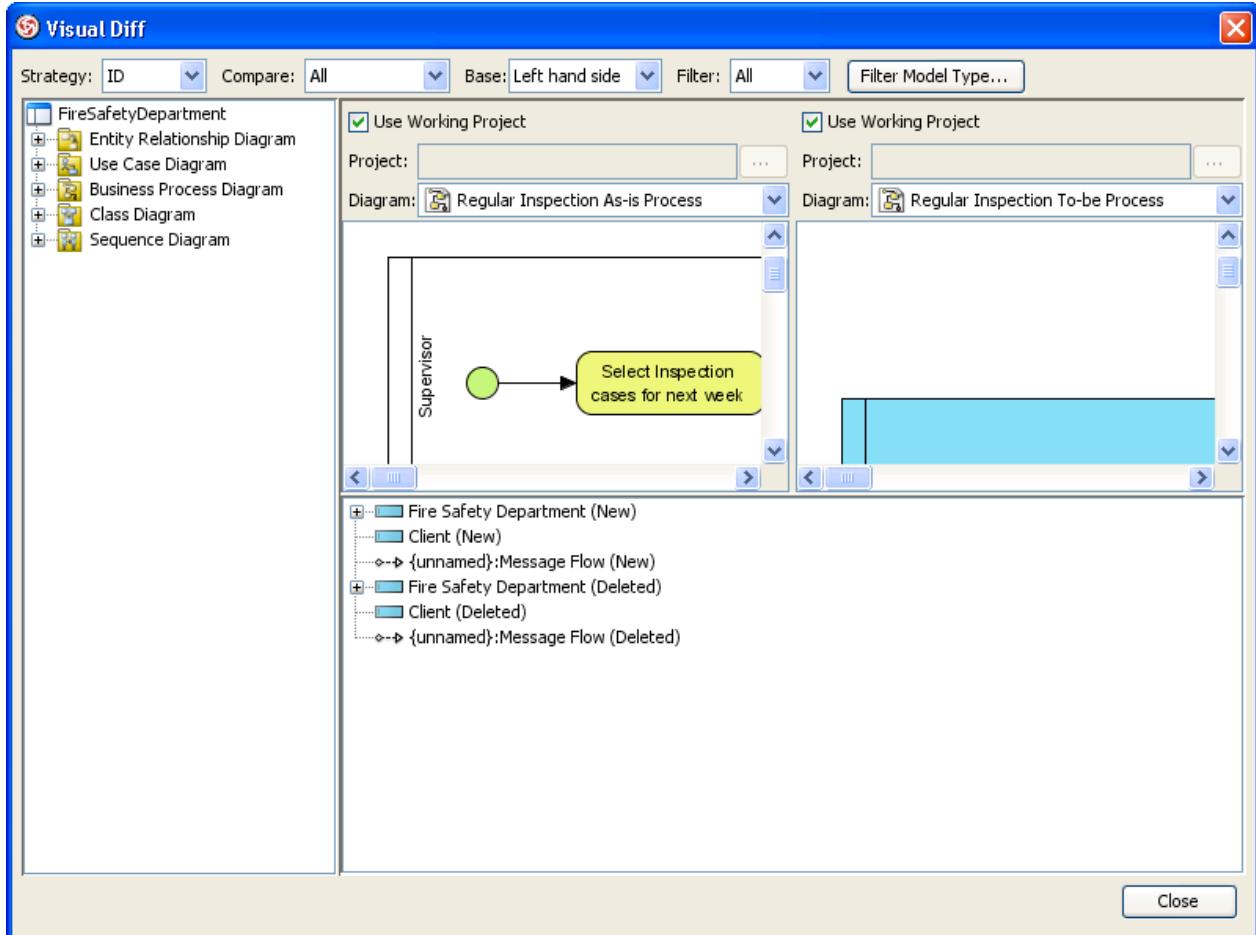


Figure 11-12 Diagrams selected

4. Select Name as **Strategy**.

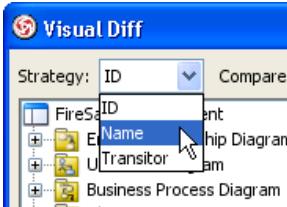


Figure 11-13 Select a comparison strategy

Below is a description of available **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 useful when visualizing 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. Typical examples are to compare databases and class models.
Transitor	Shapes will be matched base on their transition established by Model Transitor. This way of comparison is useful when visualizing differences between Models.

Table 11-3 Description of comparison strategies

5. Select *Model Element* as the scope of comparison.

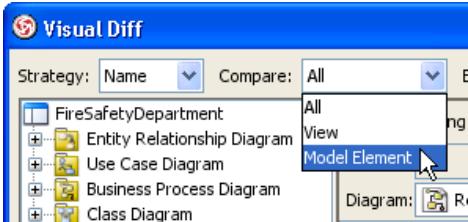


Figure 11-14 Select the item to compare

Below is a description of available **Compare** options.

Option	Description
All	Both view and model element are displayed.
View	Differences such as coordinates, width, height and color of shapes are displayed.
Model Element	Differences such as model name is displayed.

Table 11-4 Description of compare options

6. Select *Left hand side* as **Base**.

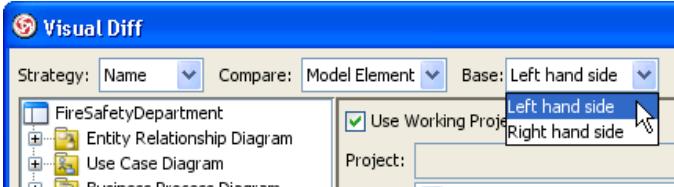


Figure 11-15 Select a base

Below is a description of available **Bases**.

Base	Description
Left hand side	The default base selection which cause comparison to be made base on the diagram on the left 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	Cause comparison to be made base 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.

Table 11-5 Description of bases

7. Select All as the Filter.

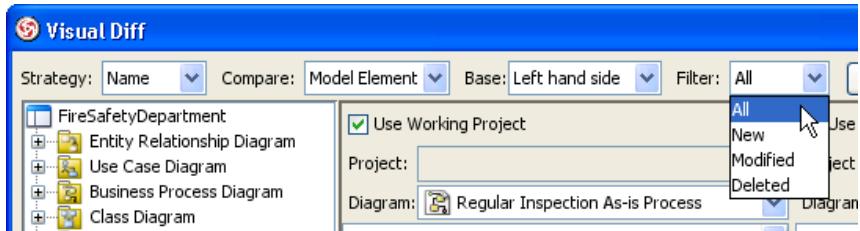


Figure 11-16 Select a filter

Below is a description of available Filters.

Filter	Description
All	Display all kinds of differences which includes the addition, modification and removal of shapes.
New	Display only results about the addition of shape, and hide the rest.
Modified	Display only results about the modification of shapes, and hide the rest.
Deleted	Display only results about the removal of shapes, and hide the rest.

Table 11-6 Description of filters

8. Once everything is set, we can see the differences of the two diagrams from the result pane.

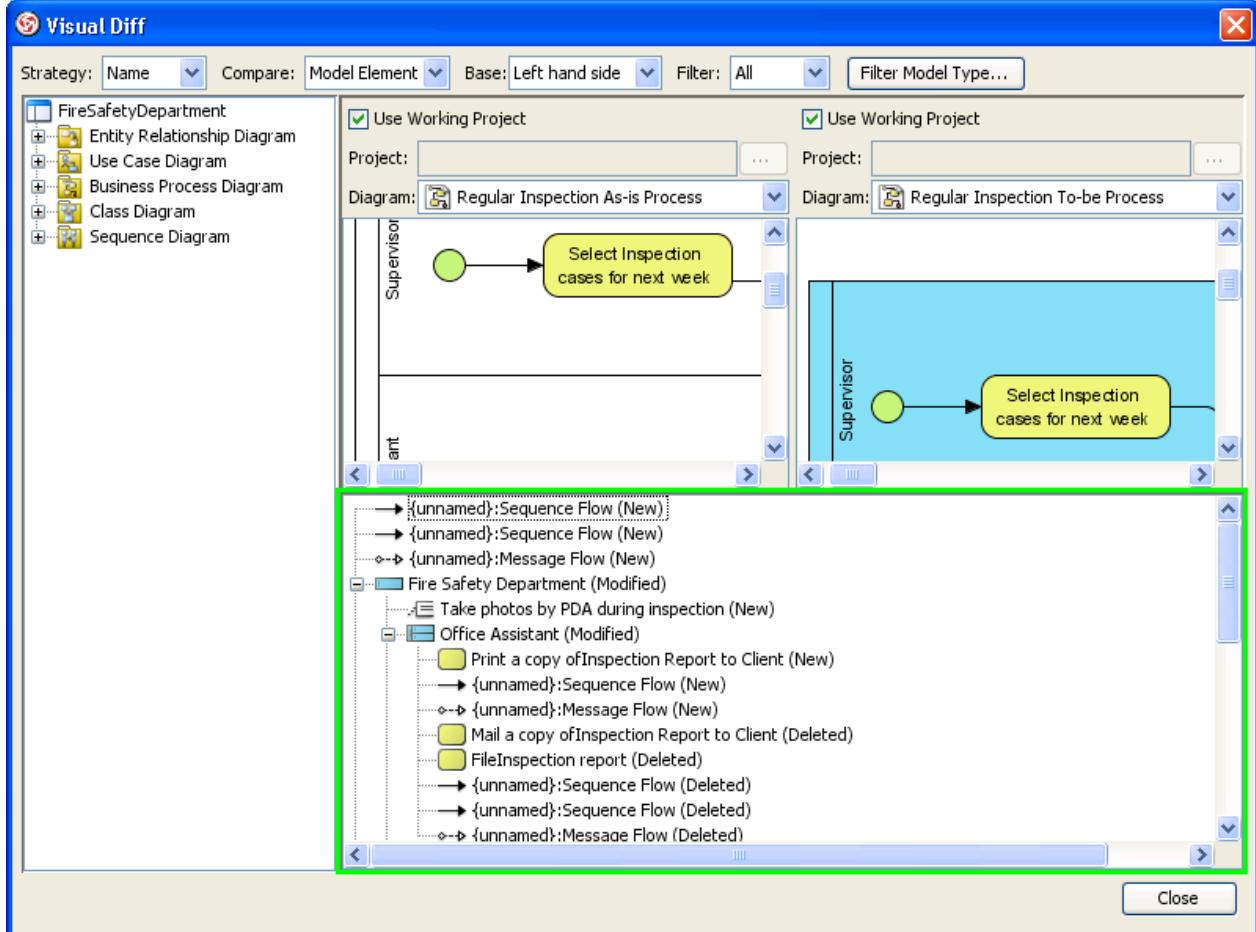


Figure 11-17 Result of comparison is obtained

9. It is possible to click on a node to select the shape in diagrams. Selected shape is painted in dark purple.

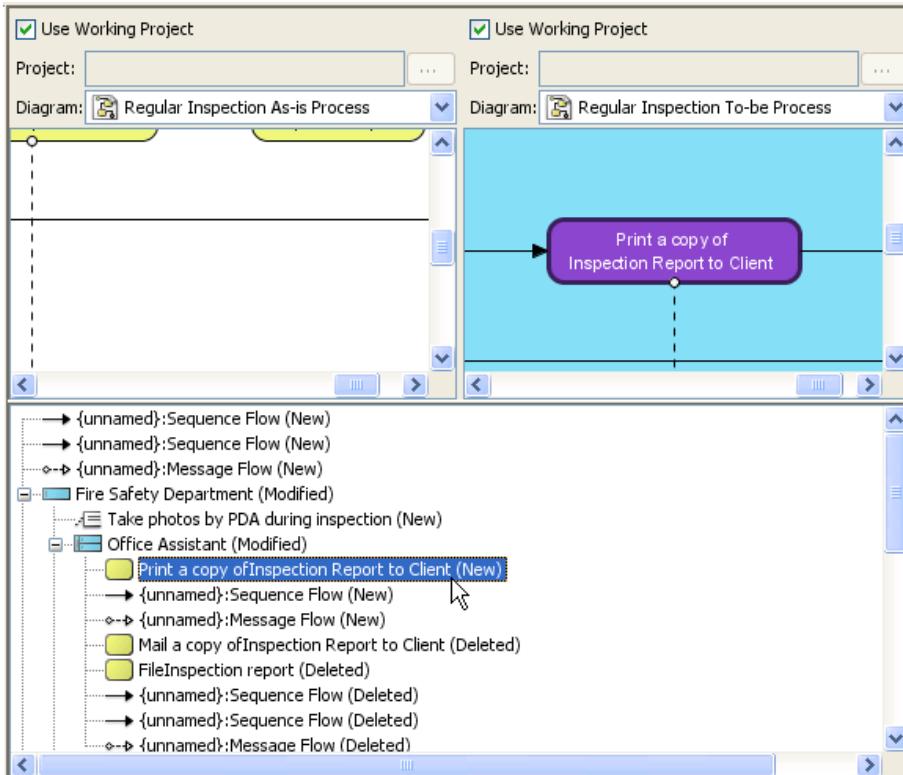


Figure 11-18 Select a node to cause the shape to be selected in diagram

Below is a description of results.

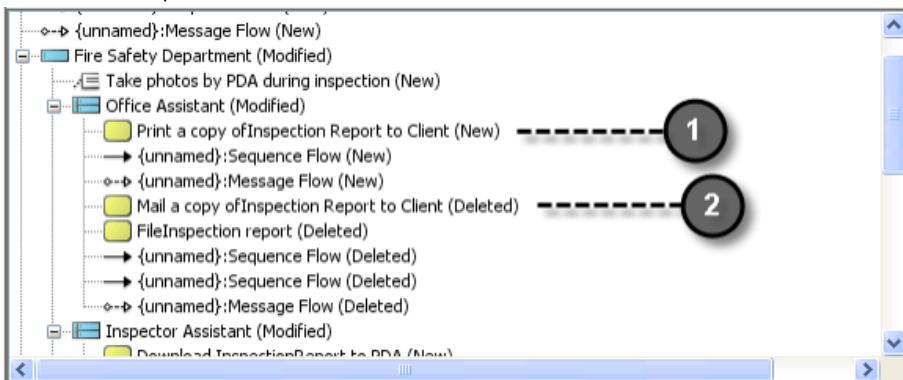


Figure 11-19 Result of comparison

Result	Description
1	The Task is newly added.
2	The Task is deleted.

Table 11-7 Description of result

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, thus confirming the database is exactly following the initial business requirements regardless of the changes.

Here are two ERDs, one for modeling the Logical Model, and the other one is for modeling the Physical Model. We will make use of **Visual Diff** to find the differences between Logical and Physical ERD.

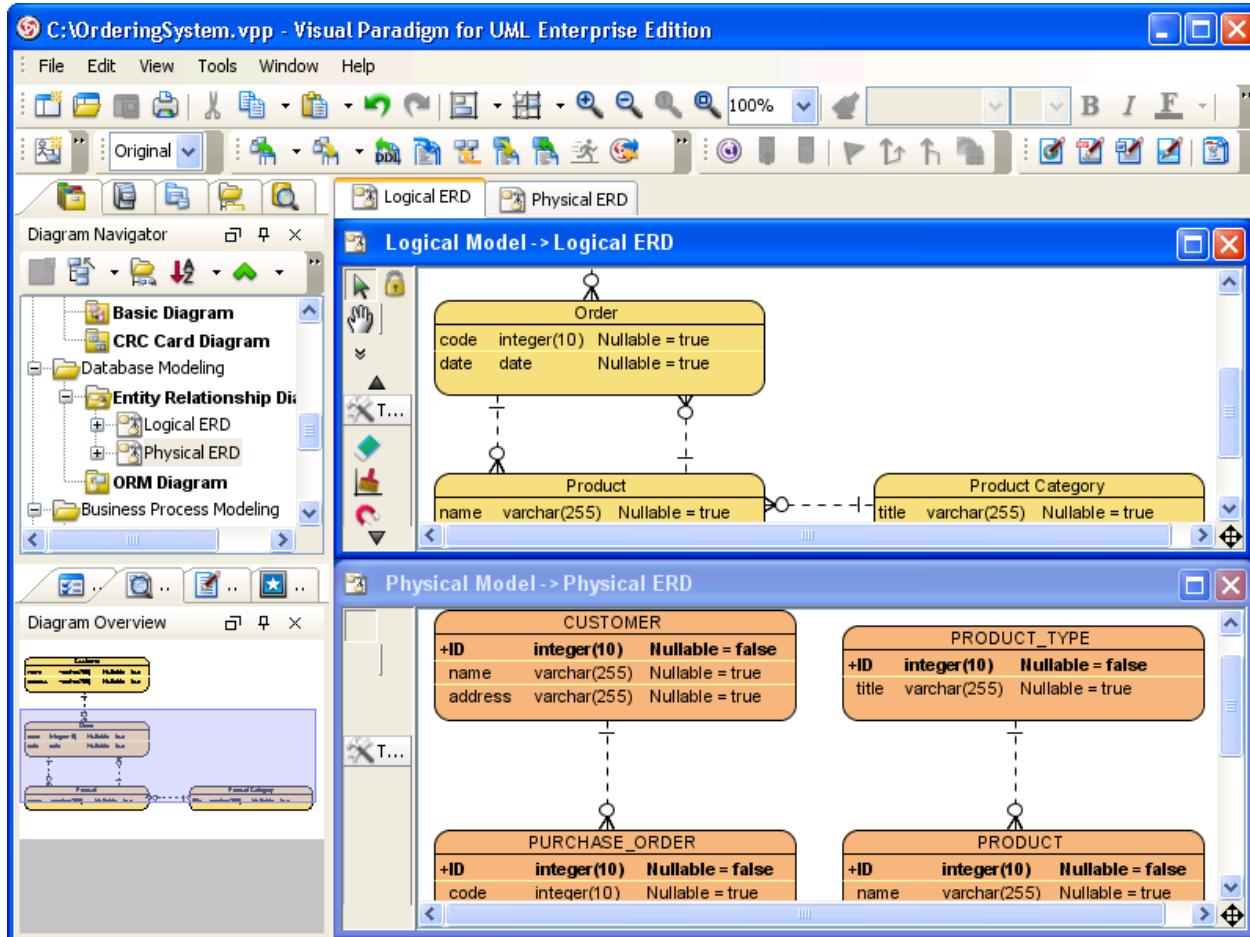


Figure 11-20 ERDs for Logical and Physical Model

- From the diagram of *Logical ERD*, select **Tools > Visual Diff...** from the main menu.

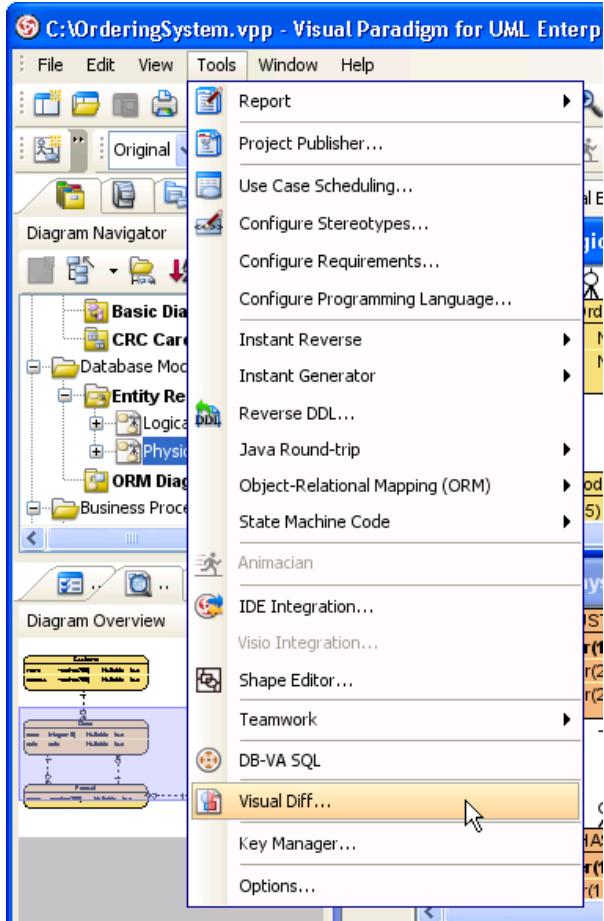


Figure 11-21 Launching Visual Diff through the main menu

NOTE: Besides starting through the main menu, you can start Visual Diff through the ways below:

- Right-click on diagram background and selecting **Utilities > Visual Diff...** from the popup menu.
- Click on the **Visual Diff** button on the **Tools** toolbar.

This starts **Visual Diff**.

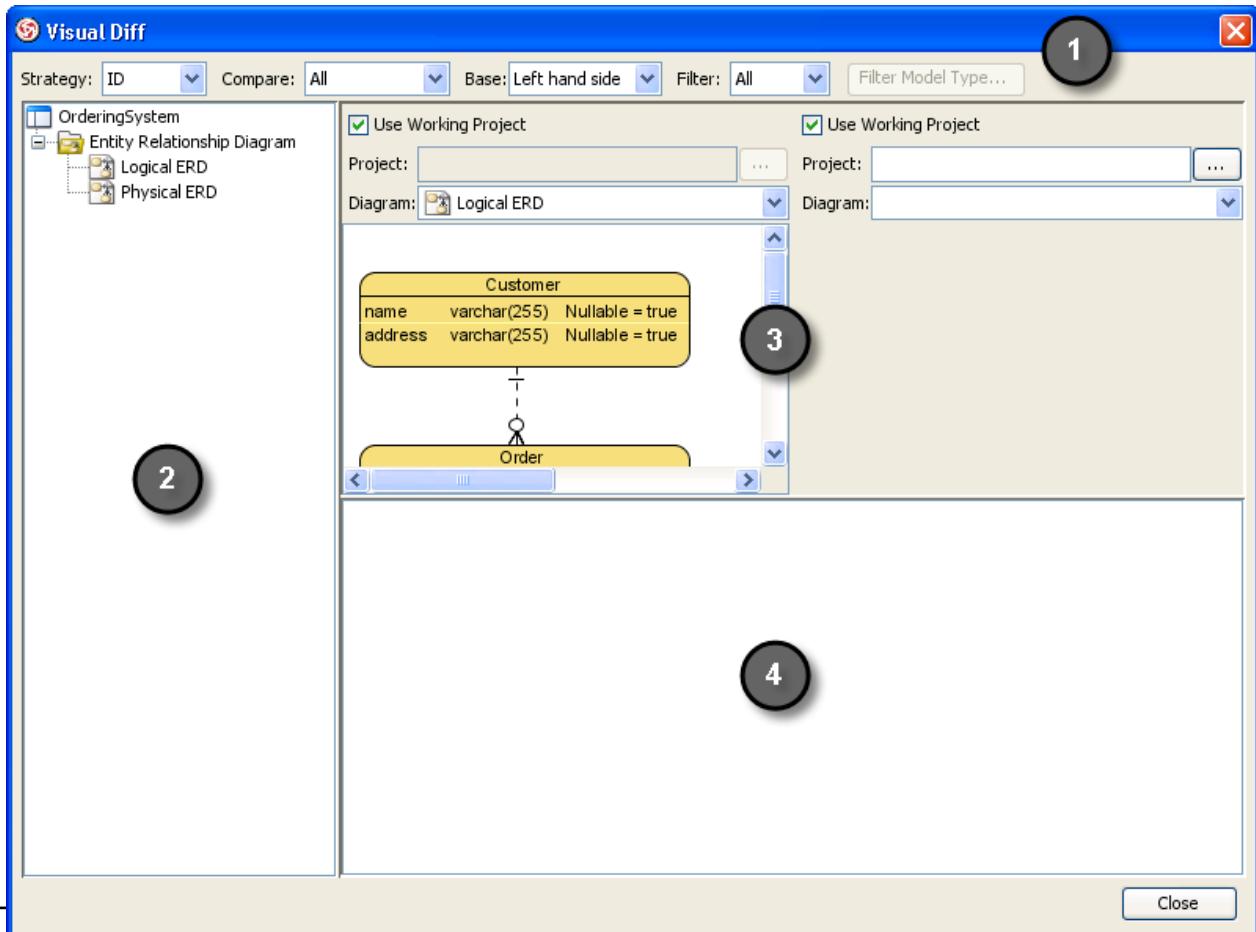


Figure 11-22 The Visual Diff dialog box

2. The left hand side is showing the currently opening diagram. Let's keep it unchanged. Now, select **Use Working Project** for the right hand side so that we can select a diagram in the same project to compare with.

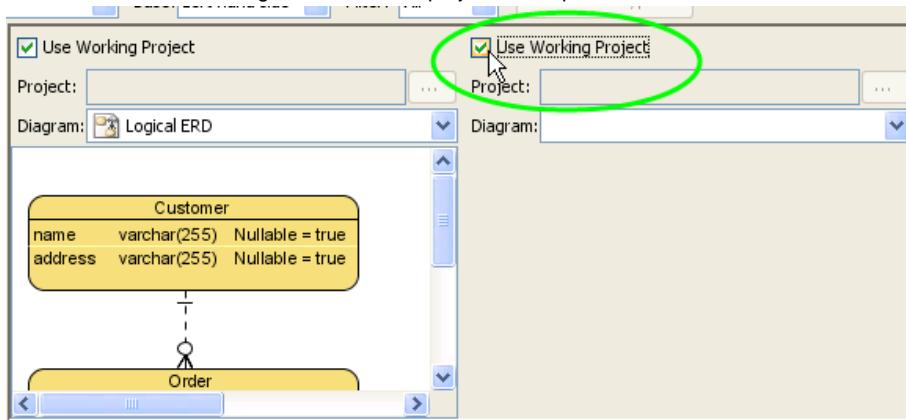


Figure 11-23 Select Use Working Project

NOTE: To compare with diagram in another project, uncheck **Use Working Project** and select the project file in the Project field.

3. Select the *Physical ERD* to compare with.

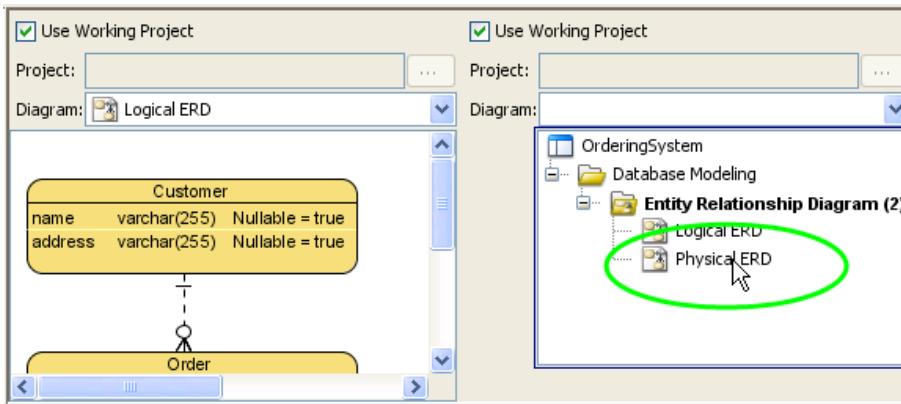


Figure 11-24 Select diagram to compare

Visual Diff is updated to show the two diagrams side by side. The result pane at the bottom is updated, too. However, we need to configure **Visual Diff** in order to compare in the way we want.

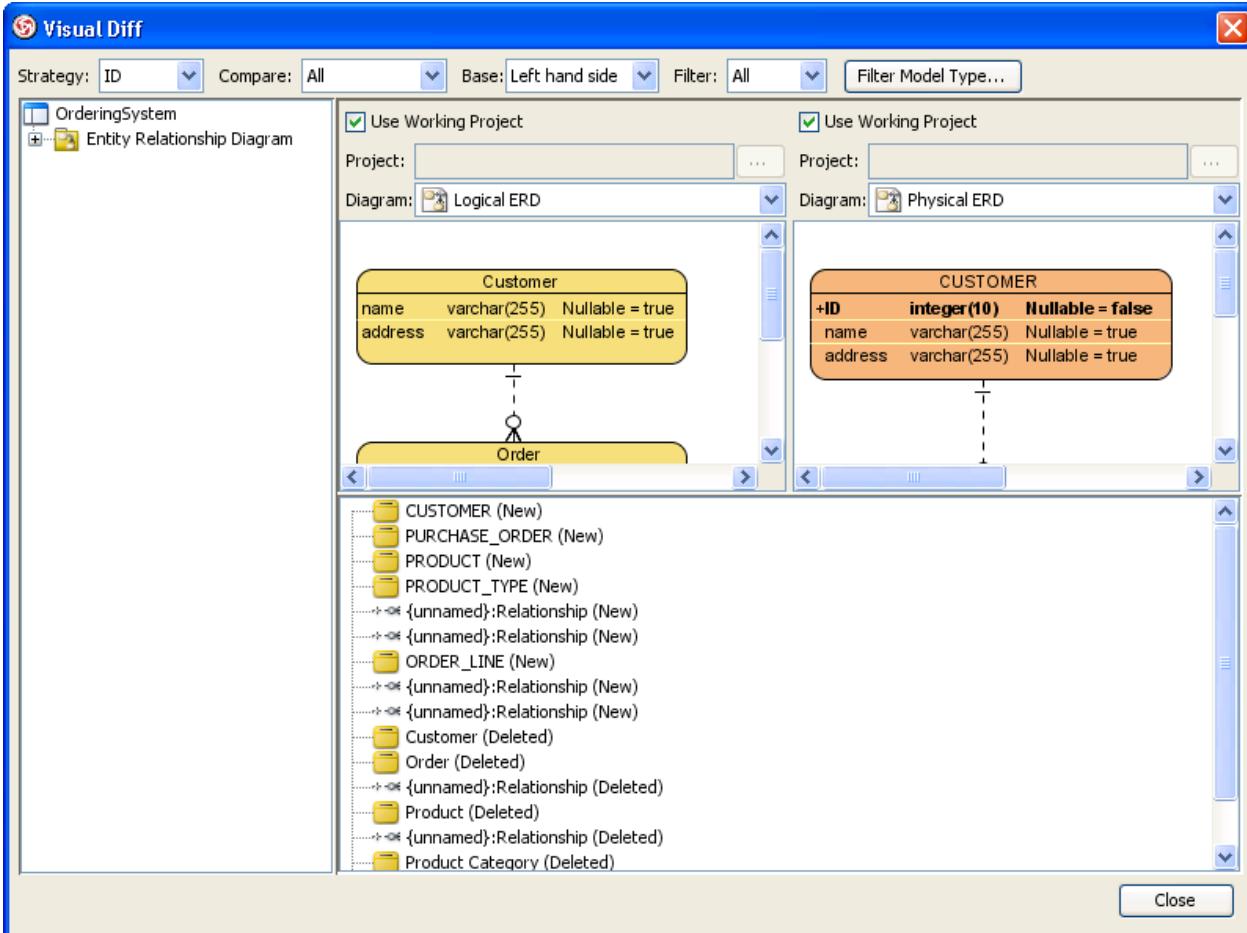


Figure 11-25 Diagrams selected

4. Select **Transitor** as **Strategy**.

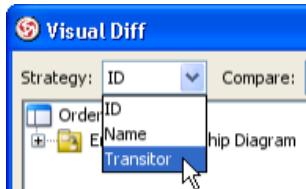


Figure 11-26 Select a comparison strategy

Below is a description of available **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 useful when visualizing 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. Typical examples are to compare databases and class models.
Transitor	Shapes will be matched base on their transition established by Model Transitor. This way of comparison is useful when visualizing differences between Models.

Table 11-9 Description of comparison strategies

5. Select **Model Element** as the scope of comparison.

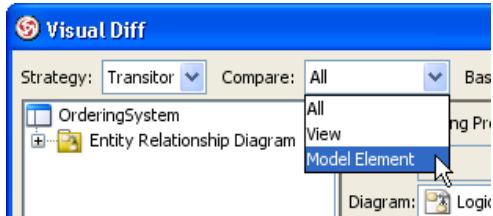


Figure 11-27 Select the item to compare

Below is a description of available **Compare** options.

Option	Description
All	Both view and model element are displayed.
View	Differences such as coordinates, width, height and color of shapes are displayed.
Model Element	Differences such as model name is displayed.

Table 11-10 Description of compare options

6. Select **Left hand side** as **Base**.

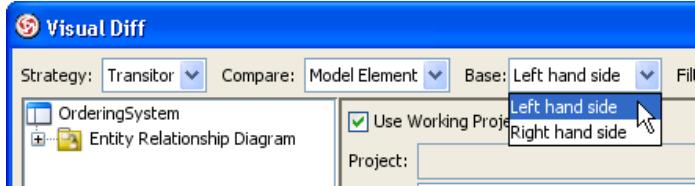


Figure 11-28 Select a base

Below is a description of available **Bases**.

Base	Description
Left hand side	The default base selection which cause comparison to be made base on the diagram on the left 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	Cause comparison to be made base 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.

Table 11-11 Description of bases

7. Select All as the Filter.

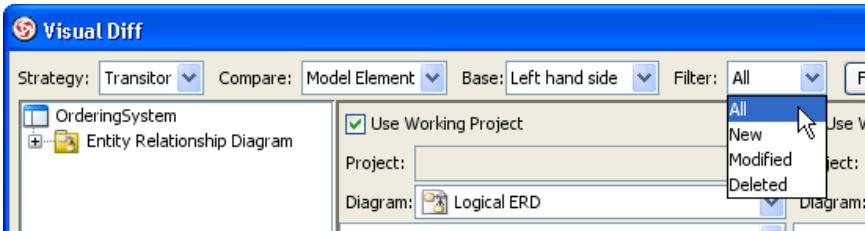


Figure 11-29 Select a filter

Below is a description of available Filters.

Filter	Description
All	Display all kinds of differences which includes the addition, modification and removal of shapes.
New	Display only results about the addition of shape, and hide the rest.
Modified	Display only results about the modification of shapes, and hide the rest.
Deleted	Display only results about the removal of shapes, and hide the rest.

Table 11-12 Description of filters

8. Once everything is set, we can see the differences of the two diagrams from the result pane.

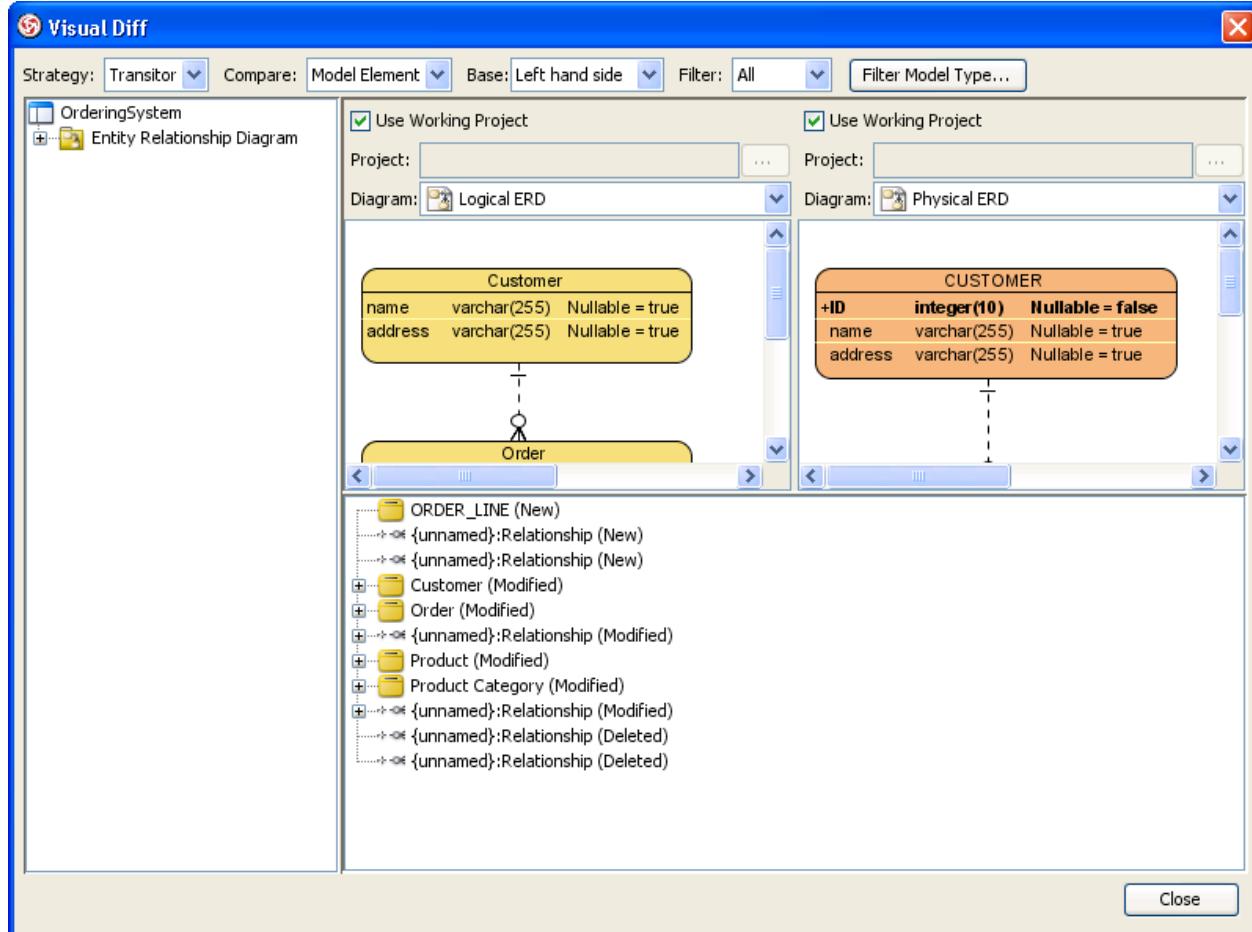


Figure 11-30 Result of comparison is obtained

9. It is possible to click on a node to select the shape in diagrams. Selected shape is painted in dark purple.

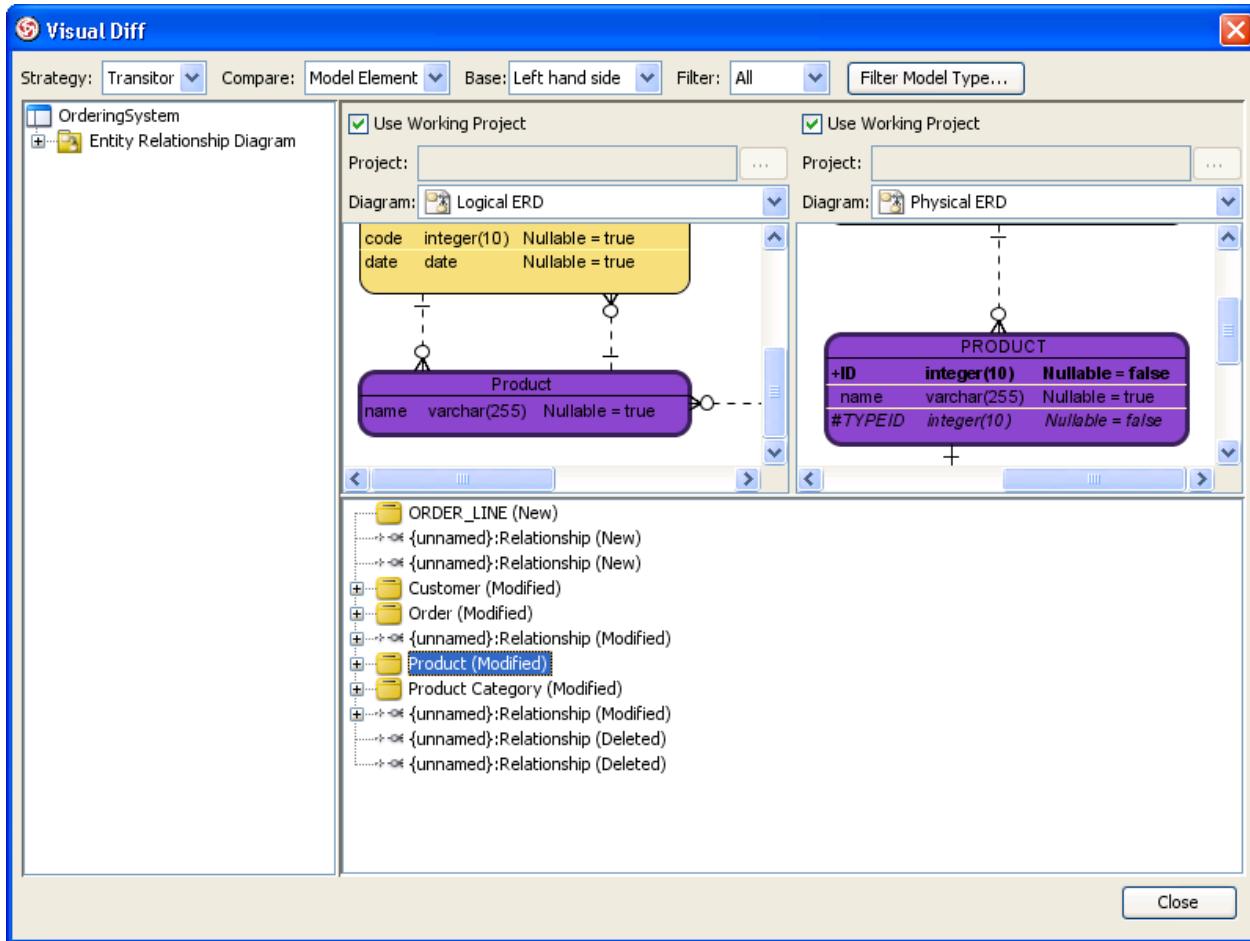


Figure 11-31 Select a node to cause the shape to be selected in diagram

Below is a description of results.

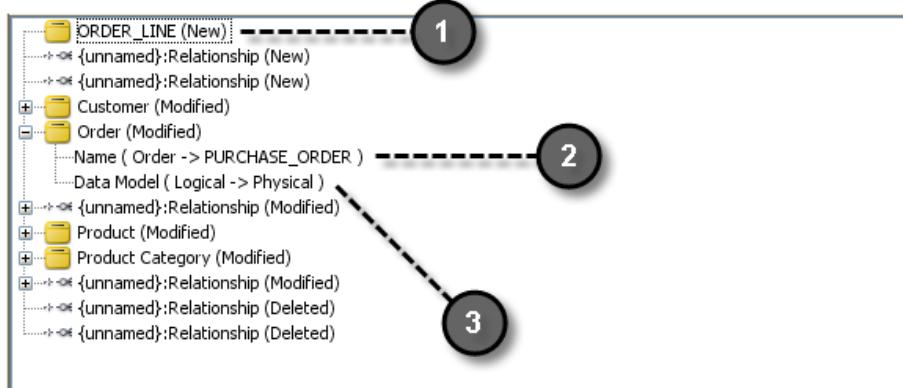


Figure 11-32 Result of comparison

Result	Description
1	The Entity is newly added.
2	The Entity is renamed.
3	The Data Model is changed from Logical to Physical.

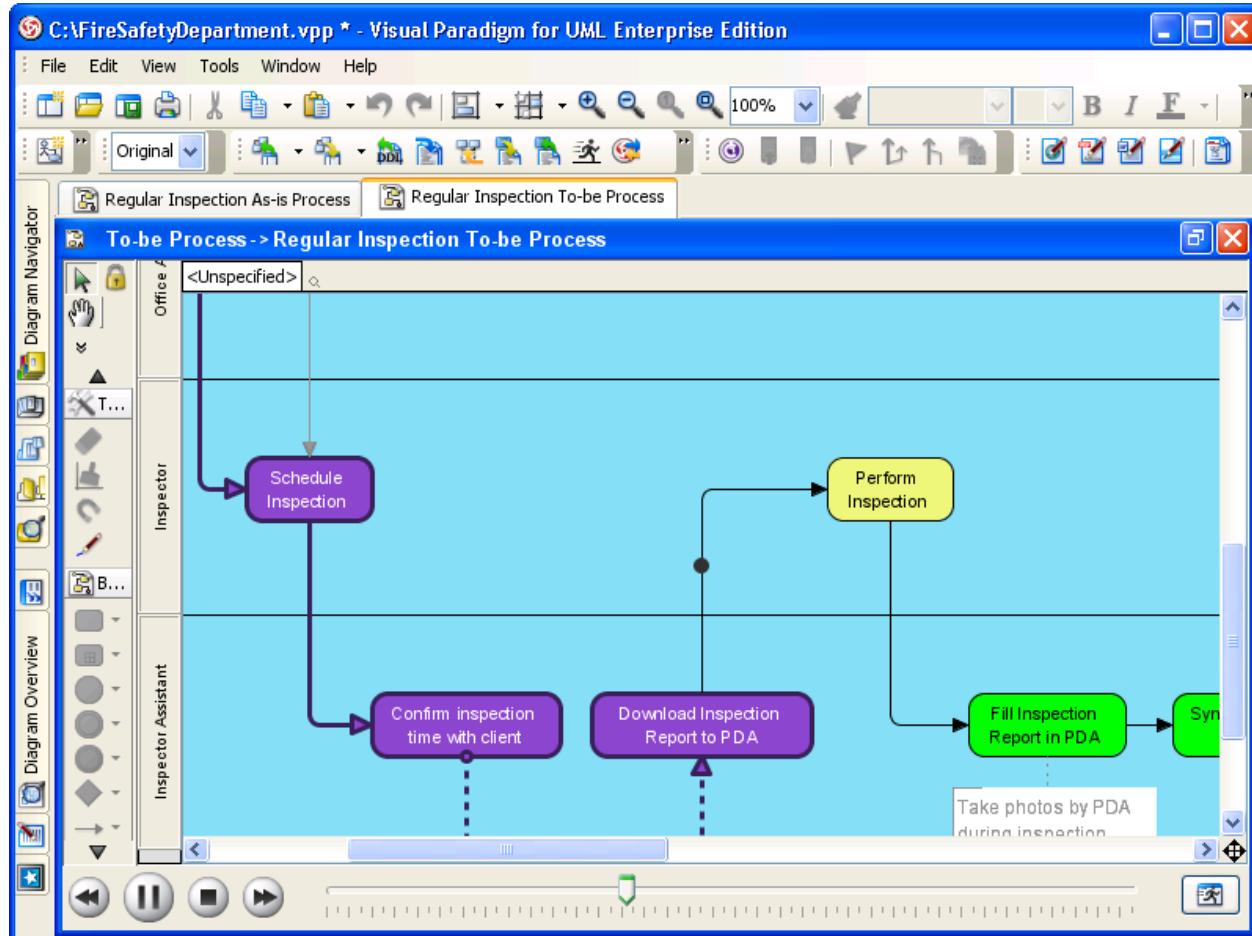
Table 11-13 Description of result

What is animacian?

Animacian is a tool that helps you makes possible paths in a diagram active 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

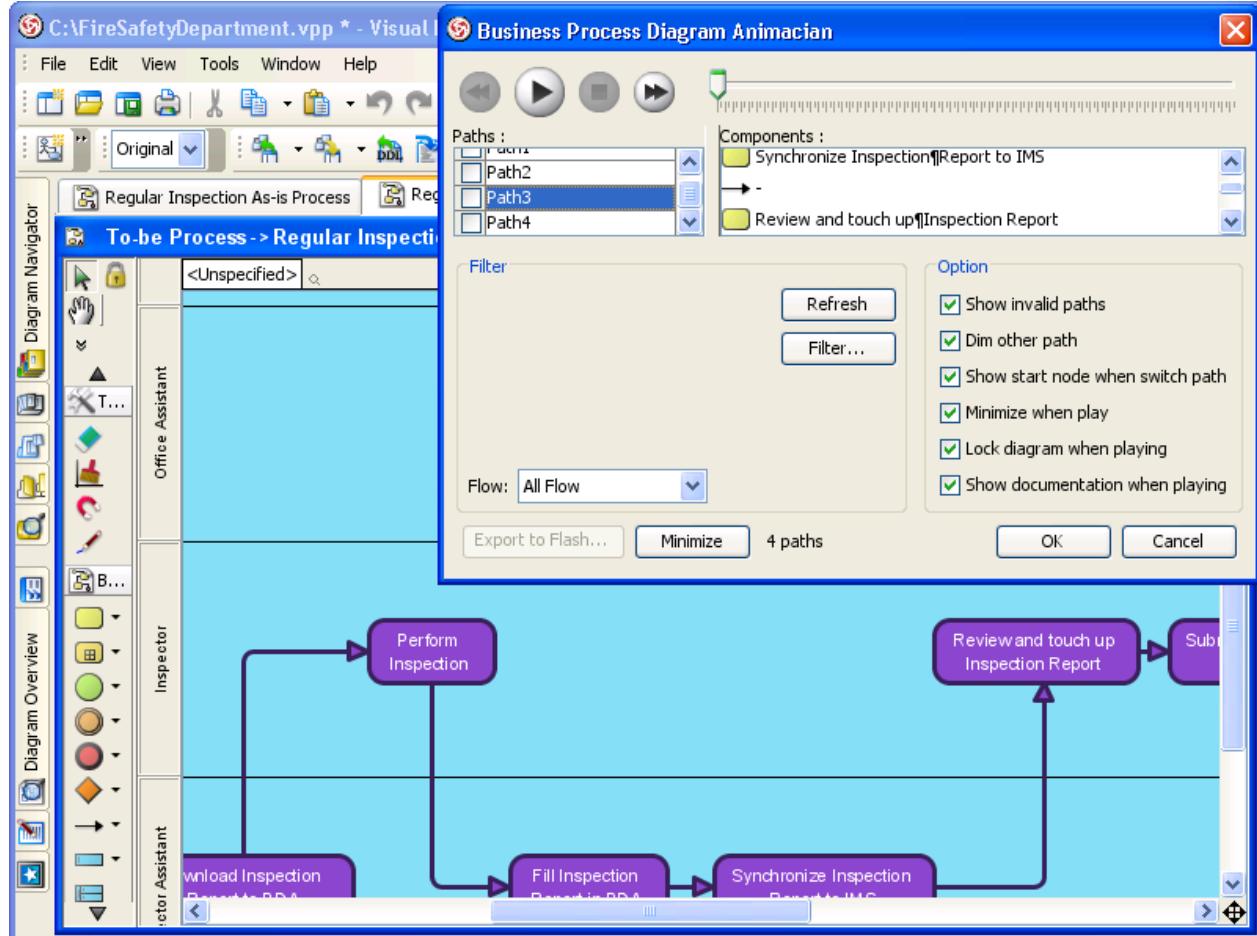
Animation can be played directly on diagram. When the animation begin, a tiny black dot will be attached to the begining of the path selected to animate. During the animation, the black dot will traverse through the path, shapes that lie on the path will be painted in purple one by one, when being approached 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. Animacian helps finding out all possible paths in a diagram. When opening the Animacian 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.

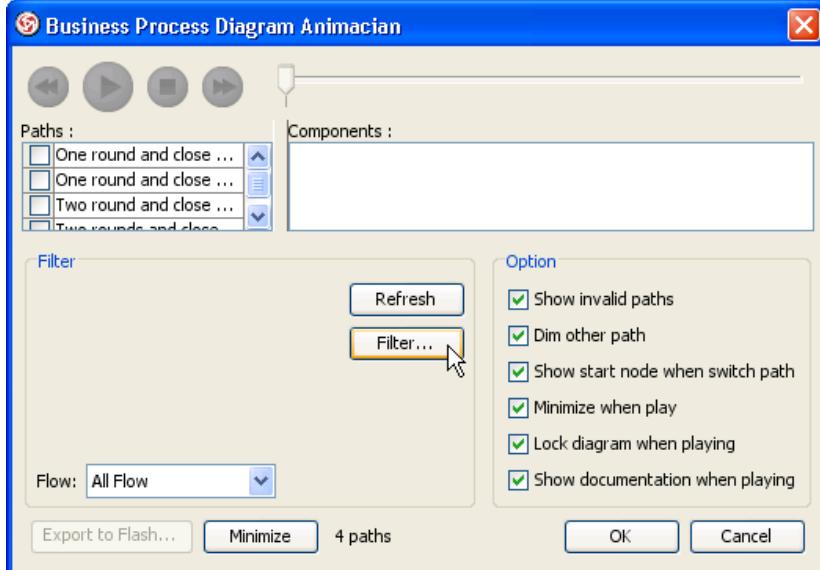


Paths are identified from a diagram

Filter business paths base on conditions

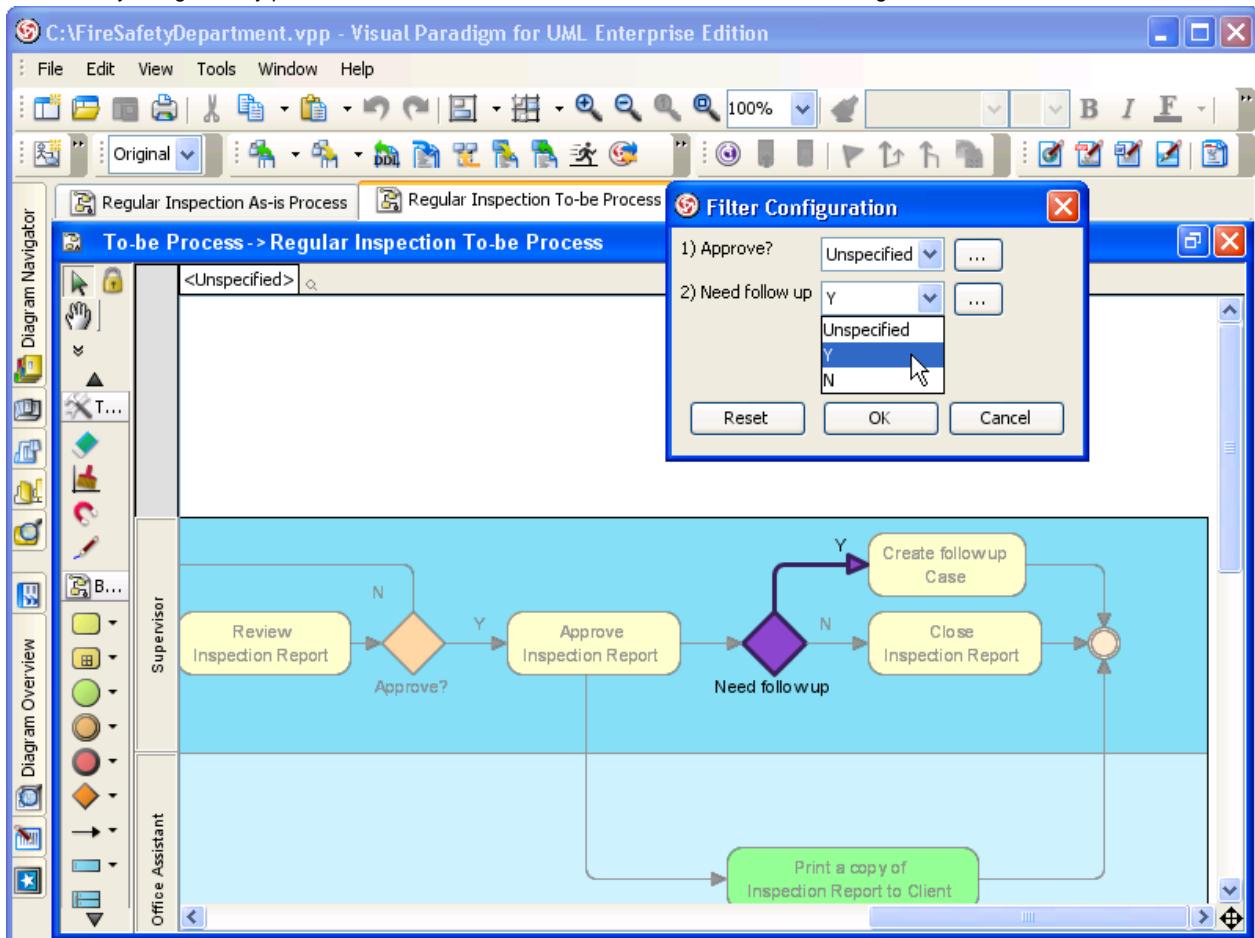
The Animacian dialog box is where you can configure animation and select path to animate. You can apply filter on path identification, which clears undesired path that does not match certain condition. To configure filter:

1. Click **Filter...** in the Animacian dialog box.



Clicking on the Filter button in the Animacian dialog box

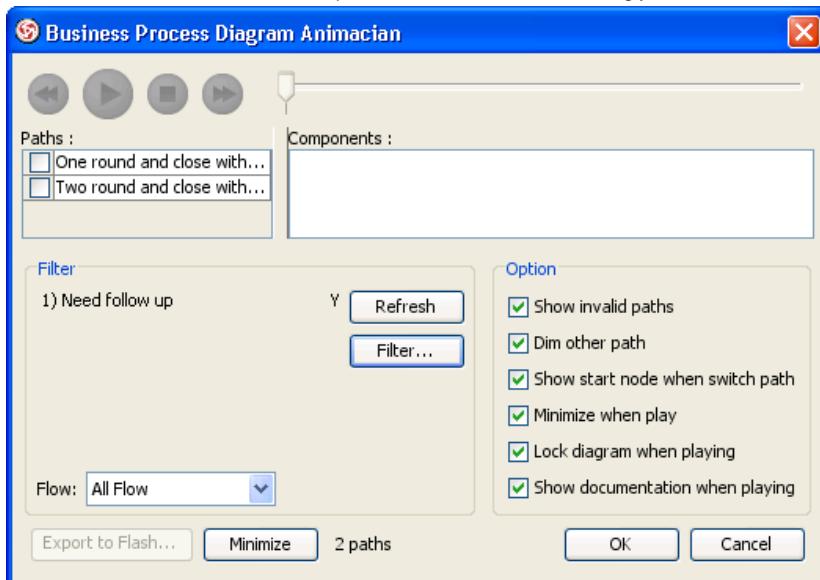
2. In the **Filter Configuration** dialog box, conditional flows, if any, are listed. Click on a drop down menu to select the expected outgoing flow of a condition. By doing so, only paths that cover the selected flow are identified, the rest will be ignored.



Configuring filter

NOTE: A condition can have more than two outgoing flows. You can allow multiple outgoing flows by pressing the ... button next to the drop down menu, and selecting the flows in the **Select Multiple Values** dialog box.

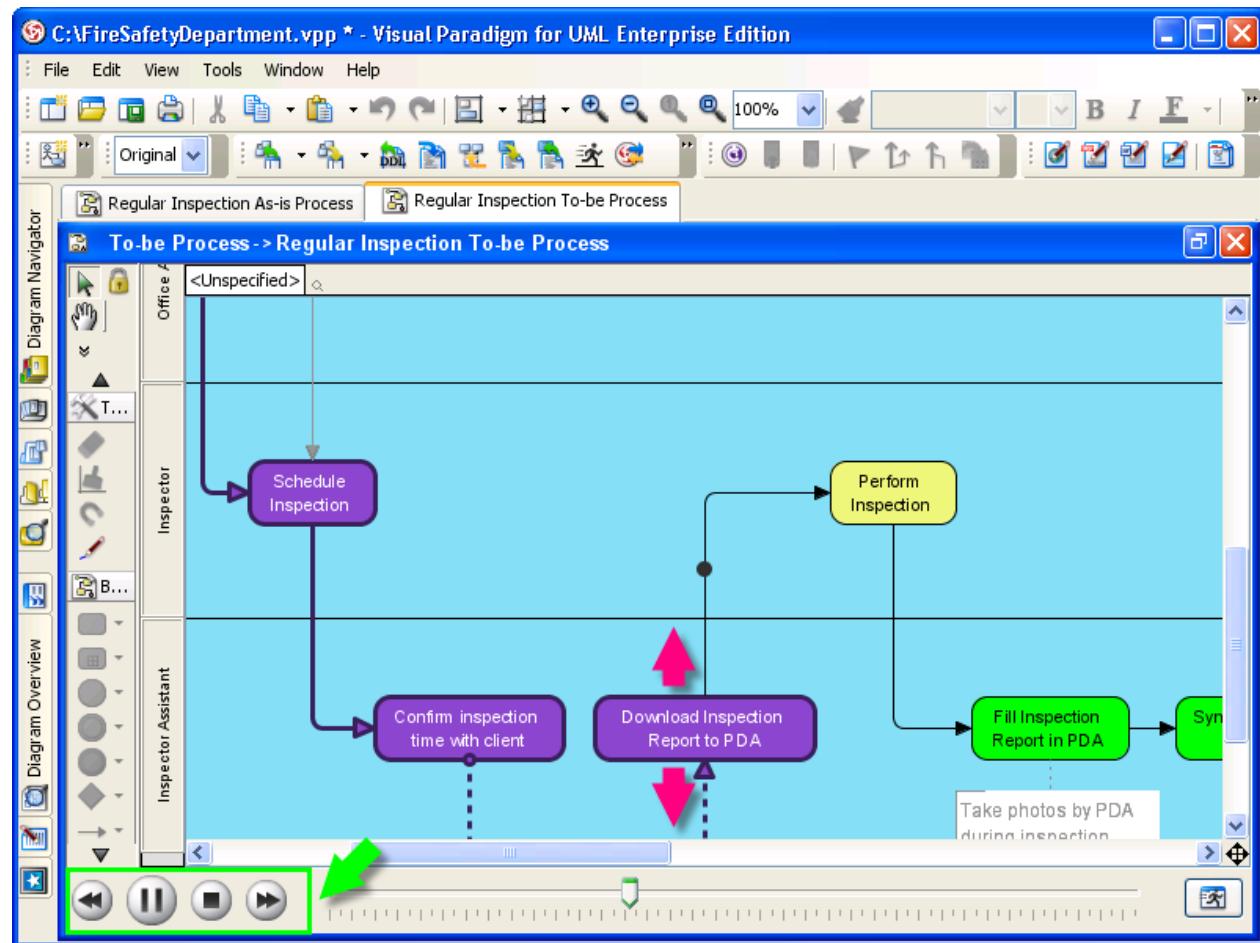
3. Click **OK** to confirm. The identified paths will be reduced accordingly.



Paths are reduced as a result of applying filter

Walking through a path step-by-Step

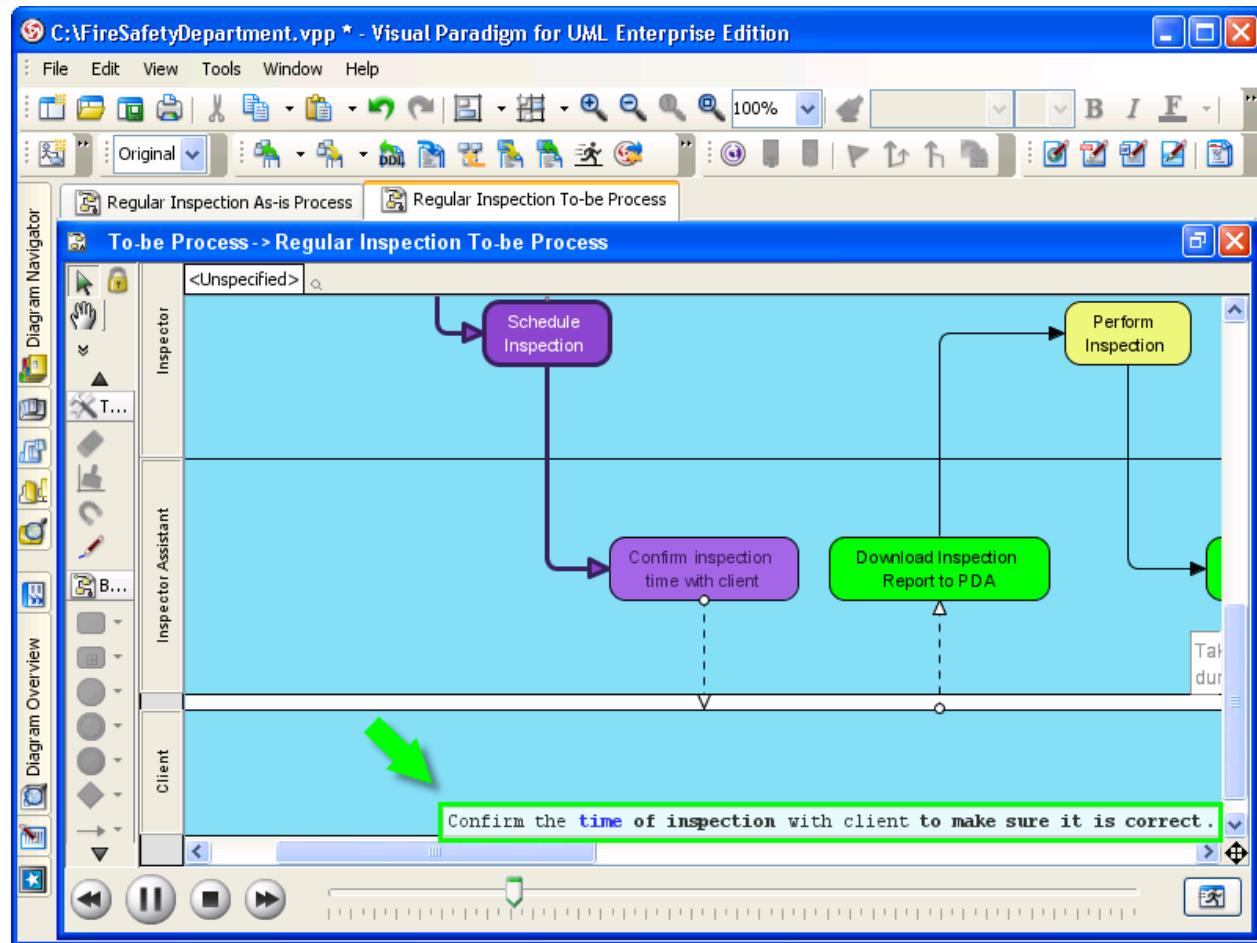
Instead of letting the animation to run itself, you can control it yourself. The horizontal bar that appear at the bottom of VP-UML when animating lets you control the animation. Besides pausing, playing and stopping the animation, you can also move a shape backward or forward by pressing the  and  button. By making use with the forward and backward buttons, you can walk through a path shape by shape.



Walking though a path step by step

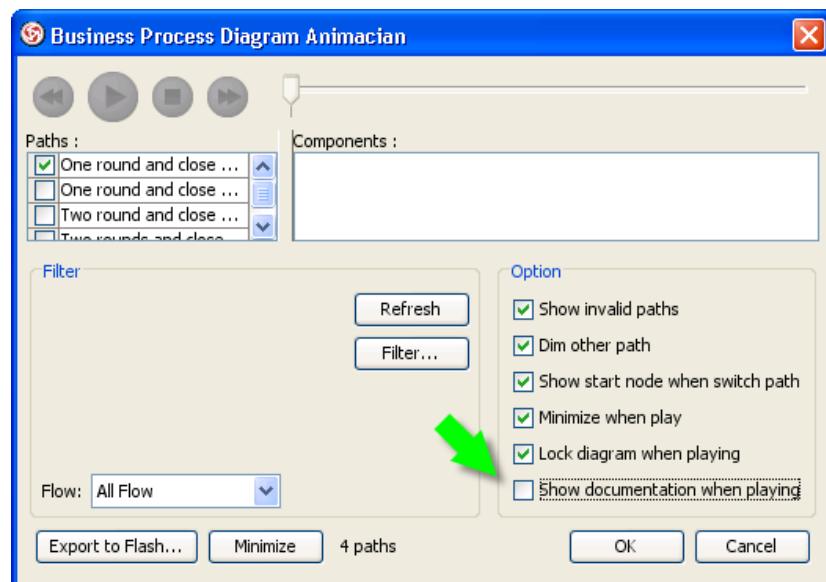
Showing documentation of current step when playing animation

When walking through a path, the documentation of the visiting shape, if written, will appear instantly at the bottom right corner of VP-UML.



Documentation of shape appear when animating

To turn this function on or off, open the **Animacian** dialog box, then check or uncheck **Show documentation when playing**.



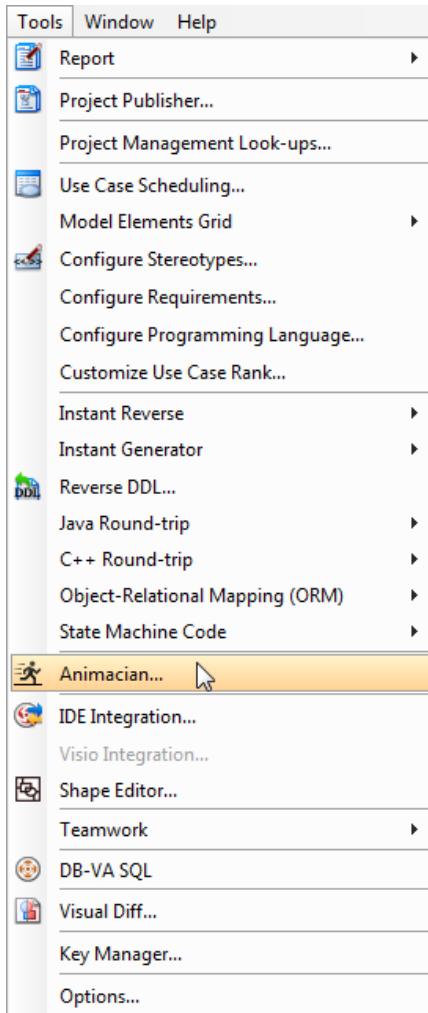
Option for showing documentation when playing

Animating business process diagram

By animating a business process diagram with Animacian, 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.

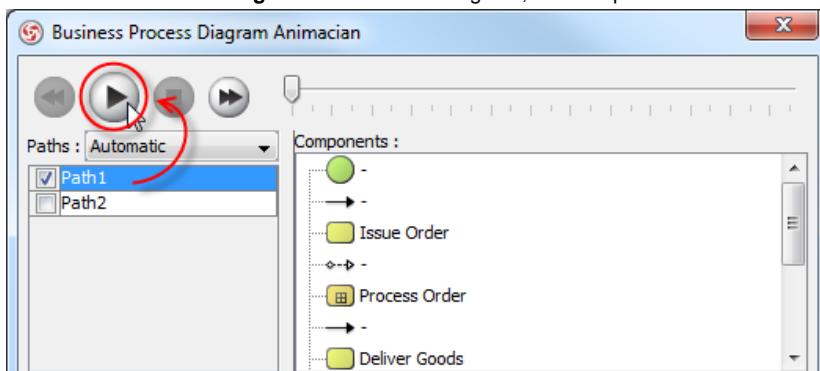
Launching an animation

1. Select **Tools > Animacian...** from the main menu.



*Launching **Animacian**
through the main menu*

2. In **Business Process Diagram Animacian** dialog box, select a path and then click **Play**.



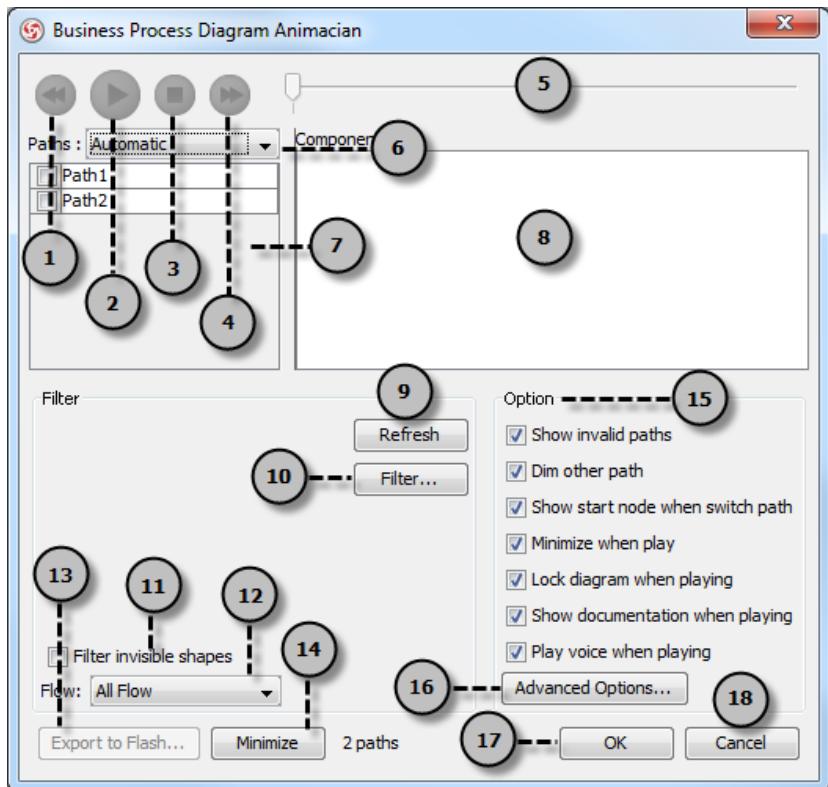
*Clicking **Play** in **Business Process Diagram Animacian** dialog box*

NOTE: Animacian can also be started by using any of the ways below:

- Right-click on the diagram background and select **Utilities > Animacian...** from the popup menu.
- Click on the **Animacian** button in the toolbar.

Overview of Animacian

The **Business Process Diagram Animacian** dialog box will pop out after clicking **Animacian....** This dialog box is where you can select an execution path to play an animation.



Business Process Diagram Animacian dialog box

No.	Name	Description
1	Backward	Move one shape backward in the flow.
2	Play	Play or continue to play the animation with Animacian 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 this option when you want to select the possible path(s) manually.
7	Paths list	It lists all possible ways of executing a business process. 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	Flow	It provides three options for viewing the possible paths in Paths list . All Flow: Make a path continue when reaching sequence and message flow. Sequence Flow Only: Ignore message flows on diagram. When a path reaches a message flow, end the path. Message Flow Only: Ignore sequence flows on diagram. When a path reaches a message flow, end the path.
13	Export to Flash...	Select an output path for exporting this diagram's animation to Adobe Flash.
14	Minimize	Click to minimize this dialog box.
15	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 documentation when playing:** It shows documentation of shape at the bottom right of diagram when playing the animation.
- Play voice when playing:** Voice can be recorded as documentation of model element. Check this if you want to play recorded voice when running animation.

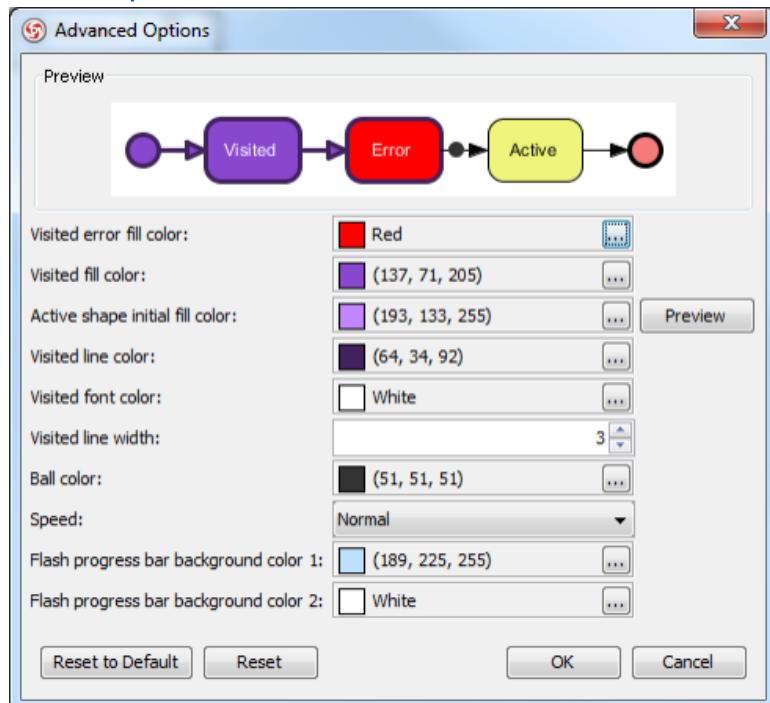
16 Advanced Options... It provides the color and speed options for animation.

17 OK Click this button to confirm the settings and close Animacian.

18 Cancel Click this button to close Animacian without saving the editing.

Description of **Business Process Diagram Animacian** dialog box

Advanced options



Advanced Options dialog box

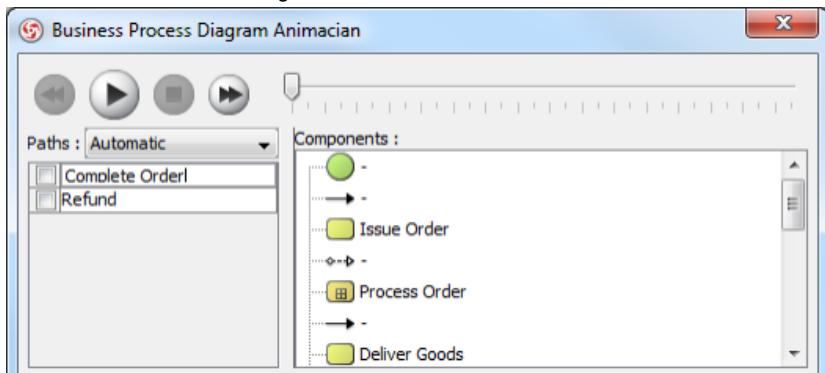
Name	Description
Visited error fill color	The background color of visited shape that cause an error. An error means the flow object that causes a path invalid.
Visited fill color	The background color of visited shapes.
Active shape initial fill color	When playing an animation, a tiny black ball will 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.

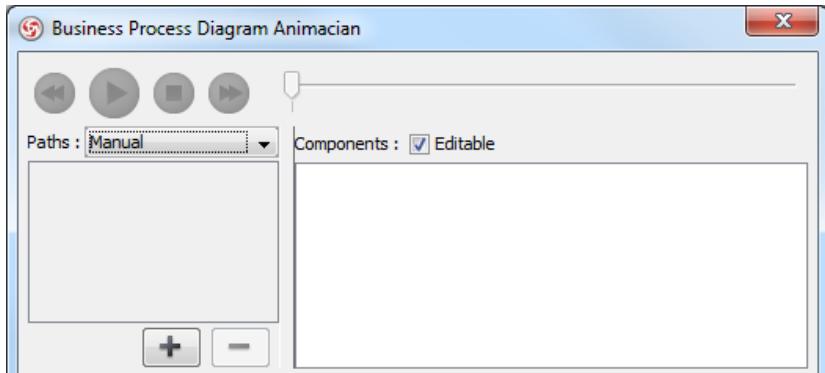


Naming the paths

Creating a manual path

In **Business Process Diagram Animacian** 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**.

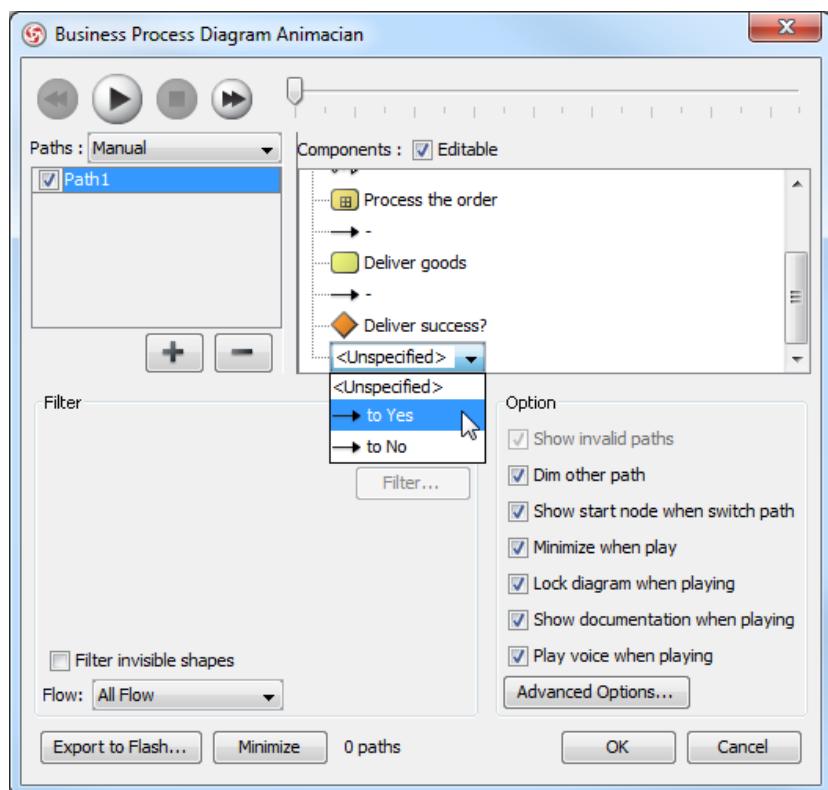


*Selecting **Manual** in **Paths***

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 a gateway 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

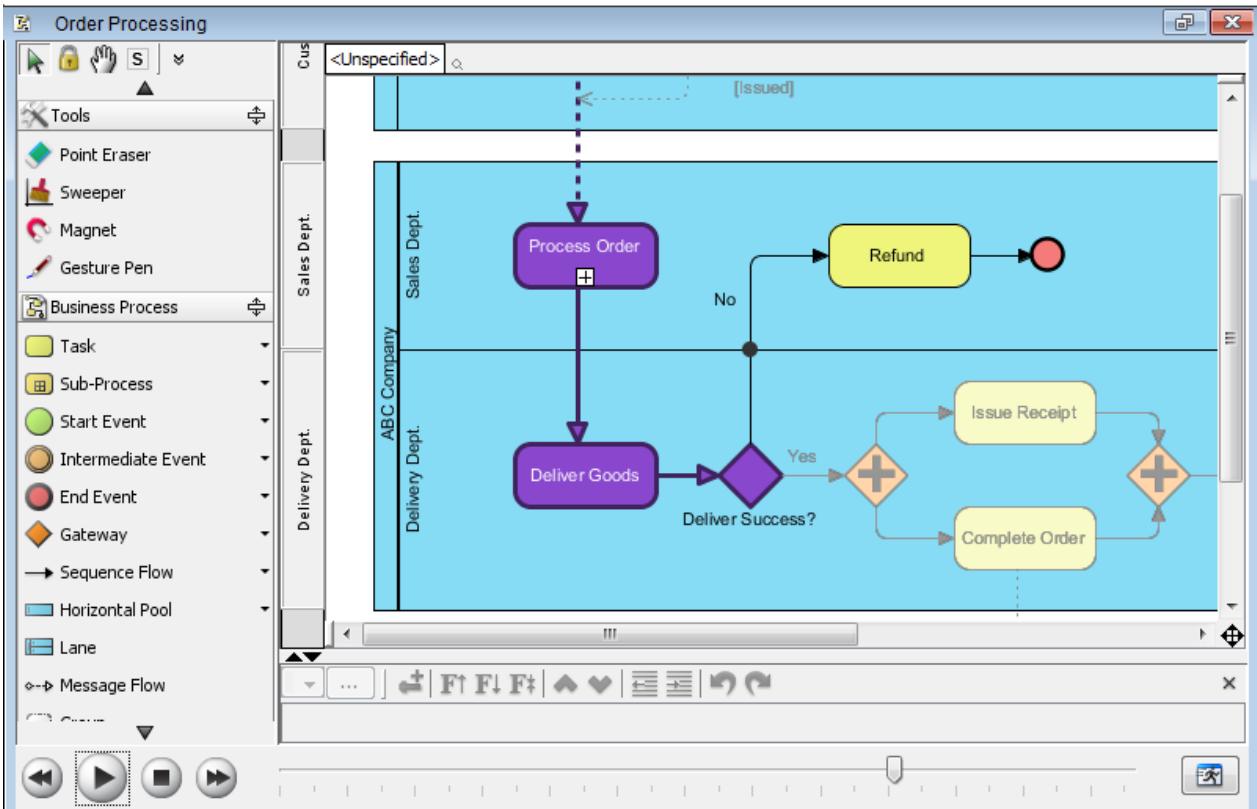
1. When everything is ready, click **Play** to start the animation of the selected path.
2. After click **Play**, **Business Process Diagram Animacian** dialog box will be minimized to the bottom of your diagram, with several buttons and a slider revealing on it.

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 Animacian .

Description of Animacian 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.



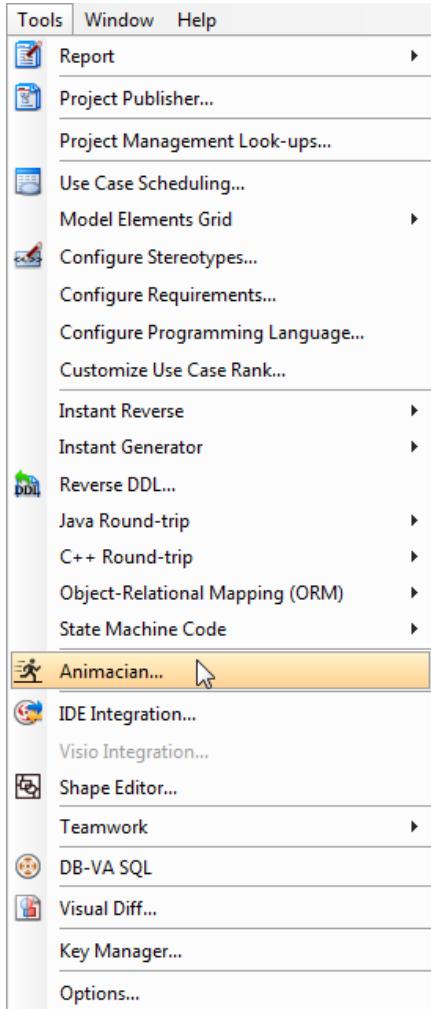
Reviewing the animation

Animating sequence diagram

By animating a sequence diagram with Animacian, you can see interaction between lifelines and the flow of message calls in active.

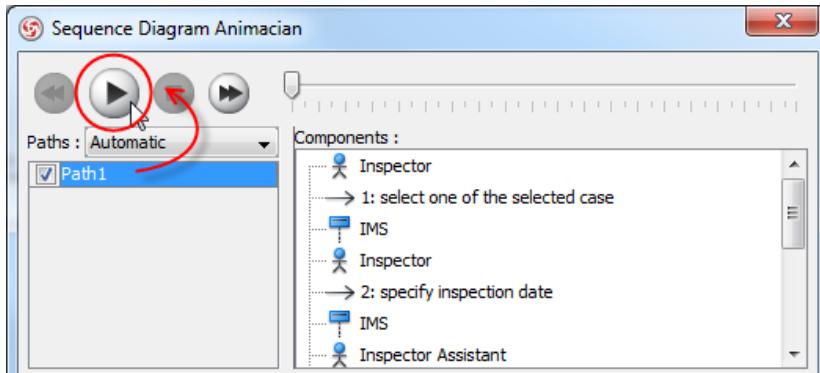
Launching an animation

1. Select **Tools > Animacian...** from the main menu.



*Launching Animacian
through the main menu*

2. In **Sequence Diagram Animacian** dialog box, select a path and then click **Play**.



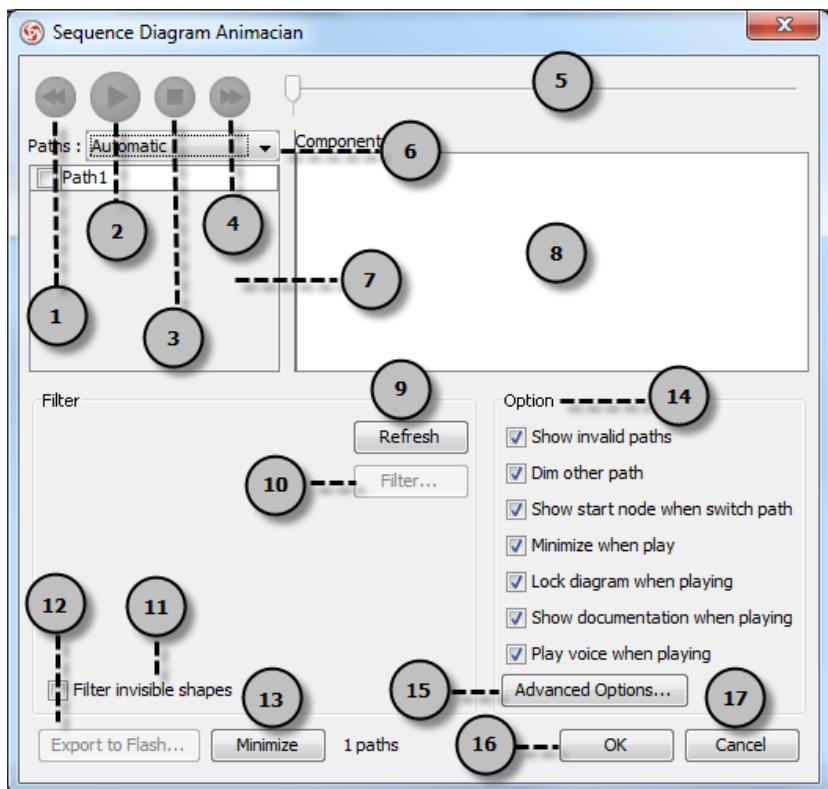
Clicking Play in Sequence Diagram Animacian dialog box

NOTE: Animacian can also be started by using any of the ways below:

- Right-click on the diagram background and select **Utilities > Animacian...** from the popup menu.
- Click on the **Animacian** button in the toolbar.

Overview of Animacian

The **Sequence Diagram Animacian** dialog box will pop out after clicking **Animacian....** This dialog box is where you can select an execution path to play an animation.



Sequence Diagram Animacian dialog box

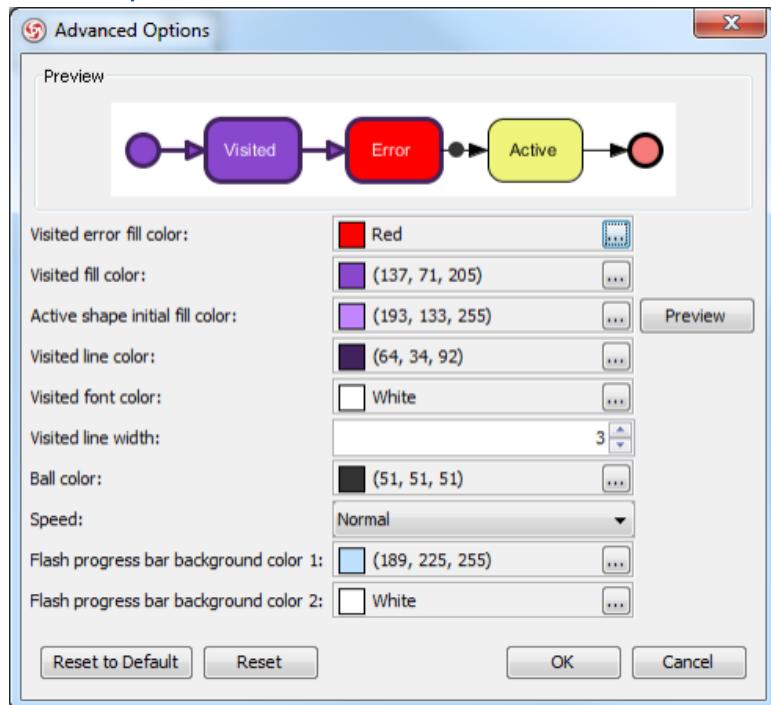
No.	Name	Description
1	Backward	Move one shape backward in the flow.
2	Play	Play or continue to play the animation with Animacian minimized.
3	Stop	Terminate the animation.
4	Forward	Advance to the next shape in the flow.
5	Slider	It is used for controlling the flow of animation.
6	Paths	It provides 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 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 documentation when playing: It shows documentation of shape at the bottom right of diagram when playing the animation.
Play voice when playing: Voice can be recorded as documentation 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 Animacian.
17 Cancel	Click this button to close Animacian without saving the editing.

Description of **Sequence Diagram Animacian** dialog box

Advanced options



Advanced Options dialog box

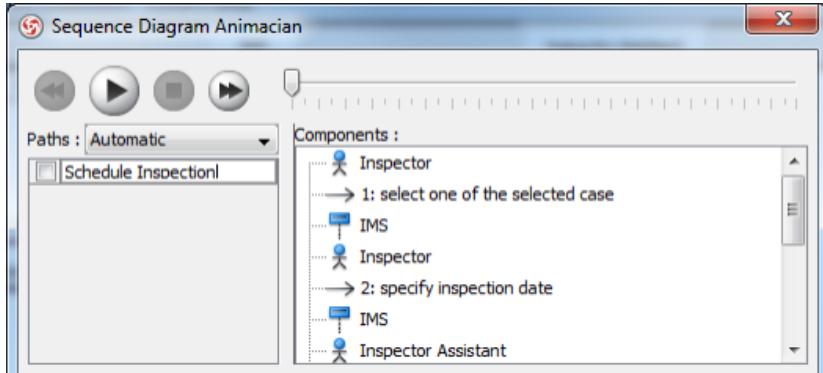
Name	Description
Visited error fill color	The background color of visited shape that cause an error. An error means the flow object that causes a path invalid.
Visited fill color	The background color of visited shapes.
Active shape initial fill color	When playing an animation, a tiny black ball will 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.

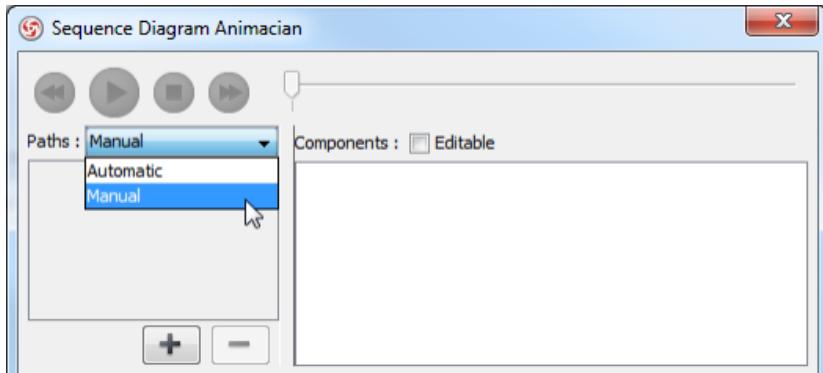


Naming the path

Creating a manual path

In **Sequence Diagram Animacian** 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**.

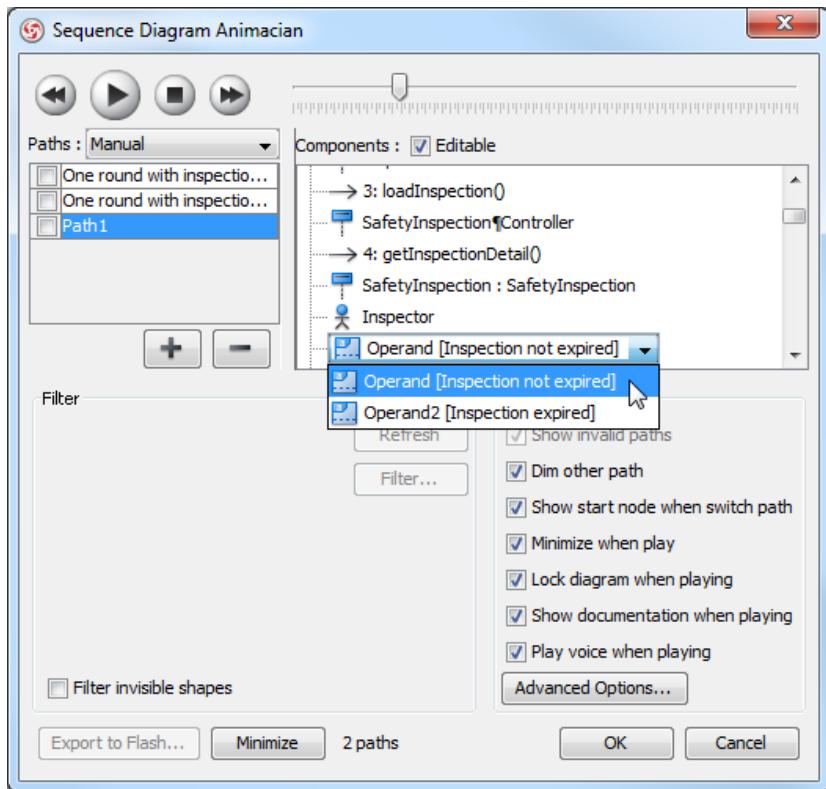


*Selecting **Manual** in **Paths***

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

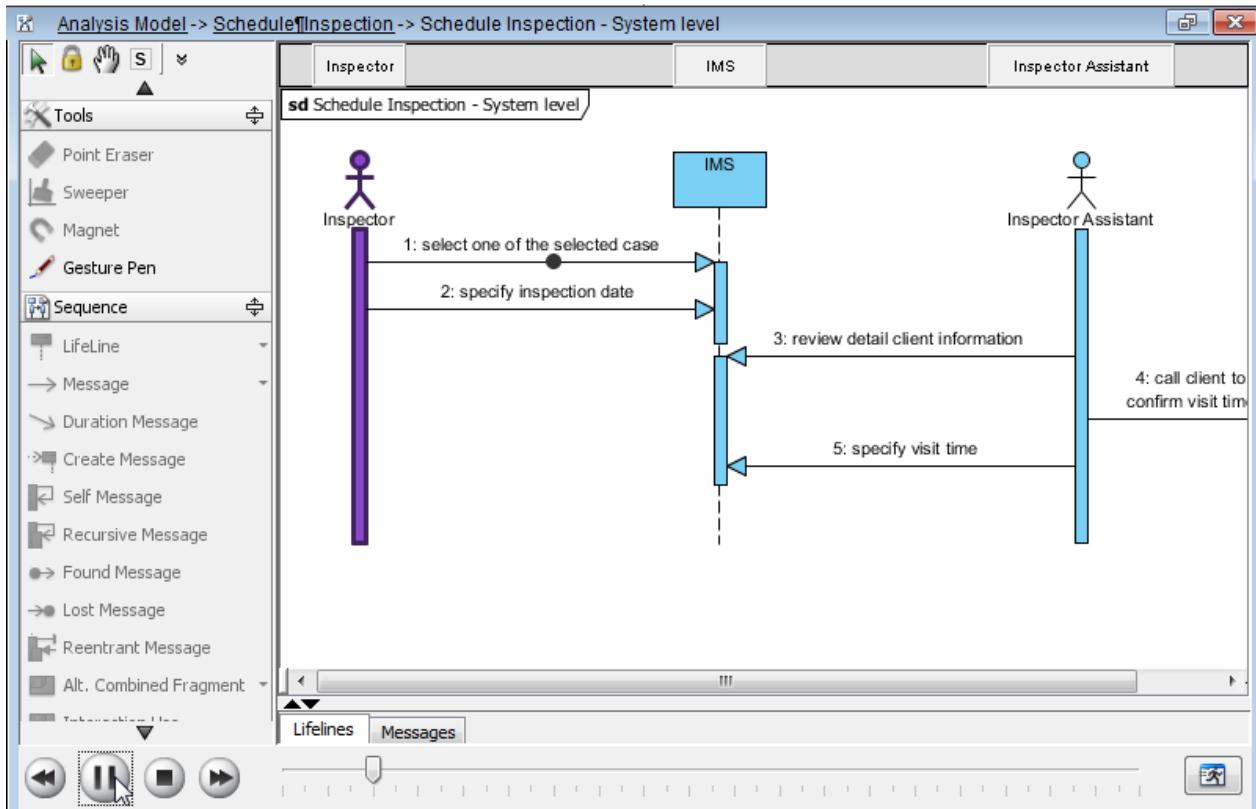
1. When everything is ready, click **Play** to start the animation of the selected path.
2. After click **Play**, **Sequence Diagram Animacian** dialog box will be minimized to the bottom of your diagram, with several buttons and a slider revealing on it.

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 Animacian.

Description of Animacian 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.

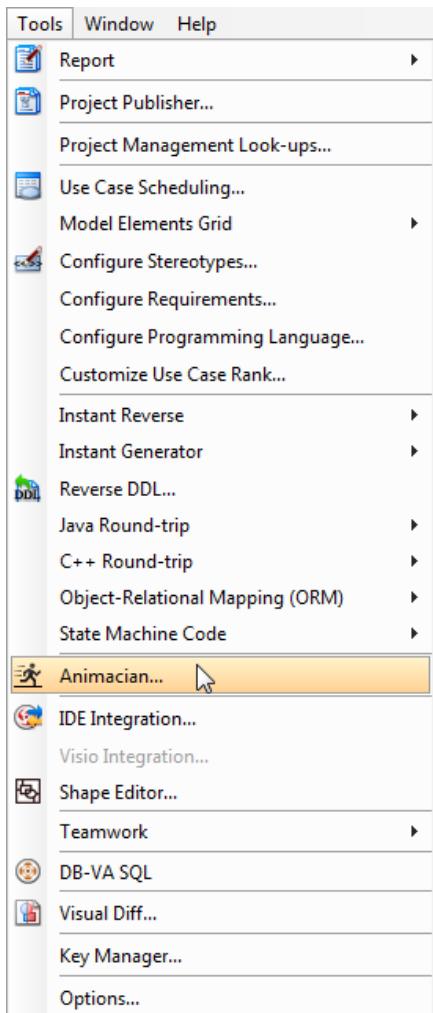


Reviewing the animation

Animating activity diagram

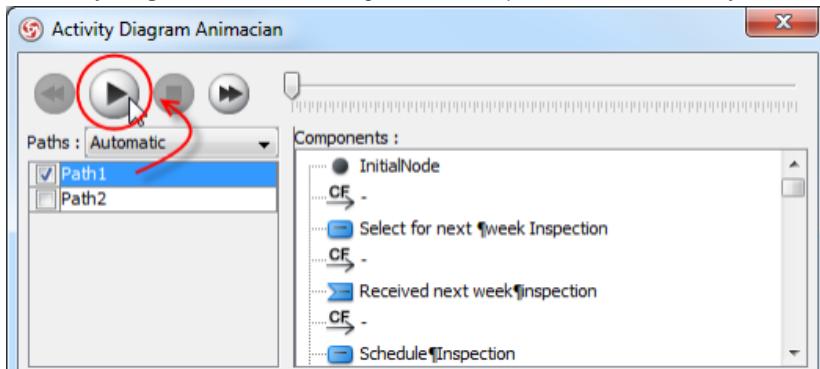
By animating a sequence diagram with Animacian, you can see the flow of actions in active.

1. Select **Tools > Animacian...** from the main menu.



*Launching **Animacian** through the main menu*

2. In **Activity Diagram Animacian** dialog box, select a path and then click **Play**.



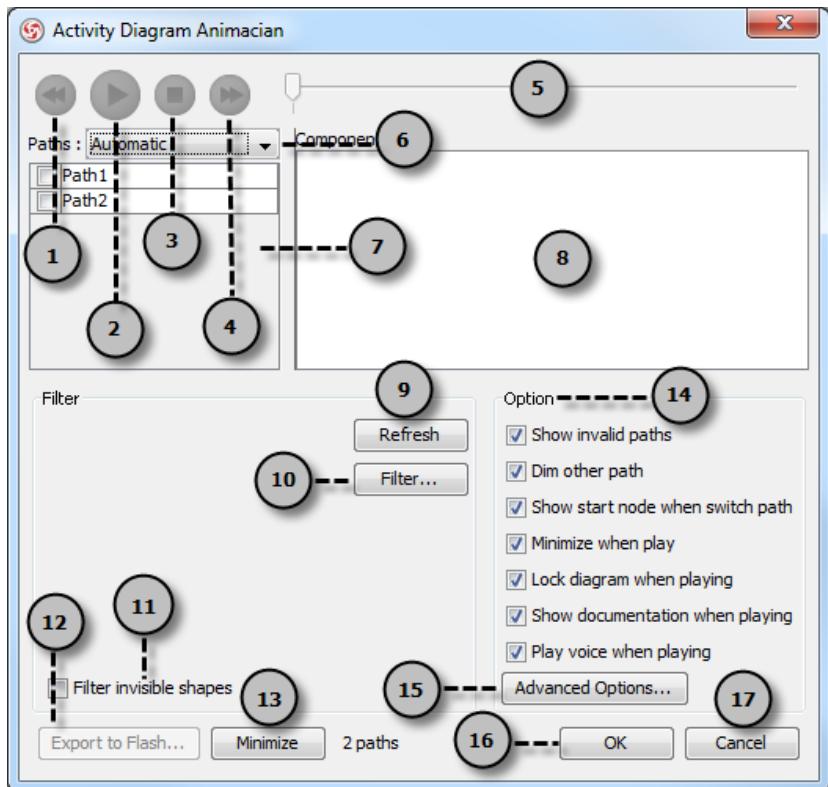
*Clicking **Play** in Activity Diagram Animacian dialog box*

NOTE: Animacian can also be started by using any of the ways below:

- Right-click on the diagram background and select **Utilities > Animacian...** from the popup menu.
- Click on the **Animacian** button in the toolbar.

Overview of Animacian

The **Activity Diagram Animacian** dialog box will pop out after clicking **Animacian....** This dialog box is where you can select an execution path to play an animation.



Activity Diagram Animacian dialog box

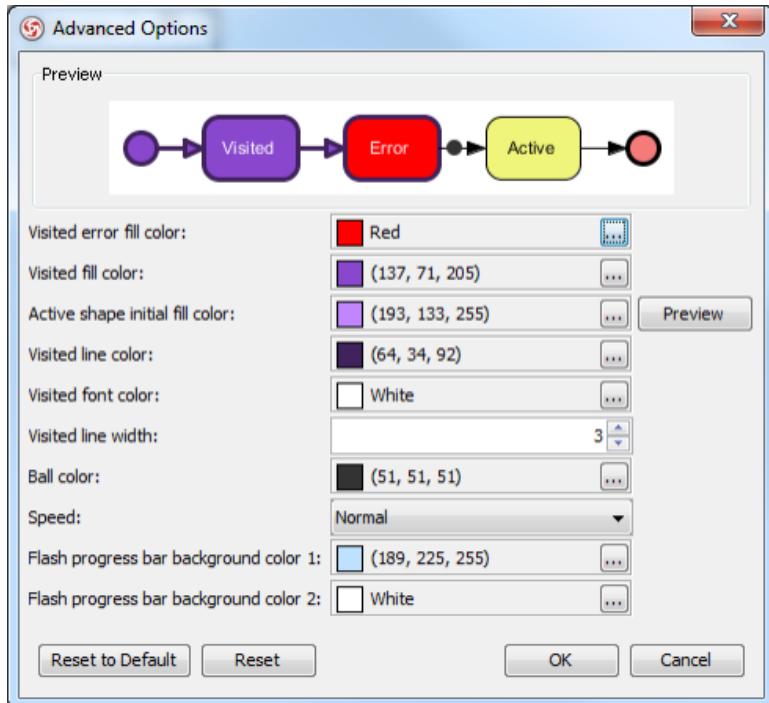
No.	Name	Description
1	Backward	Move one shape backward in the flow.
2	Play	Play or continue to play the animation with Animacian minimized.
3	Stop	Terminate the animation.
4	Forward	Advance to the next shape in the flow.
5	Slider	It is used for controlling the flow of animation.
6	Paths	It provides 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 documentation when playing: It shows documentation of shape at the bottom right of diagram when playing the animation.
Play voice when playing: Voice can be recorded as documentation 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 Animacian. |
| 17 Cancel | Click this button to close Animacian without saving the editing. |
-

Description of **Activity Diagram Animacian** dialog box

Advanced options



Advanced Options dialog box

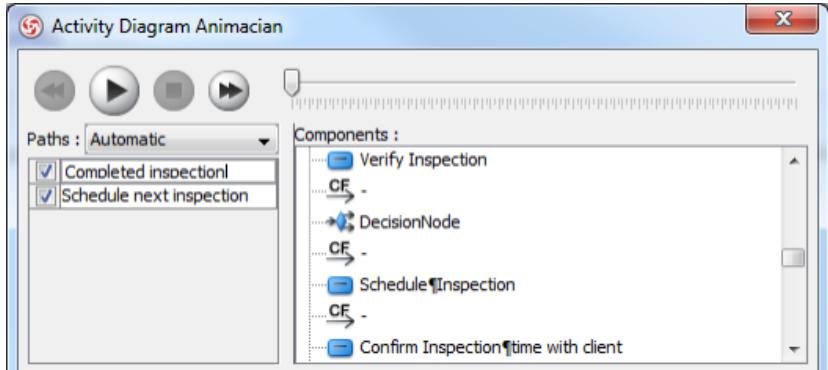
Name	Description
Visited error fill color	The background color of visited shape that cause an error. An error means the flow object that causes a path invalid.
Visited fill color	The background color of visited shapes.
Active shape initial fill color	When playing an animation, a tiny black ball will 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.

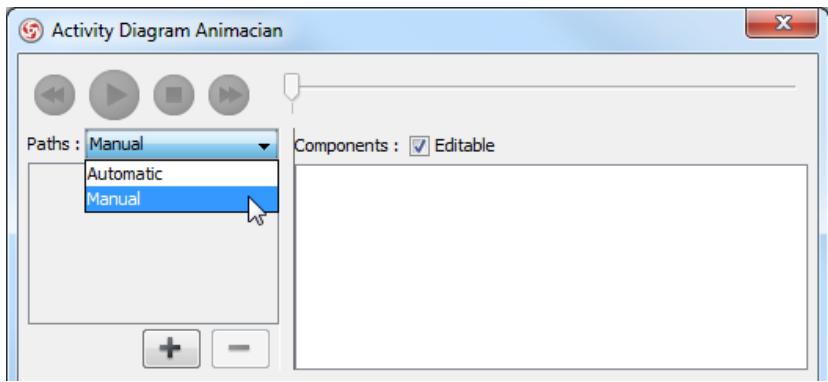


Naming the paths

Creating a manual path

In **Activity Diagram Animacian** 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**.

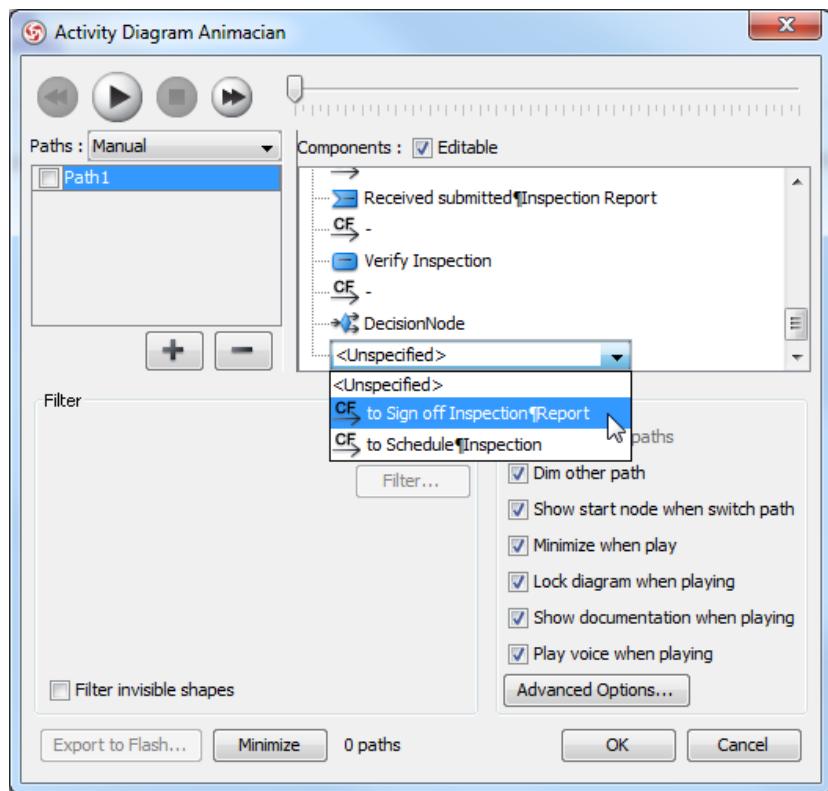


*Selecting **Manual** in **Paths***

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

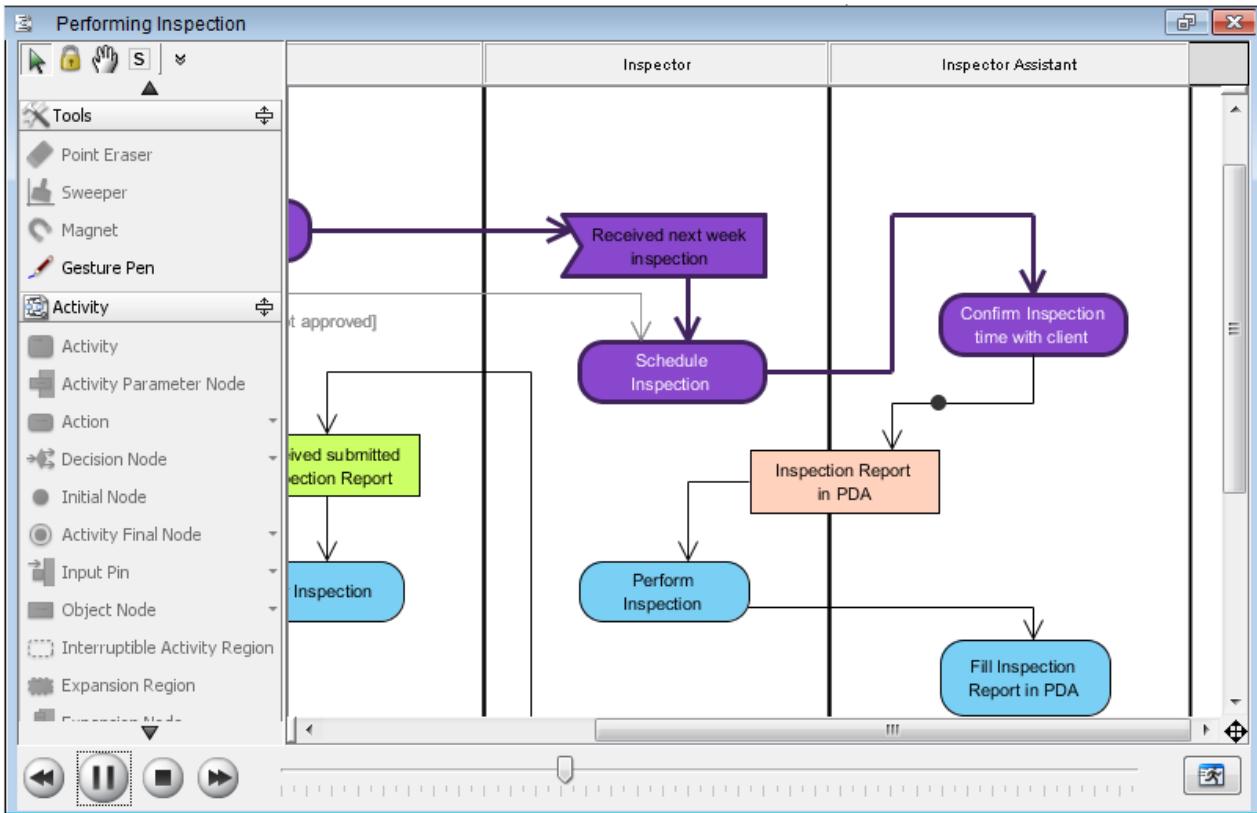
1. When everything is ready, click **Play** to start the animation of the selected path.
2. After click **Play**, **Activity Diagram Animacian** dialog box will be minimized to the bottom of your diagram, with several buttons and a slider revealing on it.

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 Animacian .

Description of Animacian 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

Export animation to adobe flash

Exporting flash movie from animacian

- To launch Animacian, select **Tools > Animacian** from the main menu.

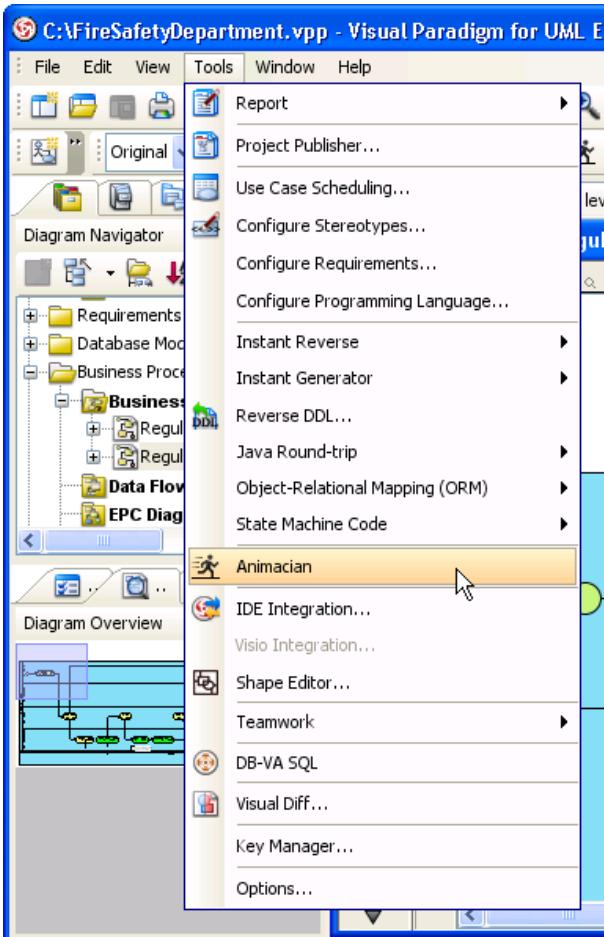


Figure 12-42 Opening Animacian through the main menu

NOTE: Animacian can also be started by any of the ways below:

- Right-click on the diagram background and select **Utilities > Animacian...** from the popup menu.
- Click on the Animacian button  in the toolbar.

This starts Animacian. The **Animacian** dialog box is where you can select an execution path to play an animation.

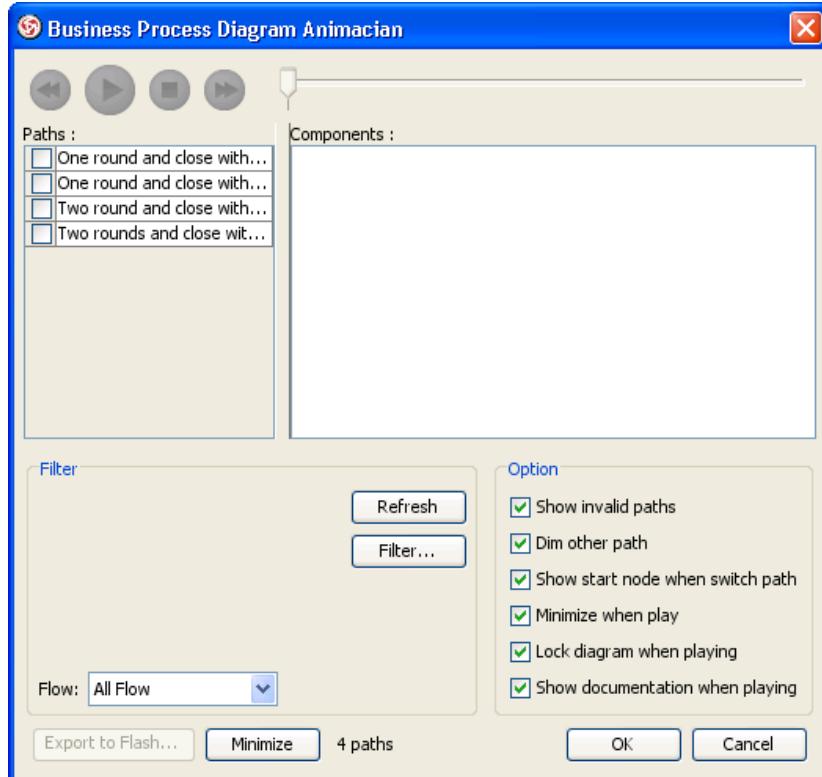


Figure 12-43 The Animacian dialog box

2. From the **Paths** list, select the execution paths to export as Flash movie.

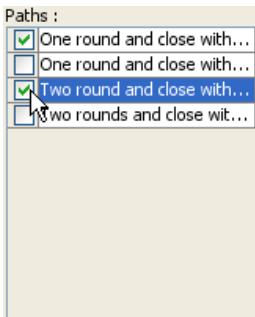


Figure 12-44 Select paths to animate

3. Click the **Export to Flash...** button.

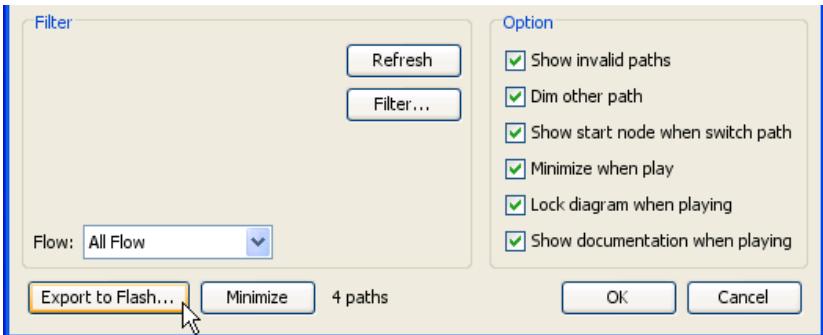


Figure 12-45 Clicking the Export to Flash button

This shows the **Export to Flash** dialog box.

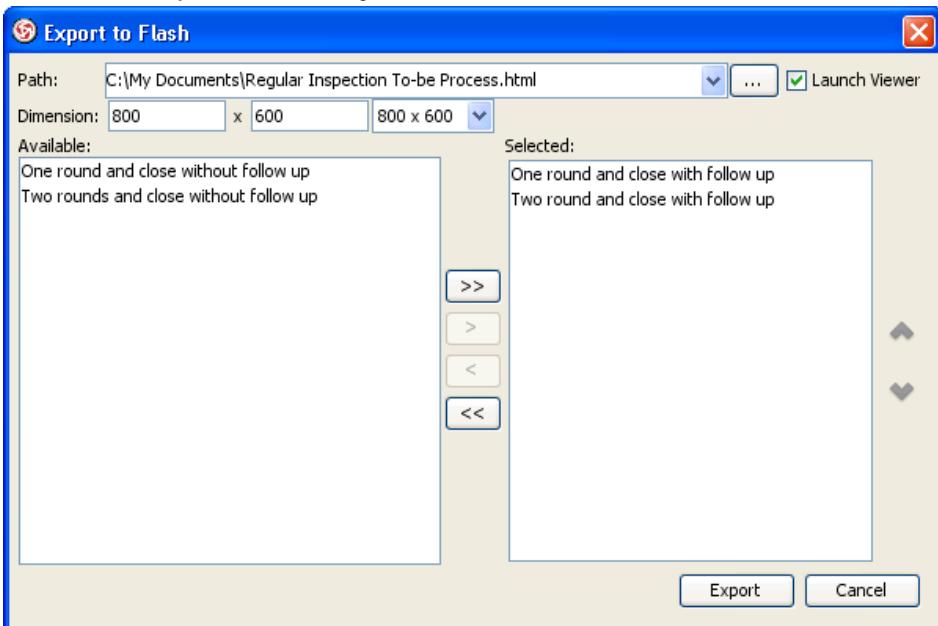


Figure 12-46 The Export to Flash dialog box

Here is a description of the **Export to Flash** dialog box.

Region	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.

Table 12-10 Description of the Export to Flash dialog box

4. 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.
5. Choose or enter the dimension of movie if necessary. Note that the dimension determines the size of viewable region instead of the size of diagram.

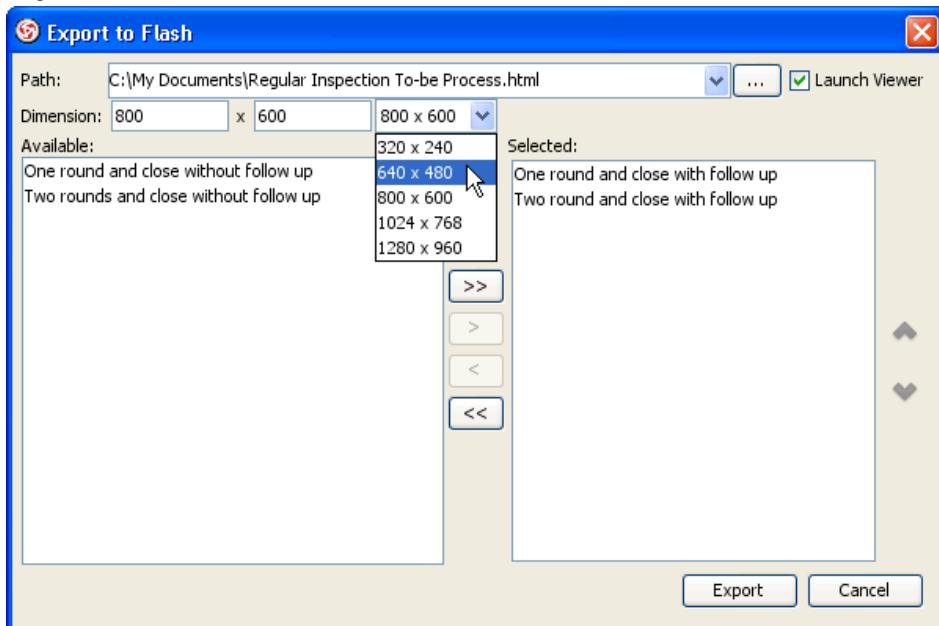


Figure 12-47 Changing the Flash movie dimension

6. Click **Export**.

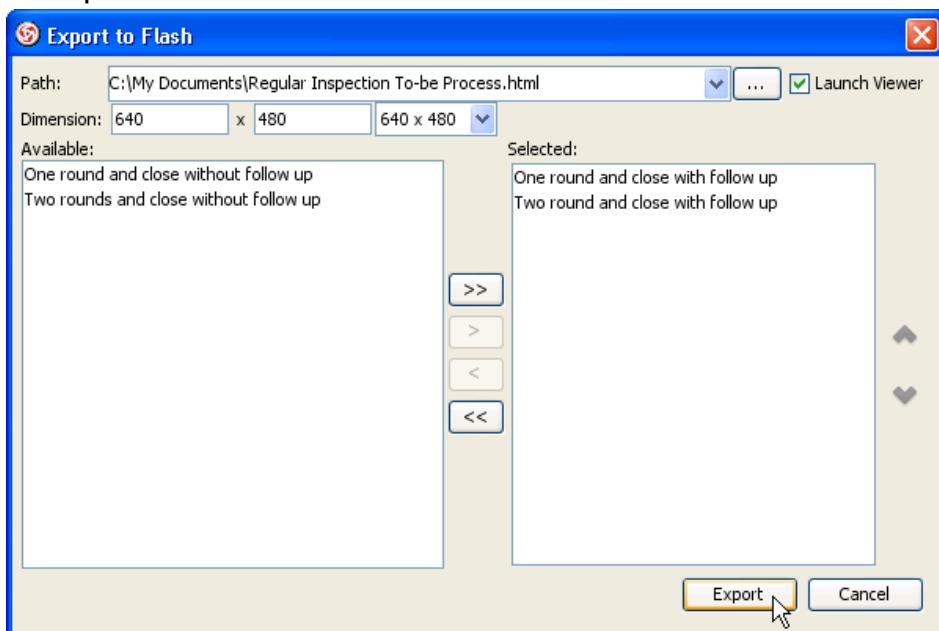


Figure 12-48 Clicking the Export button

7. Open the HTML file in the web browser to play the movie.

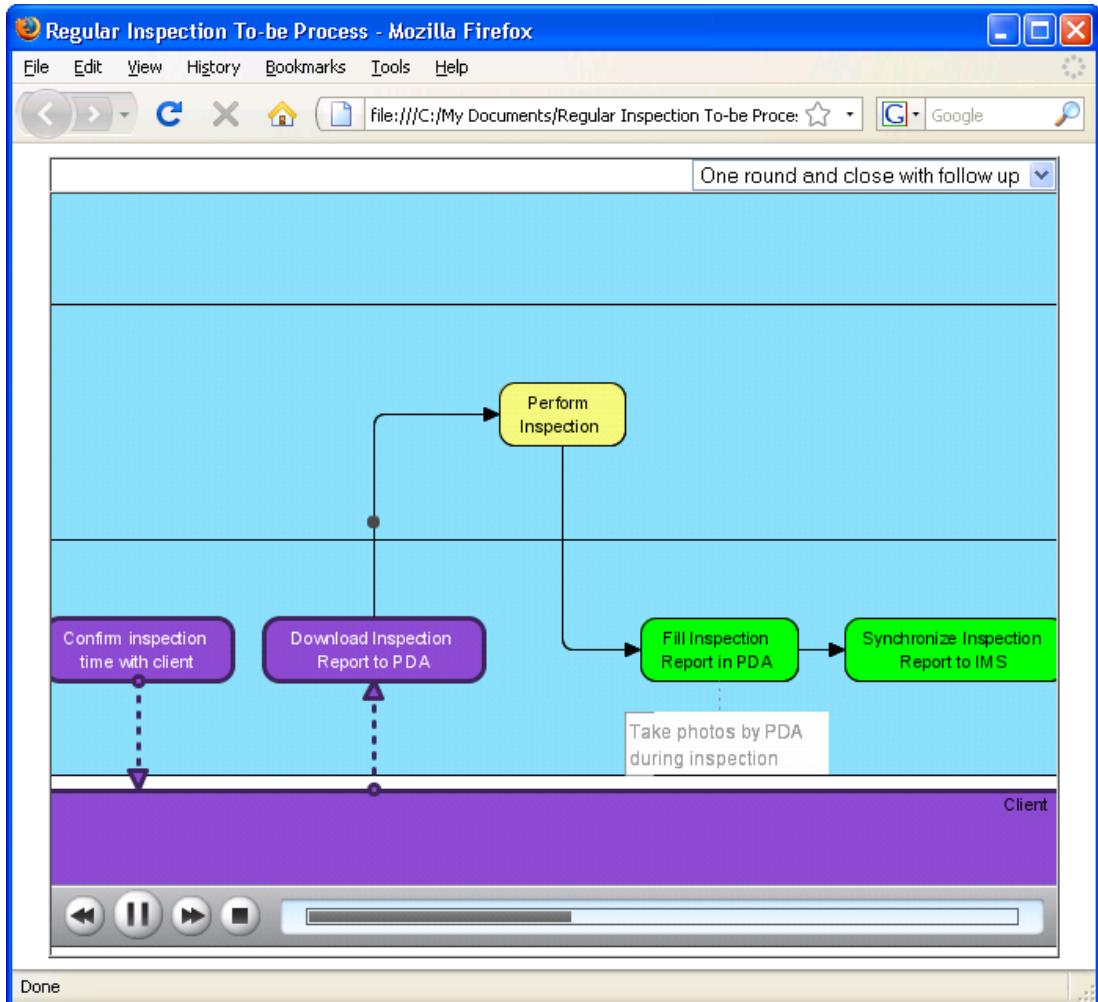


Figure 12-49 Animation is playing

8. 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.

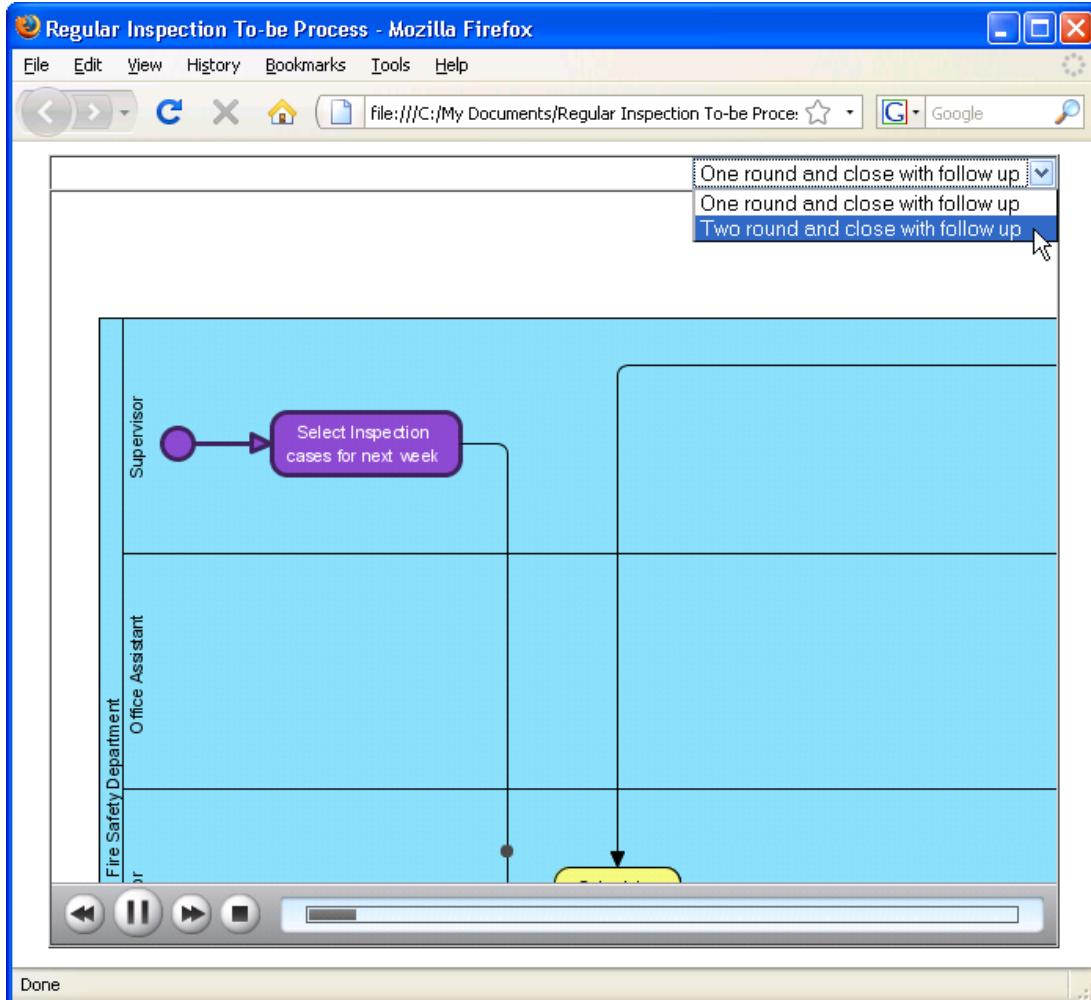


Figure 12-50 To play another path

Navigation in exported movie

You can control the flow of movie by clicking on the buttons at the bottom of the movie or by pressing shortcut keys. Here is a list of buttons.

Button	Description
	Backward. Move one shape backward in the flow.
	Pause. Temporary stop playing the movie. Can play again by pressing the button.
	Play. Play or continue to play the movie content.
	Forward. Advance to the next shape in the flow.
	Stop. Terminate the movie.

Table 12-11 Buttons in exporting Flash movie

You can also click on the slider to move the animation to a desired position.

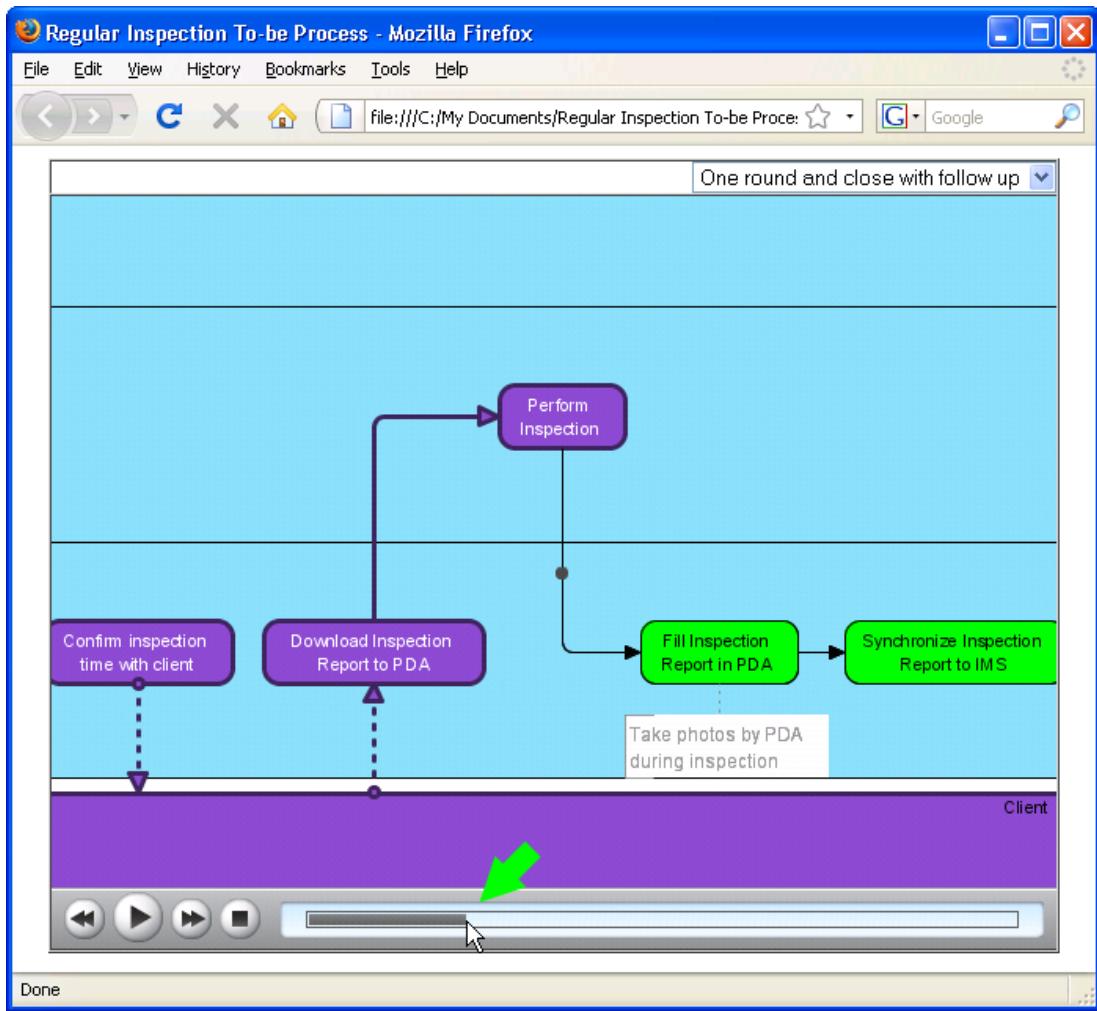


Figure 12-51 Navigate in the Flash movie by dragging slider

RTF documentation

Opening the documentation of model element

To read or edit the documentation of model element, right click on the diagram element and select **Open Specification...** from the popup menu.

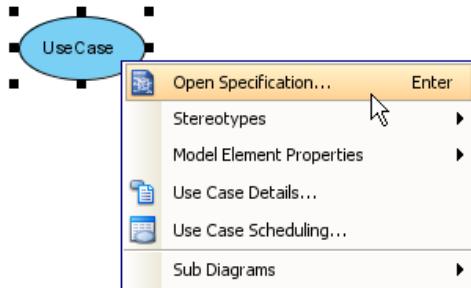


Figure 13-01 To open the specification of use case

or open the documentation pane.

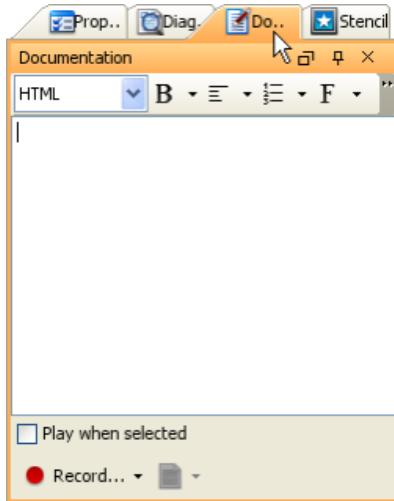


Figure 13-02 To open the documentation pane

Documentation editor

The documentation pane is where you can describe the chosen model element. You also can save the content as template and reuse in other model elements. Furthermore, you can print the content.

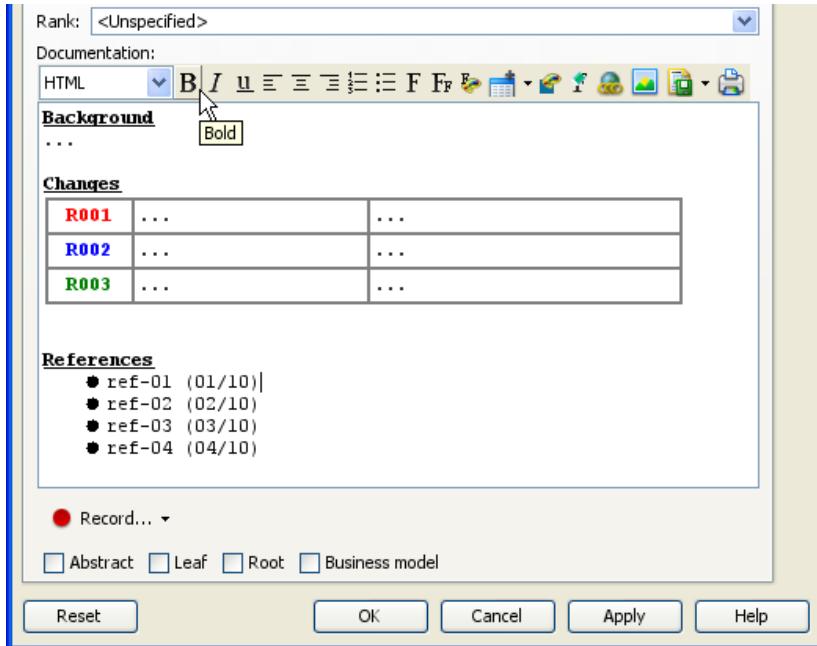


Figure 13-03 The documentation editor

Documentation template

Saving template

Design template in any documentation editor, and select **Saved Template > Save as template...** from the toolbar above the editor. Enter the template name in the dialog box and click **OK** to confirm.

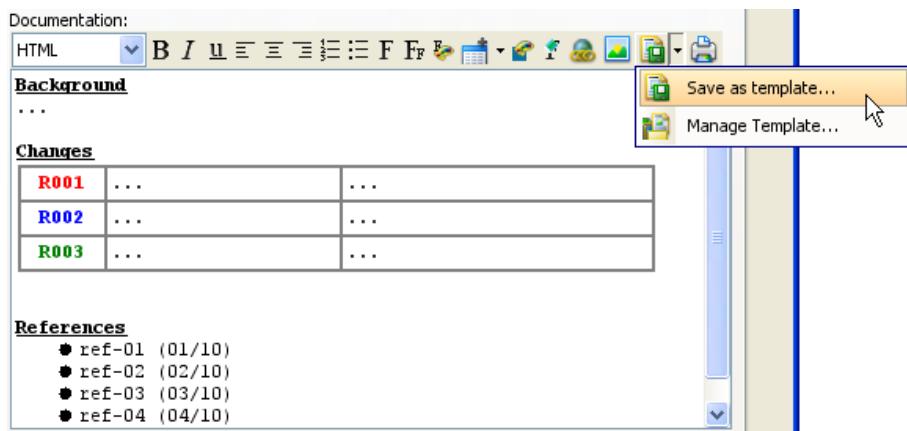


Figure 13-04 To save editor's content as template

Reusing template

In the documentation editor that we want to start editing with template, select **Saved Template > Manage Template...** from the toolbar above the editor, and pickup a template to apply.

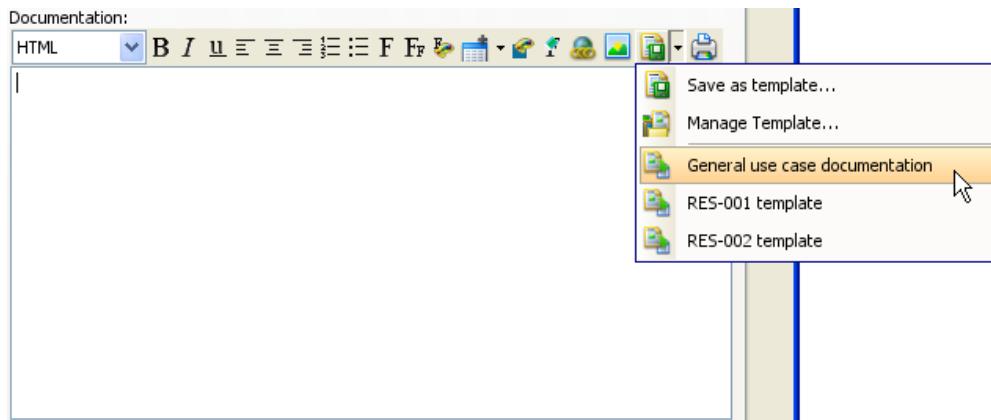


Figure 13-05 To apply a template in editor

Printing documentation

Click on the **Print...** button above the documentation editor, and proceed with setting up the page, and start printing.

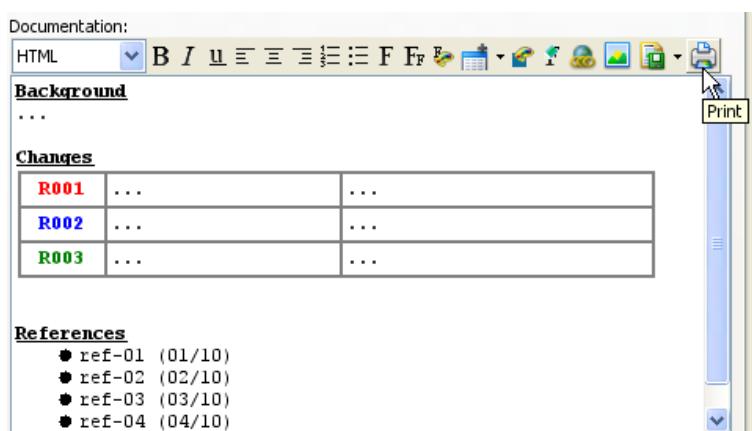


Figure 13-06 To print the documentation

Voice documentation

Opening the documentation of model element

To read or edit the documentation of model element, right click on the diagram element and select Open Specification... from the popup menu.

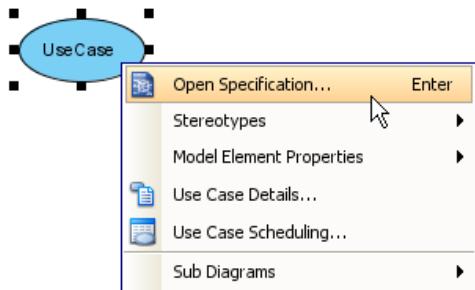


Figure 13-07 To open specification of use case

or open the documentation pane.

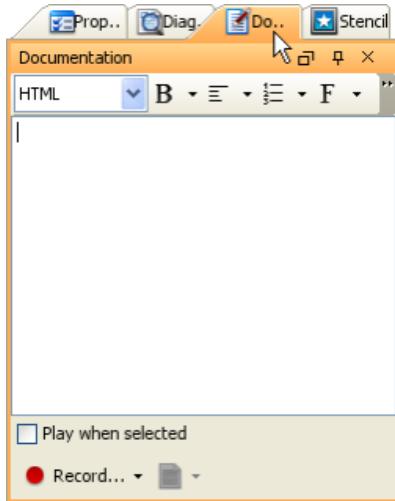


Figure 13-08 To open the documentation pane

Recording voice

1. Open the documentation of model element.
2. Click **Record...** below the documentation pane.

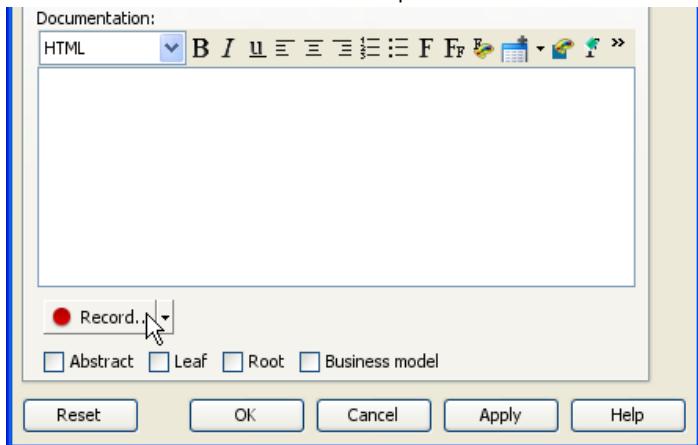


Figure 13-09 To record voice

3. In the **Record Voice** dialog box, click on the **Record** button to start recording.

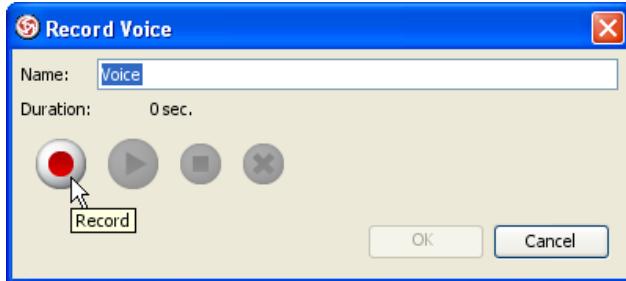


Figure 13-10 To start recording

NOTE: In order to record sound, make sure you have your audio input device is active.

4. Record the voice.
5. Click the **Stop** button to end recording.

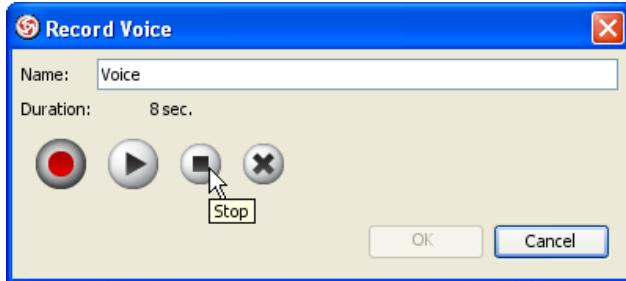


Figure 13-11 To stop recording

NOTE: You can play the voice by pressing the **Play** button, or record again by pressing **Clear**, and rerun the previous steps.

6. Name the voice clip.

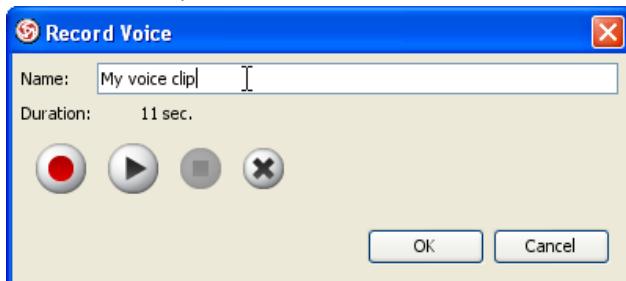


Figure 13-12 To rename voice clip

7. Click **OK** to confirm recording.

Linking voice to documentation

1. Open the documentation of model element.

2. Click **Manage...** below the documentation pane.

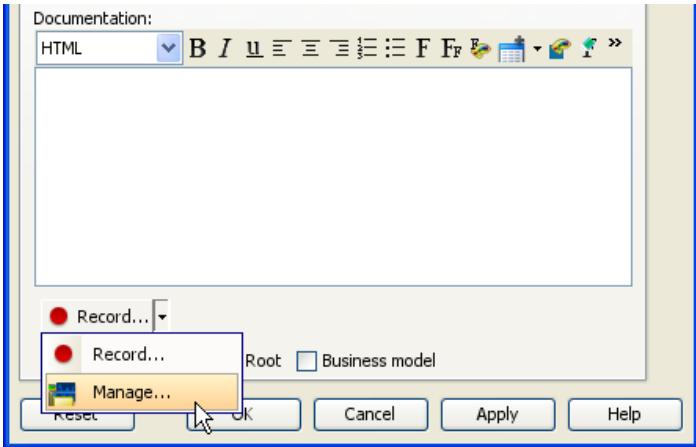


Figure 13-13 To manage voice

3. In the **Manage Voice** dialog box, click the **Add** button, and select either **Link to File**.

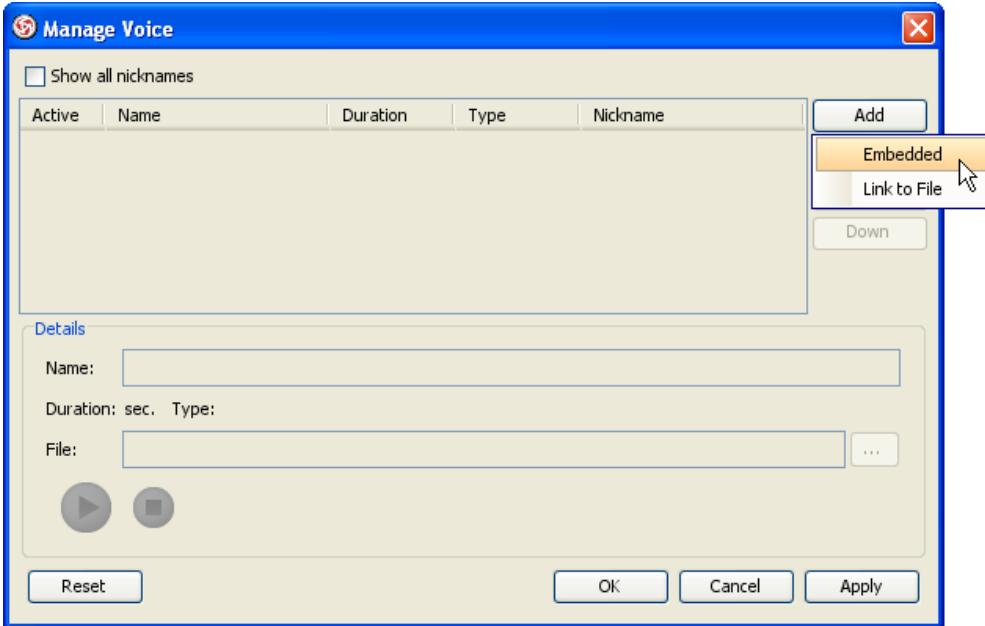


Figure 13-14 To embed a voice clip

4. Select the audio file add.

5. Name the voice clip.

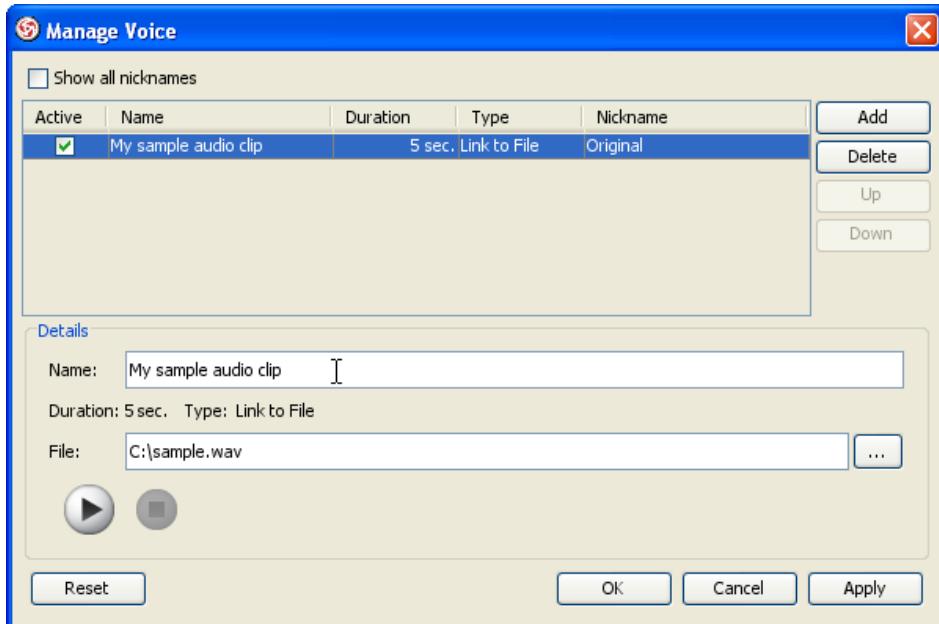


Figure 13-15 To rename a voice clip

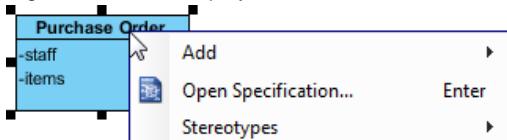
6. Click **OK** to confirm.

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.

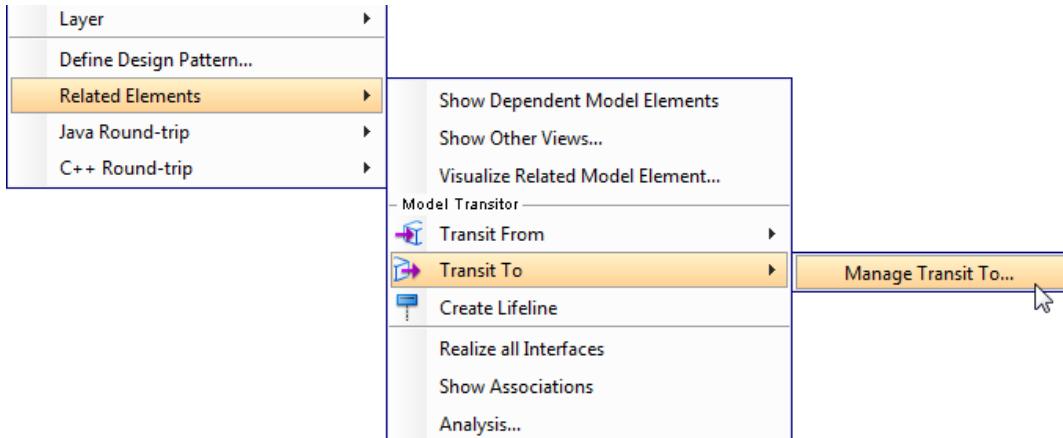
Adding a transition between shapes

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.



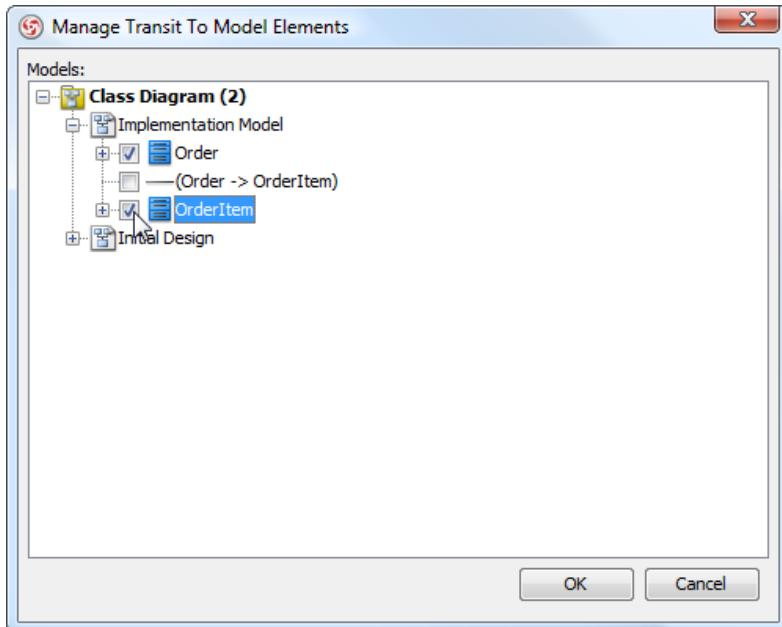
Right clicking on a class

2. If you are right clicking on the source shape, select **Related Elements > Transit To > Manage Transit To...** from the popup menu. If you are right clicking on the target shape, select **Related Elements > Transit From > Manage Transit From...** from the popup menu.



To manage transition

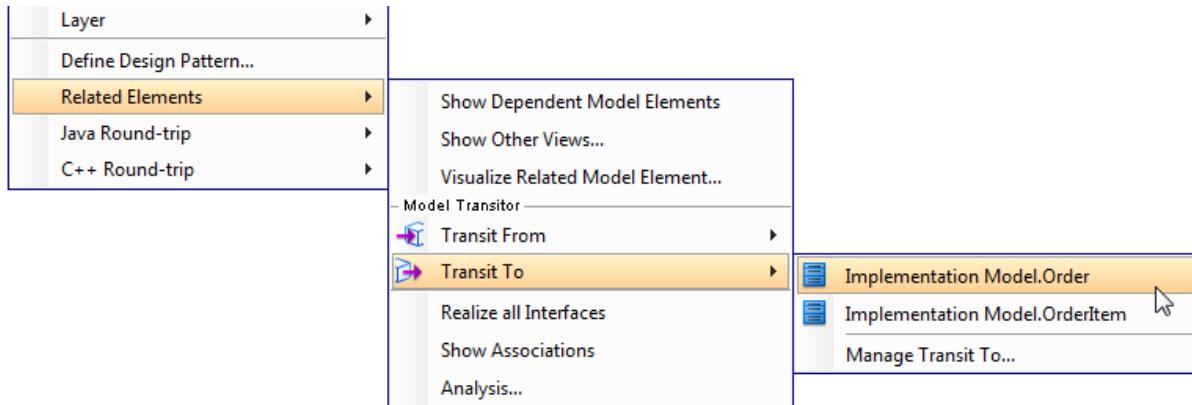
3. In the **Manage Transit To/From Model Elements** dialog box, select the shape(s) you want to transit to/from. You can select multiple shapes to transit to/from. For example, an initial *Purchase Order* class will be transited to an *Order* class and an *OrderItem* class.



Select shapes to transit to

Navigating transited shape

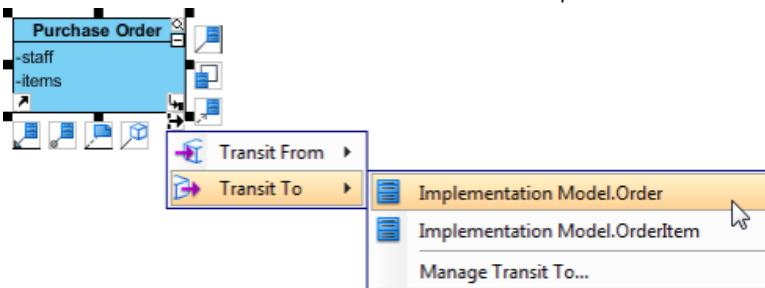
Once a transition is added between two shapes, you can navigate between them. There are two methods to navigate to a transited shape. The first way is to go through the transitor popup menu of a shape. Right click on a shape and select **Related Elements > Transit From/To** from the popup menu, then the shape to navigate to.



Navigate to a transited shape through the popup menu

The second method is to make use of the resource centric interface. Here are the steps:

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.

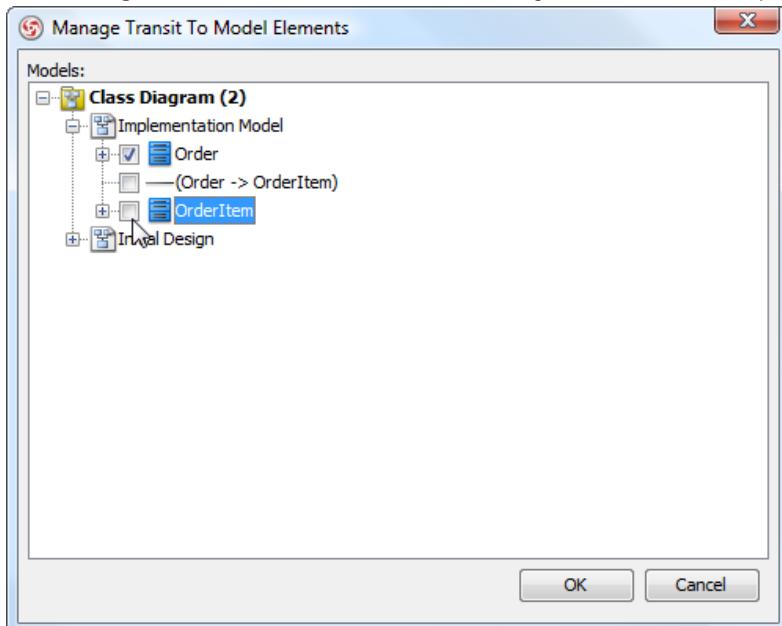


Navigate to a transited shape through the resource centric interface

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.



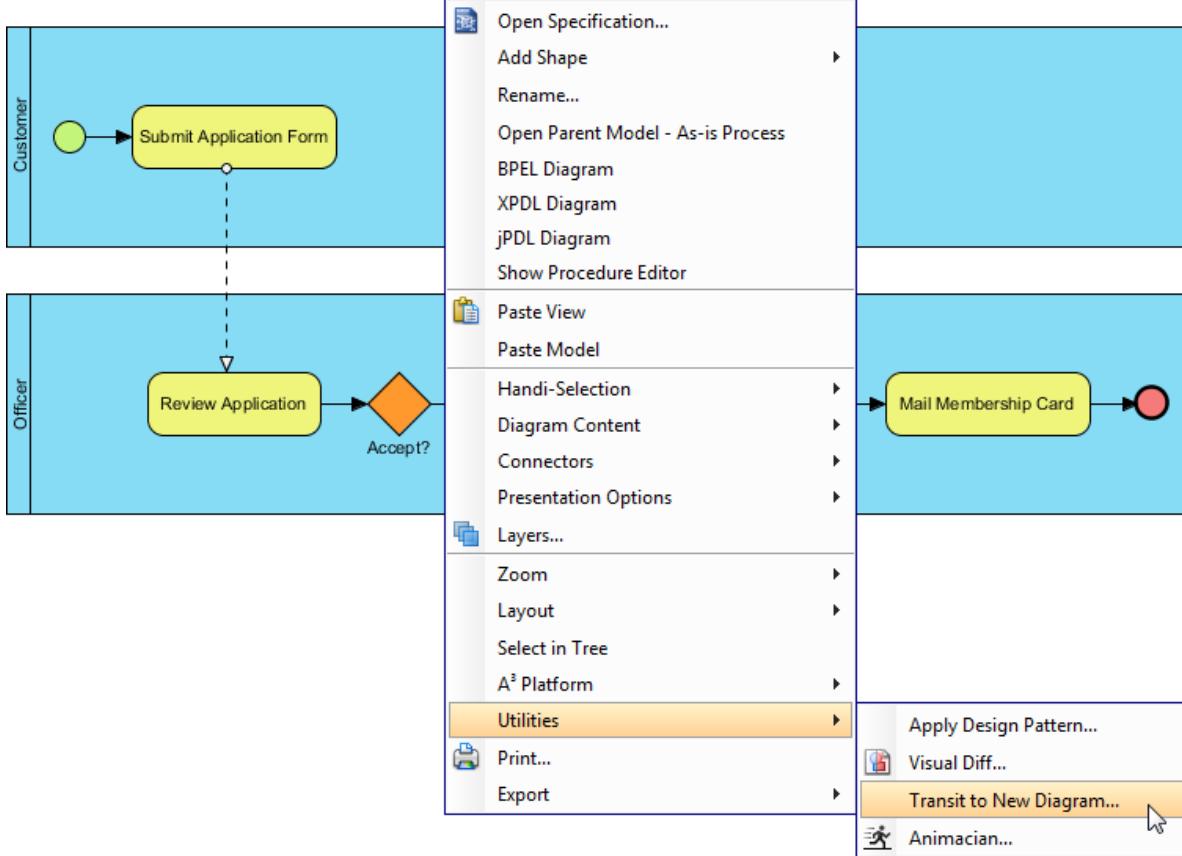
De-select shapes to withdraw from transition

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 phases 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.

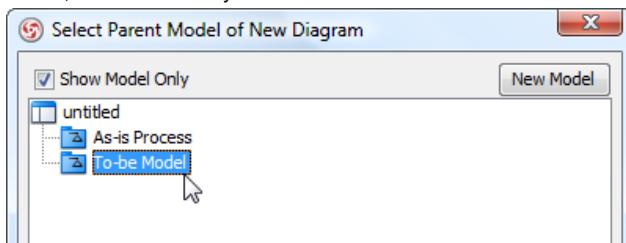
Transiting to a new diagram

1. Right click on the background of diagram that you want to transit from. Select **Utilities > Transit to New Diagram...** from the popup menu.



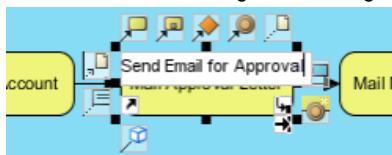
To transit to a new diagram

2. The **Select Parent Model of New Diagram** dialog box appear, 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 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** to continue.



Selecting a model for storing the new diagram

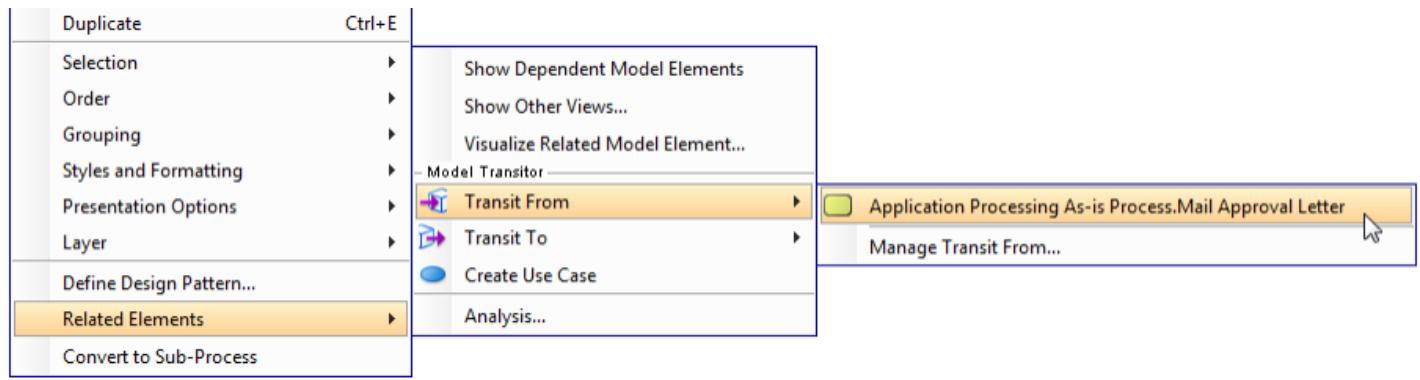
3. You can then start editing the new diagram.



Editing transited diagram

Navigating transited shape

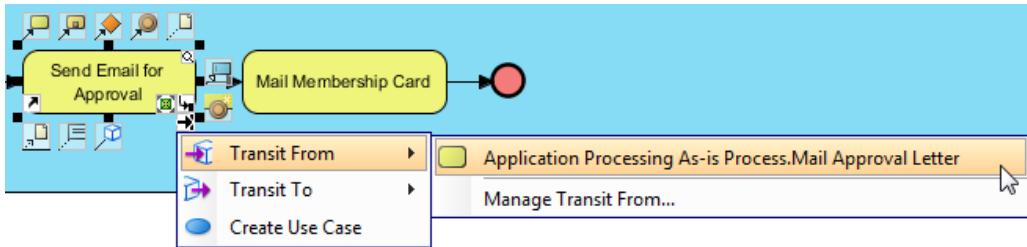
By adding a transition between diagrams, transitions are automatically added between shapes in the two diagrams. With the transition between shapes, you can navigate between them. There are two methods to navigate to a transited shape. The first way is to go through the transitor popup menu of a shape. Right click on a shape and select **Related Elements > Transit From/To** from the popup menu, then the shape to navigate to.



Navigate to a transited shape through the popup menu

The second method is to make use of the resource centric interface. Here are the steps:

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

Drawing use case diagrams

Let's see how you can draw use case diagram in Visual Paradigm for UML through an example.

It is recommended to group related diagrams and model elements in a Model, this could improve the performance when save and load the project.

Switching to model explorer

To switch to Model Explorer, select menu **View > Panes > Model Explorer**.

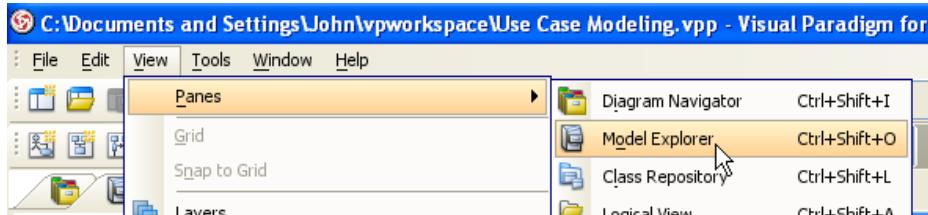


Figure 1-1 Switch to Model Explorer

Creating "Use case model"

Right-click on empty space of Model Explorer and select **Model > New Model...** from the popup menu.

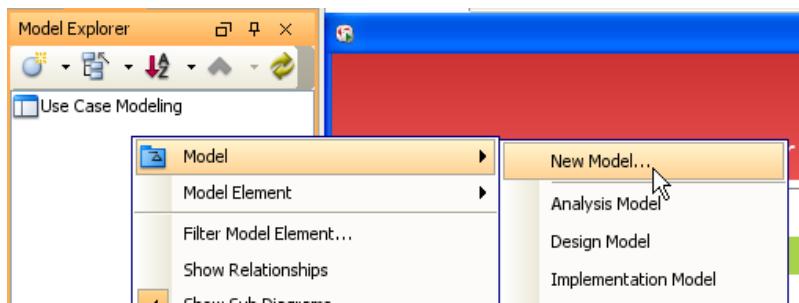


Figure 1-2 Create new Model

When the **Model Specification** dialog box appears, rename the Model to *Use Case Model* and click **OK**.

Creating use case diagram

Right-click on the Model and select **Diagram > UML Diagrams > Use Case Diagram**.

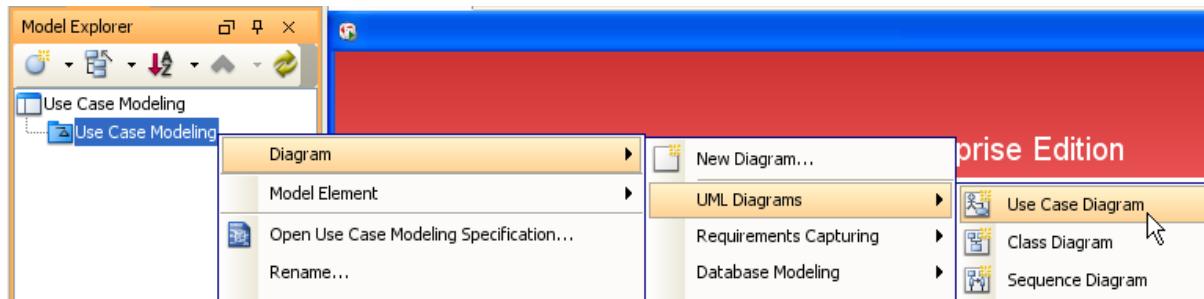


Figure 1-3 Create use case diagram

Rename the diagram to *Regular Inspection Use Case Diagram*.

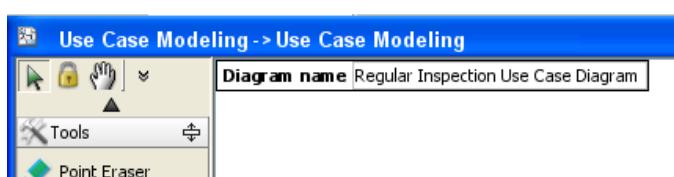


Figure 1-4 Rename diagram

Drawing system

To draw a System, click **System** on the diagram toolbar and then click on the diagram.



Figure 1-5 Create System

Name the System *Inspection Management System (IMS)*.

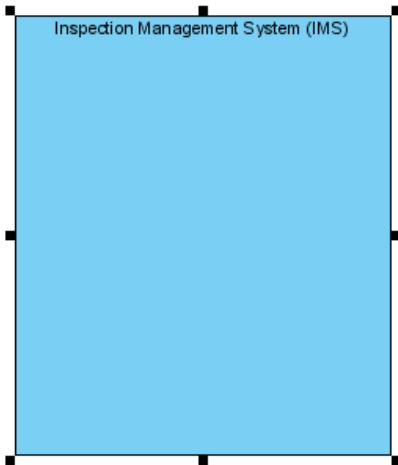


Figure 1-6 Rename System

Drawing actors

To draw a Actor, click **Actor** on the diagram toolbar and then click on the diagram. Note that an Actor should be placed outside the System.

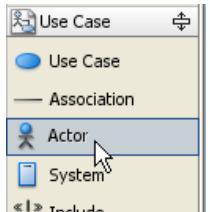


Figure 1-7 Create Actor

Name the Actor *Inspector*.

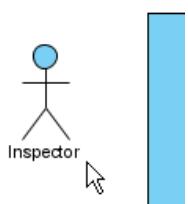


Figure 1-8 Inspector Actor

Create three more Actors namely **Inspector Assistant**, **Office Assistant** and **Supervisor**.

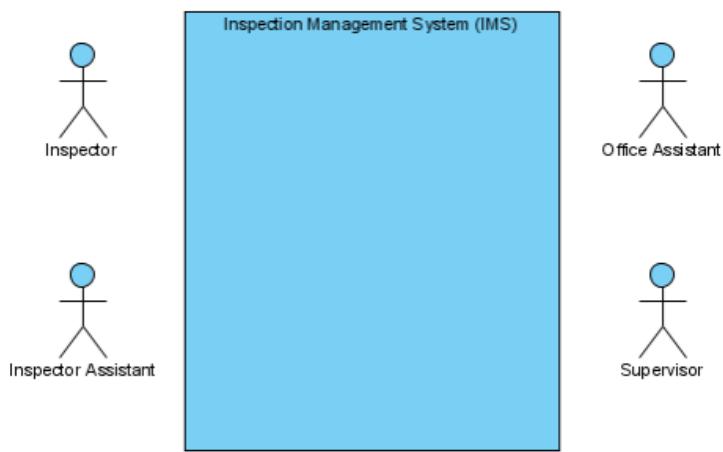


Figure 1-9 Actors Inspector Assistant, Office Assistant and Supervisor

Drawing use cases

Mouse over the Actor **Inspector**, click on the resource **Association -> Use Case** and drag.

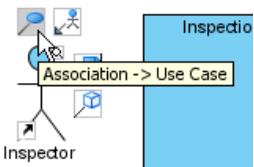


Figure 1-10 Create Use Case with resource

Move the mouse over the System and then release the mouse button. A Use Case together with an Association are created. Name the Use Case **Review and touch up Inspection Report**.



Figure 1-11 Rename Use Case

Follow the same steps to create other Use Cases until the diagram looks like the picture below.

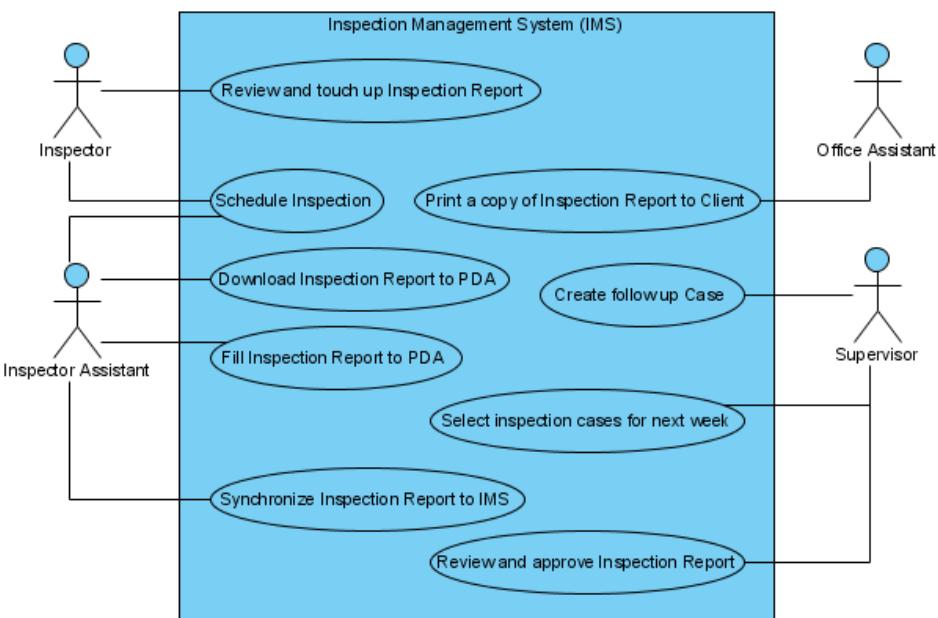


Figure 1-12 Use Cases created

Line wrapping use case name

There is a problem with the diagram that the Use Cases are too wide which make them look strange.

We can solve this by simply resizing the Use Case, and then the name will be line-wrapped automatically.



Figure 1-13 Resize Use Case to wrap caption

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

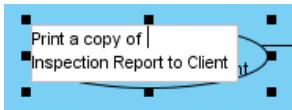


Figure 1-14 Force new line by Alt + Enter

Resize Use Cases until the diagram looks like the picture below.

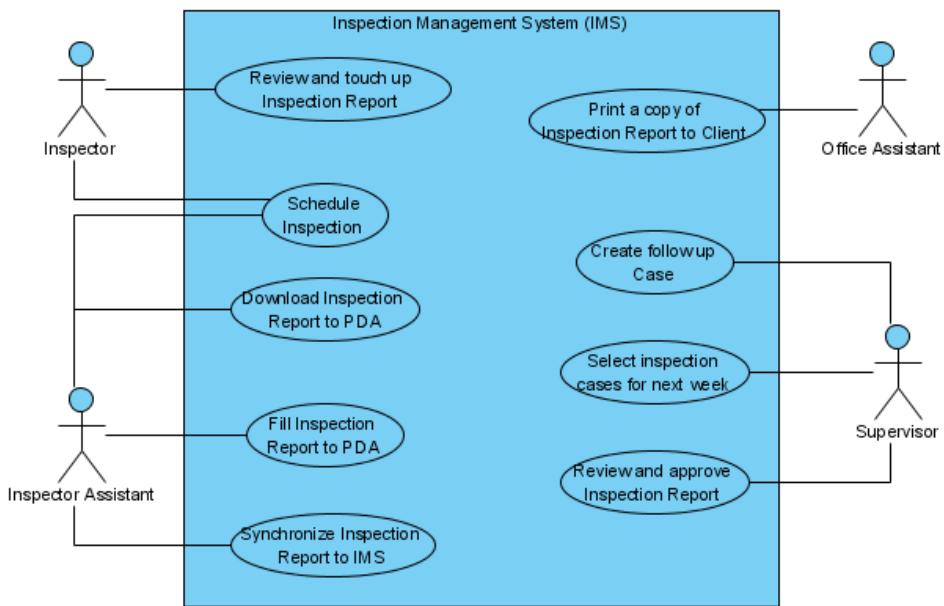


Figure 1-15 Resized Use Cases

Drawing <<Extend>> relationship

To create an Extend relationship, mouse over a Use Case (**Review and approve Inspection Report** in this example), click the resource **Extend -> Use Case** and drag.

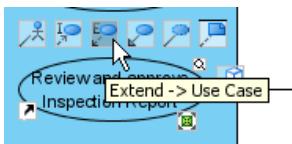


Figure 1-16 Create Extend relationship

Move the mouse to empty space of the System and then release the mouse button. A Use Case together with an Extend relationship is created. Name the Use Case **Re-submit Inspection Report**.

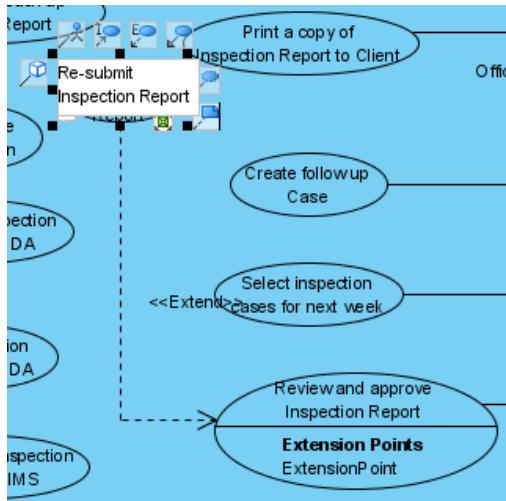


Figure 1-17 Extend relationship created

An extension point is created automatically for the Use Case **Review and approve Inspection Report**. Double-click it and rename it to *Inspection not completed*.



Figure 1-18 Edit extension point

Drawing <<Include>> relationship

To create an Include relationship, mouse over a Use Case (**Review and touch up Inspection Report** in this example), click the resource **Include -> Use Case** and drag.

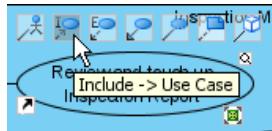


Figure 1-19 Create Include relationship

Move the mouse to empty space of the System and then release the mouse button. A Use Case together with an Include relationship is created. Name the Use Case **Login**.

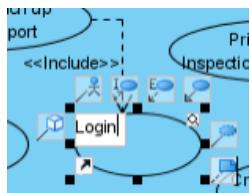


Figure 1-20 Include relationship created

Structuring use case with package

When there are many Use Cases, it would be nice to organize them with Package.

Click **Package** on the diagram toolbar (located in the **Common** category).

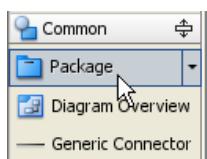


Figure 1-21 Create a Package

Click and drag the mouse to create a Package that surrounds the Use Cases related to the Actor **Supervisor** as shown in the picture below.

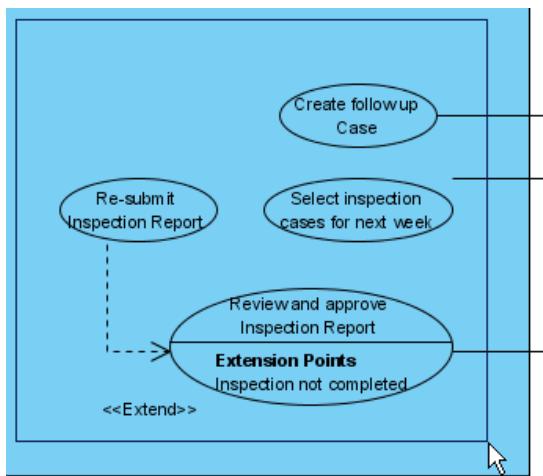


Figure 1-22 Surround Use Cases with Package

Name the Package *supervisor*.

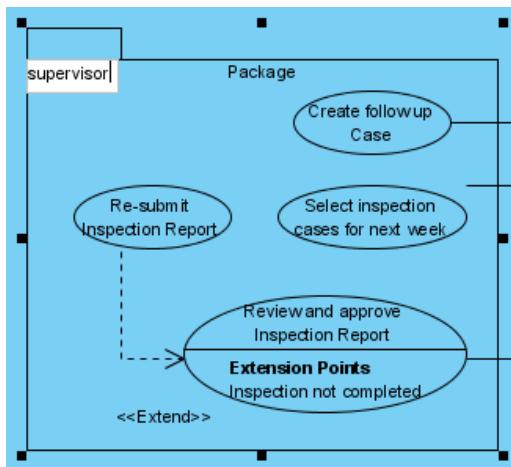


Figure 1-23 Rename Package

Assigning IDs to actors/Use cases

It is possible to assign IDs to actors and use cases. 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 **Options** dialog box by selecting **Tools > Options** from the main menu. Select **Diagramming** from the list on the left hand side, and open the **Use Case Diagram** tab. From there you can adjust the format of IDs.

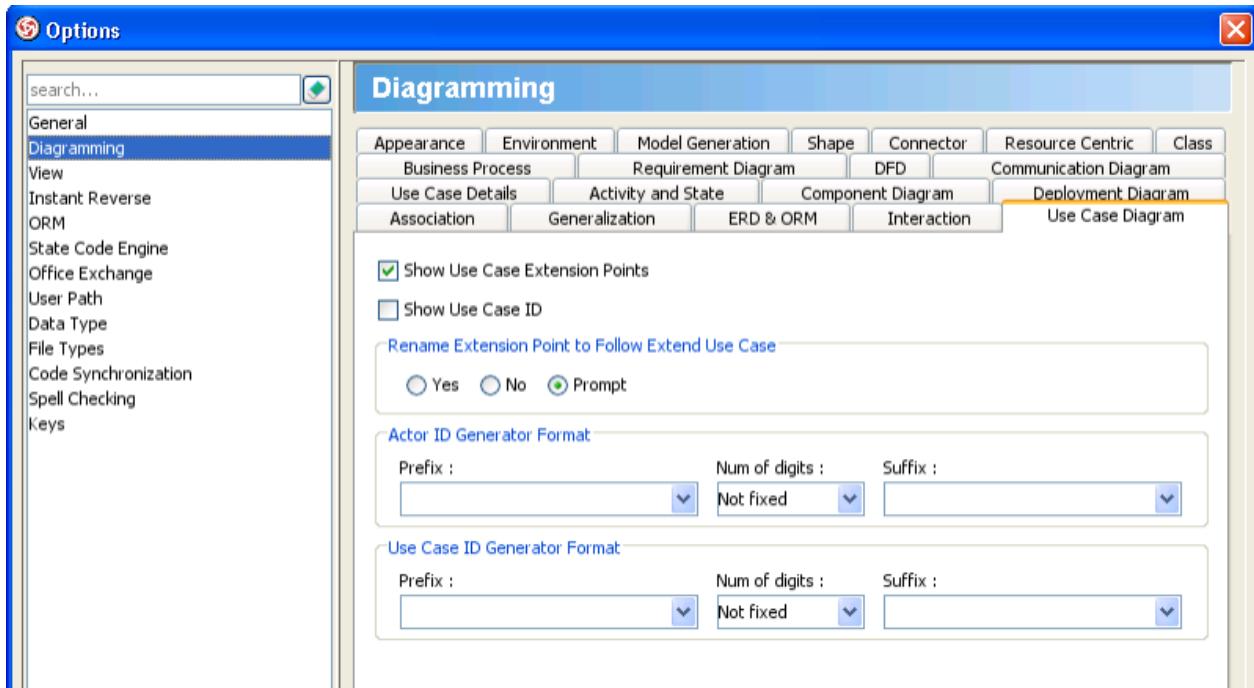


Figure 1-24 Defining format of ID

Below is a description of options.

Option	Description
Prefix	Text to add 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 append to the number

Table 1-1 Options for formatting ID

Showing ID on diagram

By default, ID is just a text property that won't appear on diagram. However, you can make it appear within a use case.



Figure 1-25 A use case with ID displayed

To make it appear, you can either set the global option (refer to the previous section), or set it through diagram's option by right clicking on the business process diagram, selecting **Presentation Options > Show ID of Elements**, and then the type of presenting the ID.

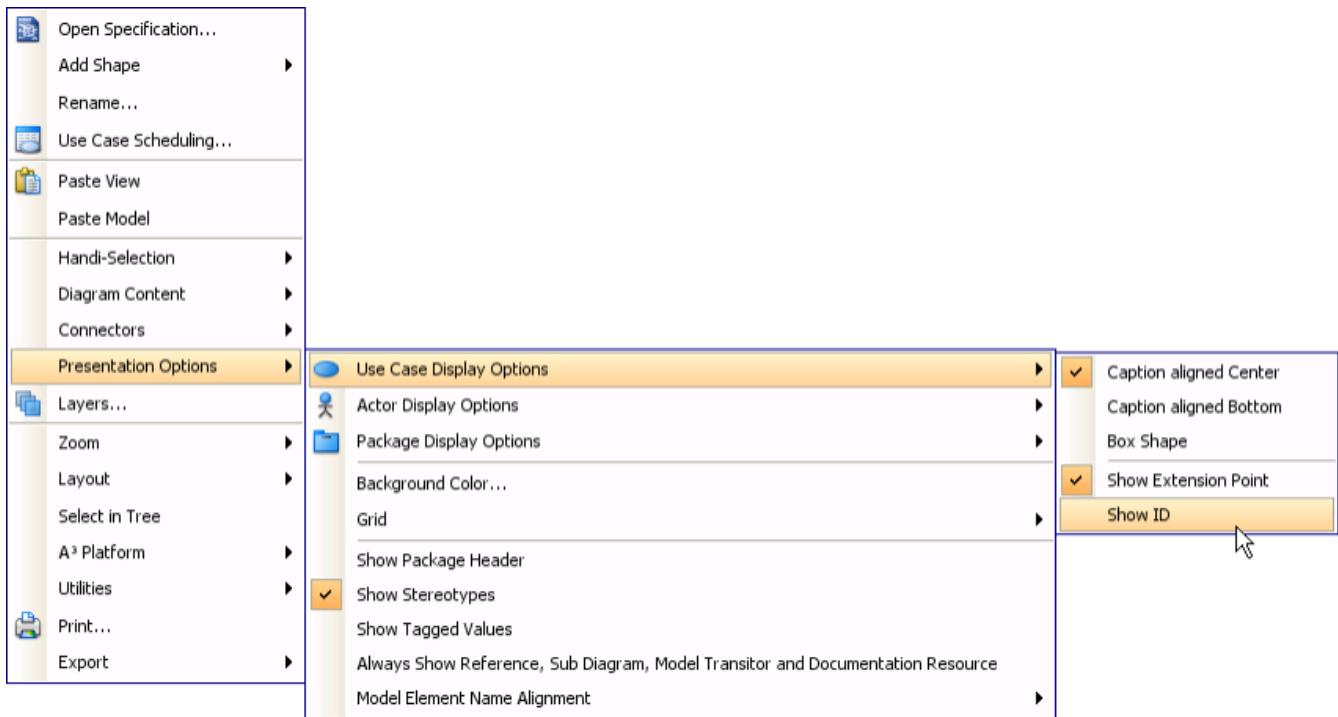


Figure 1-26 To show ID on a diagram

NOTE: Showing of ID is supported only for use case, but not for actor

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 **Property Pane**

Documenting use case details

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 use case.

Opening use case details

To start writing or reading use case details, right click on the target use case and select **Use Case Details...** from the popup menu.

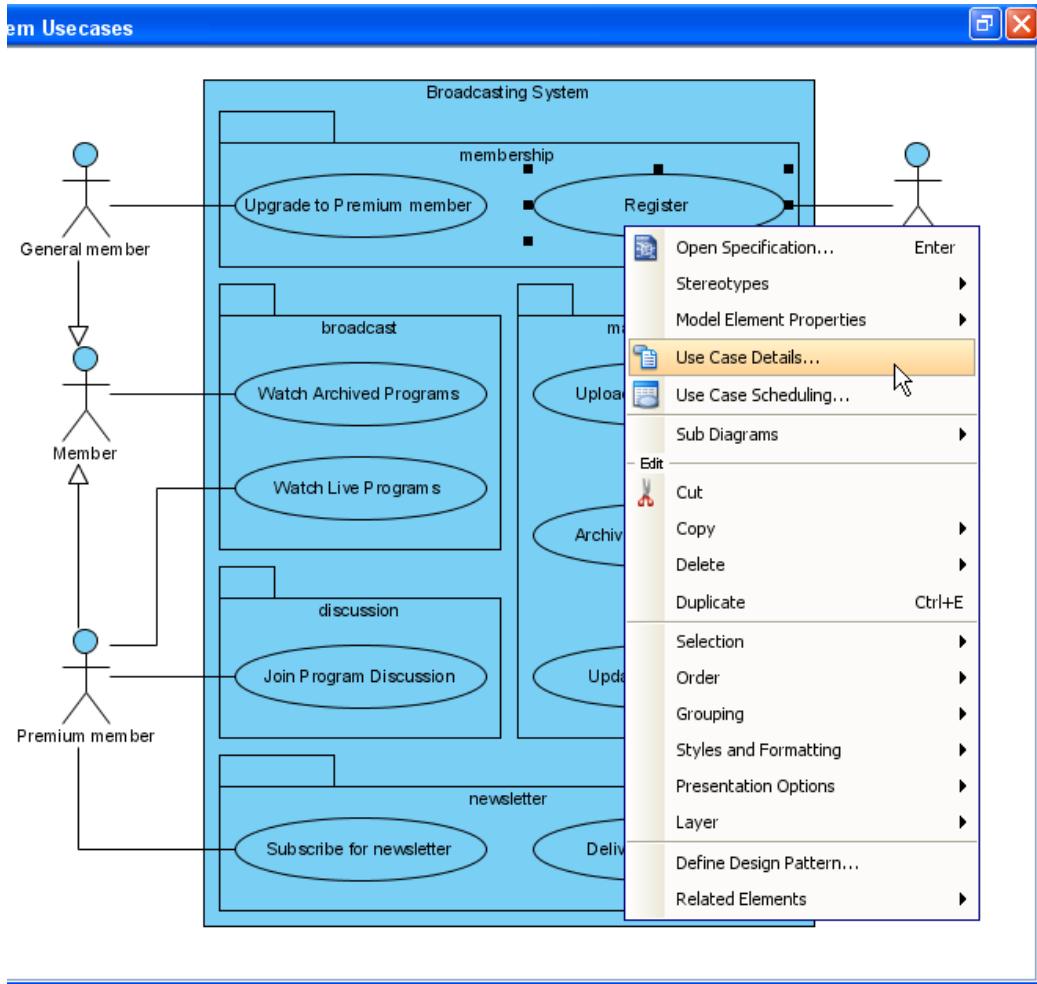


Figure 1-27 To open use case details

Entering basic information

Basic information refers to all general information of a use case.

Rank and justification determine how important the use case is.

Primary actors list the actors being involved in a use case. Actors that are connected to a use case are automatically added as primary actors. Supporting actor are actors who are beneficial from the system, but will have no direct interaction with it. Both primary and supporting actors can be added manually by pressing the plus button and select the actors in the dialog box popped up.

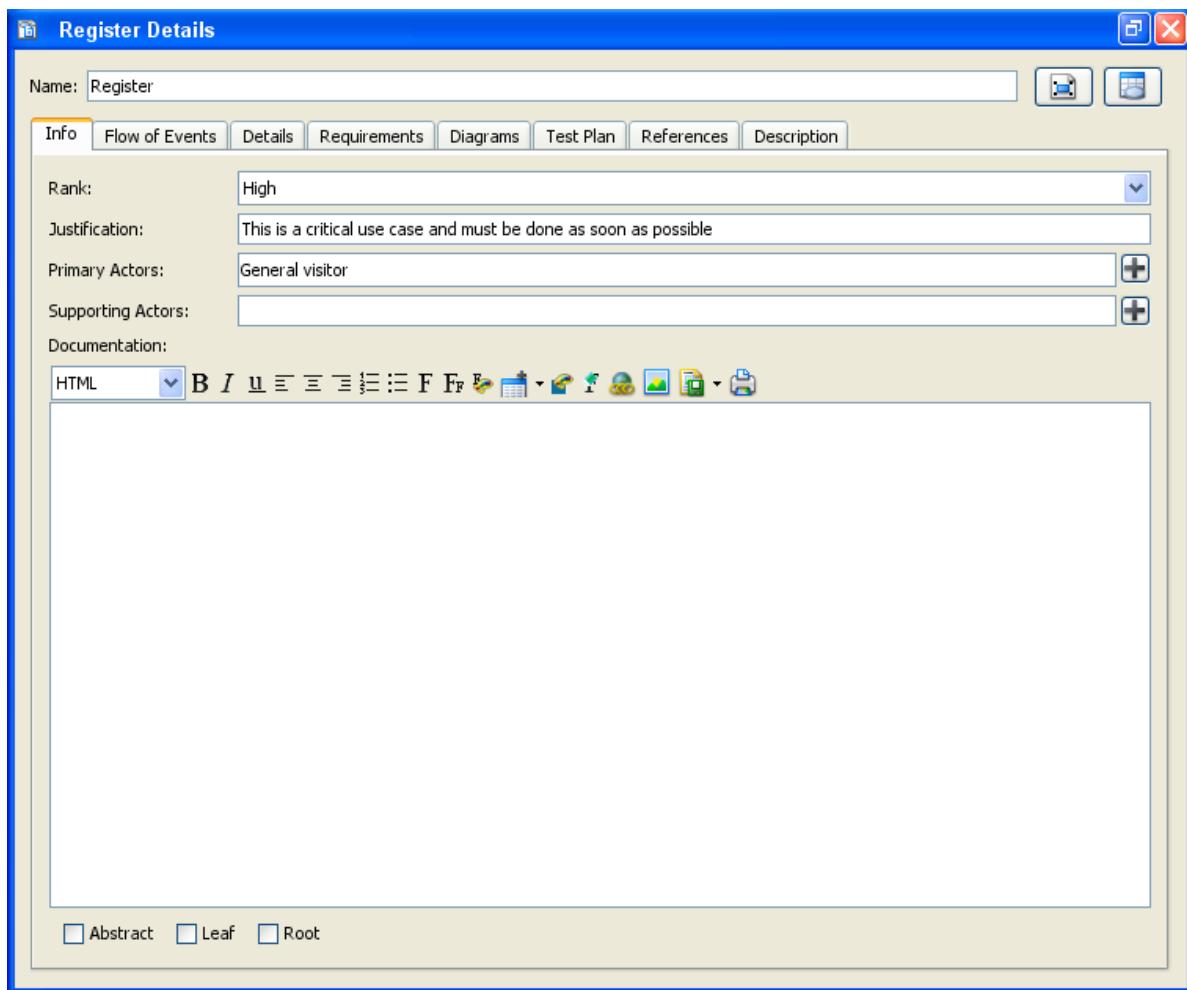


Figure 1-28 Basic information of use case

Entering flow of events

Flow of events refers to the steps required to walk through and fulfilling a use case. You may define multiple flow of events under a use case. There can be extension to certain event, too. For more information about documenting flow of events, read the next chapter **Documenting Flow of Events**.

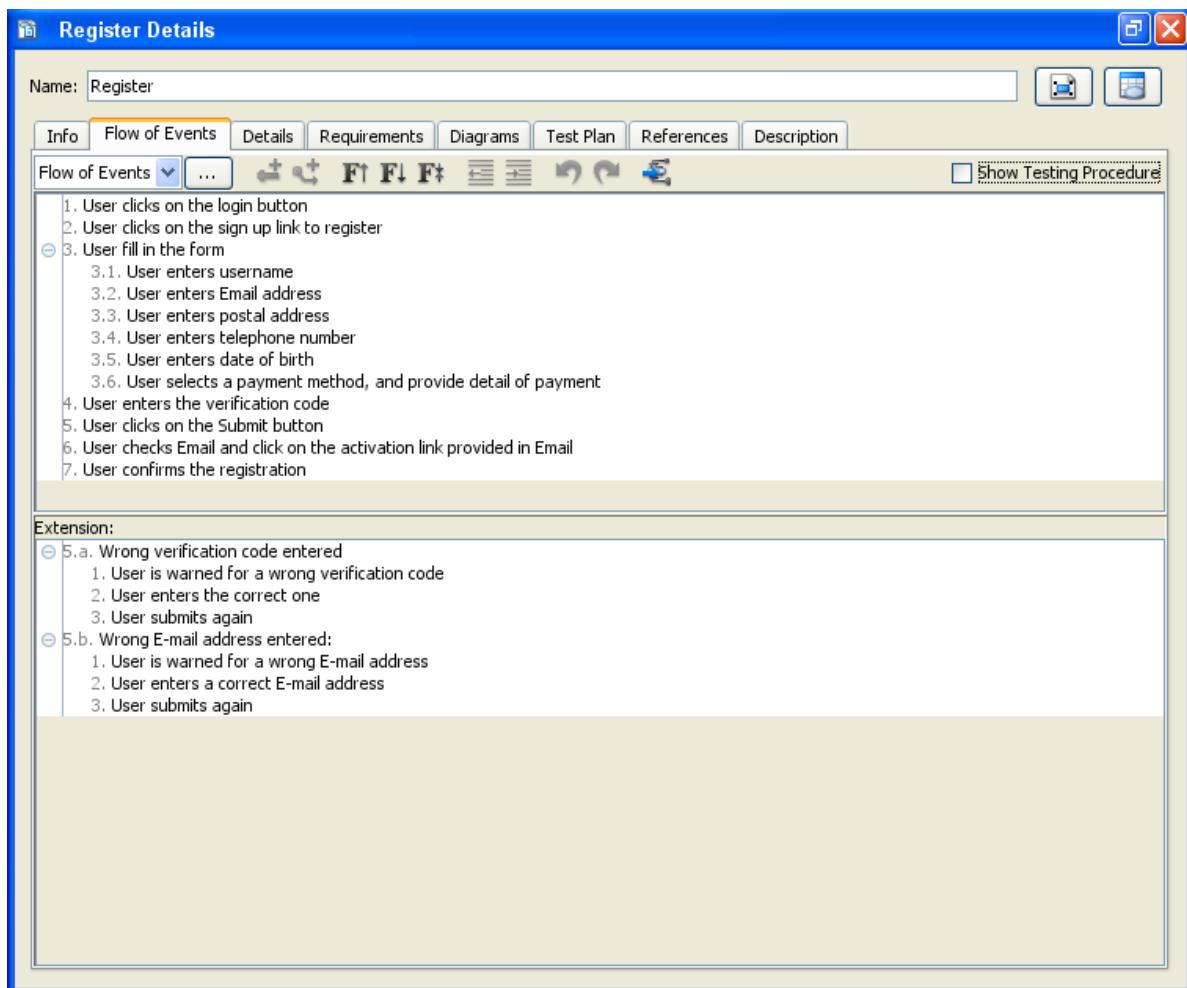


Figure 1-29 Flow of events of use case

Entering details

Details are predefined, detailed fields of a use case, which includes level, complexity, status, implementation status, pre and post conditions, author and assumptions.

The screenshot shows a software window titled "Register Details". At the top, there is a toolbar with icons for saving, closing, and other operations. Below the toolbar, a navigation bar contains tabs: Info, Flow of Events, Details (which is selected), Requirements, Diagrams, Test Plan, References, and Description. A "Name:" field contains the value "Register". To the right of the name field are two small icons: a document and a calendar.

The main area contains several input fields and text areas:

- Level:** Summary
- Complexity:** Medium
- Use Case Status:** Initial
- Implementation Status:** Scheduled
- Preconditions:** User does not have an account
- Post-conditions:** An account is created for user
- Author:** Peter
- Assumptions:** (empty text area)

Figure 1-30 Details of use case

Adding requirements

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

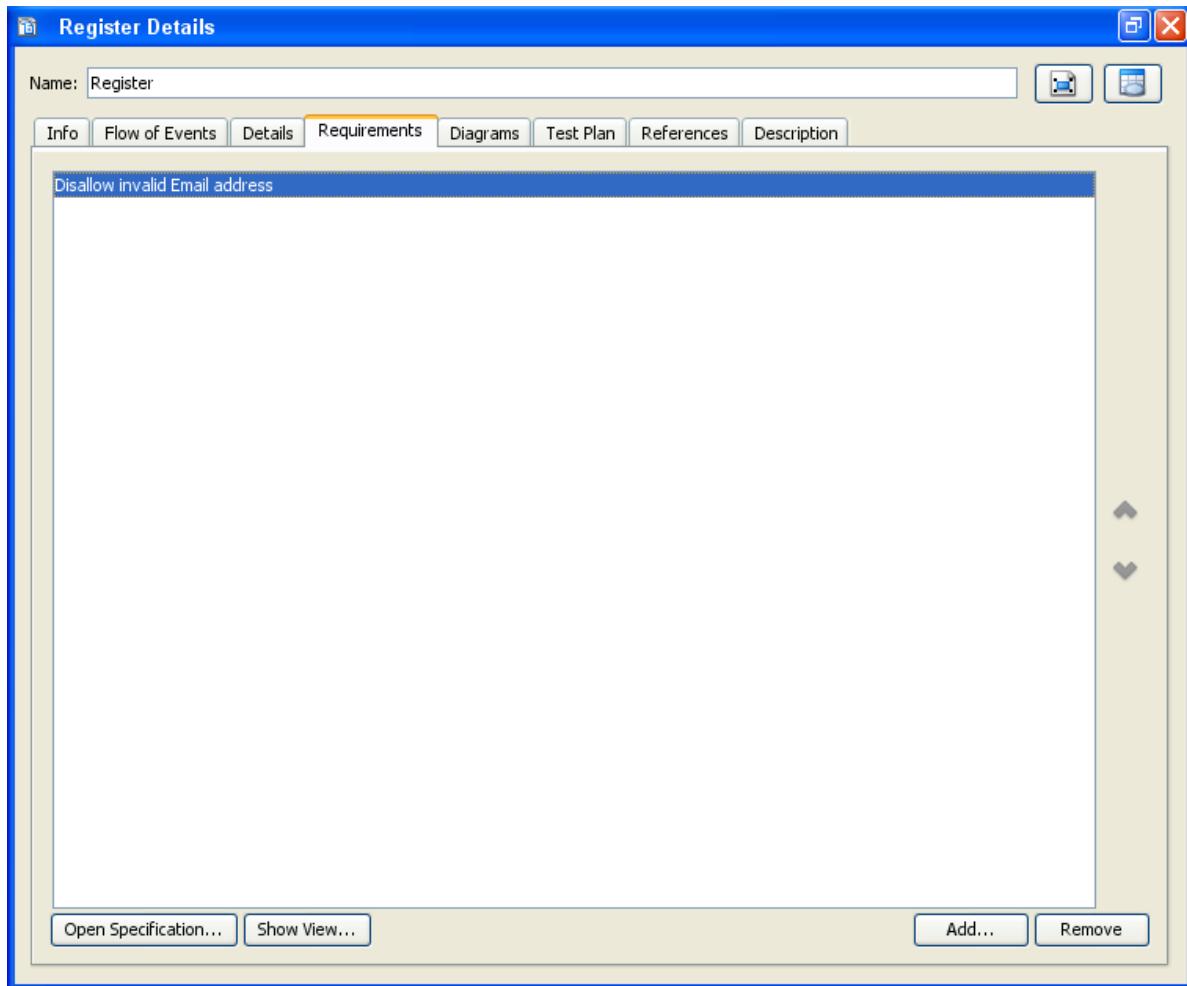


Figure 1-31 Requirements of use case

To add requirement(s) to a use case:

1. click on the **Add...** button at the bottom right of dialog box.

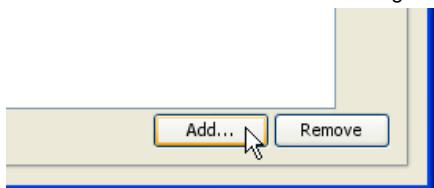


Figure 1-32 To add a requirement

2. In the **Requirements** dialog box, look for and select the requirements to add, and click **OK** to confirm the selection.

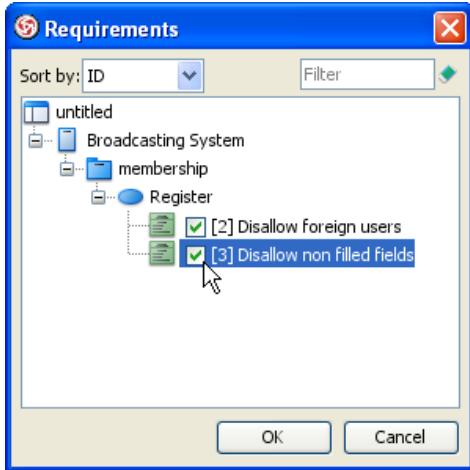


Figure 1-33 Select requirements to add

NOTE: The Requirements page is for adding existing requirements as requirements. If you want to define new requirements, read the next section *Adding Sub-Diagrams*, try to add a requirement diagram as subdiagram and define the requirements in the diagram. Those requirements 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.

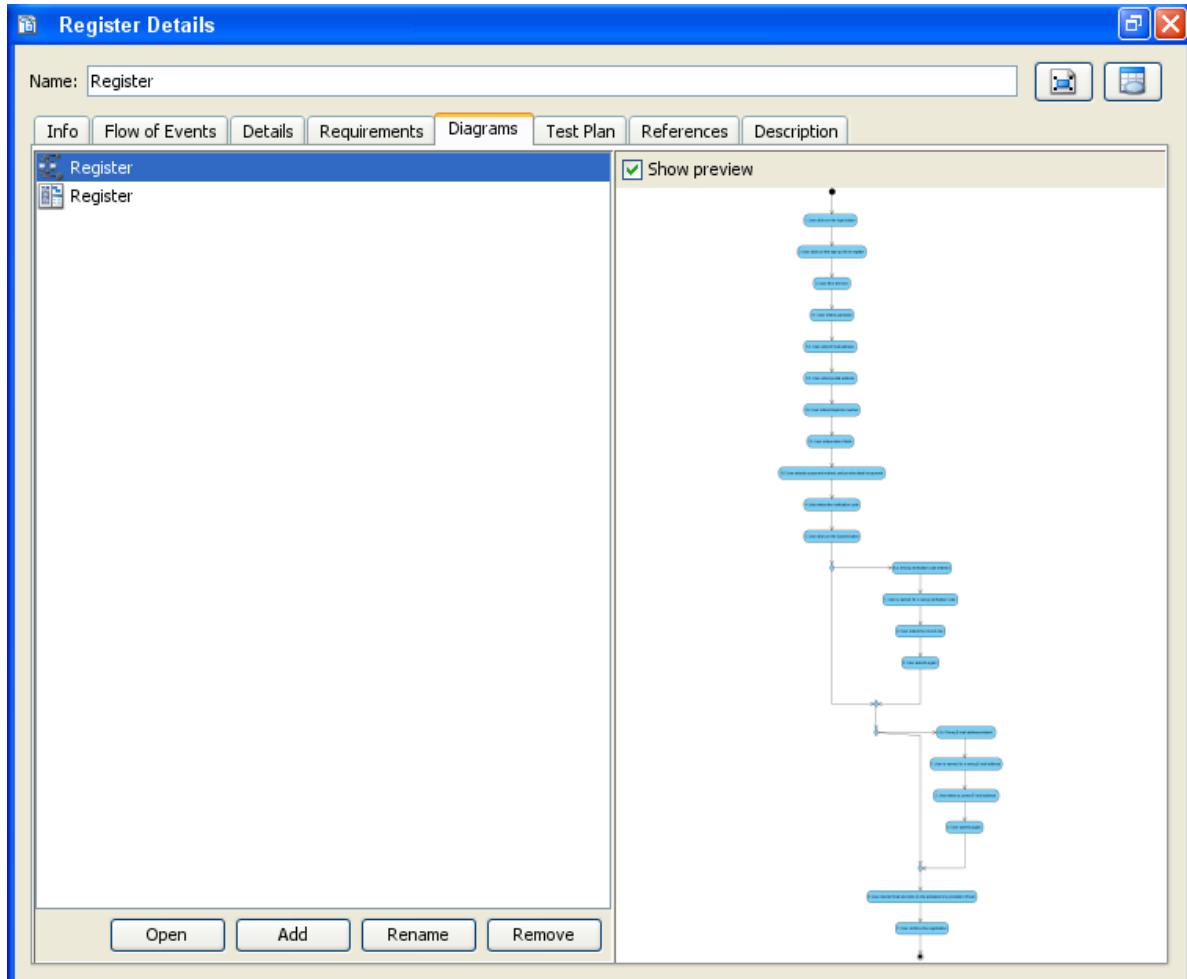


Figure 1-34 Sub-diagrams of use case

Adding a sub-diagram

1. Click Add at the bottom of **Diagrams** page.

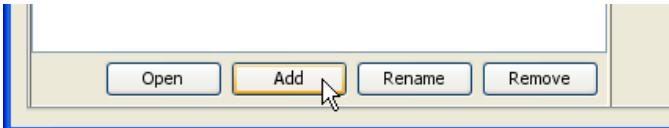


Figure 1-35 To add a sub-diagram

2. To add a new diagram, select the type of diagram to add.

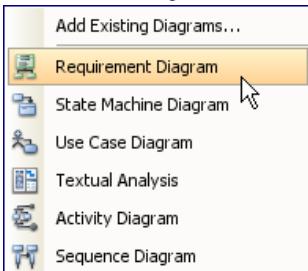


Figure 1-36 To select the type of diagram to add

To add existing diagrams, select **Add Existing Diagrams...** and select the diagrams to add as sub-diagram.

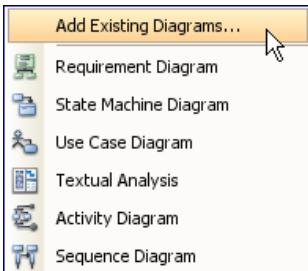


Figure 1-37 To add existing diagrams

Opening a sub-diagram

Select the diagram to open and click on 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.

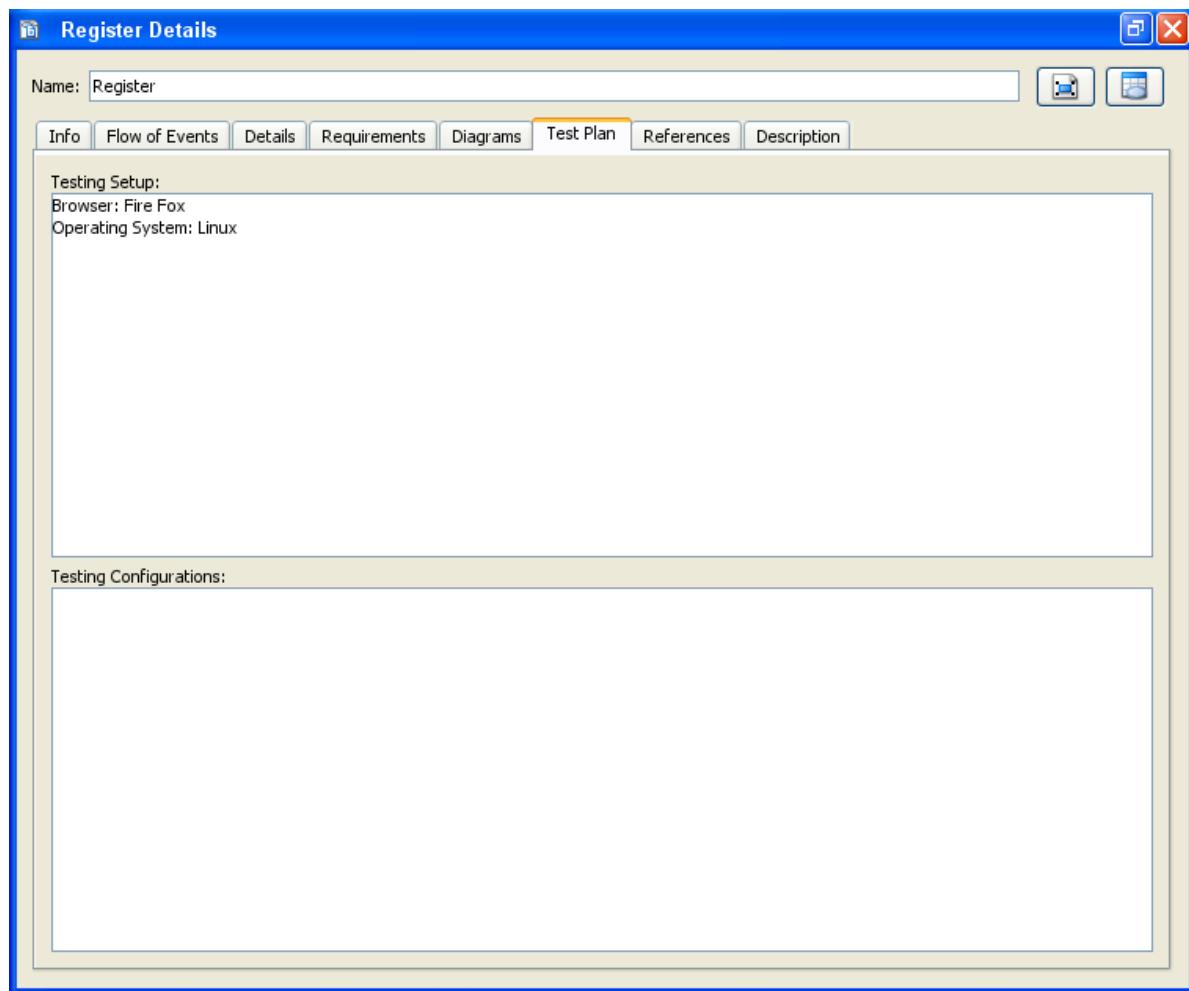


Figure 1-38 Test Plan of use case

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 more dimensions.

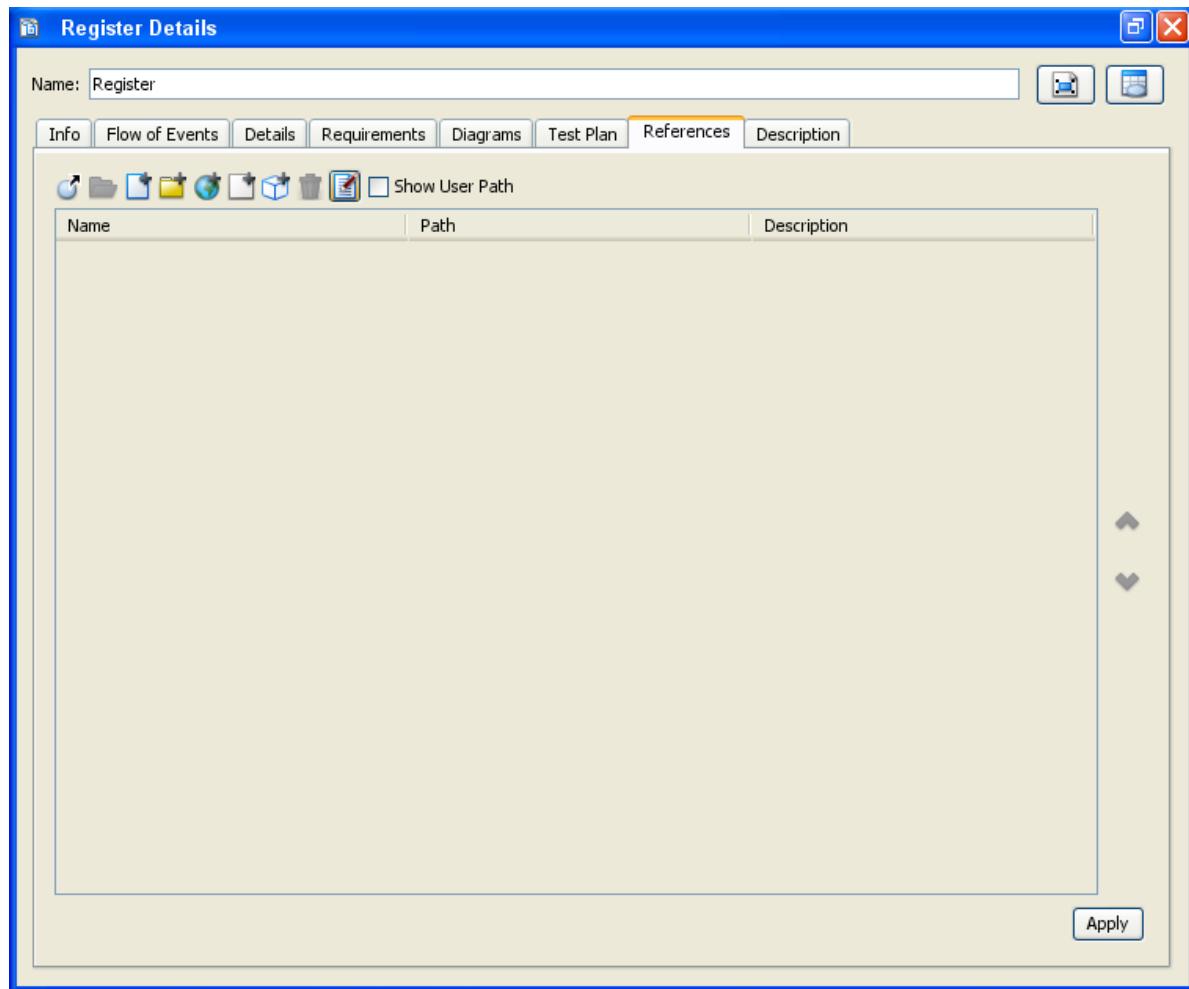


Figure 1-39 References of use case

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 with use case description. But for users who were running the old version, they can still activate the **Description** pane to access the saved data.

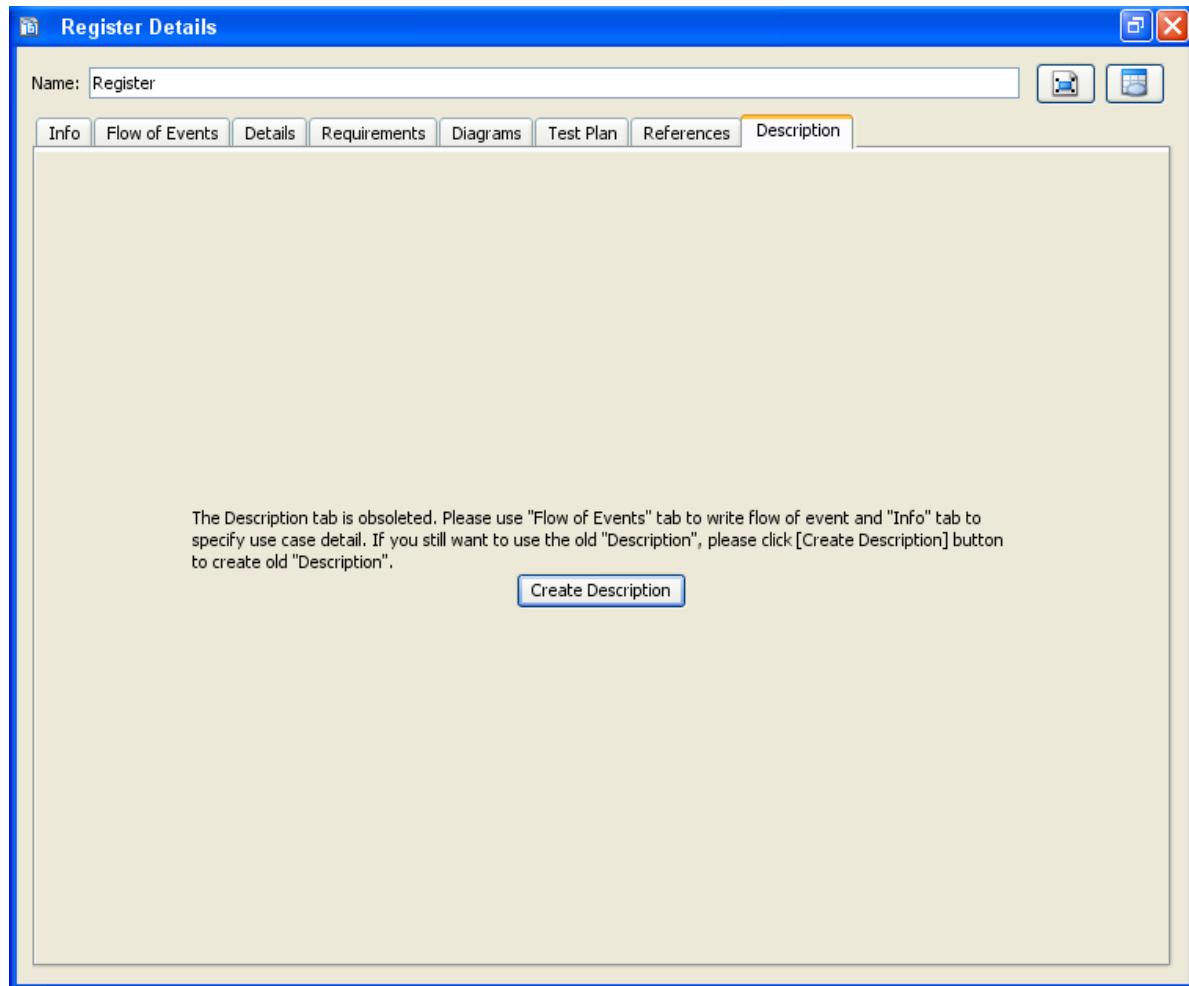


Figure 1-40 Obsolete use case description

Obsolete use case details can be converted to flow of events by clicking on the **Convert** button at bottom right.

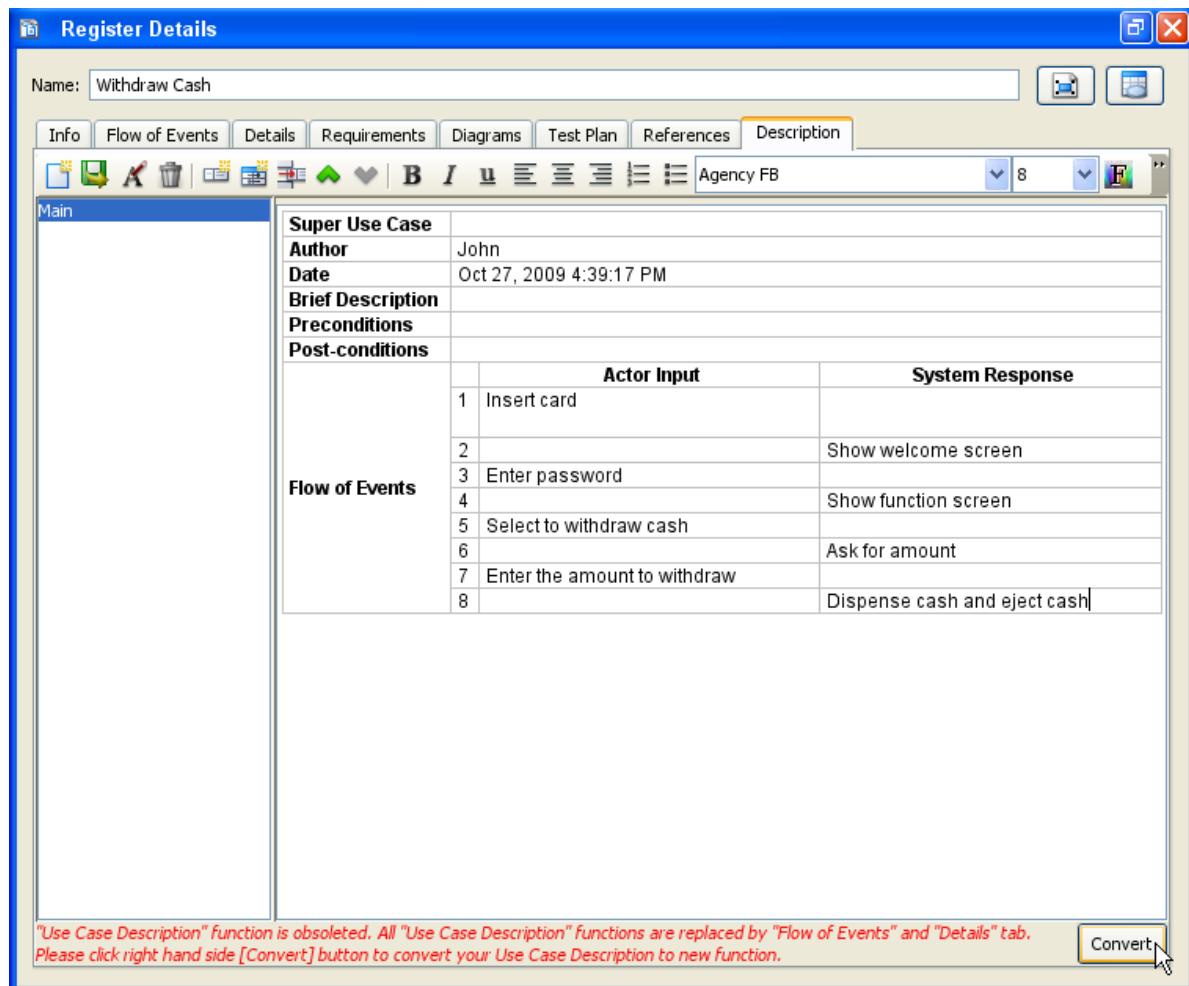


Figure 1-41 To convert use case description

Documenting flow of events

Flow of events refers to the steps required to walk through and fulfilling a use case. You may define multiple flow of events under a use case. There can be extension to certain event, too.

Opening flow of events editor

1. Right click on the target use case and select **Use Case Details...** from the popup menu.

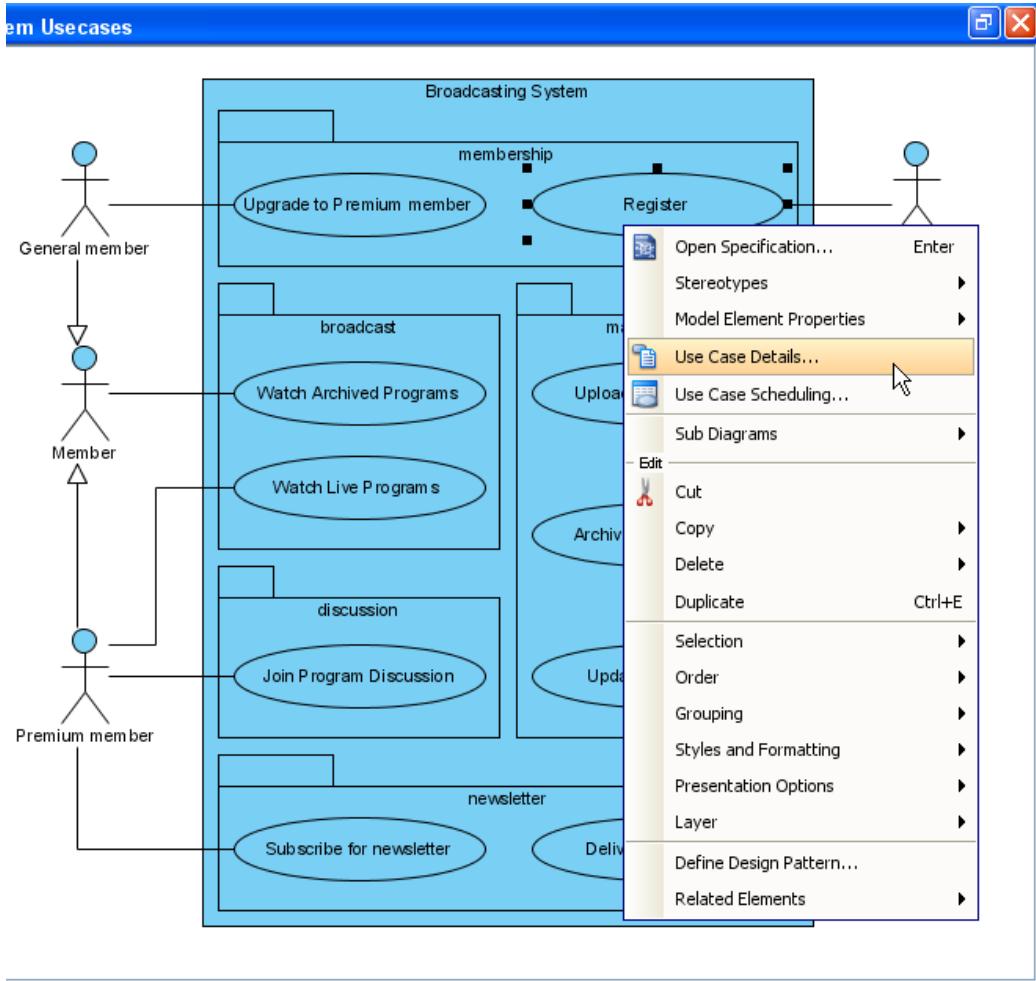


Figure 1-42 To open use case details

2. Click on the **Flow of Events** tab.

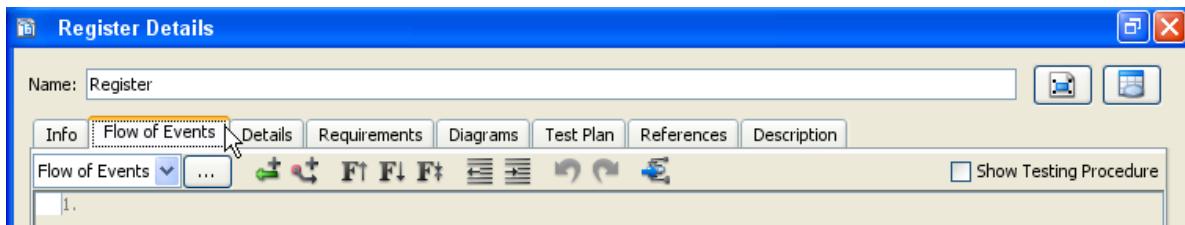


Figure 1-43 To open flow of events editor

The flow of events editor

The upper part of the editor is for editing the flow of events, while the bottom part is for editing the extensions. Furthermore, when **Show Testing Procedure** is checked, two columns will be shown for you to edit the procedures and expected results of testing procedure.

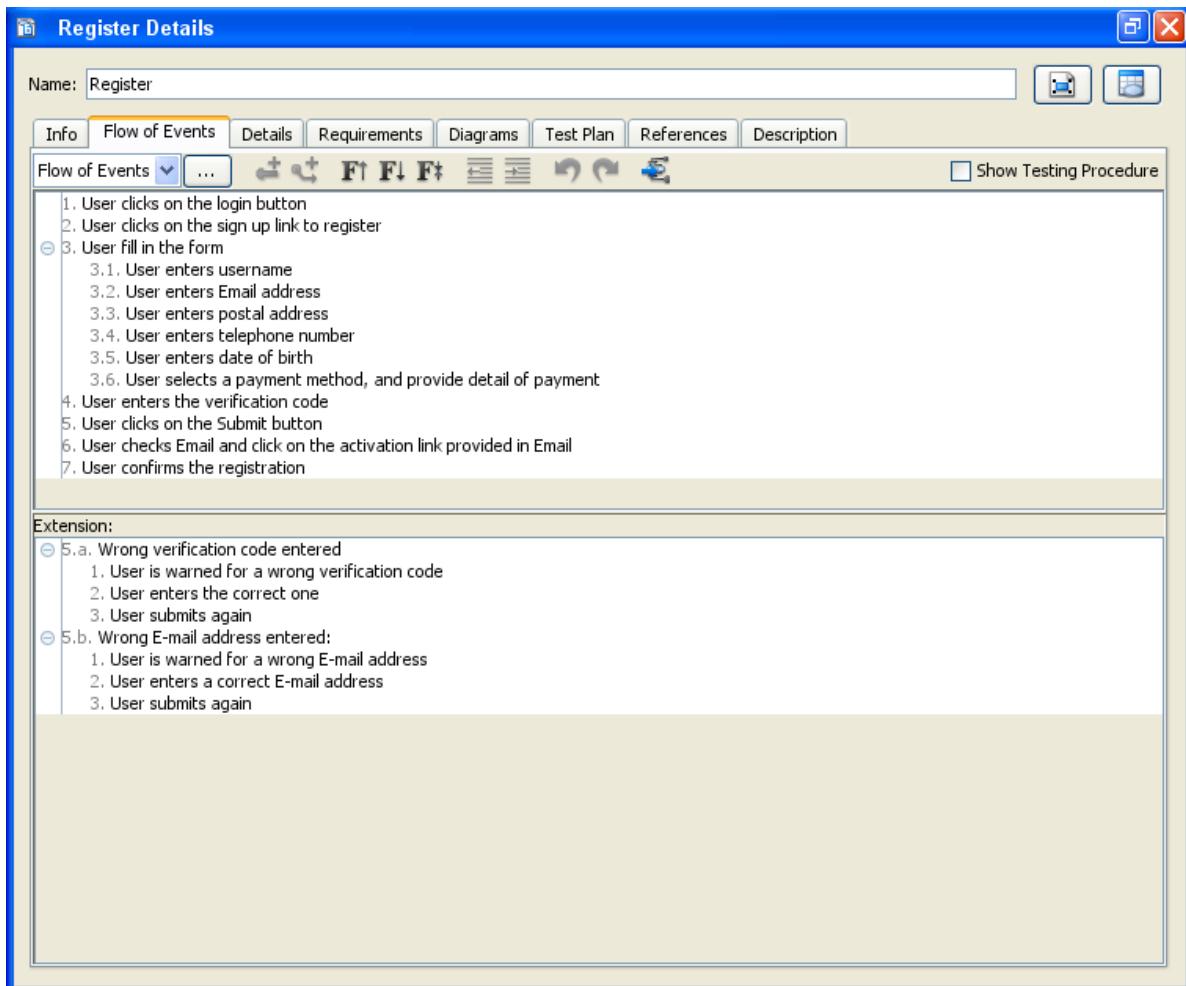


Figure 1-44 The flow of events editor

Control	Description
	To add a new step in flow of events or extensions pane
	To add an extension from a step in flow of events
	To increase the font size of flow of events and extensions content
	To decrease the font size of flow of events and extensions content
	To reset the font size to default
	To decrease the indentation of flow of events by one level
	To increase the indentation of flow of events by one level
	To undo editing
	To redo editing
	To synchronize the flow of events to activity diagram (generate if not exists)
<input type="checkbox"/> Show Testing Procedure	To show the procedures and expected results columns for editing testing procedure

Table 1-2 Controls in flow of events editor

Documenting flow of events

1. Click on the first row to start editing.

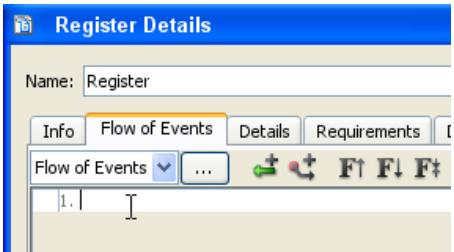


Figure 1-45 To click on the first row to start editing

2. Enter the event.

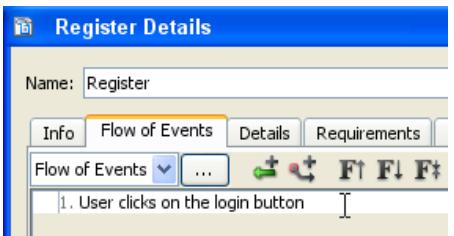


Figure 1-46 To enter the event

3. To enter the next event, press the **Enter** key, or clicking on the button at the toolbar.

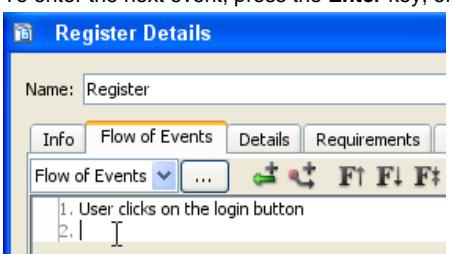


Figure 1-47 Proceed to the next step

4. Enter the second event. Repeat the previous steps until all events are documented.

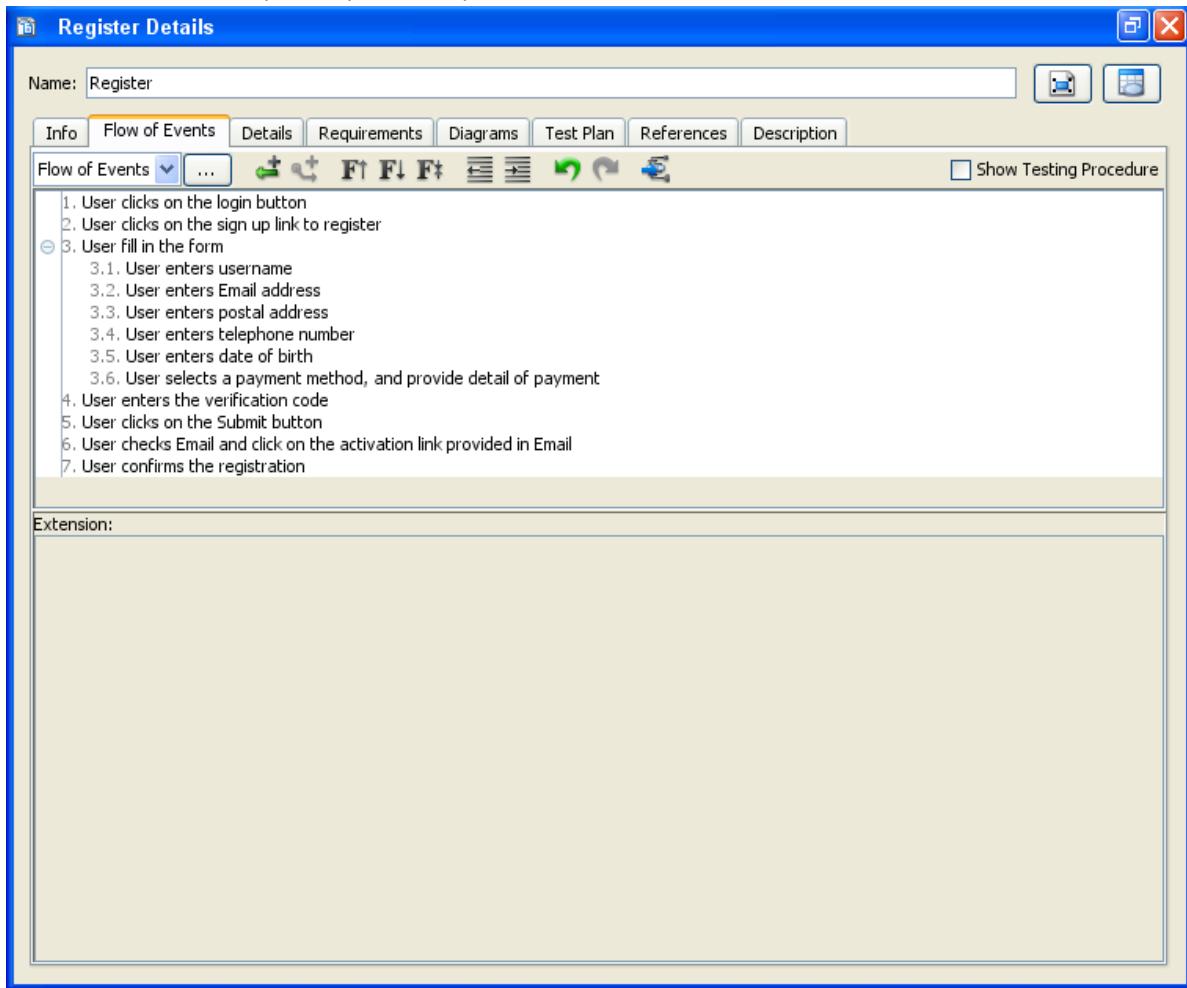


Figure 1-48 Flow of events filled

NOTE: To add a sub event, click on the row of sub event, and either press the **Tab/Ctrl-Period** button or click on the button at the toolbar to indent the event row. On the contrary, press **Shift-Tab/Ctrl-Comma** or click on the button to decrease the indentation.

Adding extension

An extension can be seen as an extension point of use case. It documents the alternative flow that can be extended from the main flow. To add an extension:

1. Select the event for adding extension.

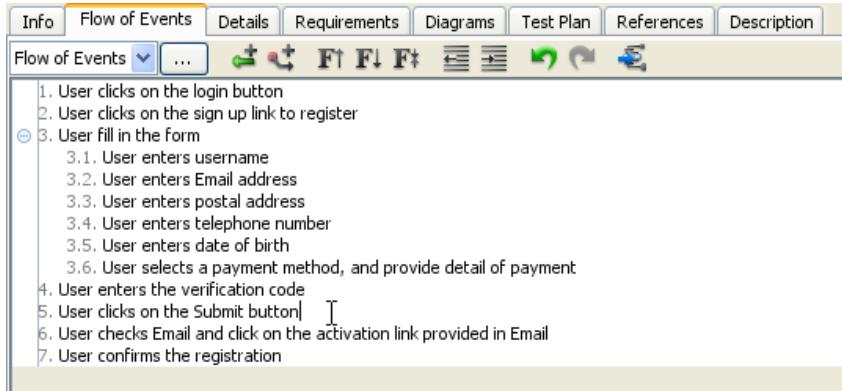


Figure 1-49 Select an event for adding extension

2. Press the **Shift-Enter** key, or clicking on the button at the toolbar.

3. Enter the extension at the bottom pane. You may press the **Enter** key to add a child item.

Name: Register

Info Flow of Events Details Requirements Diagrams Test Plan References Description

Flow of Events ... F↑ F↓ F‡ ⌂ ⌃ ⌁ ⌂ ⌃ ⌁ ⌁ Show Testing Procedure

1. User clicks on the login button
2. User clicks on the sign up link to register
3. User fill in the form
3.1. User enters username
3.2. User enters Email address
3.3. User enters postal address
3.4. User enters telephone number
3.5. User enters date of birth
3.6. User selects a payment method, and provide detail of payment
4. User enters the verification code
5. User clicks on the Submit button
6. User checks Email and click on the activation link provided in Email
7. User confirms the registration

Extension:

- 5.a. Wrong verification code entered
 - 1. User is warned for a wrong verification code
 - 2. User enters the correct one
 - 3. User submits again
- 5.b. Wrong E-mail address entered:
 - 1. User is warned for a wrong E-mail address
 - 2. User enters a correct E-mail address
 - 3. User submits again

Figure 1-50 Extensions filled

NOTE: To add a sub item, click on the row of sub event, and either press the **Tab/Ctrl-Period** button or click on the button at the toolbar to indent the event row. On the contrary, press **Shift-Tab/Ctrl-Comma** or click on the button to decrease the indentation

Documenting testing procedure

Testing procedure can be documented for each of the flow of events item. To document testing procedure:

1. Check **Show Testing Procedure**.

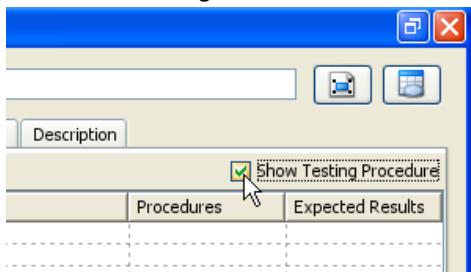


Figure 1-51 To show testing procedure

This shows two more columns - **Procedures** and **Expected Results**.

2. Enter the testing procedure for the event that we need to test.

Flow of Events		...									Show Testing Procedure
Steps			Procedures		Expected Results						
1. User clicks on the login button			1. Visit the main page. 2. Click on the login button at the bottom right of screen								
2. User clicks on the sign up link to register											
3. User fill in the form											
3.1. User enters username											
3.2. User enters Email address											
3.3. User enters postal address											
3.4. User enters telephone number											
3.5. User enters date of birth											
3.6. User selects a payment method, and provide detail of payment											
4. User enters the verification code											
5. User clicks on the Submit button											

Figure 1-52 Testing procedures entered

3. Enter the result we expected by running the testing procedure.

Flow of Events		...									Show Testing Procedure
Steps			Procedures		Expected Results						
1. User clicks on the login button			1. Visit the main page. 2. Click on the login button at the bottom right of screen					The login page appear, with the sign up link appear at the bottom of page			
2. User clicks on the sign up link to register											
3. User fill in the form											
3.1. User enters username											
3.2. User enters Email address											
3.3. User enters postal address											
3.4. User enters telephone number											
3.5. User enters date of birth											
3.6. User selects a payment method, and provide detail of payment											
4. User enters the verification code											
5. User clicks on the Submit button											

Figure 1-53 Expected results entered

4. Repeat the previous steps until the testing procedure is documented.

Creating multiple flow of events

1. Click on the ... button at the toolbar.

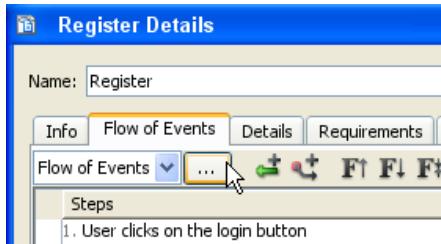


Figure 1-54 To create multiple flow of events

2. In the Flow of Events dialog box, click on the New Flow of Events button.

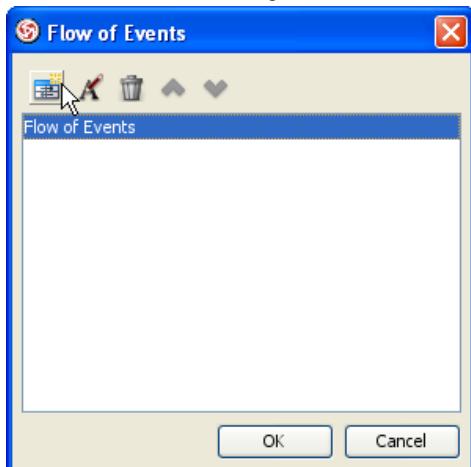


Figure 1-55 To create a flow of events

3. Enter the name of flow of events. Click **OK** to confirm.

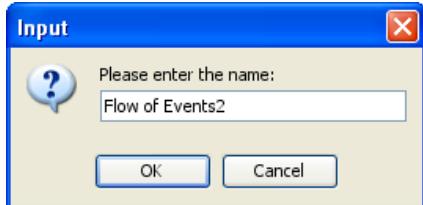


Figure 1-56 To enter the name of flow of events

4. Click **OK** to close the **Flow of Events** dialog box.

Generating activity diagram from flow of events

To generate activity diagram from flow of events, right click on the flow of events editor and select **Synchronize to Activity Diagram** from the popup menu.

Steps	Procedures	Expected Results
1. User clicks on the login button	1. Visit the main page. 2. Click on the login button at the bottom right of screen	The login page appear, with the sign up link appear at the bottom of page
2. User clicks on the sign up link to register		
3. User fill in the form		
3.1. User enters username 3.2. User enters Email address 3.3. User enters postal address 3.4. User enters telephone number 3.5. User enters date of birth 3.6. User selects a payment method, and provide detail of payment	Synchronize to Activity Diagram	
4. User enters the verification code		
5. User clicks on the Submit button		

Extension:

Steps	Procedures	Expected Results
5.a. Wrong verification code entered 1. User is warned for a wrong verification code 2. User enters the correct one 3. User submits again		
5.b. Wrong E-mail address entered: 1. User is warned for a wrong E-mail address 2. User enters a correct E-mail address 3. User submits again		

Figure 1-57 To generate activity diagram from flow of events

Elaborating use case

A Use Case can be elaborated using sequence diagrams and activity diagrams to model its interactions and activity flows respectively.

Elaborating use case by sequence diagrams

To elaborate a Use Case by sequence diagrams, create a sequence diagram as sub diagram to the Use Case.

Creating sub diagram

Right-click on the Use Case and select **Sub Diagrams > Sequence Diagram > Create Sequence Diagram** from the popup menu.

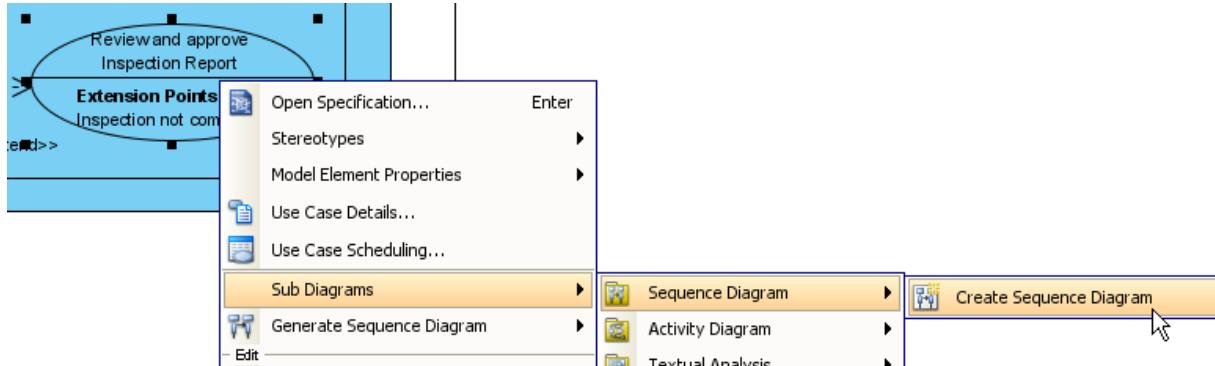


Figure 1-58 Create sequence diagram as sub diagram

Drawing sequence diagram

Draw the sequence diagram to show how user interacts with the system in the Use Case.

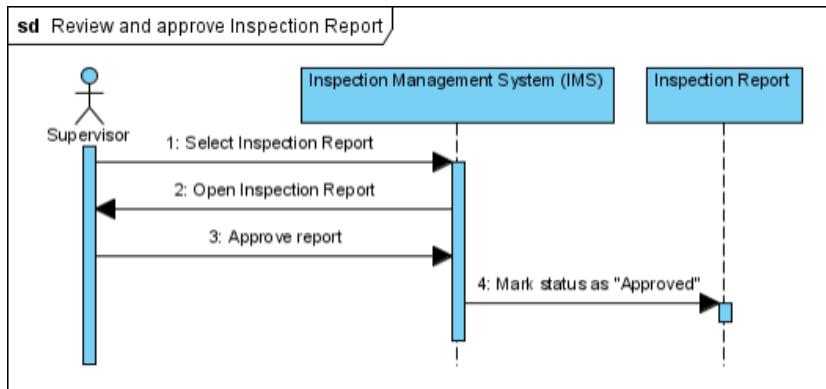


Figure 1-59 Drawn sequence diagram

Navigate to parent use case

To navigate back to the parent Use Case, click the Use Case name (name that appears before diagram name) on the diagram title bar.



Figure 1-60 Navigate to parent Use Case

Elaborating use case by activity diagrams

To elaborate a Use Case by activity diagrams, fill in the flow of events of use case and synchronize the events to activity diagram.

Synchronize flow of events to activity diagram

1. Open the use case details by right clicking on use case and selecting **Use Case Details...** from the popup menu.

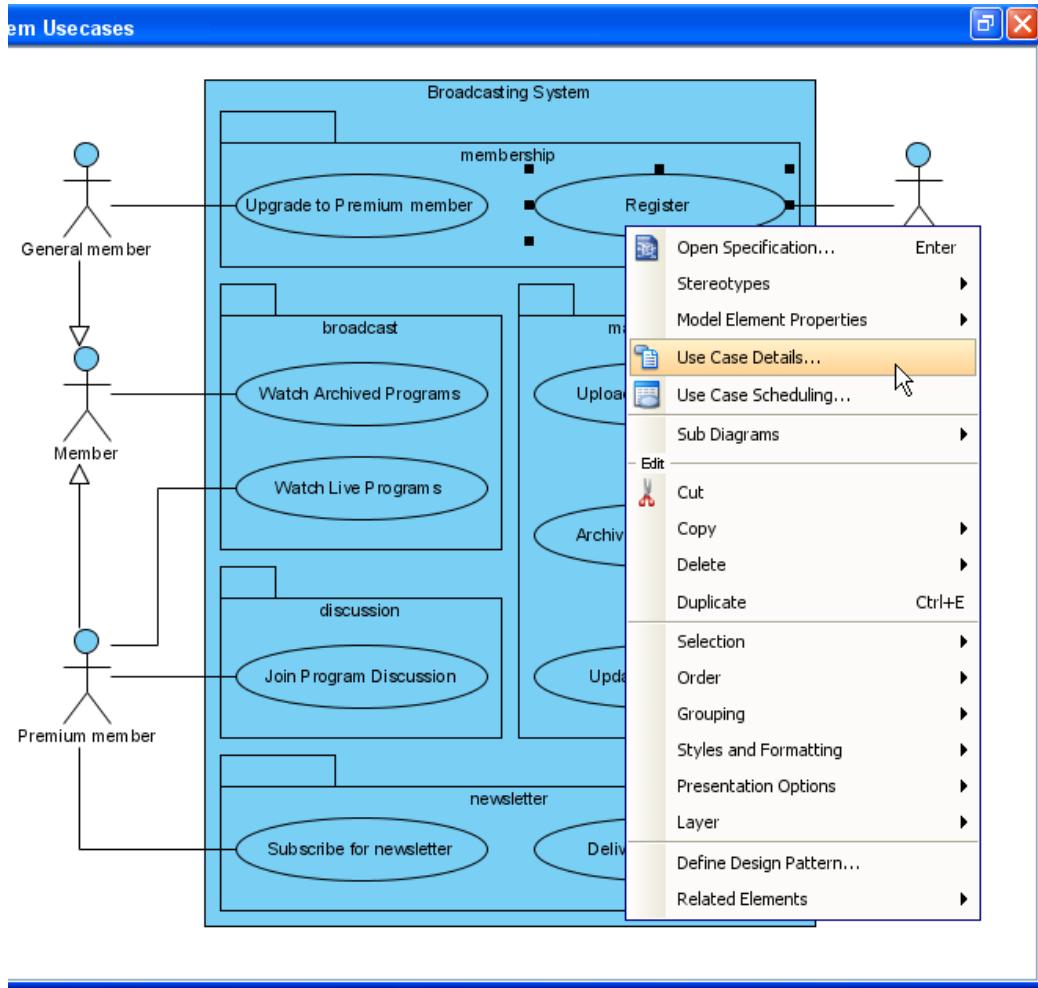


Figure 1-61 To open use case details

2. Open the **Flow of Events** tab.

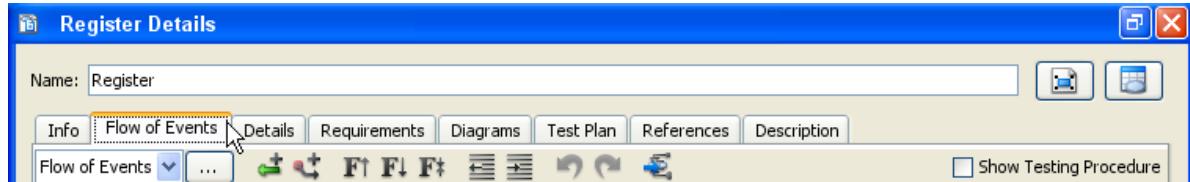


Figure 1-62 To open flow of events editor

3. Right-click on the flow of events and select **Synchronize to Activity Diagram** from the popup menu.

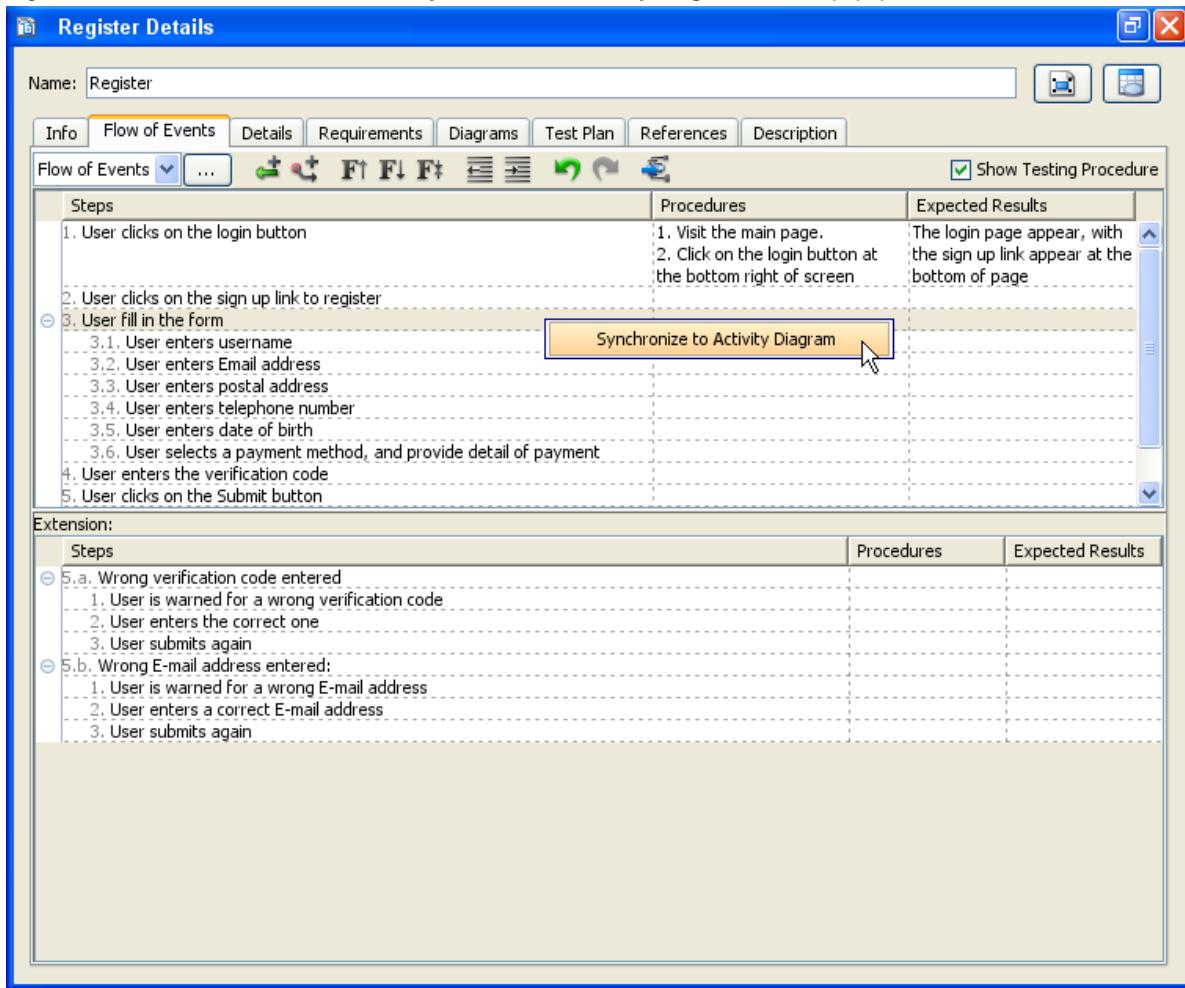


Figure 1-63 To generate activity diagram from flow of events

Navigate to flow of events

To navigate back to the flow of events, right click on an action in activity diagram and select **Related Elements > Open Flow of event** from the popup menu.

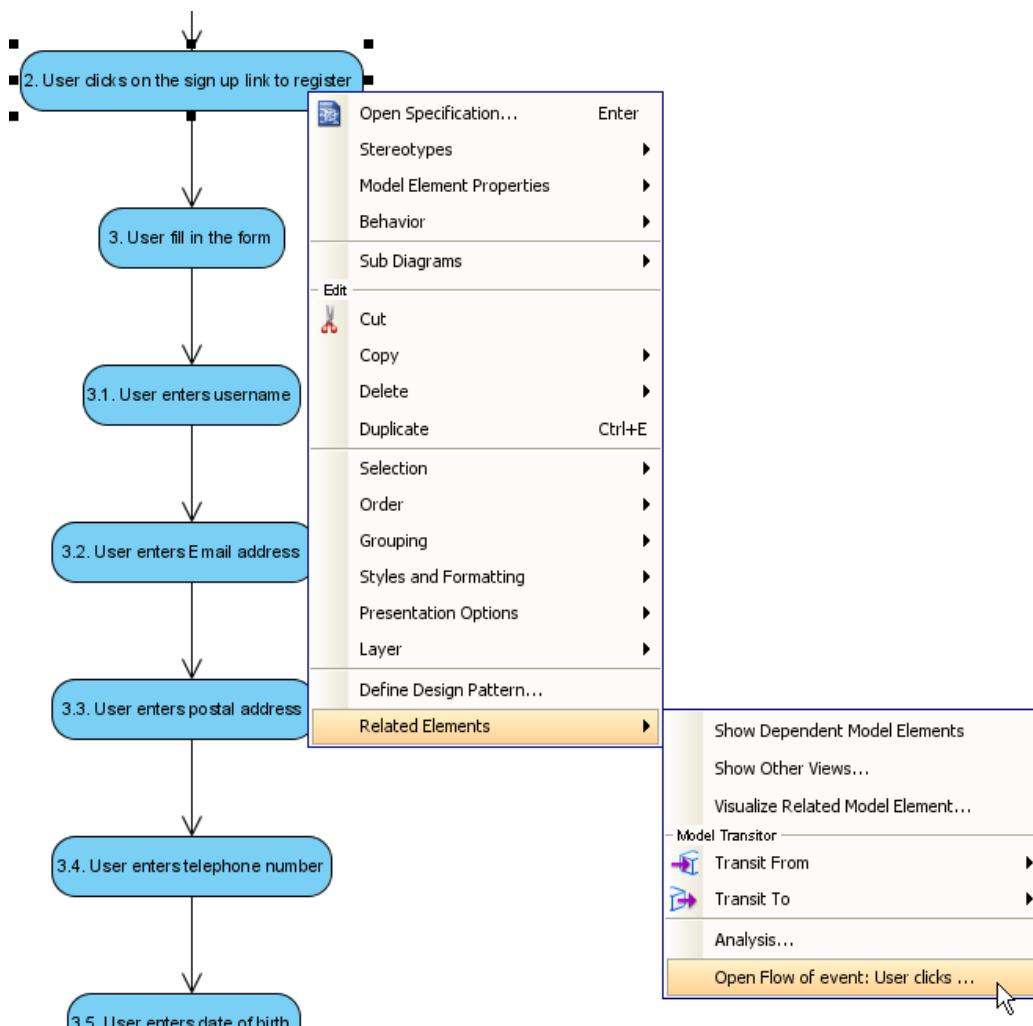


Figure 1-64 To open flow of events from activity diagram

Update activity diagram from flow of events

To synchronize changes made in flow of events to activity diagram, right click on flow of events and select **Synchronize to Activity Diagram** from the popup menu.

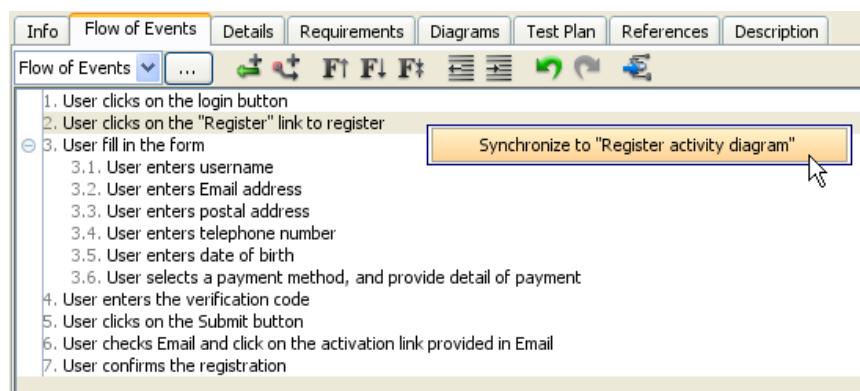


Figure 1-65 To update changes from flow of events to activity diagram

Managing actors with actors grid

Actors Grid is a table with actors listed in it. It lets you to access all actors in a project or diagram, check their related use cases, lookup and create actors.

Opening the actors grid

To open actors grid, select **Tools > Model Elements Grid > Open Actors Grid** from the main menu.

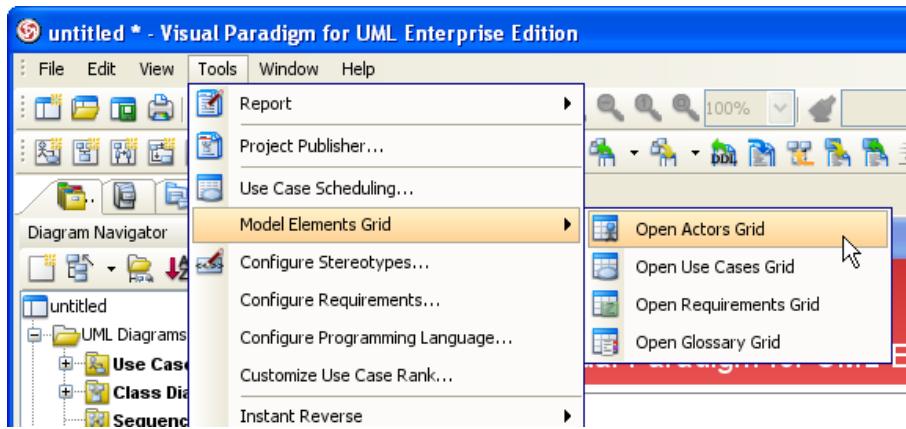


Figure 1-66 To open actors grid

The actors grid appear.

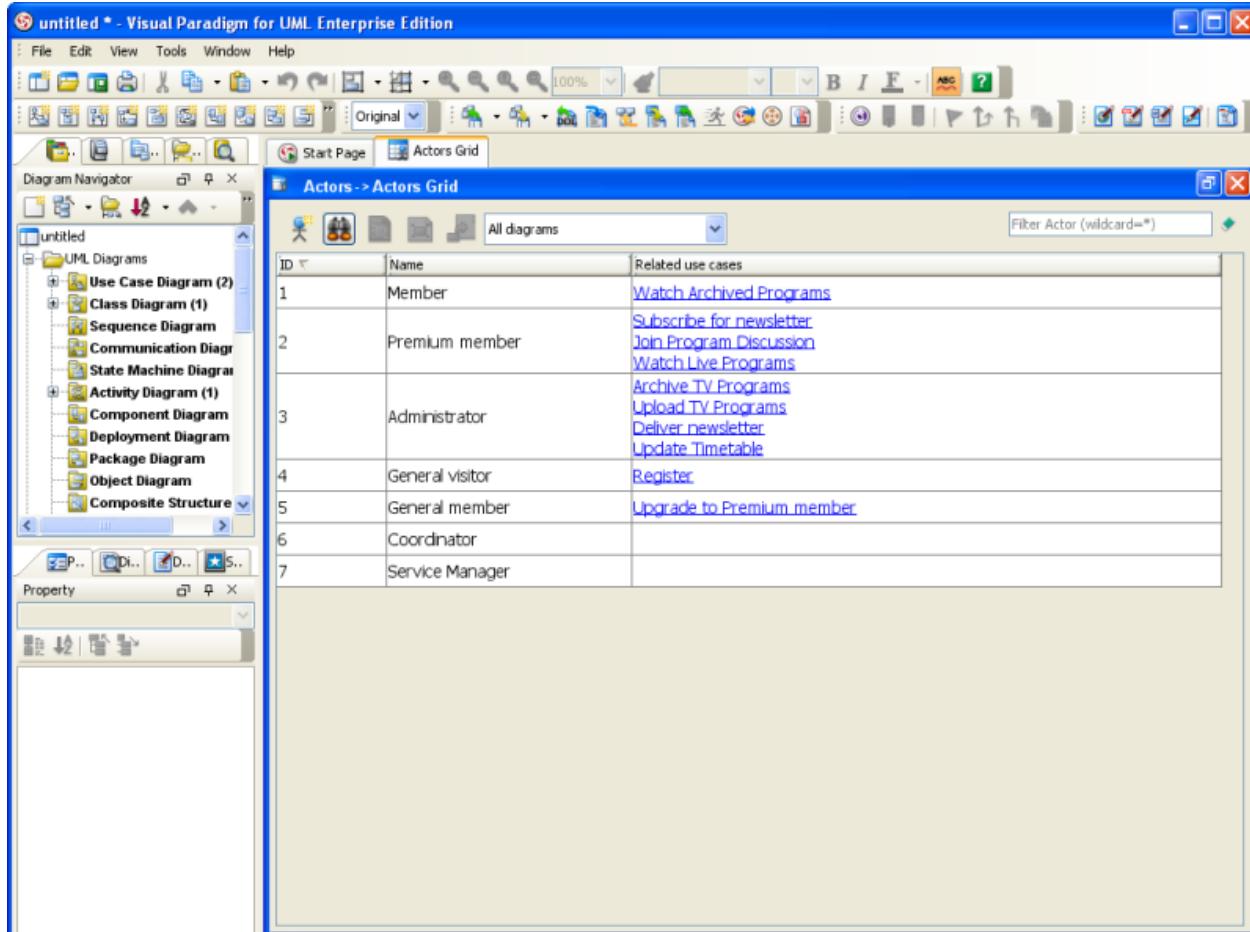


Figure 1-67 The actors grid

Field	Description
	Create a new actor.
	To find an actor by providing its name or documentation.
	To open the specification of actor selected in grid.

	To show the view(s) of actor selected in grid.
	To visualize an actor selected in grid.
All diagrams	
Filter Actor (wildcard=*)	

Table 1-3 The fields in actors grid

Accessing use case from actor

Click on the name of use case in the **Related use case** cell of the actor that we want to access its related use case.

ID	Name	Related use cases
1	Member	Watch Archived Programs
2	Premium member	Subscribe for newsletter Join Program Discussion Watch Live Programs
3	Administrator	Archive TV Programs Upload TV Programs Deliver newsletter Update Timetable

Figure 1-68 Access use case from actor

Creating actor

1. Click on above the grid.

Start Page		Actors Grid
Actors -> Actors Grid		
All diagrams		
ID	Name	
1	Member	
2	Premium member	

Figure 1-69 To create an actor in grid

2. Name the actor. You may optionally reset the ID by double-clicking on the ID cell and entering a new one.

6	Coordinator	
7	Service Manager	
8	General Clerk	<input type="text"/>

Figure 1-70 Name actor

NOTE: The actors created in actors grid are automatically put under the Actors model. You may move to another model through dragging and dropping in Model Explorer.

Visualizing an actor

You can form a diagram with actor, or show it in an existing diagram by visualizing it in actors grid. To visualize an actor:

1. Select the actor to visualize.

6	Coordinator	
7	Service Manager	
8	General Clerk	

Figure 1-71 Selecting an actor to visualize

2.

Click on  above the grid.

Actors -> Actors Grid	
ID	Name
1	Member

Figure 1-72 Create textual analysis

3.

If you want to visualize actor in a new diagram, keep **Create new diagram** selected, select the type of diagram to create and specifying the diagram name.

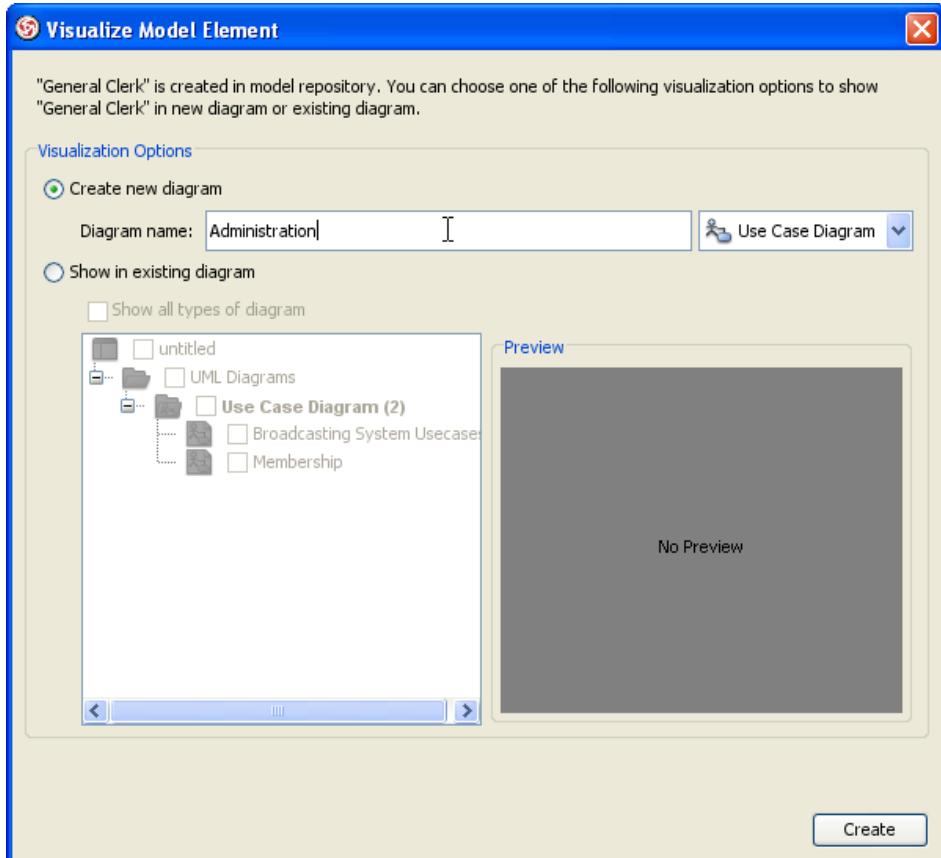


Figure 1-73 Name a new diagram

If you want to visualize actor in an existing diagram, select **Show in existing diagram** and select a diagram in the diagram list.

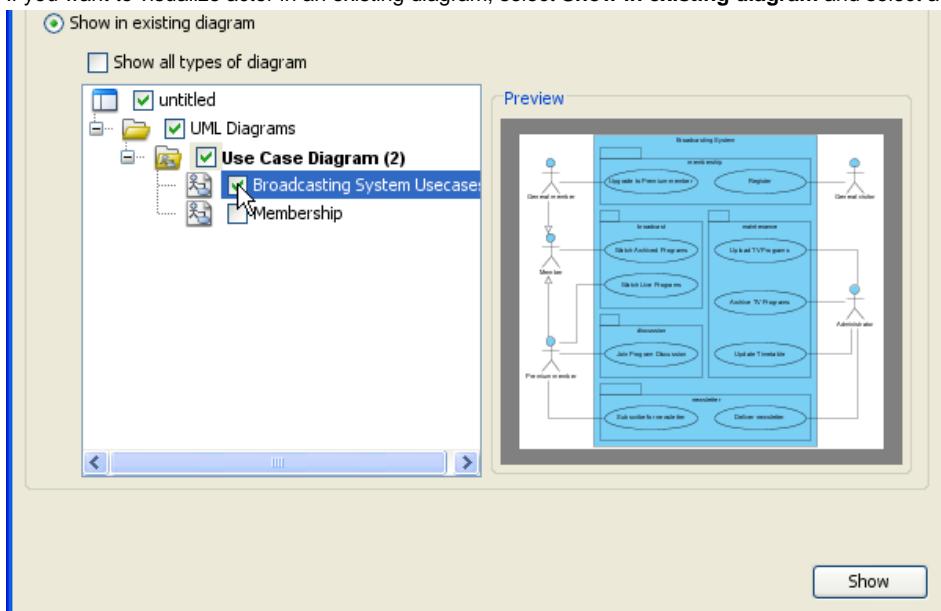


Figure 1-74 Select an existing use case diagram to visualize the actor

- Click **Create** at the bottom of the dialog box.

Filter actors

By filtering actors, actors that do not match the required naming convention are filtered. To filter, enter the name of actor, or part of its name at the top right of grid. You can make use of the asterisk (*) character to represent any character(s).

The screenshot shows a software interface titled 'Actors -> Actors Grid'. At the top, there are icons for creating, deleting, and saving, followed by a dropdown menu 'All diagrams' and a search bar containing the text '* member'. Below the header is a table with columns 'ID', 'Name', and 'Related use cases'. Two rows are visible: row 2 contains 'Premium member' with three related use cases: 'Subscribe for newsletter', 'Join Program Discussion', and 'Watch Live Programs'; row 5 contains 'General member' with one related use case: 'Upgrade to Premium member'.

ID	Name	Related use cases
2	Premium member	Subscribe for newsletter Join Program Discussion Watch Live Programs
5	General member	Upgrade to Premium member

Figure 1-75 To filter actor by name

Find actor

To find out actors that match specific naming or documentation pattern:

- Click above the grid.

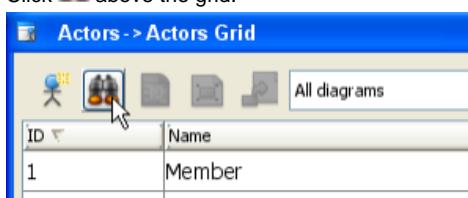


Figure 1-76 To find an actor

- In the **Search Text** text box, enter the text to search.

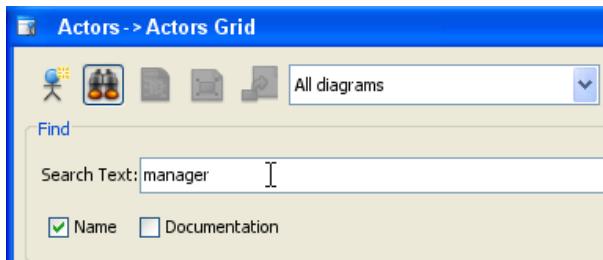


Figure 1-77 Enter actor name

- Specify whether to search the names and/or documentation of actors.

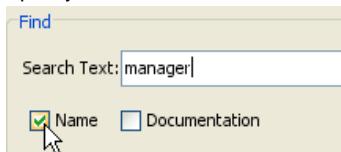


Figure 1-78 Select whether to search the name and/or documentation of actors

4. Click **Find**. The grid is updated to highlight the matched actors.

The screenshot shows a software interface titled "Actors -> Actors Grid". At the top, there are several icons and a dropdown menu labeled "All diagrams". On the right, there is a search bar with the placeholder "Filter Actor (wildcard=*)" and a "Find" button. Below the search bar, there is a "Find" section with a "Search Text" input field containing "manager", a checked "Name" checkbox, and an unchecked "Documentation" checkbox. A "Find" button is highlighted with a yellow border. The main area is a table with columns "ID", "Name", and "Related use cases". The rows represent different actor types:

ID	Name	Related use cases
1	Member	Watch Archived Programs
2	Premium member	Subscribe for newsletter Join Program Discussion Watch Live Programs
3	Administrator	Archive TV Programs Upload TV Programs Deliver newsletter Update Timetable
4	General visitor	Register
5	General member	Upgrade to Premium member
6	Coordinator	
7	Service Manager	
8	General Clerk	

Figure 1-79 Execute the finding

Managing use cases with use cases grid

Use Cases Grid is a table with use cases listed in it. It lets you to access all use cases in a project or diagram, check their related actors, lookup and create use case.

Opening the use cases grid

To open use cases grid, select **Tools > Model Elements Grid > Open Use Cases Grid** from the main menu.

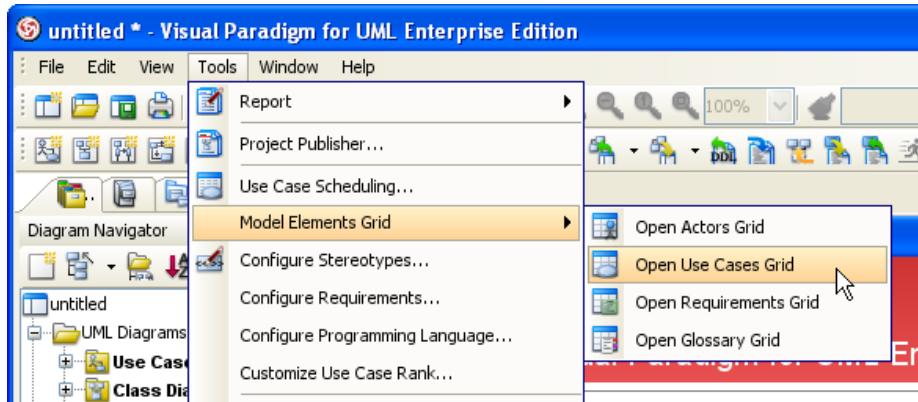
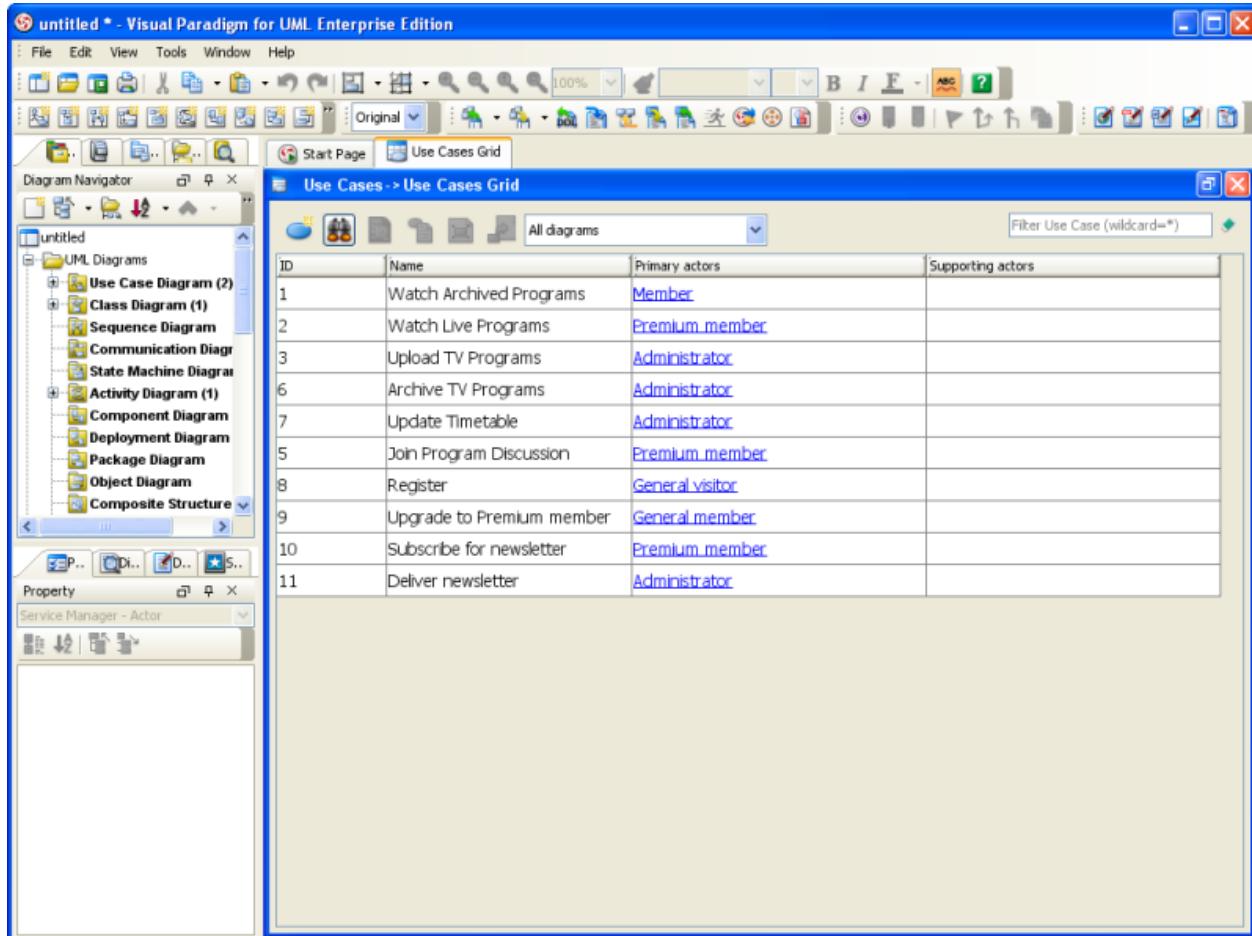


Figure 1-80 To open use cases grid

The use cases grid appear.



ID	Name	Primary actors	Supporting actors
1	Watch Archived Programs	Member	
2	Watch Live Programs	Premium member	
3	Upload TV Programs	Administrator	
6	Archive TV Programs	Administrator	
7	Update Timetable	Administrator	
5	Join Program Discussion	Premium member	
8	Register	General visitor	
9	Upgrade to Premium member	General member	
10	Subscribe for newsletter	Premium member	
11	Deliver newsletter	Administrator	

Figure 1-81 The use cases grid

Field	Description
	Create a new use case.
	To find a use case by providing its name or documentation.
	To open the specification of use case selected in grid.
	To open the use case details of use case selected in grid.

	To show the view(s) of use case selected in grid.
	To visualize a use case selected in grid.
All diagrams	To filter use cases by selecting the diagram(s) (or all diagrams) that contain the use casesuse case.
Filter Use Case (wildcard=*)	To filter use cases by name. The rubber on the right hand side is for clearing the filter content.

Table 1-4 The fields in use cases grid

Accessing actor from use case

Click on the name of actor in the **Primary actors/Supporting actors** cell of the use case that we want to access its related actor.

ID	Name	Primary actors
1	Watch Archived Programs	Member
2	Watch Live Programs	Premium_member
3	Upload TV Programs	Administrator

Figure 1-82 Access actor from use case

Creating use case

1. Click on above the grid.

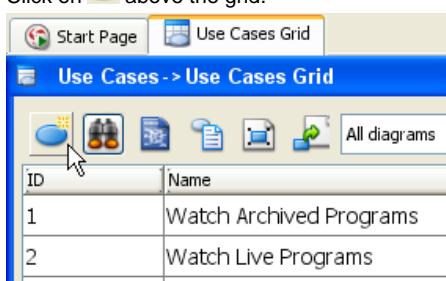


Figure 1-83 To create a use case in grid

2. Name the use case. You may optionally reset the ID by double-clicking on the ID cell and entering a new one.

10	Subscribe for newsletter	Pr
11	Deliver newsletter	Adr
12	Remove membership	I

Figure 1-84 Name use case

NOTE: The use cases created in use cases grid are automatically put under the Use Cases model. You may move to another model through dragging and dropping in Model Explorer.

Browsing use case's details

To browse for the details of a use case, select the use case in grid and click on the button. This opens its use case details.

Visualizing a use case

You can form a diagram with use case, or show it in an existing diagram by visualizing it in use cases grid. To visualize a use case:

1. Select the use case to visualize.

10	Subscribe for newsletter	Pr
11	Deliver newsletter	Ac
12	Remove membership	

Figure 1-85 Selecting a use case to visualize

2.

Click on  above the grid.

ID	Name
1	Watch Archived Programs

Figure 1-86 Create textual analysis

3.

If you want to visualize use case in a new diagram, keep **Create new diagram** selected, select the type of diagram to create and specifying the diagram name.

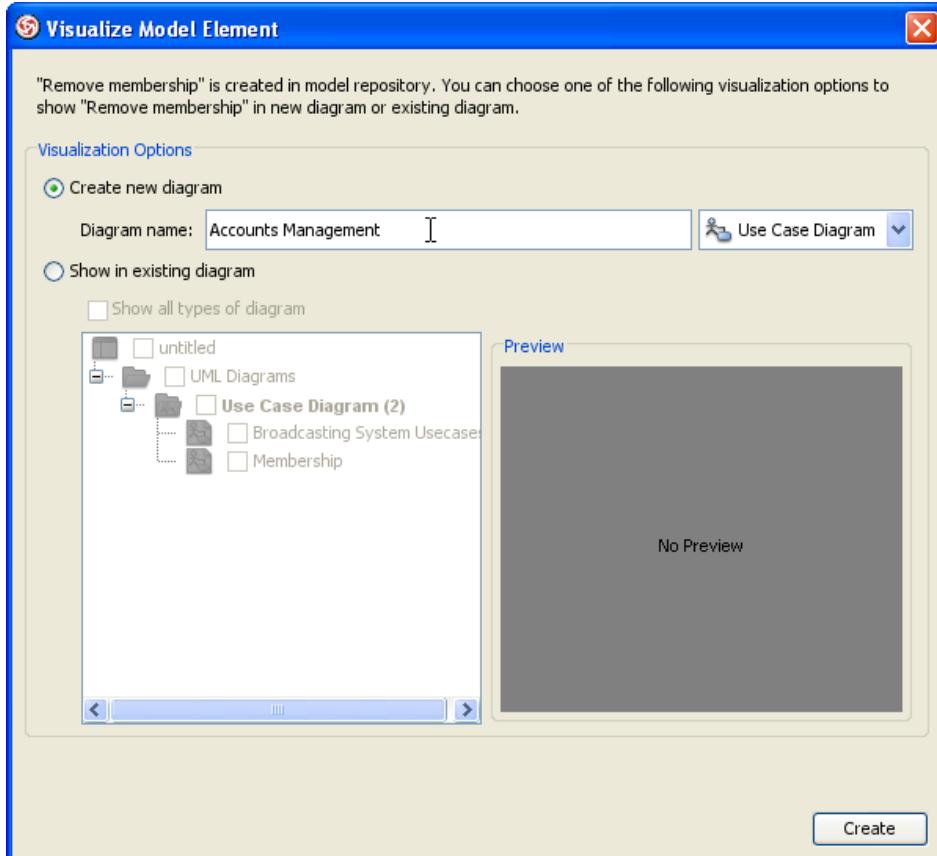


Figure 1-87 Name a new diagram

If you want to visualize use case in an existing diagram, select **Show in existing diagram** and select a diagram in the diagram list.

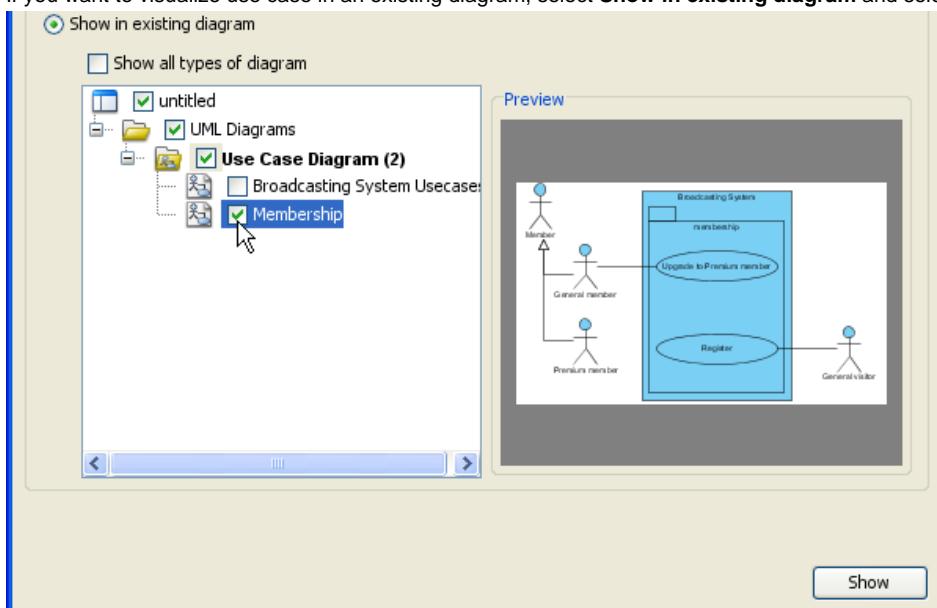


Figure 1-88 Select an existing use case diagram to visualize the use case

- Click **Create** at the bottom of the dialog box.

Filter use cases

By filtering use cases, use cases that do not match the required naming convention are filtered. To filter, enter the name of use case, or part of its name at the top right of grid. You can make use of the asterisk (*) character to represent any character(s).



The screenshot shows a software interface titled 'Use Cases -> Use Cases Grid'. At the top, there is a toolbar with icons for file operations and a search bar containing the text '* program'. Below the toolbar is a dropdown menu labeled 'All diagrams'. The main area is a grid table with columns: 'ID', 'Name', 'Primary actors', and 'Supporting actors'. The data in the grid is as follows:

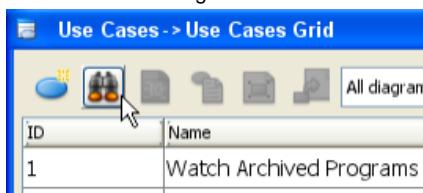
ID	Name	Primary actors	Supporting actors
1	Watch Archived Programs	Member	
2	Watch Live Programs	Premium member	
3	Upload TV Programs	Administrator	
6	Archive TV Programs	Administrator	
5	Join Program Discussion	Premium member	

Figure 1-89 To filter use case by name

Find use case

To find out use cases that match specific naming or documentation pattern:

- Click  above the grid.

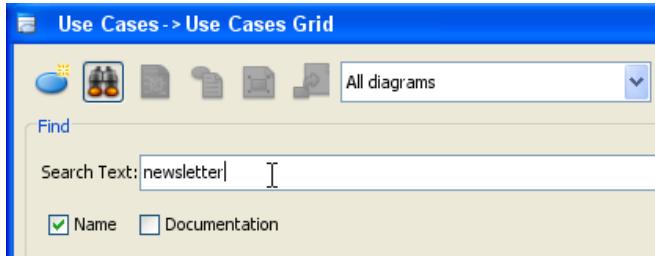


The screenshot shows the same software interface as Figure 1-89, but the search results are more limited. The grid only contains one row for 'Watch Archived Programs'.

ID	Name
1	Watch Archived Programs

Figure 1-90 To find a use case

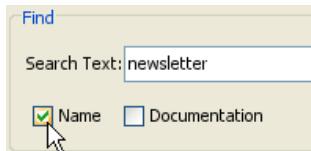
- In the **Search Text** text box, enter the text to search.



The screenshot shows the software interface with the 'Find' dialog box open. The 'Search Text' field contains the text 'newsletter'. There are two checkboxes below the search field: 'Name' (which is checked) and 'Documentation'.

Figure 1-91 Enter use case name

- Specify whether to search the names and/or documentation of use cases.



The screenshot shows the 'Find' dialog box with the 'Search Text' field containing 'newsletter'. The 'Name' checkbox is checked, while the 'Documentation' checkbox is unchecked.

Figure 1-92 Select whether to search the name and/or documentation of use cases

4. Click **Find**. The grid is updated to highlight the matched use cases.

The screenshot shows a software interface titled 'Use Cases -> Use Cases Grid'. At the top, there are icons for file operations (New, Open, Save, Print, Find, Filter) and a dropdown menu labeled 'All diagrams'. A search bar at the top right contains the text 'Filter Use Case (wildcard=*)' with a green search icon. Below the search bar is a 'Find' section with a 'Search Text' input field containing 'newsletter', a 'Find' button with a magnifying glass icon, and two checkboxes: 'Name' (checked) and 'Documentation' (unchecked). The main area is a grid table with columns: 'ID', 'Name', 'Primary actors', and 'Supporting actors'. The rows list various use cases, with the 10th row, 'Subscribe for newsletter', highlighted in blue. The 'Primary actors' column for this row contains the word 'Premium member' in blue, indicating it is the matched result for the search term 'newsletter'.

ID	Name	Primary actors	Supporting actors
1	Watch Archived Programs	Member	
2	Watch Live Programs	Premium member	
3	Upload TV Programs	Administrator	
6	Archive TV Programs	Administrator	
7	Update Timetable	Administrator	
5	Join Program Discussion	Premium member	
8	Register	General visitor	
9	Upgrade to Premium member	General member	
10	Subscribe for newsletter	Premium member	
11	Deliver newsletter	Administrator	
12	Remove membership		

Figure 1-93 Execute the finding

Performing textual analysis

Using textual analysis, you can identify candidate objects from a problem statement, create model elements from the candidate objects, and finally draw diagram using the created model elements.

Creating textual analysis

To create textual analysis, select menu **File > New Diagram > Requirements Capturing > Textual Analysis**.

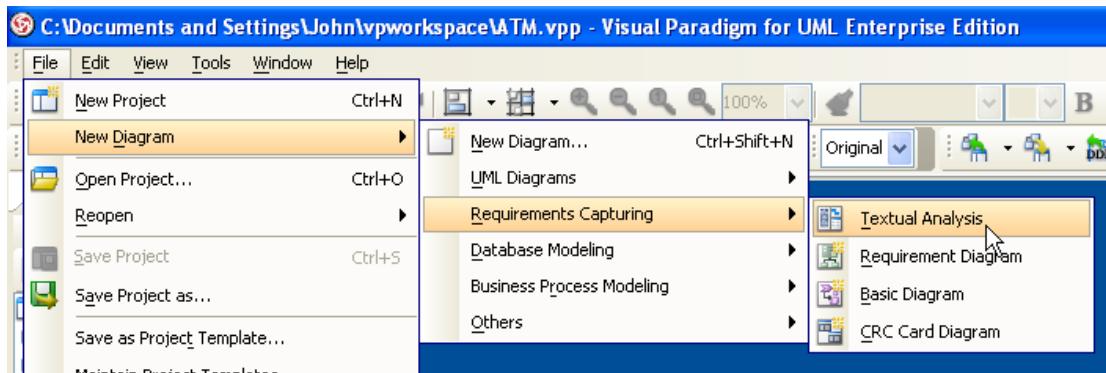


Figure 2-1 Create textual analysis

Entering problem statement

You can either enter problem statement by typing in the text area, or by importing a text file. To import a text file, click **Import Text File** on the toolbar.

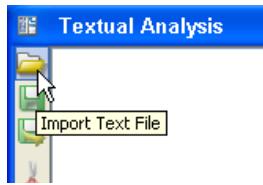


Figure 2-2 Import text file

When the file dialog box appears, select the text file to import. After that, the imported problem statement will appear in the text area.

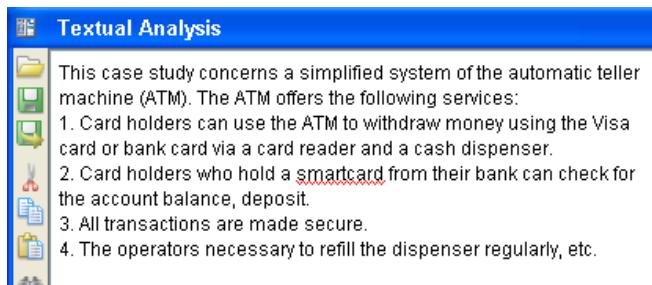


Figure 2-3 Imported problem statement

Creating candidate object by drag and drop

To create candidate object by drag and drop, select the words (**Card holders** in this example) in the text area and drag.

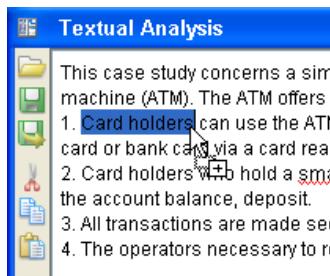


Figure 2-4 Drag words in text area

Move the mouse over the pane on the right of the text area and then release the mouse button. A candidate object is created. The occurrences of the candidate object name are highlighted in the text area.

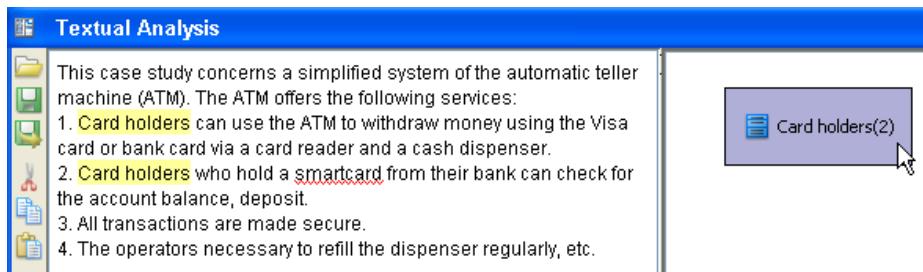


Figure 2-5 Candidate object created

Candidate object created by drag and drop is set to "Class" type by default. However "Card holders" should be Actors. To change the candidate object type to Actor, right-click on the candidate object and select **Actor** from the popup menu.

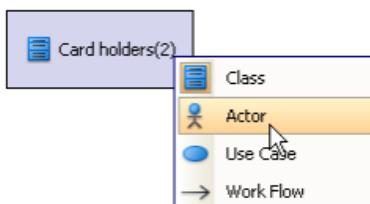


Figure 2-6 Change candidate object type to Actor

Creating candidate object by popup menu

To create candidate object (e.g. of type "Use Case") by popup menu, select the words (e.g. withdraw money) in the text area, right-click on the selection and select **Add text as Use Case** from the popup menu.

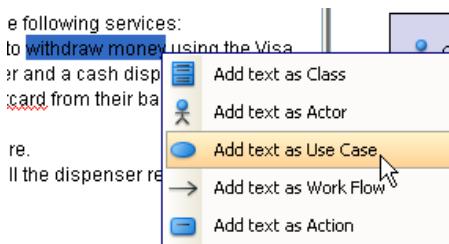


Figure 2-7 Change candidate object type to Actor

A candidate object "withdraw money" of type "Use Case" is created.

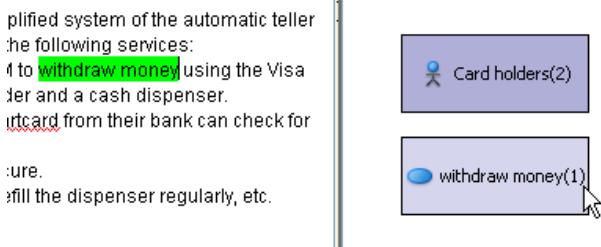


Figure 2-8 Candidate object created for "withdraw money"

Writing description for candidate object

The table in the textual analysis editor allows you to view and edit details of candidate objects. You can edit description of a candidate object by typing in its **Class Description** cell. Before you edit the description, you may resize the row to make more room for the content.

No.	Candidate Class	Extracted Text	Type	Class Description	Occurrences
1	Card holders	Card holders	Actor		2
2	withdraw money	withdraw money	Use Case		1

Figure 2-9 Resize rows

Double-click on the **Class Description** cell to enter description.

Type	Class Description
Actor	ATM card holders
Use Case	Take money out of the ATM card holder's bank account.

Figure 2-10 Enter description

Creating model elements

To create model element for candidate object, say **Card holders** in this example, right-click it and select **Create Actor Model Element** from the popup menu.

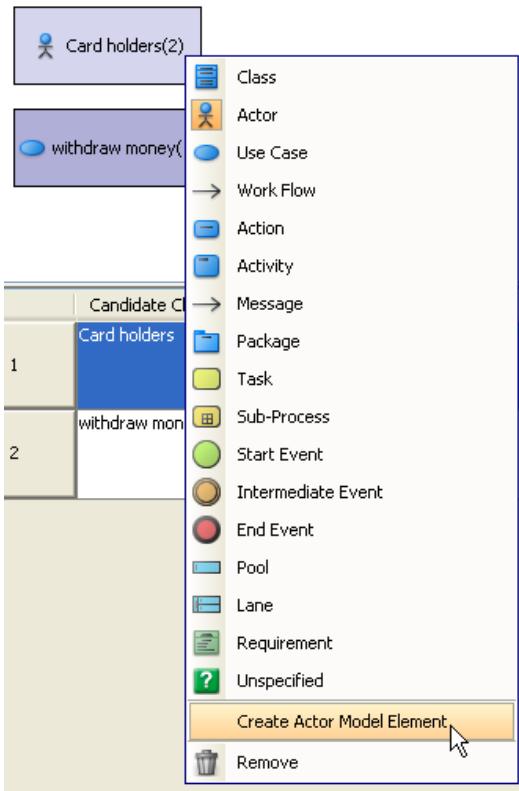


Figure 2-11 Create Actor model element from candidate object

Follow the same step to create model element for **withdraw money**. You should see the created model elements in the Model Explorer.

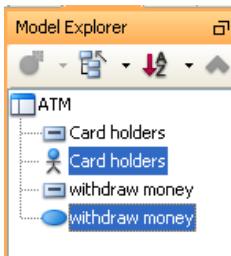


Figure 2-12 Model elements created

Creating use case diagram

Now we are going to use the model elements in use case diagram. To create a use case diagram, select menu **File > New Diagram > UML Diagrams > Use Case Diagram**.

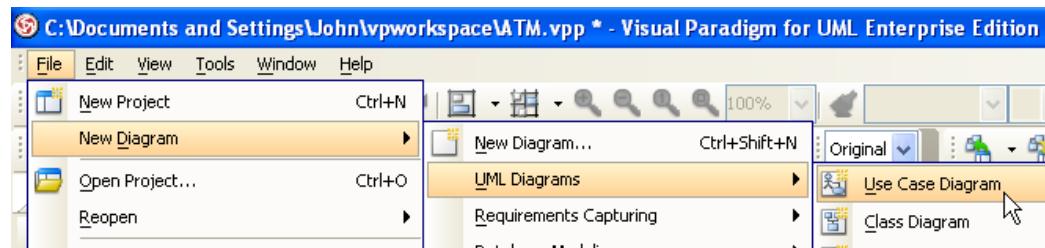


Figure 2-13 Create use case diagram

Visualize created model elements

To visualize the created model elements, simply select them in Model Explorer, drag and drop to the use case diagram.

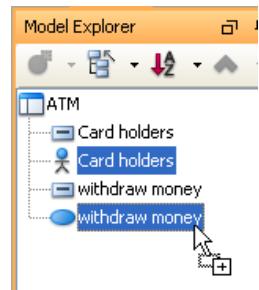


Figure 2-14 Drag model elements in Model Explorer

Shapes will be created for the dropped model elements.

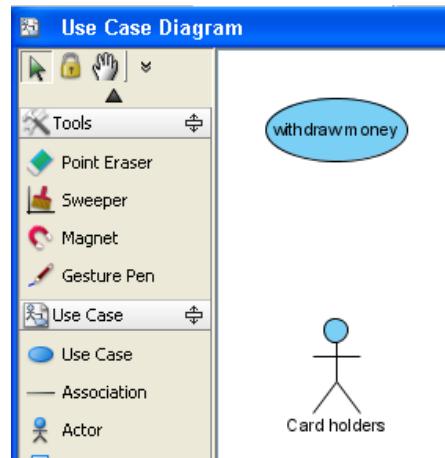


Figure 2-15 Model elements visualized

Drawing requirement diagrams

Creating requirement diagram

To create requirement diagram, select menu **File > New Diagram > Requirements Capturing > Requirement Diagram**.

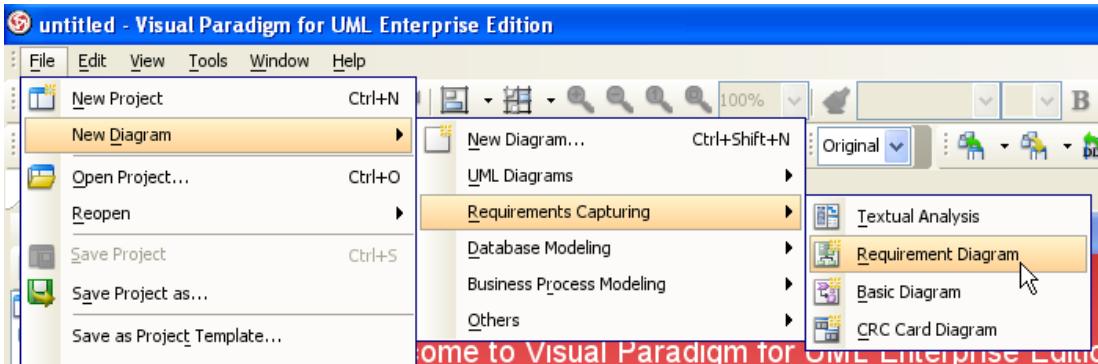


Figure 2-16 Creating requirement diagram

Creating requirement

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

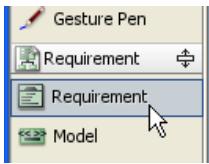


Figure 2-17 Creating Requirement

Decomposing requirement

To decompose a Requirement, click the **Containment** -> **Requirement** resource and drag.

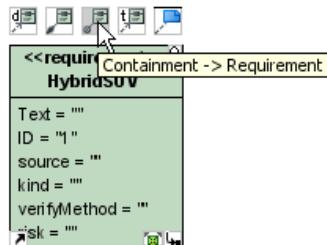


Figure 2-18 Decomposing Requirement

Move the mouse over empty space of the diagram and then release the mouse button, a Requirement together with a Containment relationship will be created.

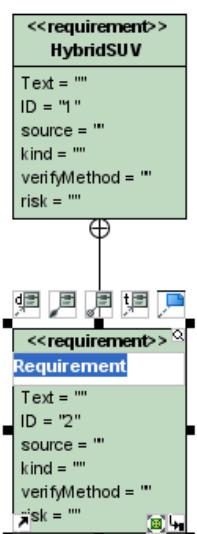


Figure 2-19 Requirement and Containment created

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.

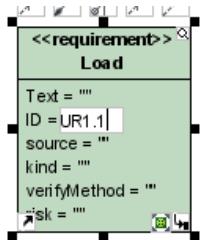


Figure 2-20 Inline editing Requirement properties

Edit requirement properties with specification dialog box

You can also open specification dialog box of a Requirement to edit its properties. Click **Open Specification** resource of the Requirement.

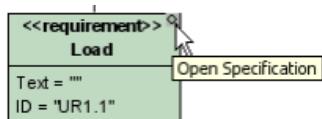


Figure 2-21 Open specification of Requirement

The **Requirement Specification** dialog box shows. Edit the properties and click **OK** to apply the changes.

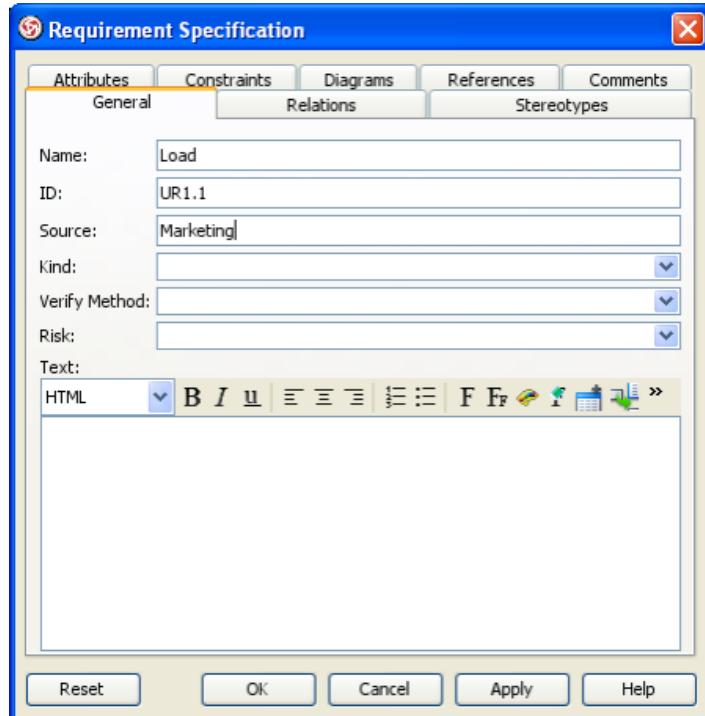


Figure 2-22 Requirement Specification

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.

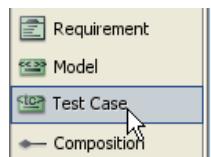


Figure 2-23 Creating Test Case

Click Verify -> Requirement resource of Test Case and drag.

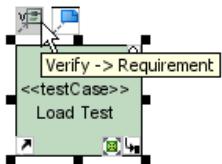


Figure 2-24 Linking Requirement with Test Case

Move the mouse over a Requirement and then release the mouse button, a Verify relationship will be created from the Test Case to the Requirement.

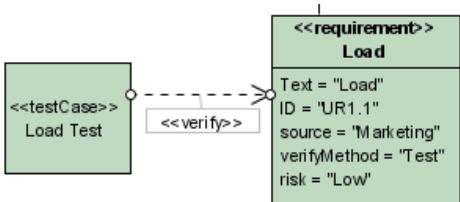


Figure 2-25 Verify relationship created

Defining your own set of requirement types

Creating new requirement type

To create new Requirement type, select menu **Tools > Configure Requirements....**

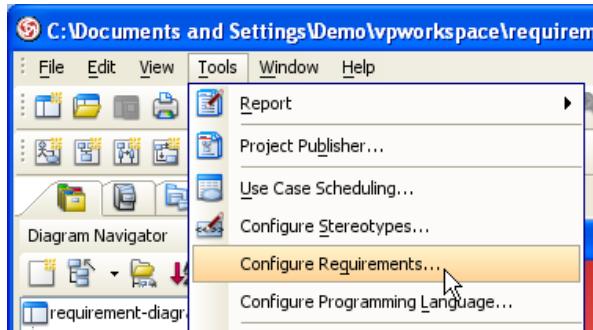


Figure 2-26 Configure Requirements

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

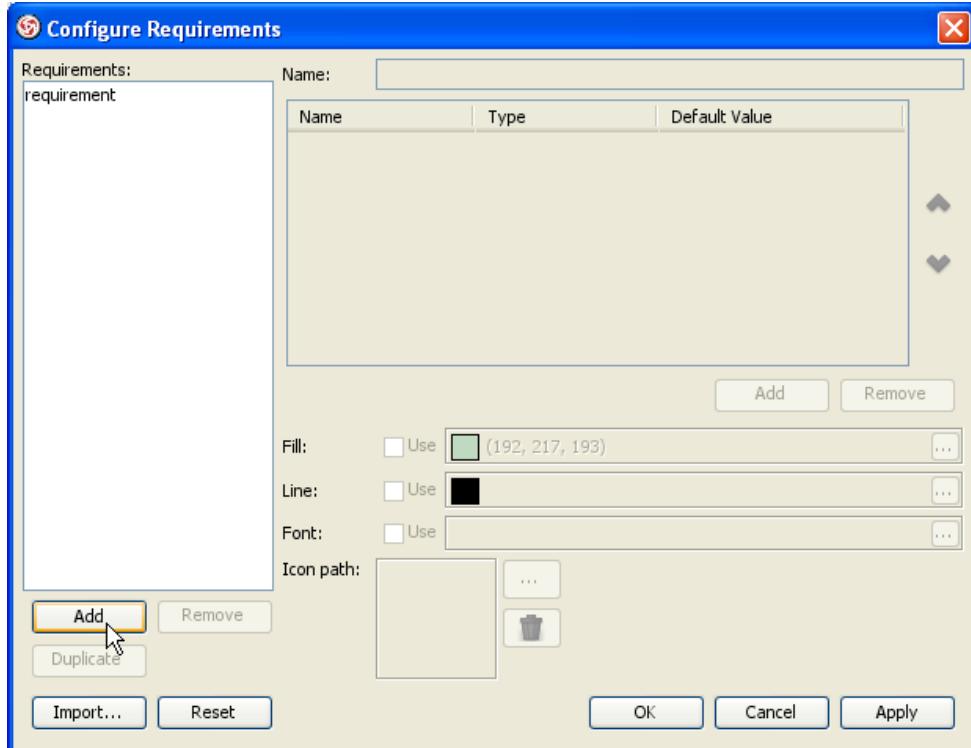


Figure 2-27 Configure Requirements dialog box

Enter name of the Requirement type.

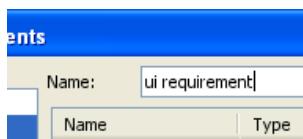


Figure 2-28 Enter name of Requirement type

The "ui requirement" will have four attributes: **Text** (HTML Text), **ID** (Text), **source** (Text), **risk** (Enumeration). Let's add them.

Click **Add** below the attribute table and select **Documentation Attribute** (an attribute with rich text content).

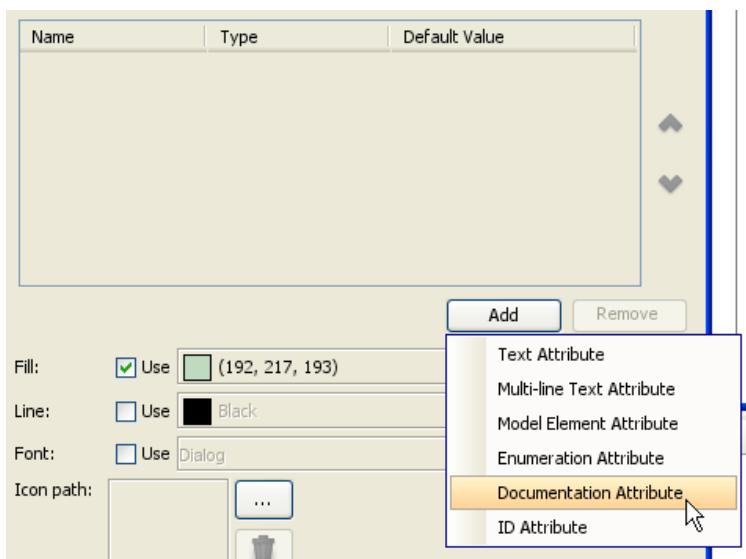


Figure 2-29 Add documentation attribute

An attribute named *Text* of type *HTML Text* is created.

Let's add the **ID** attribute. Click **Add** and select **Text Attribute** (an attribute with plain text value).

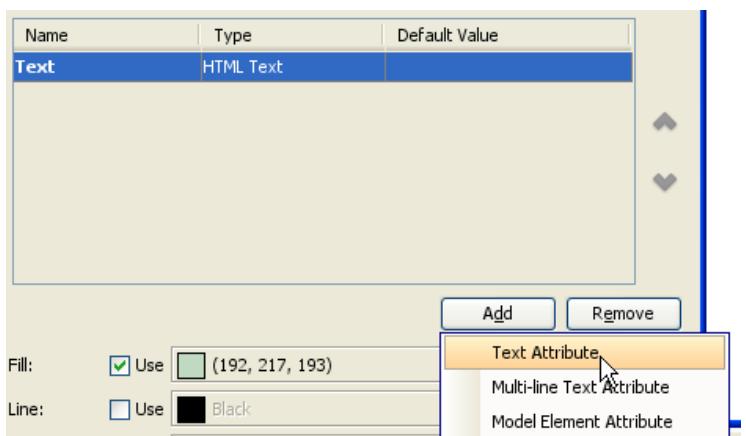


Figure 2-30 Add text attribute

Name the new attribute *ID*.

Name	Type
Text	HTML Text
ID	Text

Figure 2-31 ID attribute

Follow the previous steps to add a *Text* attribute named *source*.

Name	Type
Text	HTML Text
ID	Text
source	Text

Figure 2-32 source attribute

Let's add the **risk** attribute. Click **Add** and select **Enumeration Attribute** (an attribute with a list of selectable values).

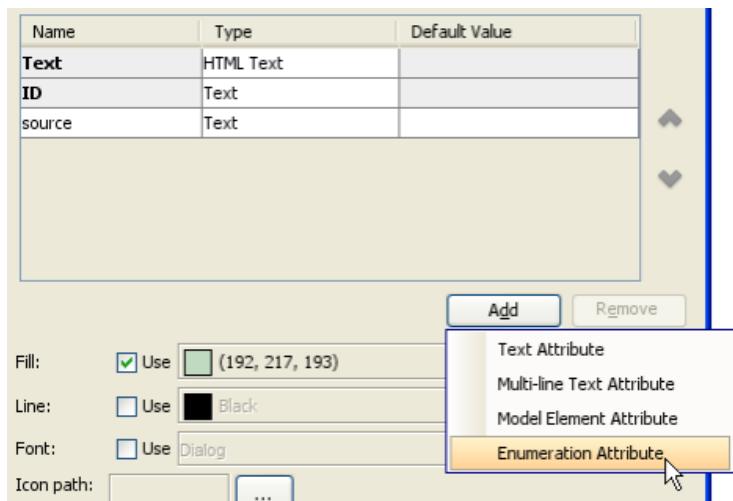


Figure 2-33 Add enumeration attribute

Name the attribute *risk*, and then click **Edit Enumeration....**

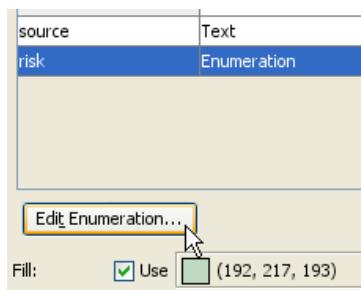


Figure 2-34 Edit enumeration

The **Edit Enumeration** dialog box appears. Click **Add** and then name the new item *High*.

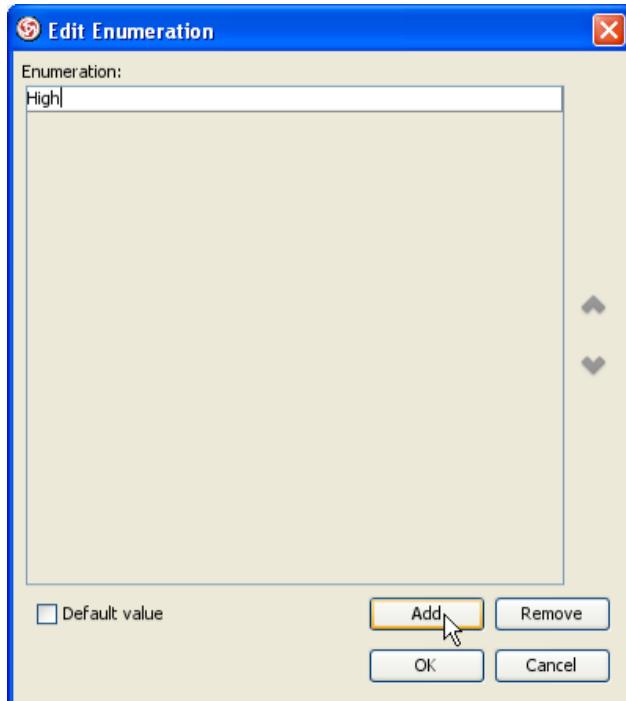


Figure 2-35 Add enumeration

Continue to create other items *Medium* and *Low*.

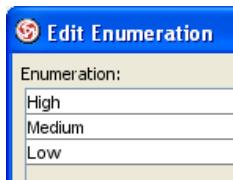


Figure 2-36 Enumeration values

Select **Medium** and select **Default value** to make it the default value for this attribute.

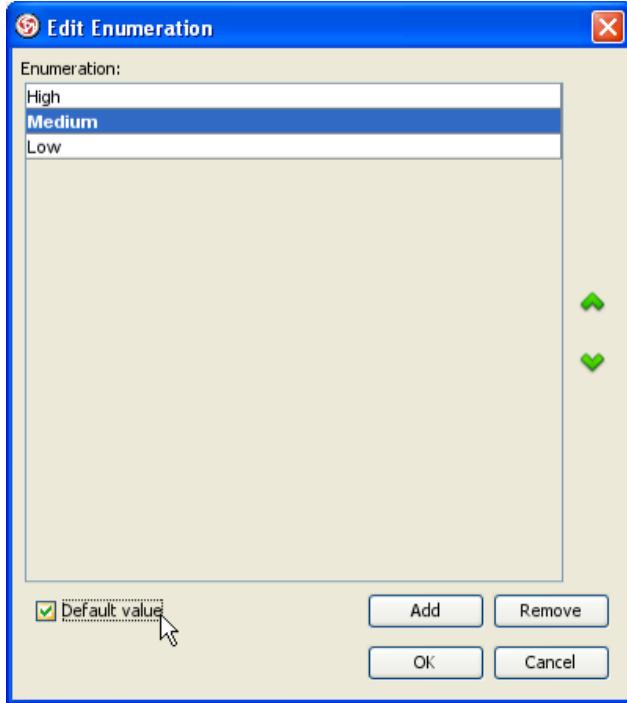


Figure 2-37 Set default value

Click **OK** to apply the changes.

You can also specify appearance of the shape that belongs to this Requirement type. Let's change the fill color and font of "ui requirement".

Click on the ... button of the **Fill** property.



Figure 2-38 Edit fill property

Select **Blue** as fill color and click **OK**.

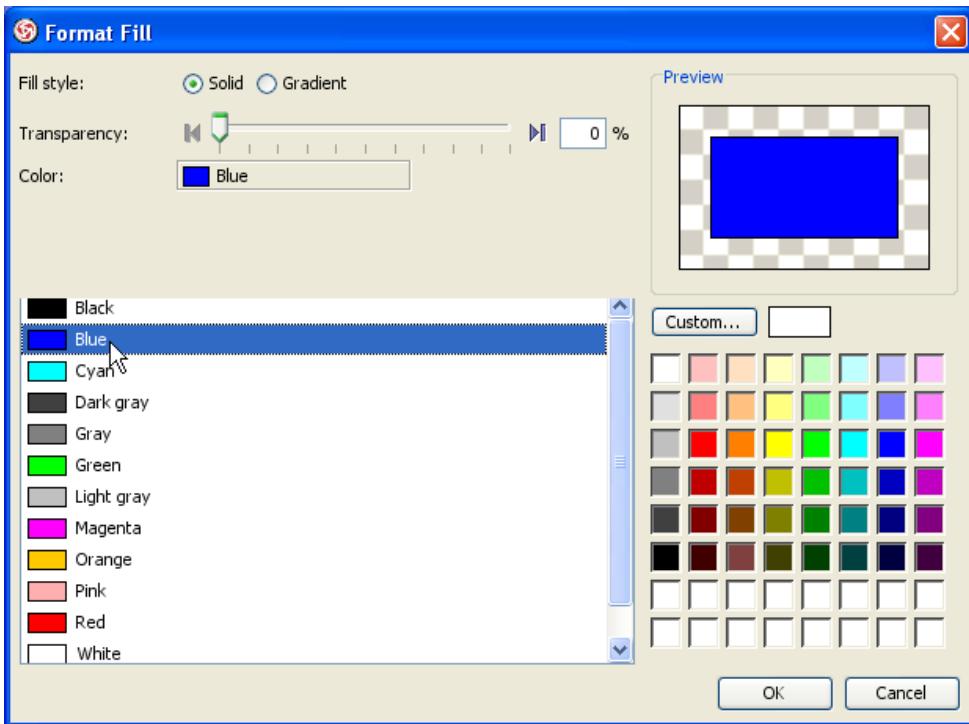


Figure 2-39 Format Fill dialog box

Select **Use** of the **Font** property to indicate the font settings will be applied to shape, and then click the ... button.

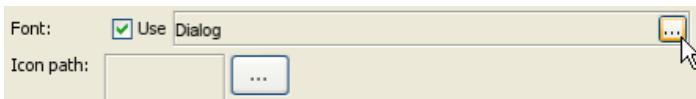


Figure 2-40 Edit font property

Select **Bold** in **Font Style** and click **OK**.

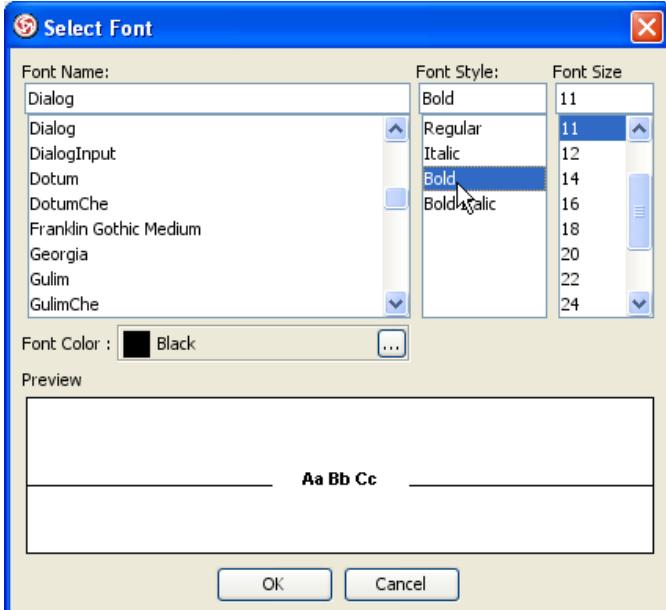


Figure 2-41 Select Font dialog box

Click **OK** in **Configure Requirements** dialog box to apply the changes.

After that, click the drop down button beside the **Requirement** button on the diagram toolbar of a requirement diagram, select **Ui requirement**.



Figure 2-42 Create Requirement of new type

Click on the diagram to create the shape, you can see the Requirement created has the attributes you defined, and the styles and formats are applied to the shape.

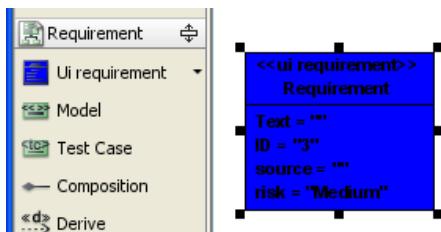


Figure 2-43 Requirement of new type created

Drawing CRC card diagram

Creating CRC card diagram

Select menu File > New Diagram > Requirements Capturing > CRC Card Diagram to create a CRC card diagram.

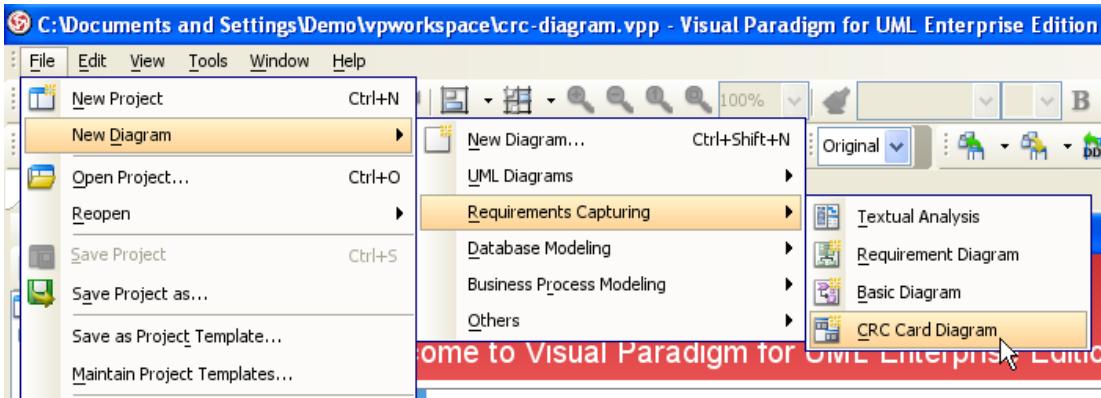


Figure 2-44 Create CRC card diagram

Creating CRC card

Click **CRC Card** on the diagram toolbar and then click on the diagram to create a CRC card. Name it *Shipment*.

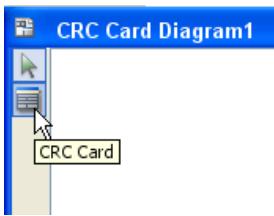


Figure 2-45 Create CRC card

Editing CRC card properties

Double-click **Description** to edit it and enter *Hold shipment information*.

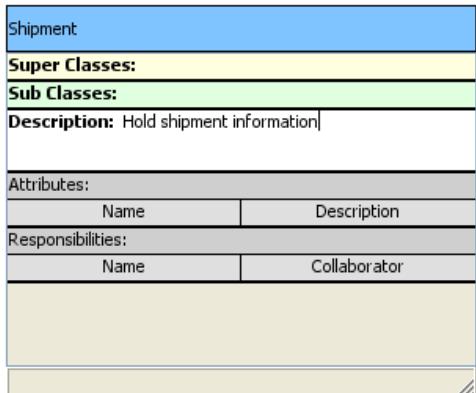


Figure 2-46 Edit description

Adding attributes

Right-click on the **Attributes** heading and select **Add > Attribute** from the popup menu.

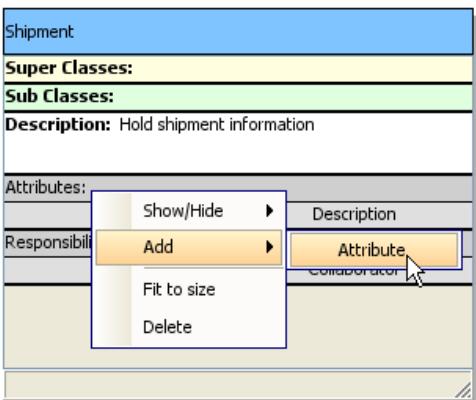


Figure 2-47 Add attribute

Edit attribute name to *ID* and description to *The id of shipment*.

Attributes:	
Name	Description
ID	The id of shipment
Responsibilities:	
Name	Collaborator

Figure 2-48 Attribute added

Continue to add other attributes.

Attributes:	
Name	Description
ID	The id of shipment
Customer ID	The id of customer
Size	The length, width, height of shipment
Weight	The weight of shipment
Delivery address	The delivery address

Figure 2-49 All attributes added

Adding responsibilities

Right-click on the **Responsibilities** heading and select **Add > Responsibility** from the popup menu.

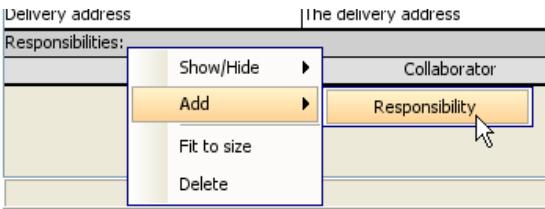


Figure 2-50 Add responsibility

Edit responsibility name to *get the size*.

Responsibilities:	
Name	Collaborator
get the size	Collaborator

Figure 2-51 Responsibility added

Continue to add other responsibilities.

Responsibilities:	Name	Collaborator
get the size		
get the weight		
get delivery address		

Figure 2-52 All responsibilities added

Managing requirements with requirements grid

Requirements Grid is a table with requirements listed in it. It lets you to access all requirements in a project or diagram, lookup requirements by criteria and create requirements.

Opening the requirements grid

To open requirements grid, select **Tools > Model Elements Grid > Open Requirements Grid** from the main menu.

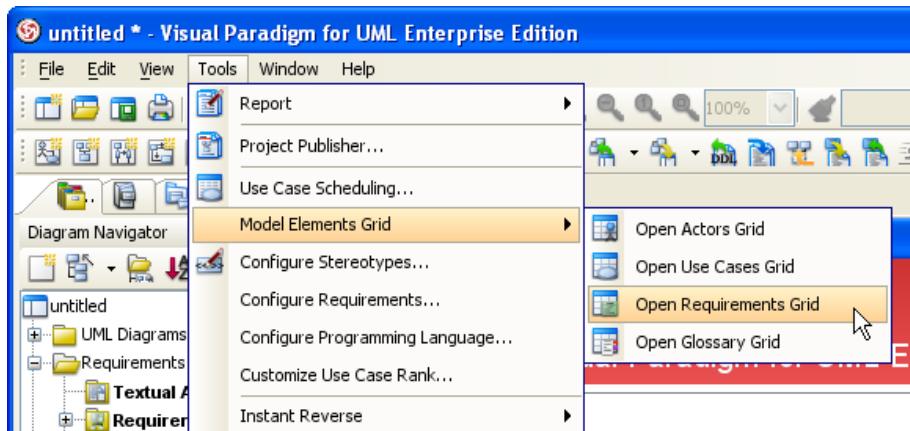


Figure 2-53 To open requirements grid

The requirements grid appear.

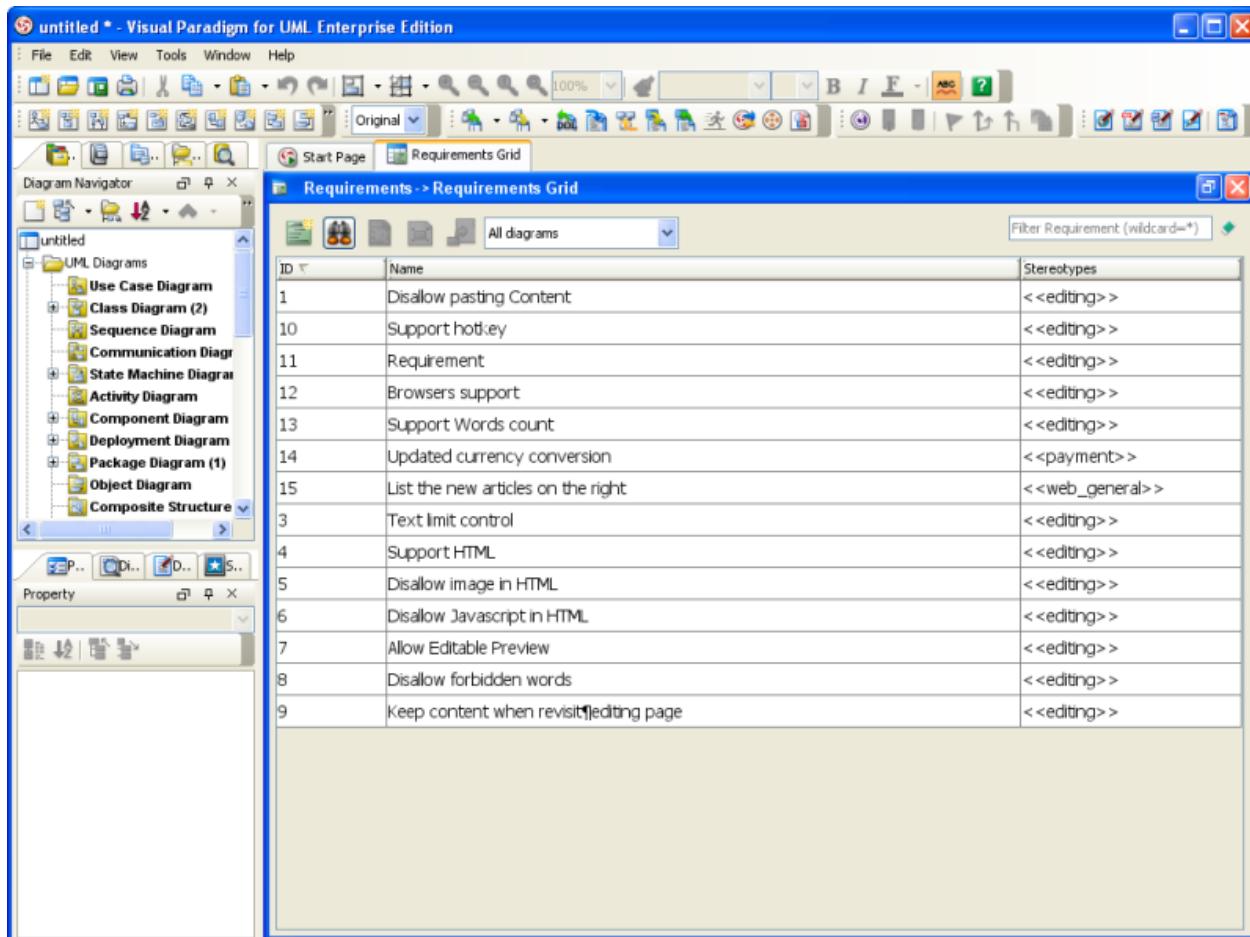


Figure 2-54 The requirements grid

Field	Description
	Create a new requirement.
	To find a requirement by providing its name or documentation.
	To open the specification of requirement selected in grid.

	To show the view(s) of requirement selected in grid.
	To visualize a requirement selected in grid.
All diagrams	To filter requirements by selecting the diagram(s) (or all diagrams) that contain the requirements.
Filter Requirement (wildcard=*)	To filter requirements by name. The rubber on the right hand side is for clearing the filter content.

Table 2-1 The fields in requirements grid

Creating requirement

- Click on above the grid.

Requirements -> Requirements Grid	
	All diagrams
ID	Name
1	Disallow pasting Content
10	Support hotkey

Figure 2-55 To create a requirement in grid

- Name the requirement. You may optionally reset the ID by double-clicking on the ID cell and entering a new one.

14	Updated currency conversion	<-
15	List the new articles on the right	<-
17	Support Spell Checking	
3	Text limit control	<-
4	Support HTML	<-

Figure 2-56 Name requirement

NOTE: The requirements created in requirements grid are automatically put under the Requirements model. You may move to another model through dragging and dropping in Model Explorer.

Visualizing a requirement

You can form a diagram with requirement, or show it in an existing diagram by visualizing it in requirements grid. To visualize a requirement:

- Select the requirement to visualize.

14	Updated currency conversion	.
15	List the new articles on the right	.
17	Support Spell Checking	
3	Text limit control	.
4	Support HTML	.

Figure 2-57 Selecting a requirement to visualize

- Click on above the grid.

Requirements -> Requirements Grid	
	All diagrams
ID	Name
1	Disallow pasting Content

Figure 2-58 Create textual analysis

3. If you want to visualize requirement in a new diagram, keep **Create new diagram** selected, select the type of diagram to create and specifying the diagram name.

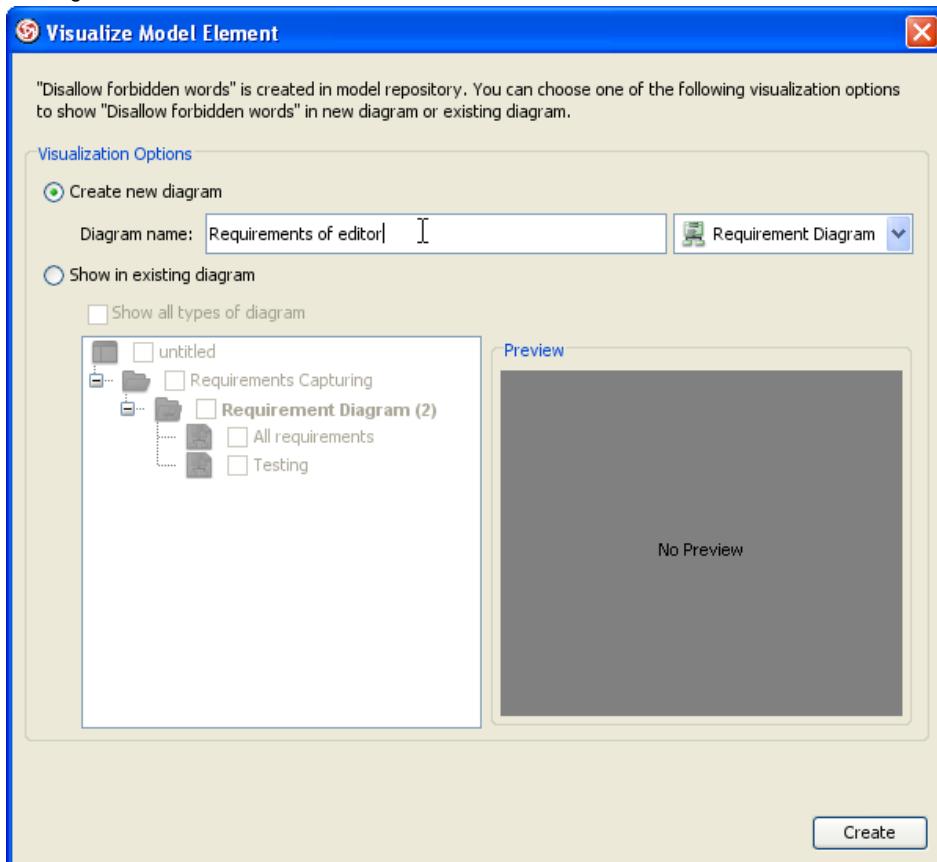


Figure 2-59 Name a new diagram

If you want to visualize requirement in an existing diagram, select **Show in existing diagram** and select a diagram in the diagram list.

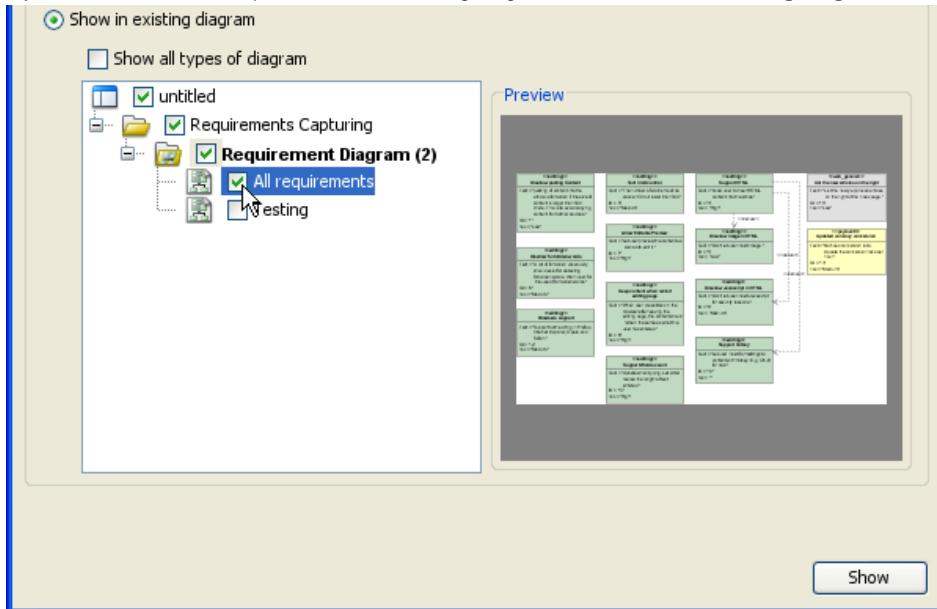


Figure 2-60 Select an existing requirement diagram to visualize the requirement

4. Click **Create** at the bottom of the dialog box.

Filter requirements

By filtering requirements, requirements that do not match the required naming convention are filtered. To filter, enter the name of requirement, or part of its name at the top right of grid. You can make use of the asterisk (*) character to represent any character(s).

The screenshot shows a software window titled "Requirements -> Requirements Grid". At the top, there is a toolbar with icons for file operations and a dropdown menu set to "All diagrams". To the right of the menu is a search bar containing the text "* HTML". The main area is a table with three columns: "ID", "Name", and "Stereotypes". There are three rows of data:

ID	Name	Stereotypes
4	Support HTML	<<editing>>
5	Disallow image in HTML	<<editing>>
6	Disallow Javascript in HTML	<<editing>>

Figure 2-61 To filter requirement by name

Find requirement

To find out requirements that match specific naming or documentation pattern:

1. Click above the grid.

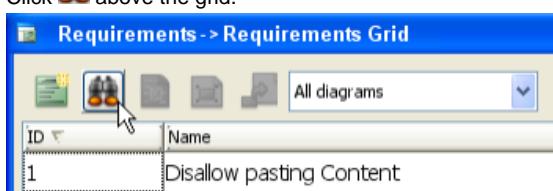


Figure 2-62 To find a requirement

2. In the **Search Text** text box, enter the text to search.

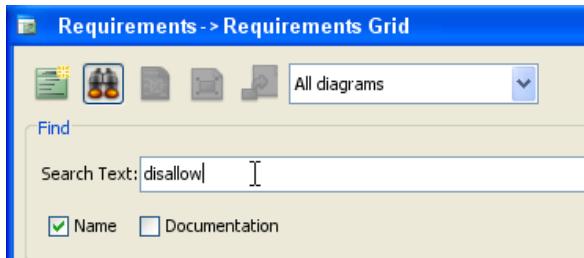


Figure 2-63 Enter requirement name

3. Specify whether to search the names and/or documentation of requirements.

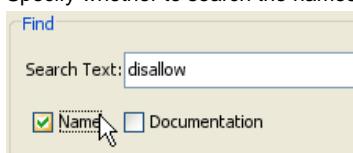


Figure 2-64 Select whether to search the name and/or documentation of requirements

4. Click **Find**. The grid is updated to highlight the matched requirements.

The screenshot shows a software interface titled "Requirements -> Requirements Grid". At the top, there are icons for file operations (New, Open, Save, Print) and a dropdown menu "All diagrams". To the right is a search bar with the placeholder "Filter Requirement (wildcard=*)" and a "Find" button. Below the search bar, there is a "Find" section with a "Search Text:" input field containing "Disallow", a "Name" checkbox (which is checked), and a "Documentation" checkbox (which is unchecked). A "Find" button is highlighted with a yellow border. The main area is a table with columns "ID", "Name", and "Stereotypes". The table contains 15 rows of requirement data. The rows are highlighted in blue, indicating they match the search term "Disallow". The requirements listed are: Disallow pasting Content, Support hotkey, Requirement, Browsers support, Support Words count, Updated currency conversion, List the new articles on the right, Support Spell Checking, Text limit control, Support HTML, Disallow image in HTML, Disallow Javascript in HTML, Allow Editable Preview, Disallow forbidden words, and Keep content when revisit editing page. The "Stereotypes" column for most rows shows "<<editing>>". The last row has a different background color.

ID	Name	Stereotypes
1	Disallow pasting Content	<<editing>>
10	Support hotkey	<<editing>>
11	Requirement	<<editing>>
12	Browsers support	<<editing>>
13	Support Words count	<<editing>>
14	Updated currency conversion	<<payment>>
15	List the new articles on the right	<<web_general>>
17	Support Spell Checking	
3	Text limit control	<<editing>>
4	Support HTML	<<editing>>
5	Disallow image in HTML	<<editing>>
6	Disallow Javascript in HTML	<<editing>>
7	Allow Editable Preview	<<editing>>
8	Disallow forbidden words	<<editing>>
9	Keep content when revisit editing page	<<editing>>

Figure 2-65 Execute the finding

Drawing activity diagrams

Creating activity diagram

Select menu **File > New Diagram > UML Diagrams > Activity Diagram** to create an activity diagram.

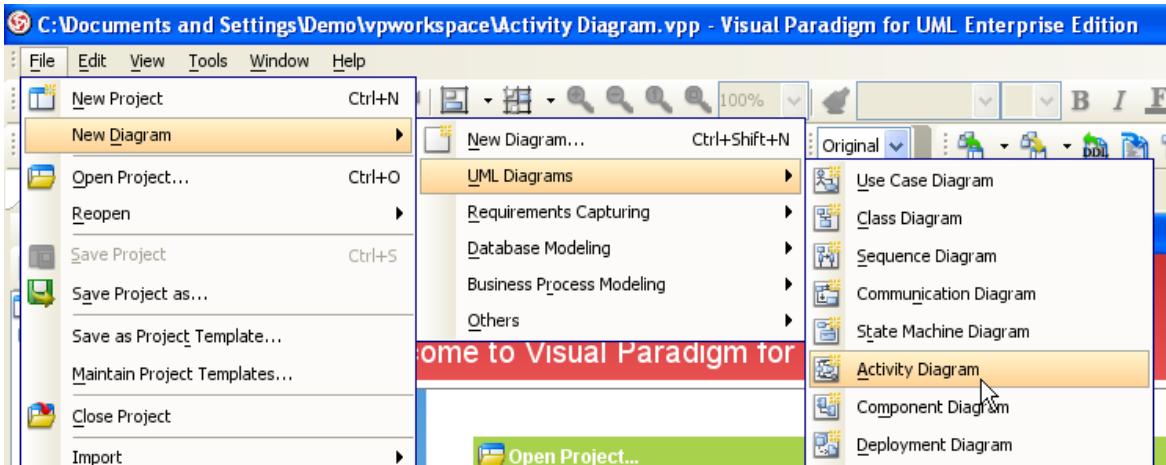


Figure 3-1 Create activity diagram

Creating swimlane

Let's create a vertical swimlane. Click the drop down button beside **Horizontal Swimlane** on the diagram toolbar and then select **Vertical Swimlane**.

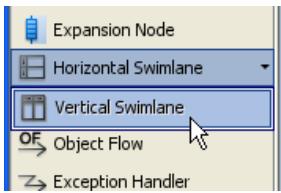


Figure 3-2 Create swimlane

Click on the diagram to create the swimlane.

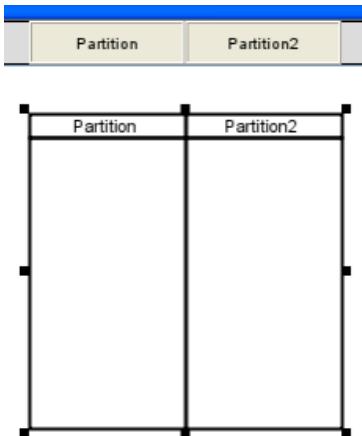


Figure 3-3 Swimlane created

Double-click the partition name to rename it.

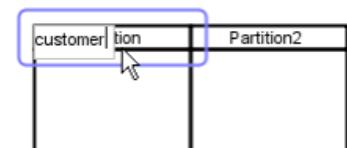


Figure 3-4 Rename partition

Inserting partition to swimlane

To insert partition to swimlane, right-click on a partition and select **Insert Partition After** from the popup menu.

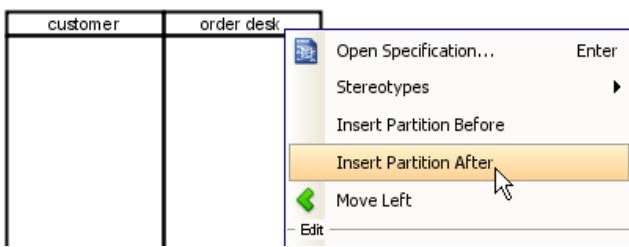


Figure 3-5 Insert partition to swimlane

A partition is inserted.



Figure 3-6 Partition inserted

Creating initial node

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

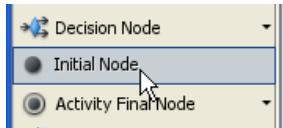


Figure 3-7 Create initial node

Click inside the partition to create the initial node there.

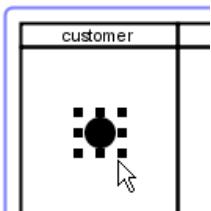


Figure 3-8 Initial node created

Creating action

Mouse over the initial node until its resources are visible. Click on the **Control Flow -> Action** resource and drag.

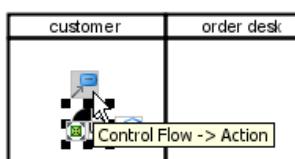


Figure 3-9 Create action

Move the mouse to where you want to place the action to, and then release the mouse button. An action is created and is connected to the initial node with a control flow.



Figure 3-10 Action created

Similarly you can create a new action using the **Control Flow -> Action** resource of an action.

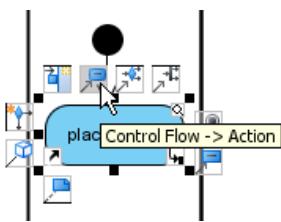


Figure 3-11 Create a new action from an action

A new action is created and is connected to the action with a control flow.

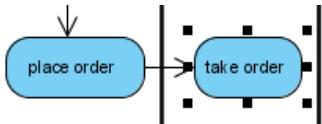


Figure 3-12 Action created

Continue to complete the activity diagram.

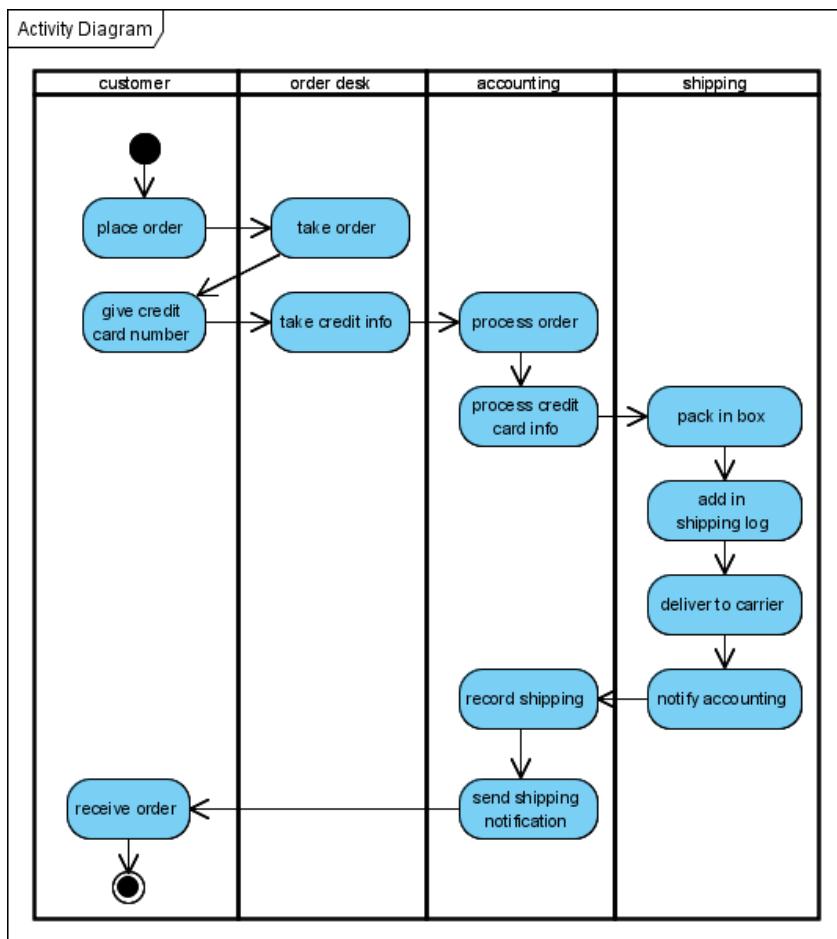


Figure 3-13 Completed activity diagram

Working with scenario

A scenario is a diagram formed by the internal interaction of a sequence of actions, 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.

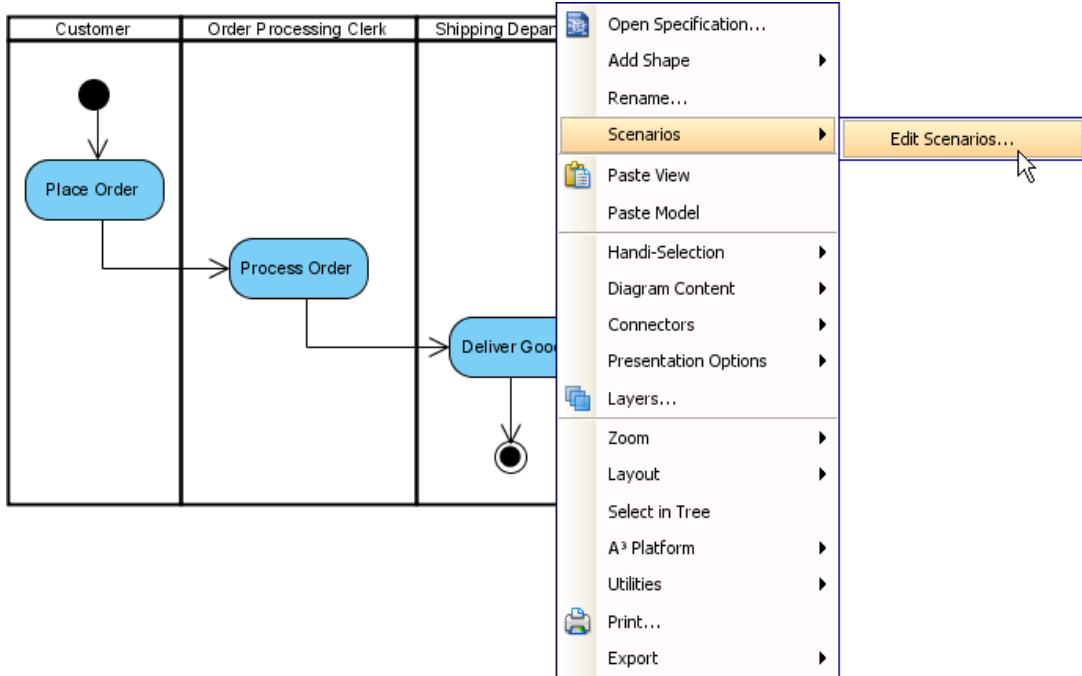


Figure 3-14 To edit scenario

2. In the **Edit Scenarios** dialog box, click Add... at the bottom left corner.

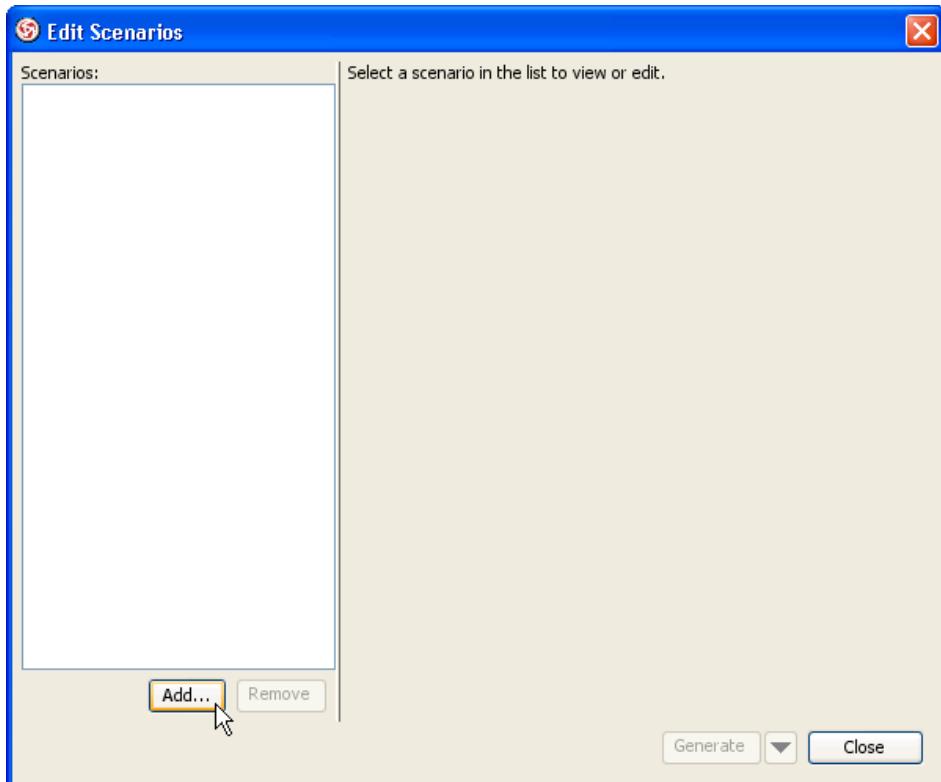


Figure 3-15 To add a scenario

3. Select a path for generating scenario. Click **OK** to confirm.

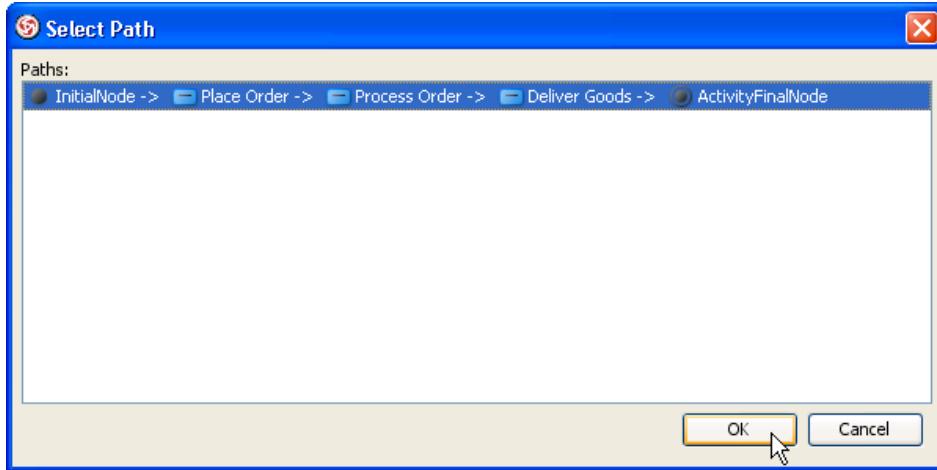


Figure 3-16 Select a path for generating scenario

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.

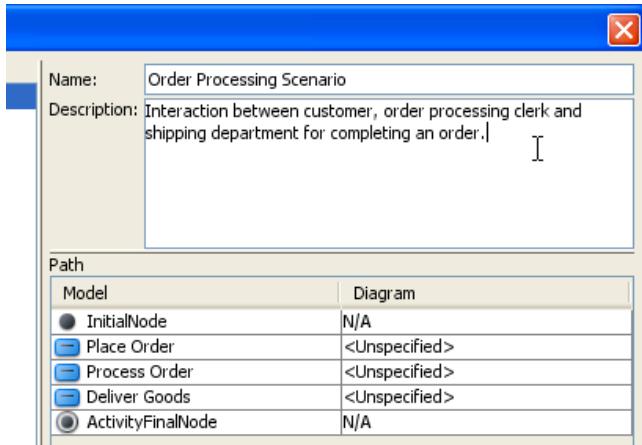


Figure 3-17 Name and describe scenario

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.

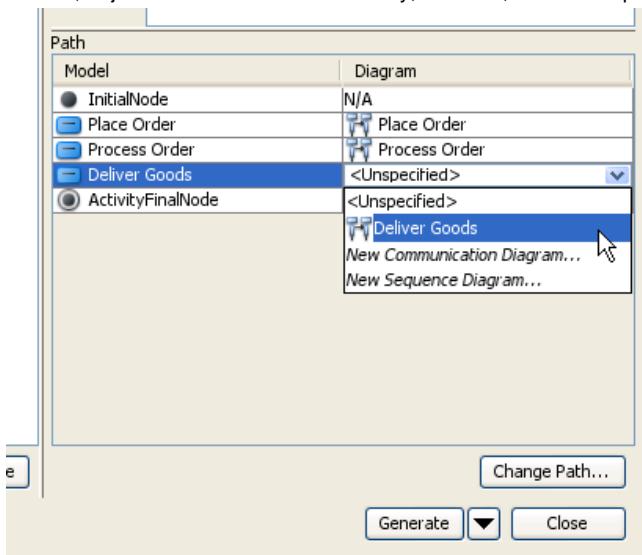


Figure 3-18 Select diagram for action

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

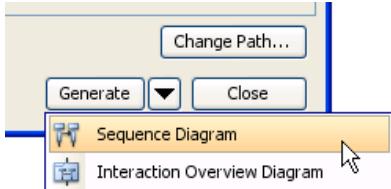


Figure 3-19 Generate scenario with specific diagram type

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.

Drawing state machine diagrams

Creating state machine diagram

Select menu **File > New Diagram > UML Diagrams > State Machine Diagram** to create a state machine diagram.

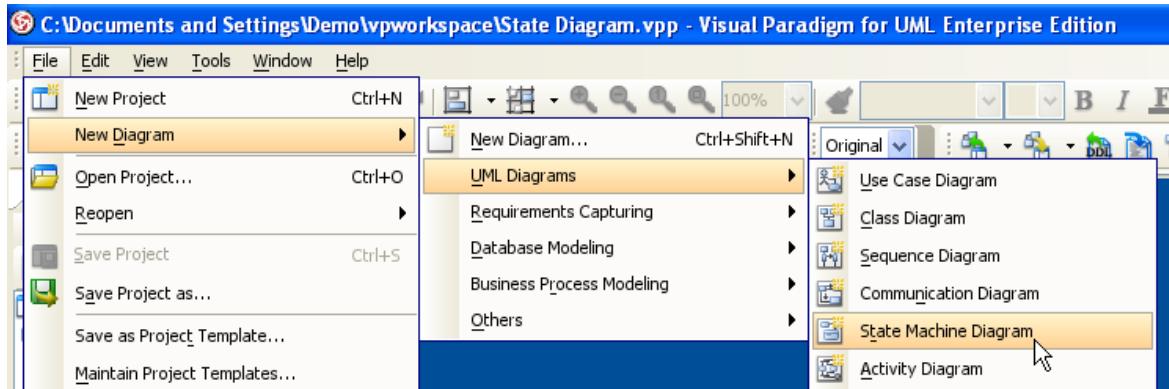


Figure 3-20 Create state machine diagram

Before we draw the diagram, let's change the connector style to curve. Right-click on the diagram and select **Connectors > Curve** from the popup menu.

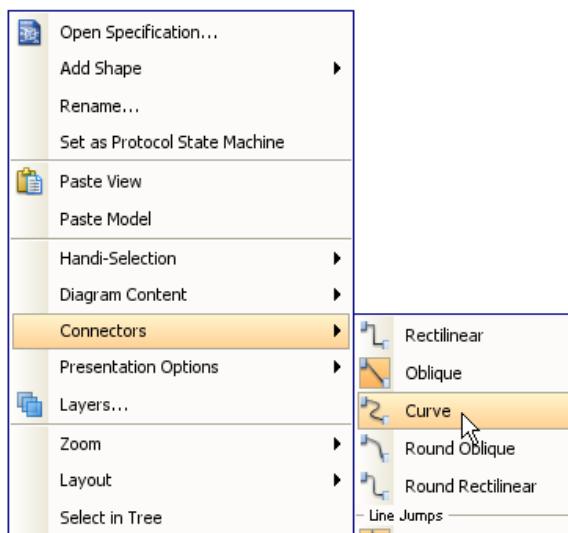


Figure 3-21 Selecting curve connector style

Creating states and transitions

Move the mouse over the initial pseudo state until its resources are visible. Click the **Transition -> State** resource and drag.



Figure 3-22 Create state from initial pseudo state

Move the mouse to where you want to place the state to, and then release the mouse button. A state is created and is connected to the initial pseudo state with a transition.

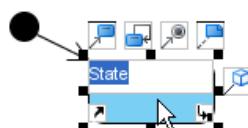


Figure 3-23 State and transition created

Drag the connector to make it curved.

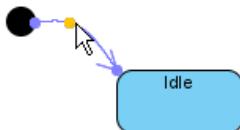


Figure 3-24 Drag the connector to make it curved

Similarly you can use the **Transition -> Final State** resource to create a final state.



Figure 3-25 Create final state

Final state does not show caption by default. To show it, right-click on the diagram and select **Presentation Options > Show Shape Caption > Final State**.

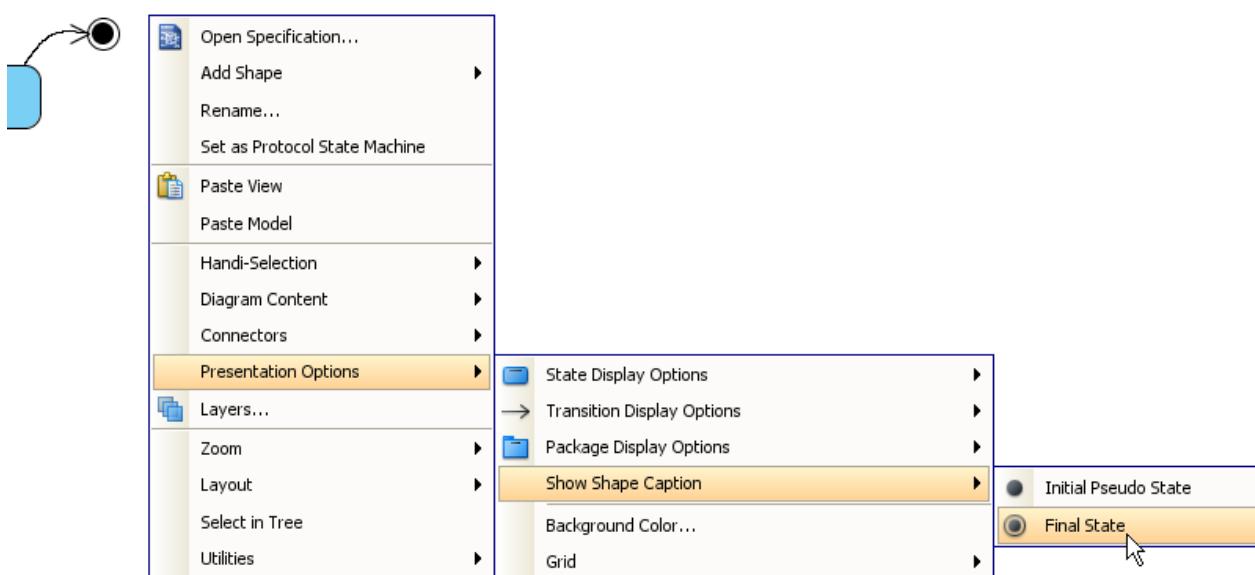


Figure 3-26 Show caption of final state

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.

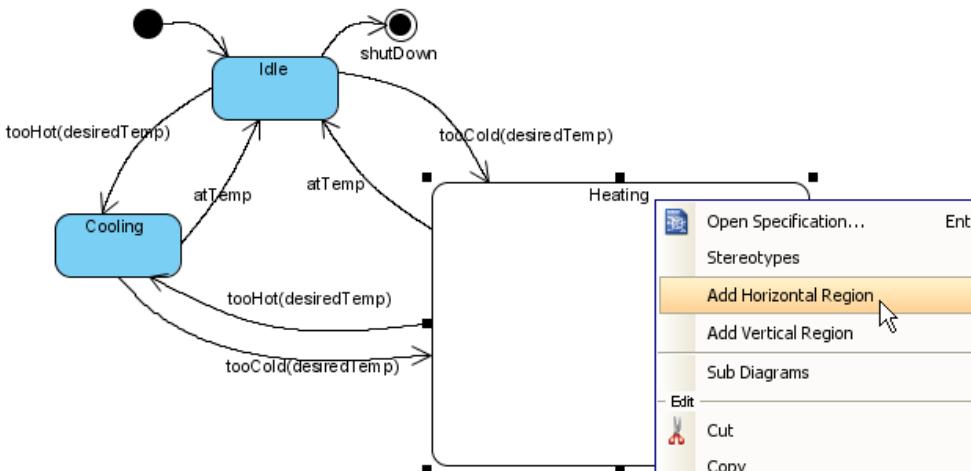


Figure 3-27 Add region to state

Then you can draw the substates inside the region.

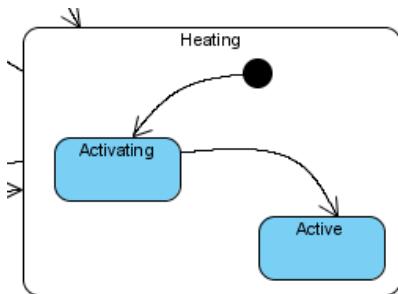


Figure 3-28 Substates in a composite state

Modeling properties of transition

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



Figure 3-29 Open specification of transition

The **Transition Specification** dialog box appears. Here you can edit its name, effect, guard, etc. Let's click **Edit...** of the **Effect** property.

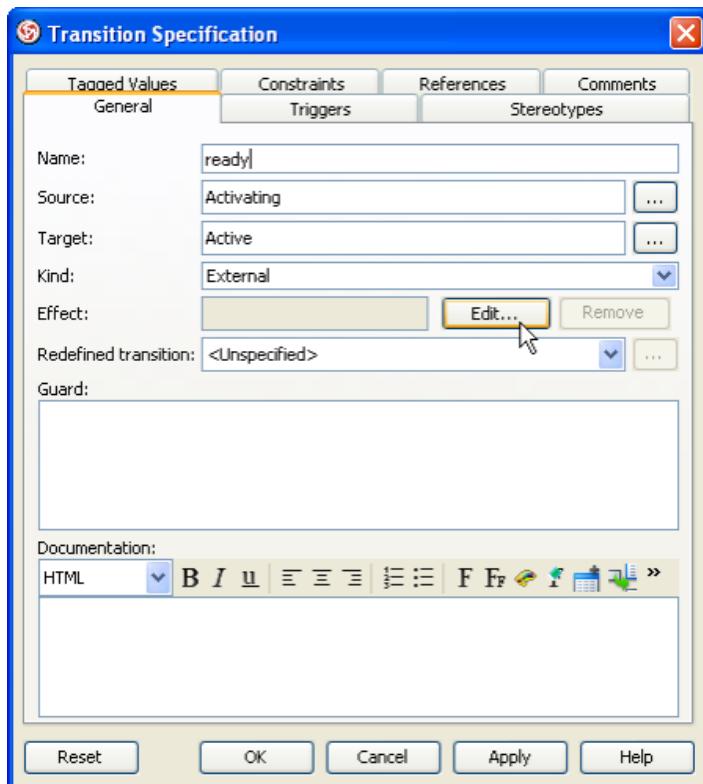


Figure 3-30 Transition Specification dialog box

The Activity Specification (Effect) dialog box appears. Let's change its name, and then click **OK** to apply the change.

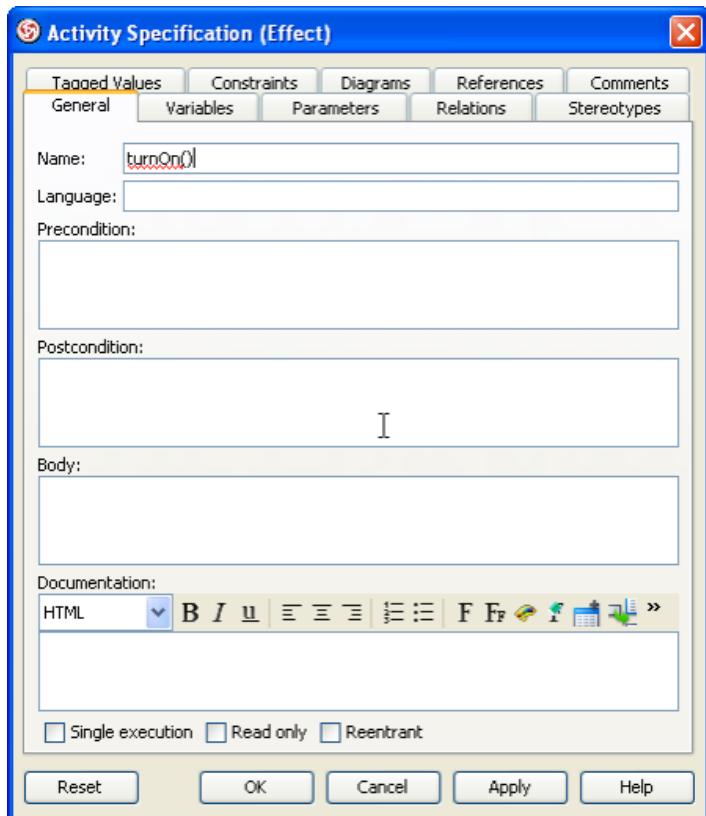


Figure 3-31 Edit effect property of transition

Click **OK** in the Transition Specification dialog box to close it. The name (ready) and effect (turnOn()) are shown on the transition caption.

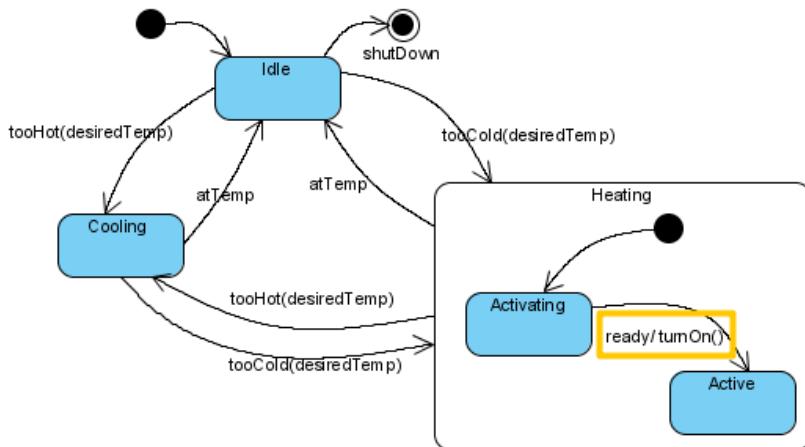


Figure 3-32 Name and effect shown in caption of transition

Drawing timing diagrams

Creating timing diagram

Select menu **File > New Diagram > UML Diagrams > Timing Diagram** to create a timing diagram.

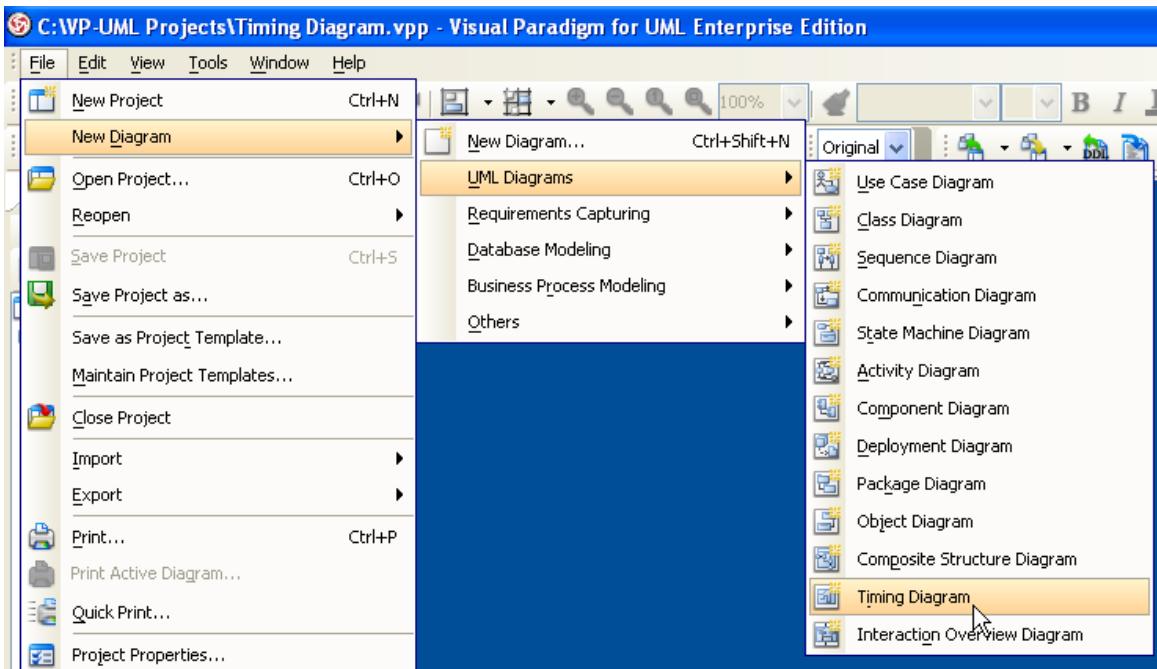


Figure 3-33 Create timing diagram

Creating timing frame

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

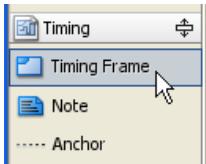


Figure 3-34 Create timing frame

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

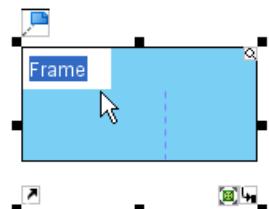


Figure 3-35 Rename frame

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

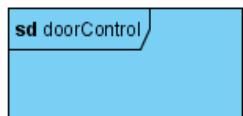


Figure 3-36
Frame renamed

Adding lifeline to frame

To add lifeline to frame, right-click the frame and select **Add Lifeline** from the popup menu.

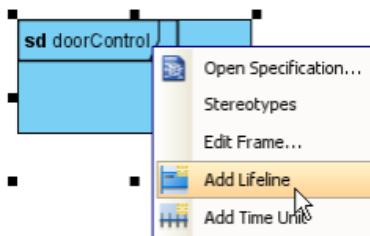


Figure 3-37 Add lifeline

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

Adding time unit to frame

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

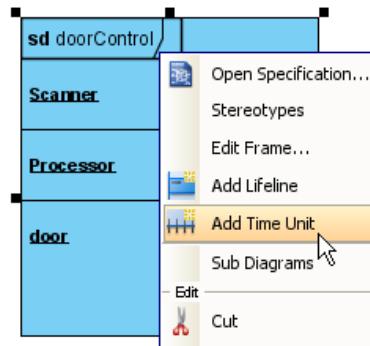


Figure 3-38 Add time unit

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

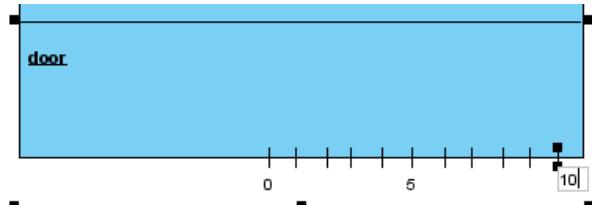


Figure 3-39 Rename time unit

Adding state/condition to lifeline

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

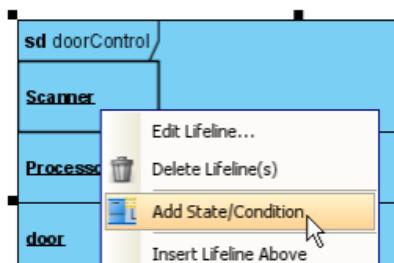


Figure 3-40 Add state/condition

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

Dragging time instance

Mouse over the line segment of a time instance, click and drag it.

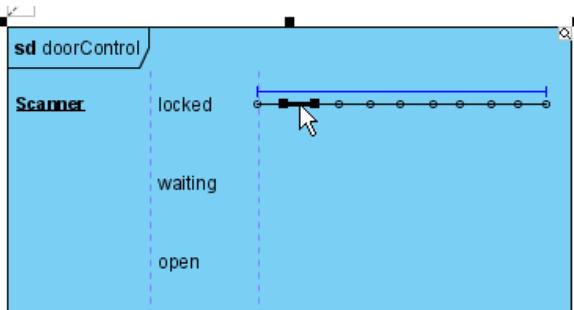


Figure 3-41 Drag time instance

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

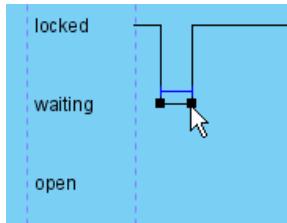


Figure 3-42 Dragged time instance

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.

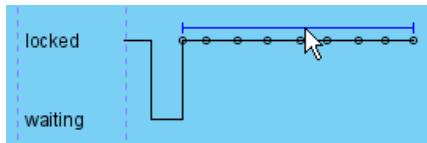


Figure 3-43 Move a group of time instances

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

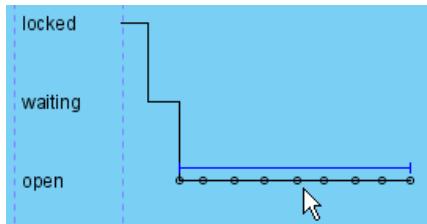


Figure 3-44 Moved group of time instances

Adding time messages to frame

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



Figure 3-45 Edit frame

The Edit Frame dialog box appears. Select the **Time Messages** tab and click **Add....**

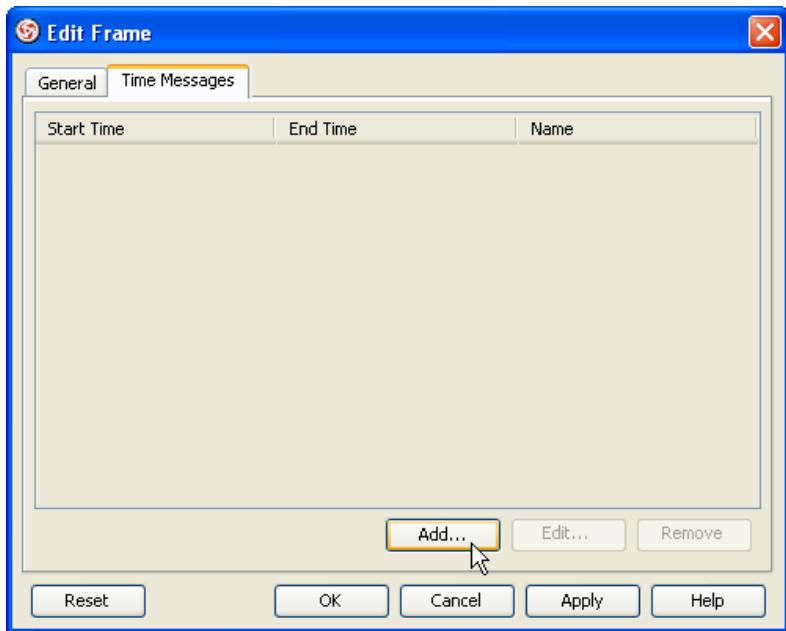


Figure 3-46 Add time message

The **Add Time Message** dialog box appears. 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.

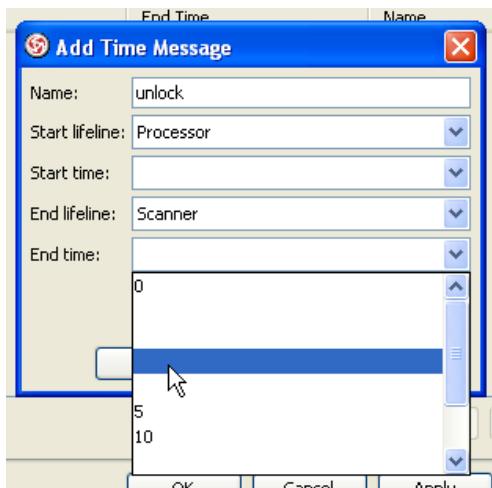


Figure 3-47 Select end time of time message

The time message is shown on the frame.

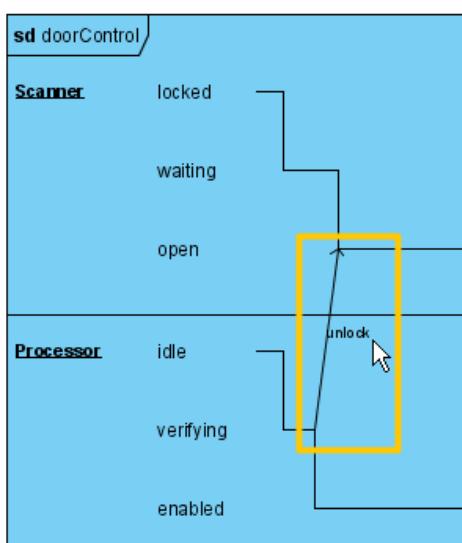


Figure 3-48 Time message

Switching to compact view mode

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

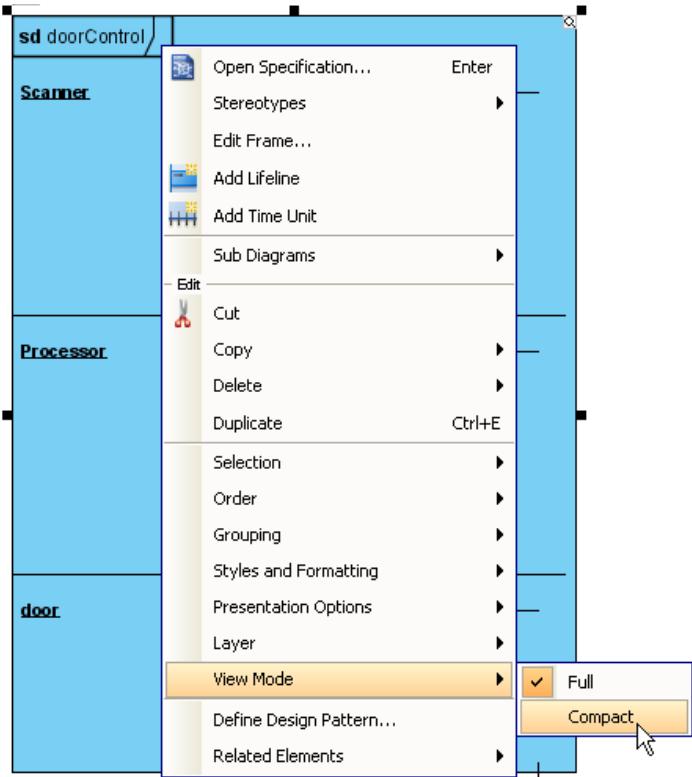


Figure 3-49 Switch to compact view mode

The frame will be shown in compact mode.

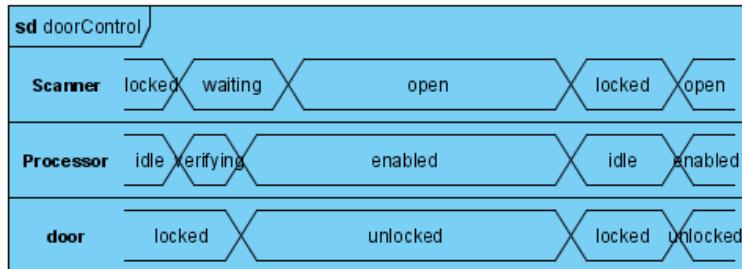


Figure 3-50 Frame shown in compact mode

Drawing sequence diagrams

Creating sequence diagram

Select menu **File > New Diagram > UML Diagrams > Sequence Diagram** to create a sequence diagram.

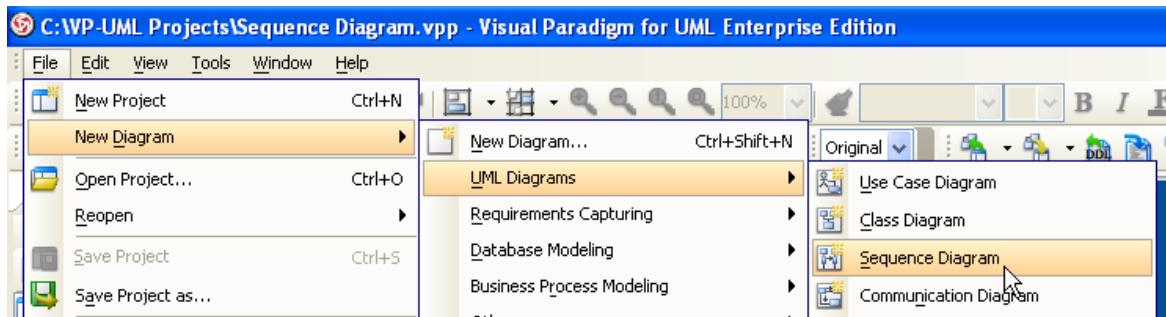


Figure 4-1 Create sequence diagram

Creating actor

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

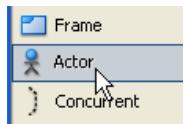


Figure 4-2 Create actor

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 the resource-centric interface. Click on the **Message -> LifeLine** resource beside an actor/lifeline and drag.

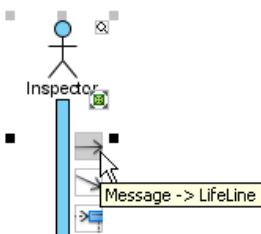


Figure 4-3 Create lifeline

Move the mouse to empty space of the diagram and then release the mouse button. A new lifeline will be created and connected to the actor/lifeline with a message.

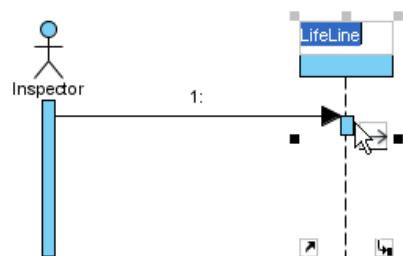


Figure 4-4 Lifeline and message created

Auto extending activation

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

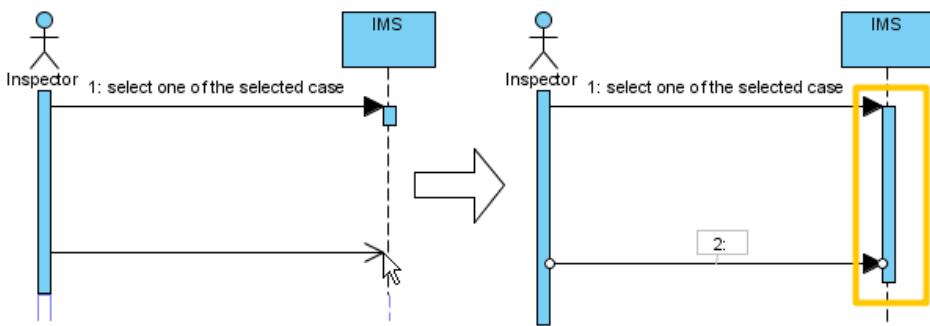


Figure 4-5 Auto extending activation

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 **Sweeper** on the diagram toolbar (under the **Tools** category).

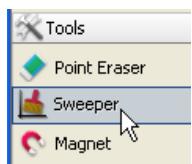


Figure 4-6
Sweeper

Click on empty space of the diagram and drag towards either 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.

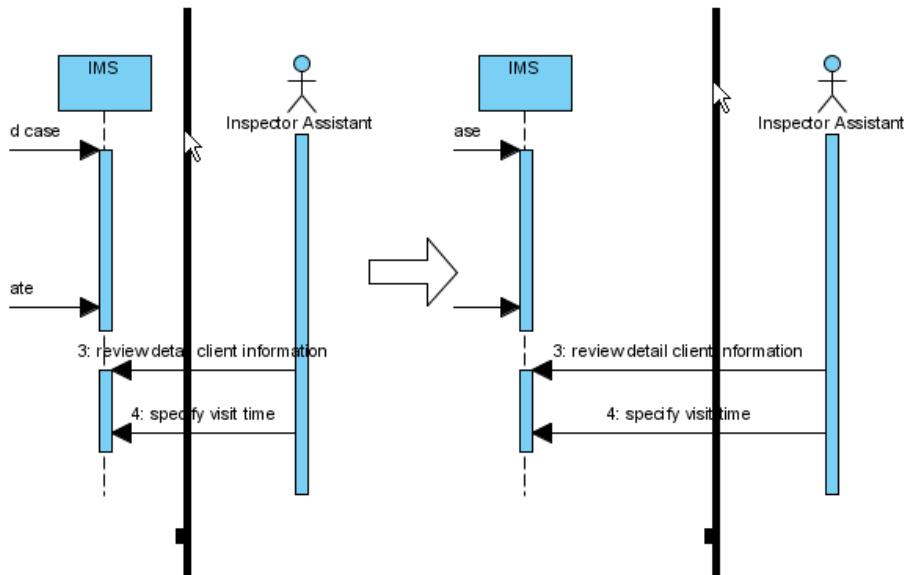


Figure 4-7 Sweep towards right

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

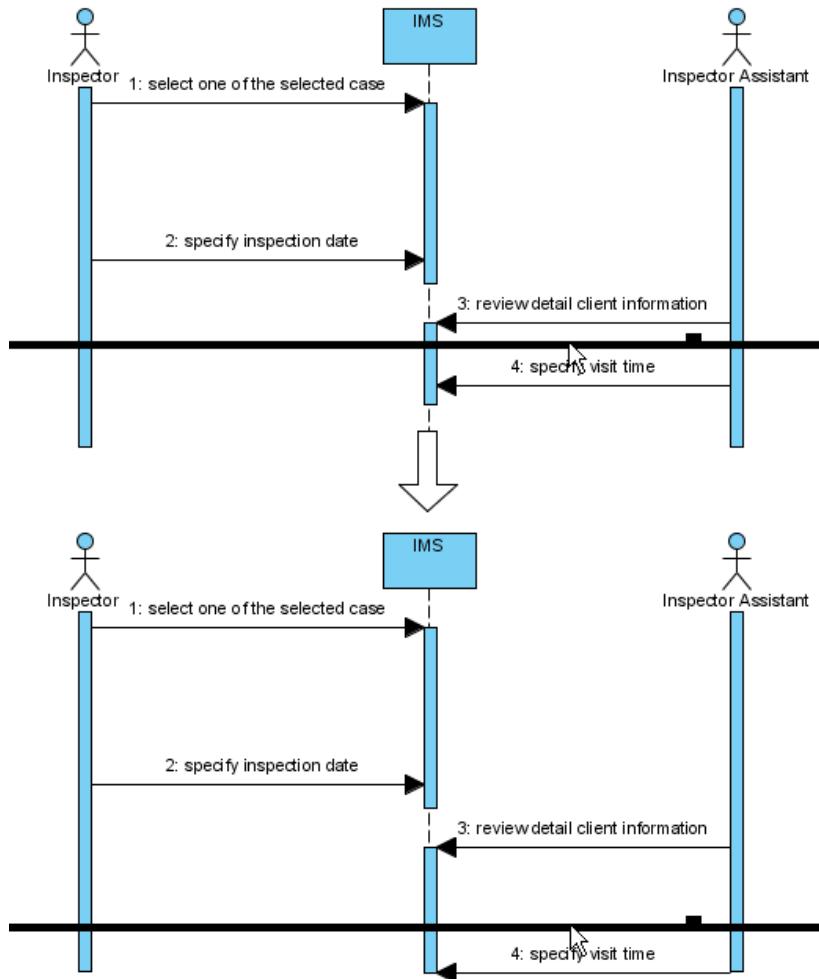


Figure 4-8 Sweep downwards

You can also use magnet to pull shapes together. To use magnet, click **Magnet** on the diagram toolbar (under the **Tools** category).

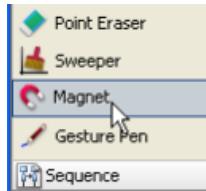


Figure 4-9 Magnet

Click on empty space of the diagram and drag towards either 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.

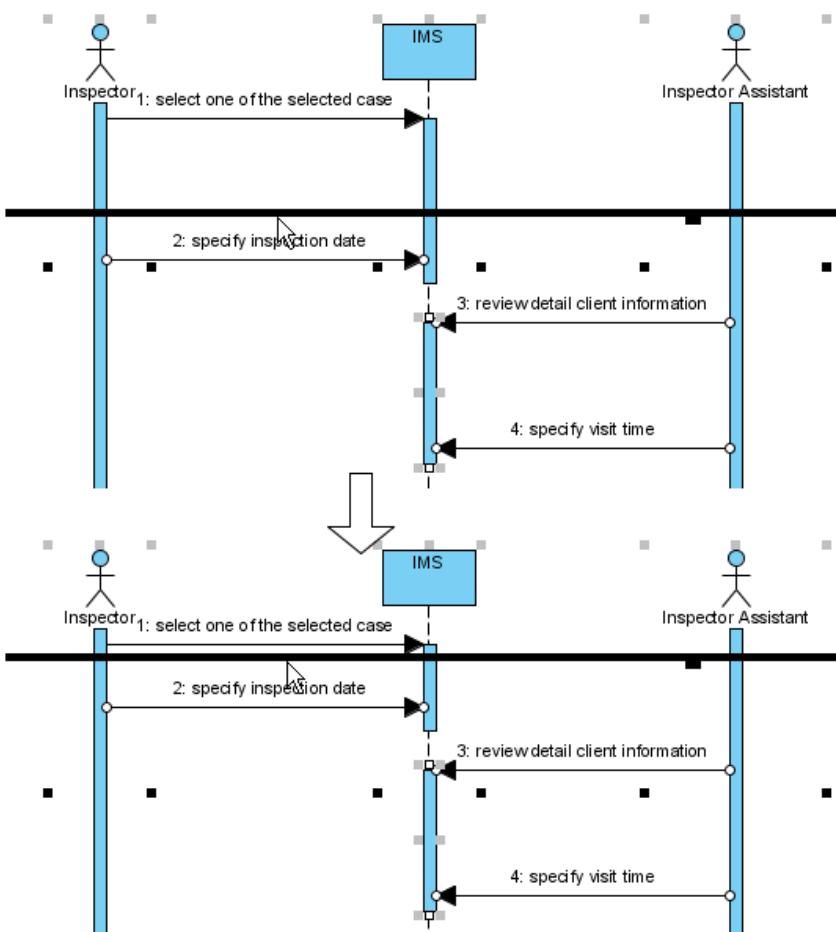


Figure 4-10 Pull shapes upwards using magnet

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.

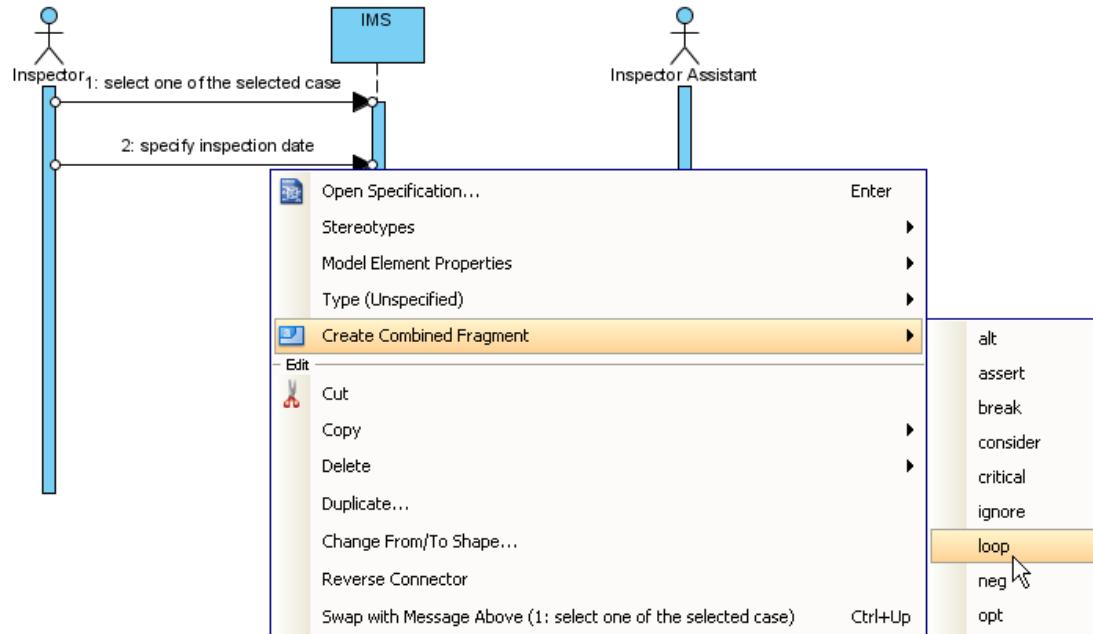


Figure 4-11 Create combined fragment for messages

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

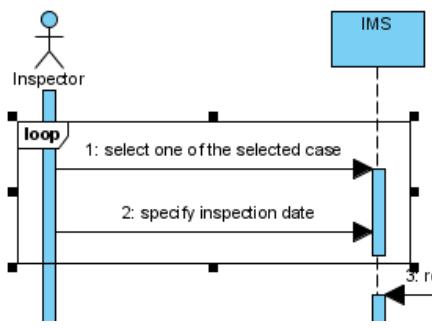


Figure 4-12 Combined fragment created

Developing sequence diagram with quick editor or keyboard shortcuts

In sequence diagram, there is, by default, an editor appear at the bottom of diagram, 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.

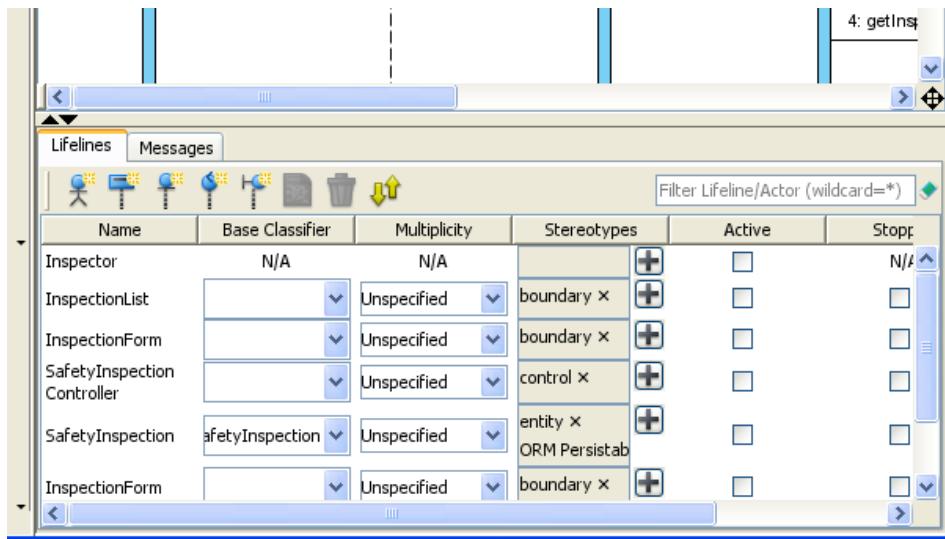


Figure 4-13 The quick editor

Editing lifelines

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

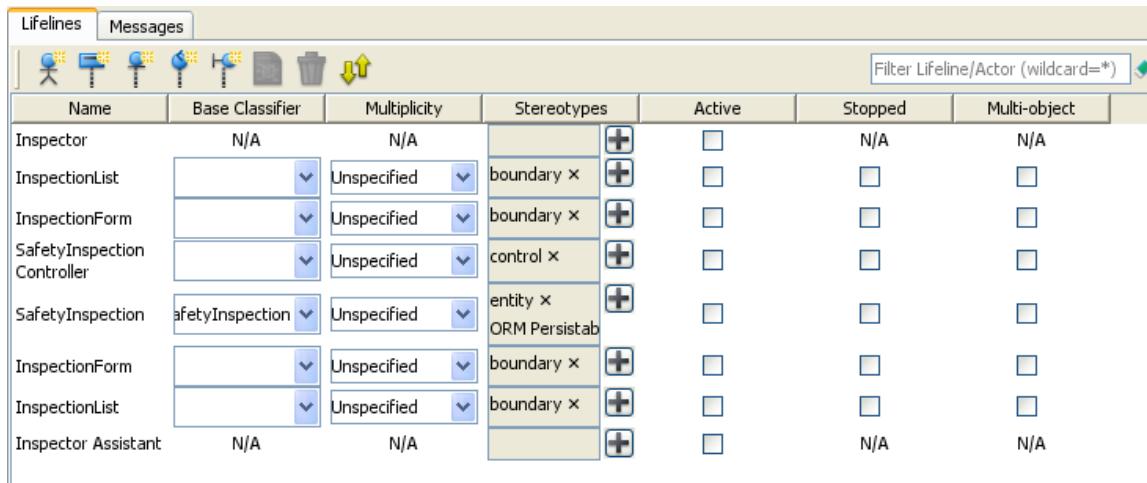


Figure 4-14 Lifelines pane in quick editor

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

Table 4-1 Buttons in Lifelines pane

Editing messages

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

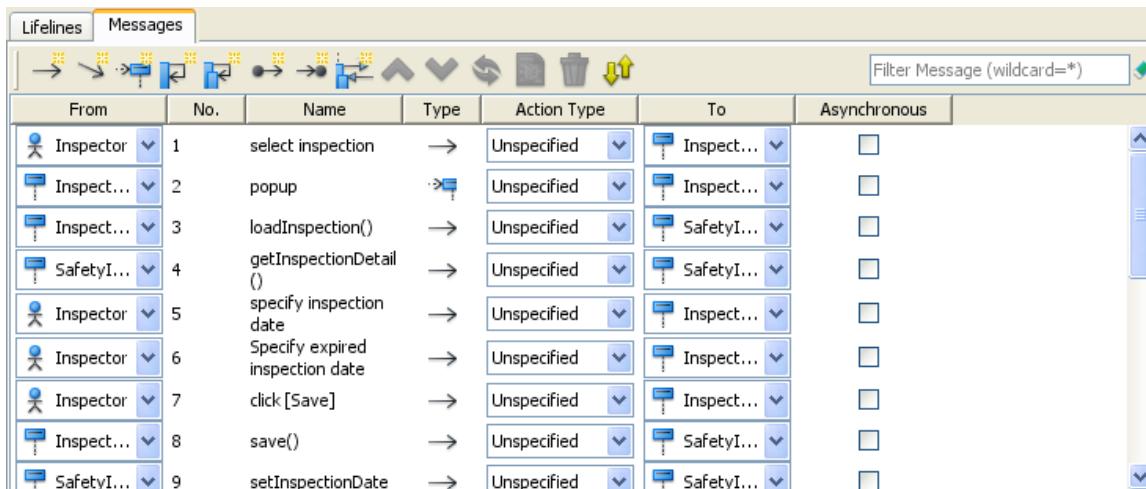


Figure 4-15 Messages pane in quick editor

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 a actor/lifeline in diagram
	Alt-Shift-R	To create a recursive message on a 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 diagram element to be selected when selecting a message in editor, and vice versa

Table 4-2 Buttons in Messages pane

Expanding and collapsing the editor

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



Figure 4-16 To collapse the quick editor

Drawing communication diagrams

Creating communication diagram

Select menu **File > New Diagram > UML Diagrams > Communication Diagram** to create a communication diagram.

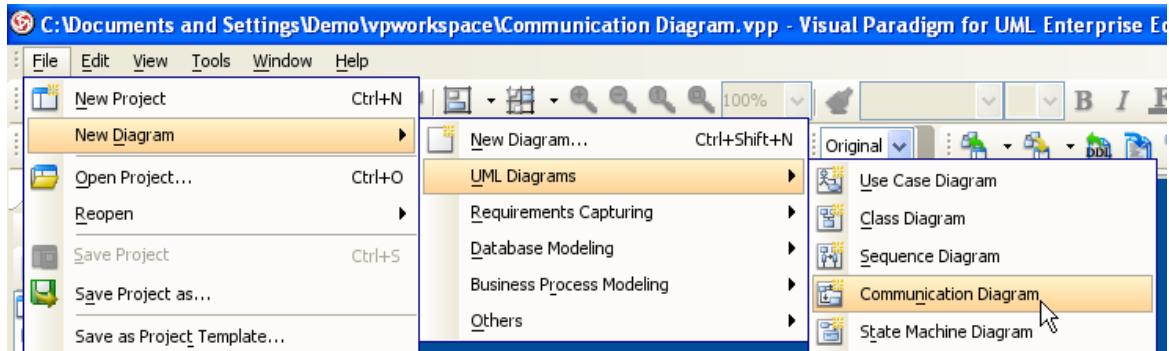


Figure 4-17 Create communication diagram

Creating actor

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

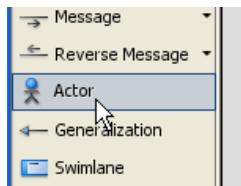


Figure 4-18 Create actor

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 the resource-centric interface. Click on the **Message -> LifeLine** resource beside an actor/lifeline and drag.



Figure 4-19 Create lifeline

Move the mouse to empty space of the diagram and then release the mouse button. A new lifeline will be created and connected to the actor/lifeline with a link (the line) and a message (the arrow).

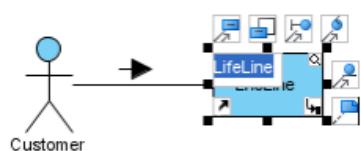


Figure 4-20 Lifeline, link and message created

Creating message on link

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



Figure 4-21 Create message on link

A message will be created on the link.

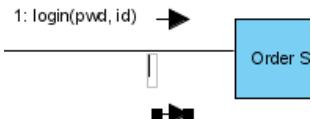


Figure 4-22 Message created on link

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 popup menu.

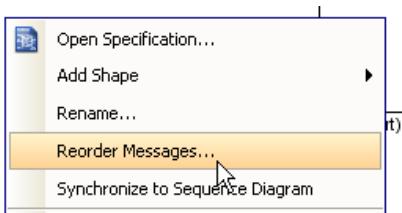


Figure 4-23 Reorder messages

The **Communication Diagram Specification** dialog box appears with the **Message** tab selected. Double-click on the **Sequence #** cell of a message to edit it. Click **OK** to apply the changes.

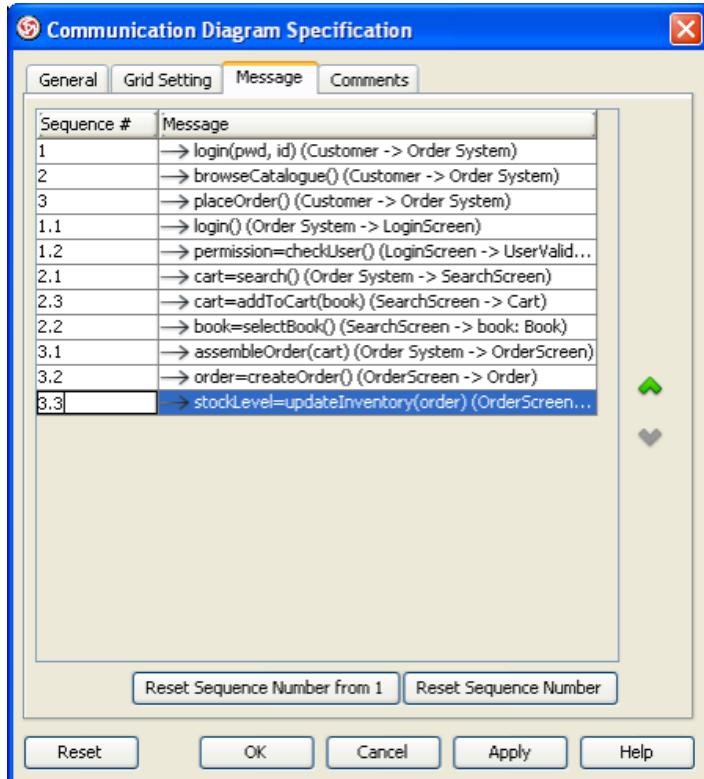


Figure 4-24 Edit sequence number of messages

The sequence number of messages are updated.

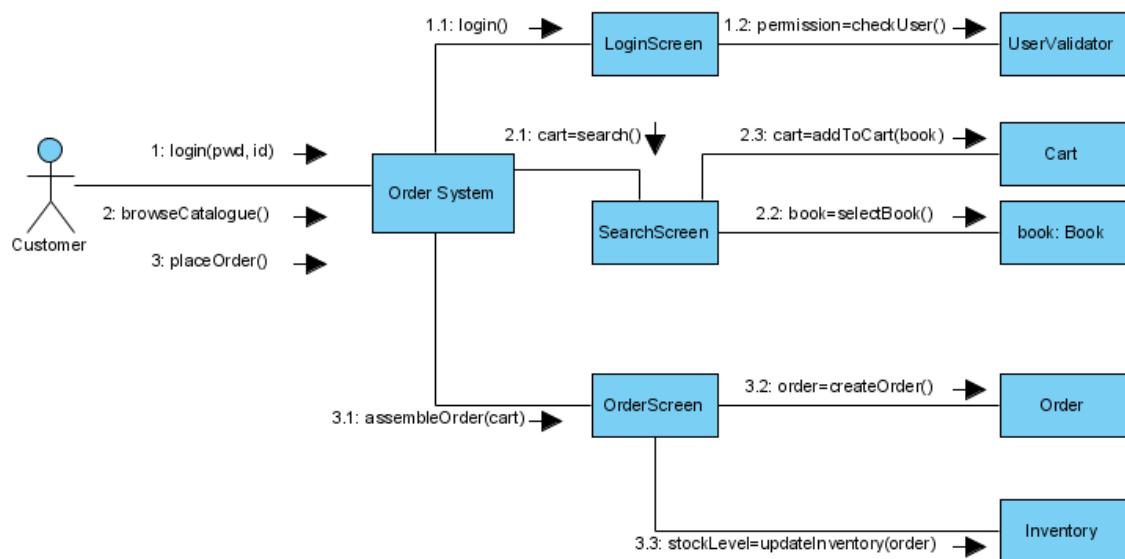


Figure 4-25 Sequence number of messages updated

Drawing interaction overview diagrams

Creating interaction overview diagram

Select menu **File > New Diagram > UML Diagrams > Interaction Overview Diagram** to create an interaction overview diagram.

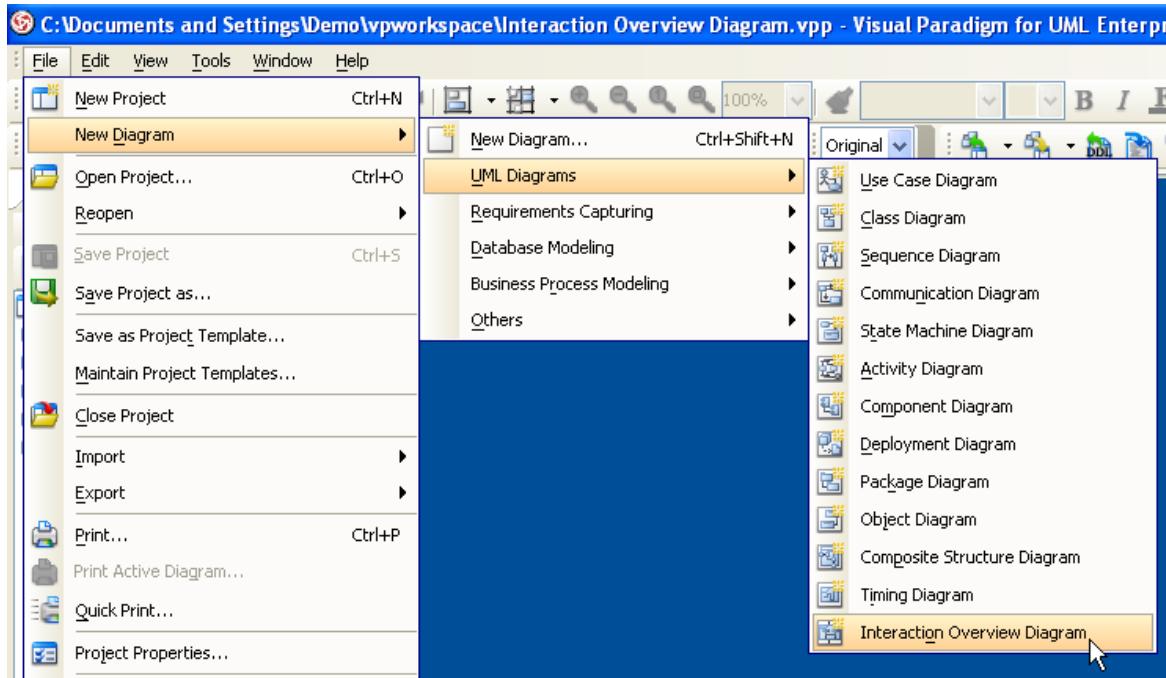


Figure 4-26 Create interaction overview diagram

Creating initial node

To create initial node, click **Initial Node** on the diagram toolbar and then click on the diagram.

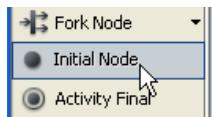


Figure 4-27 Create initial node

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 popup menu.

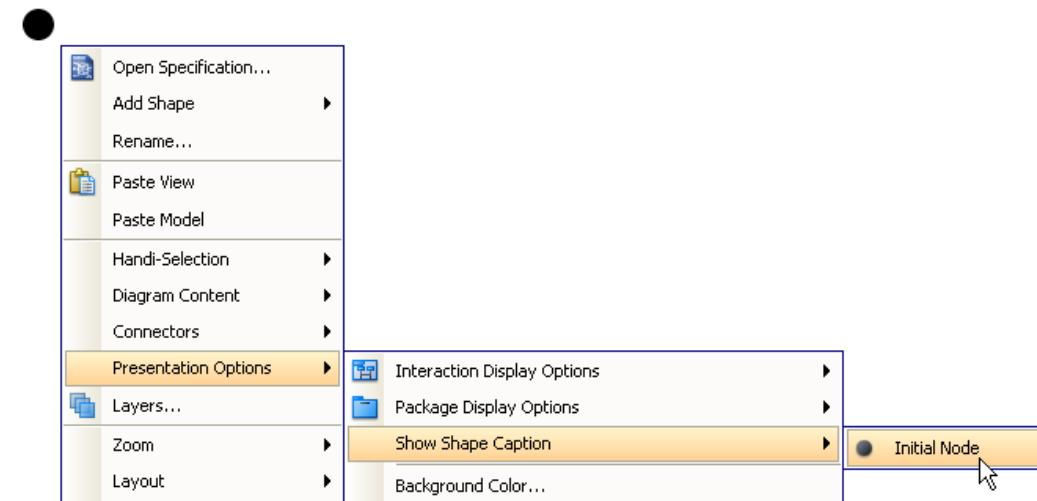


Figure 4-28 Show caption of initial node

Creating decision node

To create a decision node from an initial node, click on the **Generic Resource** beside it and drag.



Figure 4-29 Generic resource

Move the mouse to empty space of the diagram and then release the mouse button. When a popup menu shows, select **Control Flow -> Decision Node**.

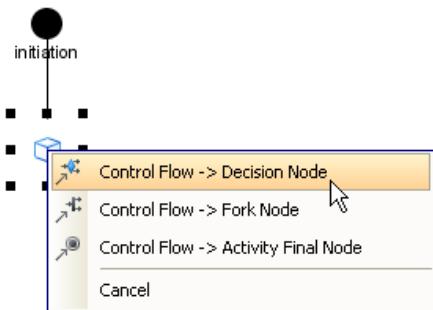


Figure 4-30 Create decision node

Right-click the diagram and select **Presentation Options > Show Shape Caption > Decision Node** from the popup menu to show caption of decision node.

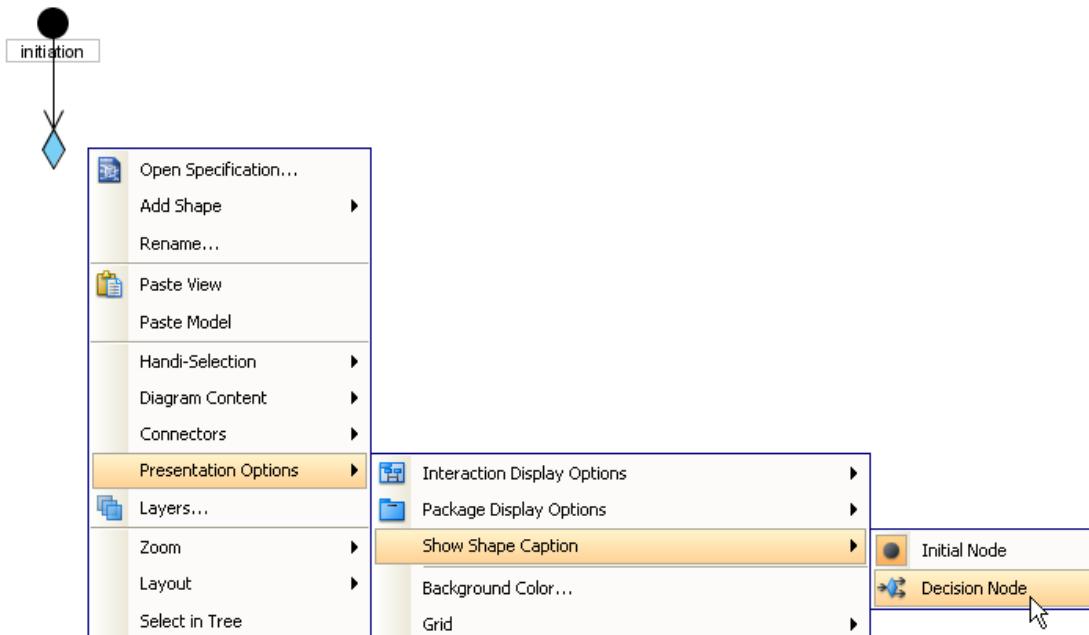


Figure 4-31 Show caption of decision node

Creating interaction use

To create an interaction use from a decision node, click on the **Control Flow -> Interaction Use** resource beside it and drag.

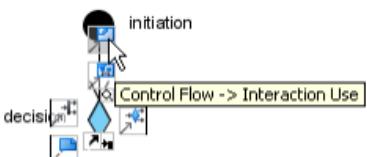


Figure 4-32 Create interaction use

Move the mouse to empty space of the diagram and then release the mouse button. An interaction use is created and connected to the decision node with a control flow.

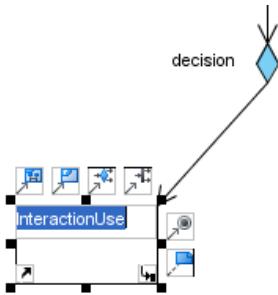


Figure 4-33 Interaction use and control flow created

Make the interaction use refers to a diagram by right-clicking on it and select **Refers to > New Sequence Diagram**.

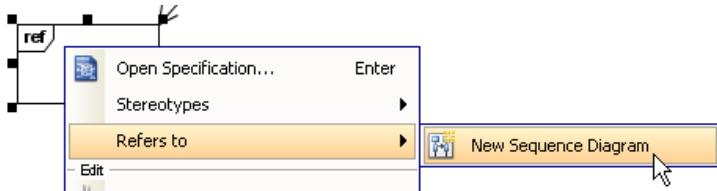


Figure 4-34 Make interaction use refers to diagram

When sequence diagram is created, rename the diagram.



Figure 4-35 Rename sequence diagram

Return to the interaction overview diagram, the interaction use caption now reflects the diagram it refers to.

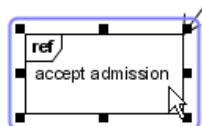


Figure 4-36 Interaction use caption updated

Creating fork node

To create a fork node from an interaction use, click on the **Control Flow -> Fork Node** resource beside it and drag.

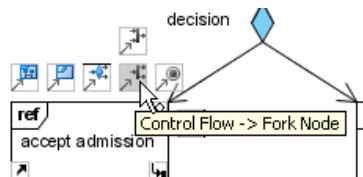


Figure 4-37 Create fork node

Move the mouse to empty space of the diagram and then release the mouse button. A fork node is created and connected to the interaction use with a control flow.

The fork node created is vertical by default, to change it to horizontal, right-click on the fork node and select **Orientation > Horizontal** from the popup menu.

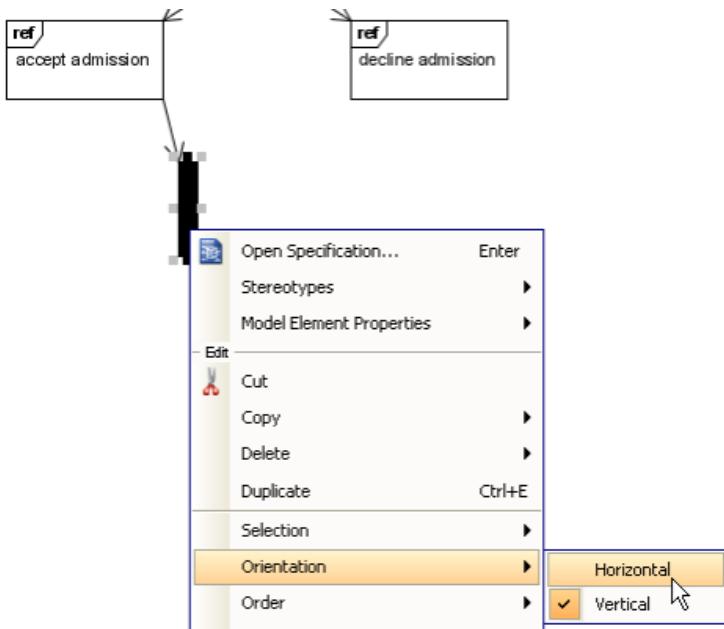


Figure 4-38 Change orientation of fork node

Right-click the diagram and select **Presentation Options > Show Shape Caption > Fork Node** from the popup menu to show caption of fork node.

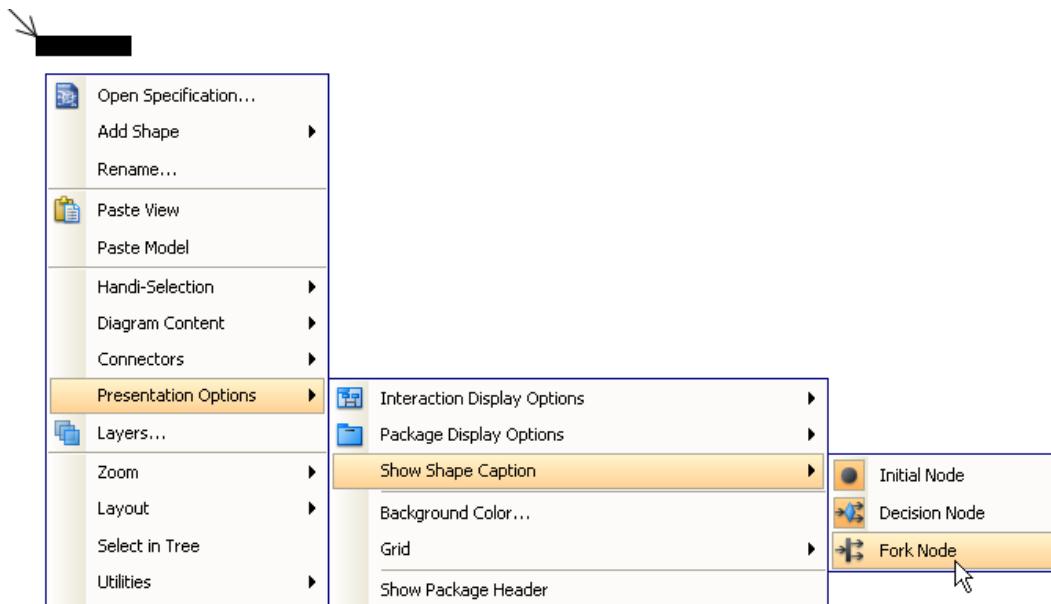


Figure 4-39 Show caption of fork node

Creating interaction

To create an interaction from a fork node, click on the **Control Flow -> Interaction** resource beside it and drag.

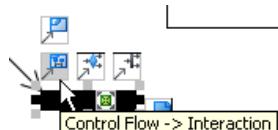


Figure 4-40 Create interaction

Move the mouse to empty space of the diagram and then release the mouse button. An interaction is created and connected to the fork node with a control flow.

A new sequence diagram is created and associated with an interaction by default. To open it, right-click on the interaction and select **Associated Diagram** > <diagram name>.

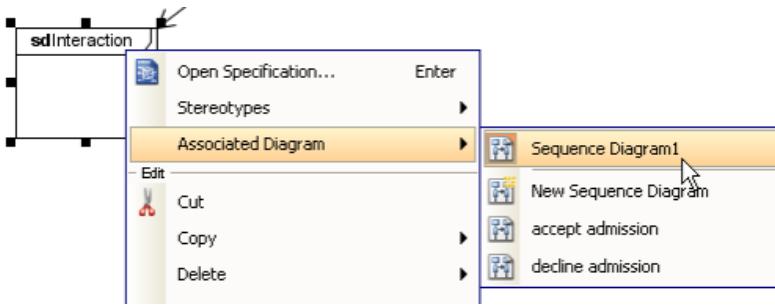


Figure 4-41 Open associated diagram of interaction

Draw the sequence diagram.

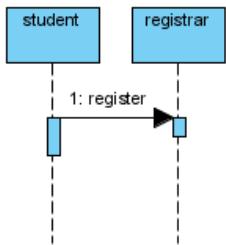


Figure 4-42
Sequence diagram

Return to the interaction overview diagram, the interaction now shows the thumbnail of the sequence diagram.

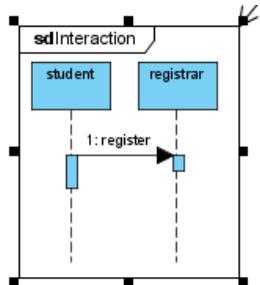


Figure 4-43 Updated diagram thumbnail in interaction

Creating join node

To create a join node from an interaction, click on the **Control Flow -> Join Node** resource beside it and drag.

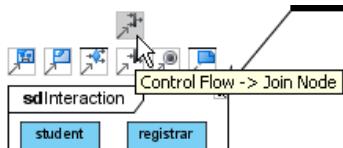


Figure 4-44 Create join node

Move the mouse to empty space of the diagram and then release the mouse button. A join node is created and connected to the interaction with a control flow.

The join node created is vertical by default, to change it to horizontal, right-click on the join node and select **Orientation > Horizontal** from the popup menu.

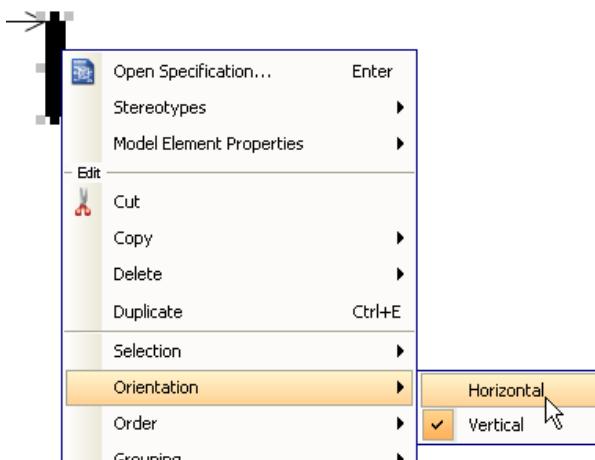


Figure 4-45 Change orientation of join node

Right-click the diagram and select **Presentation Options > Show Shape Caption > Join Node** from the popup menu to show caption of join node.

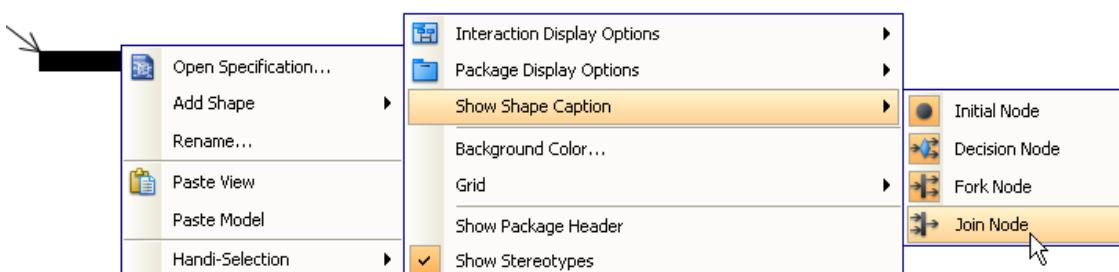


Figure 4-46 Show caption of join node

Creating activity final node

To create an activity final node from an interaction use, click on the **Control Flow -> Activity Final Node** resource beside it and drag.

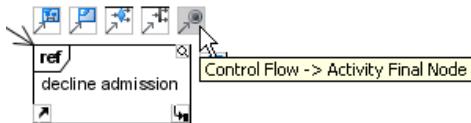


Figure 4-47 Create activity final node

Move the mouse to empty space of the diagram and then release the mouse button. An activity final node is created and connected to the interaction use with a control flow.

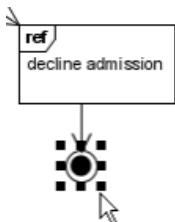


Figure 4-48 Activity final node and control flow created

Right-click the diagram and select **Presentation Options > Show Shape Caption > Activity Final Node** from the popup menu to show caption of activity final node.

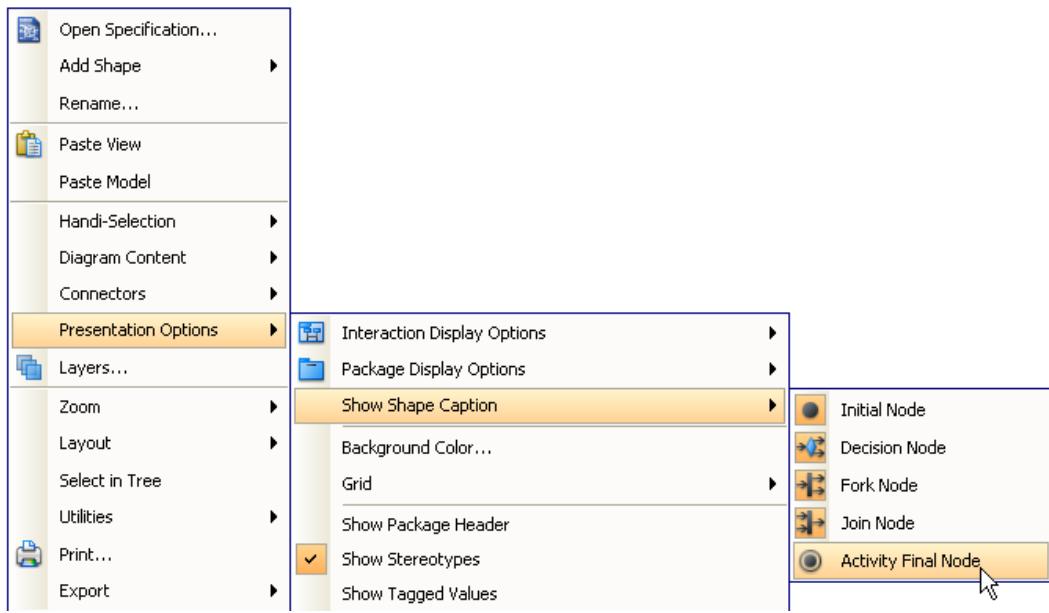


Figure 4-49 Show caption of activity final node

Continue to complete the diagram.

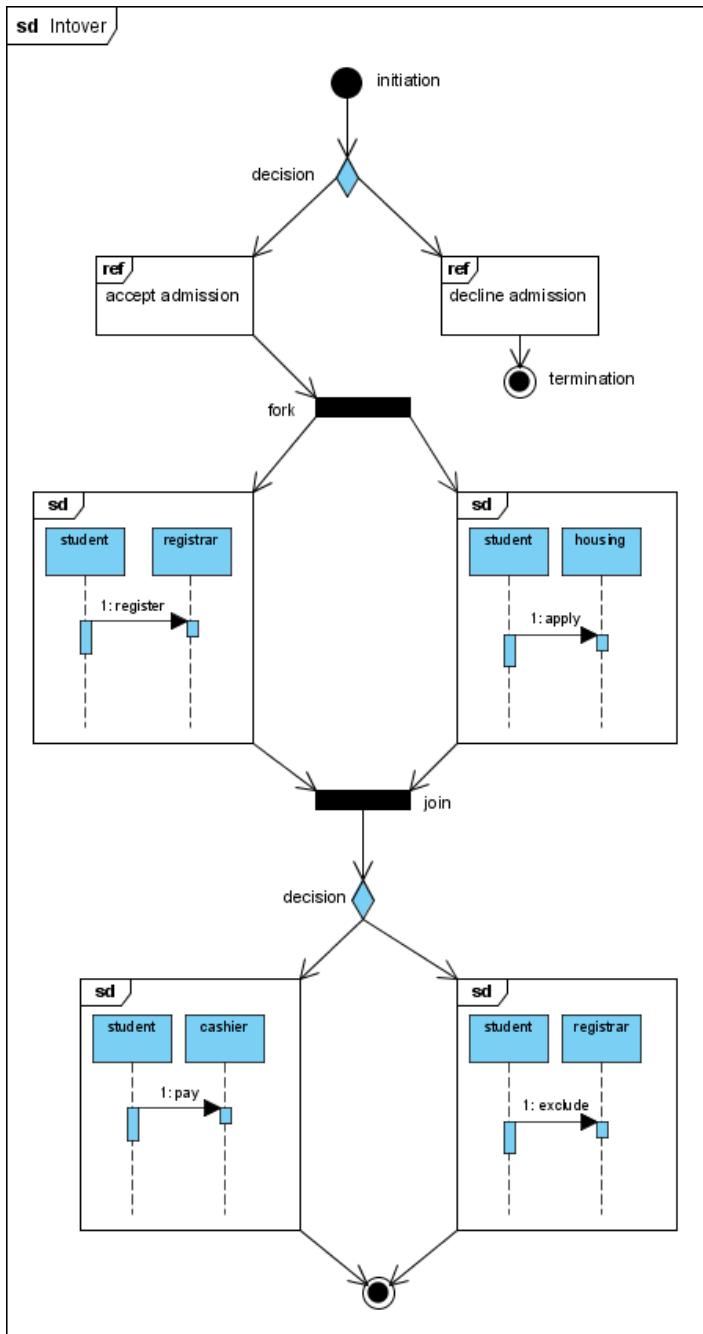


Figure 4-50 Completed diagram

Drawing class diagrams

Creating class diagram

Select menu **File > New Diagram > UML Diagrams > Class Diagram** to create a class diagram.

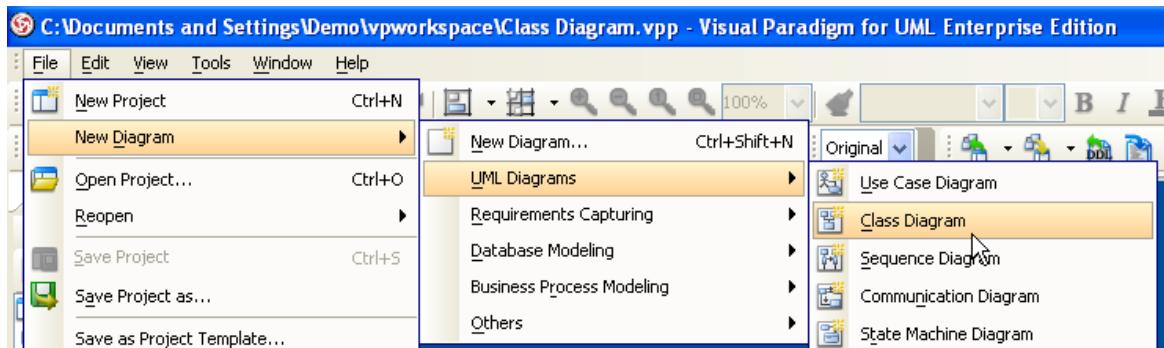


Figure 5-1 Create class diagram

Creating class

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

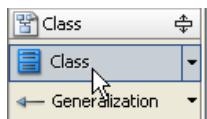


Figure 5-2 Create class

A class will be created.



Figure 5-3 Class created

Creating association

To create association from class, click the **Association -> Class** resource beside it and drag.

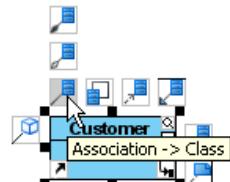


Figure 5-4 Create association

Drag to empty space of the diagram to create a new class, or drag to an existing class to connect to it. Release the mouse button to create the association.

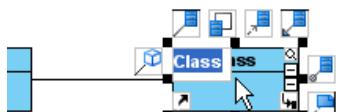


Figure 5-5 Association created

To create aggregation, use the **Aggregation -> Class** resource instead.



Figure 5-6 Create aggregation

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

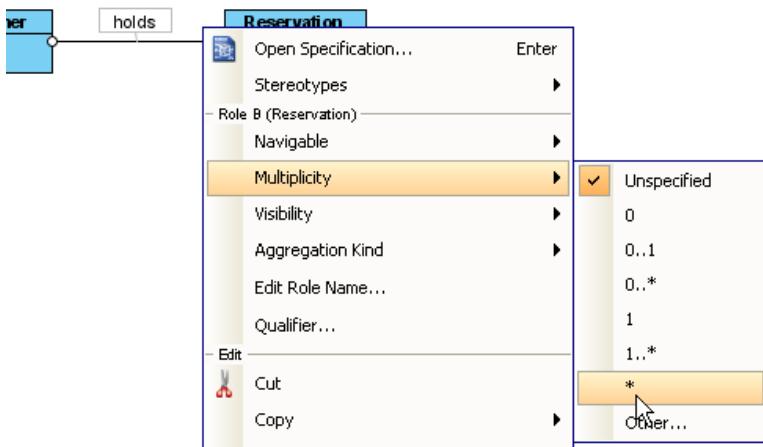


Figure 5-7 Edit multiplicity

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

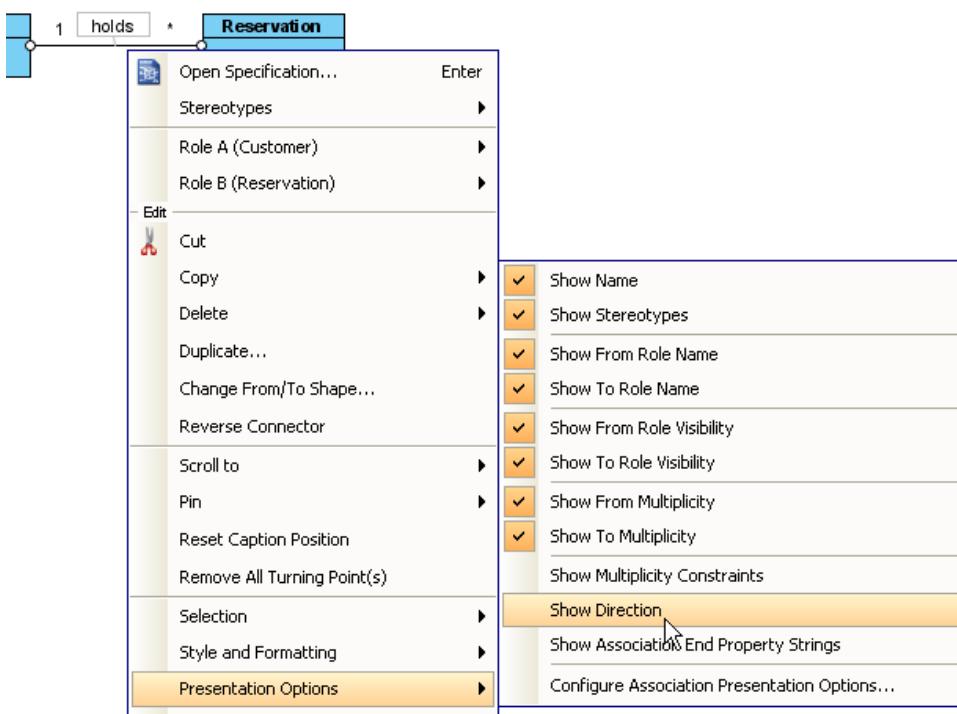


Figure 5-8 Show direction

The direction arrow is shown beside the association.



Figure 5-9 Direction shown

Creating generalization

To create generalization from class, click the **Generalization -> Class** resource beside it and drag.

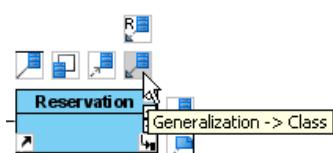


Figure 5-10 Create generalization

Drag to empty space of the diagram to create a new class, or drag to an existing class to connect to it. Release the mouse button to create the generalization.

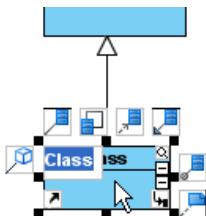


Figure 5-11 Generalization created

Creating attribute

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

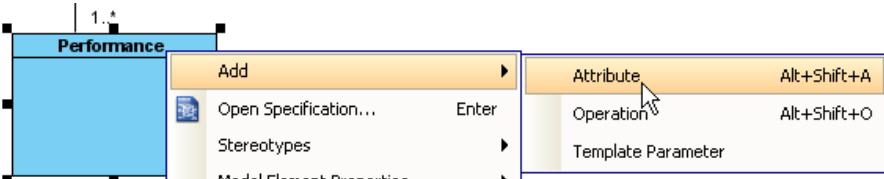


Figure 5-12 Create attribute

An attribute is created.

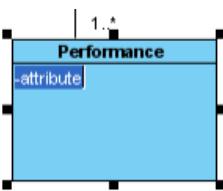


Figure 5-13 Attribute created

Creating attribute with enter key

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

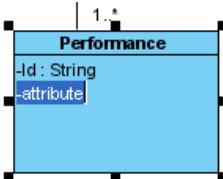


Figure 5-14 Create attribute with Enter key

Creating operation

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



Figure 5-15 Create operation

An operation is created.

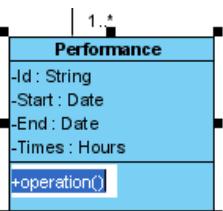


Figure 5-16 Operation created

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

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

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

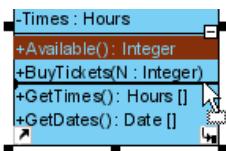


Figure 5-17 Reorder class member

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



Figure 5-18 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.

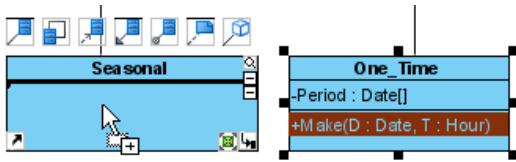


Figure 5-19 Copy class member

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

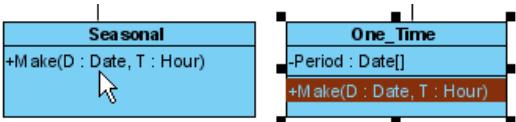


Figure 5-20 Class member copied

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.

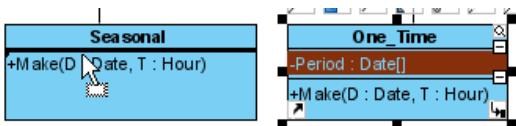


Figure 5-21 Move class member

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

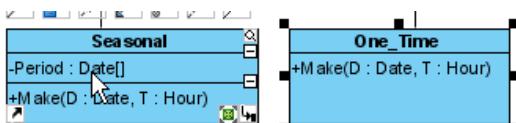


Figure 5-22 Class member moved

Model name completion for class

The model name completion feature enables quick creation of multiple views for the same class model. When create or rename class, the list of classes shows.

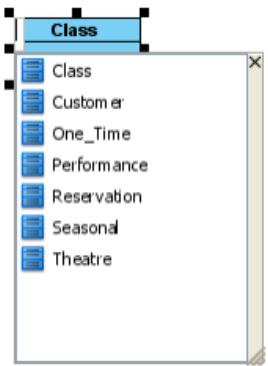


Figure 5-23 Model name completion

Type text to filter classes in the list.

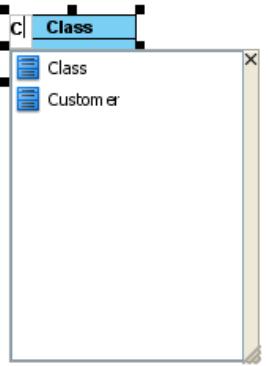


Figure 5-24 List filtered by typed text

Press up or down key to select class in the list, press Enter to confirm. Upon selecting an existing class, all class members and relationships are shown immediately.

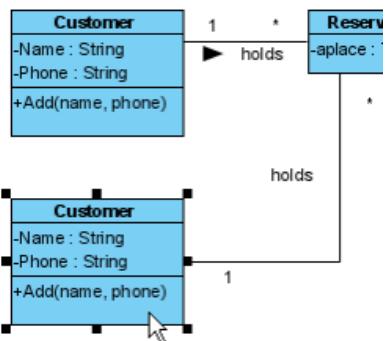


Figure 5-25 Multiple views of the same model

Continue to complete the diagram.

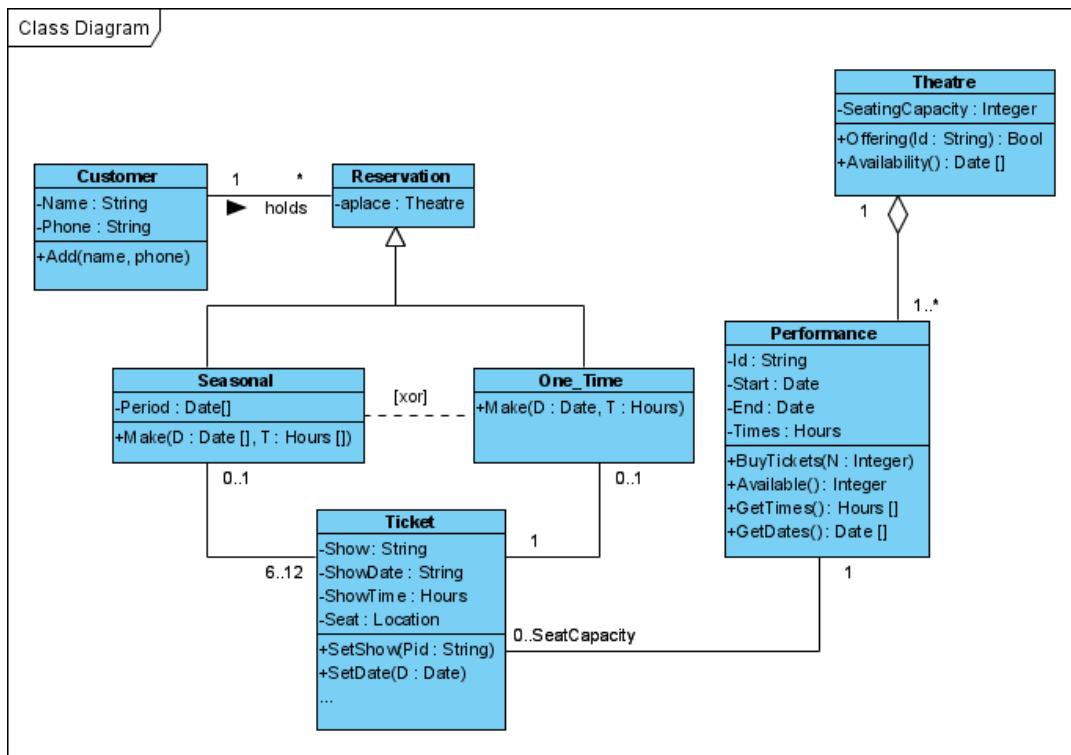


Figure 5-26 Completed diagram

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.

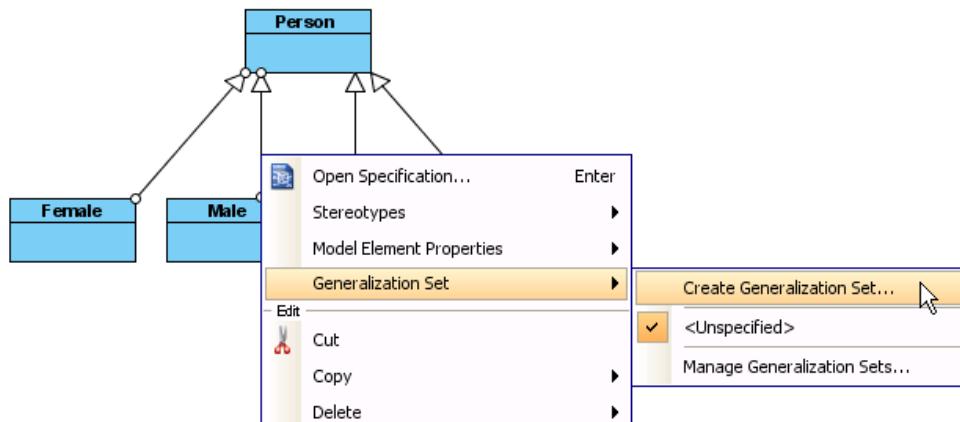


Figure 5-27 To create a generalization set

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

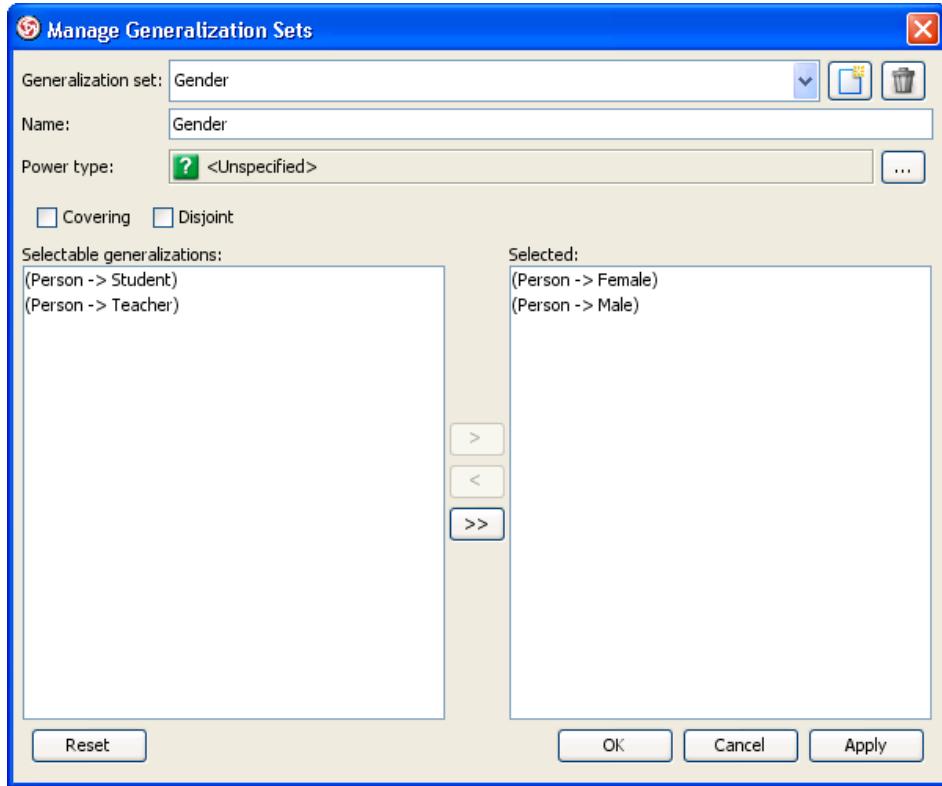


Figure 5-28 Name the generalization set

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

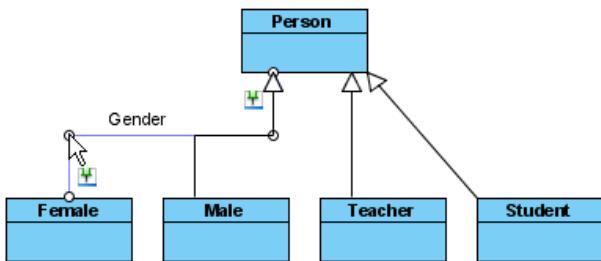


Figure 5-29 Adjust connector

Repeat the steps for other generalizations.

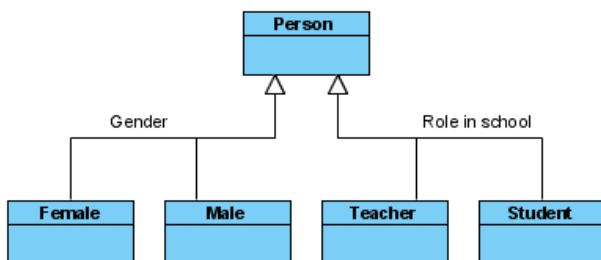


Figure 5-30 Generalization sets defined

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 delegate method, right click on the class and select **Stereotypes > Delegate** from the popup menu.

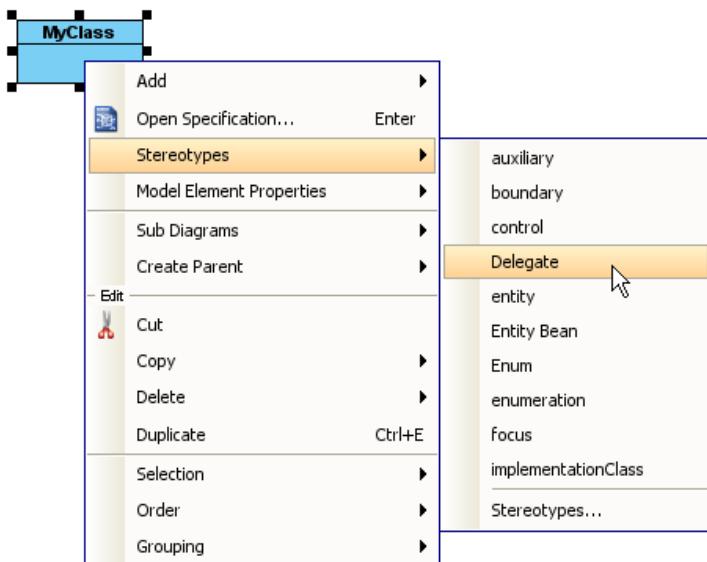


Figure 5-31 Stereotype class as Delegate

Open the specification of class by right clicking on the class and selecting **Open Specification** from the popup menu.

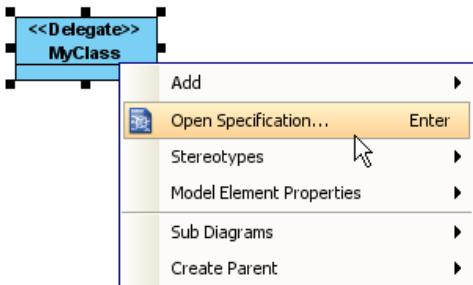


Figure 5-32 Open class specification

In the **Class Code Details** tab of the **Class Specification** dialog box, click on Edit

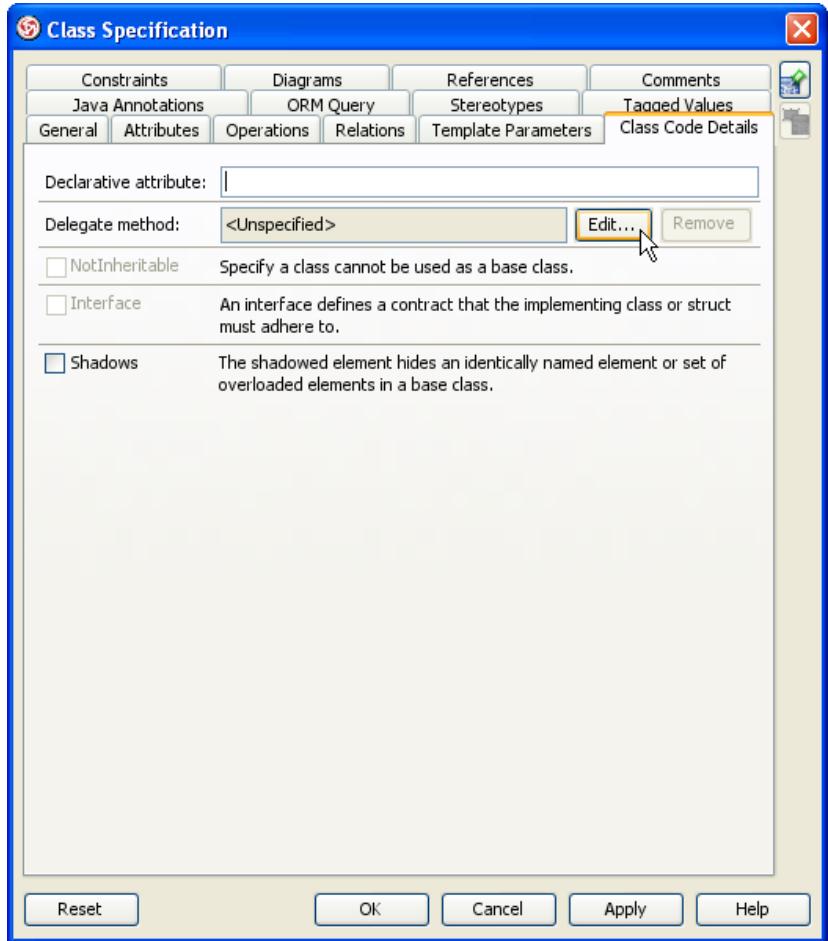


Figure 5-33 To edit delegate method

Fill in the operation specification and click **OK** to confirm editing.

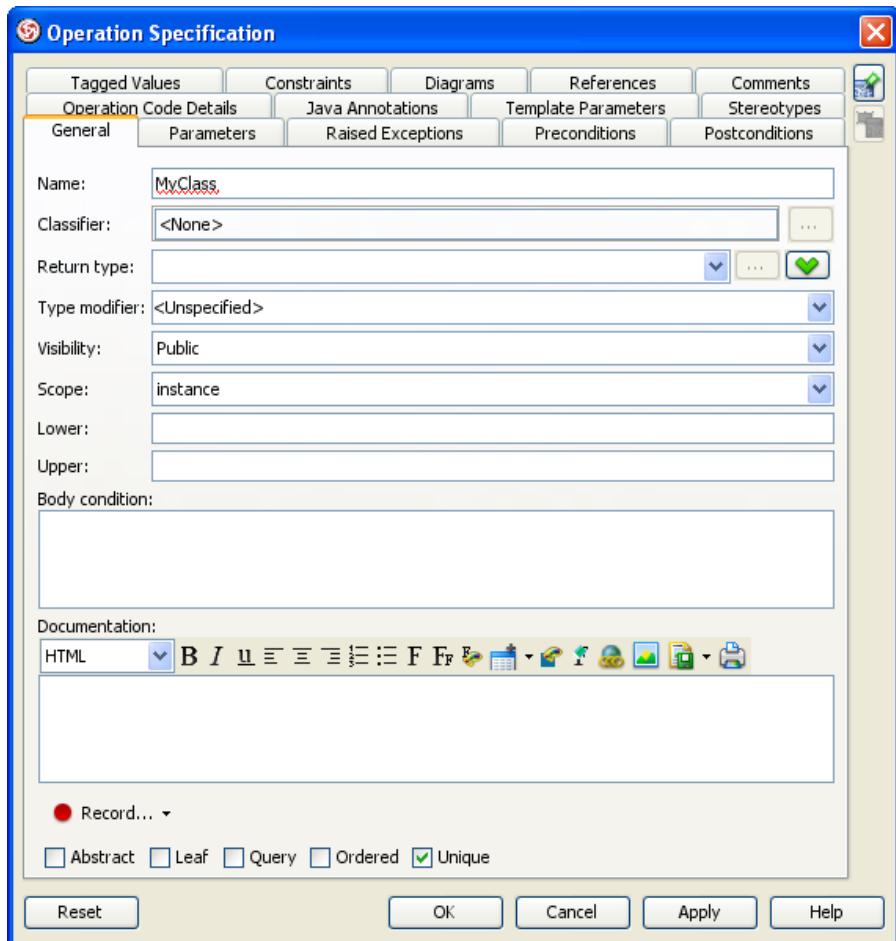


Figure 5-34 To edit operation

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 **Tools > Options** from the main menu to open the **Options** dialog box.
2. Select **Diagramming** from the list of options.
3. Open the page **Class**.
4. Open the inner page **Presentation**.

5. Change the settings for **Show attribute option** and/or **Show operation option**.

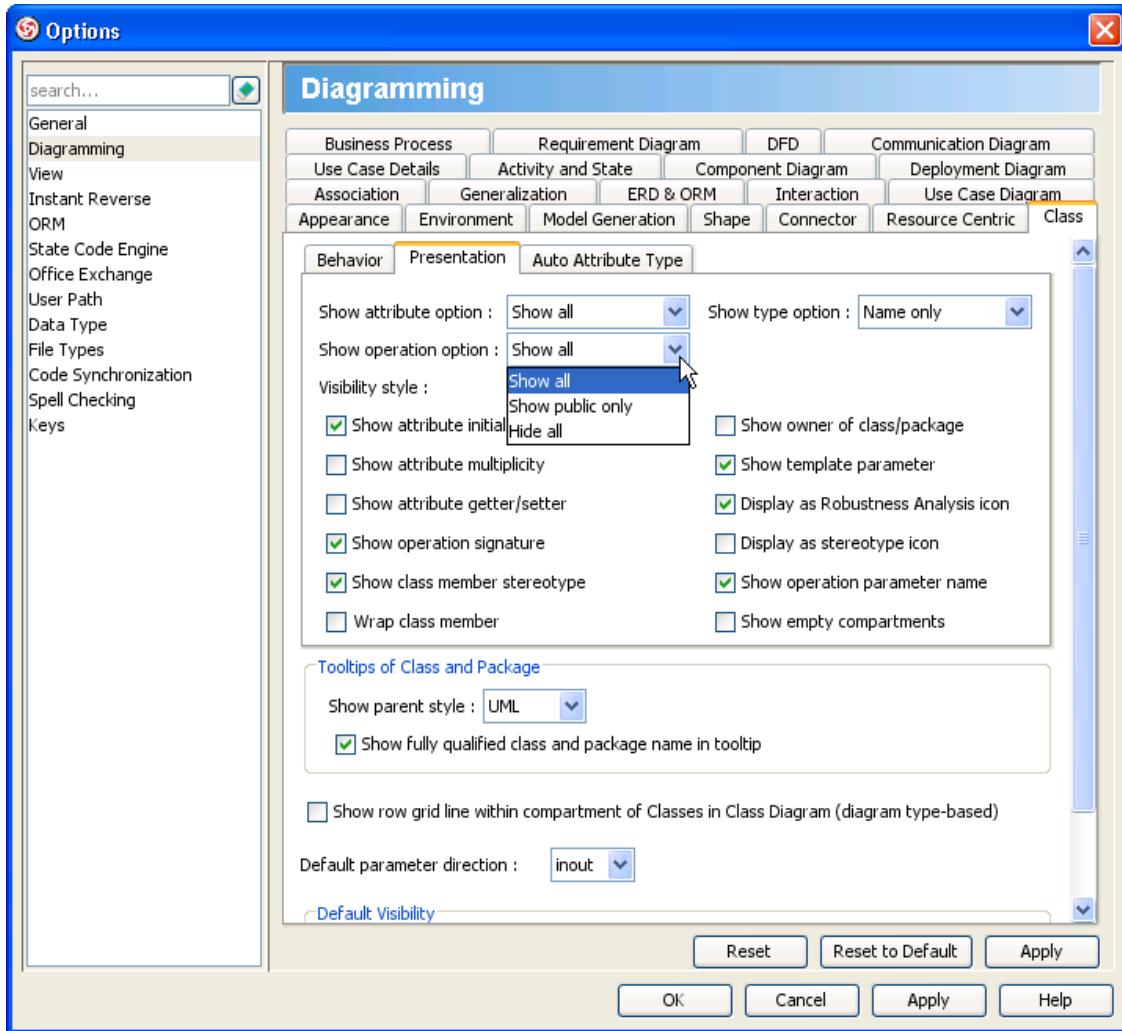


Figure 5-35 To show or hide operations

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 popup menu.

3. Select Hide All / Show All / Show Public Only.

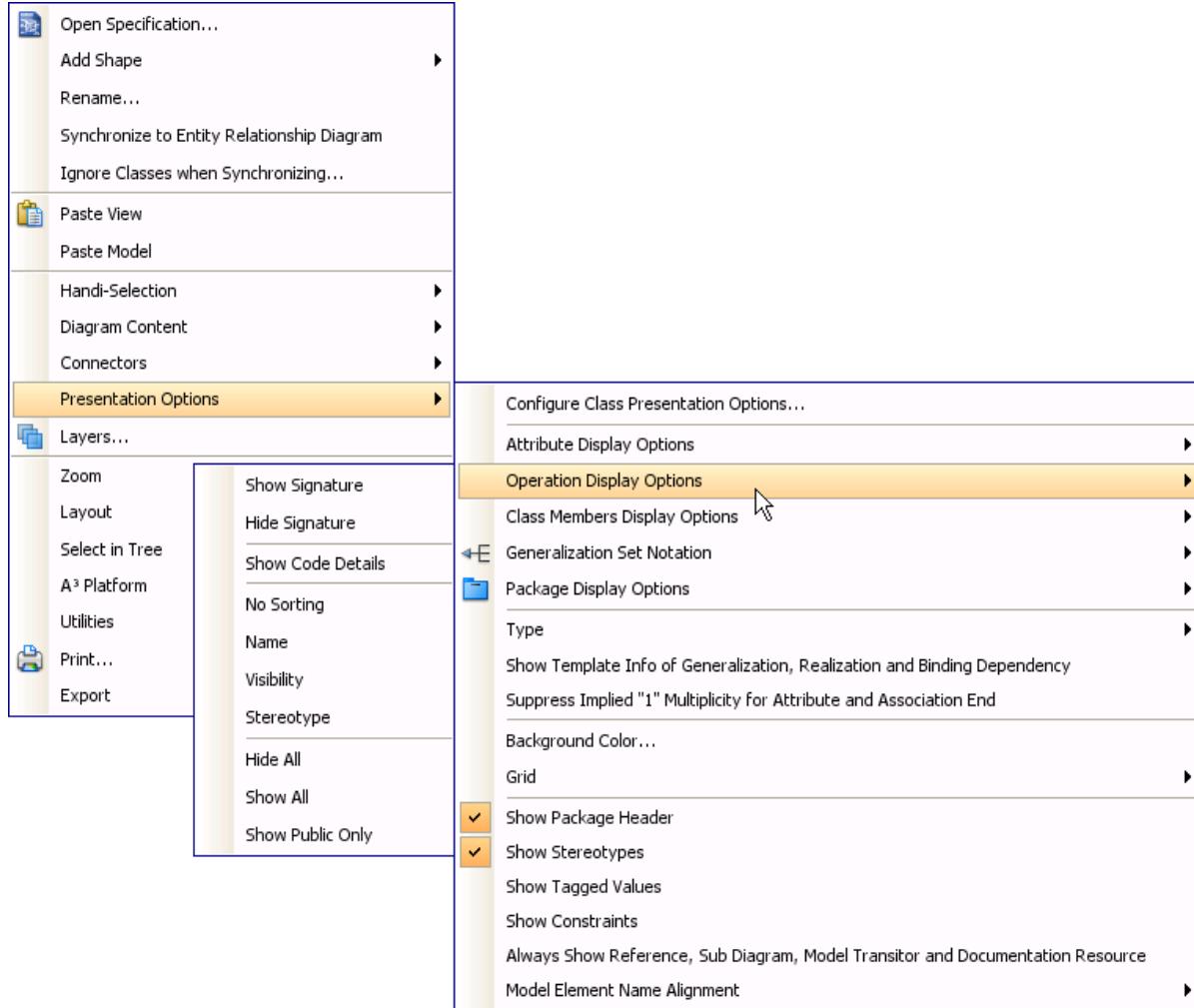


Figure 5-36 To change the operations' presentation options for classes in diagram

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.

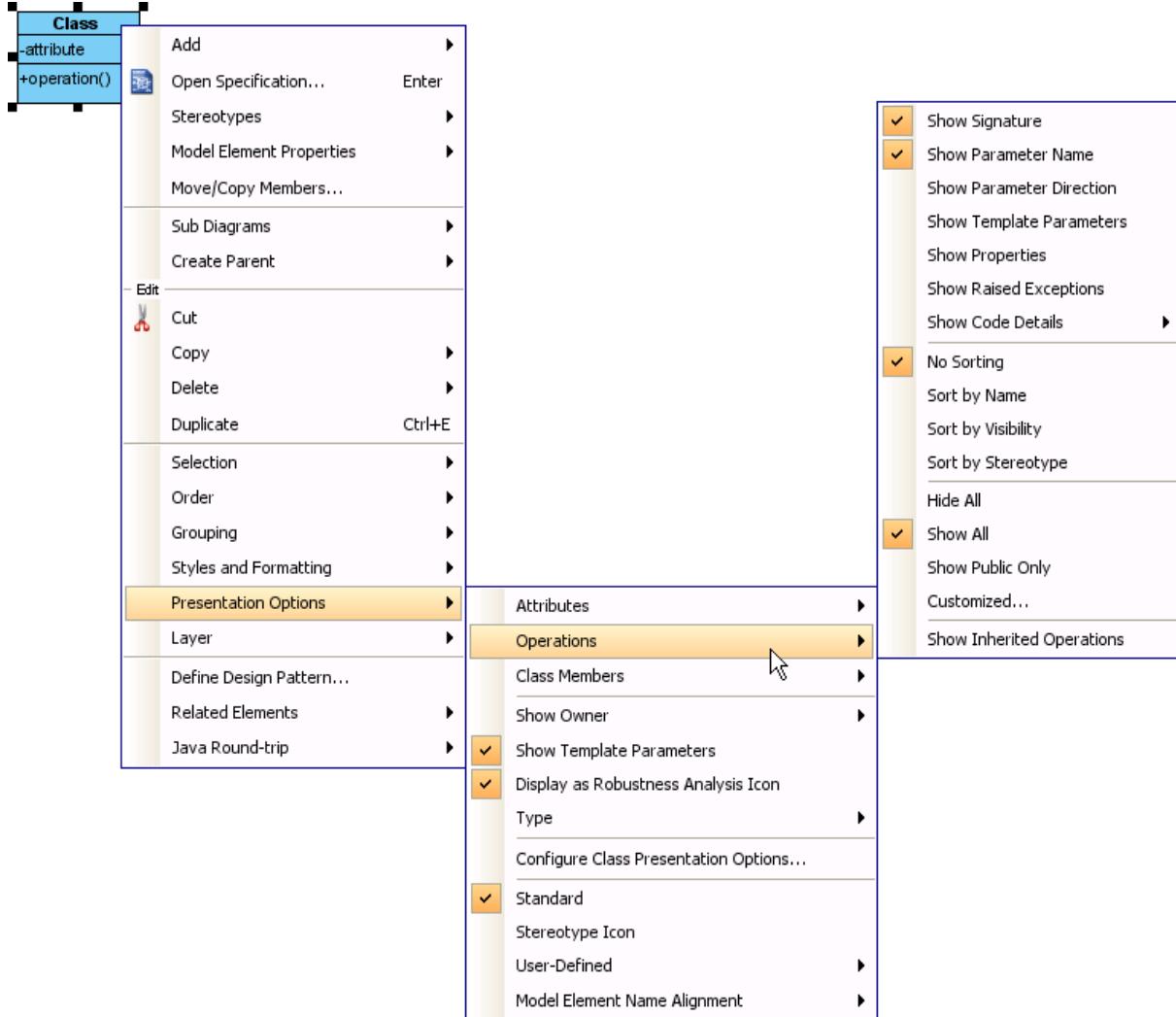


Figure 5-37 To change the operations' presentation options for a classes

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 popup menu.

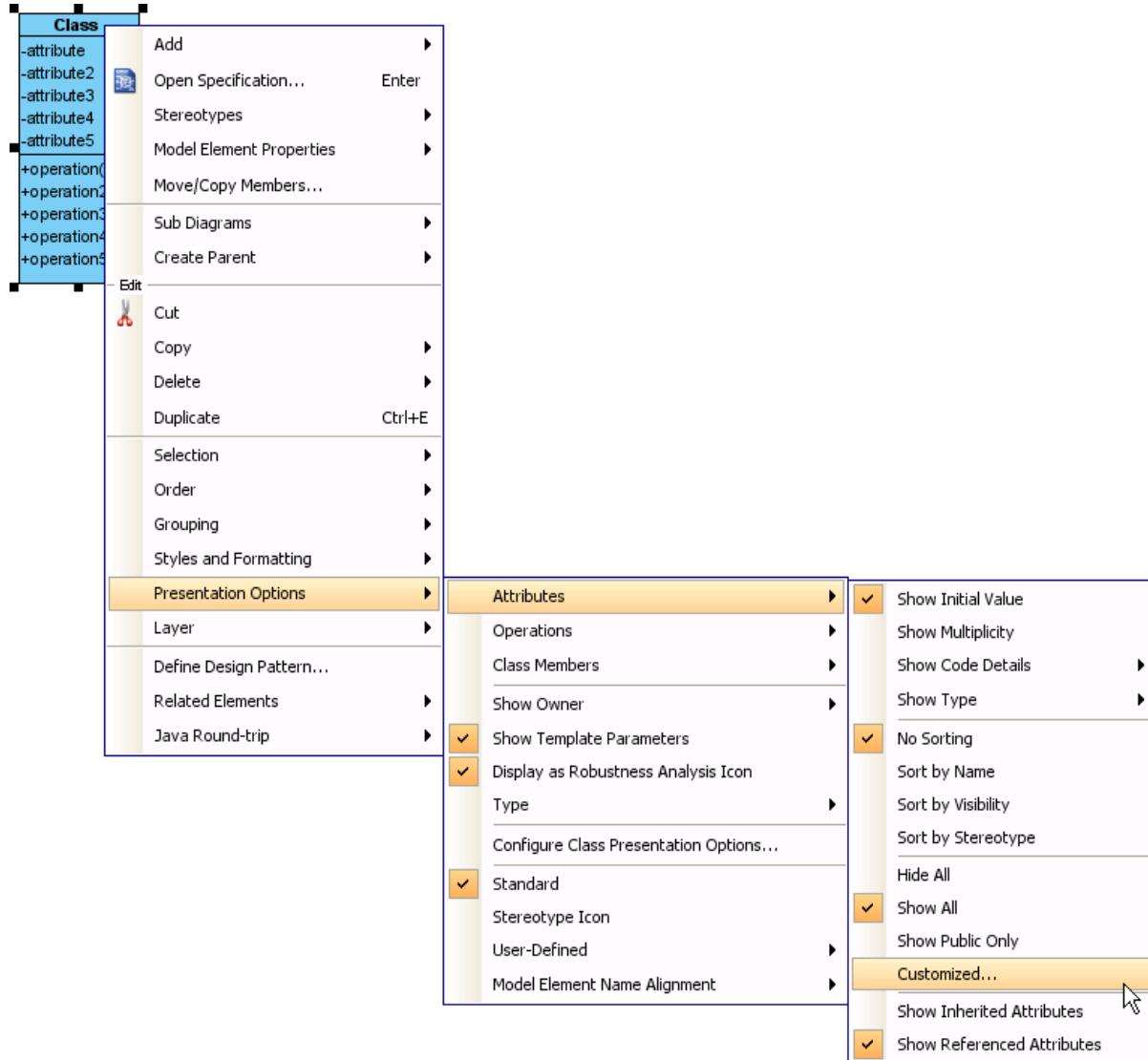


Figure 5-38 To show or hide specific class member

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

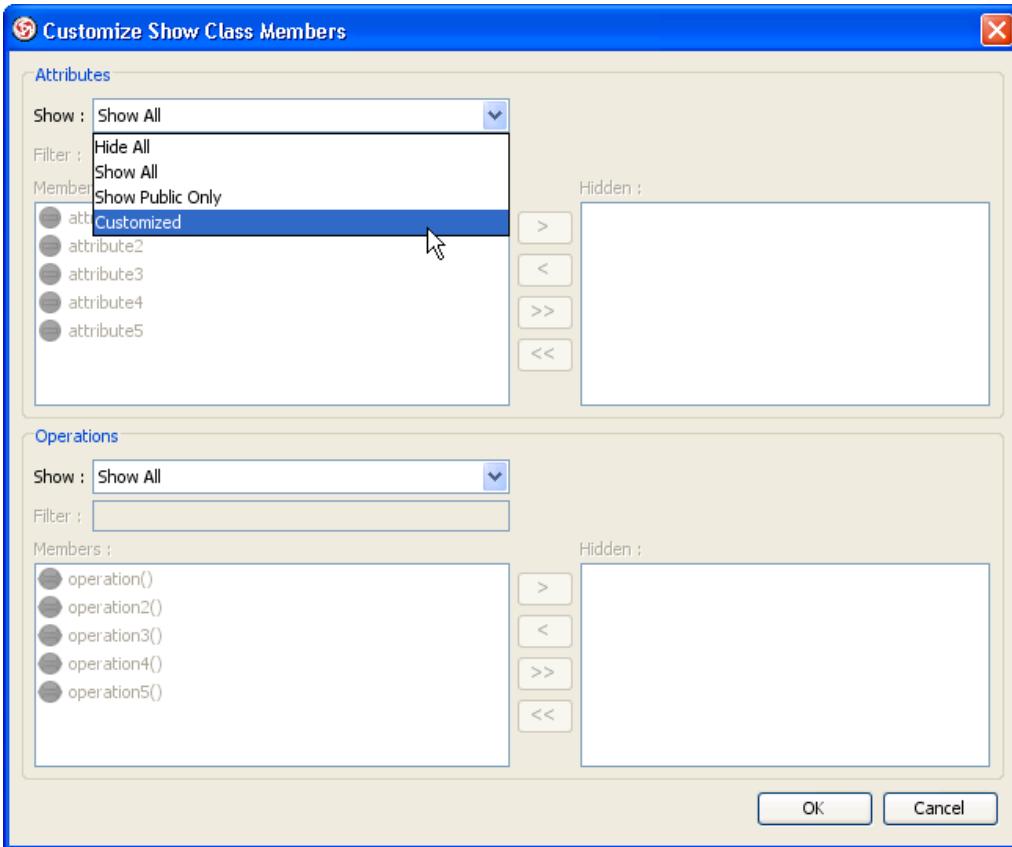


Figure 5-39 To select Customized in dialog box

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

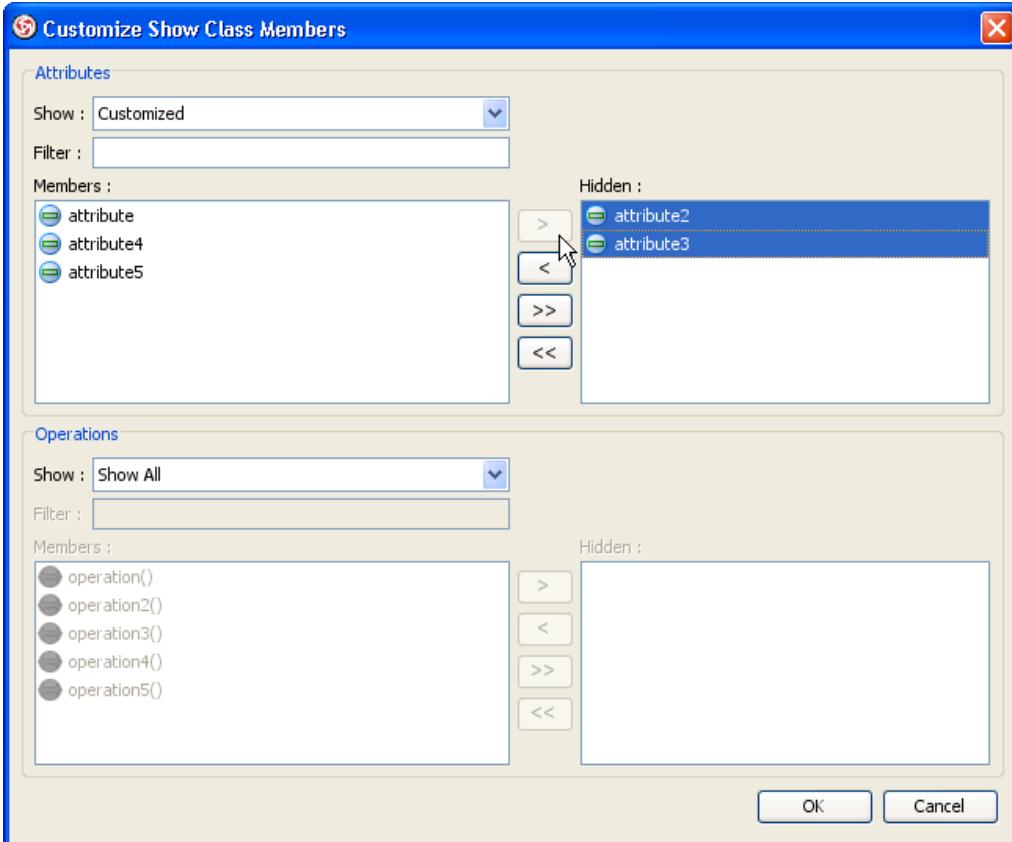


Figure 5-40 Selecting attributes to hide

5. Click **OK** to confirm.

Drawing object diagrams

Creating object diagram

Select menu **File > New Diagram > UML Diagrams > Object Diagram** to create an object diagram.

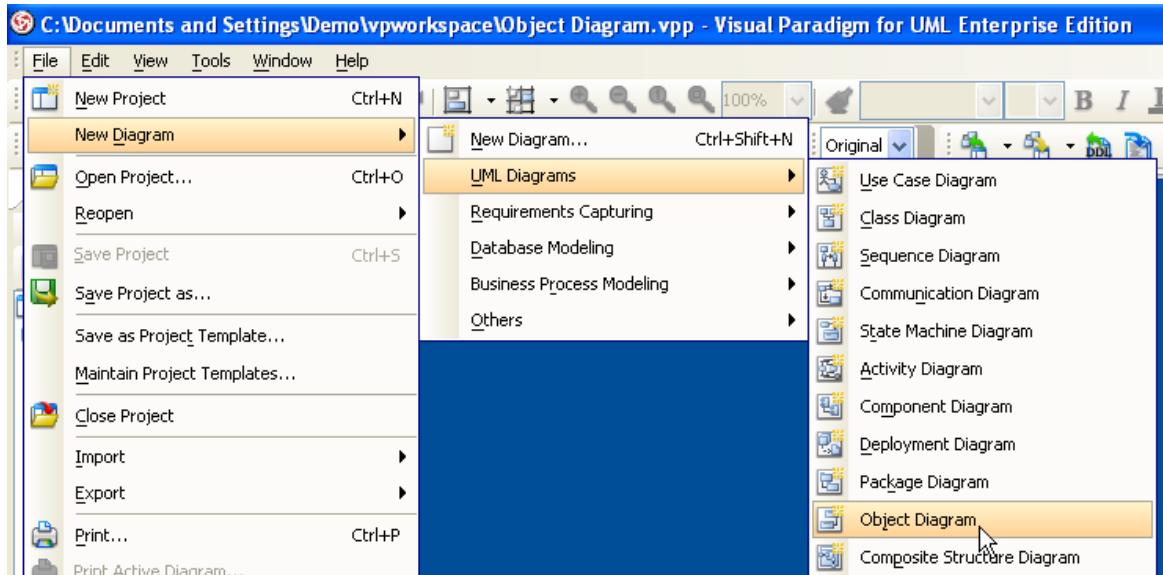


Figure 5-41 Create object diagram

Creating instance specification

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

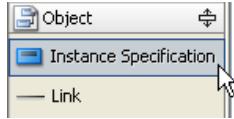


Figure 5-42 Create instance specification

An instance specification will be created.

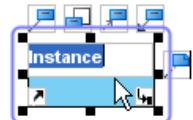


Figure 5-43 Instance specification created

Selecting classifiers

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

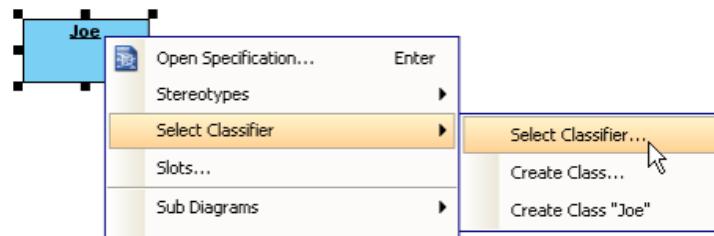


Figure 5-44 Select classifier

The **Instance Specification Specification** dialog box appears with the **Classifiers** tab selected. Select the classifiers on the left and click **Add Selected** to add them.

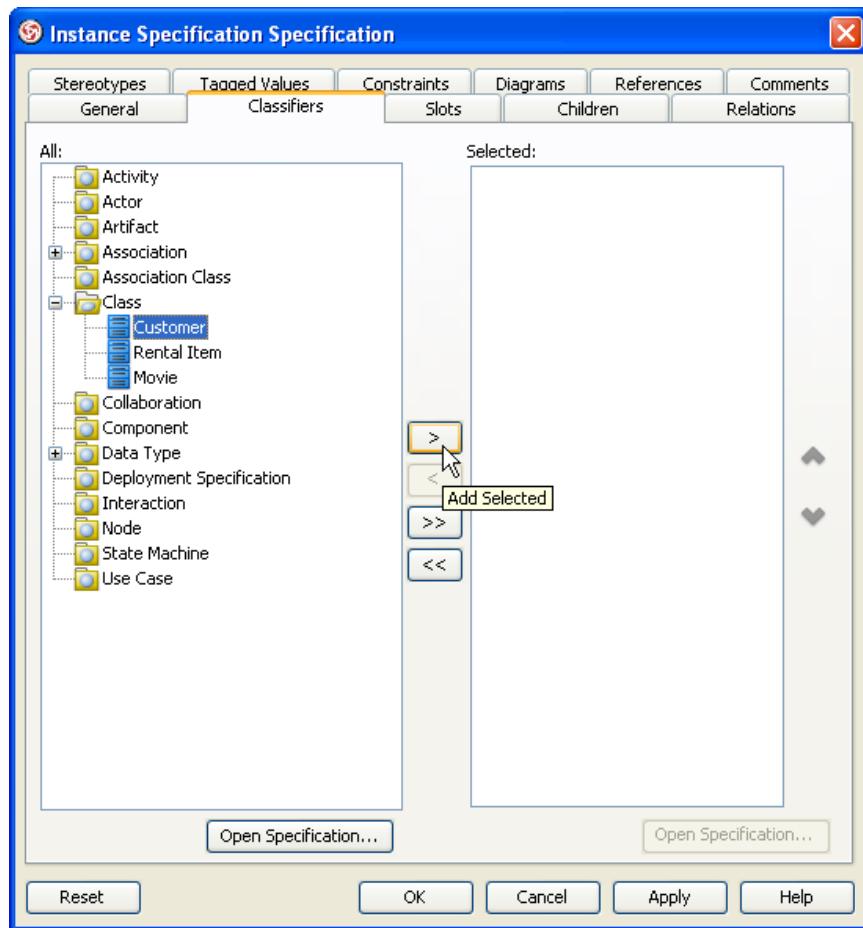


Figure 5-45 Add selected classifiers

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

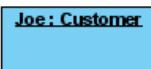


Figure 5-46 Classifiers assigned

Defining slots

To define slots for an instance specification, right-click it and select **Slots...** from the popup menu.

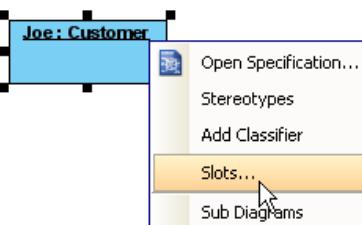


Figure 5-47 Defining slots

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**.

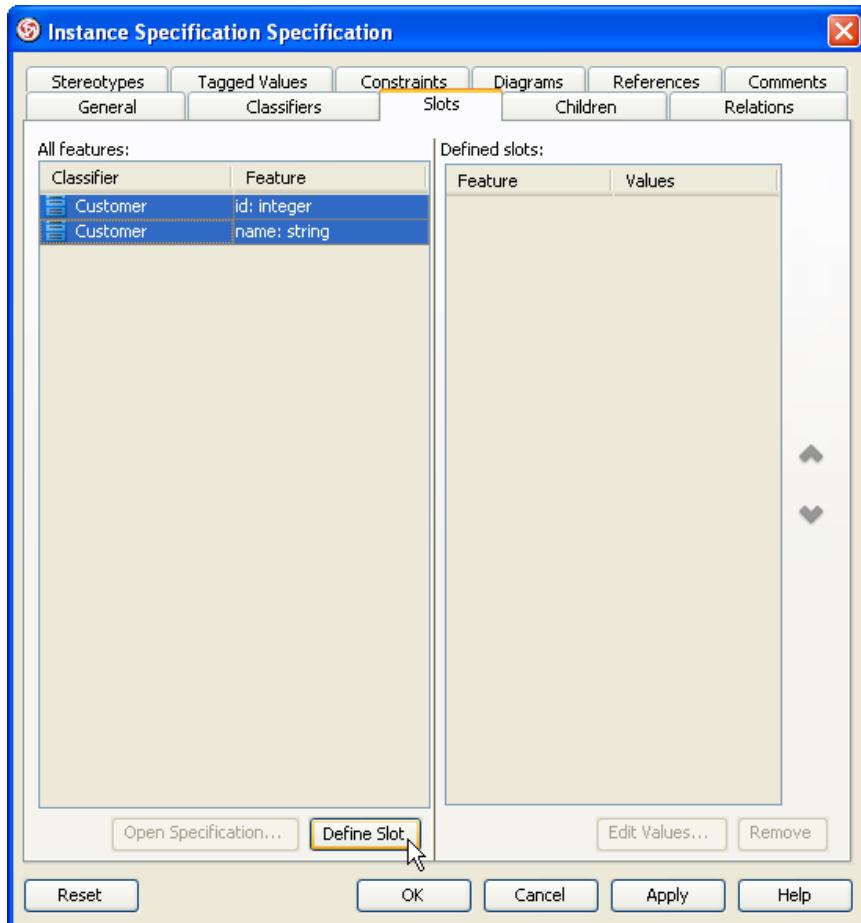


Figure 5-48 Select features to define slots

Select a slot in **Defined slots** and click **Edit Values....**

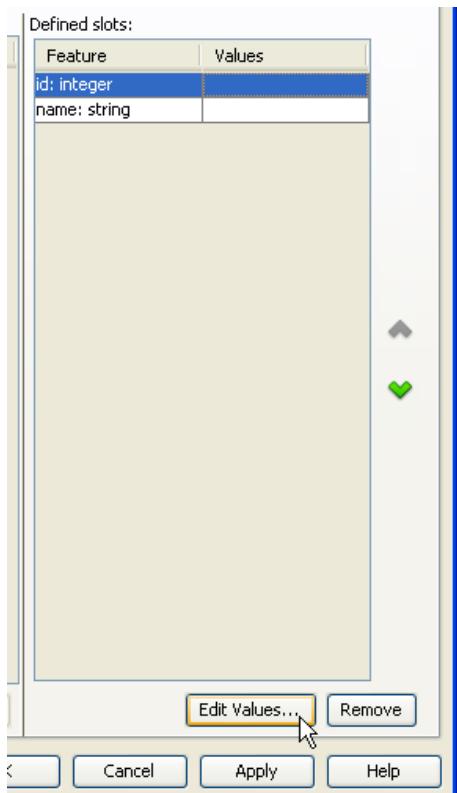


Figure 5-49 Edit values of slot

The **Slot Specification** dialog box appears with the **Values** tab selected. Click **Add** and select **Text** from the menu.

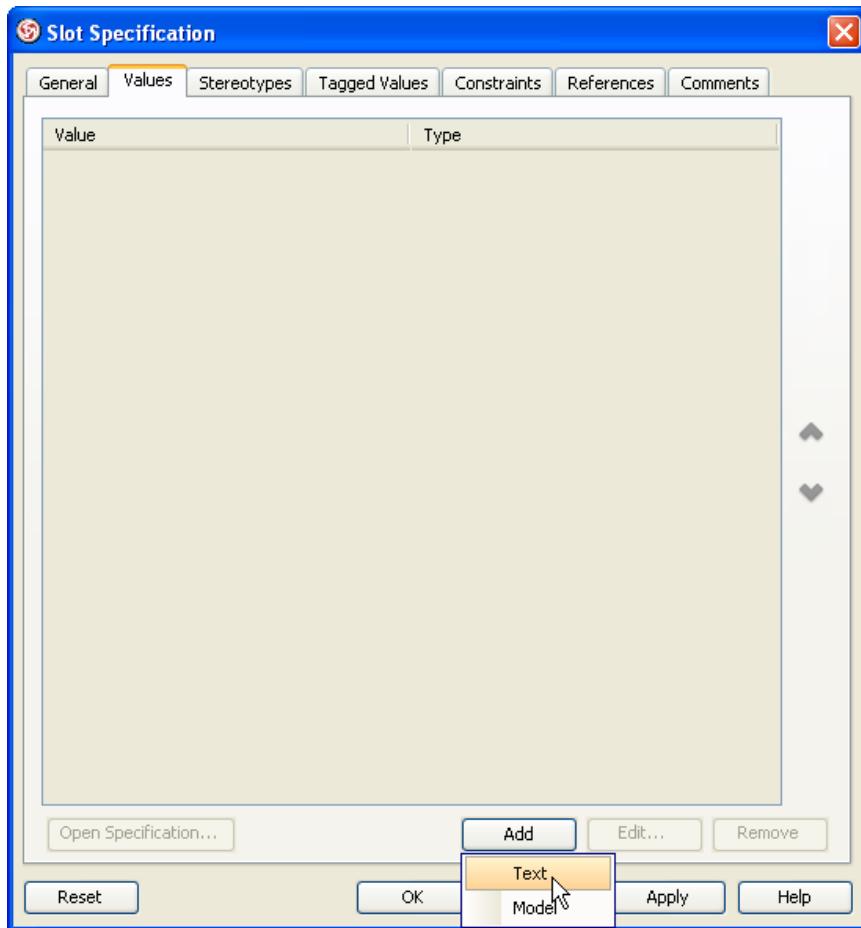


Figure 5-50 Add text value

Enter the value when prompted. Close the specification dialog boxes to apply the changes, defined slots will be shown in the instance specification.

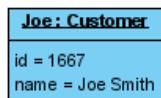


Figure 5-51 Defined slots shown

Creating link

To create link from instance specification, click the **Link -> Instance Specification** resource beside it and drag.

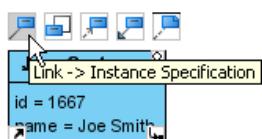


Figure 5-52 Create link

Drag to empty space of the diagram to create a new instance specification, or drag to an existing instance specification to connect to it. Release the mouse button to create the link.

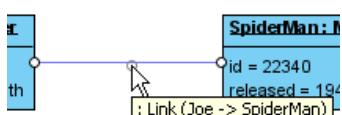


Figure 5-53 Link created

Continue to complete the diagram.

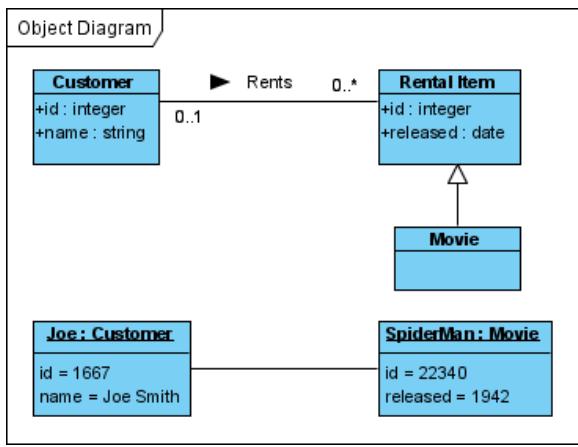


Figure 5-54 Completed diagram

Drawing package diagrams

Creating package diagram

Select menu **File > New Diagram > UML Diagrams > Package Diagram** to create a package diagram.

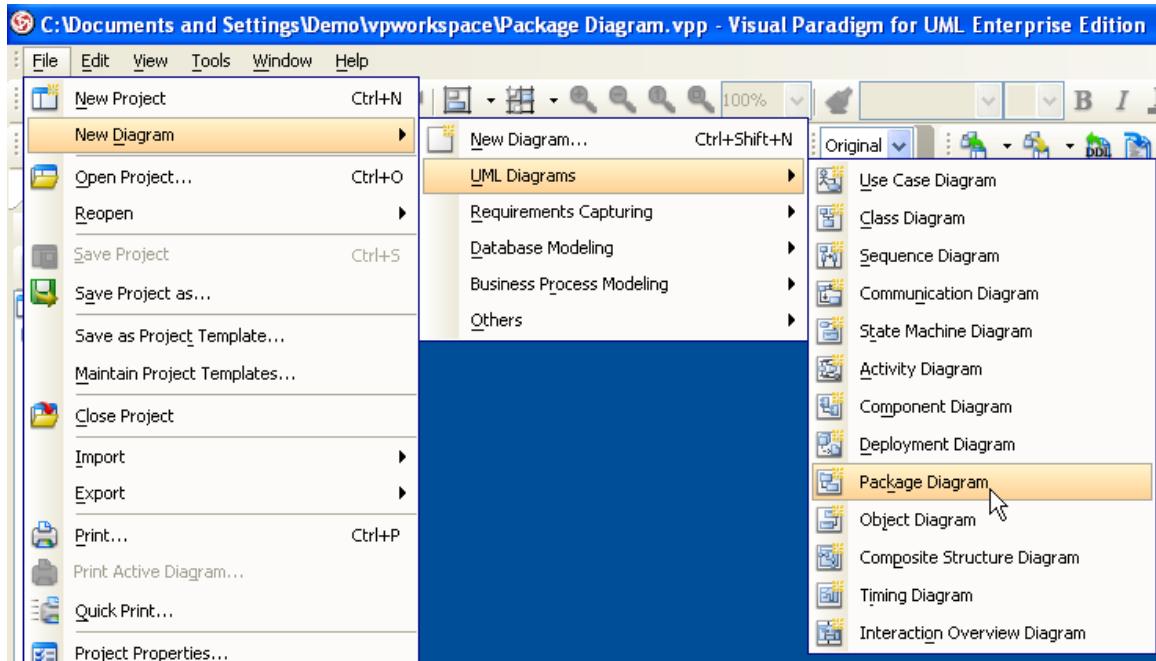


Figure 5-55 Create package diagram

Creating package

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

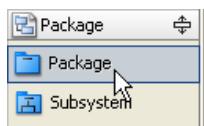


Figure 5-56 Create package

A package will be created.

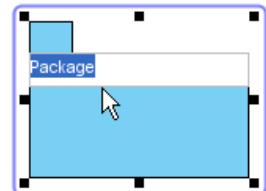


Figure 5-57 Package created

Assigning stereotypes

Right-click on the package and select **Stereotypes > Stereotypes...** from the popup menu.

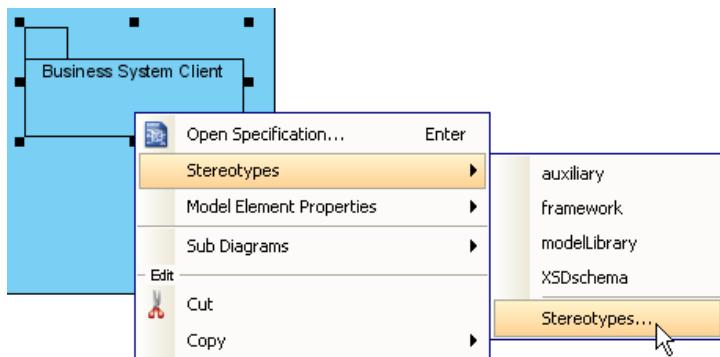


Figure 5-58 Assigning stereotypes

The **Package Specification** dialog box appears with the **Stereotypes** tab selected. The list on the left shows the selectable stereotypes.

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

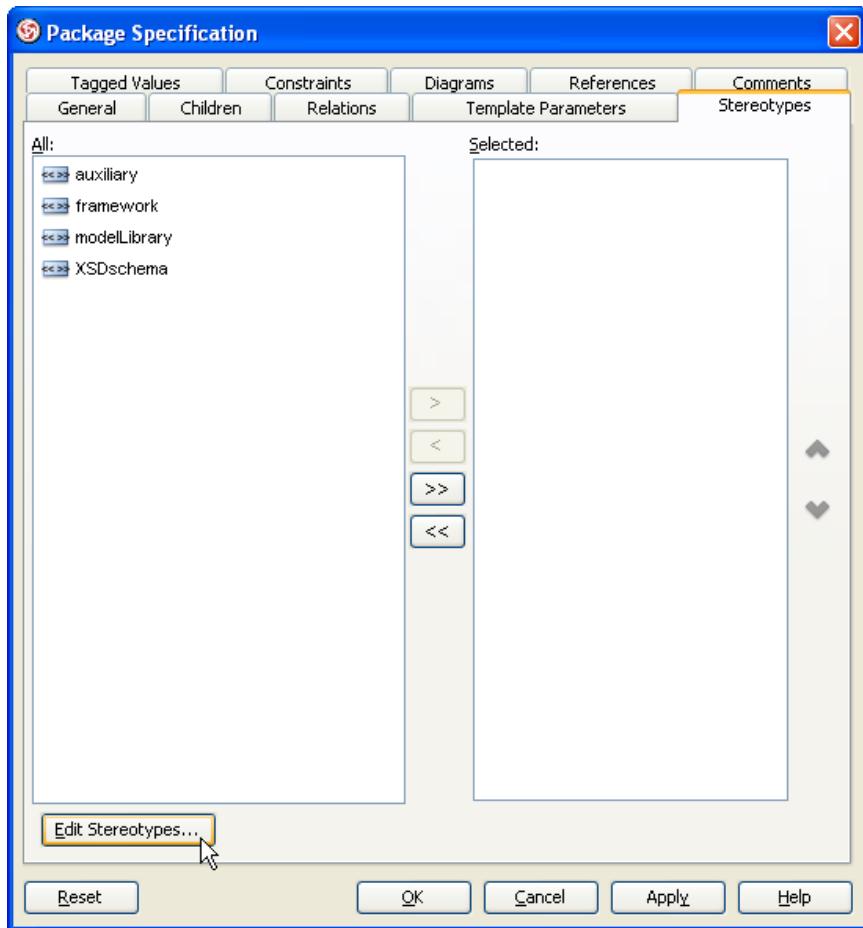


Figure 5-59 Edit stereotypes

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

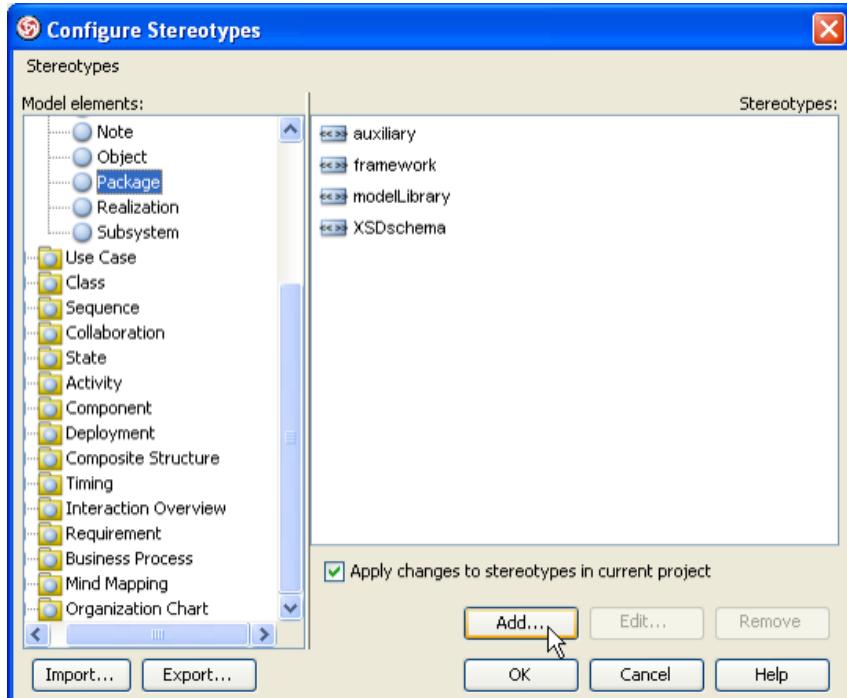


Figure 5-60 Add stereotype

Name the stereotype (e.g. *facade*). Close the Stereotype Specification and the Configure Stereotypes dialog box. You will see the added stereotype appears on the list. Select it and click **Add Selected**.

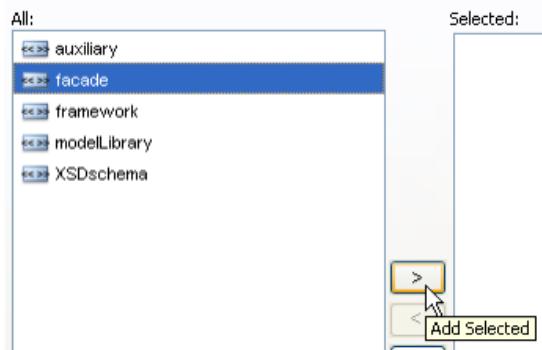


Figure 5-61 Add selected stereotypes

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

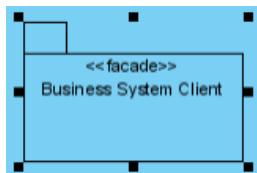


Figure 5-62 Stereotypes assigned

Continue to complete the diagram.

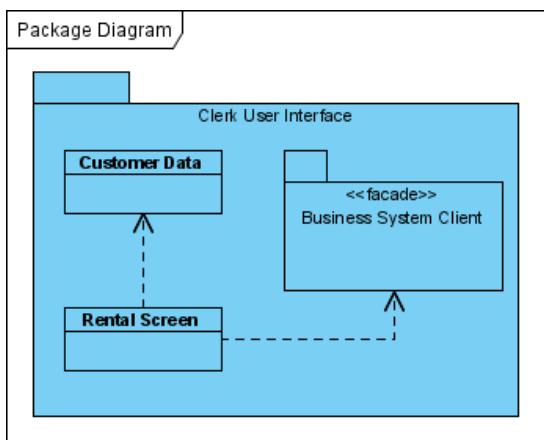


Figure 5-63 Completed diagram

Drawing composite structure diagrams

Creating composite structure diagram

Select menu **File > New Diagram > UML Diagrams > Composite Structure Diagram** to create a composite structure diagram.

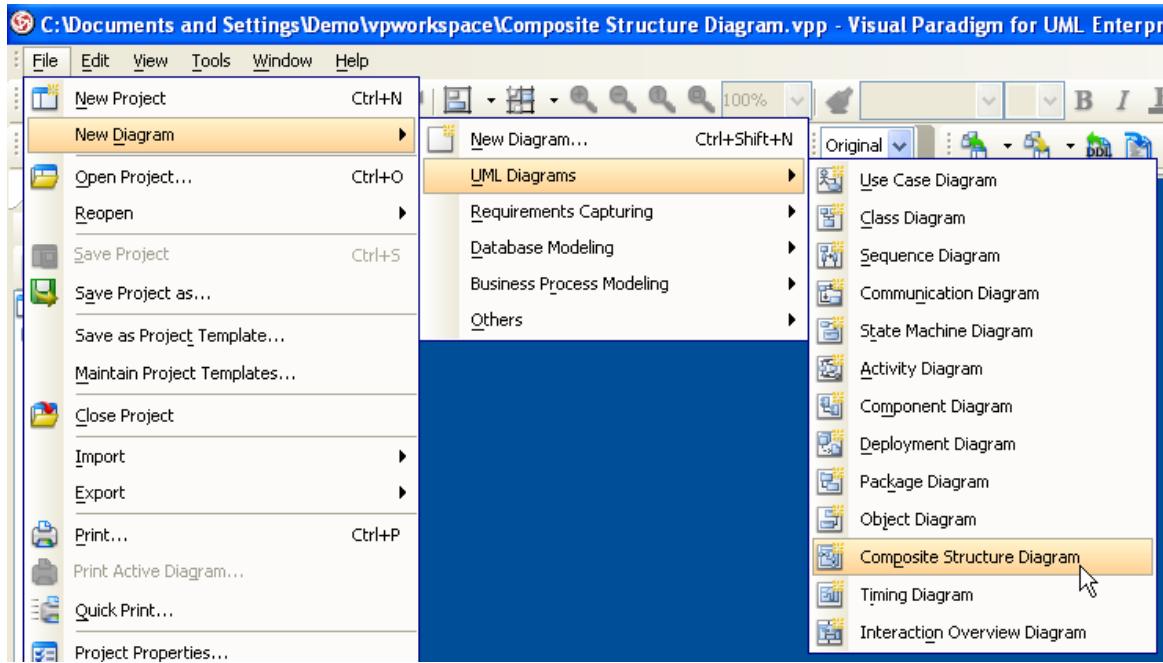


Figure 5-64 Create composite structure diagram

Creating class

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

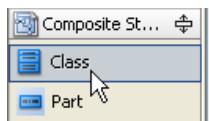


Figure 5-65 Create class

A class will be created.



Figure 5-66 Class created

Creating part

To create part, click the **New Part** resource of class.

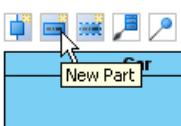


Figure 5-67 Create part

A part will be created.

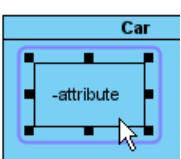


Figure 5-68 Part created

Creating port

To create port, click the **New Port** resource of class.

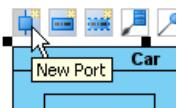


Figure 5-69 Create port

A port will be created.

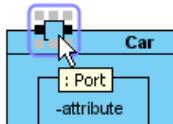


Figure 5-70 Port created

Specifying type of port

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

Click the combo box beside the **Type** field, select a class.

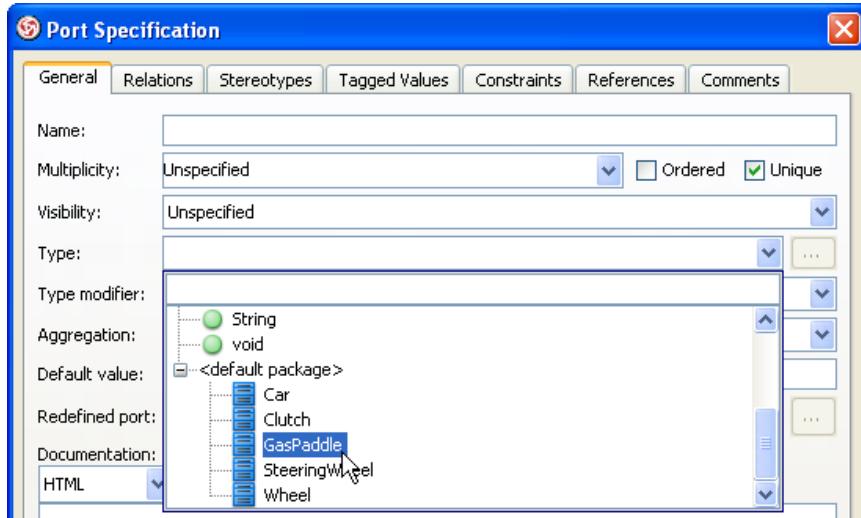


Figure 5-71 Select type

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



Figure 5-72 Type shown on port

Creating connector

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



Figure 5-73 Create connector

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

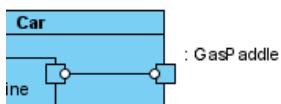


Figure 5-74 Connector created

Continue to complete the diagram.

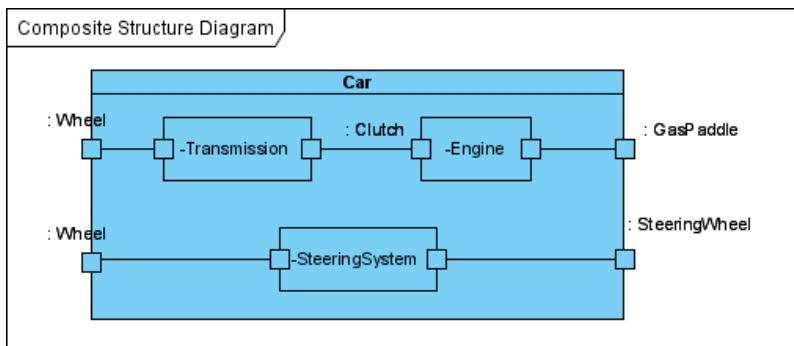


Figure 5-75 Completed diagram

Drawing component diagrams

Creating component diagram

Select menu **File > New Diagram > UML Diagrams > Component Diagram** to create a component diagram.

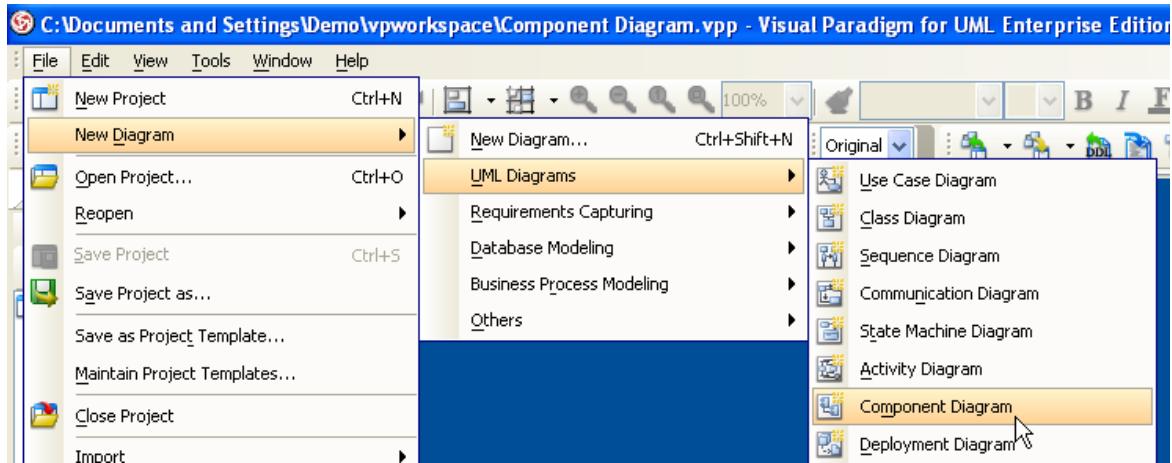


Figure 6-1 Create component diagram

Creating component

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

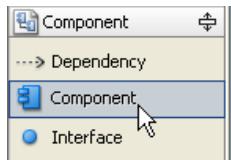


Figure 6-2 Create component

A component will be created.

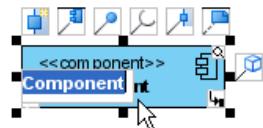


Figure 6-3 Component created

Assigning stereotypes

Right-click on the package and select **Stereotypes > Stereotypes...** from the popup menu.

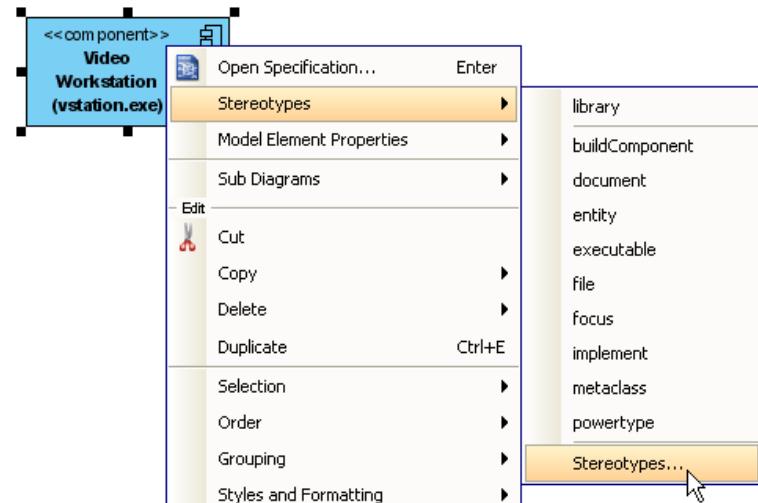


Figure 6-4 Assigning stereotypes

The **Component Specification** dialog box appears with the **Stereotypes** tab selected. The list on the left shows the selectable stereotypes.

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

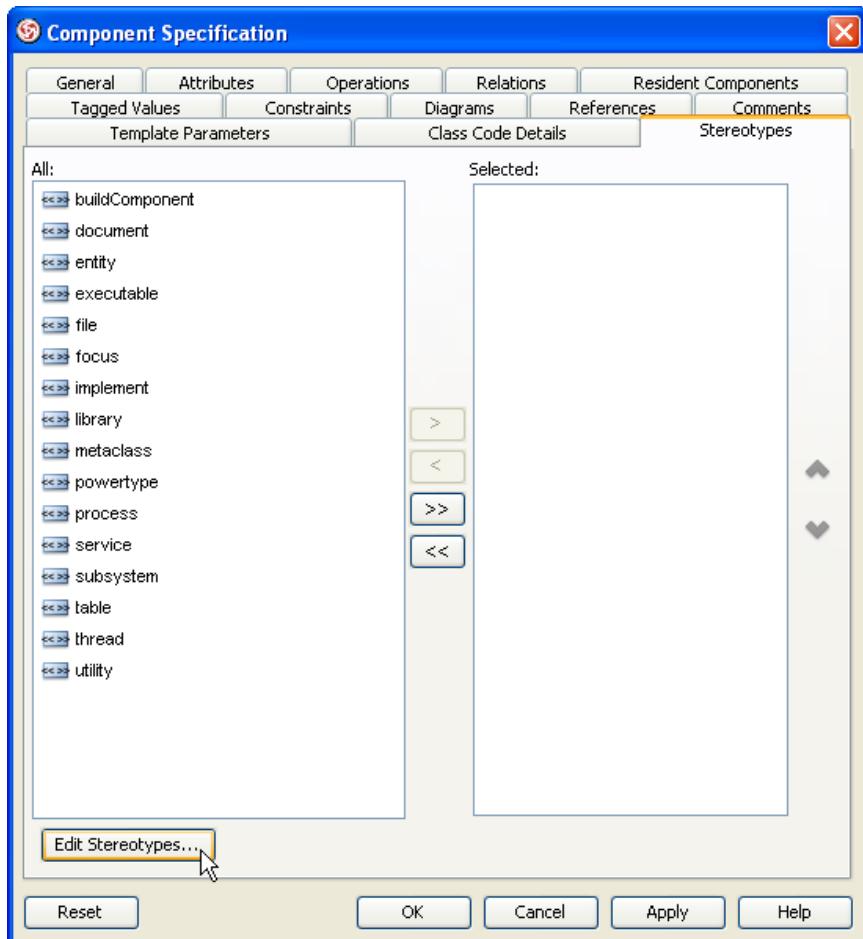


Figure 6-5 Edit stereotypes

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

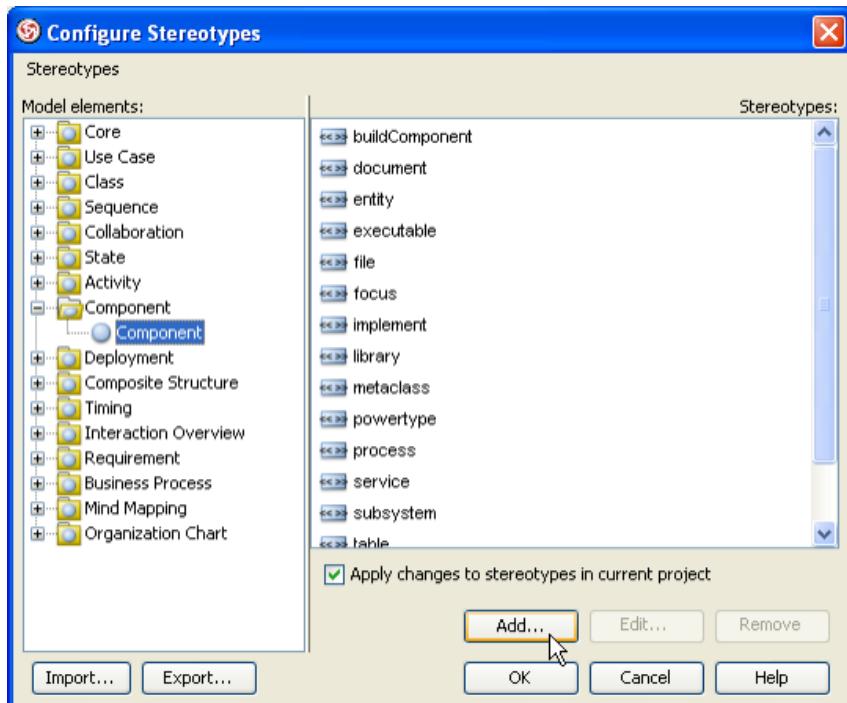


Figure 6-6 Add stereotype

Name the stereotype (e.g. *application*). Close the Stereotype Specification and the Configure Stereotypes dialog box. You will see the added stereotype appears on the list. Select it and click **Add Selected**.

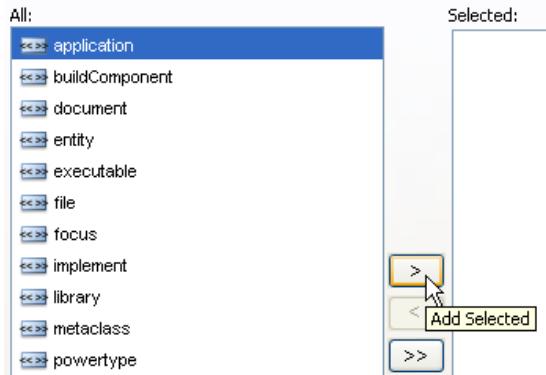


Figure 6-7 Add selected stereotypes

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

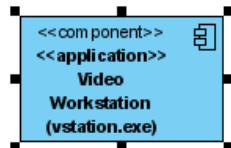


Figure 6-8 Stereotypes assigned

Creating provided interface

To create provided interface for a component, click on the **Realization -> Interface** resource beside it and drag.



Figure 6-9 Create provided interface

Move the mouse to empty space of the diagram and then release the mouse button. The provided interface is created.

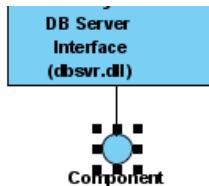


Figure 6-10 Provided interface created

Creating required interface

To create required interface for a component, click on the **Usage-> Interface** resource beside it and drag.



Figure 6-11 Create required interface

Drag to empty space of the diagram to create a new interface, or drag to an existing interface to connect to it. Release the mouse button to create the required interface.

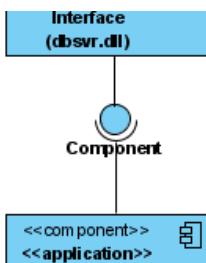


Figure 6-12 Required interface created

Creating dependency

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

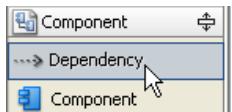


Figure 6-13 Create dependency

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

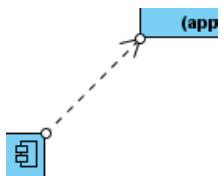


Figure 6-14 Dependency created

Continue to complete the diagram

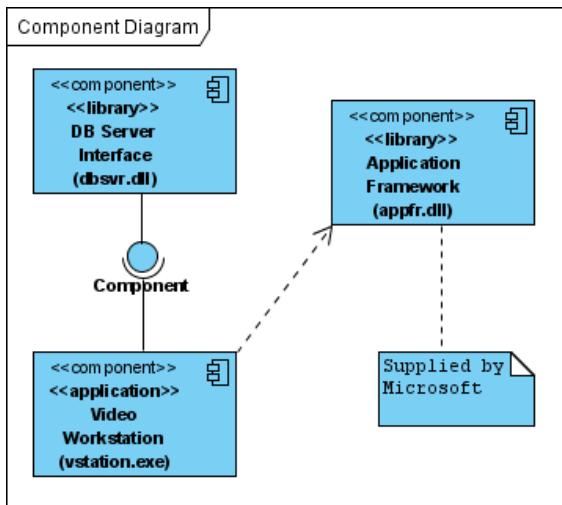


Figure 6-15 Completed diagram

Drawing deployment diagrams

Creating deployment diagram

Select menu **File > New Diagram > UML Diagrams > Deployment Diagram** to create a deployment diagram.

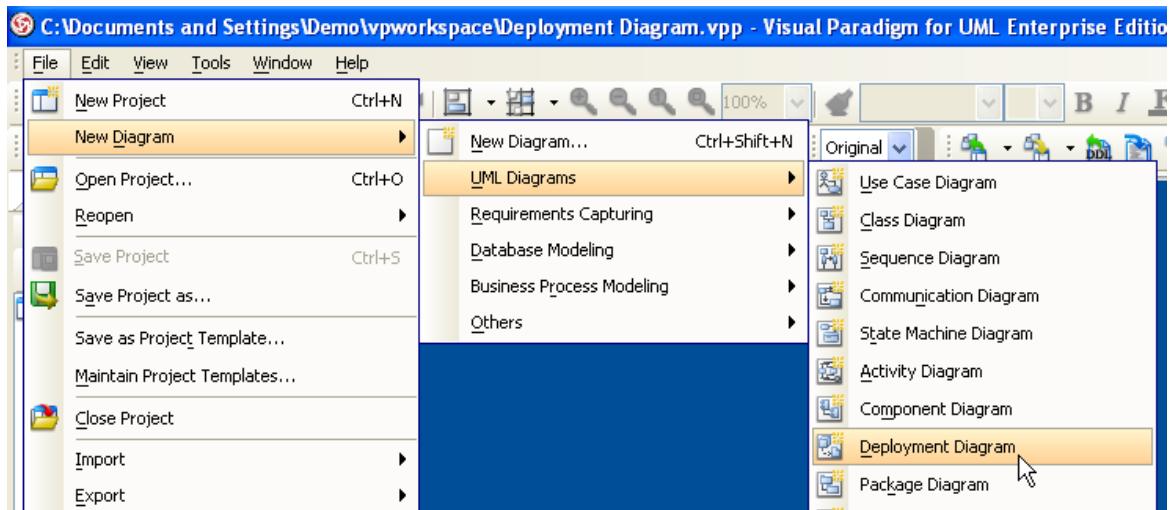


Figure 6-16 Create deployment diagram

Creating node model element

To create node model element, right-click on empty space of **Model Explorer** and select **Model Element > New Model Element** from the popup menu.

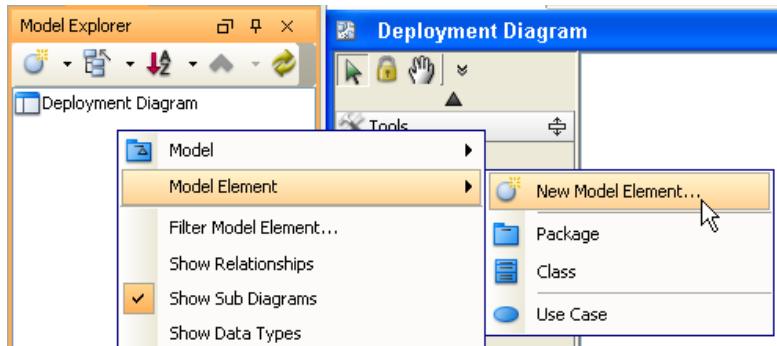


Figure 6-17 Create model element

In the **New Model Element** dialog box, type **Node** in **Model element type**, type the node name in **Model element name**. Click **OK** to confirm.

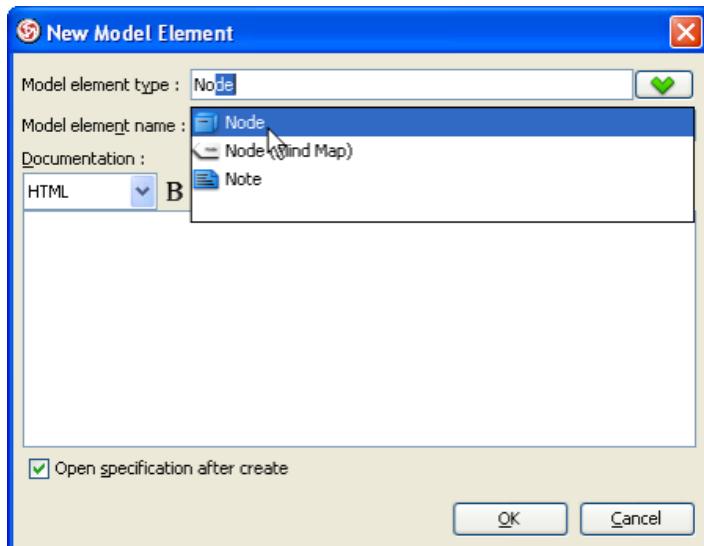


Figure 6-18 Create node

A node model element will be created.

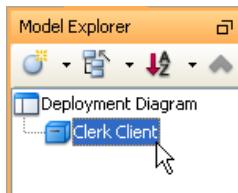


Figure 6-19 Node created

Creating instance of node

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



Figure 6-20 Create instance specification

An instance specification will be created.



Figure 6-21 Instance specification created

Selecting classifiers

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

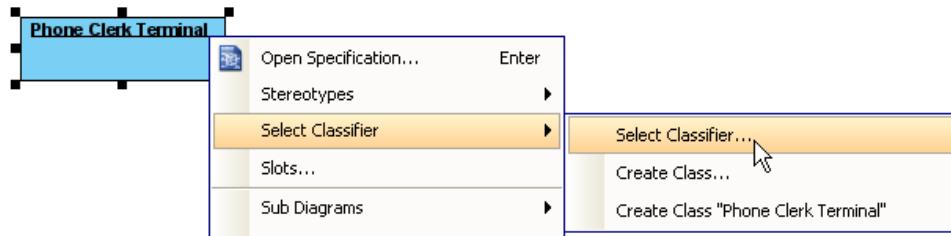


Figure 6-22 Select classifier

The **Instance Specification Specification** dialog box appears with the **Classifiers** tab selected. Select the classifiers on the left and click **Add Selected** to add them.

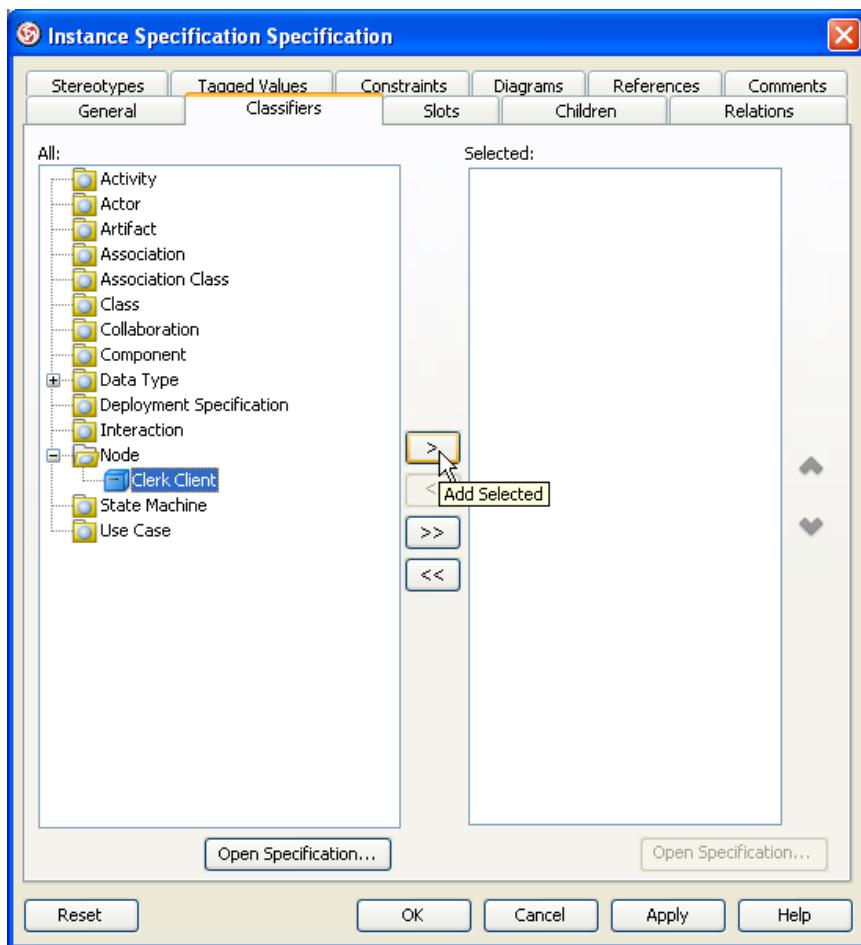


Figure 6-23 Add selected classifiers

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

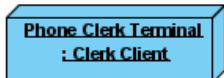


Figure 6-24 Classifiers assigned

Creating link

To create link from instance specification, click the **Link -> Instance Specification** resource beside it and drag.



Figure 6-25 Create link

Drag to empty space of the diagram to create a new instance specification, or drag to an existing instance specification to connect to it. Release the mouse button to create the link.



Figure 6-26 Link created

Creating instance of component

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

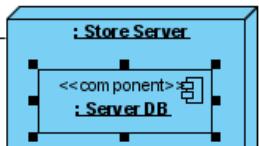


Figure 6-27 Instance of component

Creating dependency

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

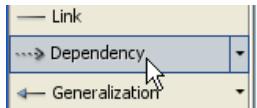


Figure 6-28 Create dependency

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

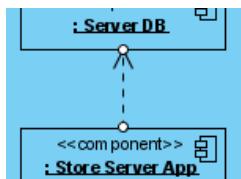


Figure 6-29 Dependency created

Continue to complete the diagram.

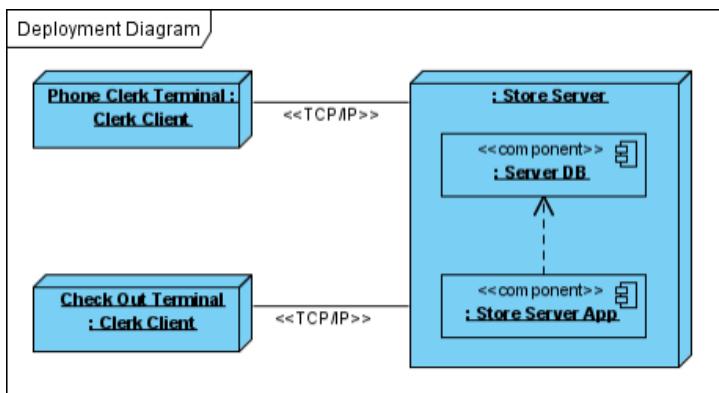


Figure 6-30 Completed diagram

Modeling and documenting test cases

Produce test case from requirement

1. In a requirement diagram, move the mouse cursor over the requirement that we want to produce **Test Case**.

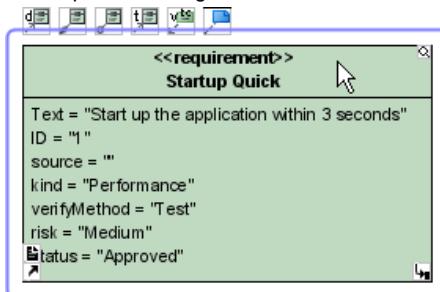


Figure 7-1 Moving mouse cursor over requirement

2. Press on the **Verify <- Test Case** resource of requirement and drag.

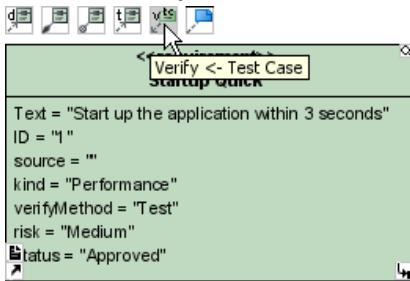


Figure 7-2 Create test case through the resource centric interface

3. Release the mouse button to create a test case. Name it.

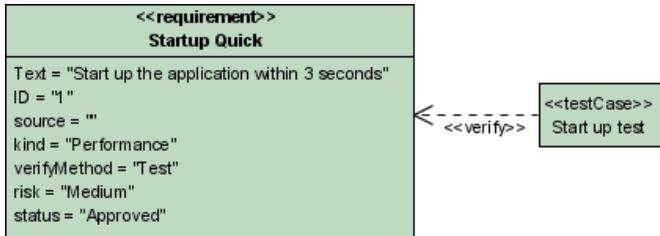


Figure 7-3 Test case is created

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.

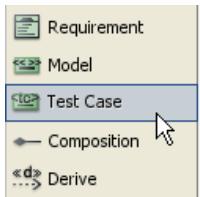


Figure 7-4 Creating Test Case

2. Press on the **Verify -> Requirement** resource of test case and drag.

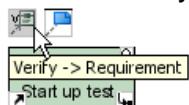


Figure 7-5 Linking Requirement with Test Case

3. Move the mouse over a requirement and then release the mouse button, a **Verify** relationship will be created from the test case to the requirement.

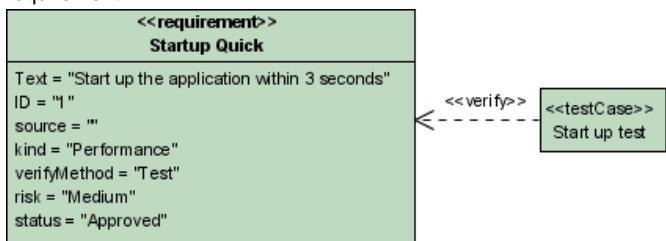


Figure 7-6 Verify relationship created

Documenting test case

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

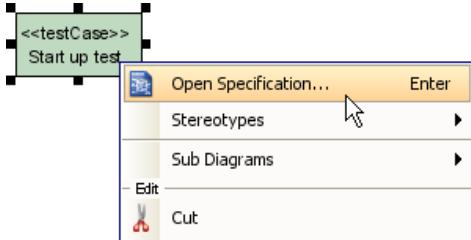


Figure 7-7 Open specification of test case

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

Steps	Procedures	Expected Results
1. Start the application	Select [MyApp] from Start menu	Prompt for checking updates
2. Reject checking update	Click [No]	The application starts within 3 seconds
3. Rerun the above steps	Rerun steps 1 and 2 five times	All attempts can start within 3 seconds

The dialog has tabs for Tagged Values, Constraints, Diagrams, References, Comments, General, Test Plans, Relations, and Stereotypes. Buttons at the bottom include Reset, OK, Cancel, Apply, and Help.

Figure 7-8 Test Plan filled

Drawing business process diagrams

Creating business process diagram

Select menu **File > New Diagram > Business Process Modeling > Business Process Diagram** to create a business process diagram.

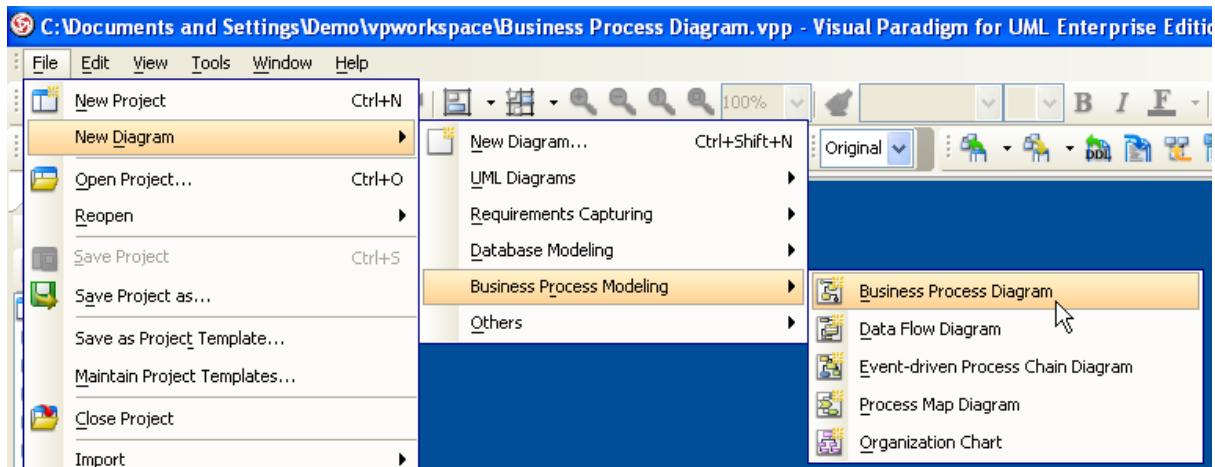


Figure 1-1 Create business process diagram

Creating pool

To create pool, click **Horizontal Pool** on the diagram toolbar and then click on the diagram.

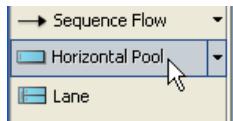


Figure 1-2 Create pool

Adding lane to pool

To add lane to pool, right-click the pool and select **Add Lane** from the popup menu.

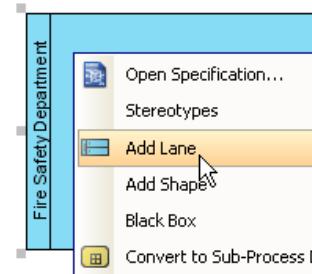


Figure 1-3 Create lane

Creating start event

To create start event, click **Start Event** on the diagram toolbar and then click on the diagram.



Figure 1-4 Create start event

Creating task

To create task from a start event, click on the **Sequence Flow -> Task** resource beside it and drag.

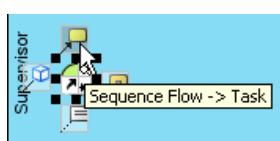


Figure 1-5 Create task

Move the mouse to where you want to place the shape to and then release the mouse button. A task is created and connected to the start event with a sequence flow.



Figure 1-6 Task and sequence flow created

Creating message flow

To create message flow, click **Message Flow** on the diagram toolbar.



Figure 1-7 Create message flow

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

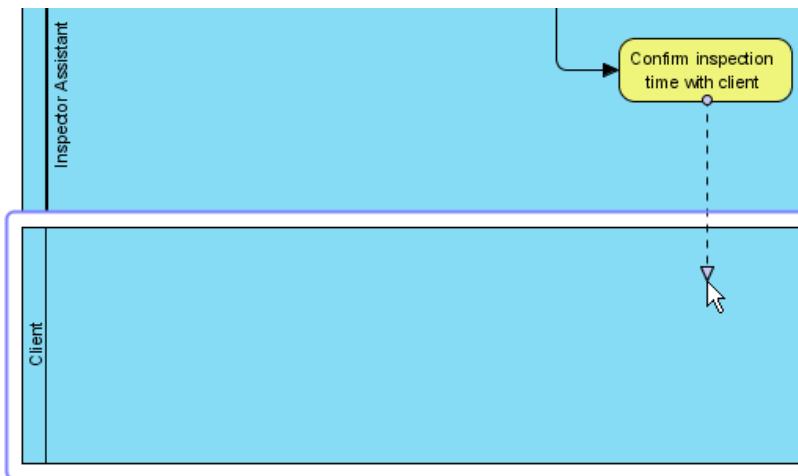


Figure 1-8 Connect shapes with message flow

Automatic connection rule checking

VP-UML provides automatic connection rule checking to make sure your business process diagram is in compliance with the BPMN specification.

For example, if you move a task from one pool to another, and if this task is connected with another shape with sequence flow, the **Invalid Connection Detected** dialog box shows to advice you to change sequence flow to message flow.

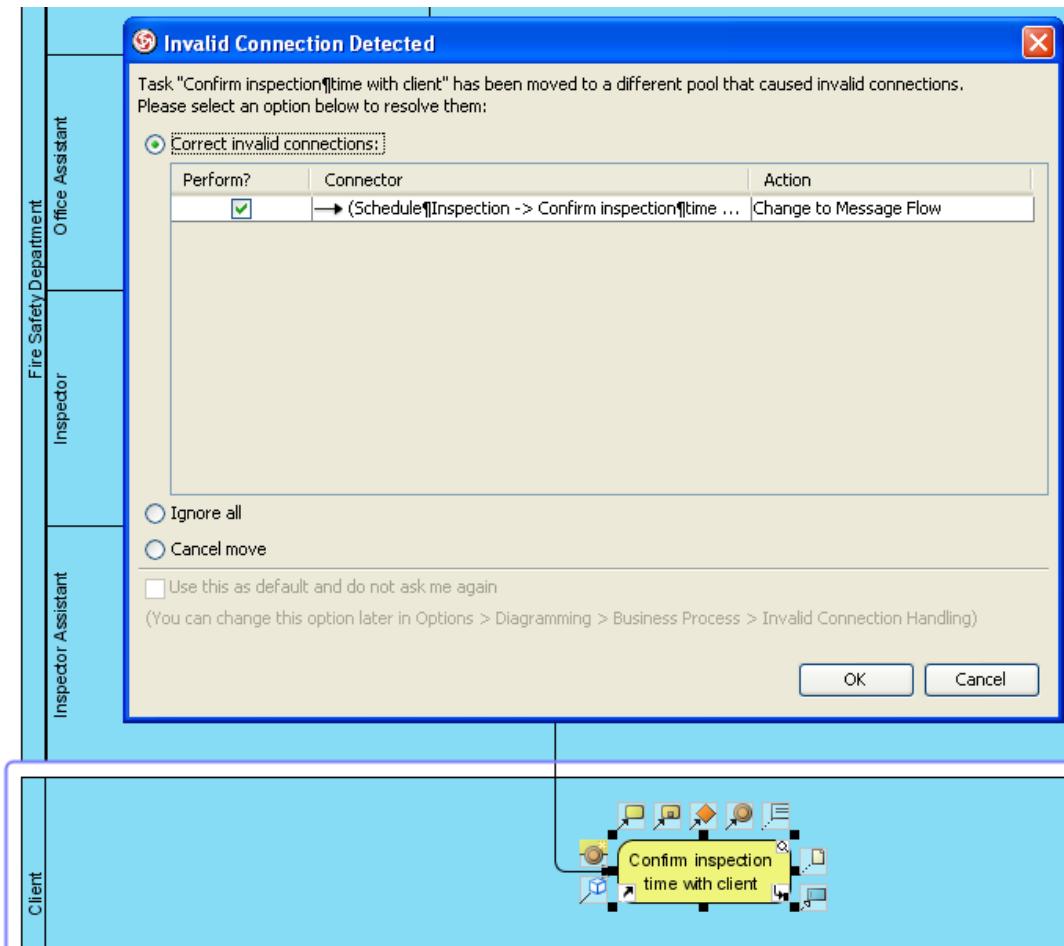


Figure 1-9 Invalid connection detected

Click **OK** to accept the correction. The sequence flow is changed to message flow automatically.

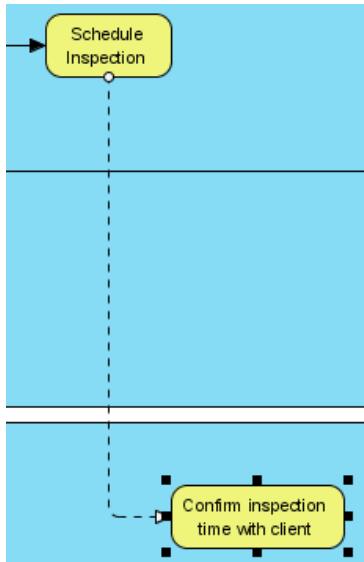


Figure 1-10 Sequence flow changed to message flow

Creating gateway

To create gateway from a task, click on the **Sequence/Message Flow -> Gateway** resource beside it and drag.



Figure 1-11 Create gateway

Move the mouse to where you want to place the shape to and then release the mouse button. A gateway is created and connected to the task with a sequence flow.

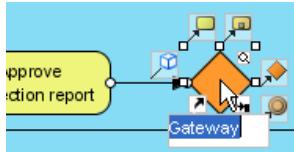


Figure 1-12 Gateway and sequence flow created

Creating intermediate event

To create intermediate event from a task, click on the **Sequence/Message Flow -> Intermediate Event** resource beside it and drag.



Figure 1-13 Create intermediate event

Move the mouse to where you want to place the shape to and then release the mouse button. An intermediate event is created and connected to the task with a sequence flow.

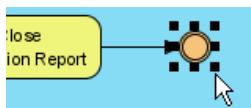


Figure 1-14 Intermediate event and sequence flow created

Creating text annotation

To create text annotation from a task, click on the **Association -> Text Annotation** resource beside it and drag.



Figure 1-15 Create text annotation

Move the mouse to where you want to place the shape to and then release the mouse button. A text annotation is created and connected to the task with an association.

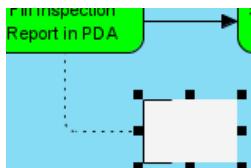


Figure 1-16 Text annotation and association created

Splitting sequence flow with shape

You can use the split sequence flow utility to split a sequence flow by task, sub-process, intermediate event or gateway.

Before this, you may want to move some shapes aside to make room for the new shape. Select **Sweeper** on the diagram toolbar.

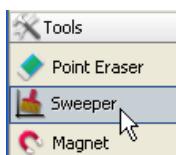


Figure 1-17
Sweeper

Click on the diagram and drag to sweep shapes aside, release the mouse button when enough room is made.

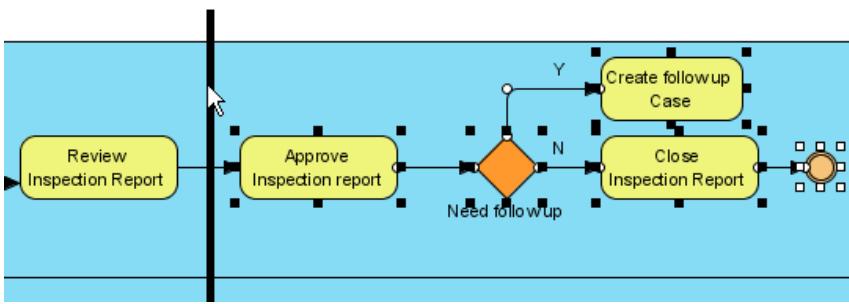


Figure 1-18 Sweep shapes aside to make more room

Click the **Split with Shape** resource beside the sequence flow.

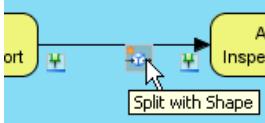


Figure 1-19 Split with Shape resource

Select the shape type to split the sequence flow.

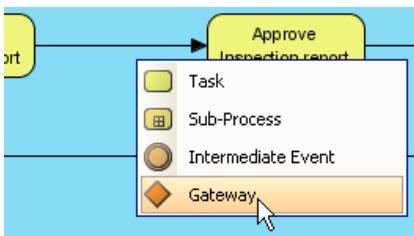


Figure 1-20 Select shape type to split sequence flow

The sequence flow is split by the selected type of shape.



Figure 1-21 Sequence flow split by the selected type of shape

Forming sub-Process diagram

To create sub-process for shapes, select the shapes in diagram, right-click on the selection and select **Form Sub-Process Diagram** from the popup menu.

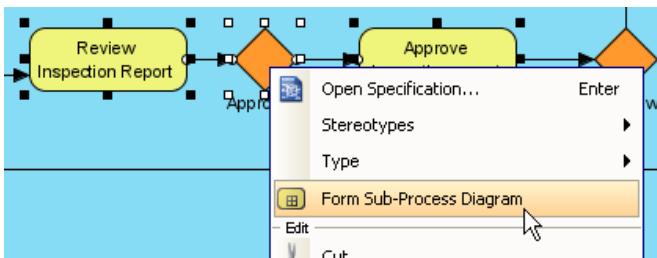


Figure 1-22 Form sub-process diagram

A new diagram is created with the selected shapes.

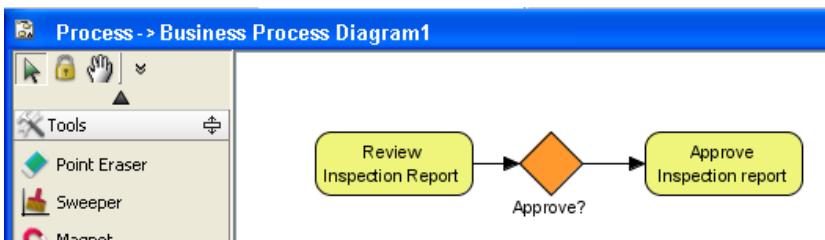


Figure 1-23 Form sub-process diagram

Return to the original diagram, you will see the selected shapes are transformed to a sub-process.

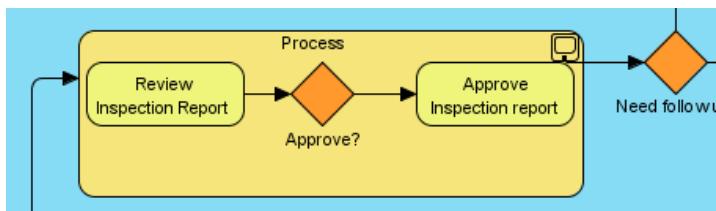


Figure 1-24 Shapes transformed to sub-process

Continue to complete the diagram.

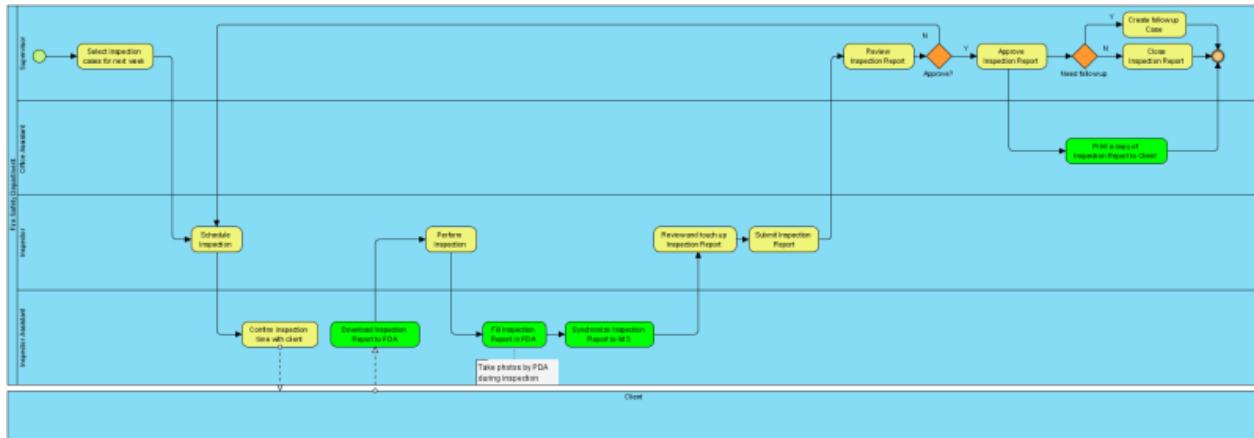


Figure 1-25 Completed diagram

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 **Options** dialog box by selecting **Tools > Options** from the main menu. Select **Diagramming** from the list on the left hand side, and open the **Business Process** tab. From there you can adjust the format, and control whether or not to display the ID on diagram.

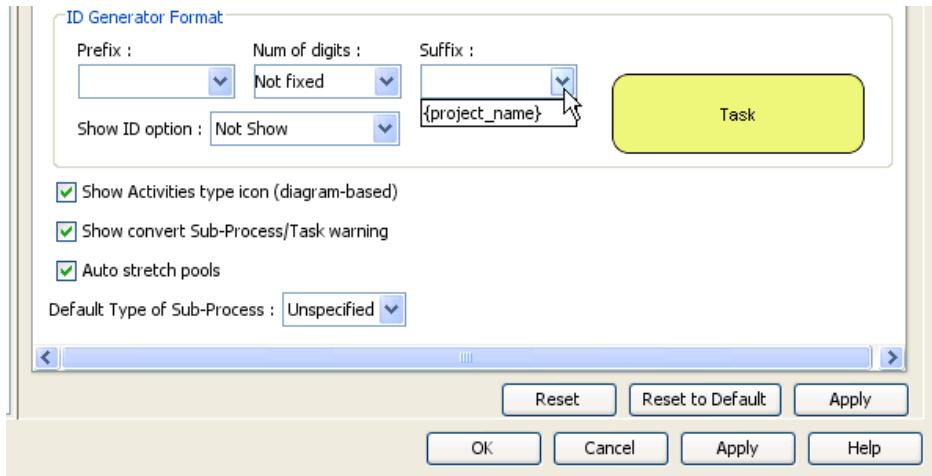


Figure 1-26 Defining format of ID

Below is a description of options.

Option	Description
Prefix	Text to add 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 append to the number
Show ID option	Whether or not to show ID on diagram, and whether to show it as label that attach to a shape, or as text below caption

Table 1-1 Options for formatting ID

Showing ID on diagram

By default, ID is just a text property that won't appear on diagram. However, you can make it appear either near or within a shape.



Figure 1-27 Different looks of a task when ID is not shown, ID is shown as label and ID is shown below caption

To make it appear, you can either set the global option (refer to the previous section), or set it through diagram's option by right clicking on the business process diagram, selecting **Presentation Options > Show ID of Elements**, and then the type of presenting the ID.

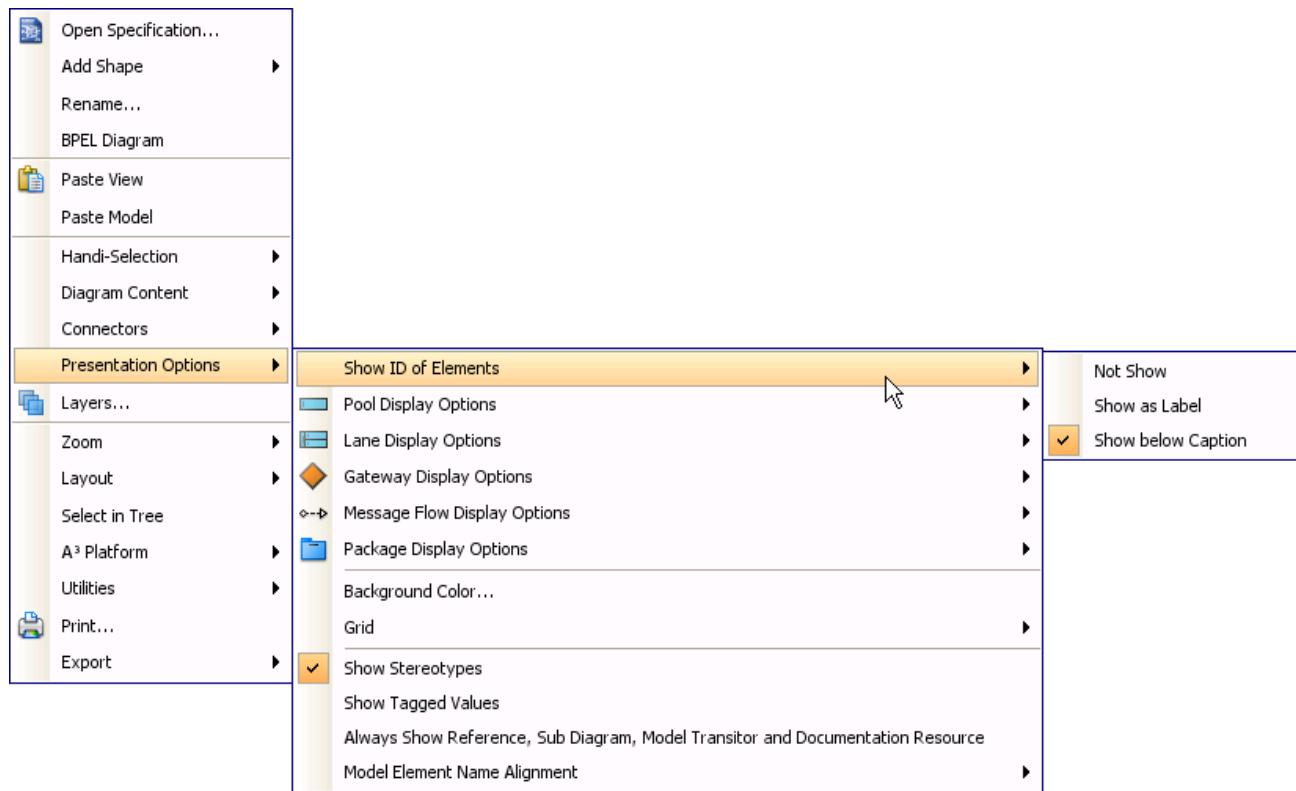


Figure 1-28 To show ID on a diagram

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**

Setting task/sub-process/gateway/event type

To change the type of task/sub-process/gateway/event, right click on the shape and select **Type**, then the type to set from popup menu.

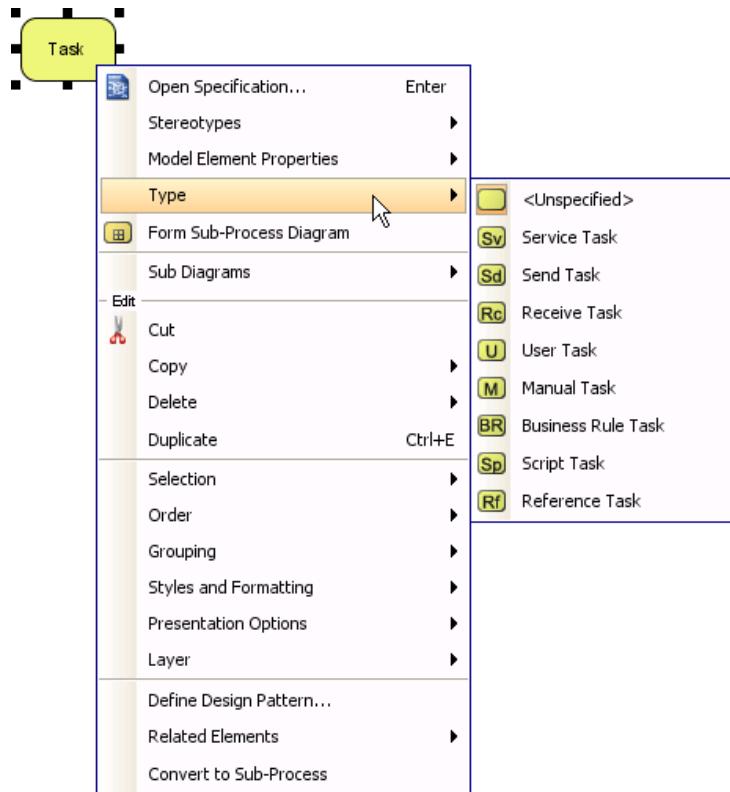


Figure 1-29 To select task type

Setting loop type for task and sub-process

To set the loop type for task/sub-process, right click on the shape and select Open Specification... from the popup menu. In the specification dialog box, update the **Loop Type** selection and click **OK** to confirm editing.

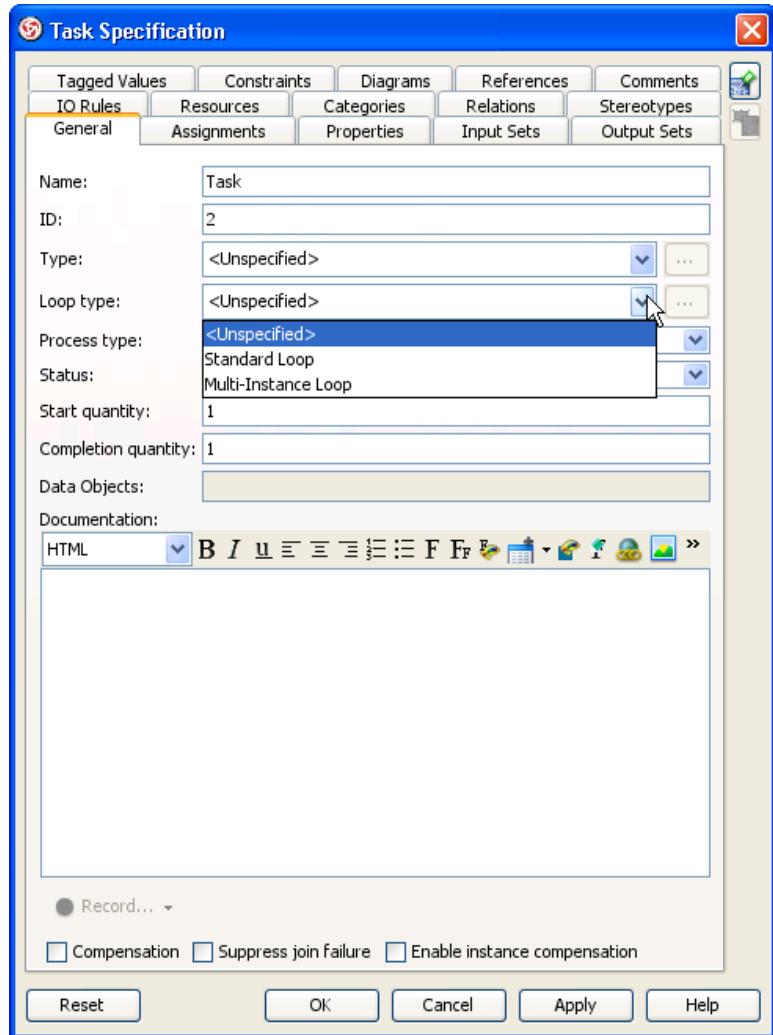


Figure 1-30 To edit the loop type

Drawing data flow diagrams

Creating data flow diagram

Select menu **File > New Diagram > Business Process Modeling > Data Flow Diagram** to create a data flow diagram.

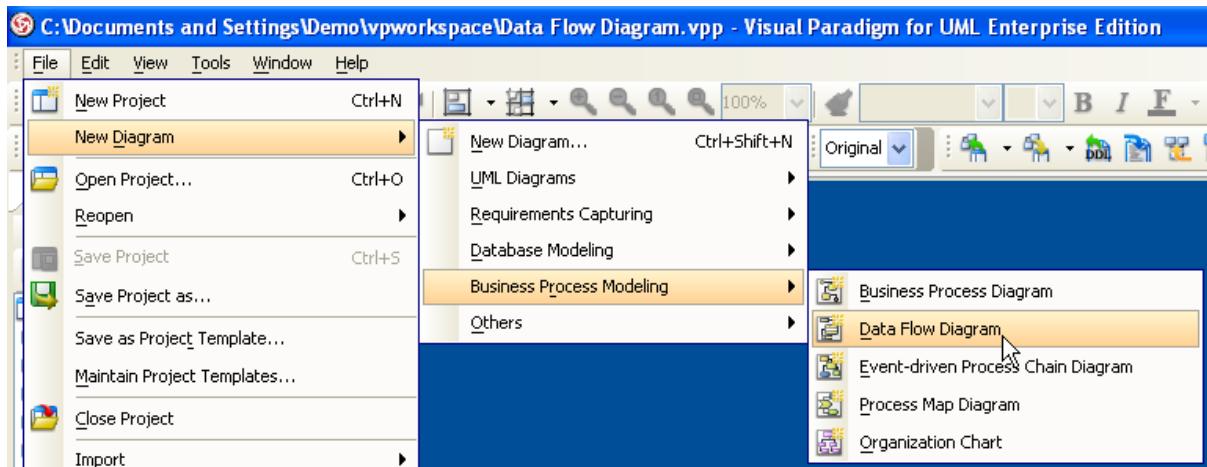


Figure 1-31 Create data flow diagram

Creating external entity

To create external entity, click **External Entity** on the diagram toolbar and then click on the diagram.

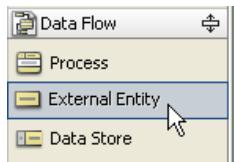


Figure 1-32 Create external entity

An external entity is created.

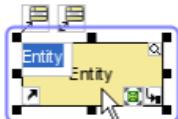


Figure 1-33 External entity created

Creating process

To create process from external entity, click the **Data Flow -> Process** resource beside it and drag.



Figure 1-34 Create process

Move the mouse to empty space of the diagram and then release the mouse button, a process will be created and connected to the external entity with a data flow.

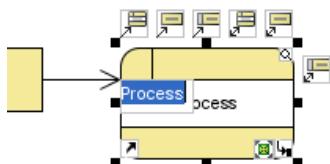


Figure 1-35 Process and data flow created

To edit properties of the process, right-click on it and select **Open Specification** from the popup menu.

Edit properties such as ID and Location in the **Process Specification** dialog box, click **OK** to apply the changes.

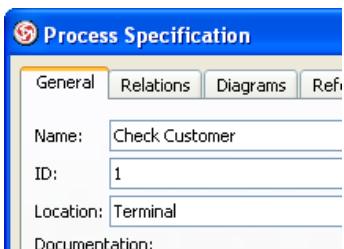


Figure 1-36 Process specification dialog box

The process is updated.



Figure 1-37 Process updated

Creating data store

To create data store from process, click the **Data Flow -> Data Store** resource beside it and drag.

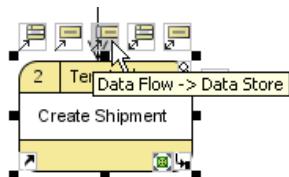


Figure 1-38 Create data store

Move the mouse to empty space of the diagram and then release the mouse button, a data store will be created and connected to the process with a data flow.

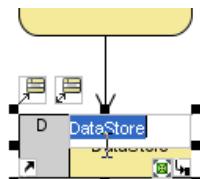


Figure 1-39 Data store and data flow created

Continue to complete the diagram.

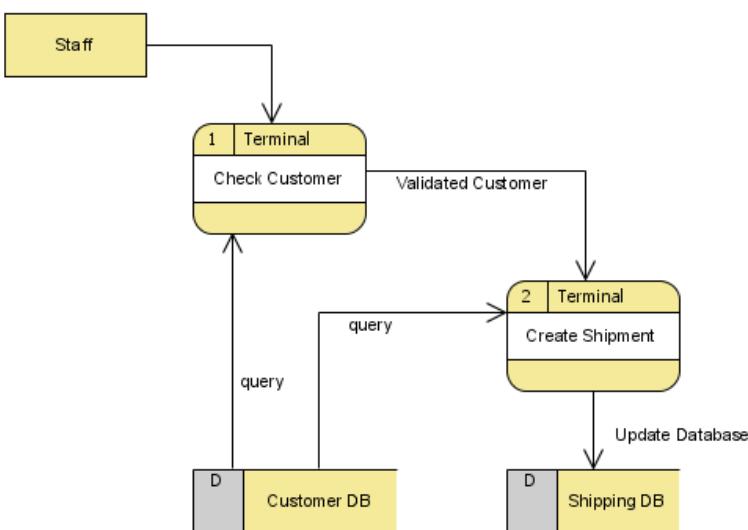


Figure 1-40 Completed diagram

Drawing event-driven process chain diagrams

Creating data flow diagram

Select menu **File > New Diagram > Business Process Modeling > Event-driven Process Chain Diagram** to create an event-driven process chain diagram.

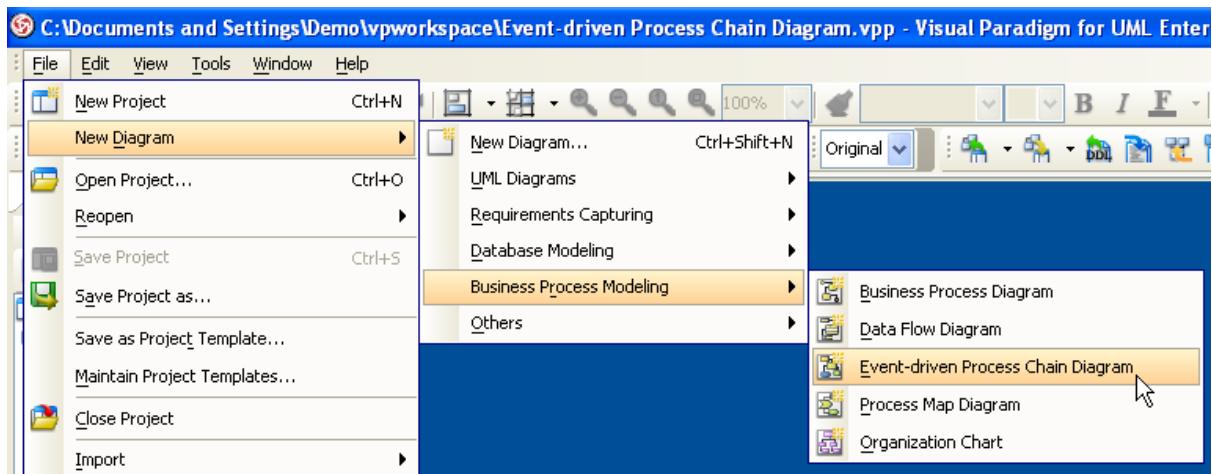


Figure 1-41 Create event-driven process chain diagram

Creating event

To create event, click **Event** on the diagram toolbar and then click on the diagram.

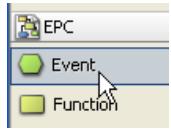


Figure 1-42 Create event

An event will be created.



Figure 1-43 Event created

Creating function

To create function from event, click the **Control Flow -> Function** resource beside it and drag .

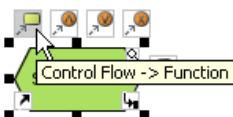


Figure 1-44 Create function

Move the mouse to empty space of the diagram and then release the mouse button, a function will be created and connected to the event with a control flow.

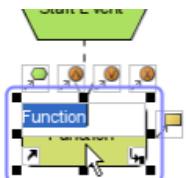


Figure 1-45 Function and control flow created

Creating operator

To create operator (e.g. XOR operator) from event, click the **Control Flow -> XOR Operator** resource beside it and drag .



Figure 1-46 Create operator

Move the mouse to empty space of the diagram and then release the mouse button, an operator will be created and connected to the function with a control flow.

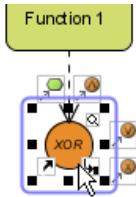


Figure 1-47 Operator and control flow created

Creating process path

To create process path from event, click the **Control Flow -> Process Path** resource beside it and drag .

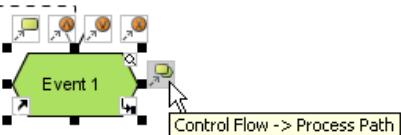


Figure 1-48 Create process path

Move the mouse to empty space of the diagram and then release the mouse button, a process path will be created and connected to the event with a control flow.

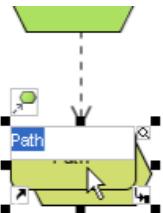


Figure 1-49 Process path and control flow created

Creating information resource

To create information resource, click **Information Resource** on the diagram toolbar and then click on the diagram.

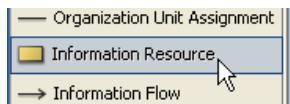


Figure 1-50 Create information resource

An information resource will be created. To connect the information resource with a function using information flow, click the **Information Flow -> Function** resource and drag .

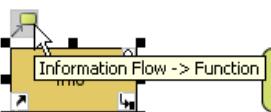


Figure 1-51 Create information flow

Mouse over a function and then release the mouse button, an information flow will be created.

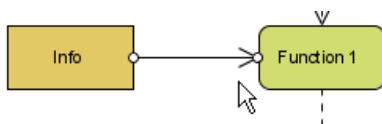


Figure 1-52 Information flow created

Continue to complete the diagram.

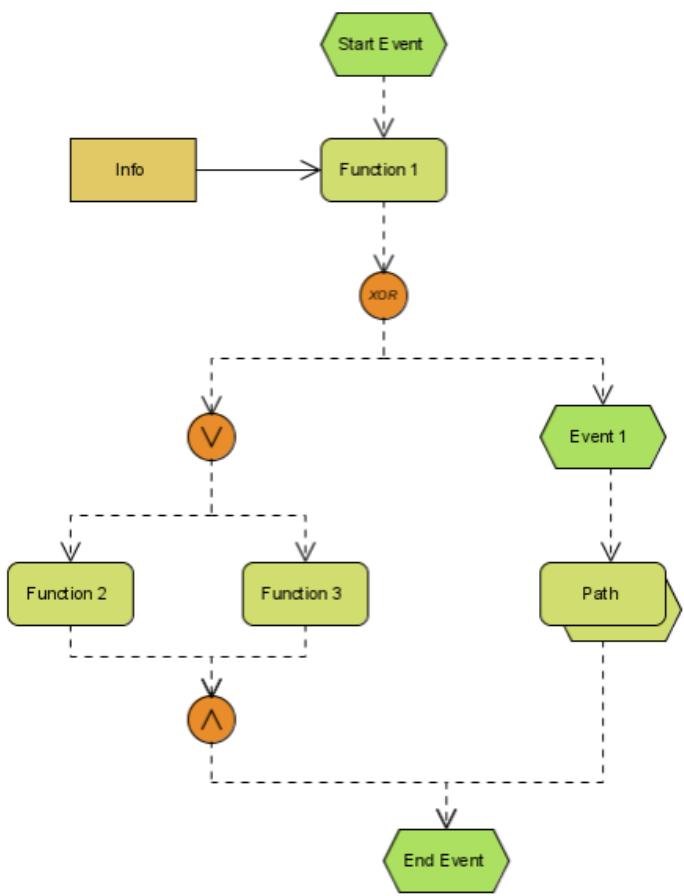


Figure 1-53 Completed diagram

Drawing process map diagram

Creating process map diagram

Select menu **File > New Diagram > Business Process Modeling > Process Map Diagram** to create a process map diagram.

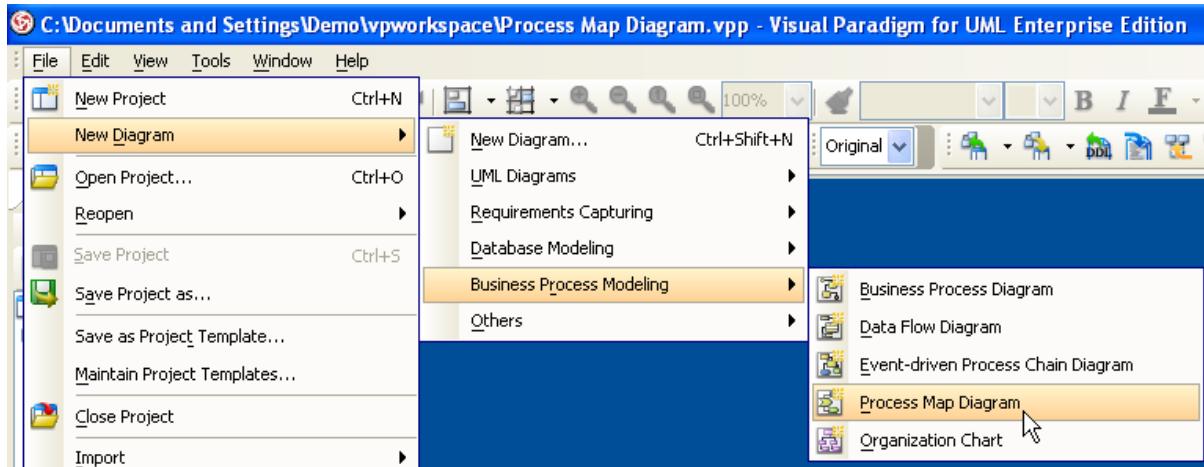


Figure 1-54 Create process map diagram

Creating receive

To create receive, click **Receive** on the diagram toolbar and then click on the diagram.

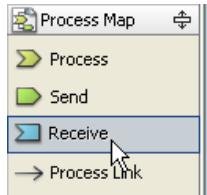


Figure 1-55 Create receive

A receive will be created.

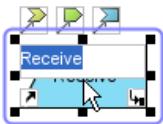


Figure 1-56 Receive created

Creating process

To create process from receive, click the **Process Link -> Process** resource beside it and drag .



Figure 1-57 Create process

Move the mouse to empty space of the diagram and then release the mouse button, a process will be created and connected to the receive with a process link.

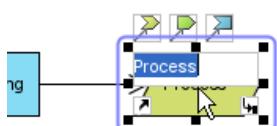


Figure 1-58 Process created

Creating send

To create send from process, click the **Process Link -> Send** resource beside it and drag .

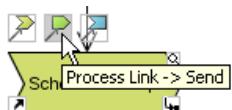


Figure 1-59 Create send

Move the mouse to empty space of the diagram and then release the mouse button, a send will be created and connected to the process with a process link.

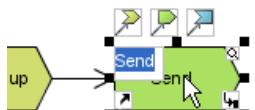


Figure 1-60 Send created

Continue to complete the diagram.

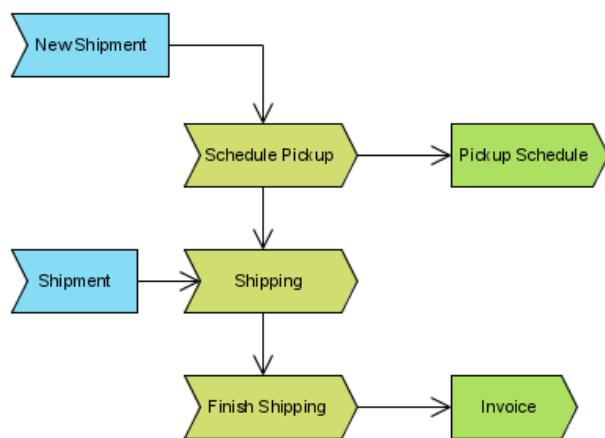


Figure 1-61 Completed diagram

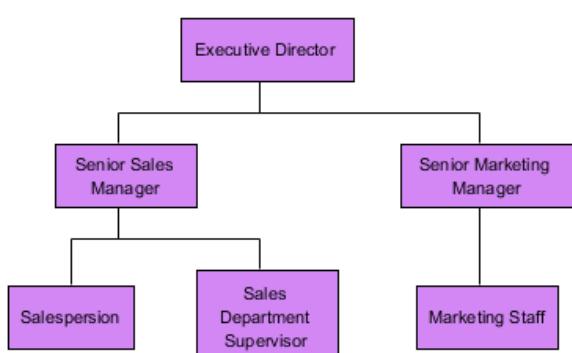
Drawing organization charts

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 the top to the bottom. The default unit will pop out when a new organization chart is created.

In VP-UML, 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.

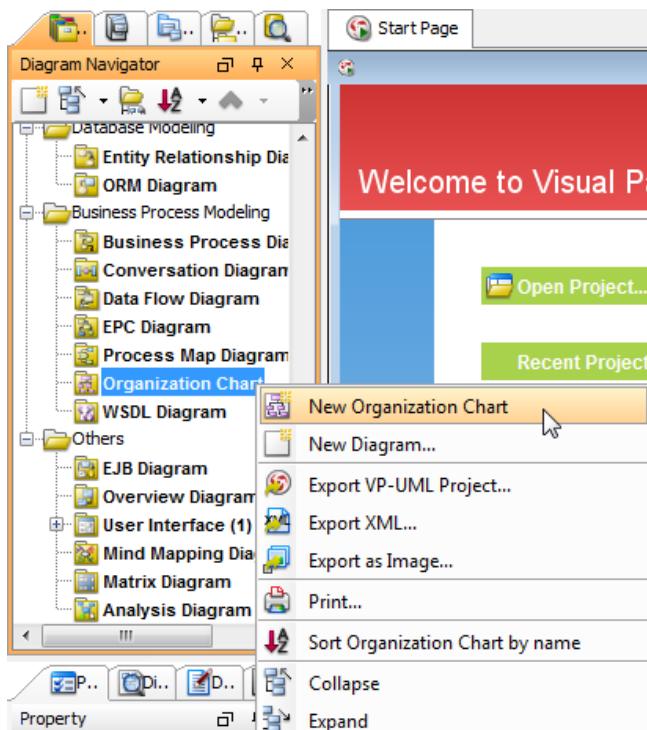
The completed organization chart is shown as follows:



Completed organization chart

Creating an organization chart

Right Click Organization Chart in Diagram Navigator and select **New Organization Chart** from the popup menu.

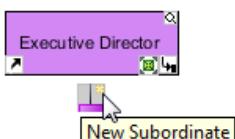


Perform new organization chart

Creating a subordinate

Subordinate, which is subject 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**.



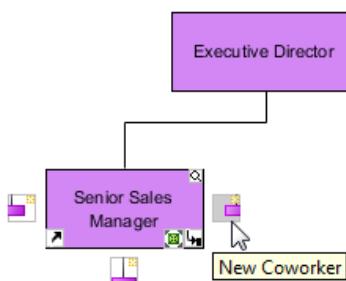
Create subordinate

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:

- Move the mouse pointer a unit and click its resource icon **New Coworker**, either on its left or right hand side.



Create coworker

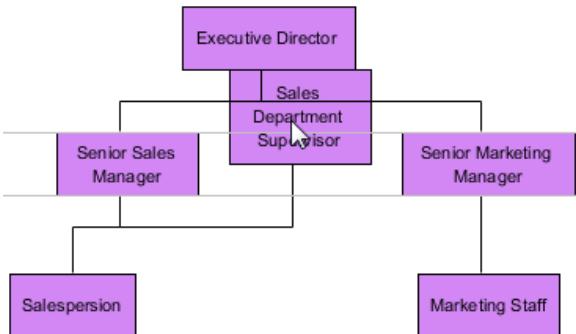
- Name the newly created coworker unit and press **Enter** to confirm editing.

NOTE: Clicking left resource icon will create coworker on the left of the unit, while clicking right resource icon will create coworker on its right.

Relocating a branch

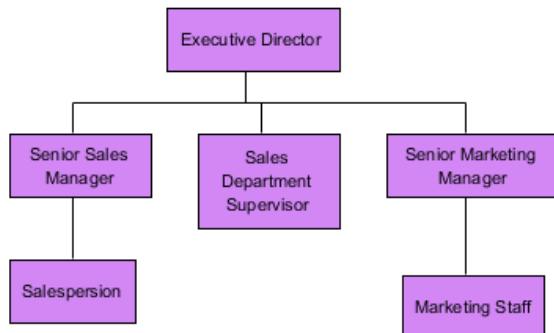
A unit can be relocated even when it has been placed under the subordination of another unit.

- Press on a branch you want to relocate and drag it to the preferred branch.



Moving a unit

- Release the mouse to confirm the position.



Completed relocating a branch

NOTE: If you are not satisfied the relocation, press **Esc** to cancel the movement.

JPDL generation

- Model the process in a business process diagram.

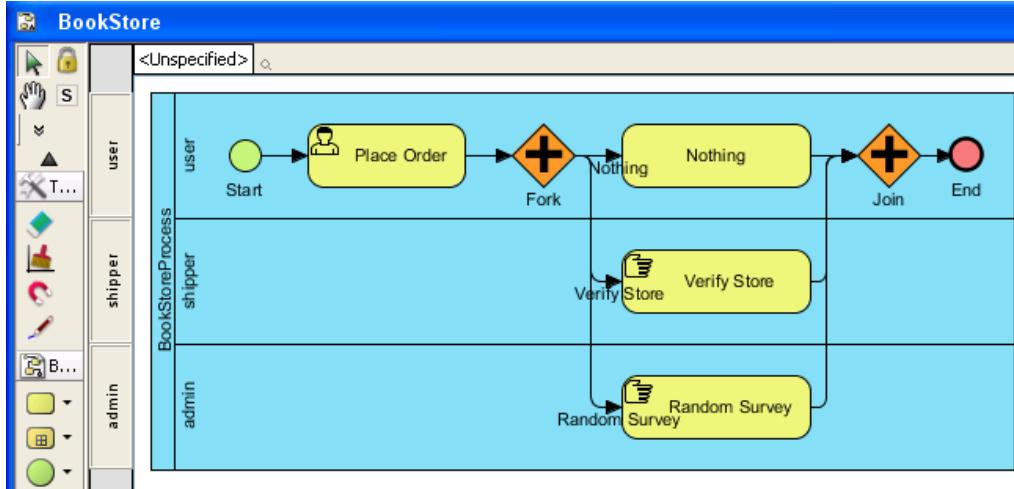


Figure 1-68 A business process diagram

- Right click on the diagram's background and select jPDL Diagram in the popup menu.

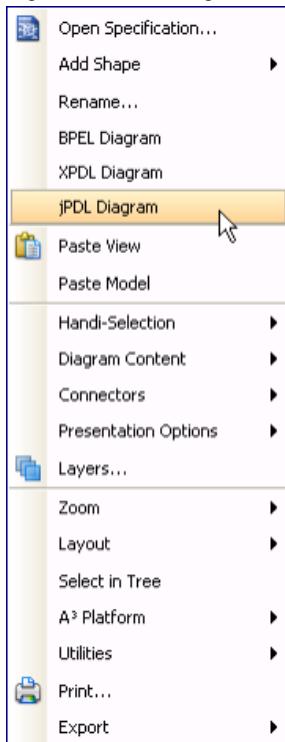


Figure 1-69 Set diagram to be a jPDL diagram

3. Right click on the diagram's background and select **Utilities > Generate jPDL...** in the popup menu.

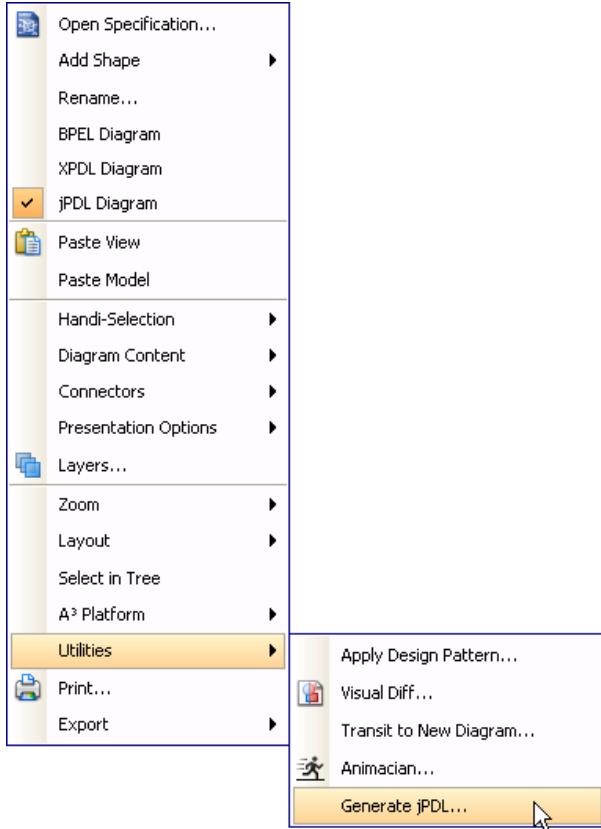


Figure 1-70 Generate jPDL

4. Select the folder to generate jPDL to. When generation is done, you should see the following files in the chosen folder:

- A zip file containing the deployable process archive
- gpd.xml (The graphical process definition)
- processdefinition.xml (The process definition)
- processimage.jpg (The image file of business process diagram)

Now, you can deploy the jPDL.

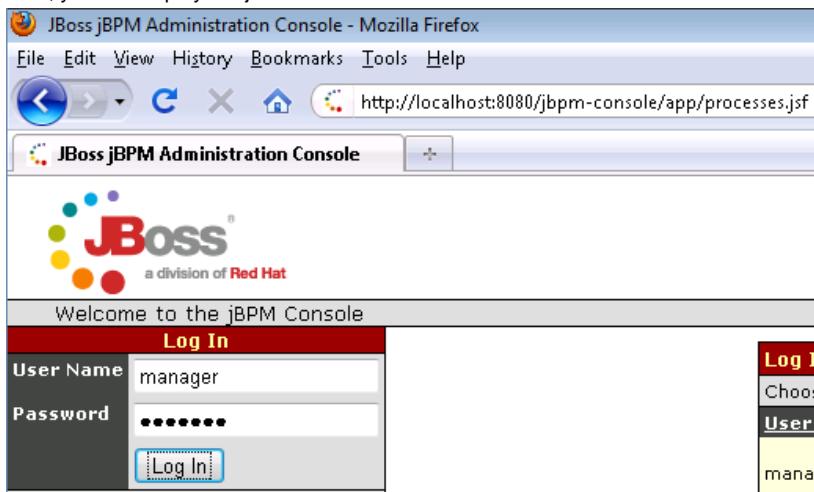
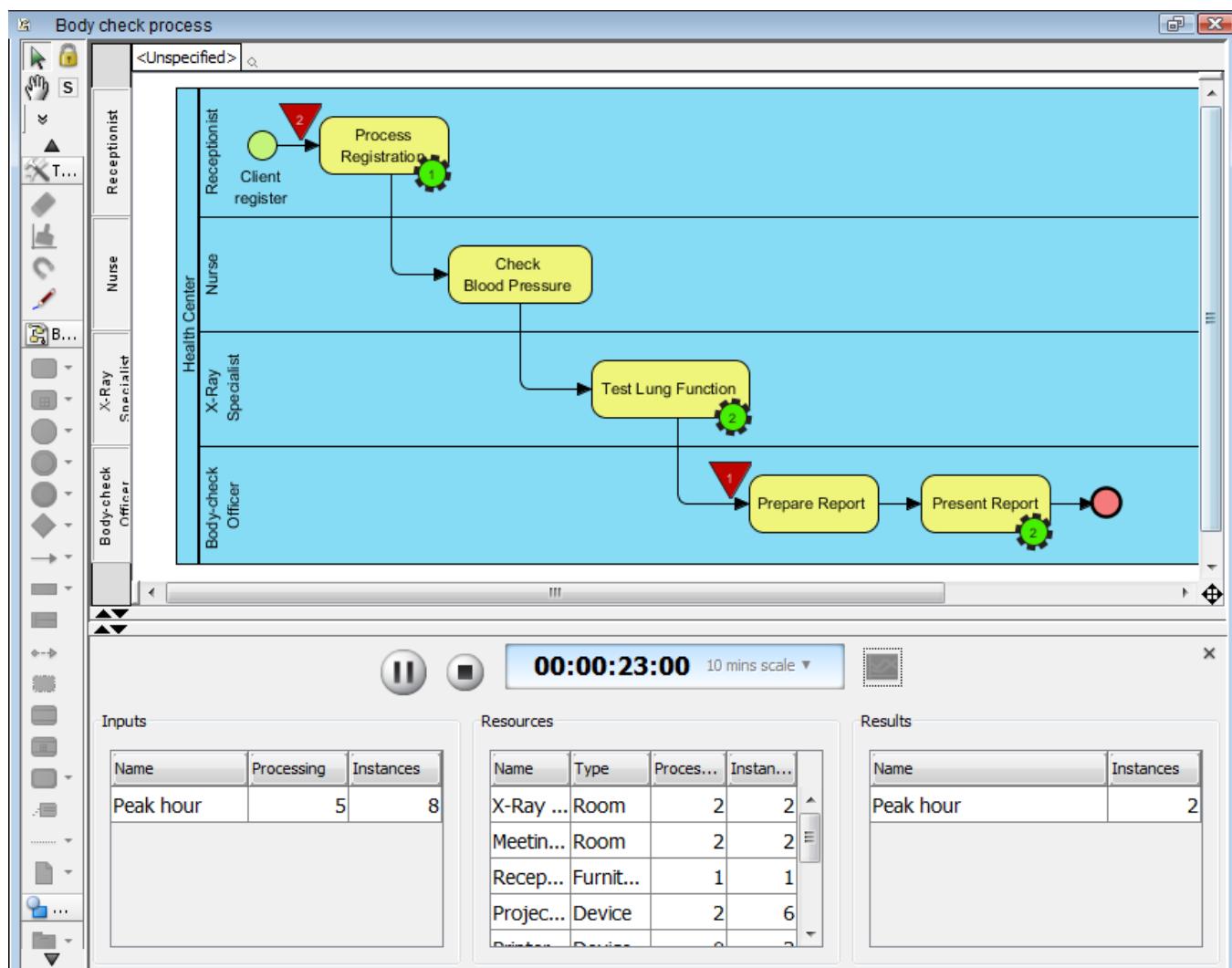


Figure 1-71 The jBPM console

What is simulacian?

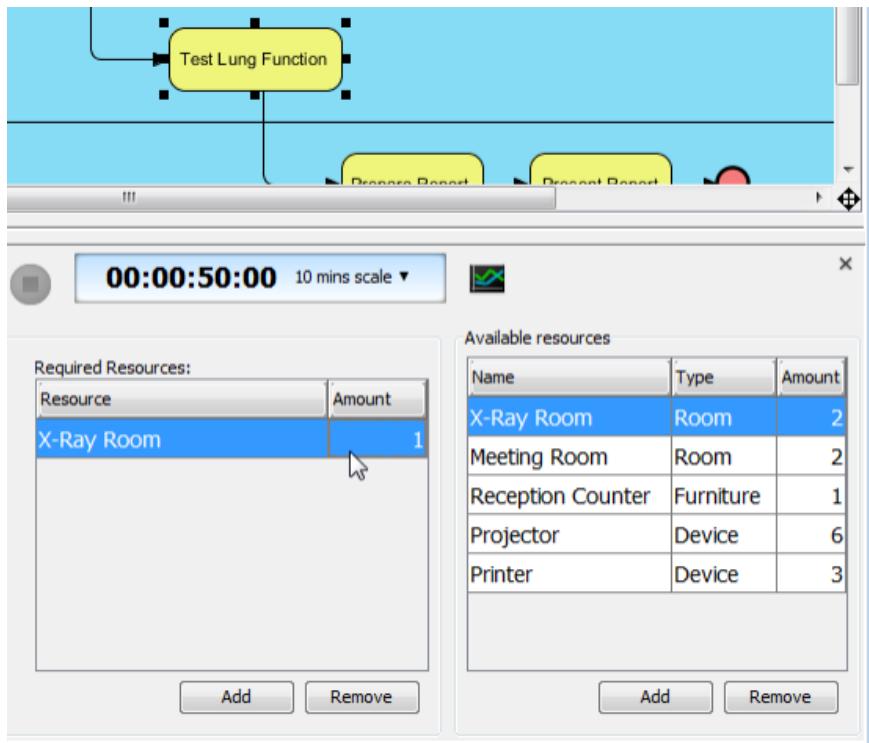
The objective of performing business process modeling is to facilitate the communication with stakeholders, to perform cost and benefit analysis and to perform process improvement, etc. Simulacian is a set of value-added tools designed to aid business process modeling. With simulacian, you can simulate 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.



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 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

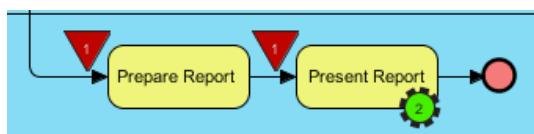
Duration is the time elapsed from the entering of flow object until the leaving of that flow object. It is understandably that the duration of flow object has significant effect to the efficiency of a business process. Imagine if it takes 5 minutes to complete the process of just one payment in a supermarket, there will accumulate a long queue waiting for paying.

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 represent 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 improve your process, you must set input that reflect 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

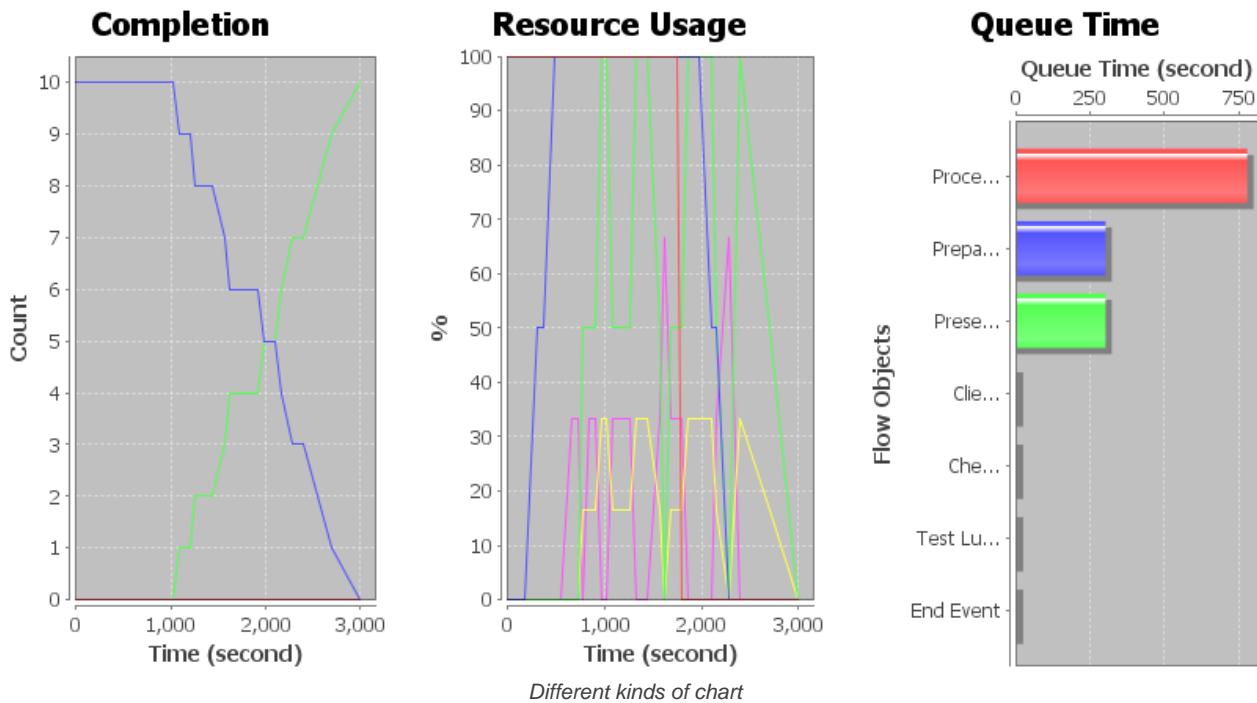
Once you have specified the available and required resources, the duration of flow objects and added input(s), you can run simulation. During simulation, diagram will be locked to avoid collision between your edit action and the simulation operation. Executing jobs are represented by a running green 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 simulacian 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 reports the completion of inputs, resource usage and queue time of flow objects against time.

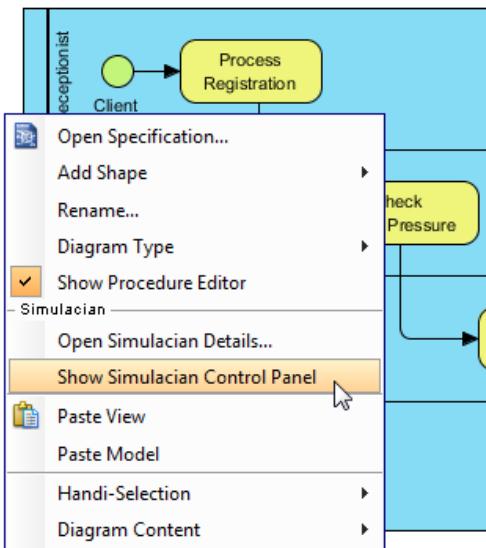


Simulacian control panel

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 inputs. All these information can be defined in simulacian control pane, which is a pane that display at the bottom of diagram, important for adjusting any settings related to simulacian. The panel will be updated base on your selection in active diagram. Besides setting simulacian details, start/pause of simulation can also be done in the panel.

Opening simulacian control panel

To open the simulacian control panel, right click on the business process diagram that you want to simulate and select **Show Simulacian Control Panel** from the popup menu.

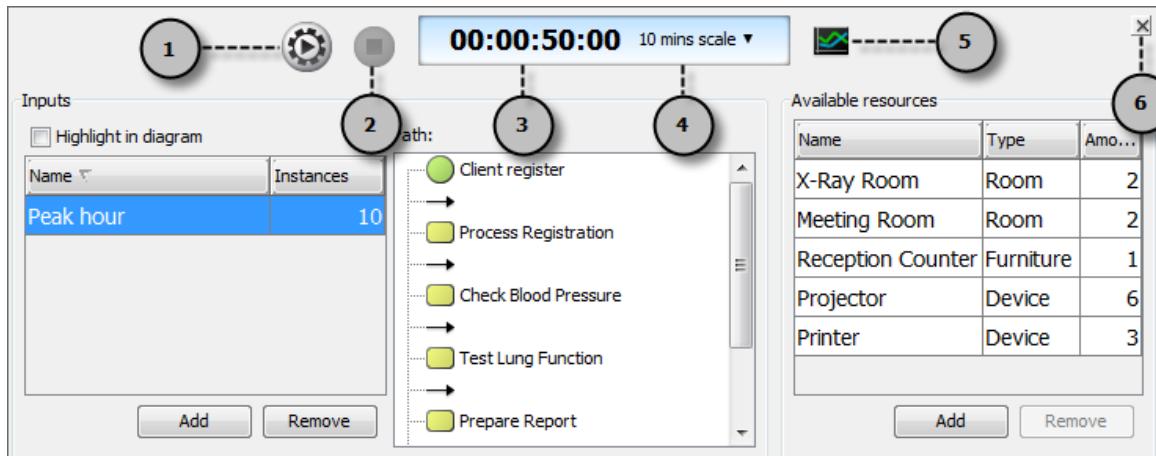


Opening simulacian control panel

NOTE: You can open simulacian control panel only for diagram that has selected **Simulacian** 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 simulacian control panel

Common



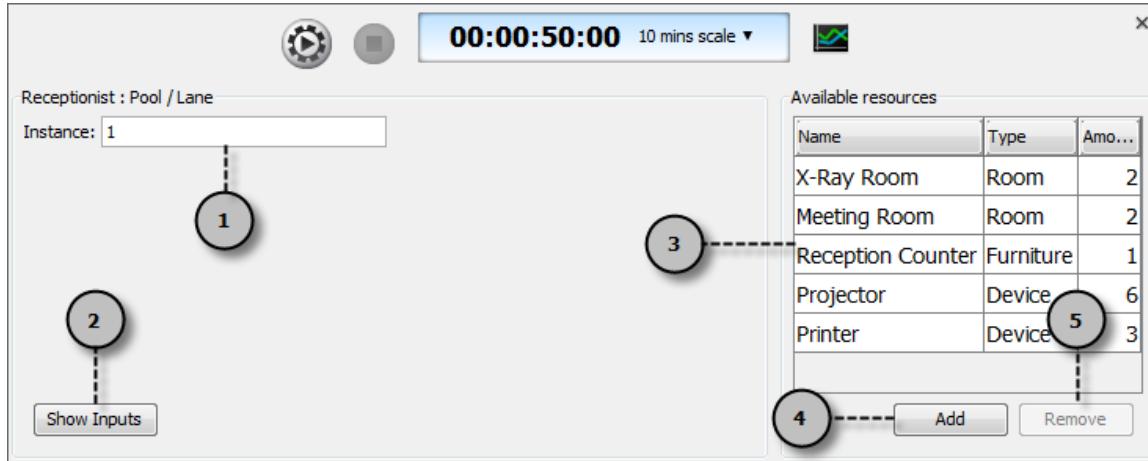
An overview of simulacian control panel

No.	Name	Description
1	Start / Pause	Click to start simulation when paused or stopped, base on the resource, duration and input settings, or to pause a simulation when it is playing.
2	Stop	Stop a simulating business process.
3	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.
4	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.

- 5 Simulacian 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 input. You can treat it as a chart form of simulation outcome.
- 6 Close Click to close the simulacian control panel. You can open it again by right clicking on the business process diagram and selecting **Show Simulacian Control Panel** from the popup menu.

Description of simulacian control panel

When selected pool/lane

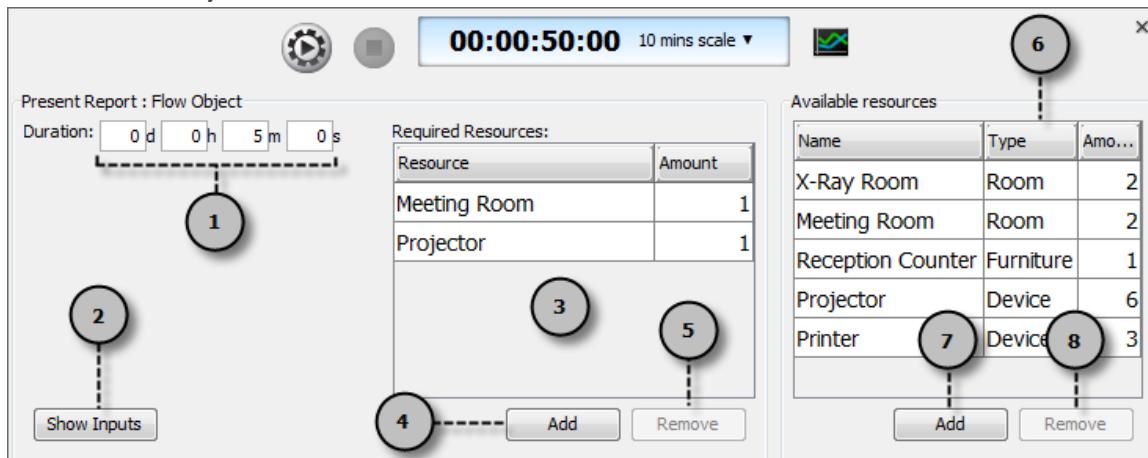


An overview of simulacian control panel while selected pool/lane

No.	Name	Description
1	Instance	<p>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 <i>Client</i> and <i>Receptionist</i>. 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 <i>Receptionist</i> to represent all receptionists instead of drawing 5 pools for representing the fact that there are 5 receptionists.</p> <p>Here the Instance field let you set the number of instances of selected pools or lanes. If there are 5 receptionists, enter 5 for instance of pool/lane <i>Receptionist</i>.</p> <p>Provided that there are sufficient resources for performing jobs, the number of instances affects the performance of process - The more instances, the more 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.</p>
2	Show inputs	Inputs are the expected way of executing a business process during simulacian. Click Show Inputs to list the inputs, and add/remove inputs 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 simulacian control panel while selected pool/lane

When selected flow objects

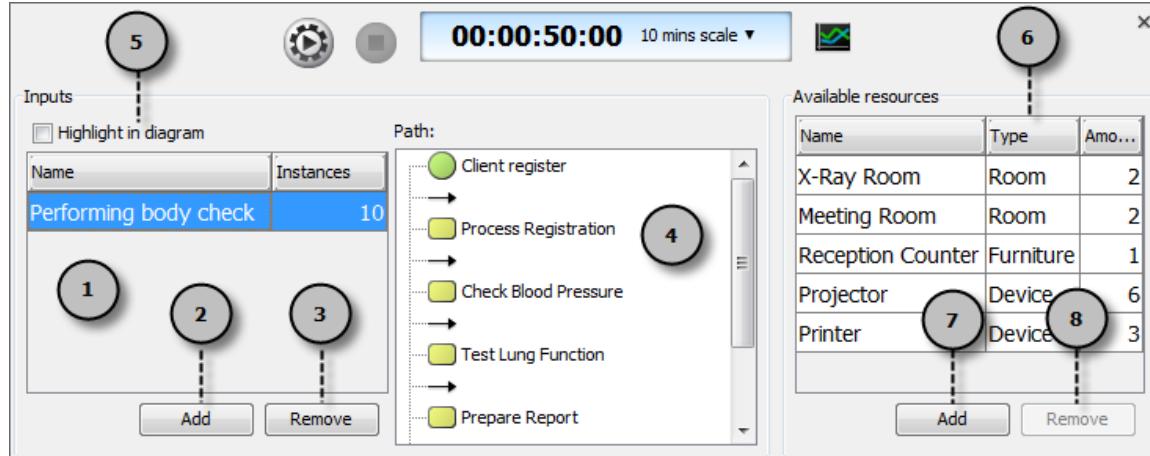


An overview of simulacian control panel while selected flow object

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, it takes five 5 minutes to present a report to client in a body check operation, set the business task <i>Present Report</i> 's duration to be 5 m, meaning 5 minutes. The duration setting affects the performance of process - The less time needed, the more efficient. 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	Show inputs	Inputs are the expected way of executing a business process during simulacian. Click Show Inputs to list the inputs, and add/remove inputs in further.
3	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 report to client, you need one resource <i>Meeting Room</i> , and one resource <i>Projector</i> .
4	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 projector (resource), the maximum of projector a flow object can require is from 0 to 6.
5	Remove required resource	Select a required resource and click this button to remove it.
6	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.
7	Add available resource	Click to add an available resource by giving its name, selecting/entering its type and setting its amount.
8	Remove available resource	Select an available resource and click this button to remove it.

Description of simulacian control panel while selected flow objects

When selected diagram background / showing inputs



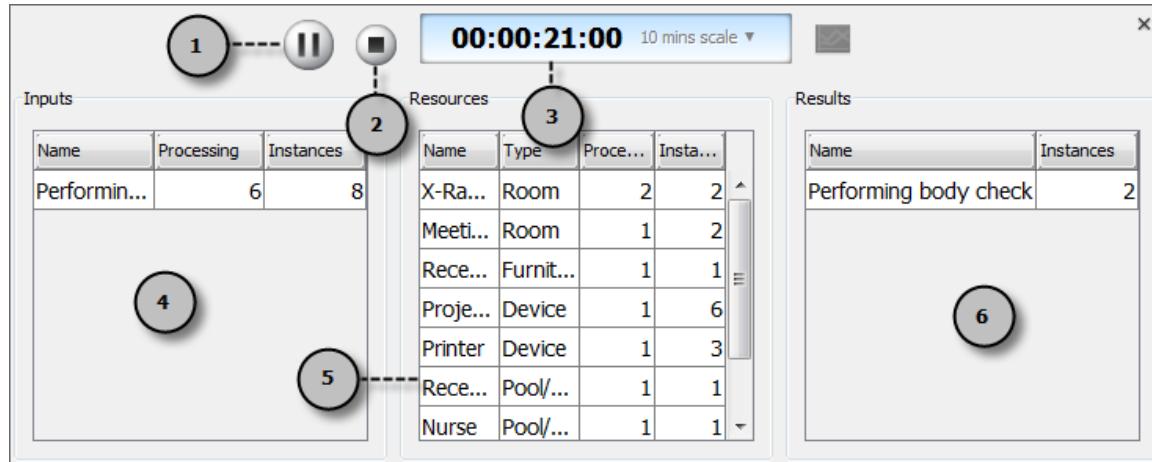
An overview of simulacian control panel while selected diagram background or showing inputs

No.	Name	Description
1	Inputs	Inputs are the definitions of how to execute a business process during simulation. An input 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 that there are 10 clients needed to perform body checking in a business process of body checking, to see if the process can handle this situation well, you will add an input <i>Performing body check</i> , with 10 as number of instances.
2	Add input	Click this button to add an input with name and number of instances.
3	Remove input	Select an input and click this button to remove it.
4	Path	The flow objects involved in an input. If there is a gateway in your diagram, you need to make a decision to the outgoing path of the gateway object.
5	Highlight in diagram	Check this button to make the diagram highlight the path involved in the input selected in Inputs table.

- | | |
|-----------------------------|--|
| 6 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. |
| 7 Add available resource | Click to add an available resource by giving its name, selecting/entering its type and setting its amount. |
| 8 Remove available resource | Select an available resource and click this button to remove it. |

Description of simulacian control panel while selected diagram background or showing inputs

When simulating



An overview of simulacian control panel while simulating

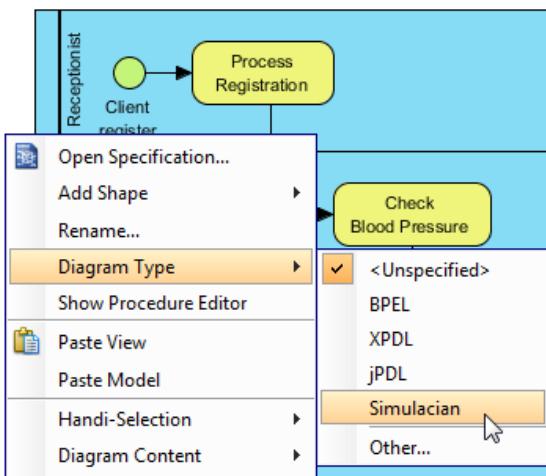
No.	Name	Description
1	Pause	To temporarily pause a simulation.
2	Stop	To stop simulating the business process.
3	Clock	Displays the time elapsed from the beginning of simulation until the current moment. It is for reflecting the duration of the execution of business process, and is based on the selection of time scale.
4	Inputs	A list of inputs 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 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 inputs. The Instances column shows the amount of completed inputs.

Description of simulacian control panel while simulating

Simlauting business process

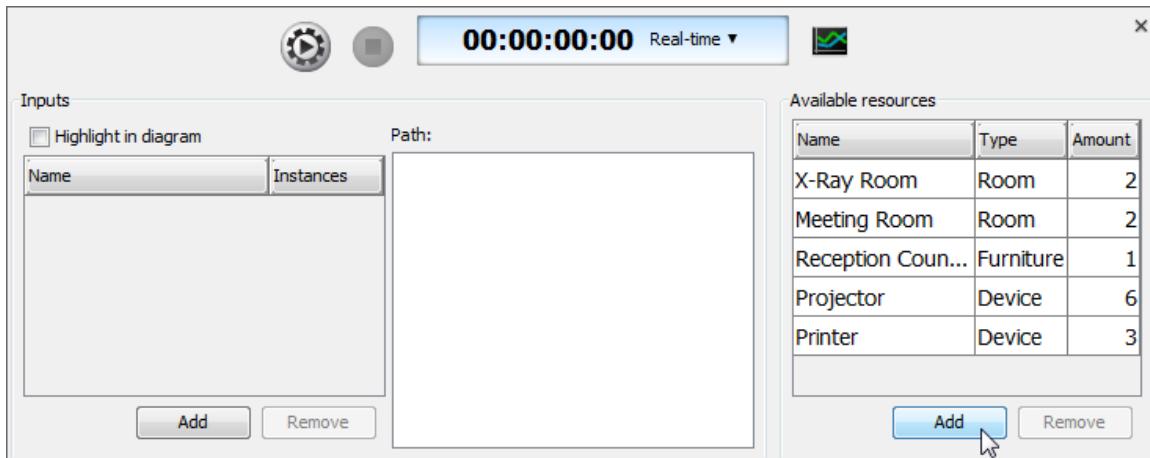
In order to simulate a business process, you must provide performance -related information to the business process diagram, such as resource consumption and duration of flow objects, so that simulacian can analyze the information and conduct a simulation. Below are what you have to do to start simulation.

1. Right click on the background of the business process diagram that you want simulate and select **Diagram Type > Simulacian** from the popup menu.



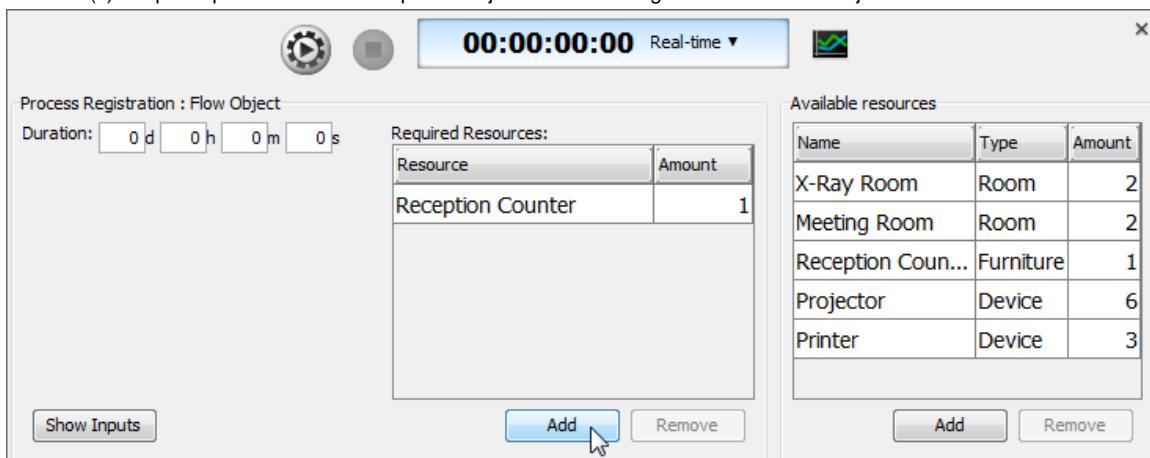
Set diagram's type to Simulacian

2. In the **Simulacian 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 simulacian?*.



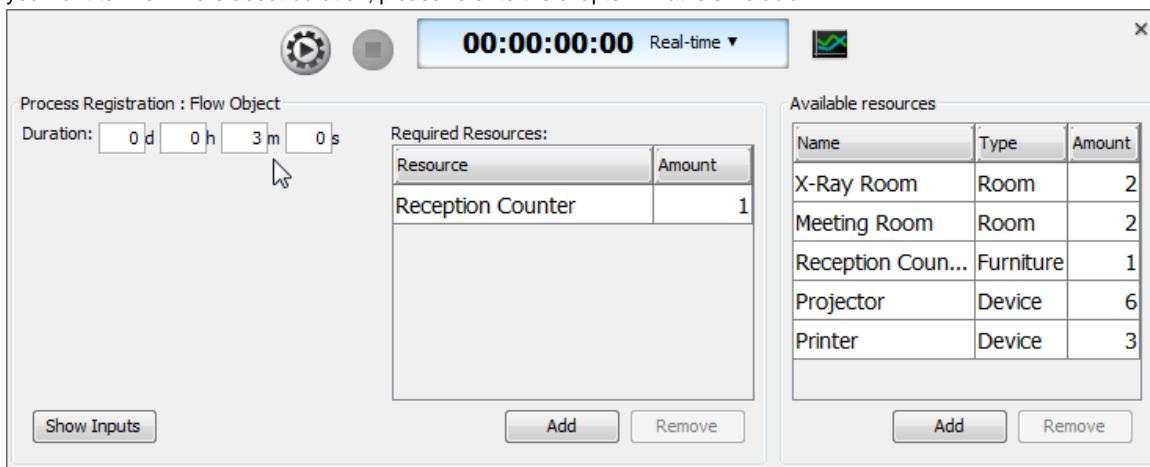
To add available resources

3. For each of the flow objects, select it in diagram, and click **Add** under **Required Resources** in **Simulacian Control Panel** to add the resource(s) the participant needed to complete the job when reaching the selected flow object.



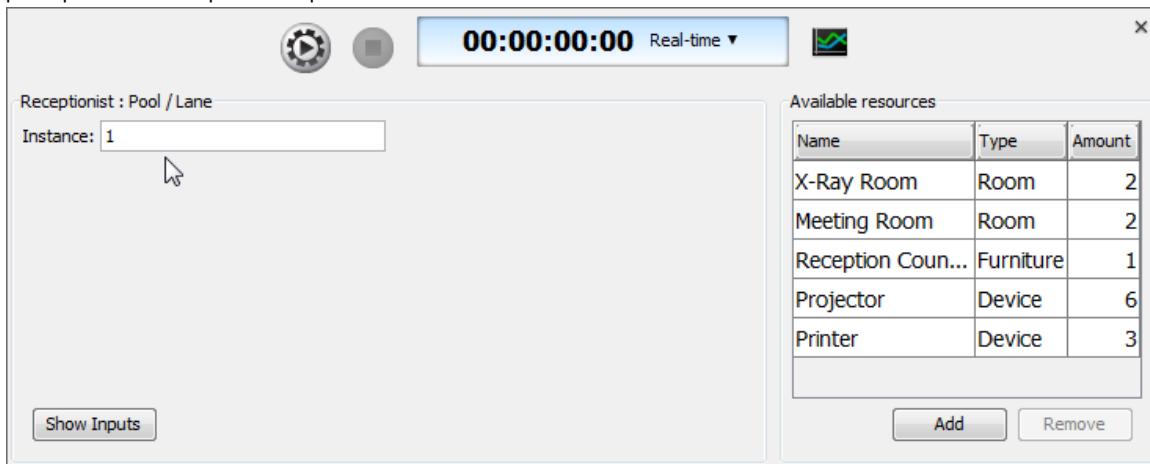
To add required resources

4. For each of the flow objects, select it in diagram, and set in **Simulacian 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 simulacian?*.



To set the duration of flow object

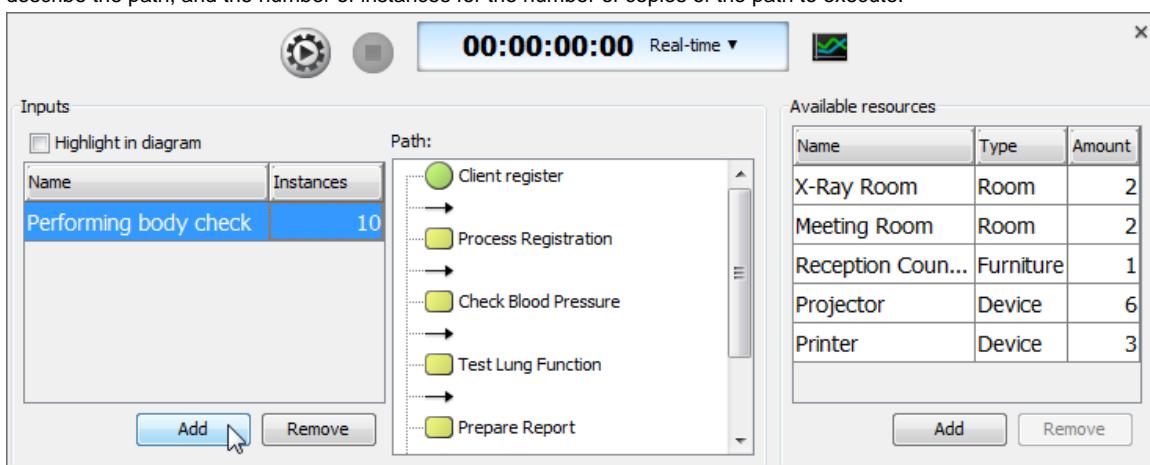
5. For each lane and pool (without lane), select it in diagram, and set its instance in **Simulacian Control Panel**, which represents the number of participants that take part in the process.



To set instance of pools or lanes

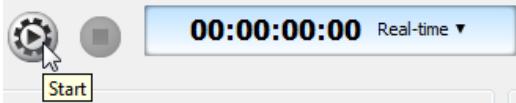
6. Click on the background of diagram or click **Show Inputs** in **Simulacian Control Panel**.

7. Click **Add** under Inputs in **Simulacian Control Panel** to add the paths to be executed during simulation. For each input, set the name that describe the path, and the number of instances for the number of copies of the path to execute.



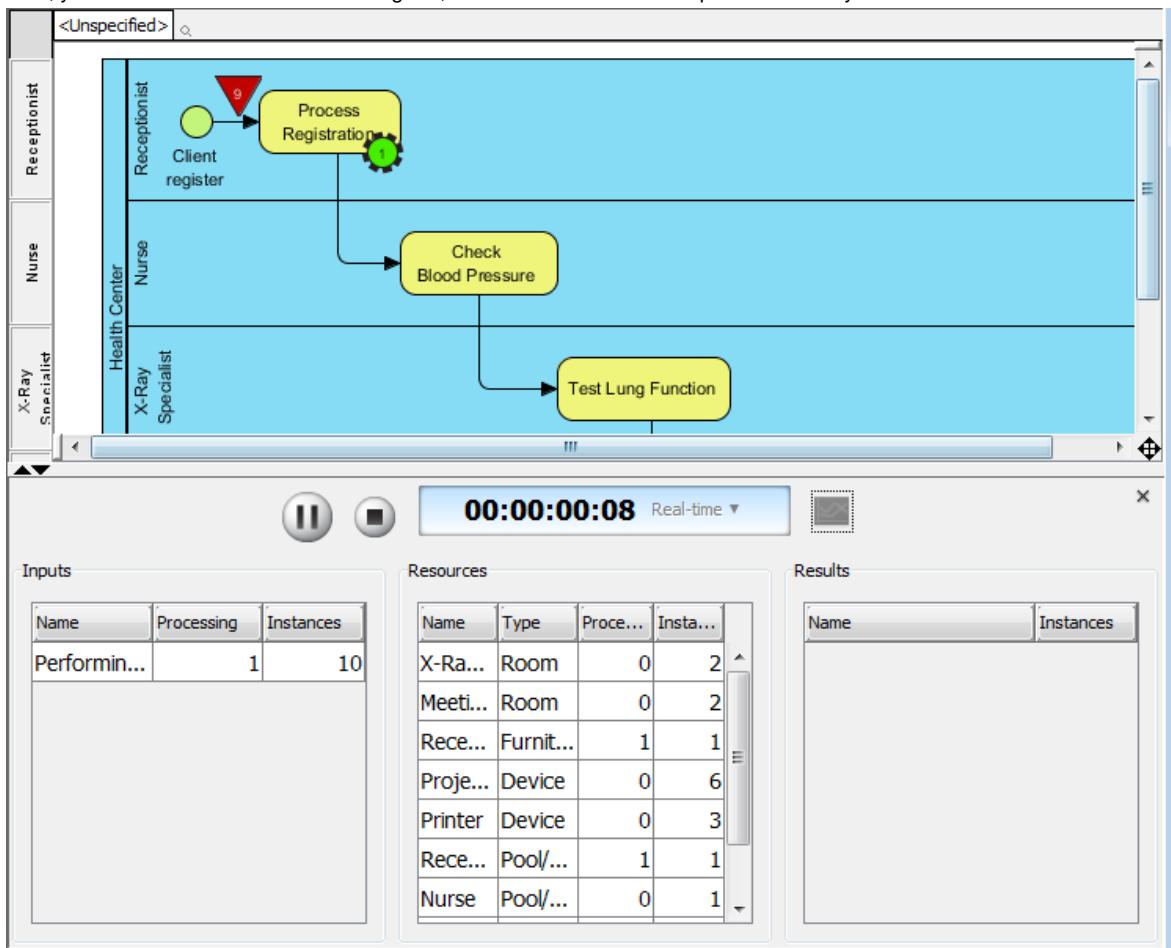
To add input

8. Click Start in **Simulacra Control Panel** to start simulation.



Start simulation

Now, you can watch the simulation in diagram, to see the executions of inputs and identify bottlenecks.



Simulacian charts

Sometimes, just by watching the simulacian outcome is not enough in finding out the bottleneck, especially when the diagram is large, and have many, many bottlenecks. In such cases, you can produce charts for simulacian outcome, which helps quantify resource consumption and queuing time for each flow object.

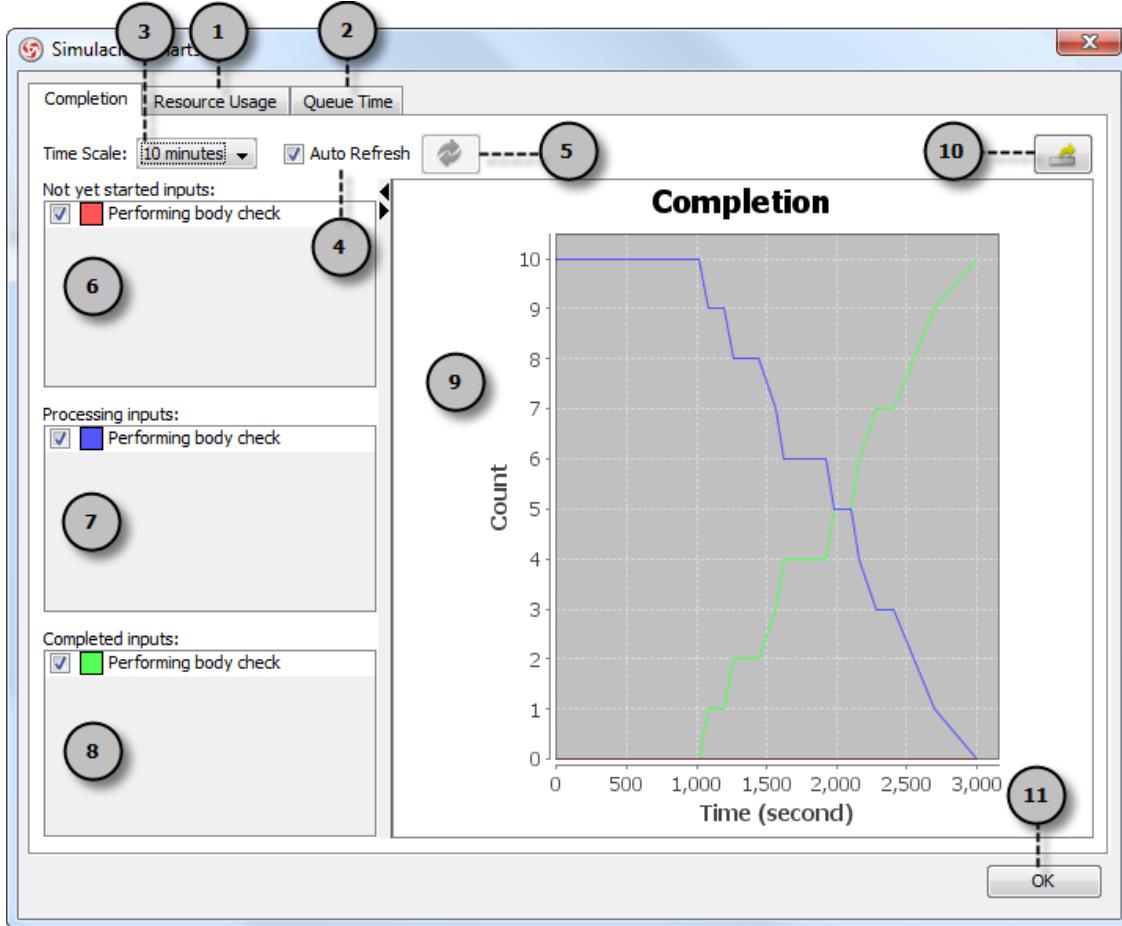
To read the charts, click **Simulacian Charts** in simulacian control panel.



To open simulacian charts window

Completion chart

The completion chart primarily shows the status of inputs completion against time.



An overview of completion chart

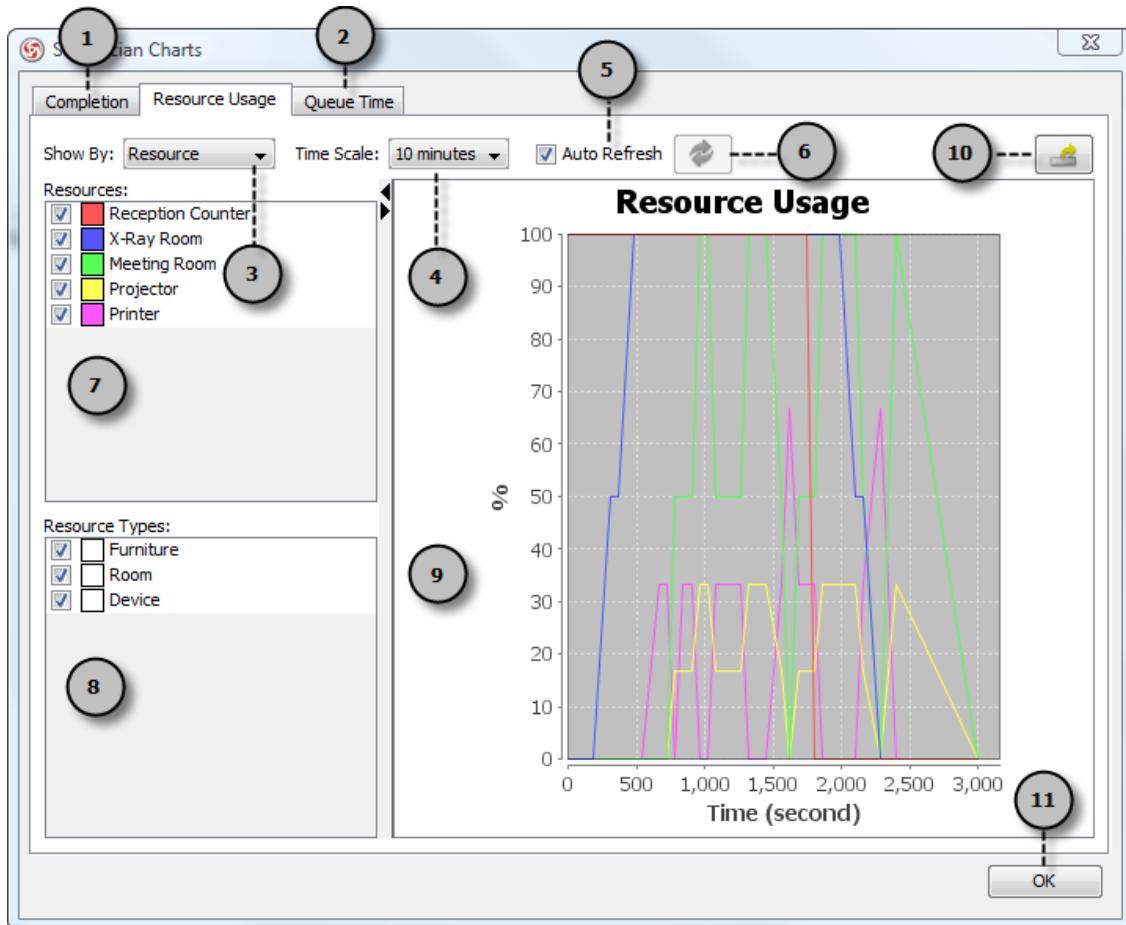
No.	Name	Description
1	Resource usage	Click to show resource usage chart.
2	Queue time	Click to show queue time chart.
3	Time scale	Control the length of chart by selecting a time scale.
4	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of inputs. Uncheck to update manually by clicking Refresh .
5	Refresh	Click to update the chart body against the chart settings such as time scale and selection of inputs. This button is available only when Auto Refresh is unchecked.
6	Not yet started inputs	Check or uncheck the inputs to show or hide in chart the change of the amount of not-yet-started inputs throughout simulation.
7	Processing inputs	Check or uncheck the inputs to show or hide in chart the change of the amount of processing inputs throughout simulation.
8	Completed inputs	Check or uncheck the inputs to show or hide in chart the change of the amount of completed inputs throughout simulation.

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 simulacian 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

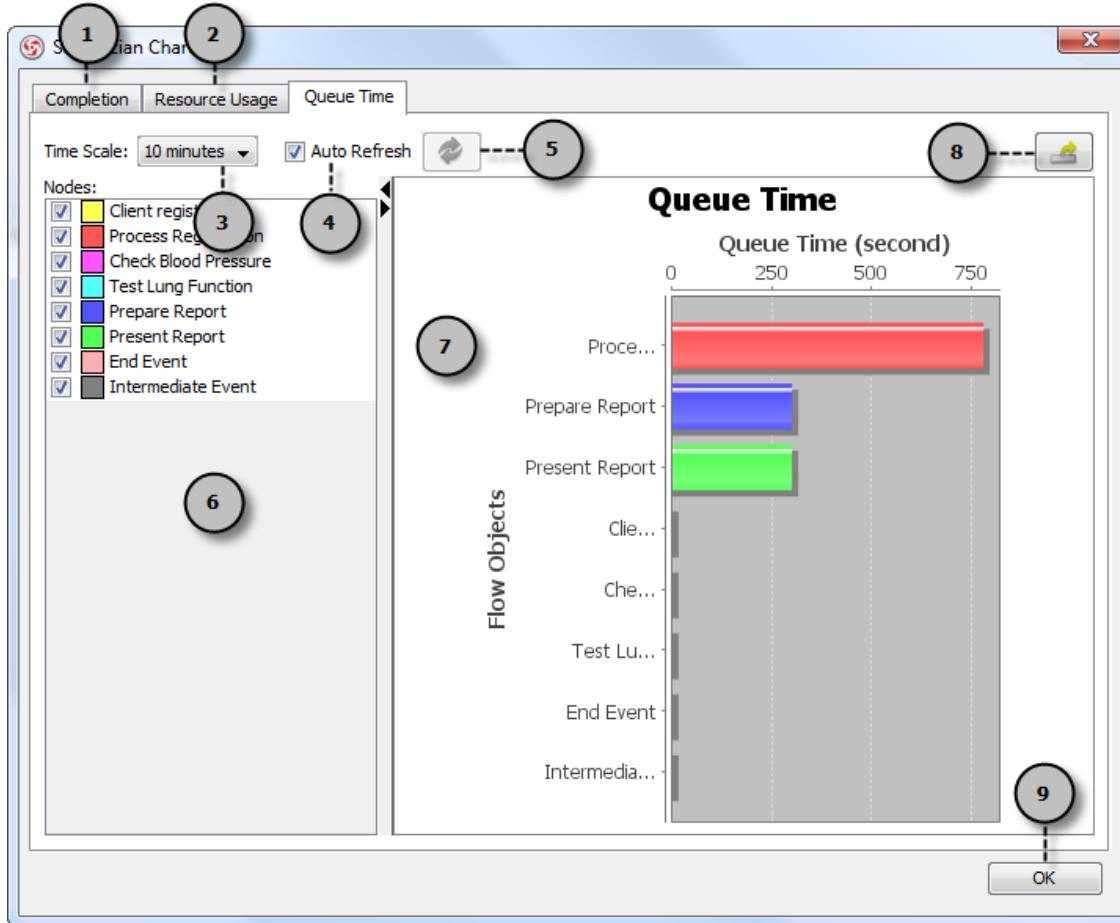
No.	Name	Description
1	Complete	Click to show completion chart.
2	Queue time	Click to show queue time chart.
3	Show by	Select what to present in the chart. Resource - Cause the chart to presents the resource consumption of each available resource. Resource Type - Cause the chart to presents the resource consumption of resource types.
4	Time scale	Control the length of chart by selecting a time scale.
5	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.
6	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.
7	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 .
8	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 .

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 simulacian 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 simulacian. 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

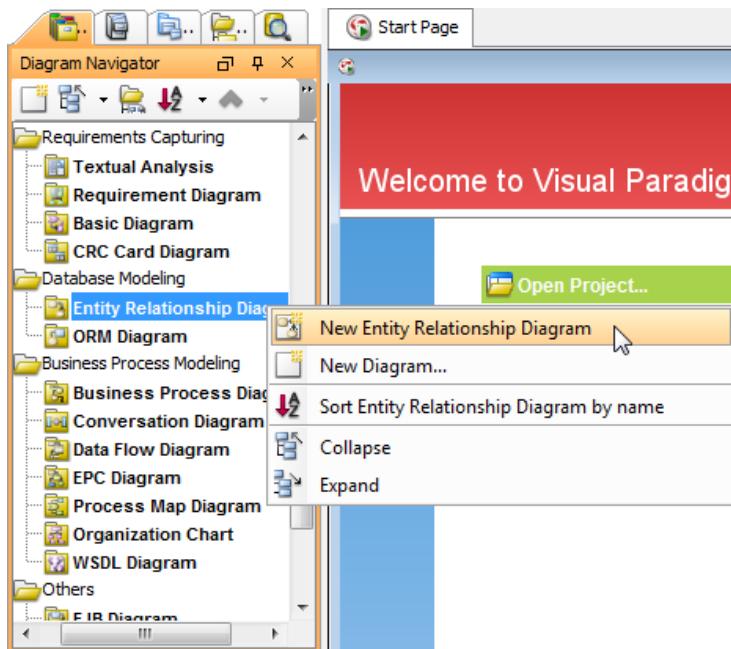
No.	Name	Description
1	Completion	Click to show completion chart.
2	Resource usage	Click to show resource usage chart.
3	Time scale	Control the length of chart by selecting a time scale.
4	Auto refresh	Check to let the chart body auto update against chart settings such as time scale and selection of nodes. Uncheck to update manually by clicking Refresh.
5	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.
6	Nodes	Check or uncheck flow objects nodes to show or hide their queue time in chart.
7	Chart body	The chart body.
8	Export	Click to export the opening chart to Microsoft Excel or image file.
9	OK	Click to close the simulacian charts window and go back to the diagram.

Description of queue time chart

Drawing entity relationship diagram

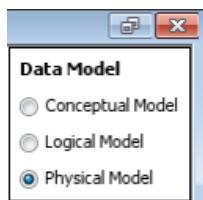
Creating entity relationship diagram

Right click on **Entity Relationship Diagram** from Diagram Navigator and select **New Entity Relationship Diagram** from the pop-up menu.



Create entity relationship diagram

Enter the name for the diagram. At the same time, a **Data Model** selection box appear on the top right corner of the diagram.

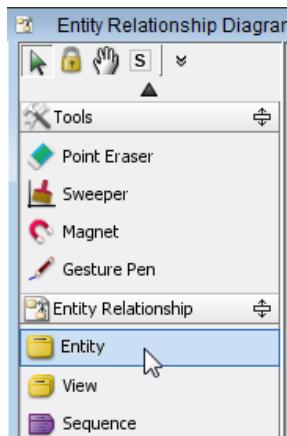


Select data model

All model elements created on diagram will follow this **Data Model** setting. Only **Physical Model** will be able to generate SQL. Leave it as default (Physical Model).

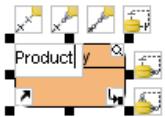
Drawing entity

Click **Entity** from diagram toolbar and drag it on the diagram pane.



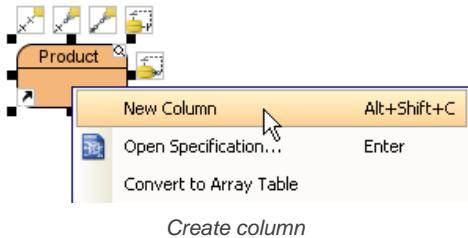
Create entity

Click on diagram and specify the entity name.



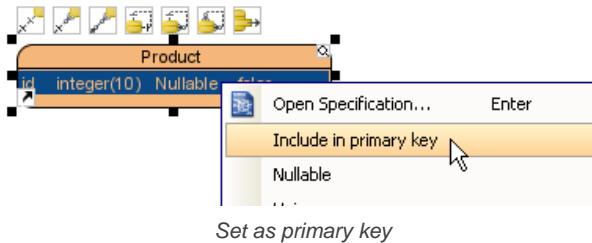
Rename an entity

Right click on the entity and select **New Column** from the pop-up menu, or press **Alt+Shift+C**.



Create column

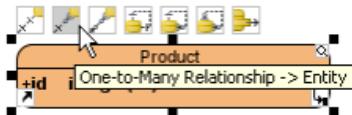
The column can be set with primary key by right clicking the column and selecting **Include in primary key** from the pop-up menu.



Set as primary key

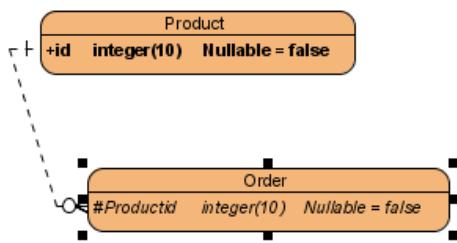
Drawing relationships

Point on the entity and either click or drag its resource icon to create another entity.



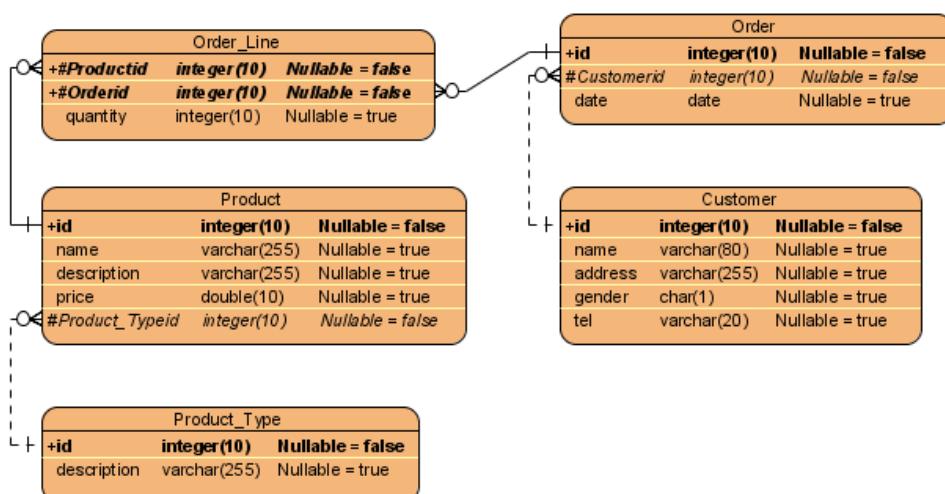
Create relationship

As a result, a new entity is created with foreign key column(s), referencing the original entity's primary key column(s).



Rename relationship

Follow the same steps to create other entities and relationship until the diagram looks like the picture below.



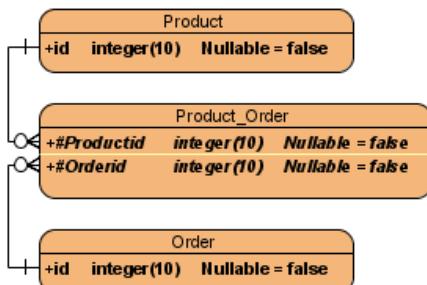
Drawing many-to-many relationship

Click or drag Many-to-Many relationship resource icon of an entity.



Create many-to-many relationship

Many-to-Many relationship will auto convert to two One-to-Many relationships and a join table. The primary key both tables will be used to create foreign key columns in join table as composite primary key.

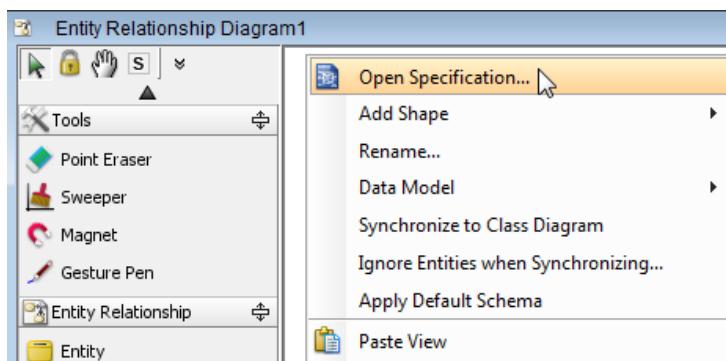


Created join table with one-to-many relationships

Defining default schema

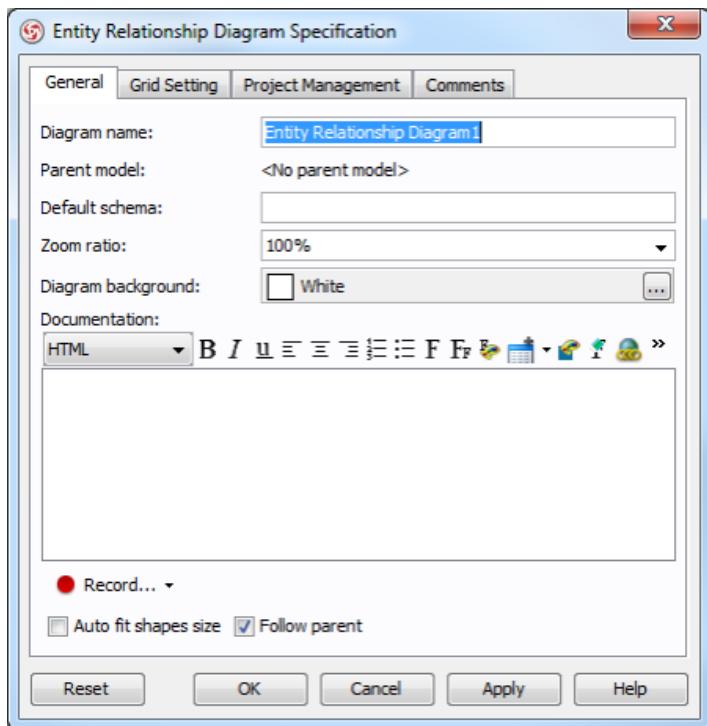
A default schema can be set on diagram, to make entities to be created on the diagram share the same schema. To define a default schema on diagram:

Right click on the diagram background and select **Open Specification...** from the pop-up menu.



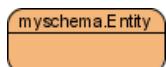
*Select **Open Specification...** from the pop-up menu*

Specify the default schema in the **Entity Relationship Diagram Specification** dialog box. Click **OK** to confirm the change.



Specify default schema

Now, when you create an entity on the diagram, the entity will be under the default schema.



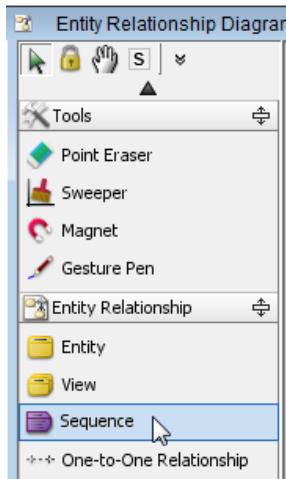
An entity with myschema as its schema

NOTE: In order to move entities in a diagram to a schema, define default schema on the owning entity relationship diagram, 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 erd to share the schema defined.

Drawing sequence

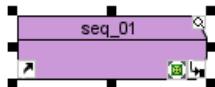
Creating sequence

Click **Sequence** from diagram toolbar and drag it on the diagram pane.



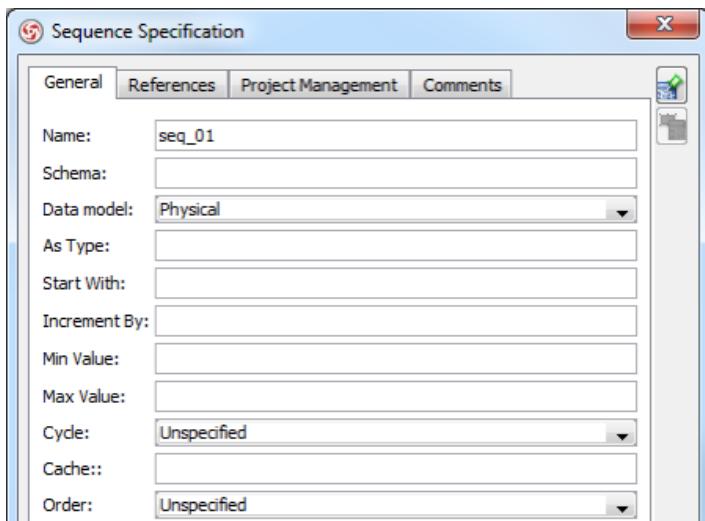
Create sequence

Type the name for sequence when it is created.



Rename sequence

Right click on the sequence and select **Open Specification...** from the pop-up menu. In **Sequence Specification** dialog box, specify sequence attributes.



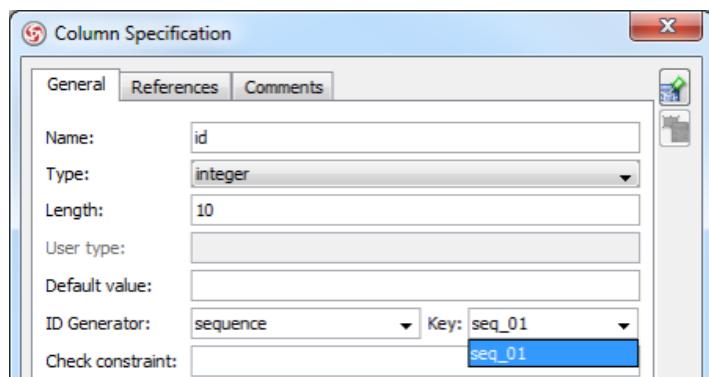
Specify sequence attributes

NOTE: Generate sequence is only supported in DB2 and Oracle.

Select sequence for entity

After creating an entity and a primary key column, right click on the primary key column and select **Open Specification...** from the pop-up menu.

In Column Specification dialog, select sequence for ID Generator, and select the sequence name for Key.



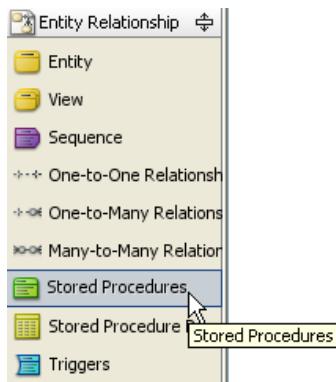
Select sequence for entity

When using Object-Relational Mapping feature, the primary key value will be inserted automatically from the sequence.

Drawing stored procedures

Creating stored procedures shape

Click **Stored Procedures** from diagram toolbar and drag it on the diagram pane.



Create stored procedure shape

Specify the name for the newly created stored procedures and press **Enter**.

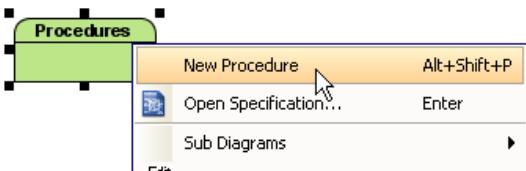


Rename procedure shape

NOTE: Procedure shape is a virtual container to group a set of stored procedures. It is not a stored procedure.

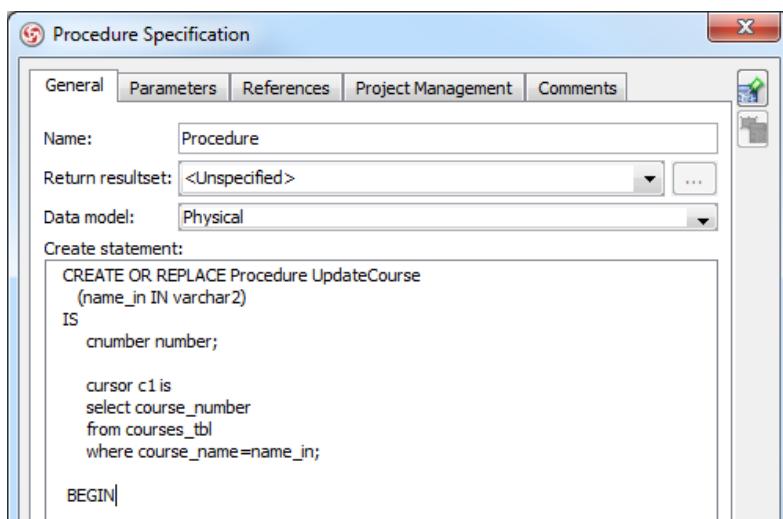
Creating procedure

Right click on the stored procedures and select **New Procedure** from the pop-up menu, or press **Alt+Shift+P**.



Create stored procedure

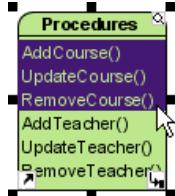
After the new procedure is created, right click on it and select **Open Specification...** from the pop-up menu. In **Procedure Specification** dialog box, specify the create statement and create parameters if necessary.



Specify create statement in **Procedure Specification** dialog box

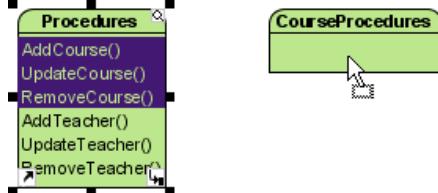
Moving or duplicating a procedure to another procedure container

1. Select the procedure to move or duplicate.



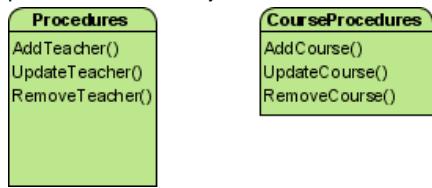
Select procedures to move or duplicate

2. Drag over the target procedure container.



Drag procedure towards the target procedures container

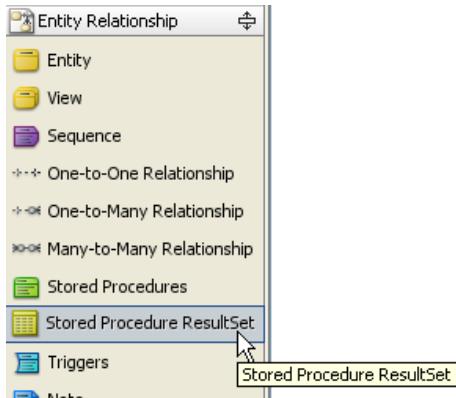
3. If you want to duplicate the procedures, press on the **Ctrl** key and release the mouse button. If you want to move them from source to target procedure container, just release the mouse button.



Procedures are moved

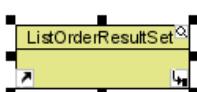
Creating stored procedure resultSet

Click **Stored Procedure ResultSet** from diagram toolbar and drag it on the diagram pane.



Create stored procedure resultSet

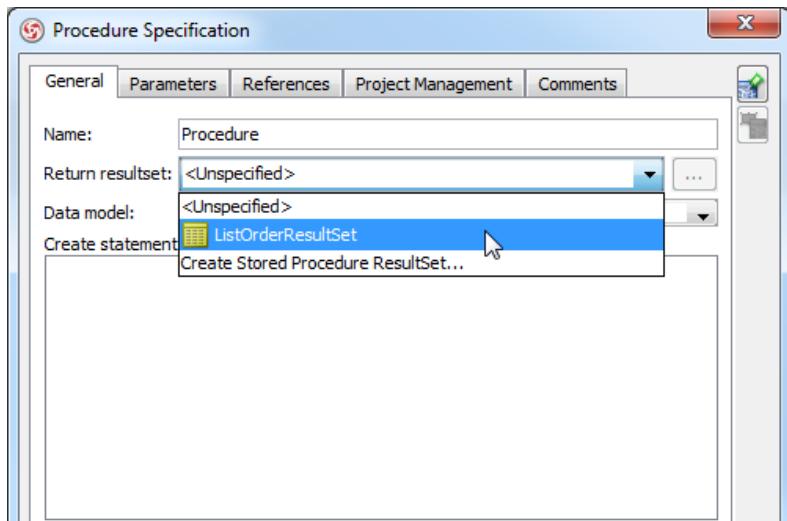
After specify the name for the newly created resultset, press **Enter**.



Rename stored procedure resultSet

The way of creating resultset column is the same as creating entity column.

Right click on the procedure that created above, select **Open Specification...** from the pop-up menu. In **Procedure Specification** dialog box, specify stored procedure return resultset.

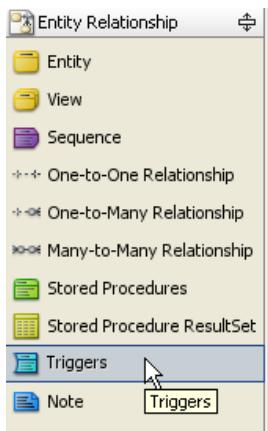


Specify stored procedure return resultset

Drawing triggers

Create a triggers shape

Click **Triggers** from diagram toolbar and drag it on the diagram pane.



Create triggers shape

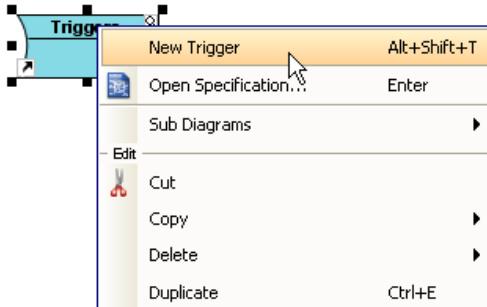
After specify the name for the newly created triggers, press **Enter**.



Rename a triggers shape

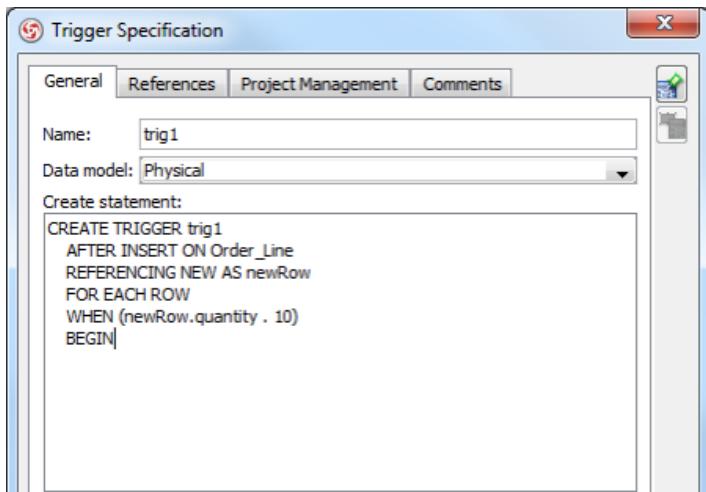
Create a new trigger

Right click the triggers and select **New Trigger** from the pop-up menu, or press **Alt+Shift+T**.



Create a new trigger

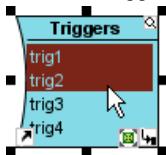
In **Triggers Specification** dialog box, specify the create statement.



*Specify the create statement in **Trigger Specification** dialog box*

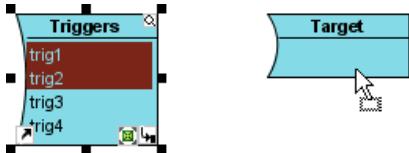
Move or duplicate triggers to another trigger container

1. Select the trigger to move or duplicate.



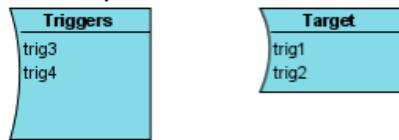
Selecting triggers to move or duplicate

2. Drag over the target trigger container.



Dragging 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.

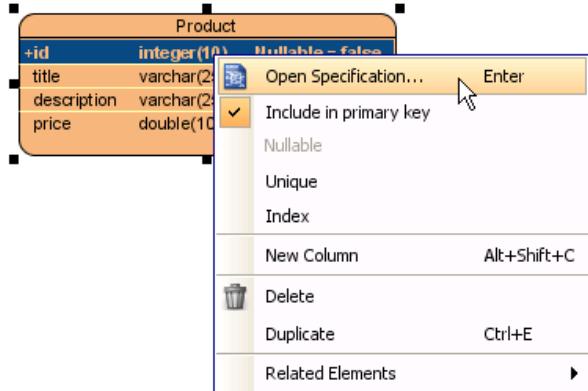


Triggers are moved

Controlling primary key values using ID generator

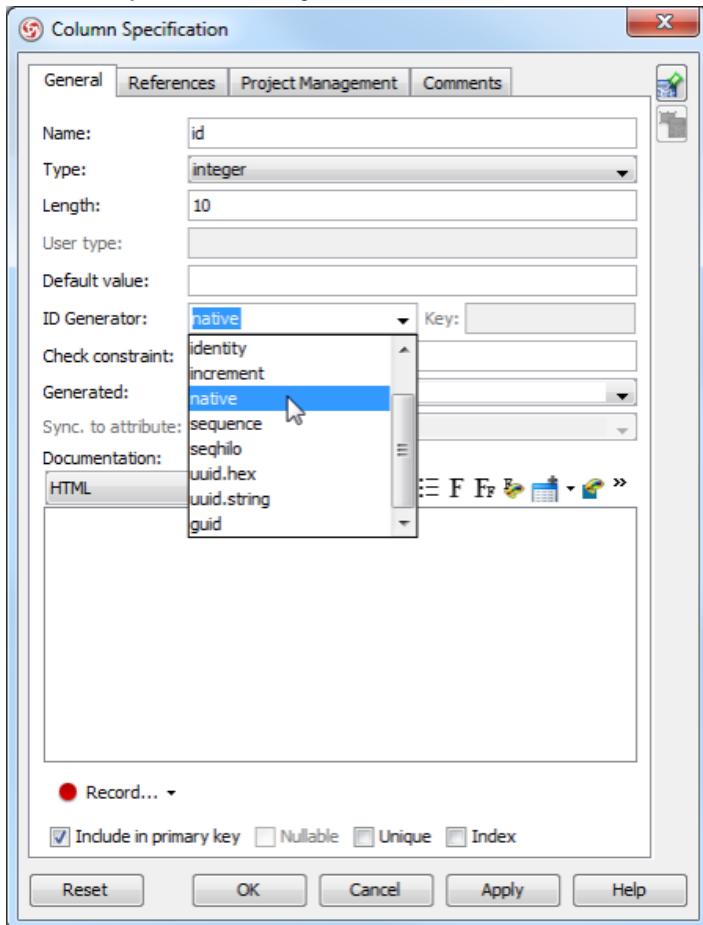
ID generator defines how a unique value will be produced for a primary key column. You can assign an ID generator to a primary key column for which strategy will be used when generating an ID in runtime.

1. Right click the primary key column that you want to select an ID generator for and select **Open Specification...** from the pop-up menu.



Select **Open Specification...** from the pop-up menu

2. In **Column Specification** dialog box, select the **ID Generator** and then click **OK** to confirm.



Select an ID generator in **Column Specification** dialog box

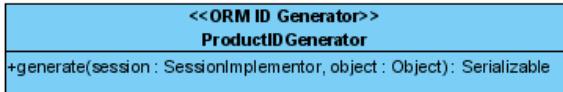
ID Generator	Description
assigned	lets the application to assign an identifier to the object before is called.
guid	uses a database-generated GUID string on MS SQL Server and MySQL.
hilo	uses a hi/lo algorithm to efficiently generate identifiers of type , or , given a table and column as a source of 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 , or .
increment	generates identifiers of type , or that are unique only when no other process is inserting data into the same table. <i>Do not use in a cluster.</i>
native	(default) picks , or depending upon the capabilities of the underlying database.
seqhilo	uses a hi/lo algorithm to efficiently generate identifiers of type , or , given a named database sequence.
sequence	uses a sequence in DB2, PostgreSQL, Oracle. The returned identifier is of type , or

The description of available ID Generator

Customizing ID generator

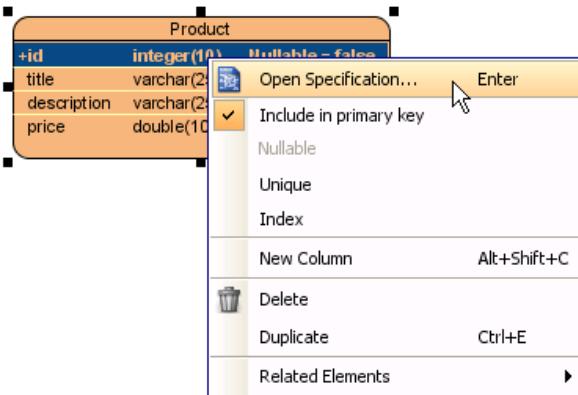
Besides the built-in strategies for generating ID, user can implement how ID will be generated by customizing an ID generator.

1. In Class Diagram, create the ID generator class, and stereotype it as **ORM ID Generator**.



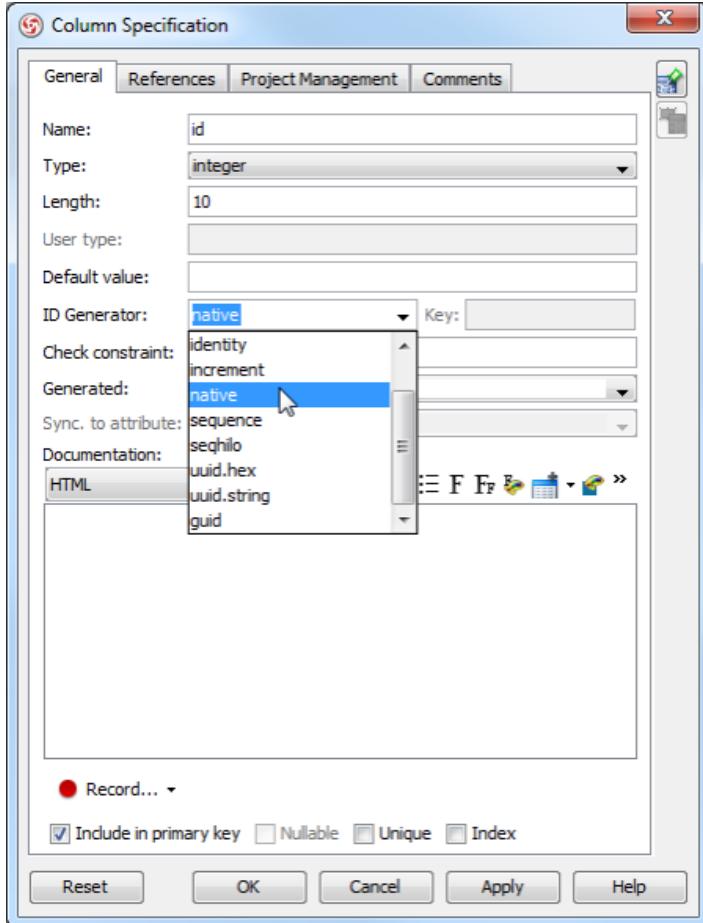
An ID generator class

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** dialog box, select the class in the **ID Generator**.



Select an ID generator in **Column Specification** dialog box

4. Click **OK** to confirm.

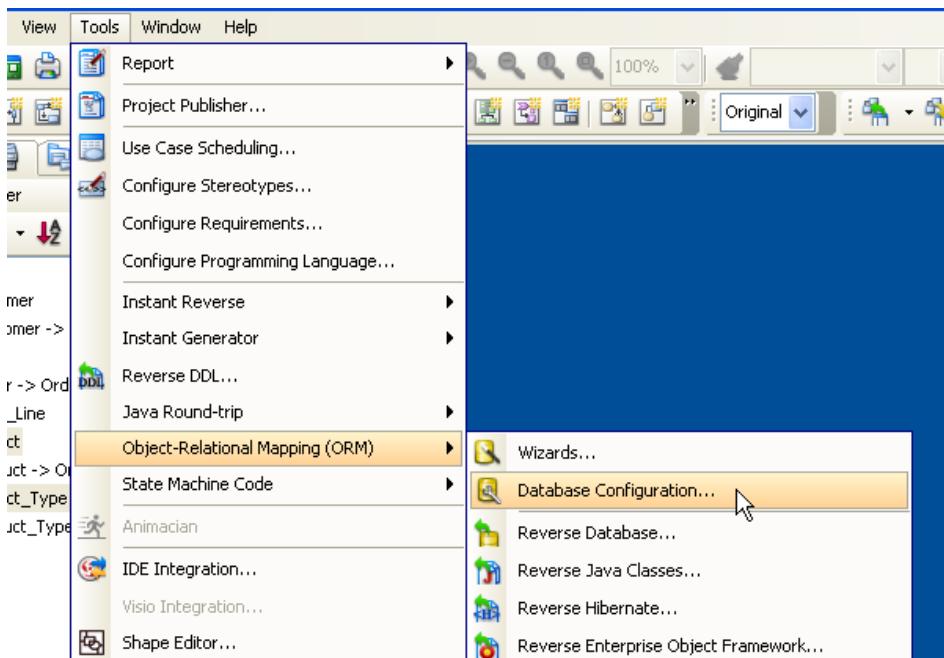
5. After generated ORM code, look for the ID generator class and implement the **generate** method by return an Integer or Long.

```
/*
 * 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
```

Database configuration

Opening database configuration dialog

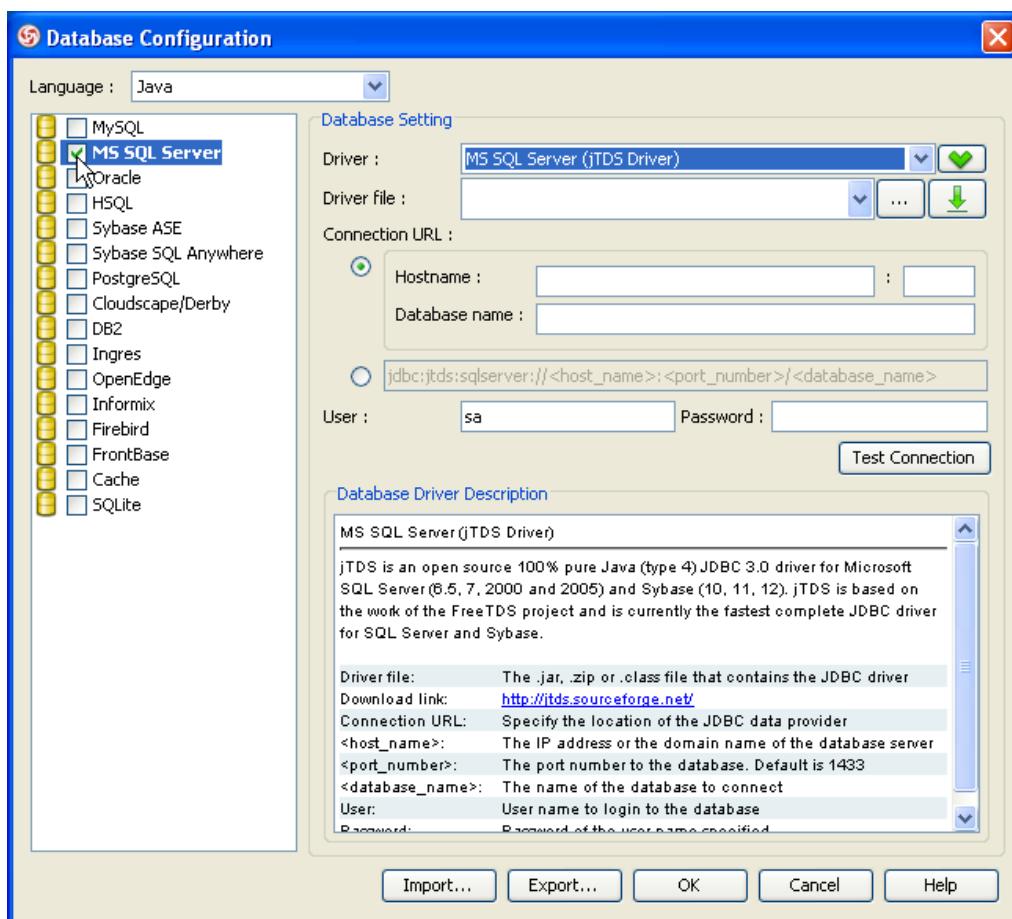
Select Tools > Object-Relational Mapping (ORM) > Database Configuration... from the main menu.



Database configuration

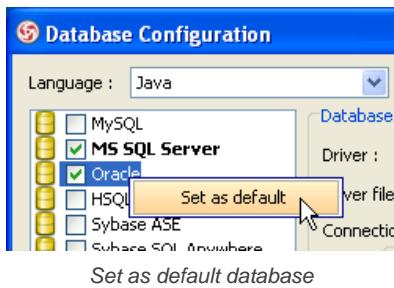
Selecting database

Click the checkbox of the database.



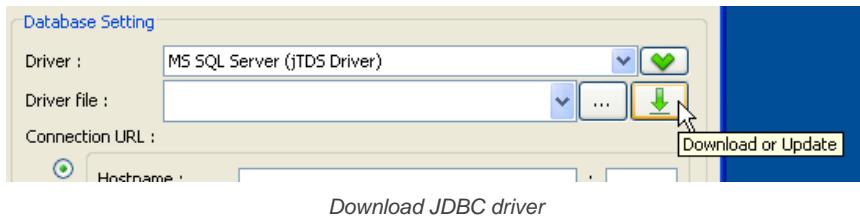
Select database

You can select multiple databases, and set one of them as default database. The default database is used for rendering the column type and generating SQL. To set a database as default, right click on the database and select **Set as default** from the popup menu.

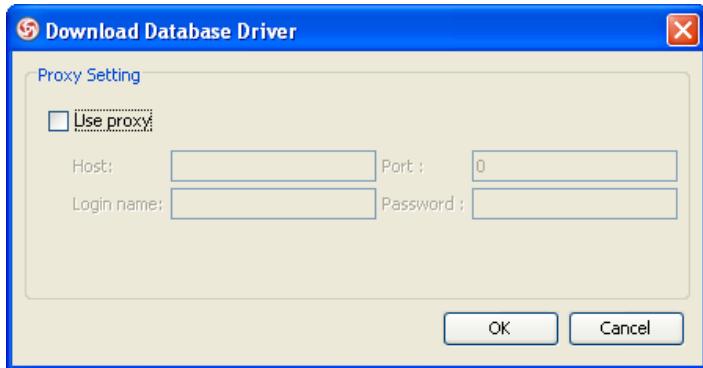


Downloading JDBC driver

If the JDBC driver is free and available to public, VP-UML can help you download automatically. Click the **Download or Update** button on **Database Setting**.

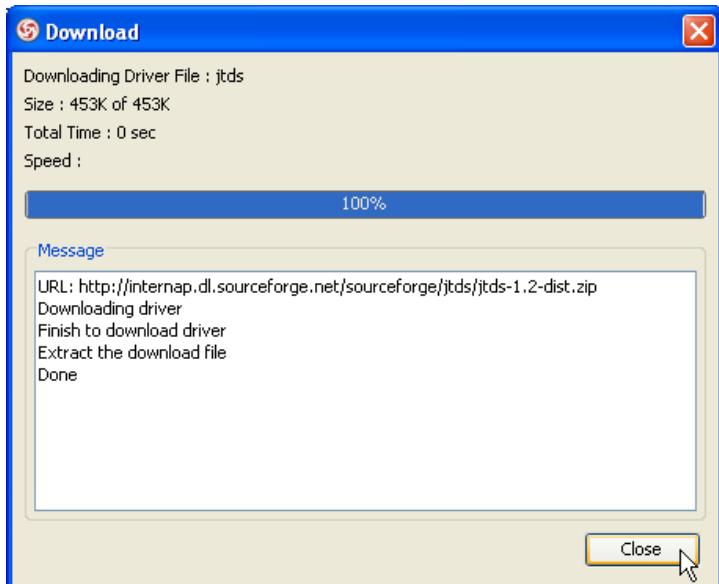


Configure proxy if required, click **OK** button to continue download.



Download JDBC proxy setting

The **Download** dialog show the URL, file size, speed, progress information, click **Close** button after finish.



Download JDBC progress

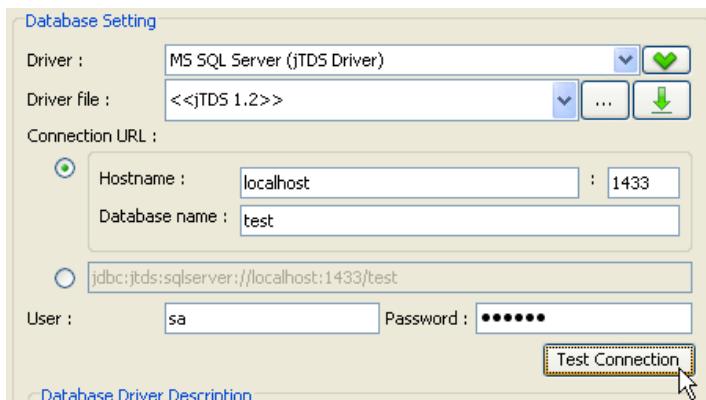
The **Driver file** field will becomes <<Driver Name>> indicating using the driver downloaded by VP-UML.



Using downloaded JDBC driver

Testing connection

After configure the database setting, click the **Test Connection** button to confirm the setting is valid.



Database Setting

Driver : MS SQL Server (jTDS Driver) 

Driver file : <<jTDS 1.2>> 

Connection URL :

Hostname : localhost : 1433
Database name : test

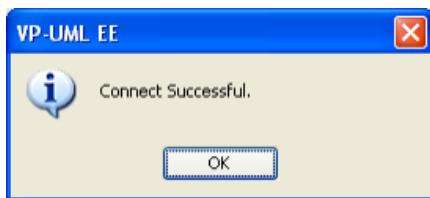
jdbc:jtds:sqlserver://localhost:1433/test

User : sa Password : *****

Test Connection

Test connection

If success, a dialog will show connect successful.

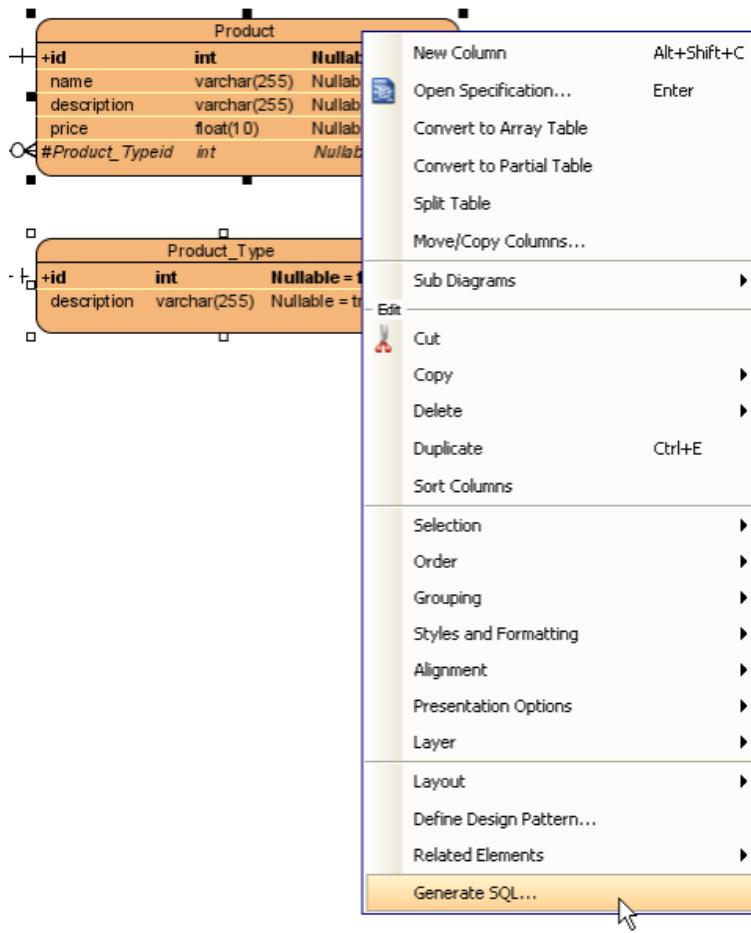


Connect successful

Generating SQL for selected entities

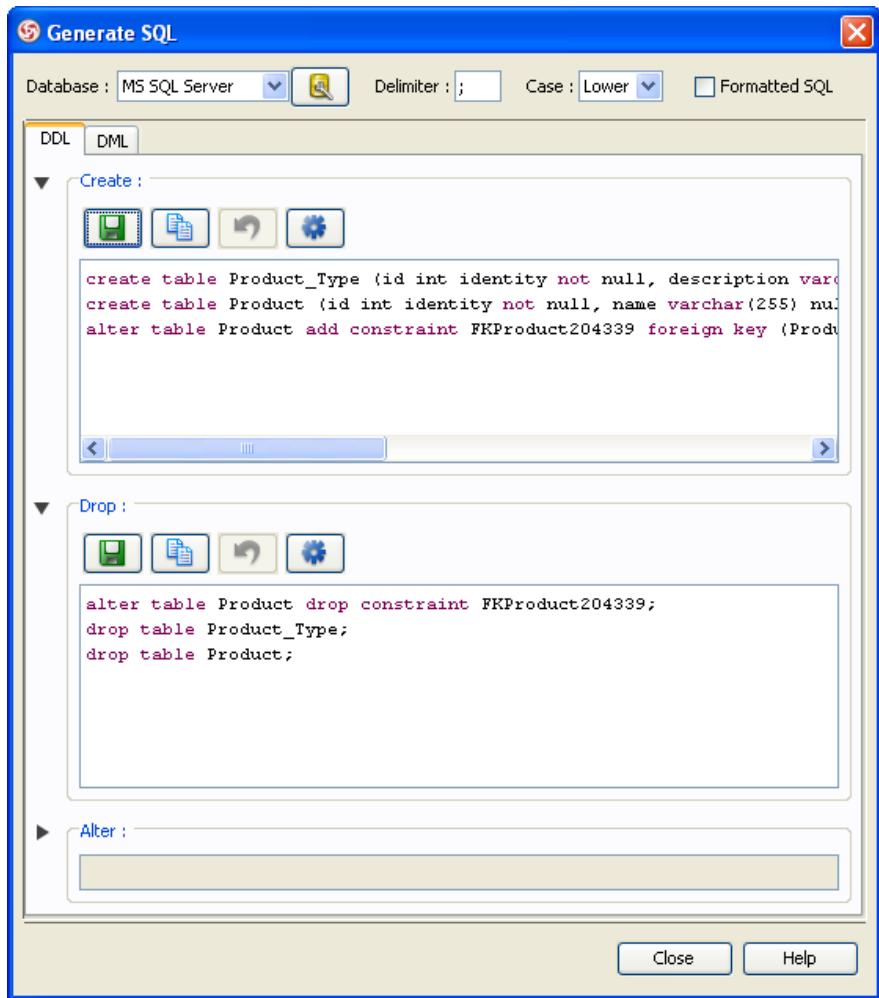
Generating SQL

Select multiple entities, right click and select **Generate SQL...** from the popup menu.



Generate SQL

The Generate SQL dialog shows the DDL and DML for the selected entities.

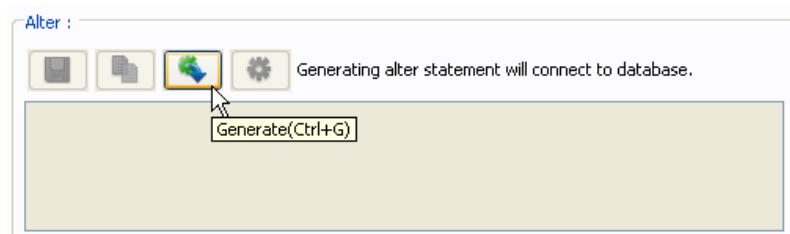


Generate SQL dialog

DDL and DML

DDL stand for Data Definition Language, include create, drop and alter statements. DML stand for Data Manipulating Lanauge, include select, insert, update, delete statements.

The generated DDL statements can directly execute to database without any modification. Generate alter statement require connection to database, query the object from database and compare with the ERD. The alter statements are not generated by default, you can click the Generate button to generate on demand.



Generate alter statements

The generate DML is a template for select, insert, update, delete statements, you are required to modify the statement before execute.

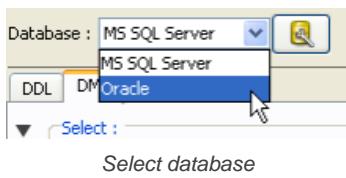
Selecting database

Click the **Database Configuration** button to open the database configuration dialog.



Database configuration

If you selected multiple database, you can select one of the database from the database combo box.



After change the database, the SQL will re-generate for the selected database.

A screenshot of a software interface showing two sections: "Create:" and "Drop:". The "Create:" section contains SQL code for creating tables and sequences. The "Drop:" section contains SQL code for dropping tables and sequences. Both sections have toolbars with icons for file operations and a gear icon for settings.

```
create table Product_Type (id number(10) not null, description varchar2(255), des
create table Product (id number(10) not null, name varchar2(255), des
alter table Product add constraint FKProduct204339 foreign key (Produ
create sequence seq_Product_Type;
create sequence seq_Product;
```

```
drop table Product_Type cascade constraints;
drop table Product cascade constraints;
drop sequence seq_Product_Type;
drop sequence seq_Product;
```

SQL updated

Generate SQL options

There are several option on the top of the **Generate SQL** dialog.

A screenshot of a software interface showing "Generate SQL options" settings. It includes fields for "Delimiter" (set to ";"), "Case" (set to "Lower"), and "Formatted SQL" (unchecked). Below these settings is a preview window showing the generated SQL code.

Delimiter : ; Case : Lower Formatted SQL

Generate SQL options

```
create table Product_Type (id number(10) not null, description varchar2(255), des
create table Product (id number(10) not null, name varchar2(255), des
alter table Product add constraint FKProduct204339 foreign key (Produ
create sequence seq_Product_Type;
create sequence seq_Product;
```

Delimiter

- **Delimiter** - append to end of each statement, used for separate two statement. If change to \\, the SQL will becomes:

A screenshot of a software interface showing the generated SQL code with a double backslash (\) as the delimiter. The code is identical to the previous screenshot but separated by double backslashes.

```
create table Product_Type (id number(10) not null, description varchar2(255), des
create table Product (id number(10) not null, name varchar2(255), des
alter table Product add constraint FKProduct204339 foreign key (Produ
create sequence seq_Product_Type\\
create sequence seq_Product\\
```

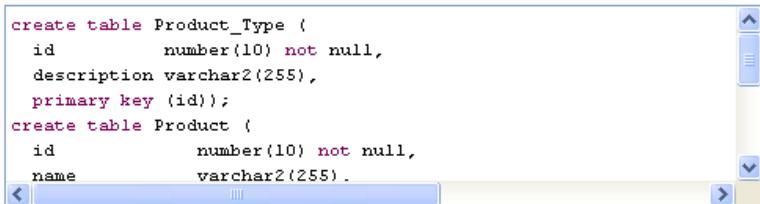
Upper SQL

- **Case** - the case for the keyword. If change to **Upper**, the SQL will becomes:

A screenshot of a software interface showing the generated SQL code with uppercase keywords. The code is identical to the previous screenshots but with uppercase words like CREATE, TABLE, NOT, NULL, FOREIGN, KEY, etc.

```
CREATE TABLE Product_Type (id number(10) NOT NULL, description varchar2(255), des
CREATE TABLE Product (id number(10) NOT NULL, name varchar2(255), des
ALTER TABLE Product ADD CONSTRAINT FKProduct204339 FOREIGN KEY (Produ
CREATE SEQUENCE seq_Product_Type;
CREATE SEQUENCE seq_Product;
```

- **Formatted SQL** - formatted sql generate high readability SQL statements. If enable, the SQL will becomes:



```
create table Product_Type (
    id          number(10) not null,
    description varchar2(255),
    primary key (id));
create table Product (
    id          number(10) not null,
    name        varchar2(255).
```

Formatted SQL

Using toolbar

There is a toolbar above each group of statements.

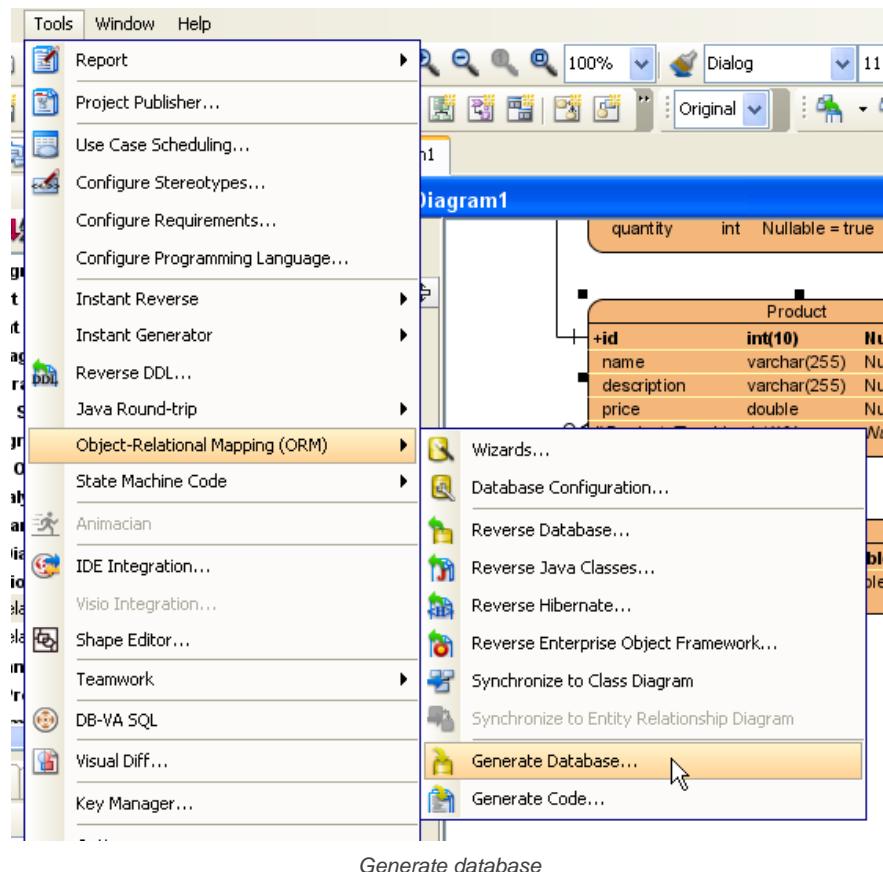


Generate SQL toolbar

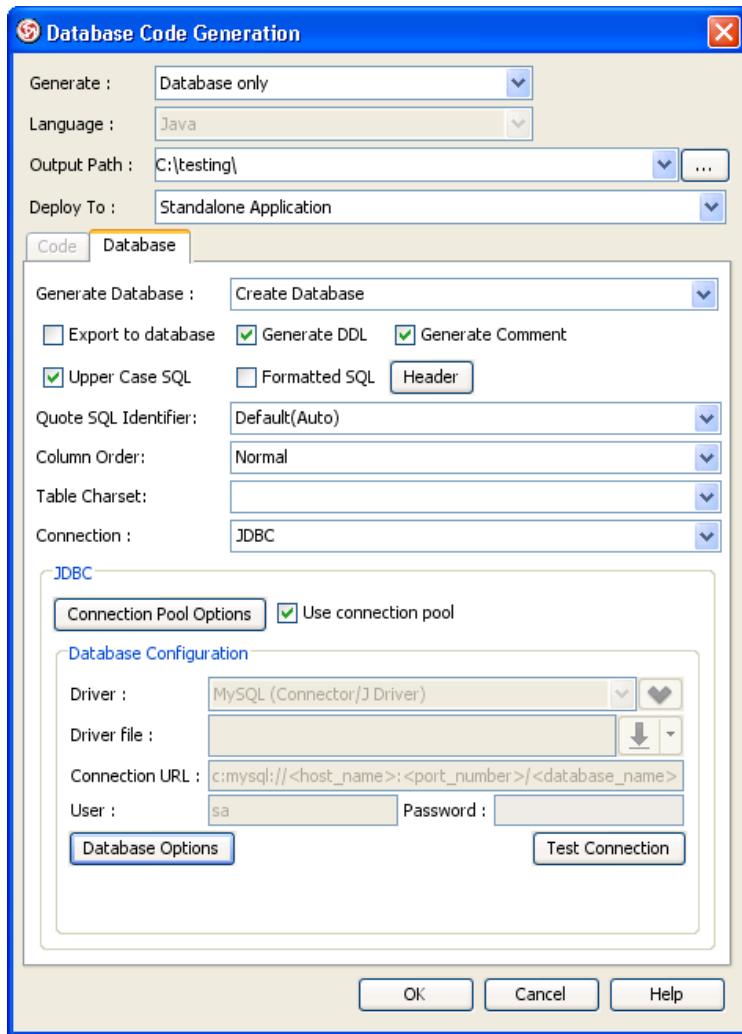
- **Save to file** - Save the generated SQL to file.
- **Copy** - Copy the generated SQL to clipboard.
- **Revert** - The textbox of SQL is editable, you can modify before execute. The **Revert** button allow you to undo the user modification.
- **Execute** - Execute the statements in the textbox to database.

Generating SQL for project

Select Tools > Object-Relational Mapping (ORM) > Generate Database... from the main menu,

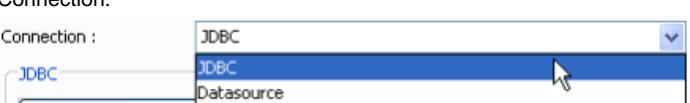


The Database Code Generation dialog provides several options to generate DDL and export to database.



Database code generation

- Generate Database:
 - 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.
- Export to database - execute the generated statements directly to database.
- Generate DDL - save the generated statements to a file.
- Generate Comment - generate comment of tables/columns to database/DDL (only available to My SQL, DB2, Oracle, Postgre SQL)
- Upper Case SQL - generate upper case for keyword (e.g. select, from, update, insert...etc).
- Formatted SQL - generate pretty format, high readability SQL.
- Quote SQL Identifier - if database object name is reserved word, it must be quoted; otherwise, it cannot be used as the object name:
 - Auto - auto detect and quote the name only if it is reserved word.
 - Yes - always quote the name.
 - No - never quote the name.
- Table Charset (Only available for MySQL) - the charset used for database connection.
- Connection:



Connection

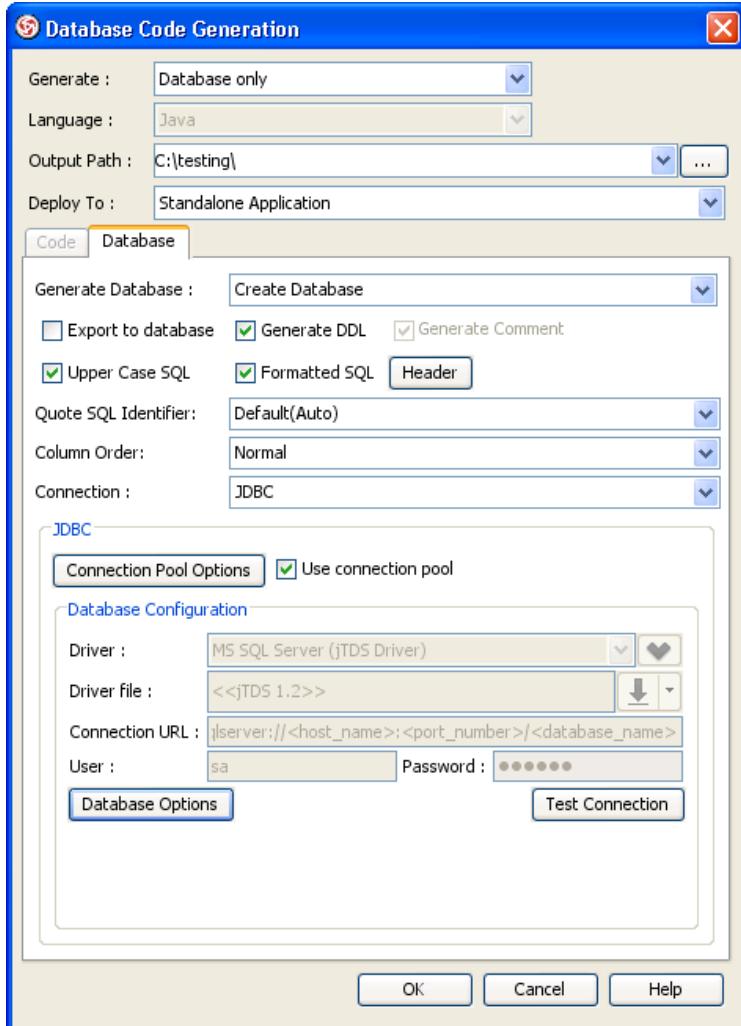
- JDBC - a standard way to connect database in Java.
- Datasource - the database connection is managed by application server.

Generating alter statements

Generate alter statements helps you to update the database with the changes on ERD. With the generate DDL option, you can preview a list of alter statements before actually applies to the database.

1. Open **Database Code Generation** dialog.

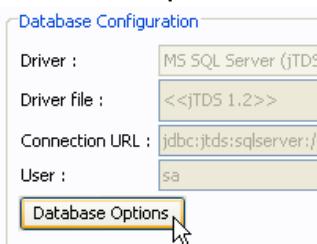
2. Select the following options:



Database code generation

- **Generate** - Database only
- **Generate Database** - Update Database
- **Export to database** - unchecked
- **Generate DDL** - checked
- **Upper Case SQL** - checked
- **Formatted SQL** - checked
- **Connection** - JDBC

3. Click **Database Options** button to configure JDBC.



Database options

4. Click **OK** button to generate. Assume some tables and columns were created in database already, the generated statements will looks like:

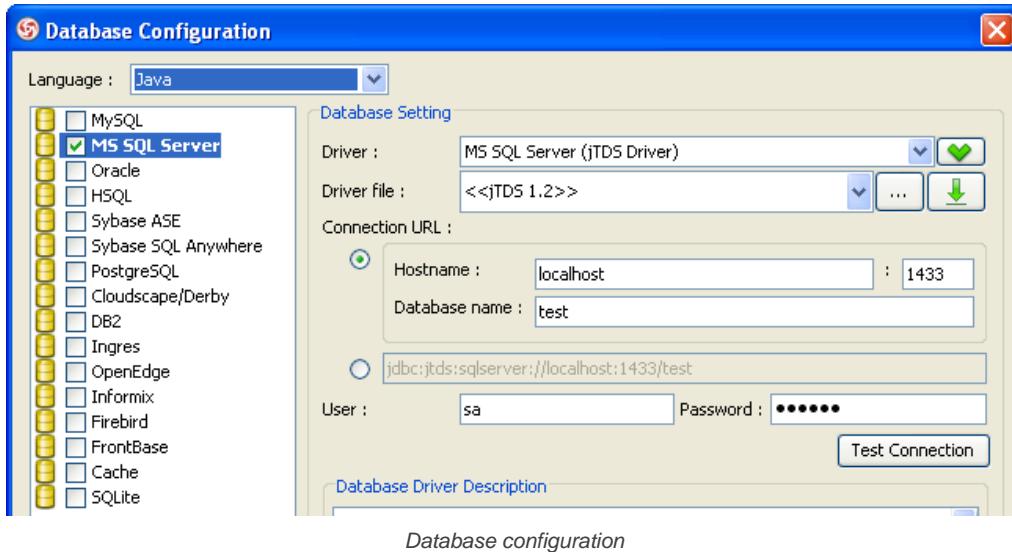
```
GO
ALTER TABLE Product
    ADD price float(10) NULL;
GO
CREATE TABLE [order] (
    id      int IDENTITY NOT NULL,
    Customerid int NOT NULL,
    [date]   datetime NULL,
    PRIMARY KEY (id));
GO
CREATE TABLE Order_Line (
    Productid int NOT NULL,
    Orderid   int NOT NULL,
    quantity  int NULL,
    PRIMARY KEY (Productid,
    Orderid));
GO
CREATE TABLE Customer (
    id      int IDENTITY NOT NULL,
    name   varchar(80) NULL,
    address varchar(255) NULL,
    gender  char(1) NULL,
    tel     varchar(20) NULL,
    PRIMARY KEY (id));
GO
ALTER TABLE Order_Line ADD CONSTRAINT FKOrder_Line37202 FOREIGN KEY (Productid)
REFERENCES Product (id);
GO
ALTER TABLE Order_Line ADD CONSTRAINT FKOrder_Line284509 FOREIGN KEY (Orderid)
REFERENCES [order] (id);
GO
ALTER TABLE [order] ADD CONSTRAINT FKOrder558759 FOREIGN KEY (Customerid) REFERENCES
Customer (id);
```

Alter statements

Export and import 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.

1. Open **Database Configuration** dialog.
2. Select database(s) and define their settings.



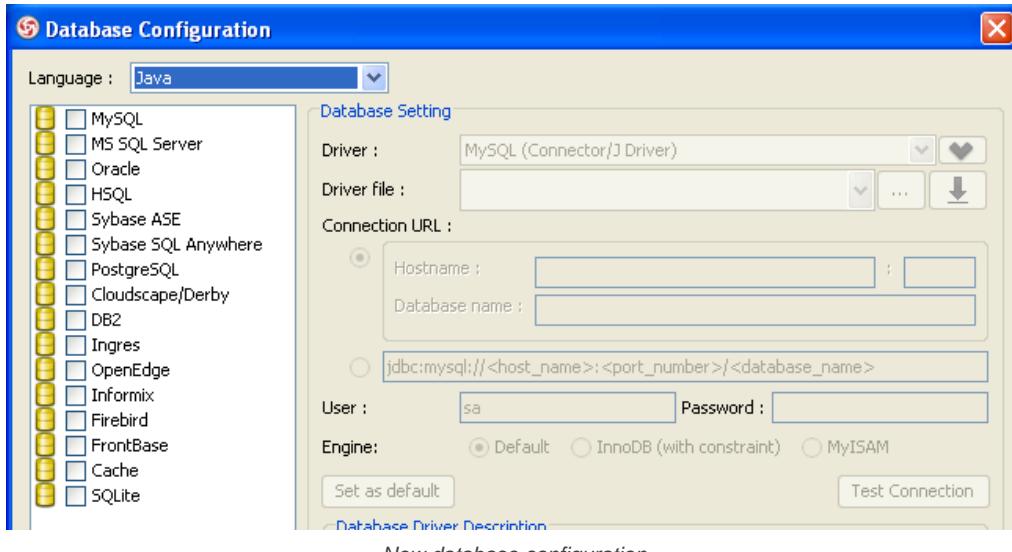
Database configuration

3. Click the **Export...** button, and specify the filename to save the setting.



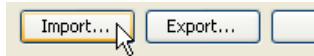
Export button

4. Create a new project.
5. Open **Database Configuration** dialog again, the setting is blank now.



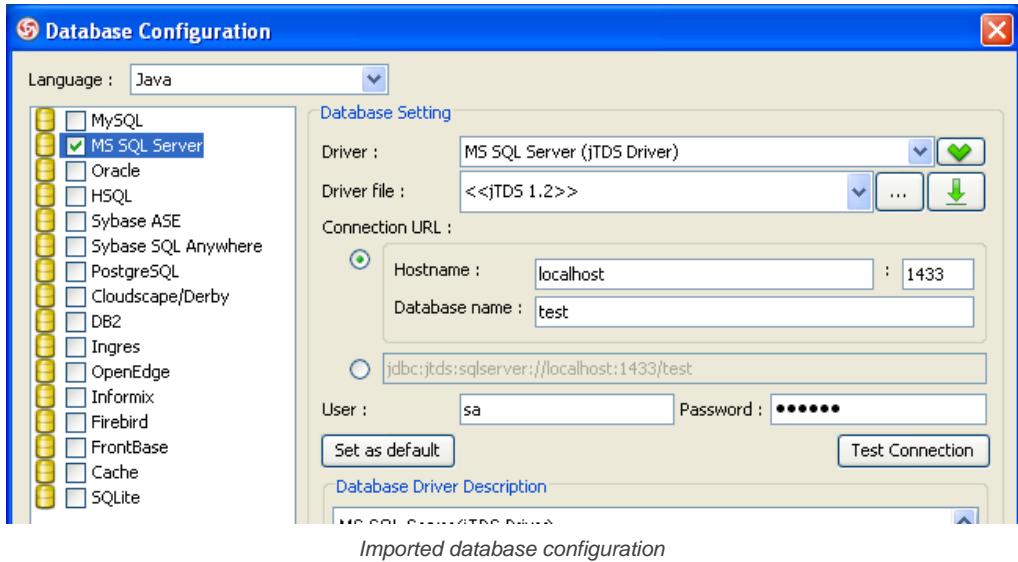
New database configuration

6. Click the **Import...** button, and select the file saved previously.



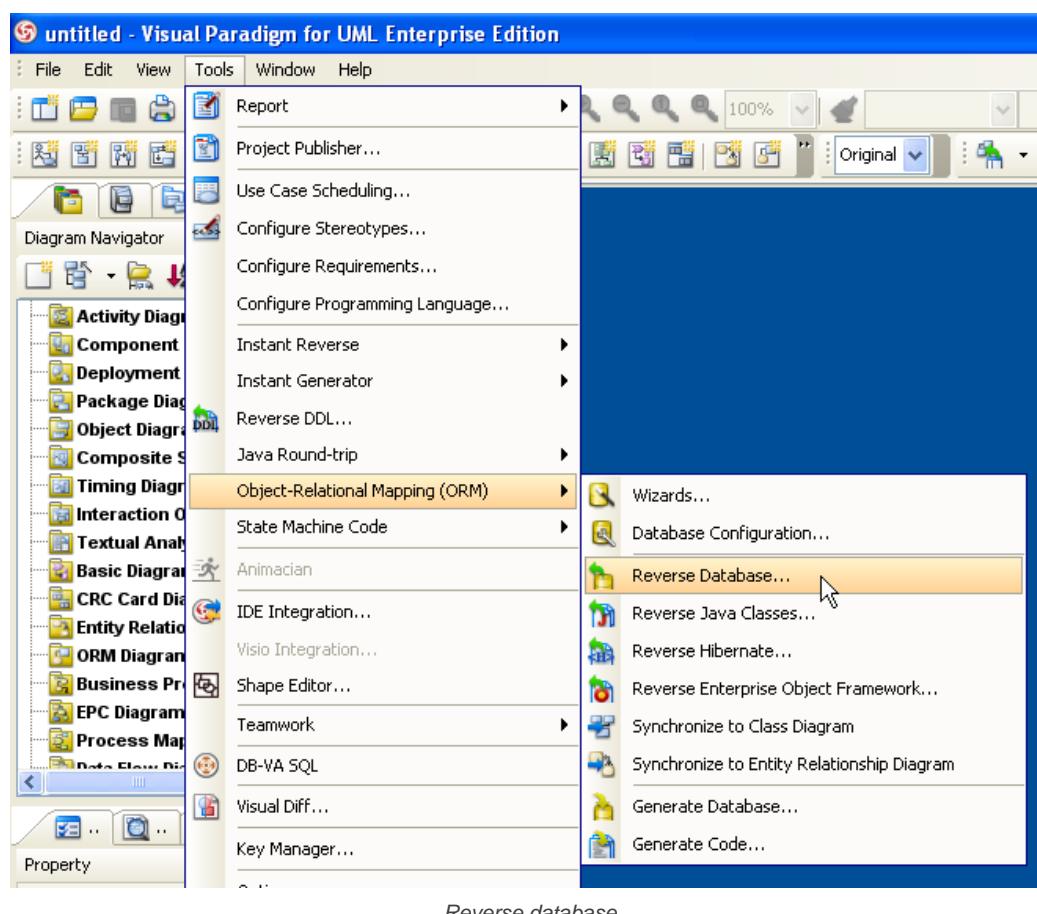
Import button

7. The setting was imported to current project.

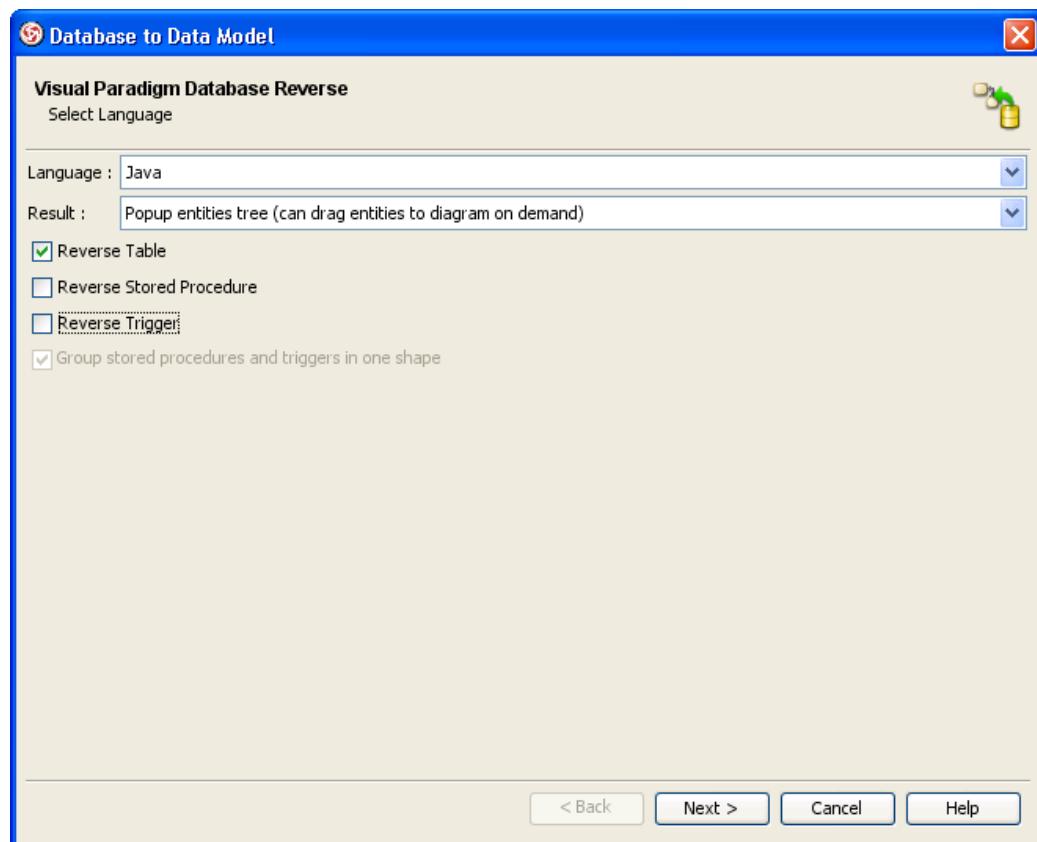


Reversing database

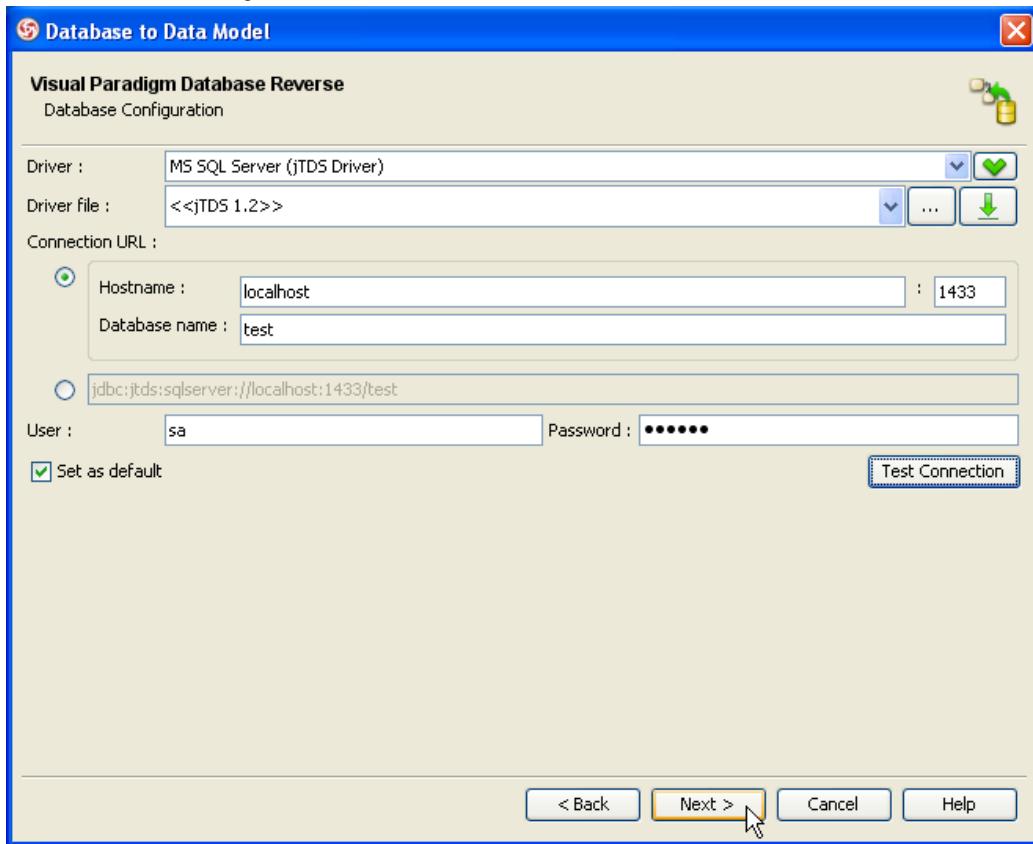
1. Create a new project.
2. Select Tools > Object-Relational Mapping (ORM) > Reverse Database... from the main menu.



3. Select Reverse Table and click Next > button to continue.

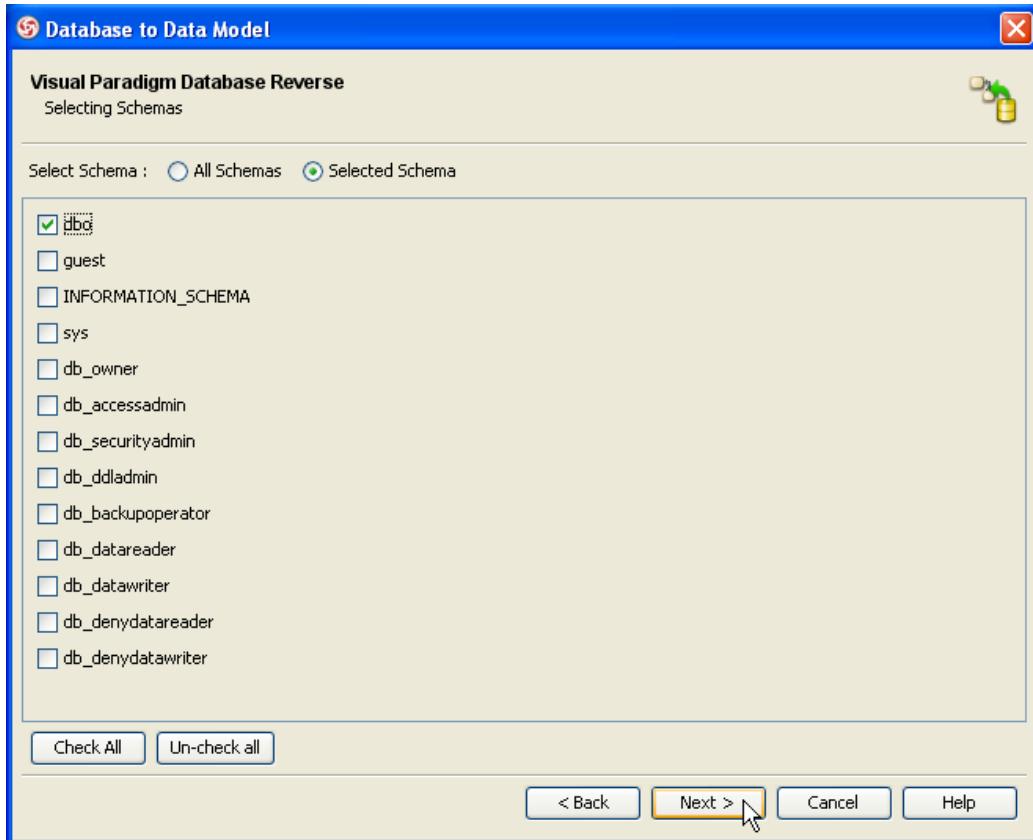


4. Fill in the database setting and click **Next >** button.



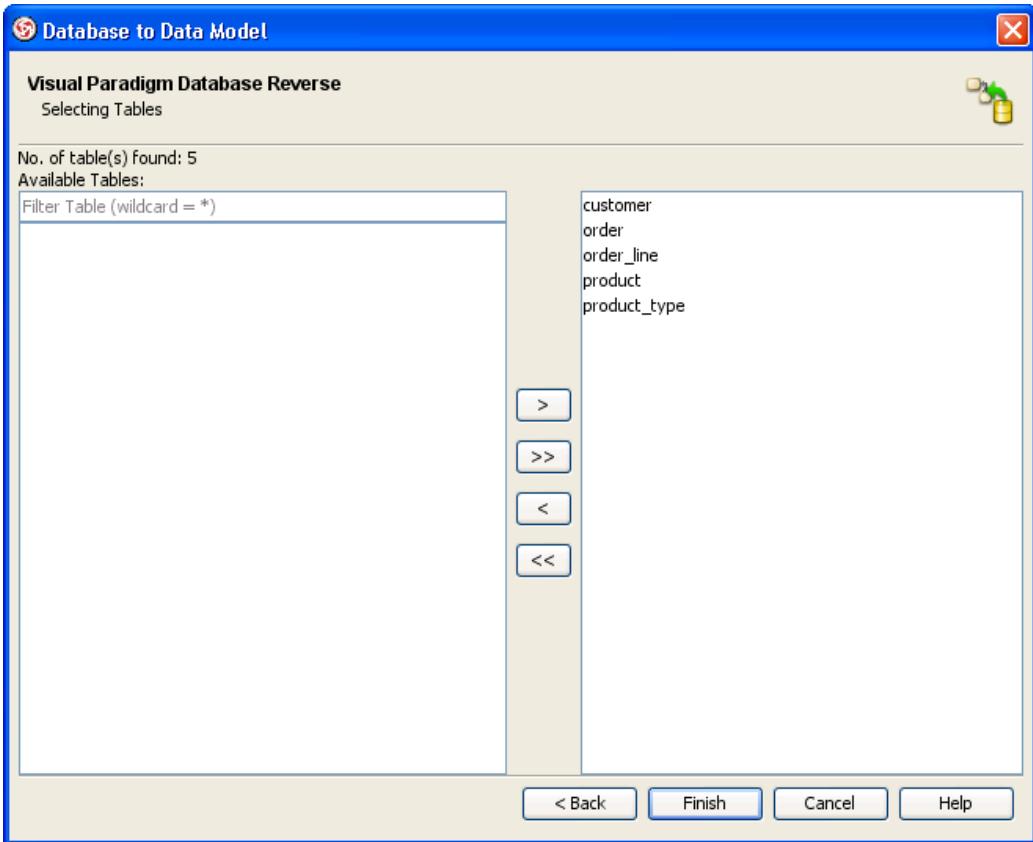
Database setting

5. Select the **Selected Schema** and check the schema containing your tables, and click **Next >** button.



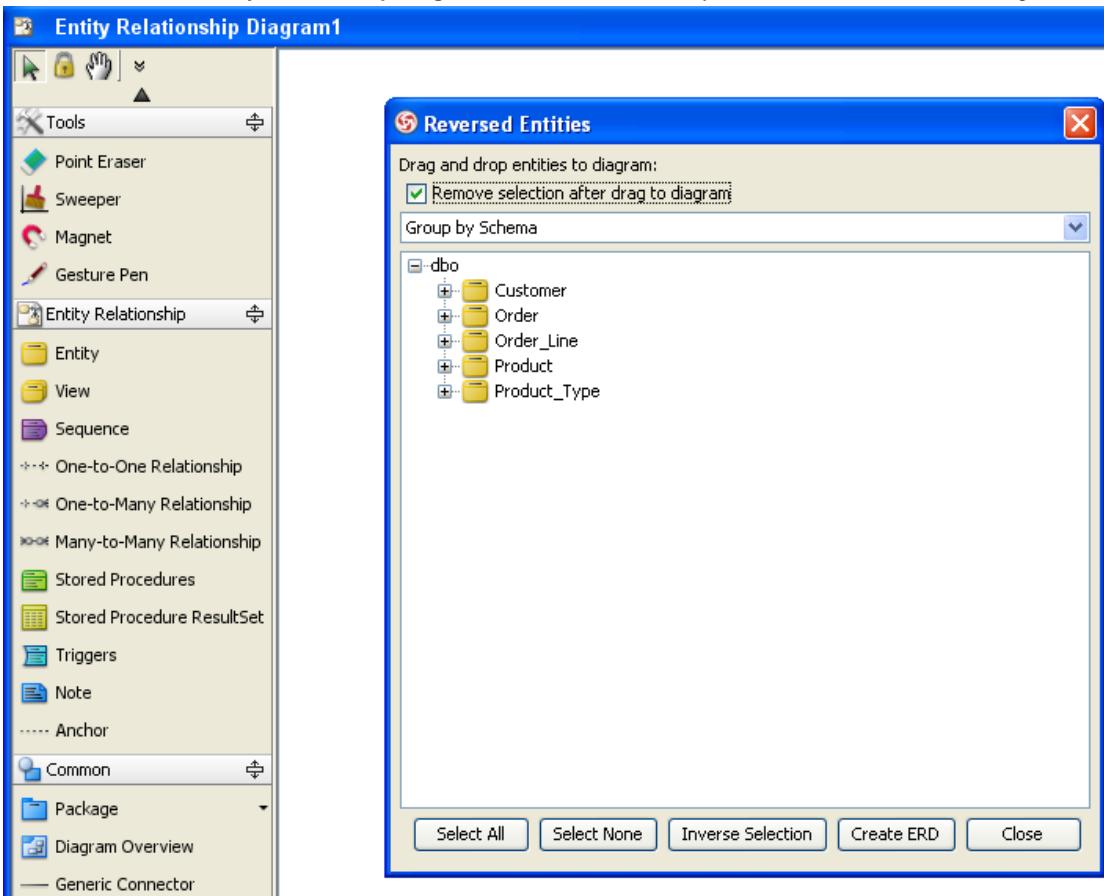
Select schema

6. Select the tables you want to reverse, and click the **Finish** button to start reverse.



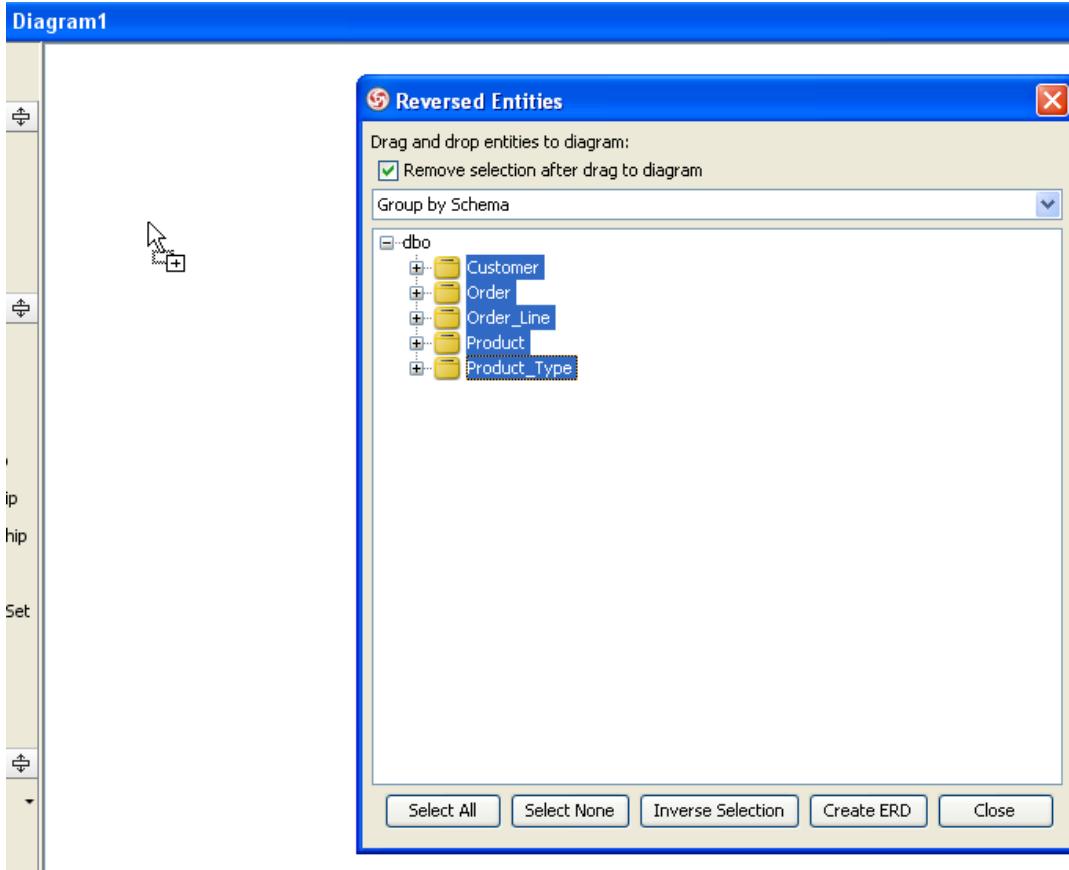
Select tables

7. After reverse, a new **Entity Relationship Diagram** is created automatically, with a **Reversed Entities** dialog.



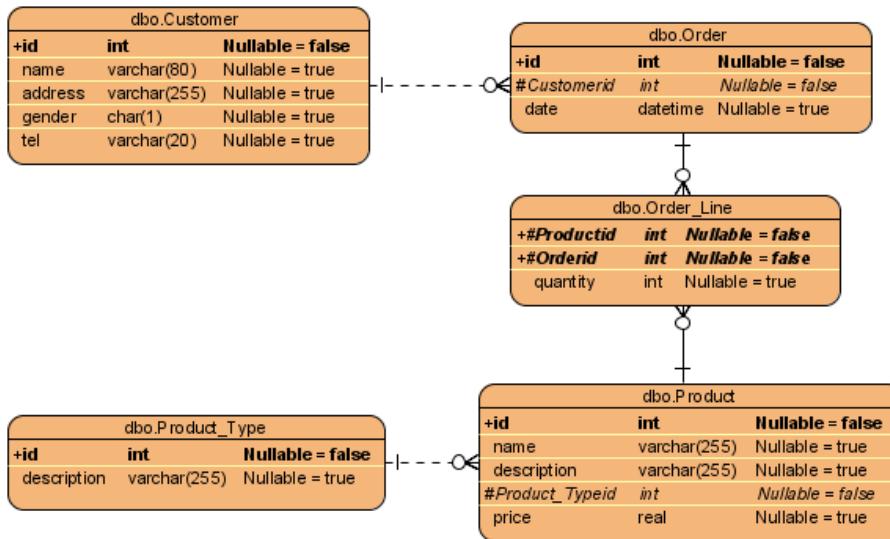
Reversed entities

8. Select the tables and drag on the diagram.



Drag and drop tables to diagram

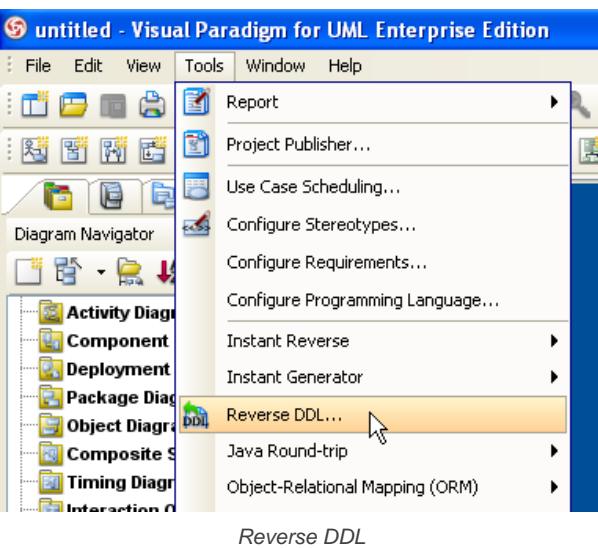
9. The tables were created on the diagram, click **Close** button to finish reversing database.



Reversed ERD

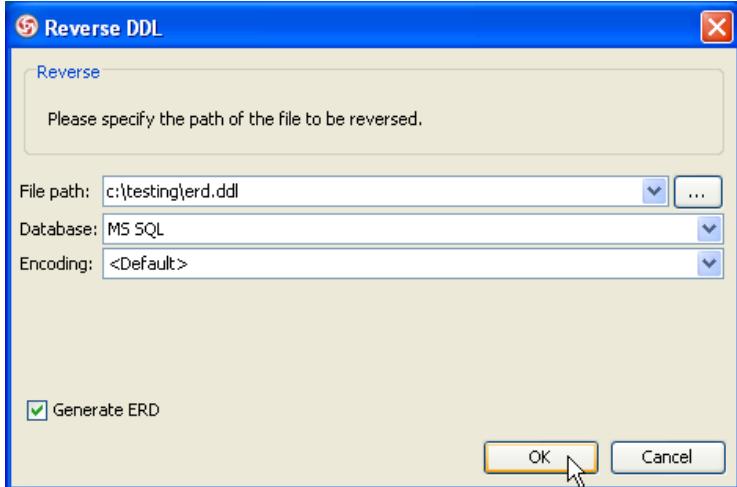
Reversing DDL

1. Create a new project.
2. Select **Tools > Reverse DDL...** from the main menu.



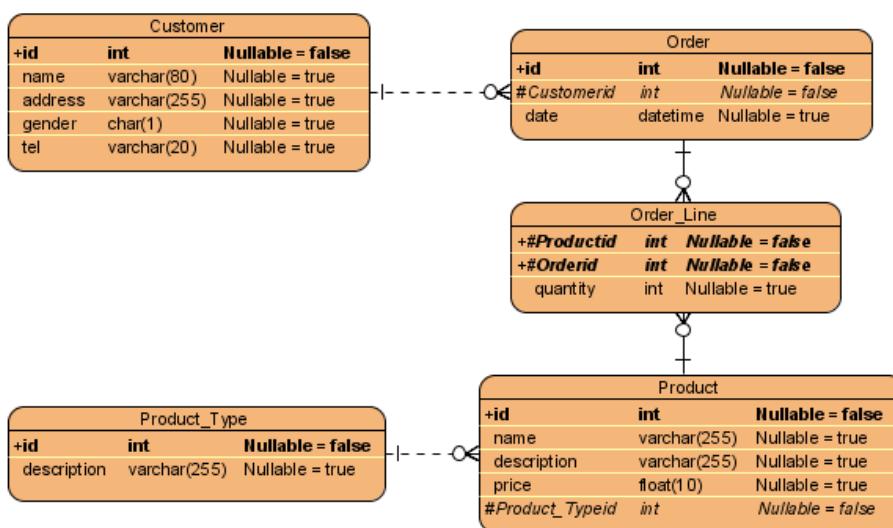
Reverse DDL

3. Fill in the filename of DDL and select database in **Reverse DDL** dialog. Click **OK** button to continue.



Reverse DDL dialog

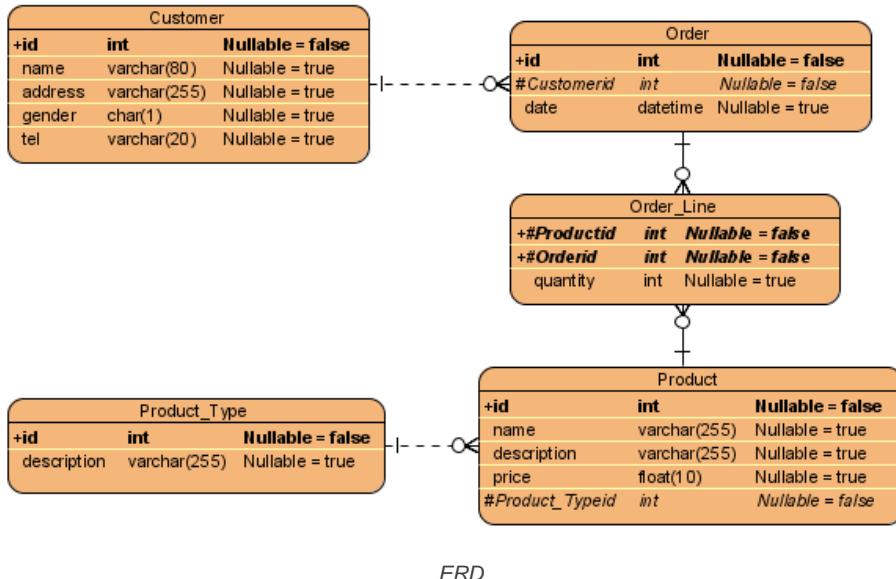
4. An Entity Relationship Diagram is created with the reversed entities.



Reversed ERD

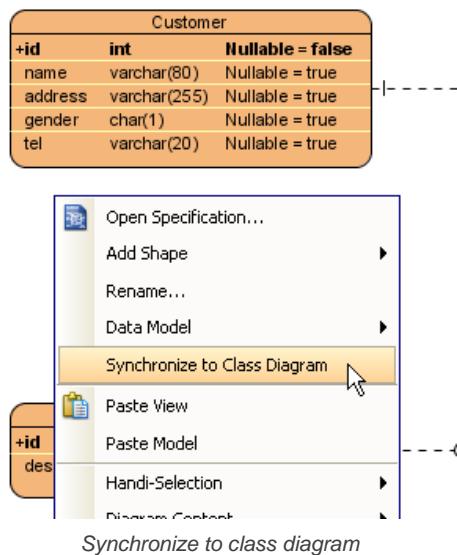
Generate class diagram from ERD

1. Open an ERD.

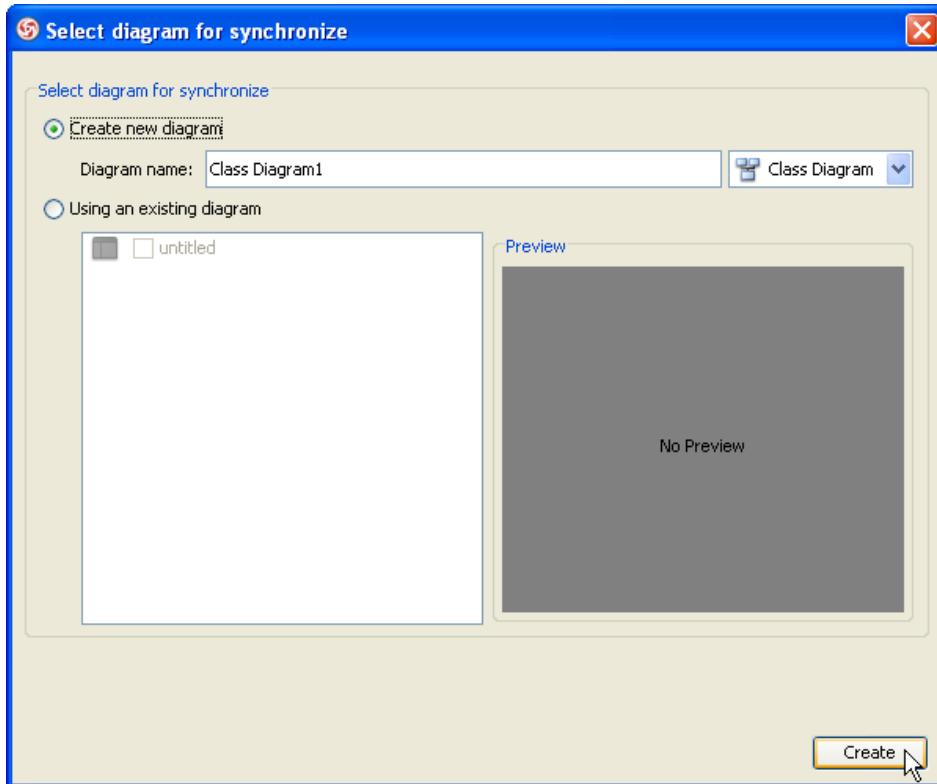


ERD

2. Right click on diagram, select **Synchronize to Class Diagram** from the popup menu.

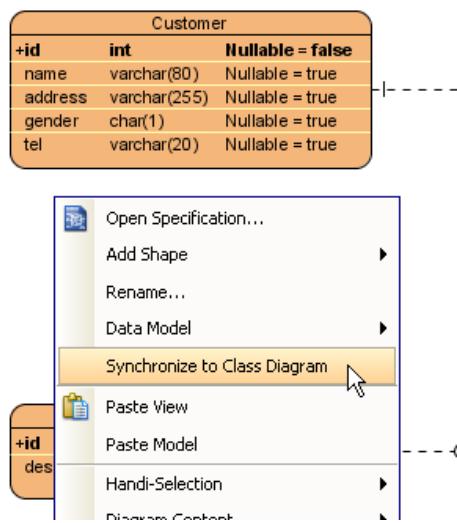


3. For the first time synchronize ERD to class diagram, it will show a **Select diagram for synchronize** dialog. Either select **Create new diagram** and specify diagram name, or select **Using an existing diagram** and select an existing diagram from the list. Click **Create** button to continue.



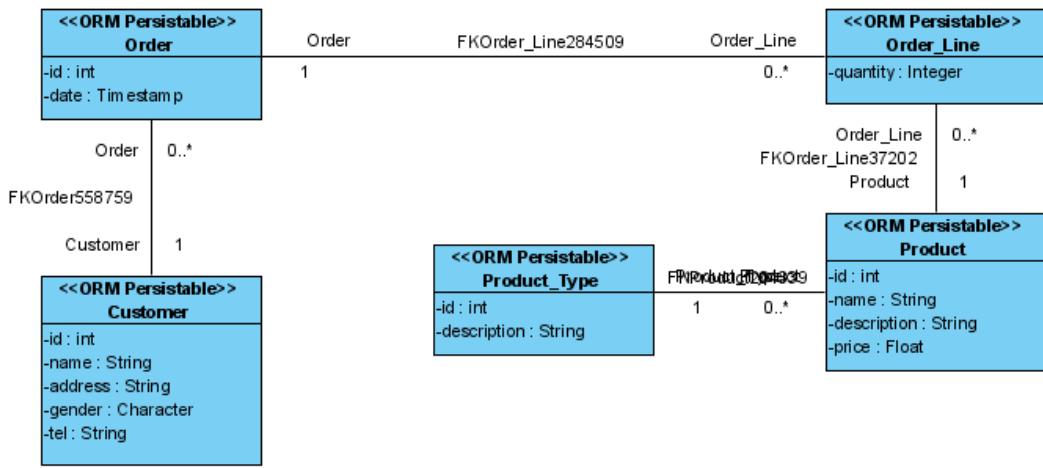
Select diagram for synchronize

4. If there are new ERD model elements created since last synchronize, a **Synchronize to Class Diagram** dialog will show for you to rename the generated model elements. Click **OK** button after finish renaming.



Synchronize to class diagram

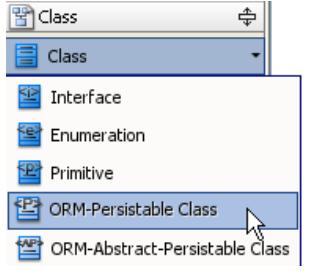
5. A class diagram with generated classes, associations created.



Generated class diagram

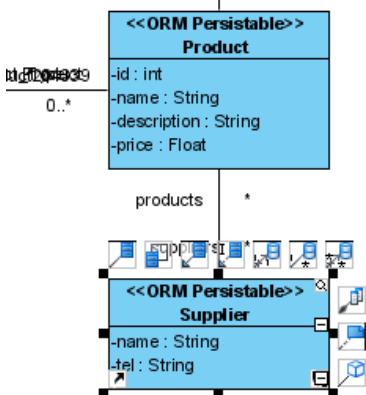
Synchronize from class diagram to ERD

1. Select ORM-Persistable Class from the diagram palette.



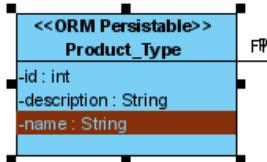
Create ORM-Persistable class

2. Click on the diagram to name it as *Supplier*, create attributes and association as follow.



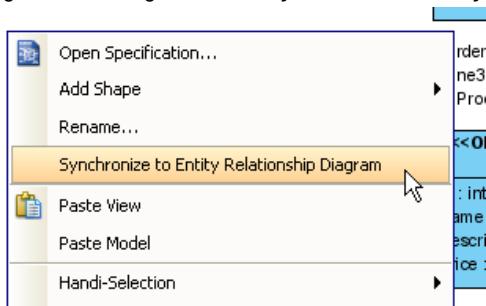
Create Supplier class

3. Add *name* attribute to Product_Type class.



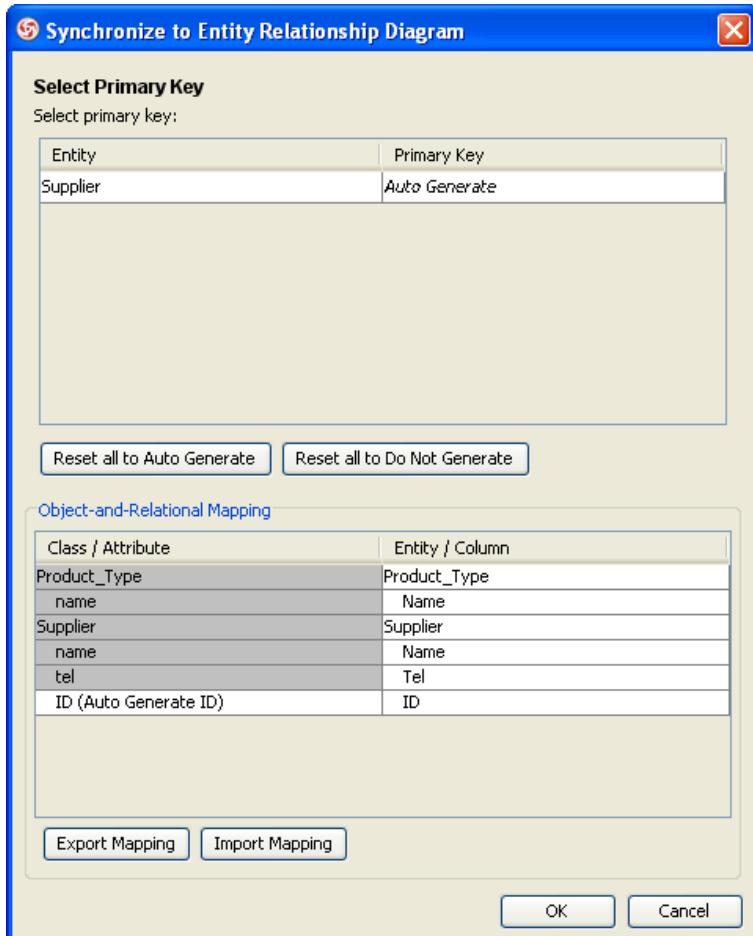
Create attribute

4. Right click on diagram, select **Synchronize to Entity Relationship Diagram** from the popup menu.



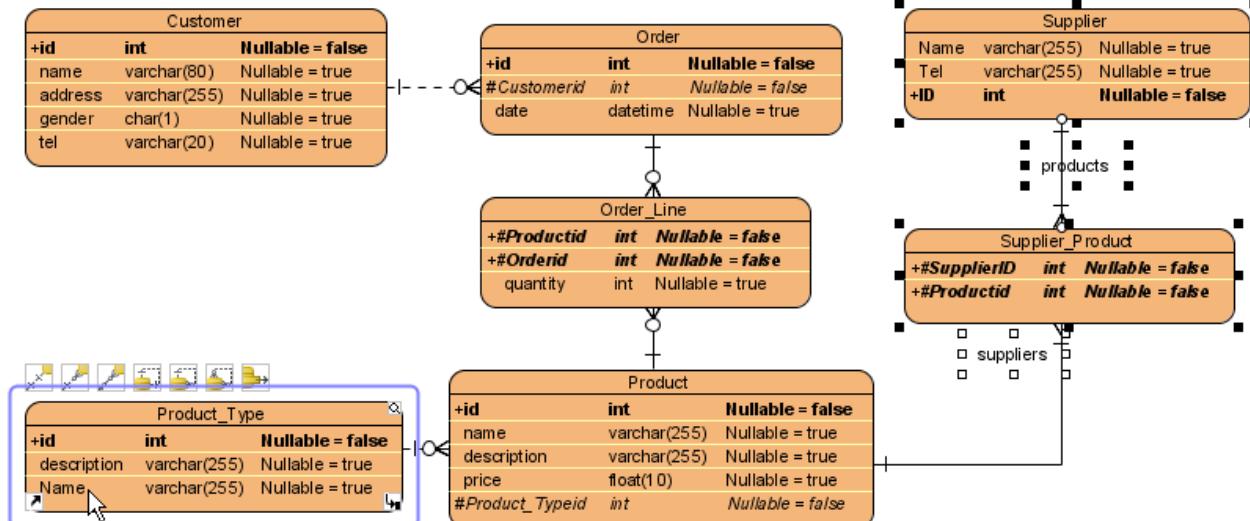
Synchronize to Entity Relationship Diagram

5. A **Synchronize to Entity Relationship Diagram** dialog appears. Select **Auto Generate** in **Select Primary Key** table and rename the entity/column if necessary. Click **OK** button to continue.



Synchronize to Entity Relationship Diagram dialog

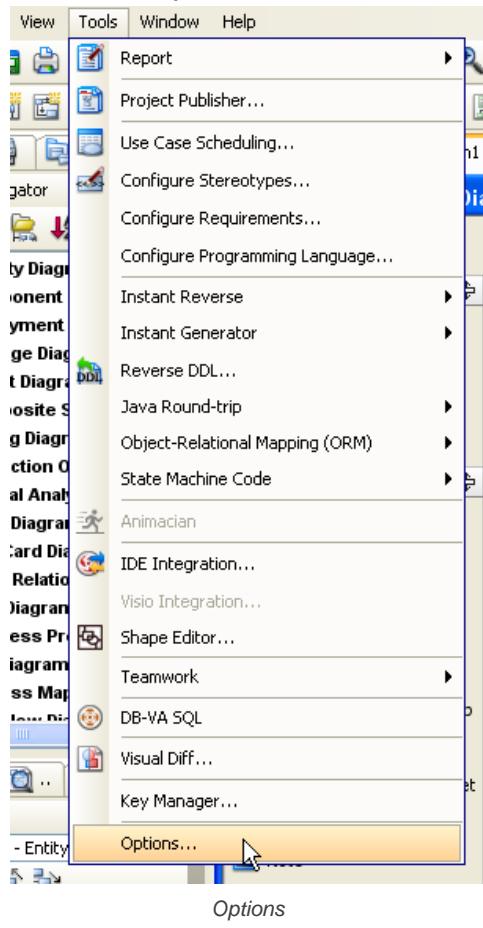
6. New entities and column are created on the ERD.



Synchronized ERD

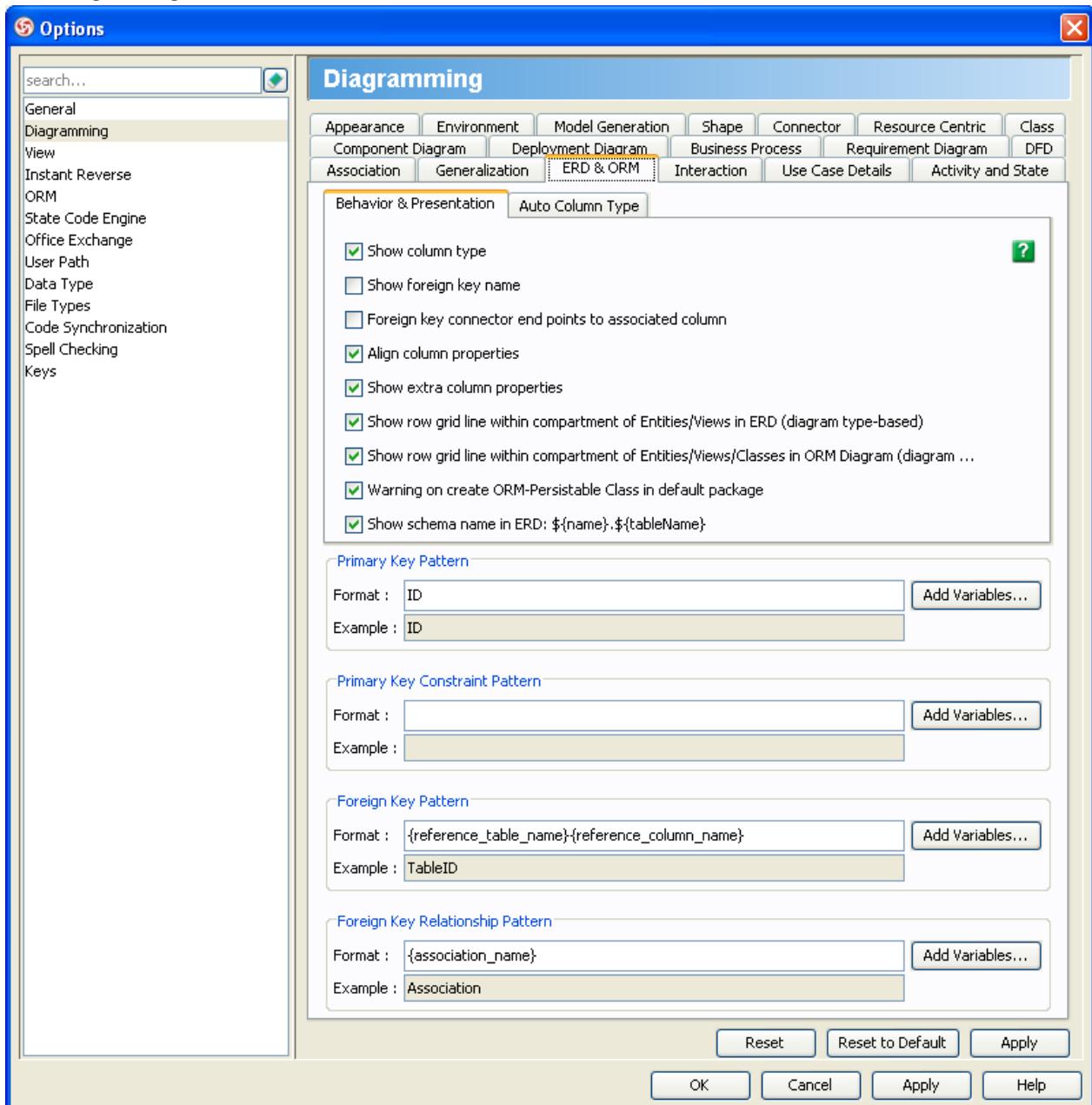
Configure key naming pattern

1. Select Tools > Options... from the main menu.



Options

2. Select **Diagramming** from the list on the left, then select **ERD & ORM** tab.



ERD & ORM option

3. Change to **Primary Key Pattern** to *PK_ID* and **Foreign Key Pattern** to *FK_{reference_table_name}*.

Primary Key Pattern

Format :	PK_ID	Add Variables...
Example :	PK_ID	

Primary Key Constraint Pattern

Format :		Add Variables...
Example :		

Foreign Key Pattern

Format :	FK_{reference_table_name}	Add Variables...
Example :	FK_Table	

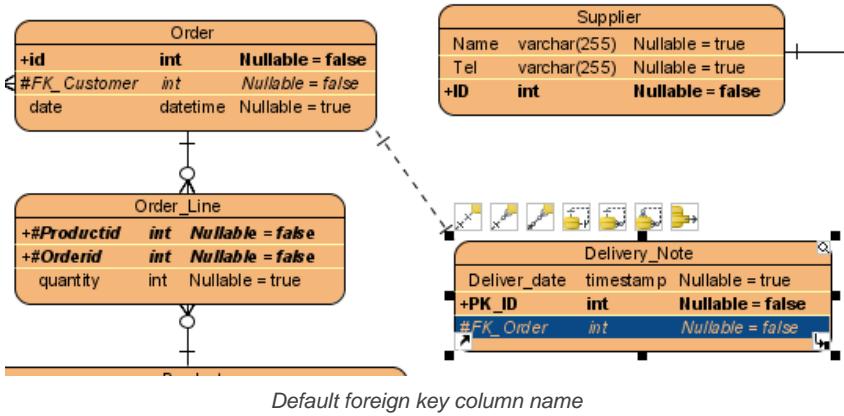
Primary key and foreign key pattern

4. Create a new ORM-Persistable class on class diagram, synchronize to ERD. In Synchronize to Entity Relationship Diagram dialog, the primary key column name is generated as *PK_ID* by default.

Class / Attribute	Entity / Column
Delivery_Note	Delivery_Note
deliver_date	Deliver_date
ID (Auto Generate ID)	PK_ID

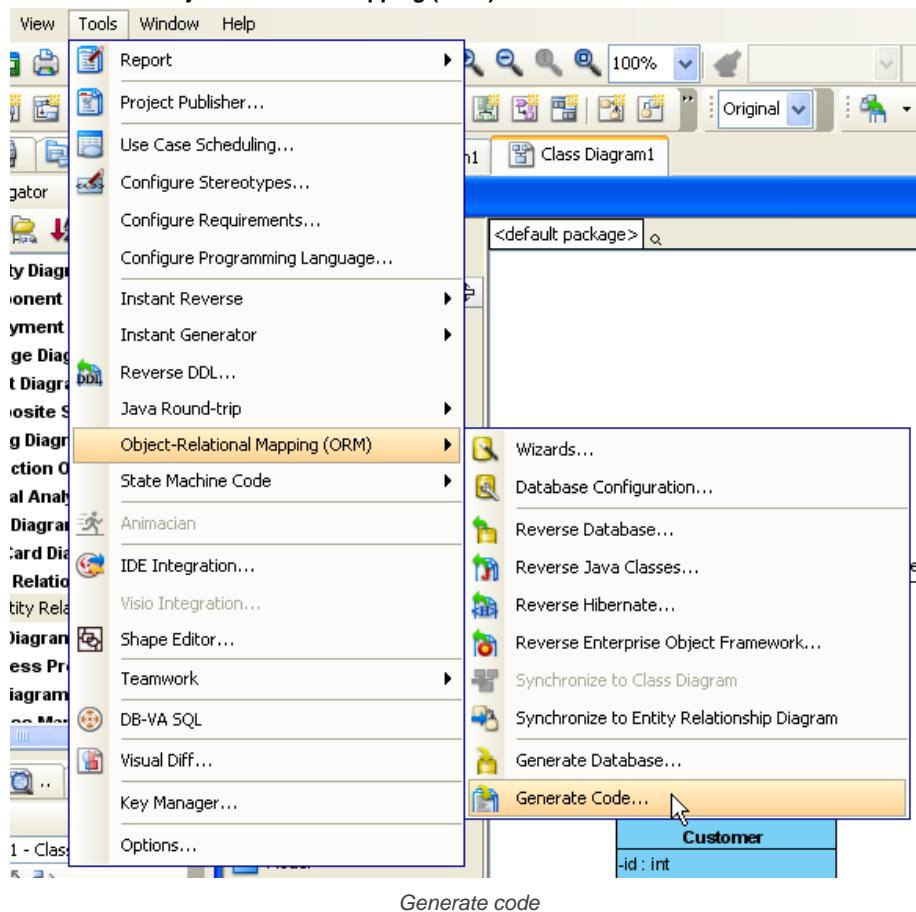
Default primary key column name

5. Create a relationship from *Order* entity to a new entity, a foreign key column named *FK_Order* is created automatically.

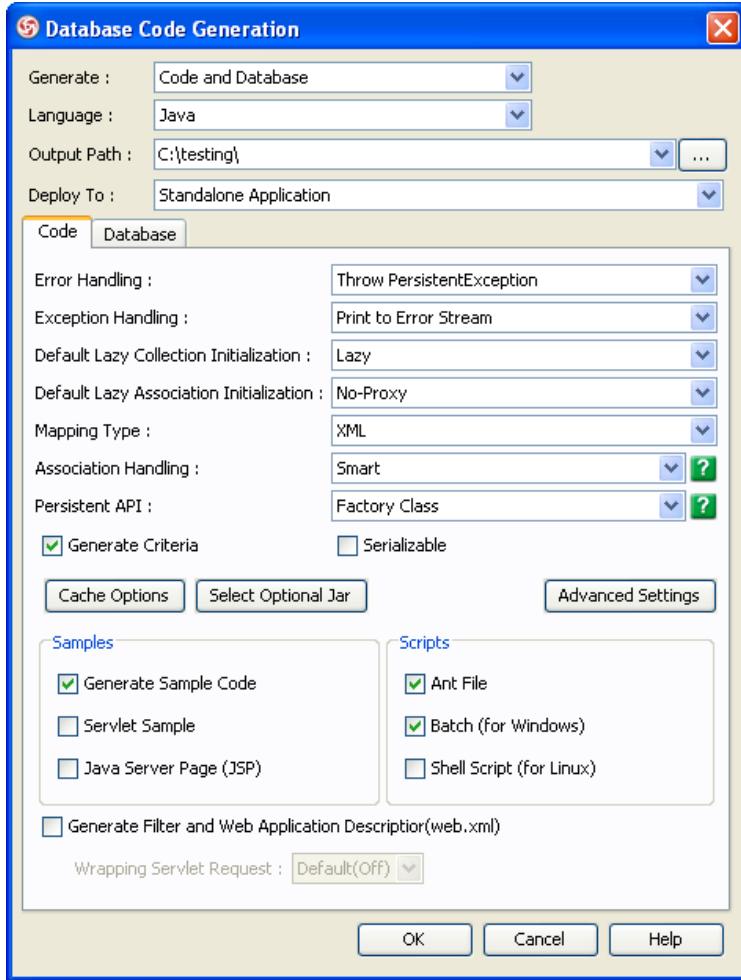


Generating code and database

1. Select Tools > Object-Relational Mapping (ORM) > Generate Code... from the main menu.



2. Fill in the **Output Path** and select options:



Database code generation

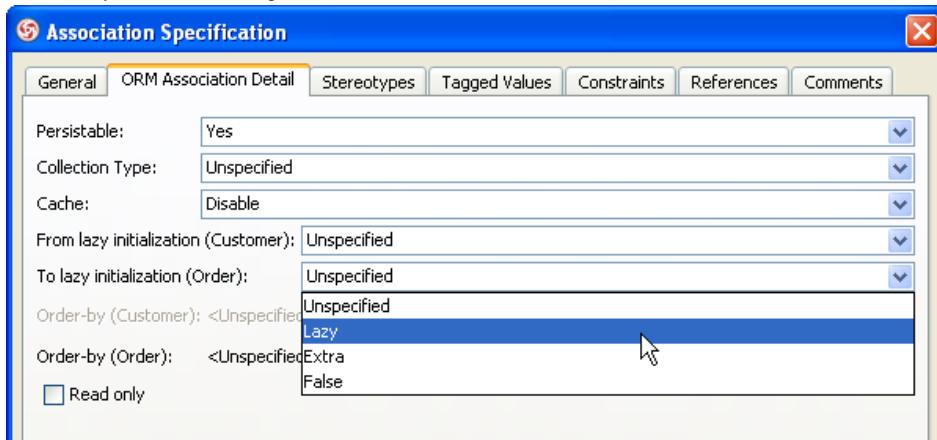
- Error Handling:
 - **Return false/null** - query method error return null, save/update/delete error return false.
 - **Throw PersistentException** - throw PersistentException when error occurs.
 - **Throw RuntimeException** - throw RuntimeException when error occurs.
 - **Throw User Defined Exception** - throw the exception specified in **User Defined Exception** field when error occurs.
- Exception Handling:
 - Do not Show - do nothing when exception throw.
 - Print to Error Stream - print the exception to System.err.
 - Print to log4j - log the exception to log4j.

3. Click **OK** button to start code generation.

Lazy collection setting

Setting lazy collection for association

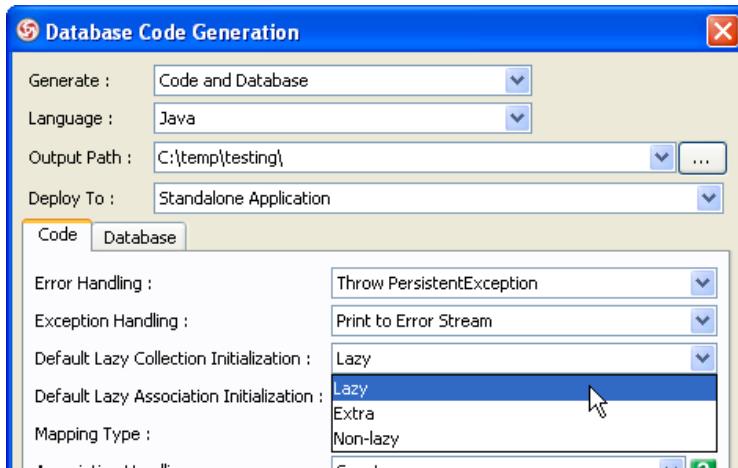
1. Open specification dialog of association.
2. Switch to ORM Association Detail tab, select **Lazy** or **Extra** for **From lazy initialization** or **To lazy initialization**, depending on which side multiplicity is *. Lazy collection is fetched when the application invokes an operation upon that 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

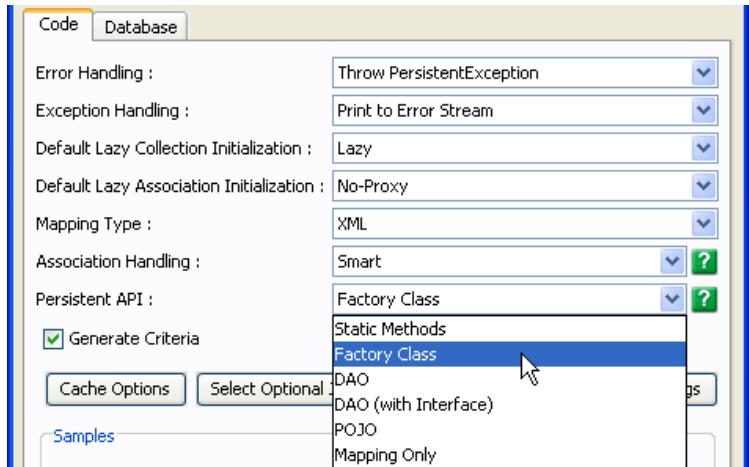
1. Open Database Code Generation dialog.
2. Specify a value for **Default Lazy Collection Initialization**.



Default lazy collection setting

Persistent API

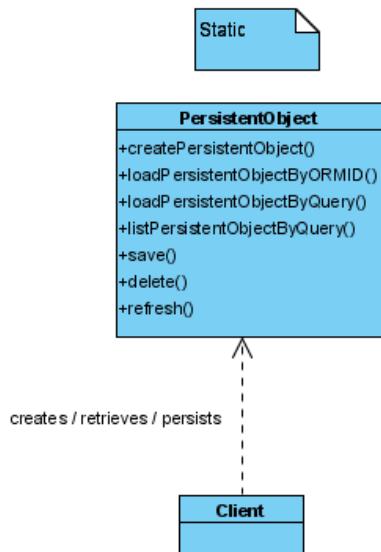
The persistent API setting supports various styles for generated code. It can be configured in **Database Code Generation** dialog.



Persistent API setting

Static methods

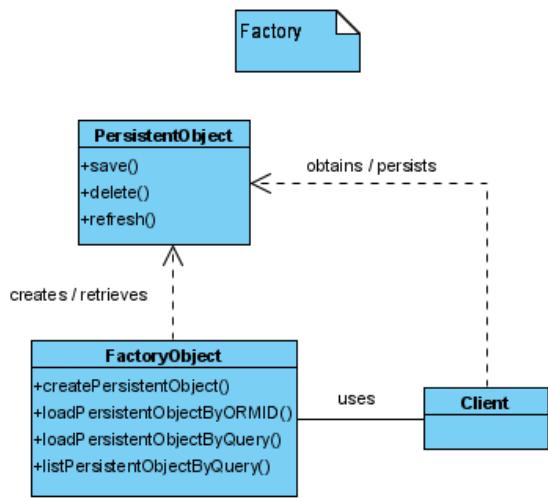
Static methods generate all persistent methods in the persistent class, client can access the methods in the same persistent object.



Static methods

Factory class

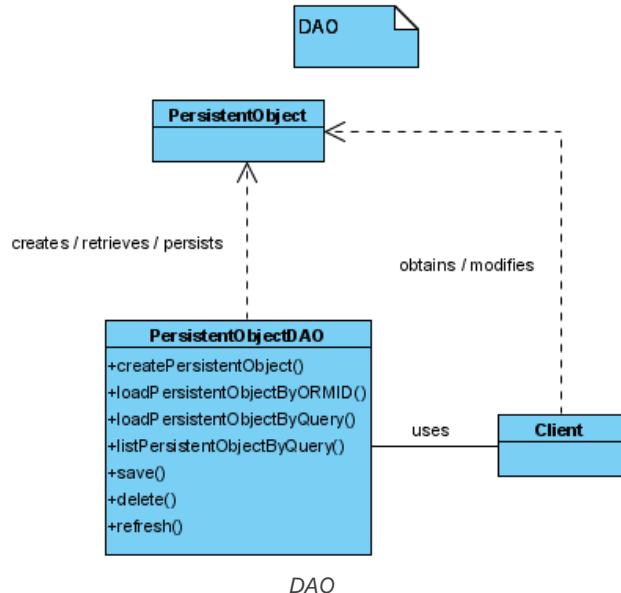
Factory class generate save/delete/refresh methods in persistent class, other persistent methods that return persistent object are generated in factory class.



Factory class

DAO

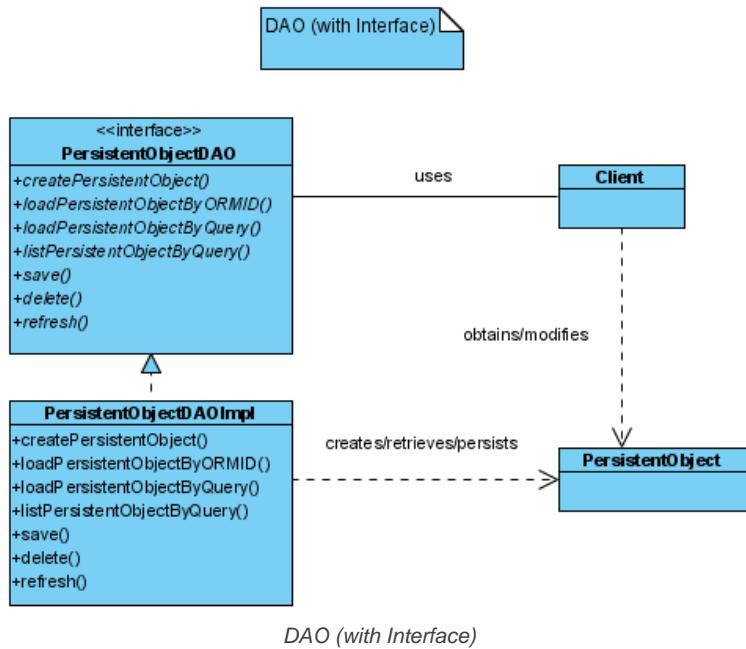
DAO generate all persistent methods in DAO class, a DAO class is generate for each persistent class.



DAO

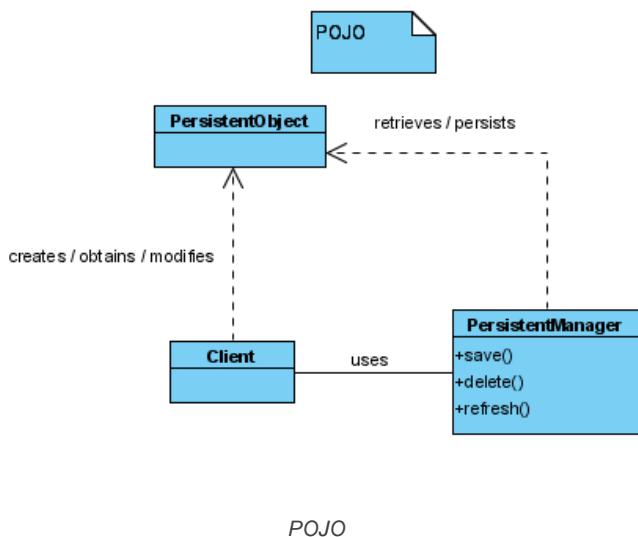
DAO (with interface)

DAO (with Interface) generate 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.



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 generate the XML mapping file required for ORM.

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 code demonstrate how to insert a *Product* record:

Selecting records

Factory method provides a convenient listByQuery method, accept condition and order by as parameter, and return array of persistent object.

The following code 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:

Another useful method to select a persistent object by ID is loadByORMID. The follow code demonstrate how to select a *Product* record by ID.

Updating records

1. Select a persistent object from database.
2. Update the persistent object.
3. Save persistent object with save method.

The following code demonstrate how to update a *Product* record:

Deleting records

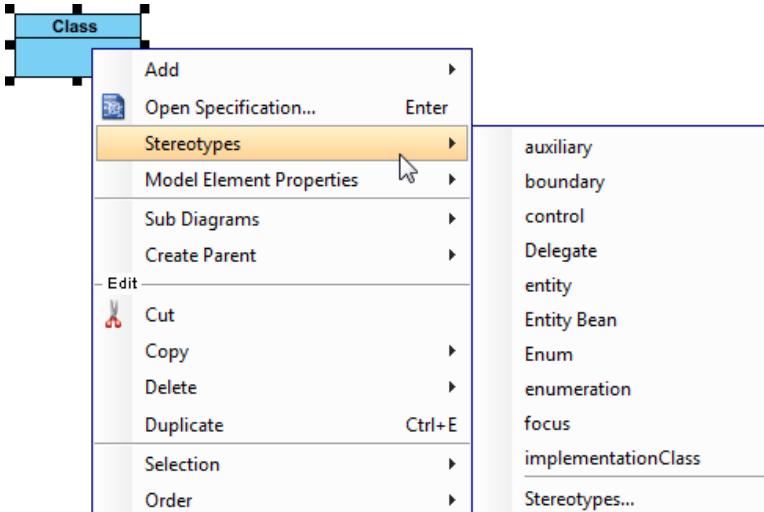
1. Select a persistent object from database.
2. Delete persistent object with delete method.

The following code demonstrate how to delete a *Product* record:

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 VP-UML, 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:

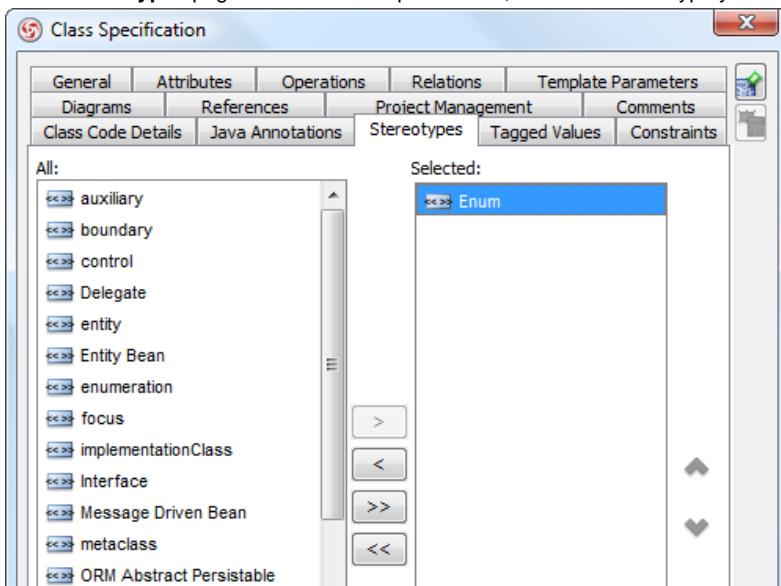
1. Right click on the model element, or the view of the model element that you want to apply stereotype to. Select **Stereotypes** from popup menu.



To 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** page of the element specification, select the stereotype you want to apply, then click **>** to assign it to the **Selected** list.

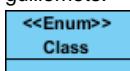


Stereotype *Enum* is selected

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.. If you want to apply ALL available stereotypes to model element, click **>>**, and likewise, clicking on **<<** removes all the applied stereotypes.

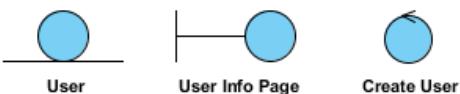
3. Click **OK** to confirm. The stereotype will then be shown within a pair of guillemets above the name of the model element. If multiple stereotypes are applied, the names of the applied stereotypes are shown as a comma-separated list with a pair of guillemets.



Stereotype *Enum* is applied to a class

Robustness analysis icon

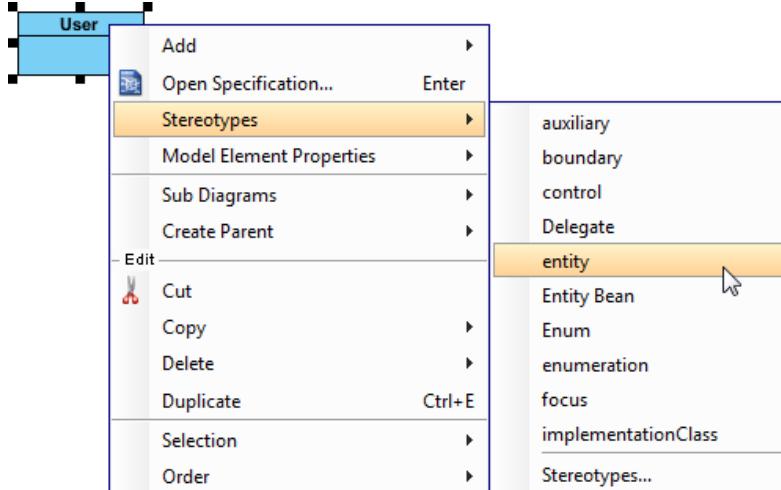
Robustness analysis helps to find out the relationships between actor, boundary, control and entity objects.



Robustness analysis icons

To draw a robustness analysis diagram with robust analysis symbols:

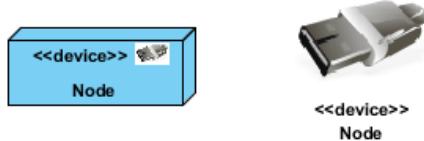
1. Create a class in diagram.
2. Depending on the type of robustness analysis symbol you want to create, apply either boundary, control and entity stereotype to the class.



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.

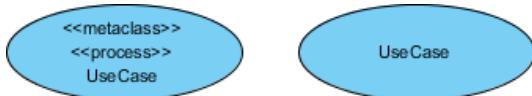


Different presentations of a model element with a stereotype that has icon defined

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.



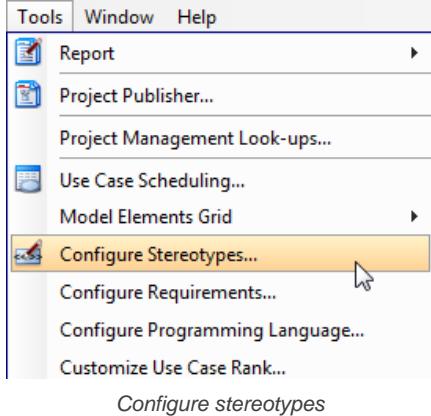
A use case with stereotype names shown and hidden

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:

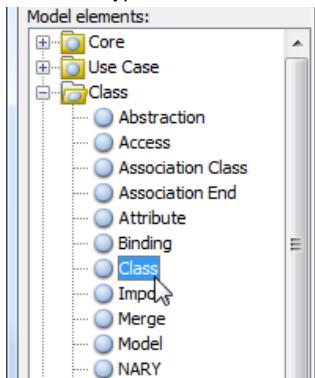
1. Select **Tools > Configure Stereotypes...** from the main menu.



2. Click on the drop down menu **Scope** at the top left corner of the **Configure Stereotypes** dialog box, 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 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.

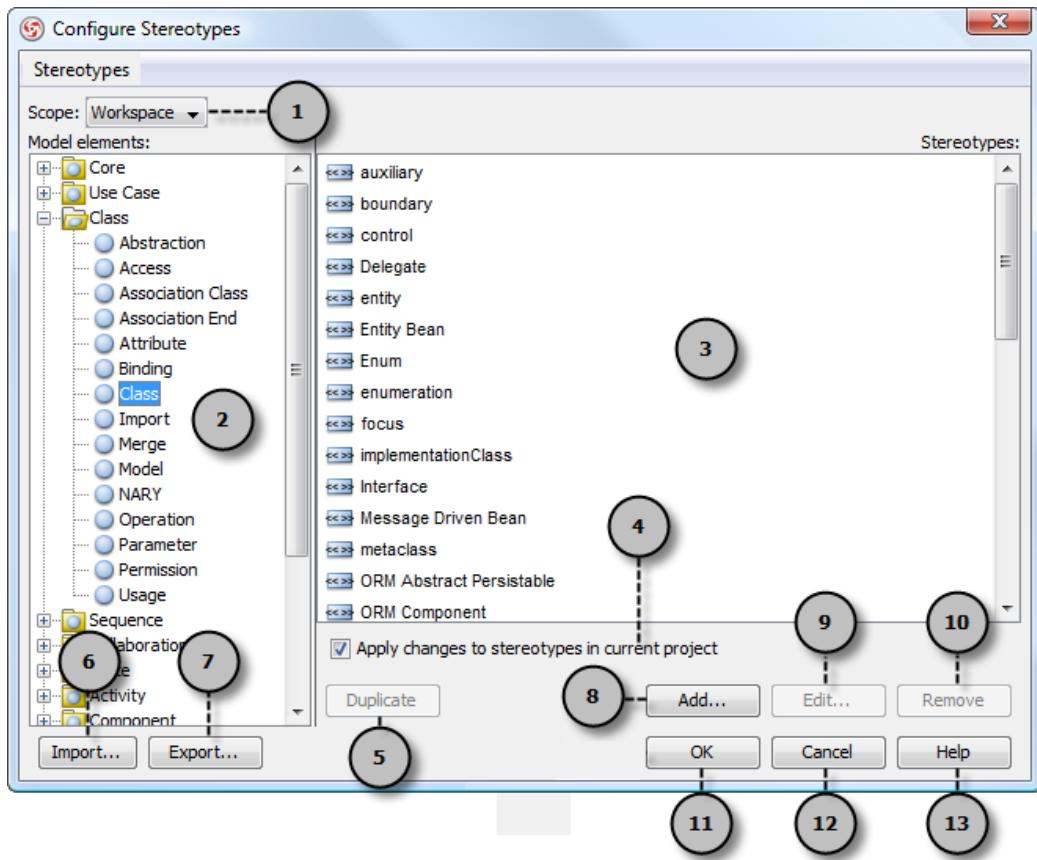
3. Select the type of model element that you want to add stereotype or edit its existing stereotypes.



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](#)



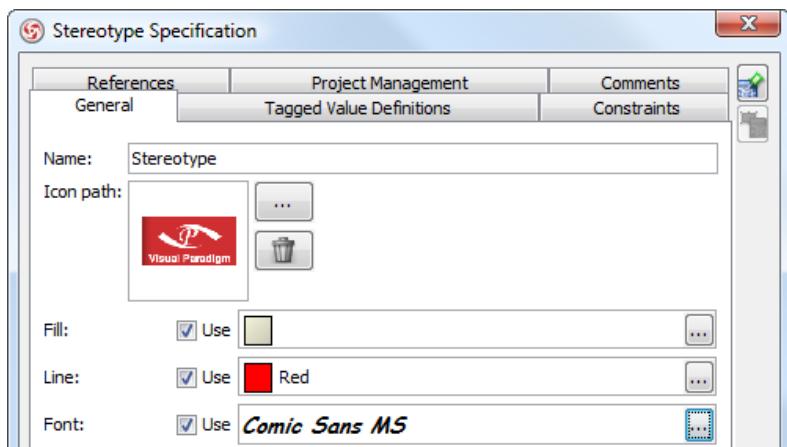
An overview of **Configure Stereotypes** dialog box

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, VP-UML will only add and update stereotypes from XML. When unchecked, VP-UML 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.

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 show 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.

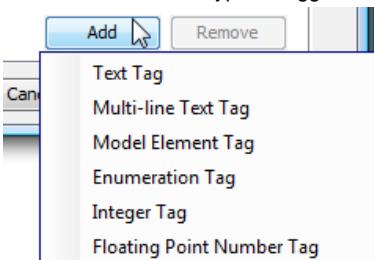
Fill, line and font styles will be applied automatically to model elements that has the stereotype applied. To adjust fill/line/font, check the corresponding Use button. Then, click on the ... button to edit the settings.

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 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. Select **Tools > Configure Stereotypes...** from the main menu.
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.

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.

The screenshot shows the 'Stereotype Specification' dialog box. At the top, there are tabs for 'References', 'Project Management', and 'Comments'. Below these are three sub-tabs: 'General' (selected), 'Tagged Value Definitions', and 'Constraints'. A vertical toolbar on the right contains icons for 'New', 'Edit', 'Delete', and 'Save'. The 'Tagged Value Definitions' tab displays a table with columns 'Name', 'Type', and 'Default Value'. The rows are:

Name	Type	Default Value
version	Floating Point Number	0.0
release-date	Text	<Unspecified>
author	Text	<Unspecified>
language	Enumeration	java
description	Multi-line Text	<Unspecified>

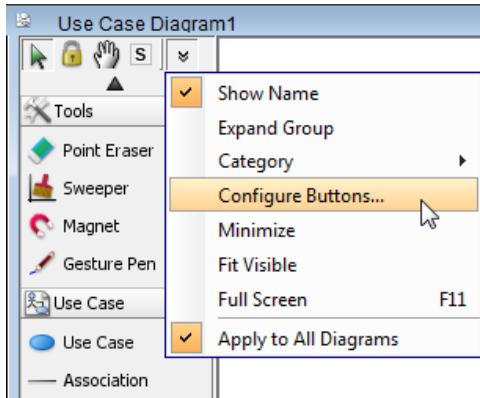
Tags defined for an API stereotype

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.

Shortcut of creating stereotyped model element

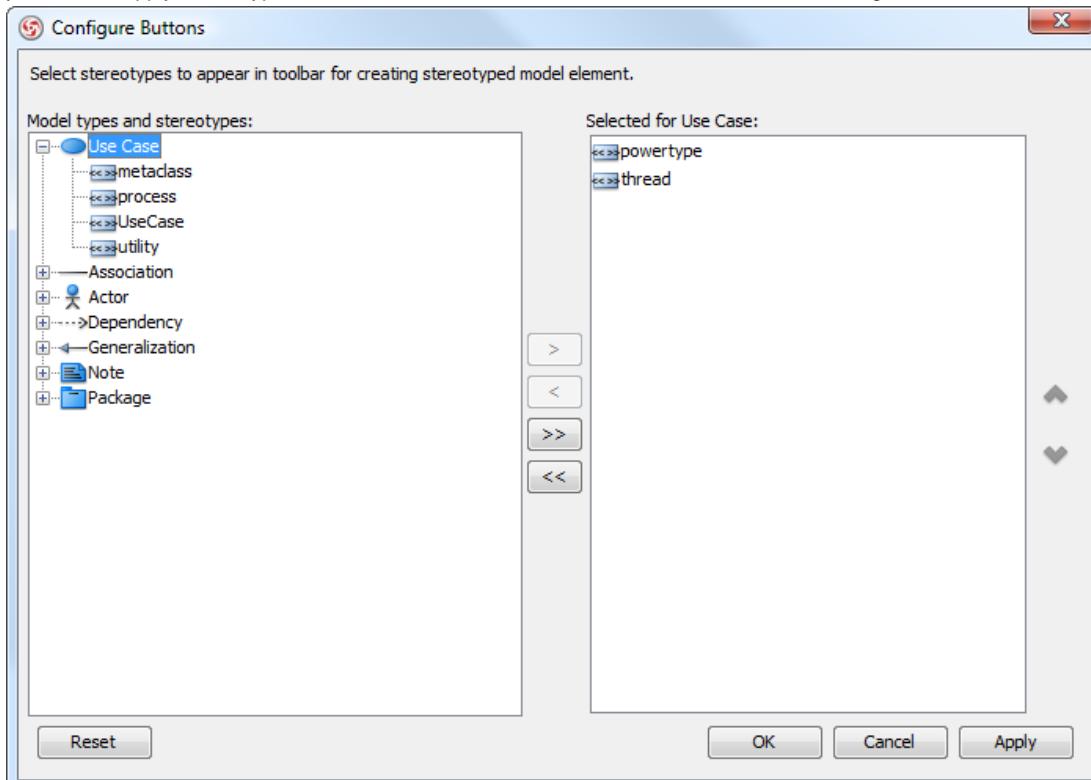
One of the ways of creating a shape is by selecting the element type in diagram toolbar, and clicking on diagram to create a shape of that type. If then you want to apply a stereotype to that model element, you will open the specification dialog box and make a stereotype selection. These steps can be 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, click on the double down arrow button at the top of diagram toolbar, then select **Configure Buttons...** from the popup menu.



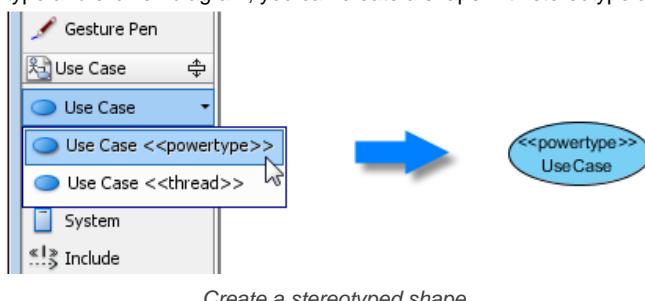
To configure buttons

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

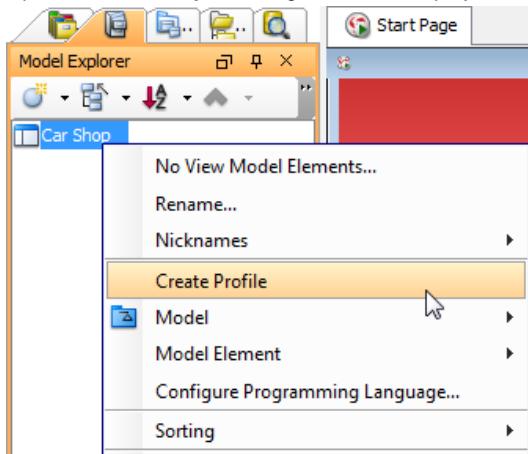
3. After all, you can find the shortcuts in diagram toolbar, under the selected type(s) of model elements. By selecting a stereotyped model element type and click on diagram, you can create a shape with stereotype applied.



Creating a profile

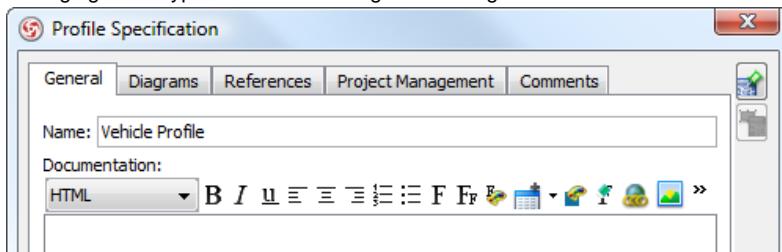
A UML profile provides a generic extension mechanism for customizing UML model elements for different purposes. Profiles are defined using stereotypes and tagged values definition for specific types of model elements. You can tailor UML meta model for different domains and platforms by creating and developing a profile.

1. Open the **Model Explorer**. Right click on the project root node and select **Create Profile** from the popup menu.



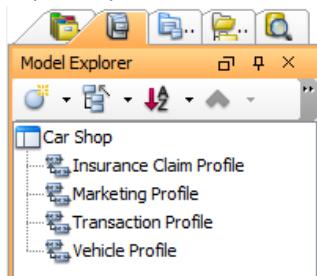
Creating a profile

2. Name the profile in **Profile Specification** and click **OK** to confirm. The naming should be clear enough to identify the part of the domain you want to create a profile for. For example, a Vehicle profile for managing stereotypes around different car types; a Transaction profile for managing stereotypes related to trading and loaning of cars.



Naming the profile

3. Repeat step 1 and 2 to create all the profiles.



All profiles are created

A profile consists of domain-specific stereotypes and tagged values definition. To develop a profile, you need to create a profile diagram. For details about how to create and draw a profile diagram, read the next chapter.

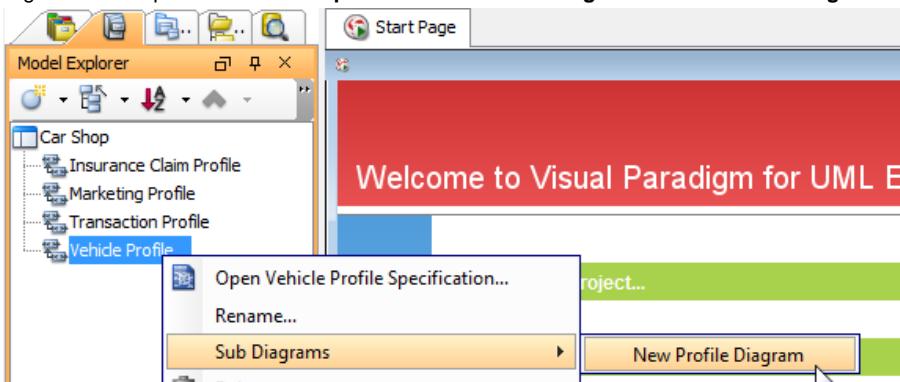
Drawing a profile diagram

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

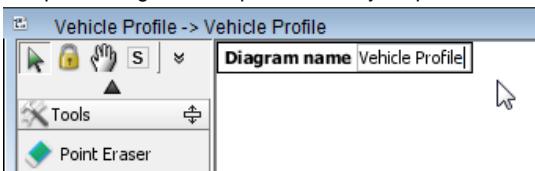
To create a profile diagram:

1. Right click on a profile in **Model Explorer** and select **Sub Diagrams > New Profile Diagram** from the popup menu.



To create a profile diagram

2. Name the diagram and press **Enter** to confirm. By default, the name of profile is applied as the name of diagram. If you attempt to create only one profile diagram for a profile, and if your profile was well-named, you can keep using the profile name as diagram name.

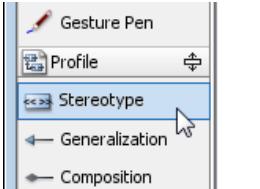


Naming a profile diagram

Drawing a stereotype

To draw a stereotype in profile diagram:

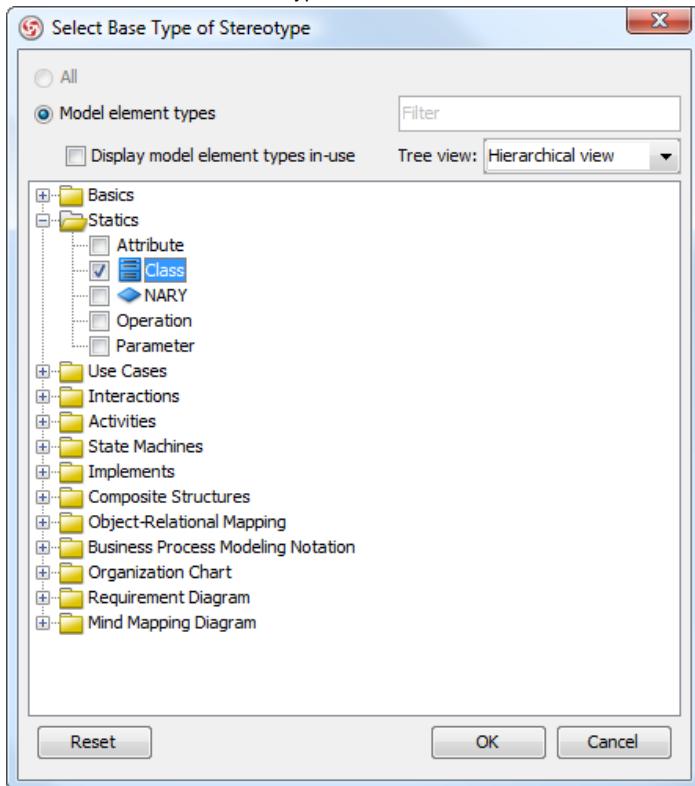
1. Select **Stereotype** in diagram toolbar.



*Selecting Stereotype
in diagram toolbar*

2. Click on the diagram to create a stereotype.

3. In the **Select Base Type of Stereotype** dialog box, 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 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.

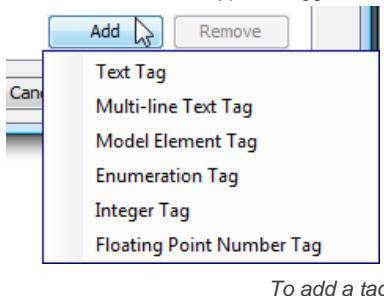
Defining tagged values for stereotypes

A stereotype may have properties, which may be referred 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 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** dialog box, open the **Tagged Value Definitions** tab.

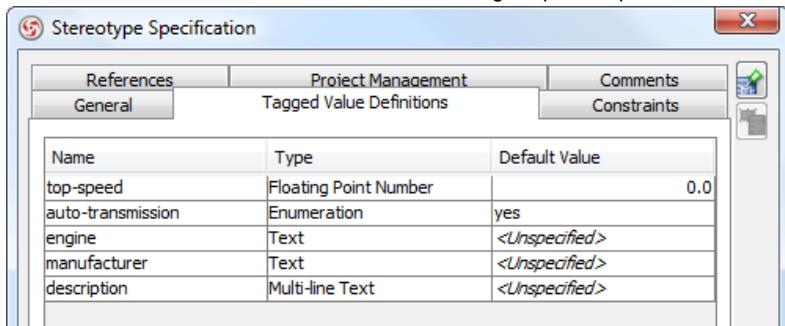
3. Click **Add**. Select the type of tagged value to define. The type of tagged value limits the type of content user can enter for a tag.



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.

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.

<<Stereotype>>
Vehicle (Class)
top-speed : Float = 0.0
auto-transmission : Enum = yes
engine : Text
manufacturer : Text
description : Multi-line Text

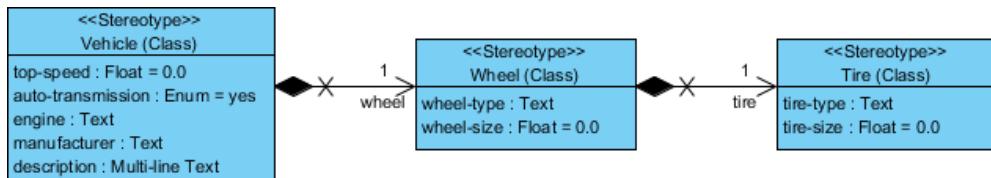
Stereotypes with tagged values defined

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

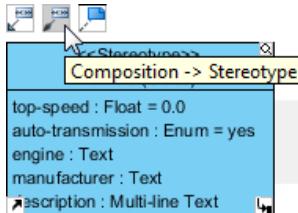
A composition relationship shows a "part of" relationship between stereotypes. The composite stereotype has responsibility for the existence and storage of the composed stereotype.



Caption Here

To create a composed stereotype:

1. Move the mouse pointer over a stereotype. Press on the resource icon **Composition > Stereotype** from the popup menu.



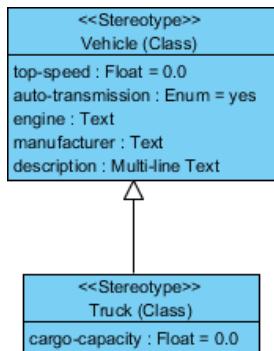
Caption Here

2. Drag it out. Release at the position you want to create the composed stereotype. Select base class, 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

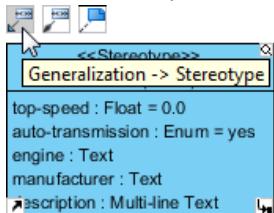
A generalization relationship shows a "kind of" relationship between stereotypes.



Caption Here

To create a specific stereotype from a general stereotype:

1. Move the mouse pointer over a stereotype. Press on the resource icon **Generalization > Stereotype** from the popup menu.



Caption Here

2. Drag it out. Release at the position you want to create the specialized stereotype. Select base class, 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*.

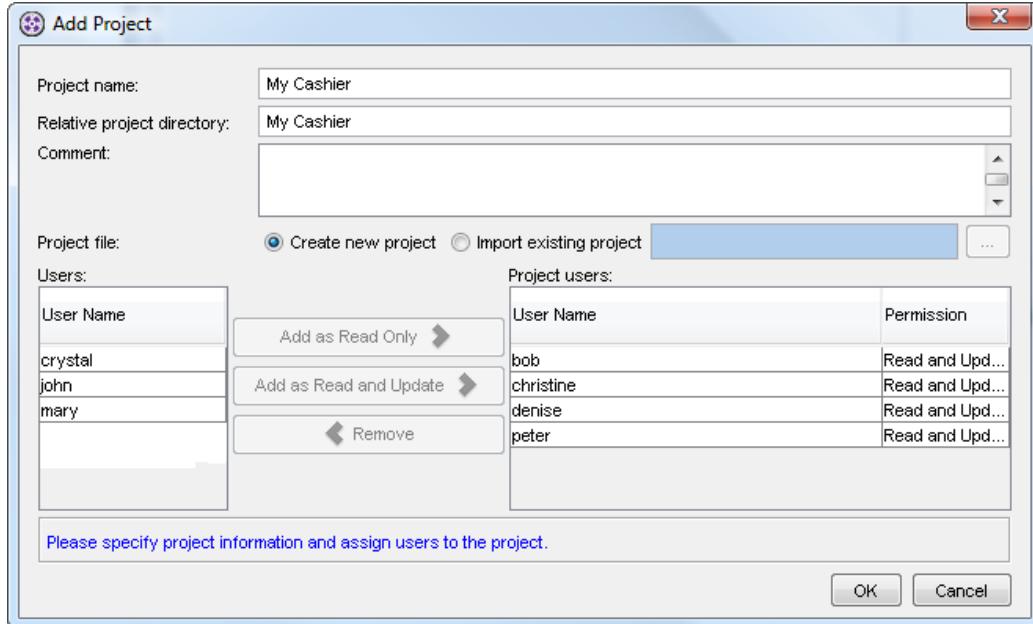
Introduction to team collaboration

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.

VP-UML's team collaboration support provides access to a central repository for managing, sharing and versioning projects. You can let team members get project from repository, start working on their own parts, let them commit (i.e. upload) their work to server, and update others' works. Visual Paradigm Teamwork Server, SVN, CVS and Perforce are the supported standards of versioning systems. In this part, we will go through team collaboration using Teamwork Server. This chapter outlines some of the key concepts in team collaboration in VP-UML.

Administration

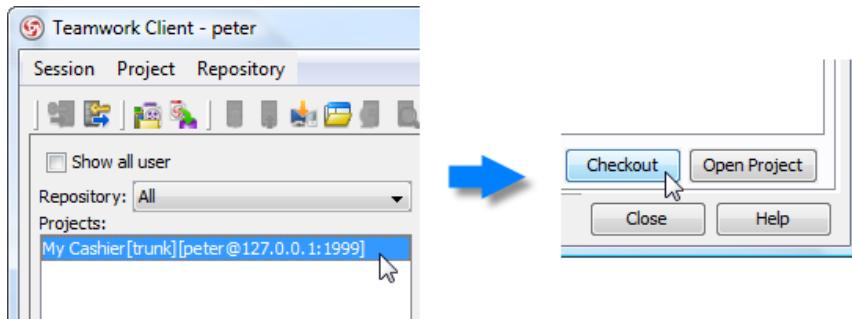
Administration is the process of setting up projects and users (i.e. team members), and deciding the access rights of members against projects. Administration is a must in order to start working.



To add a project with users assigned

Checkout project

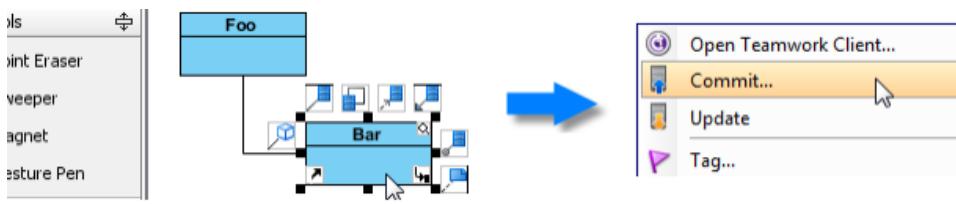
Checkout project is a process done by team members, for getting a project from repository to start working with. Team members should login into the Teamwork Server in VP-UML, then checkout the projects to work with, provided that they have the permission to do so, as granted by administrator.



A team member is checking out a project

Commit

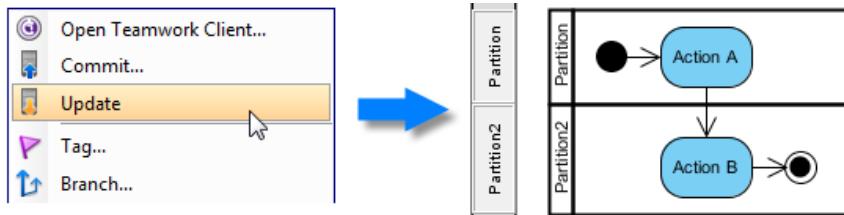
Commit is the process of uploading changes done in working copy to server. As team members make changes in a project, they can share their works by committing those changes to the server. VP-UML will try to merge changes from working copy to server copy. When merging, there may be conflict when any changes a team member made 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.



Going to commit local changes

Update

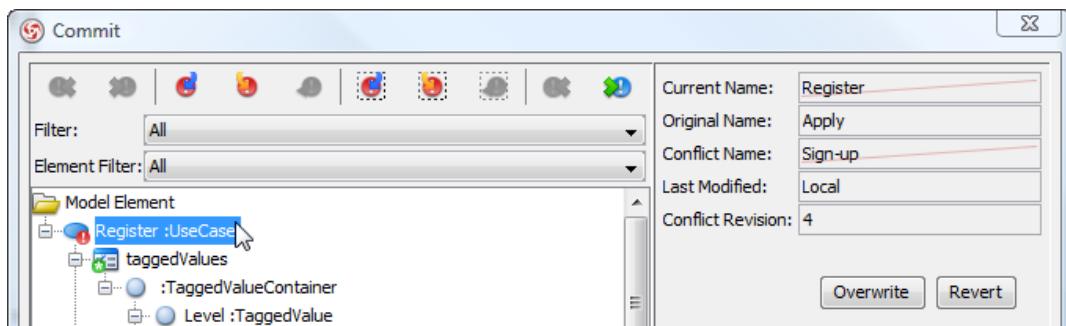
Update is the process of refreshing the working copy by merging changes that others had made and committed to server before. Similar to commit, update is a process of merging differences instead of overwriting. If your changes overlap the changes others had made, you will be asked to resolve conflict. All conflicts must be resolved in order to proceed with updating.



Update to obtain changes made by other team members

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 use case, a conflict is occurred. Whenever a conflict occurred, users have to resolve it or else to abort before commit/update operation.



A conflict is occurred

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 VP-UML, 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.

Project Details		
Revisions		
Show:		Last 10
Project revisions:		
Rev.	User	Date Time
6	denise	Jan 30, 2010, 1:...
5	peter	Jan 30, 2010, 1:...
4	peter	Jan 30, 2010, 1:...
3	peter	Jan 30, 2010, 1:...
2	peter	Jan 30, 2010, 1:...
1	Admin	Jan 30, 2010, 1:...

A list of revisions

Revert changes

Revert is the process of undoing changes. In VP-UML'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.

The screenshot shows a software interface for managing project revisions. At the top, there are tabs for 'Project Details' and 'Revisions'. The 'Revisions' tab is selected. Below the tabs, a dropdown menu says 'Show: Last 10'. A section titled 'Project revisions:' contains a table with the following data:

Rev.	User	Date Time
6	denise	Jan 30, 2010, 1:...
5	peter	Jan 30, 2010, 1:...
4	peter	Jan 30, 2010, 1:...
3	peter	Jan 30, 2010, 1:...
2	peter	Jan 30, 2010, 1:...
1	Admin	Jan 30, 2010, 1:...

At the bottom of the revision list, there are four buttons: 'Revert' (highlighted with a cursor), 'Export', 'Open', and 'Compare...'. The 'Revert' button is the primary focus of the screenshot.

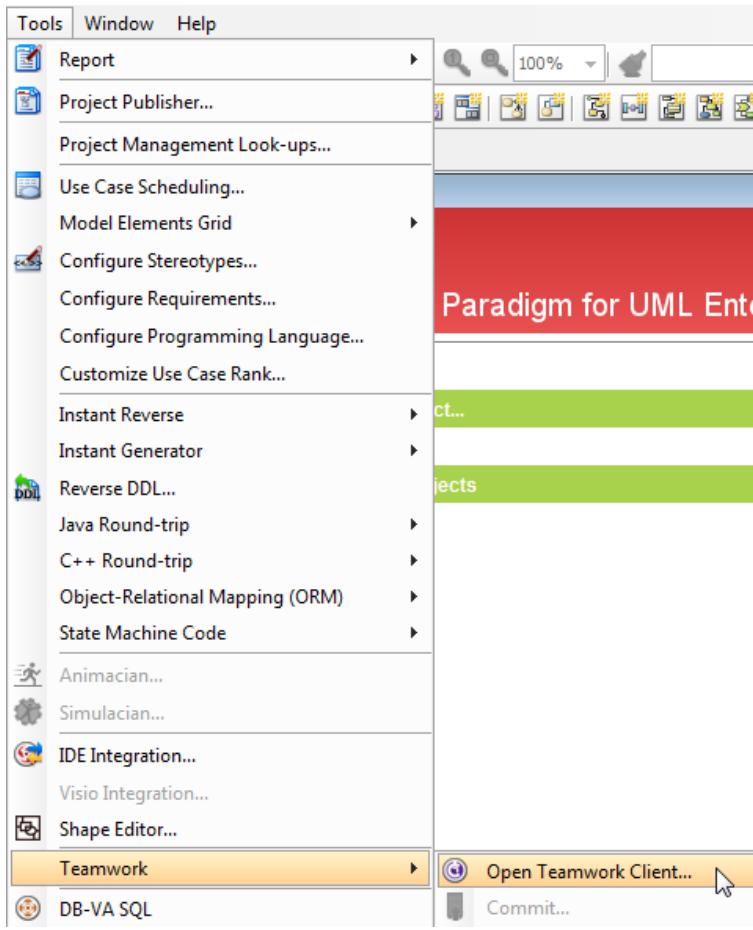
To revert two revisions

The Teamwork Client

Teamwork Client is where you can manage, checkout and open projects. It involves all teamwork activities that you can perform with.

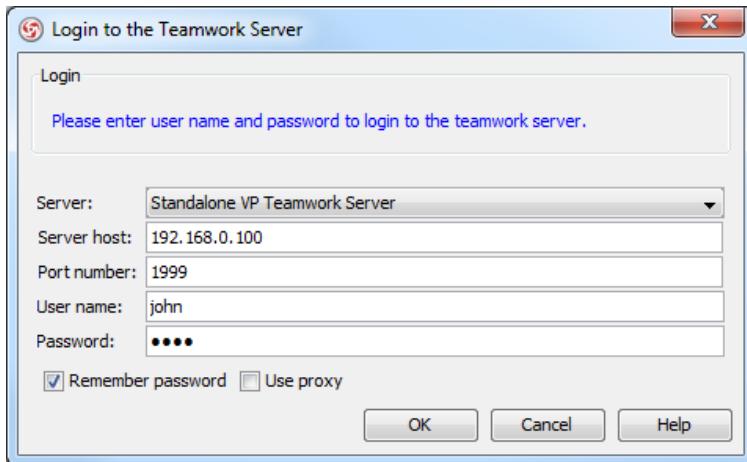
To launch teamwork client:

1. After entering VP-UML, select **Tools > Teamwork > Open Teamwork Client...** from the main menu.



Launch teamwork client from the main menu

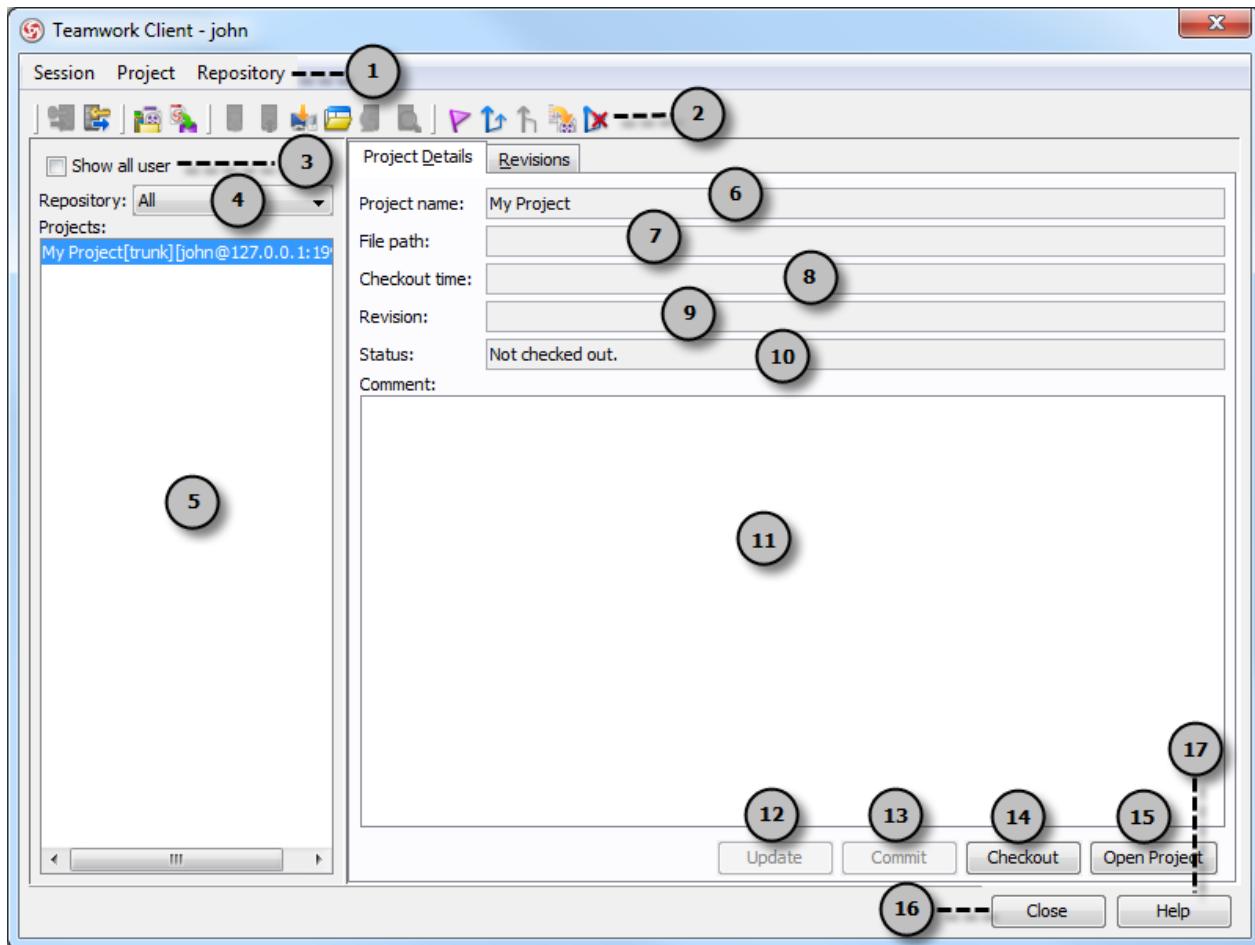
2. In the **Login to the Teamwork Server** dialog box, enter your server host, user name and password respectively. Finally, click **OK** to login to the teamwork server.



The Login to the Teamwork Server dialog box

3. In the **Manage Project** dialog box, select your project and click > to manage it. Click **OK** to proceed.

Overview of Teamwork Client dialog box



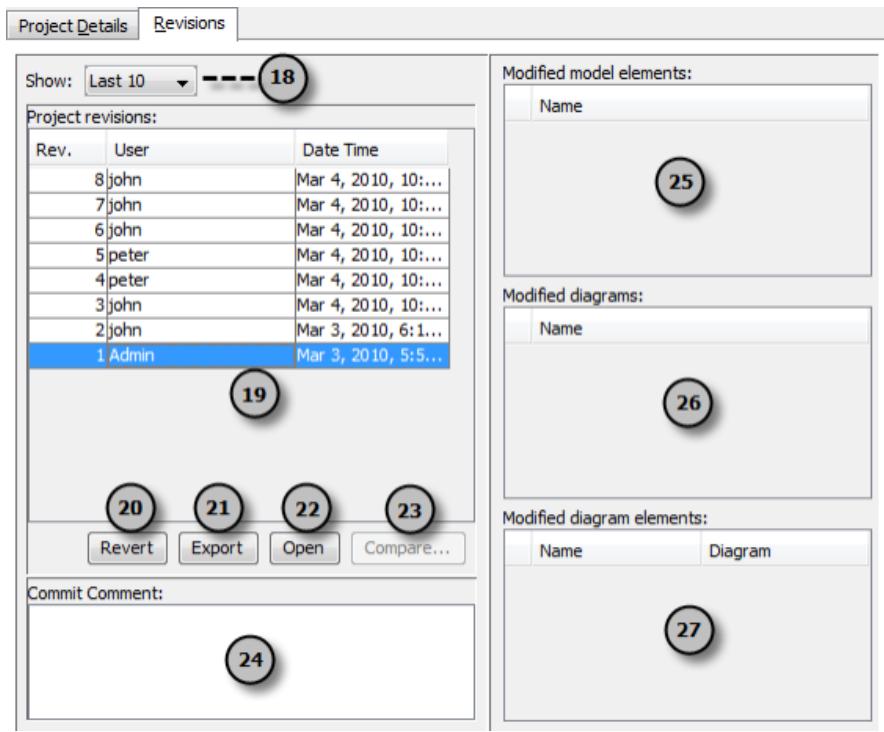
The Teamwork Client dialog box

No.	Name	Description
1	Main menu	<p>Session: It is a period of active connection to Teamwork Server.</p> <p>Login: Log into the server. After you choose it, you will be able to execute all actions.</p> <p>Logout: Log out the server. After you choose it, you will not be able to execute any actions.</p> <p>Project: It provides an access to main functions, such as commit and update.</p> <p>Manage Project: Select a project that you get involved in.</p> <p>Import Project to Repository: Import a new project to the server.</p> <p>Commit...: Commit your current modified project to the server.</p> <p>Update: Update the latest copy of project from the server to your computer.</p> <p>Checkout: Click it to checkout the project selected in Projects list. It will be disabled when the selected project has already been checked out.</p> <p>Open: Click it to open the checkout project on your computer.</p> <p>Check Update: Click it to check whether the project is up-to-date or not.</p> <p>Tag...: Create a new tag for your current project. It allows you to produce a static release version of project.</p> <p>Branch...: Create a new branch for your current project. It becomes a duplication of project to perform isolated changes.</p> <p>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.</p> <p>Switch...: Switch from a branch/ tag to another branch/ tag or from the trunk (main project) to a branch/ tag and vice versa.</p> <p>Delete Branch...: Select a branch to delete, for preventing accidental modifications in branch.</p> <p>Reset Password: Reset your account's password.</p> <p>Revert...: Click it to undo un-committed changes made on the local project copy. If no changes has been made, the project will go back to the original state.</p> <p>Repair > Force Commit Local Copy: Normally, a commit action is to merge your current changes to server. By checking it, your current modified project will replace the project copy in server. Do NOT check this option unless you are asked to do so by Visual Paradigm's support team.</p> <p>Repair > Force GC: Normally, all redundant files produced by teamwork commit action will be removed after commit. This will be done periodically. By checking it, it will be done in every commit.</p>

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.

2 Toolbar	<p>Login: Log into the server. After you choose it, you will be able to execute all actions.</p> <p>Logout: Log out the server. After you choose it, you will not be able to execute any actions.</p> <p>Manage Project: Select a project that you get involved in.</p> <p>Import Project to Repository: Import a new project to the server on the list.</p> <p>Update: Update the latest copy of project from the server to your computer.</p> <p>Commit: Commit your current modified project to the server.</p> <p>Checkout: Click it to checkout the project selected in Projects list. It will be disabled when the selected project has already been checked out.</p> <p>Open: Click it to open the checkout project on your computer.</p> <p>Revert...: Click it to undo un-committed changes made on the local project copy. If no changes has been made, the project will go back to the original state.</p> <p>Check for Update: Click it to check whether the project is up-to-date or not.</p> <p>Tag...: Create a new tag for your current project. It allows you to produce a static release version of project.</p> <p>Branch...: Create a new branch for your current project. It becomes a duplication of project to perform isolated changes.</p> <p>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.</p> <p>Switch...: Switch from a branch/ tag to another branch/ tag or from the trunk (main project) to a branch/ tag and vice versa.</p> <p>Delete Branch...: Select a branch to delete, for preventing accidental modifications in branch.</p>
3 Show all user	By checking it, the list of all eligible users who have logged into the server in this workspace will be displayed. On the contrary, by unchecking it, only the current user 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 users who have logged into Teamwork Server in this workspace will be listed. On the other hand, the project(s) managed by a specific user can be selected from the drop-down menu. If you uncheck Show all user and do not select the current user 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** dialog box*



The **Revisions** tab of the **Teamwork server** dialog box

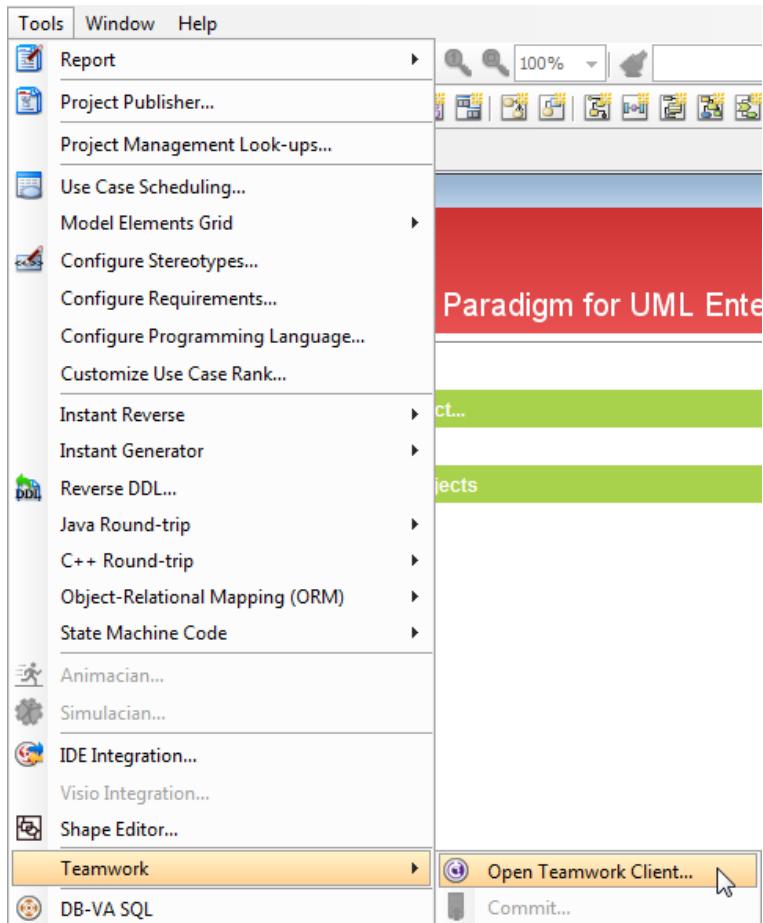
No.	Name	Description
18	Show drop-down menu	Select the number of latest project revision to view from the drop-down menu.
19	Project revisions	It lists all the latest project revisions. The number of revisions is in accordance with the show drop-down menu.
20	Revert	Click it to undo changes committed by the selected revisions.
21	Export	Export selected revisions... : Select a directory for exporting the selected revision of project. Export all revisions from repository... : Select a directory for exporting all revisions of project from repository.
22	Open	Click it to open the selected revision of project.
23	Compare...	After you have selected two revisions, click it to compare the differences between them.
24	Commit Comment	A textual description of commit given by you or your teammates before committing.
25	Modified model elements	It displays the modified model elements of the selected revision of project.
26	Modified diagrams	It displays the modified diagrams of the selected revision of project.
27	Modified diagram elements	It displays the modified diagram elements of the selected revision of project.

The description of **Revisions** tab

Checkout project

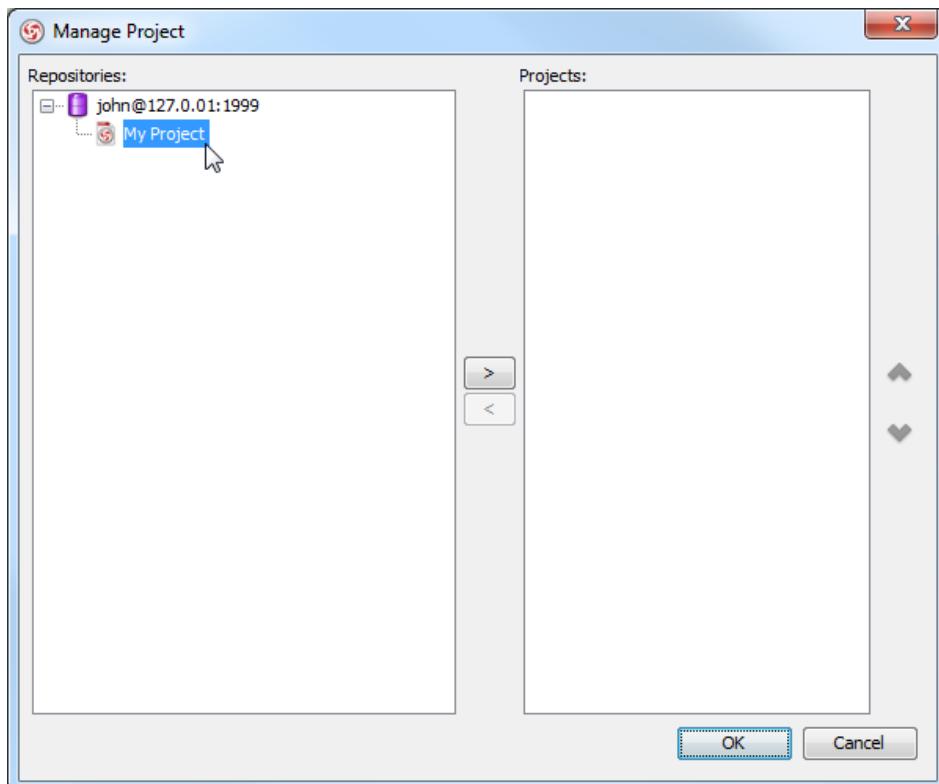
Checkout project is a process that you download a managed project from repository to your computer and start working with. Team members should log into the Teamwork Server in VP-UML, then checkout their completed projects. Open project is to open the downloaded project in VP-UML.

After entering VP-UML, select **Tools > Teamwork > Open Teamwork Client...** from the main menu.



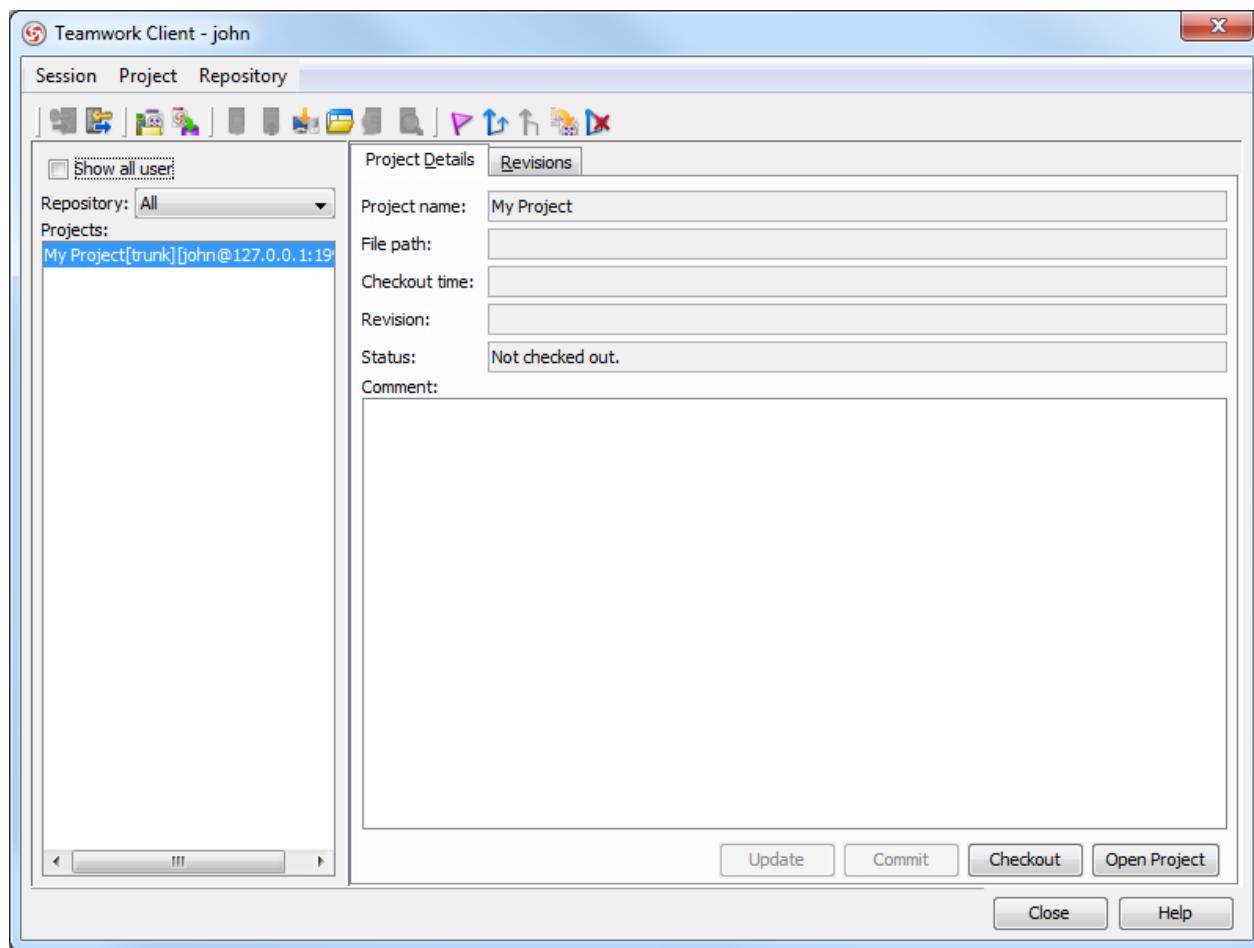
Click **Open Teamwork Client...** from the main menu

In the **Manage Project** dialog box, select the project you are going to take part in and click **>**. Click **OK** to proceed.



Select a project

In the **Teamwork Client** dialog box, select the project, and click **Checkout**. After the selected project has been checked out, click **Open Project**.



The **Teamwork Client** dialog box

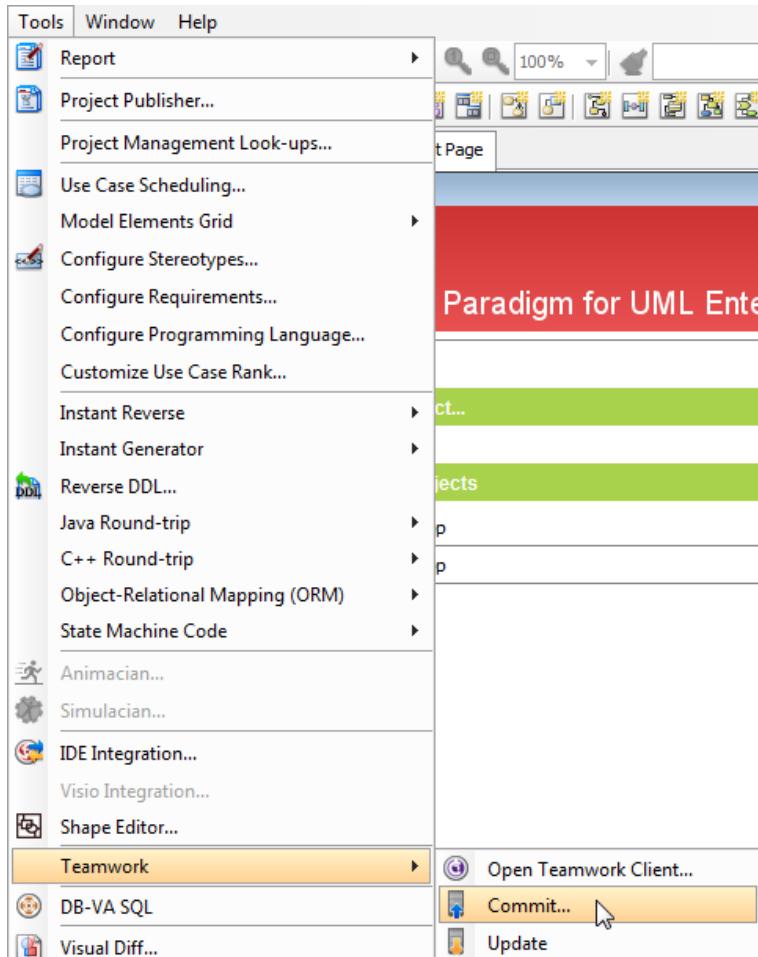
After that, the project has been downloaded from the server to your computer and you can start working with it.

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. VP-UML will help you to merge changes from working copy to server copy. During merging, a conflict may be caused when there is a contradiction between any changes a team member made 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 have to be solved before proceeding to commit.

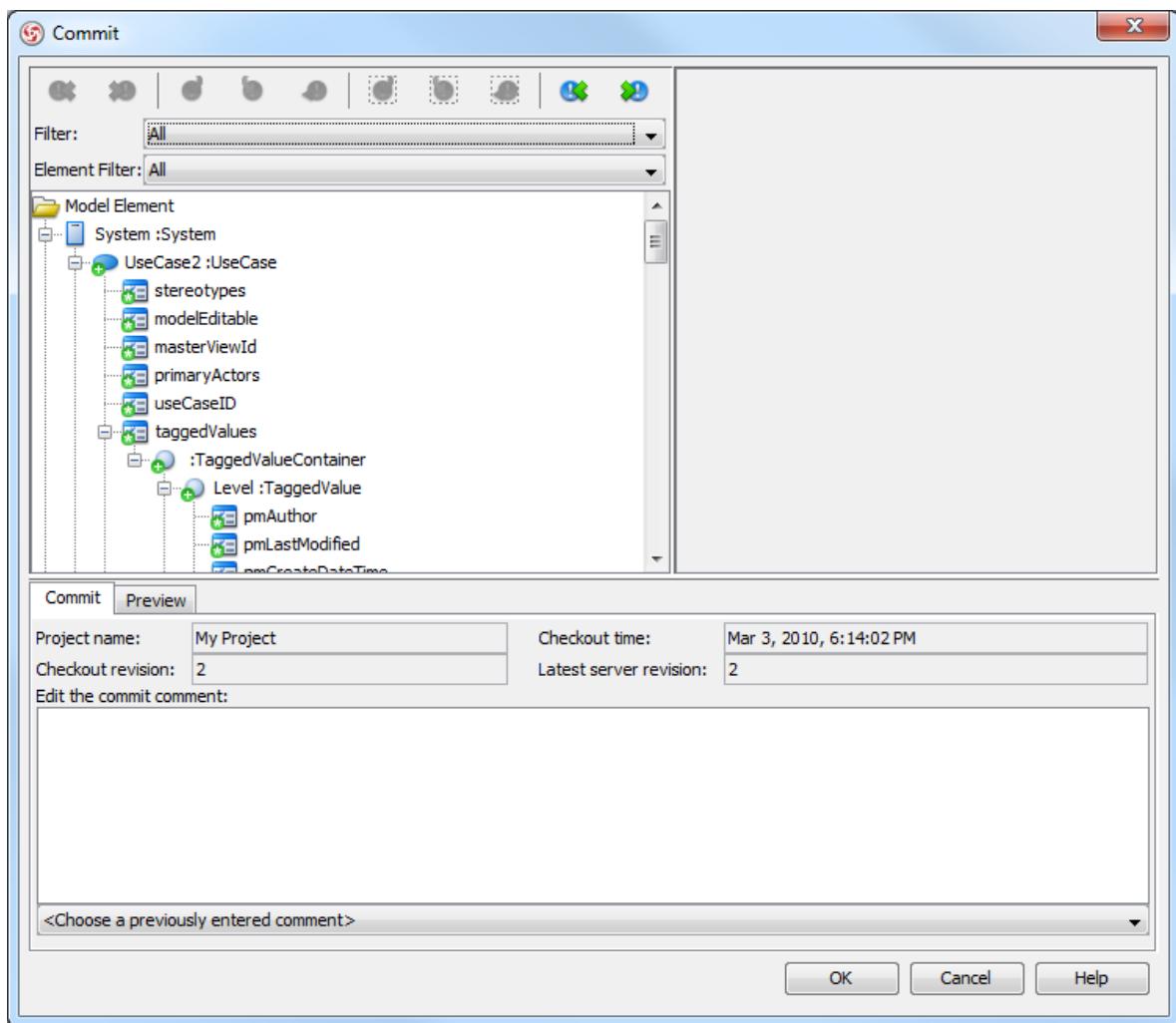
To operate commit:

Select **Tools > Teamwork > Commit...** from the main menu to commit your modified copy to the server.



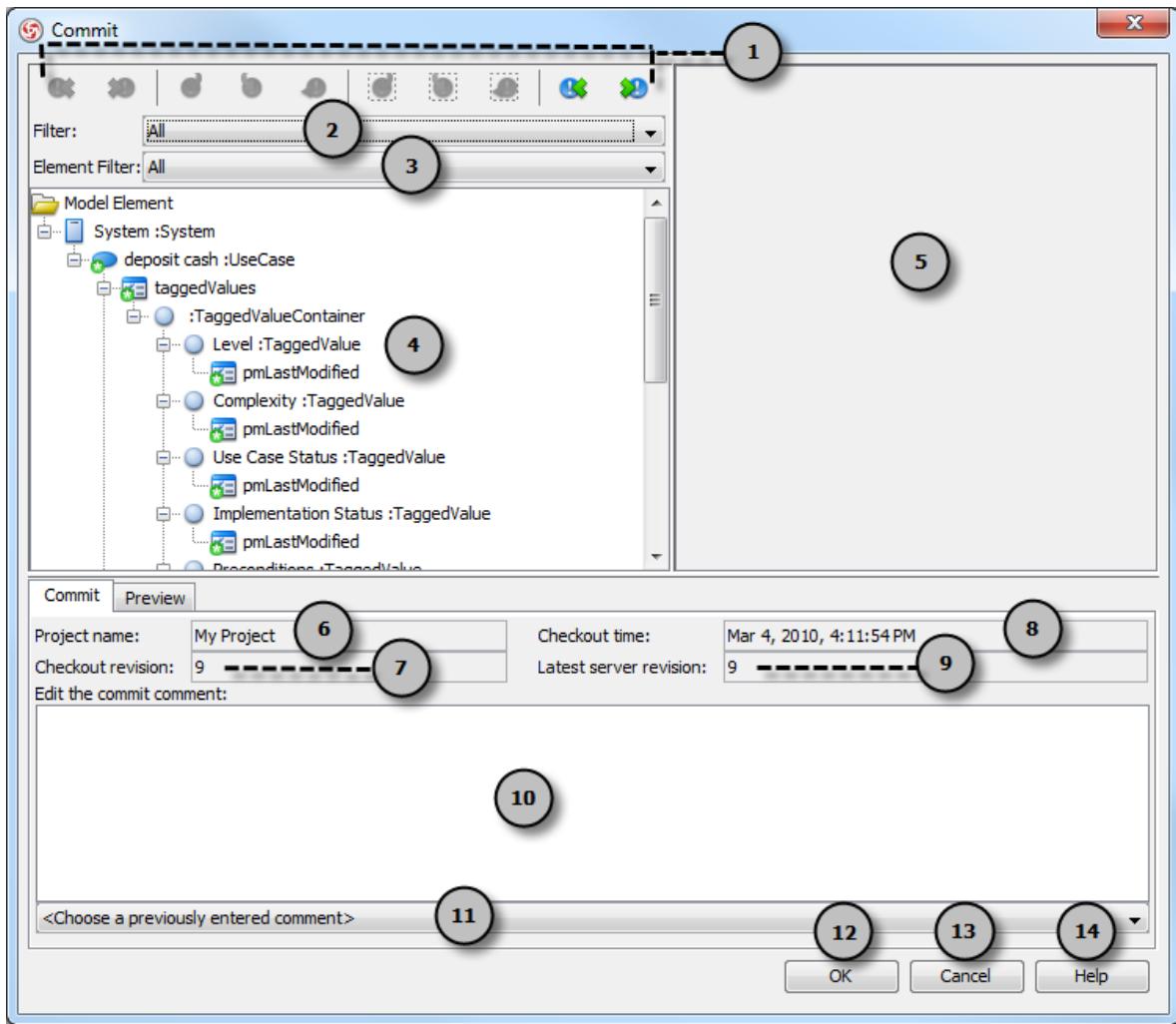
Launch commit from the main menu

The **Commit** dialog box displays the changes to be committed to the server. Click **OK** to proceed.



The **Commit** dialog box

Overview of Commit dialog box

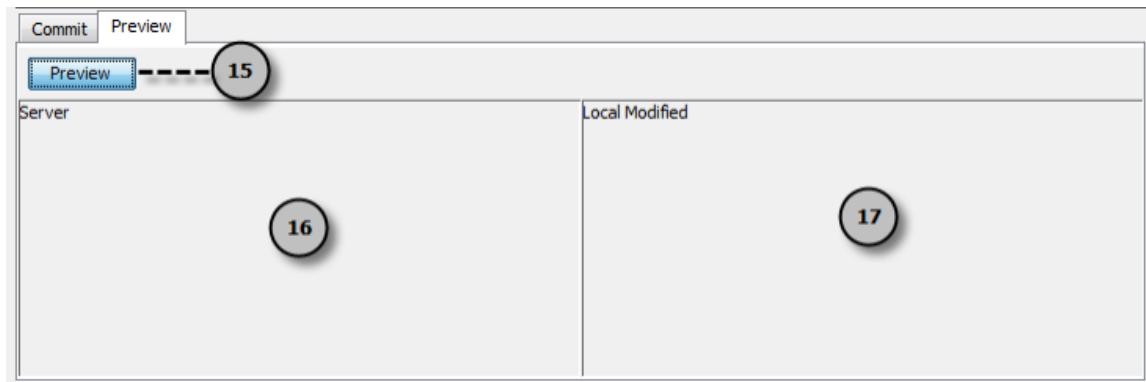


The **Commit** dialog box

No.	Name	Description
1	toolbar	<p>Select previous conflict: Select the previous conflict you made in current copy.</p> <p>Select next conflict: Select the next conflict you made in current copy when a conflict is found during committing.</p> <p>Overwrite all conflicts: Keep all modification you made in current copy when a conflict is found during committing.</p> <p>Revert all conflicts: Accept all modification other team members made previously and discard all your modification when a conflict is found during committing.</p> <p>Reset all conflicts: Cancel the decision you have just made for dealing with all conflicts when a conflict is found during committing.</p> <p>Overwrite selected conflicts: Keep the selected modification you made in current copy when a conflict is found during committing.</p> <p>Revert selected conflicts: Accept the selected modification other team members made in current copy when a conflict is found during committing.</p> <p>Reset selected conflicts: Cancel the decision you have just made for dealing with the select conflict(s) when a conflict is found during committing.</p> <p>Select previous change: Select the previous modification you have made in the sequence of list displaying in Display window.</p> <p>Select next change: Select the next modification you have made in the sequence of list displaying in Display window.</p>
2	Filter	Select a specific scope for previewing the to-be-committed project, including: All , Update from Server , Commit , Conflict , Created Elements , Modified Element and Deleted Elements .
3	Element Filter	Select a specific scope for previewing the to-be-committed model elements, including: All , TaggedValue , TaggedValueContainer and UseCase .
4	Display window	It displays a specific scope for previewing the to-be-committed project/ model elements in accordance with the selection of Filter/ Element Filter .

5 Property view window	It lists property name, current value, original value and last modified of the selected model element.
6 Project name	It shows the name of current project.
7 Checkout revision	It shows the number of current checkout revision.
8 Checkout time	It shows the time for latest checkout.
9 Latest server revision	It shows the number of latest revision in the server.
10 Edit the commit comment	You can give a comment for your current commit by typing here.
11 Choose a previously entered comment	You can search and select a comment that you entered previously in this drop-down menu.
12 OK	Click OK to proceed committing.
13 Cancel	Click Cancel to cancel committing and close the dialog box.
14 Help	Click Help to get assistance from help system.

*The description of **Commit** dialog box*



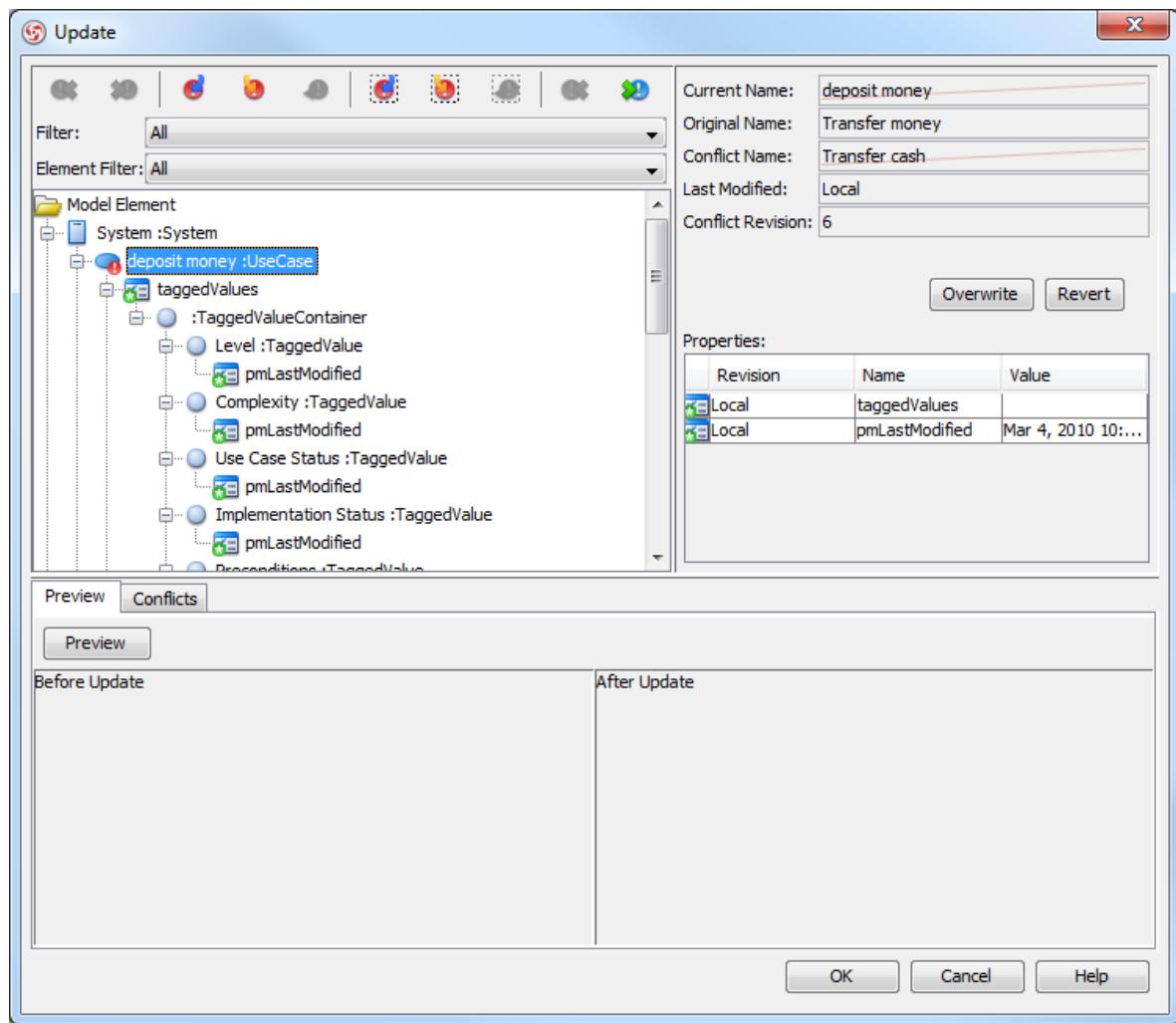
*The **Preview** tab of the **Commit** dialog box*

No.	Name	Description
15	Preview	Click it for previewing the selected model element(s).
16	Server preview window	It displays the original revision in server for previewing.
17	Local modified preview window	It displays the modified and to-be-committed revision for previewing.

*The description of **Preview** tab in the **Commit** dialog box*

Resolving conflict

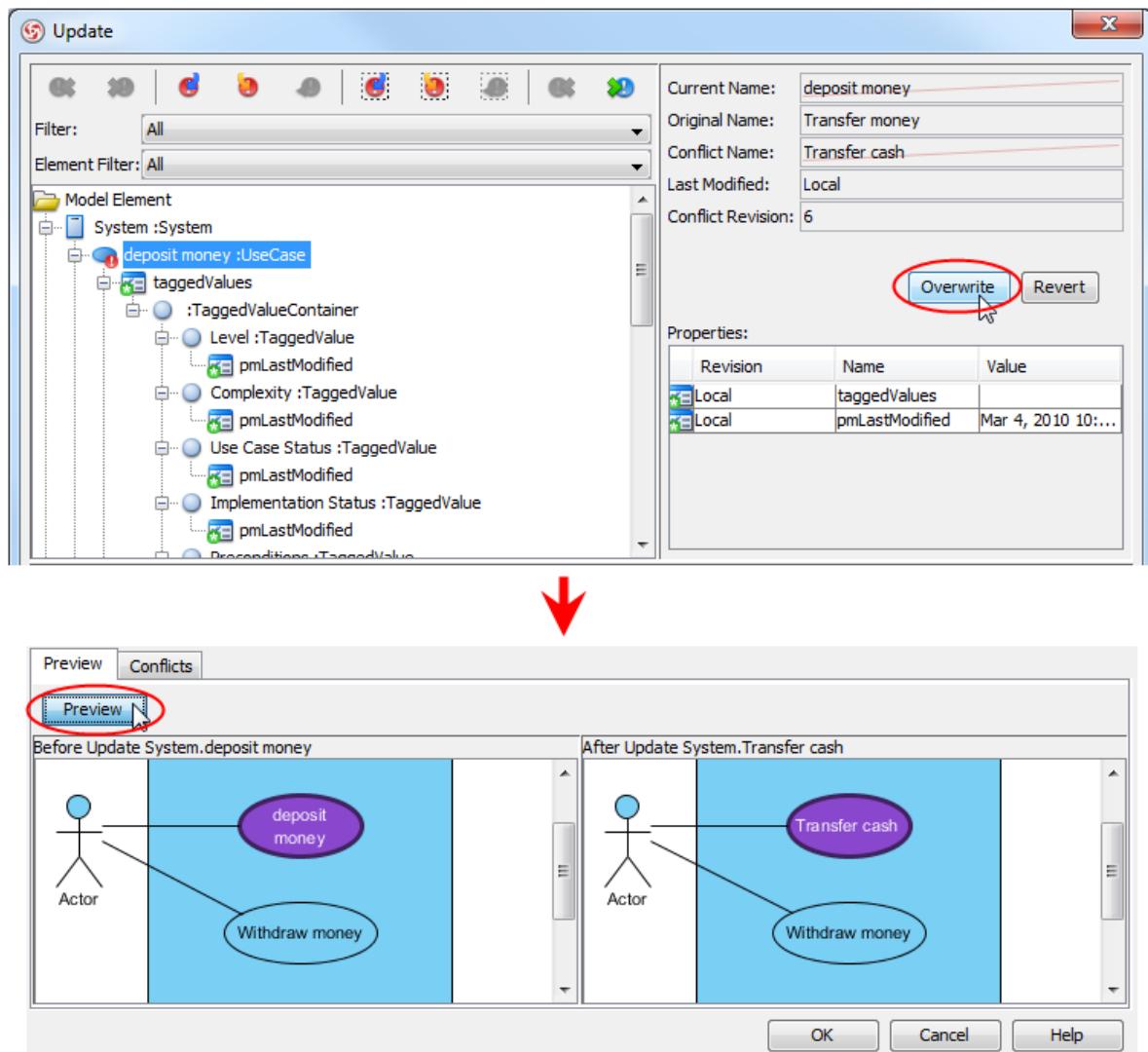
When your modified copy conflicts with the copy others made previously during committing, you can resolve the conflict by clicking either **Overwrite** or **Revert** in the **Update** dialog box.



Resolve conflict in the **Update** dialog box

Choosing **Overwrite** means keeping your modified copy while choosing **Revert** refers to accept others' copy.

You can preview the choice you made by clicking **Preview** button. The left window shows the image before update and the right window shows the image after update. Click **OK** when you want to confirm the modification.



Preview before committing

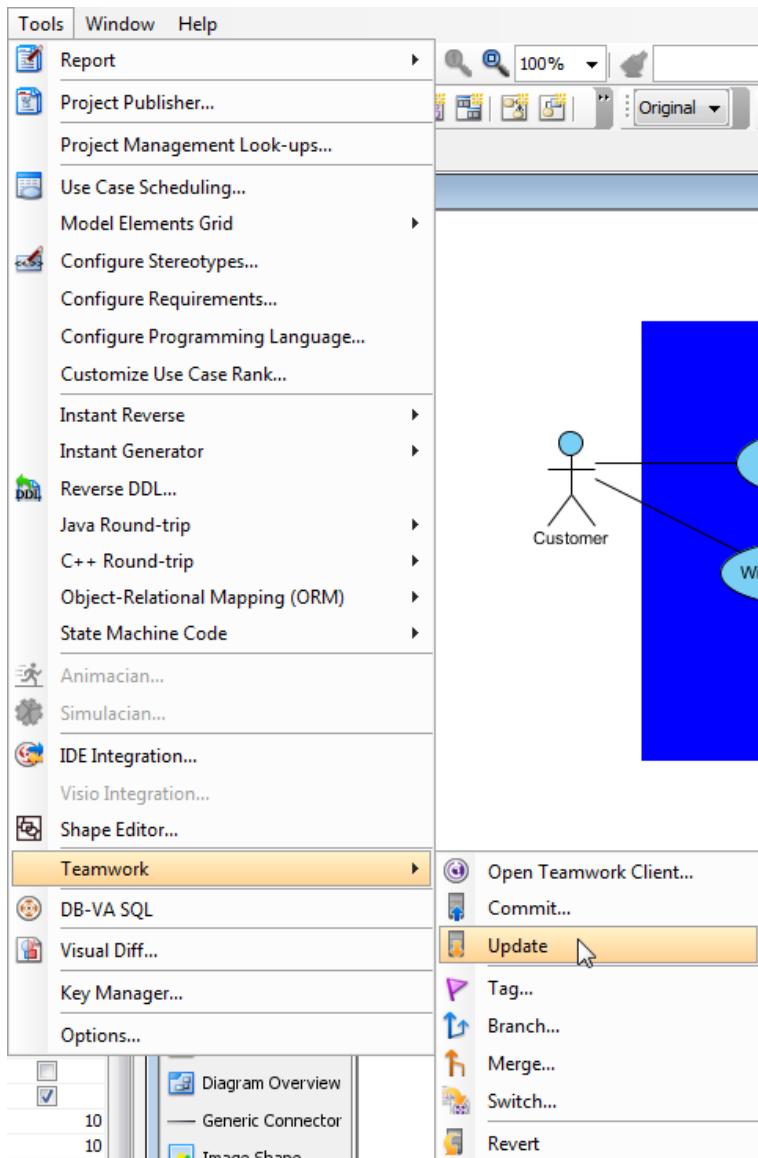
NOTE: Conflicts occur only when the same property of same shape you and other team members have made.

Update

Update is the process of refreshing your current modified 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 your changes overlap the changes others have made, you will be asked to resolve conflict. All conflicts have to be resolved before proceeding with updating.

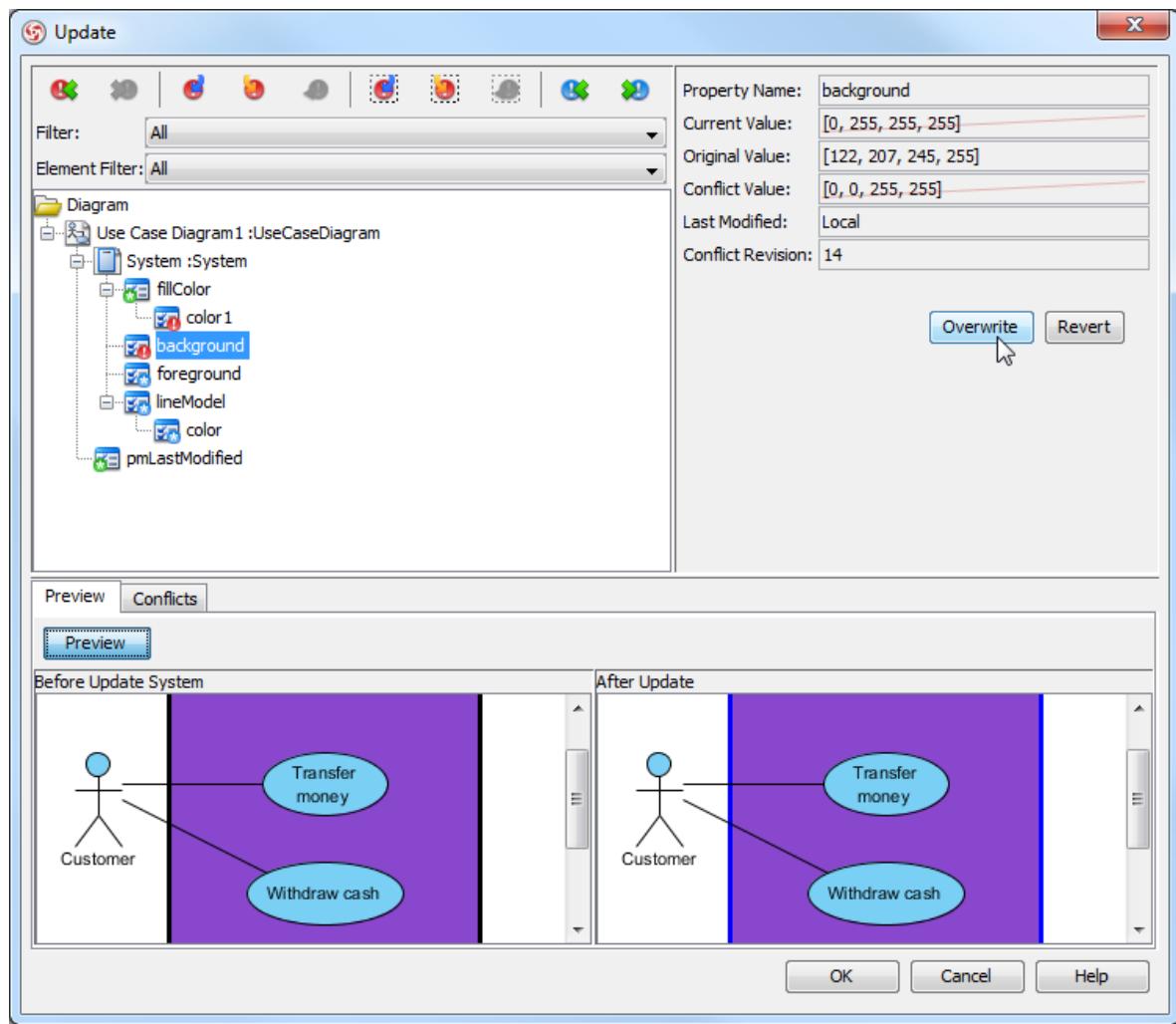
To operate update:

Select **Tools > Teamwork > Update** from the main menu to update from the server.



Select **Update** from the main menu

If other team members have made changes and have committed previously to the server, your current project will occur conflicts during updating. You can resolve the conflict by clicking either **Overwrite** or **Revert** in the **Update** dialog box. Choosing **Overwrite** means keeping your modified copy while choosing **Revert** refers to accept others' copy.



Resolve the conflicts

You can preview the choice you made by clicking **Preview** button. The left window shows the image before update and the right window shows the image after update. Click **OK** when you want to confirm the modification.

NOTE: Conflicts occur only when the same property of same shape you and other team members have made.

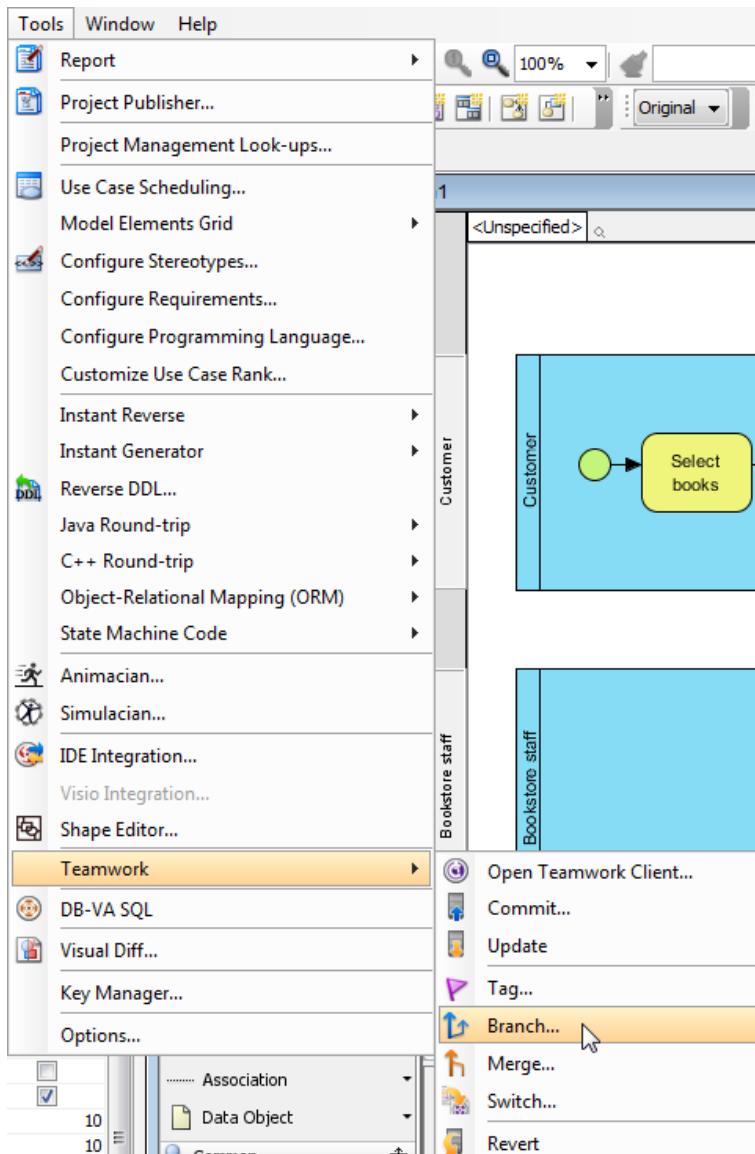
Branching

In teamwork collaboration, the term trunk refers to your main source code tree. 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 code derived from a certain point in the trunk. It sets up a space for users to work and make modification without disturbing the main line of development. As soon as the modification in branch is done, it can be merged back into the trunk.

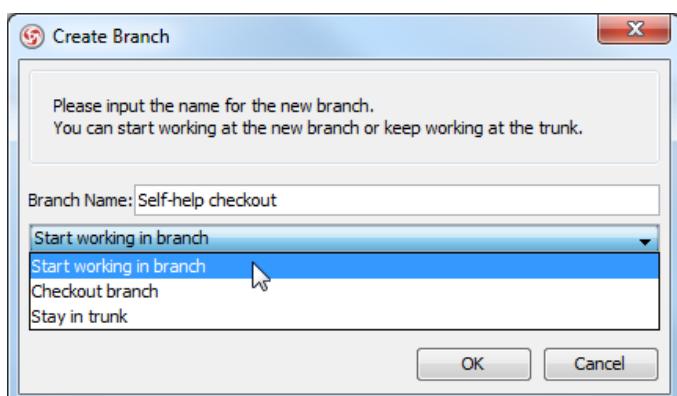
To create a branch:

Select **Tools > Teamwork > Branch...** from the main menu.



Select **Branch...** from the main menu

In **Create Branch** dialog box, enter name in **Branch Name** for your new branch and select an option from the drop-down menu. Finally, click **OK** to confirm.

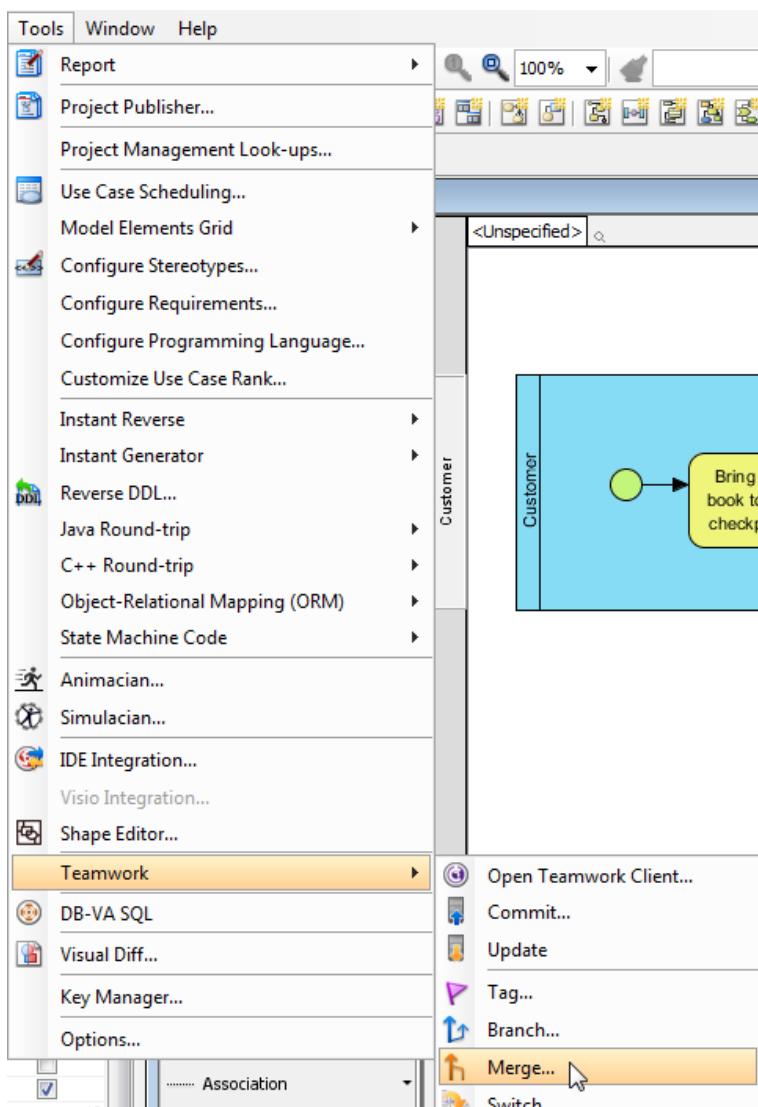


Create Branch dialog box

The three options in the drop-down menu are **Start working in branch**, **Checkout branch** and **Stay in trunk**. Selecting **Start working in branch** means you create, checkout and open a new branch and then start working on it. Selecting **Checkout branch** means you create and checkout a new branch, but just keep it in repository for working on it later on. Selecting **Stay in trunk** mean you create a new branch, but do not checkout it out and do not show it in repository either.

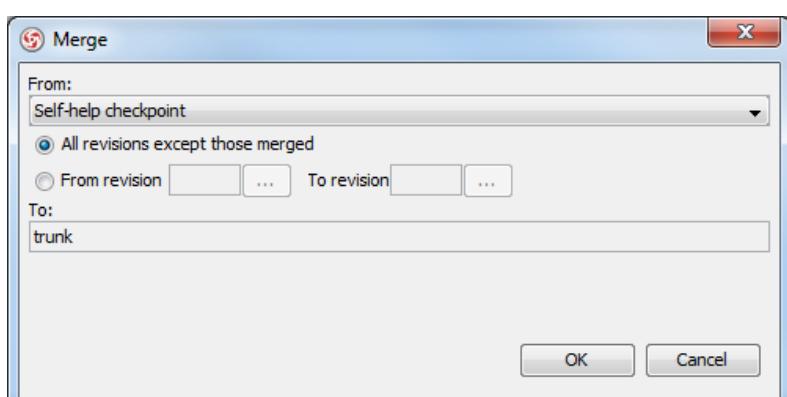
After you have finished creating a branch and have committed it, you can merge it back into the trunk.

Select **Tools > Teamwork > Merge...** from the main menu.



Select **Merge...** from the main menu

In **Merge** dialog box, select a branch you want to merge from. The specified revision of project can be merged by checking **From revision To revision** and selecting ... button. Click **OK** to confirm selecting.



Merge dialog box

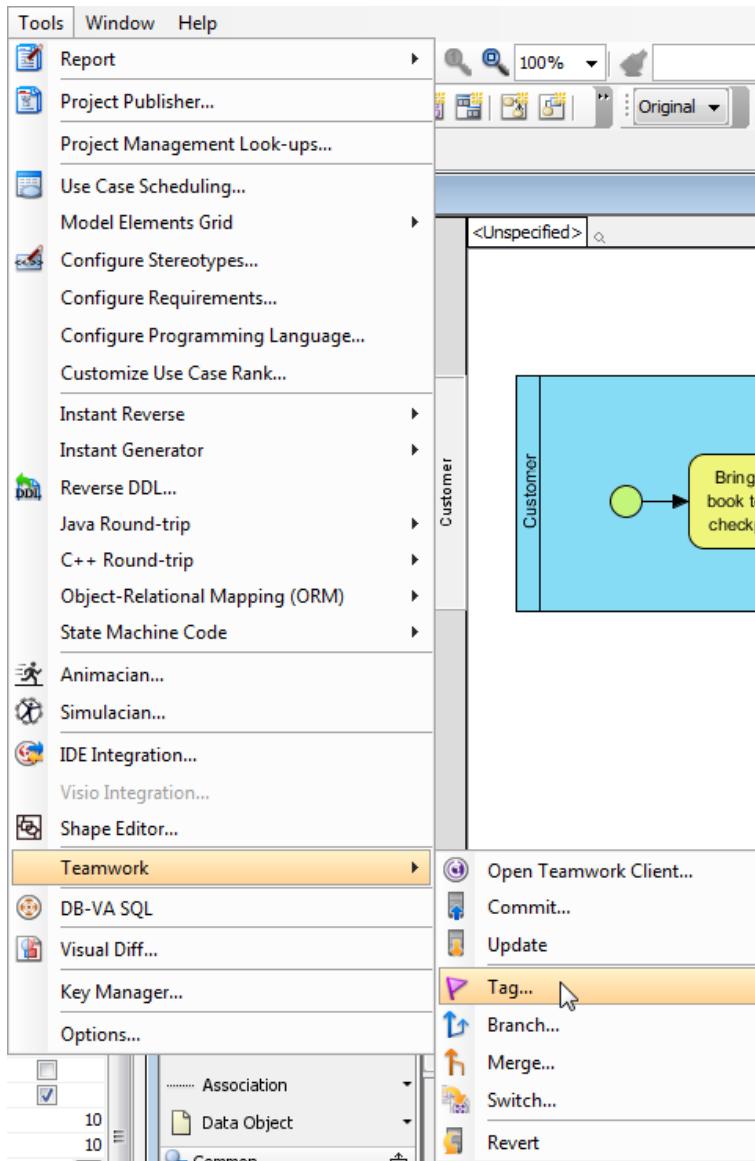
When the branch is merged to the trunk successfully, a **Message** dialog box will pop out. Press **OK** in **Message** dialog box.

Tagging

Tag refers to a named version of your code 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.

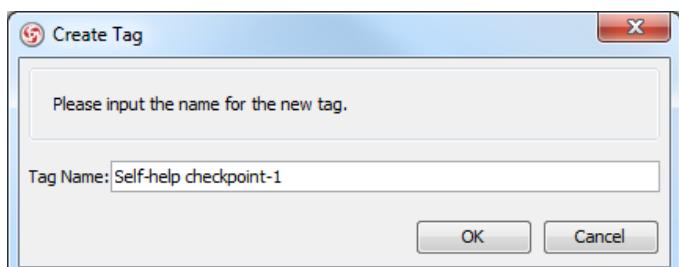
To create a tag:

Select **Tools > Teamwork > Tag...** from the main menu.



Select **Tag...** from the main menu

In **Create Tag** dialog box, enter the name for a new tag in **Tag Name**. Click **OK** to proceed.



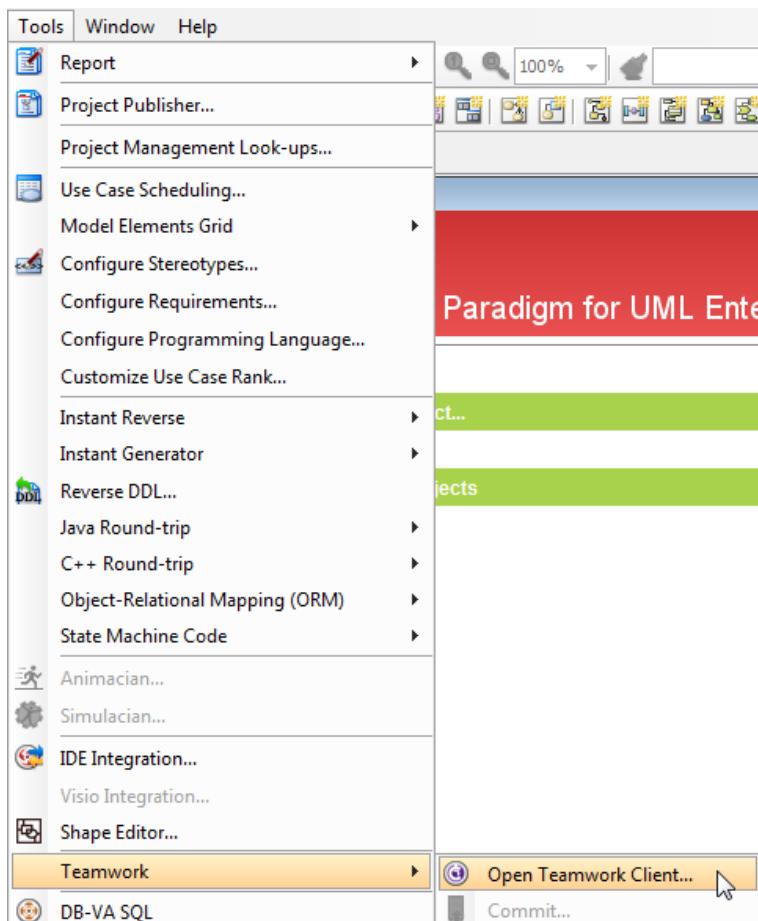
Create Tag dialog box

When a tag is created successfully, a **Message** dialog box will pop out. Press **OK** in **Message** dialog box.

Revert (Roll back) changes

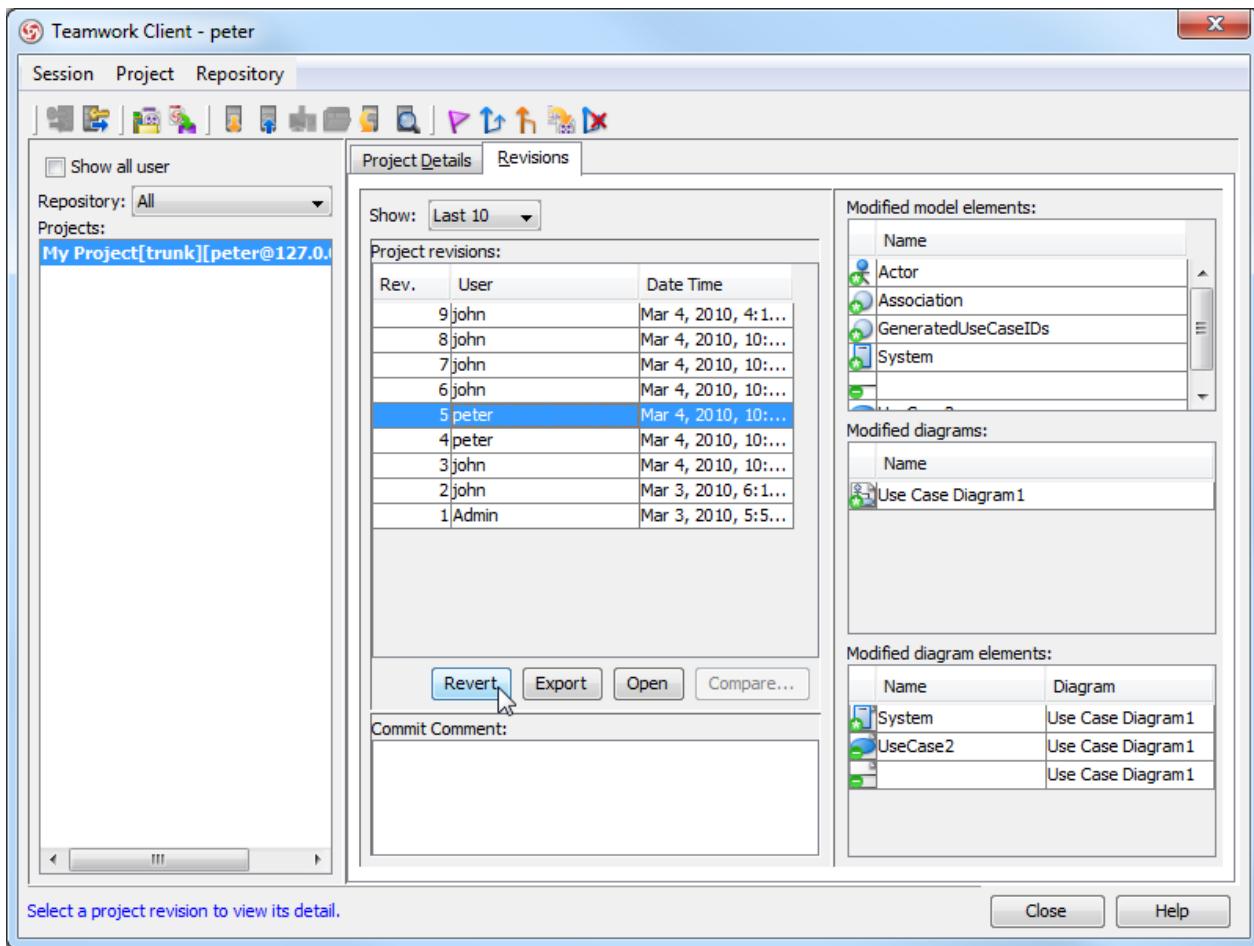
VP teamwork collaboration keeps the histories of changes of diagrams and model elements. With VP Teamwork Server, you can revert specific diagram and model element and preview the differences between the server and local modified system visually.

After entering VP-UML, select **Tools > Teamwork > Open Teamwork Client...** from the main menu.



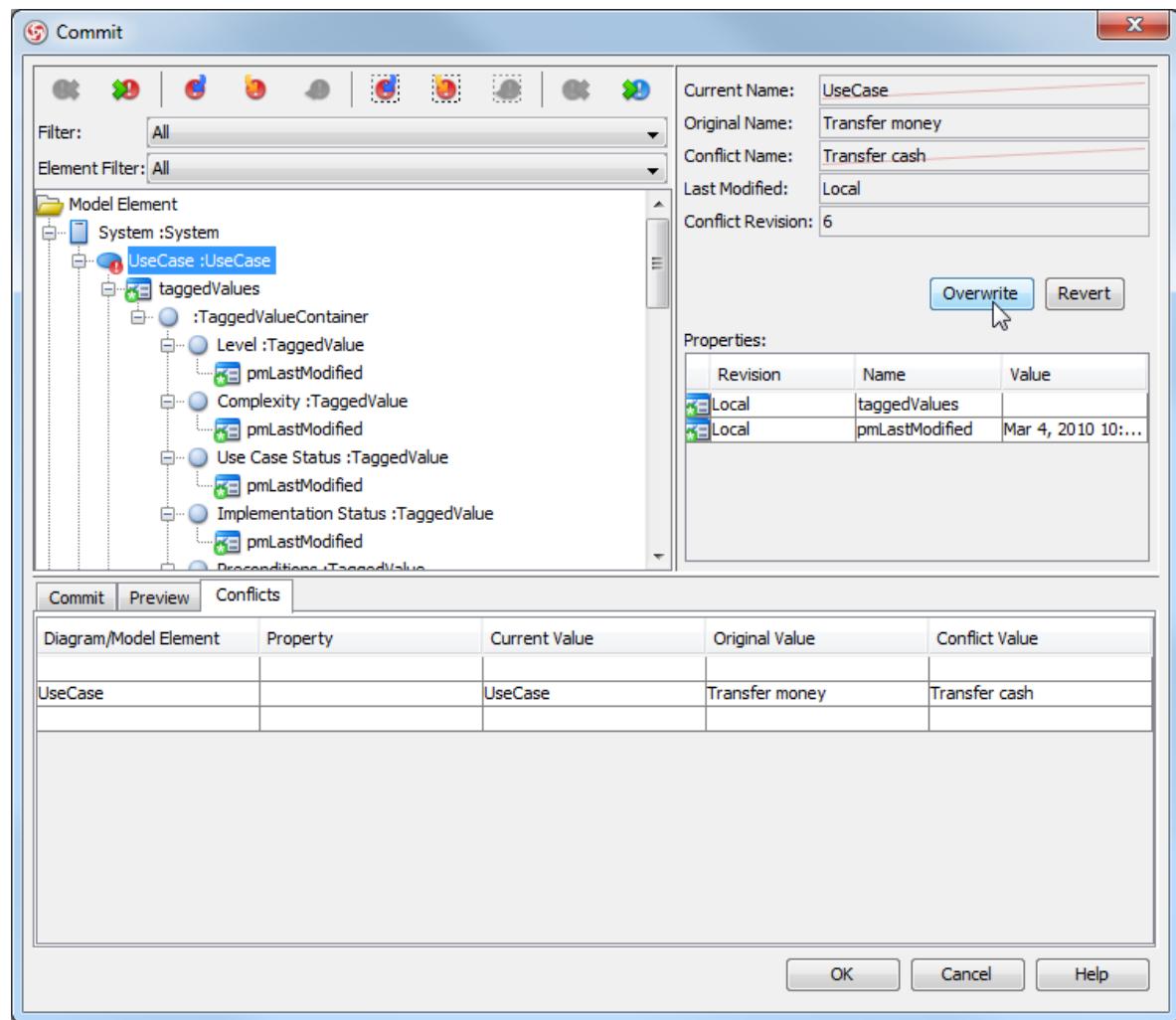
*Click **Open Teamwork Client...** from the pop-up menu*

In the **Teamwork Client** dialog box, select **Revisions** tab. Select the specified revision under **Project revisions** and click **Revert** button.



Teamwork Client dialog box

In **Commit** dialog box, you should resolve all conflicts before proceed reverting. Unfold **Preview** tab and preview the difference between the server (the latest revision in server) and the local modified (the selected revision). Click **OK** to proceed reverting.



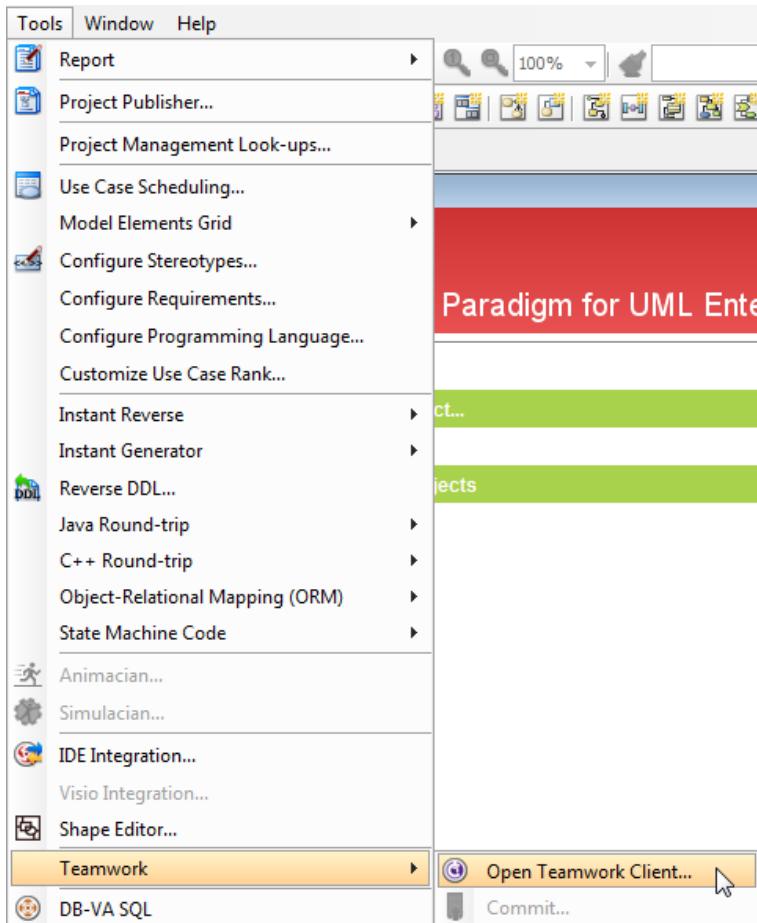
Resolving conflicts in **Commit** dialog box

Export revision

VP teamwork collaboration keeps the histories of changes of diagrams and model elements. With VP teamwork server, you can export the selected revision or all revisions to your computer for your references.

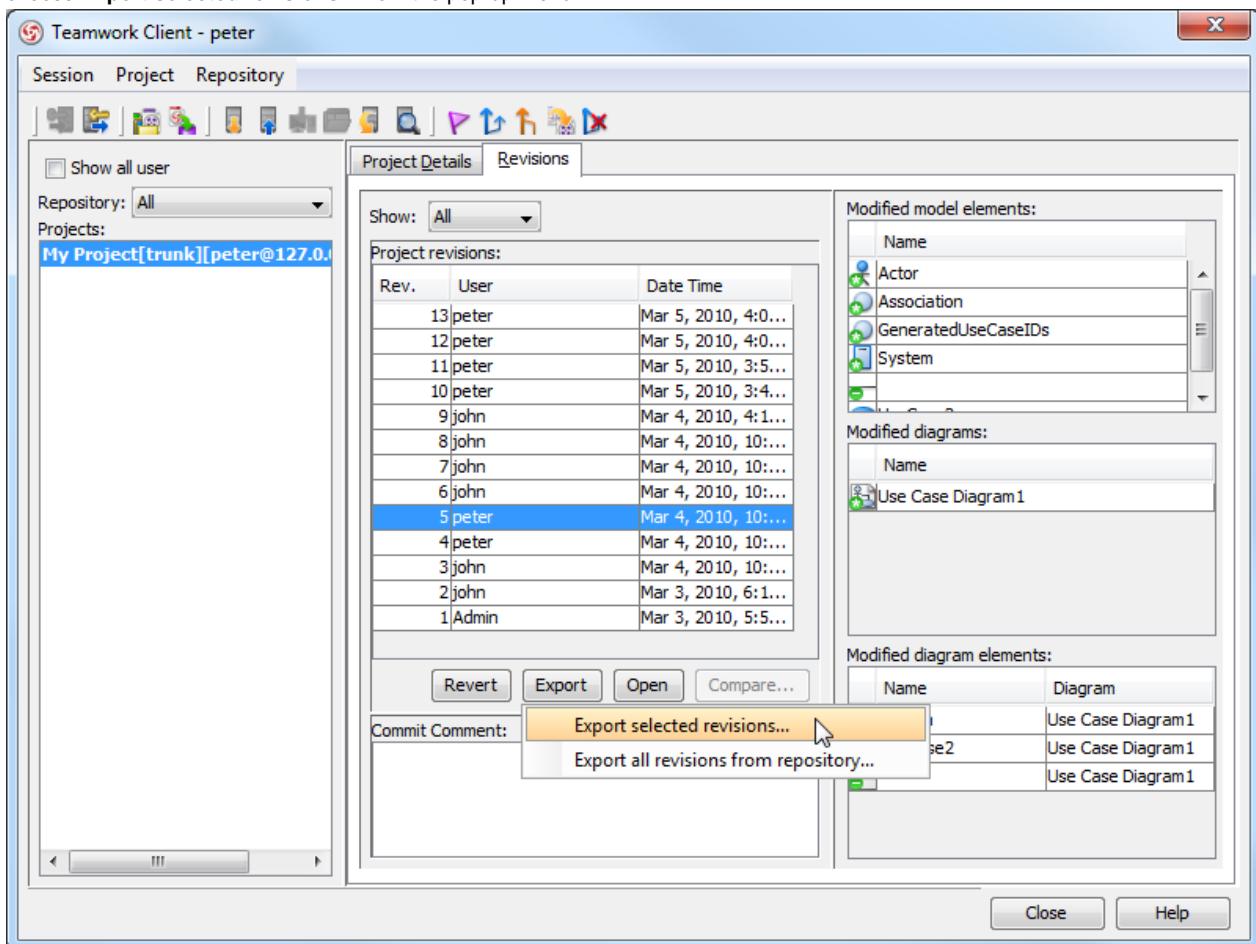
Export the selected revision

1. After entering VP-UML, select **Tools > Teamwork > Open Teamwork Client...** from the main menu.



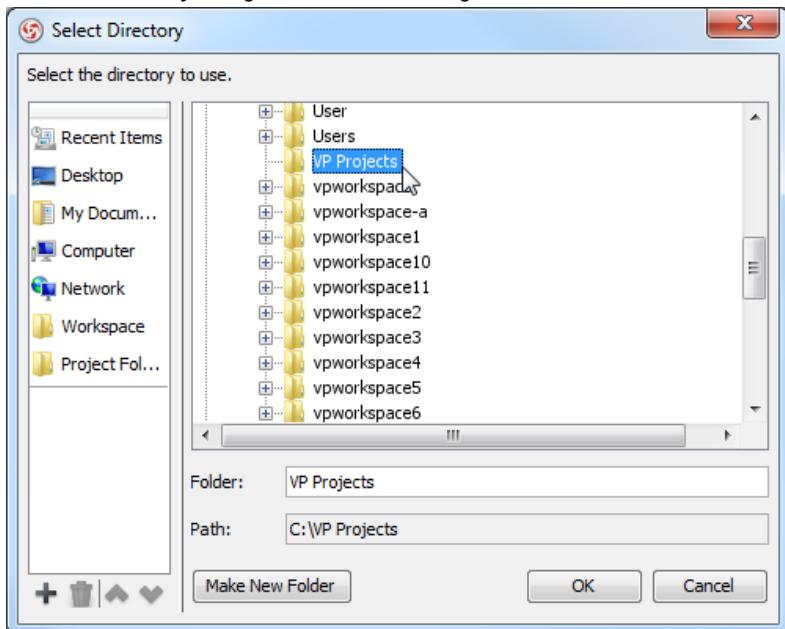
Select **Open Teamwork Client...** from the main menu

2. In the **Teamwork Client** dialog box, unfold **Revisions** tab. Select the specified revision under **Project revisions**, click **Export** button and choose **Export selected revisions...** from the pop-up menu.



*Choosing **Export selected revisions...** in Teamwork Client dialog box*

3. In **Select Directory** dialog box, select an existing folder or make a new folder for storing the selected revision.

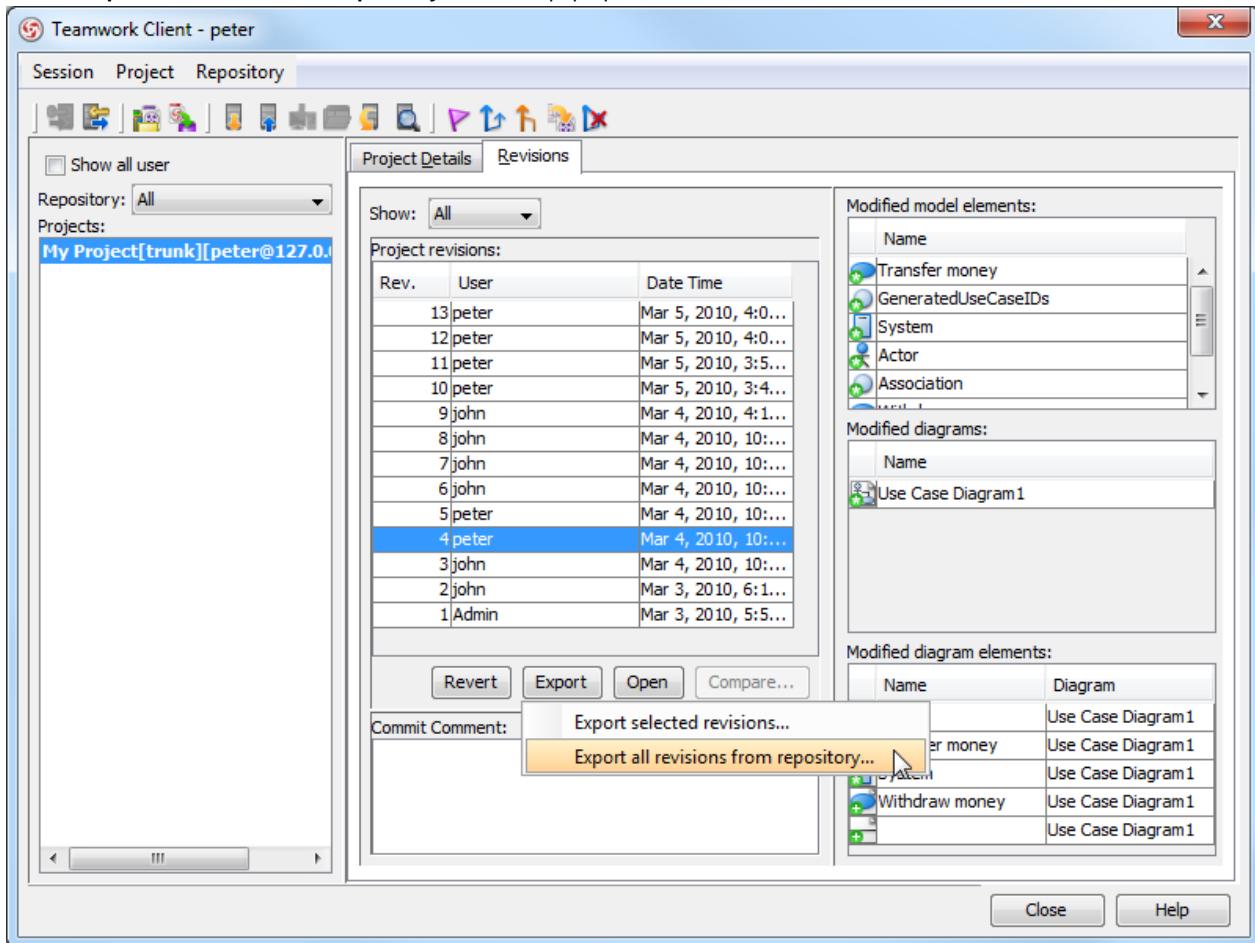


Select a directory

Export all revisions from repository

1. After entering VP-UML, select **Tools > Teamwork > Open Teamwork Client...** from the main menu.

2. In the **Teamwork Client** dialog box, unfold **Revisions** tab. Select the specified revision under **Project revisions**, click **Export** button and choose **Export all revisions from repository...** from the pop-up menu.



*Choosing **Export all revision from repository...** in **Teamwork Client** dialog box*

3. In **Select Directory** dialog box, select an existing folder or make a new folder for storing all revisions.

Working with teamwork client dialog box

To perform teamwork operations such as to commit or update project, you can click the buttons in application menus or toolbars. Besides, you can perform via the Teamwork Client dialog box. Teamwork Client dialog box enables you to perform teamwork operations, manage teamwork projects, as well to review and checkout revisions of particular project.

Opening the teamwork client

1. Open the Teamwork Client by either of the ways below:
 - Select **Tools > Teamwork > Open Teamwork Client...** from the main menu.

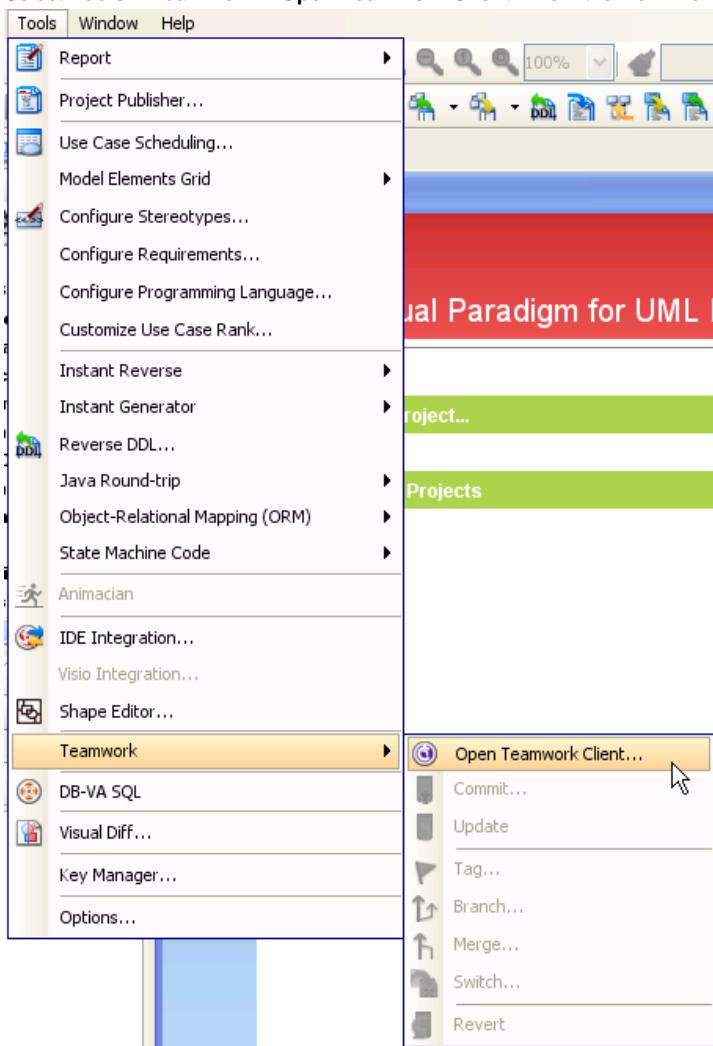


Figure 2-1 Open teamwork client through menu

- Click on the button **Open Teamwork Client** in toolbar.

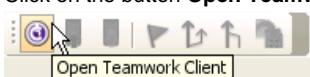


Figure 2-2 Open teamwork client through toolbar

2. In the login dialog box, fill in the server and user information to login to server.

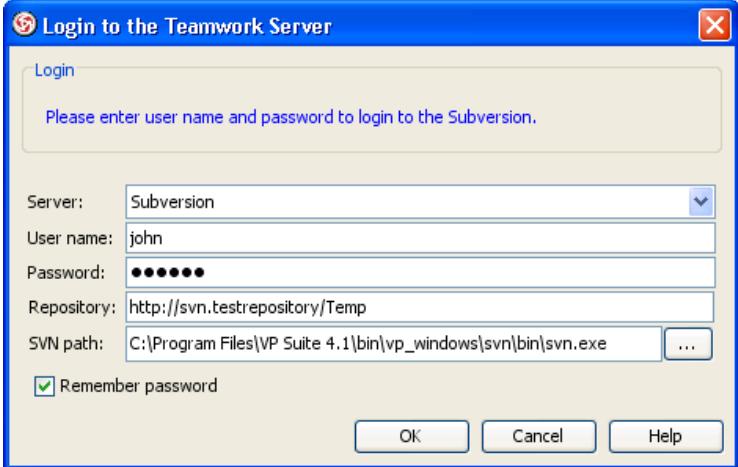


Figure 2-3 Login Subversion

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be Subversion
User name	The user name for connecting to server
Password	The password for given user
Repository	The SVN repository path
SVN Path	The filepath of svn.exe. By default the one in installation folder will be picked up. For Mac OS X client, please download SVN from the following URL and install it manually: http://subversion.tigris.org/getting.html#osx
Remember password	Remember the login password so that you don't need to enter again when login next time

Table 2-1 Description of *Login* dialog box

3. If this is the first time you connect to the SVN repository, you will be asked to select the projects to manage. By managing a project, you can check it out, open and edit it. In the **Manage Project** dialog box, select the projects to manage.

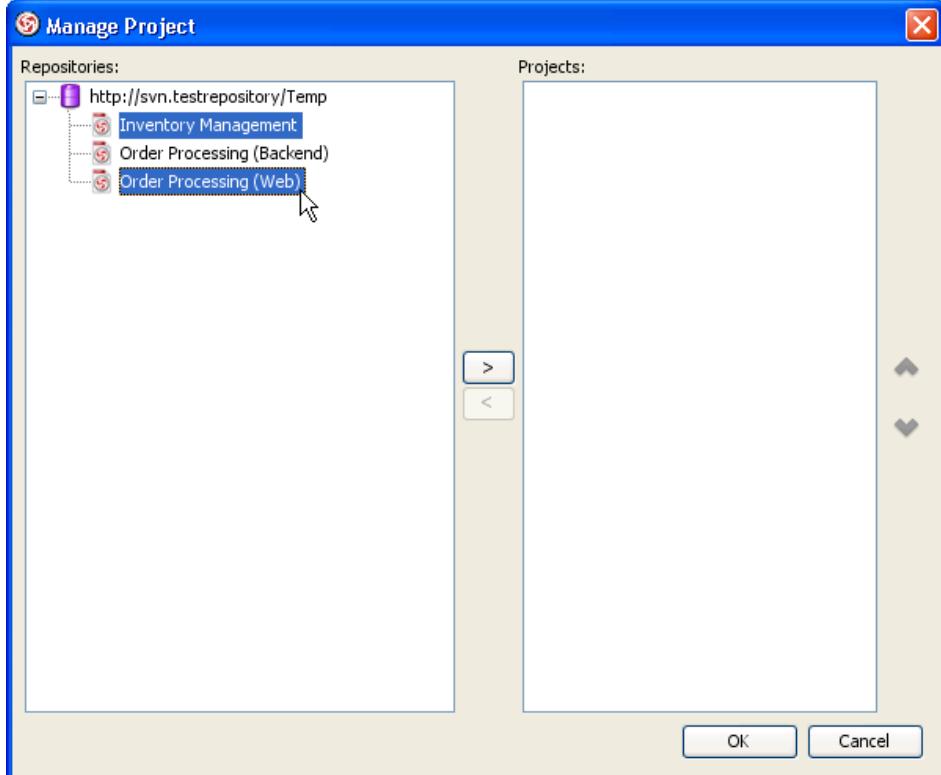


Figure 2-4 Select projects to manage

4. Click on the > button in the middle of the dialog box to add them into project selection.
5. Click **OK** to proceed. This end up showing the **Teamwork Client** dialog box.

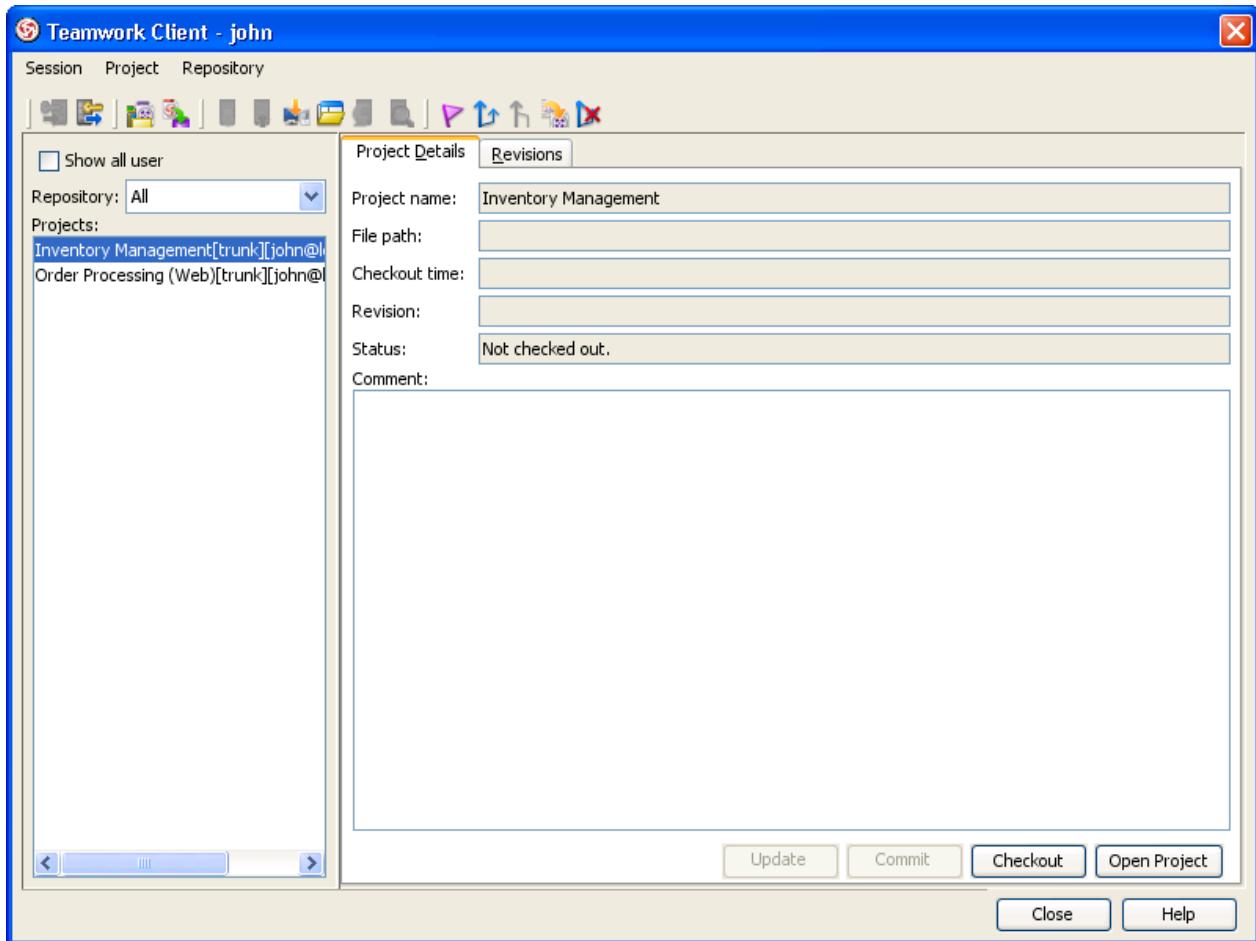


Figure 2-5 The Teamwork Client dialog box

The interface

General

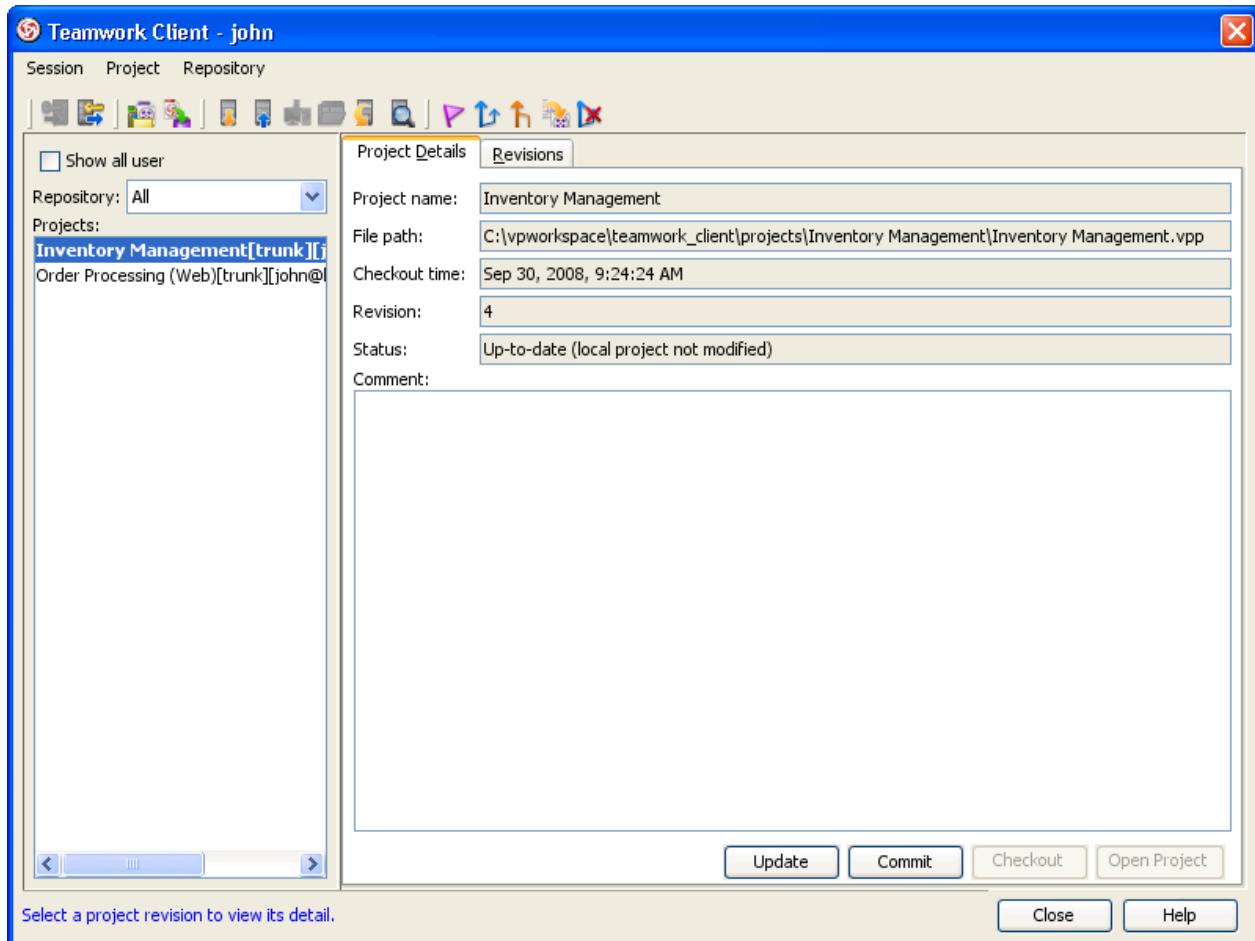


Figure 2-6 The Teamwork Client dialog box

Region	Description
	To logout from SVN repository.
	To login to SVN repository.
	To manage projects, which enables you to add or remove projects from the list of managed projects.
	To create a project in server by importing one into it. You may create a blank new project, or import an existing project file to start with.
	To update changes from server.
	To commit changes to server.
	To checkout a project from server.
	To open the project selected in Projects list. If the project is not checked out yet, it will be checked out automatically.
	To remove changes made in a revision.
	To check for changes
	To create a tag for a trunk/branch.
	To create a branch for a trunk/branch.
	To merge changes from trunk/branch.
	To switch to another trunk/branch/tag.
	To remove a branch. Note that removed branch will not delete the, but just make it not accessible.

Show all user	To show the projects manageable by all users.
Repository	To update the Projects list by showing only projects in certain repository.
Projects	The list of manageable project.
Project Details	Details of selected project, including the name, revision and status.
Revisions	Revisions of selected project.

Table 2-2 Description of the Teamwork Client dialog box

Figure 2-7 The Project Details tab in Teamwork Client dialog box

Field	Description
Project name	Name of the selected project.
File path	The file path of the project in your system.
Checkout time	The time when you checkout the project.
Revision	The revision of your local copy.
Status	The status of local project.
Comment	The comment of project entered by the administrator when creating project.

Table 2-3 Description of Project Details tab in Teamwork Client dialog box

Revisions

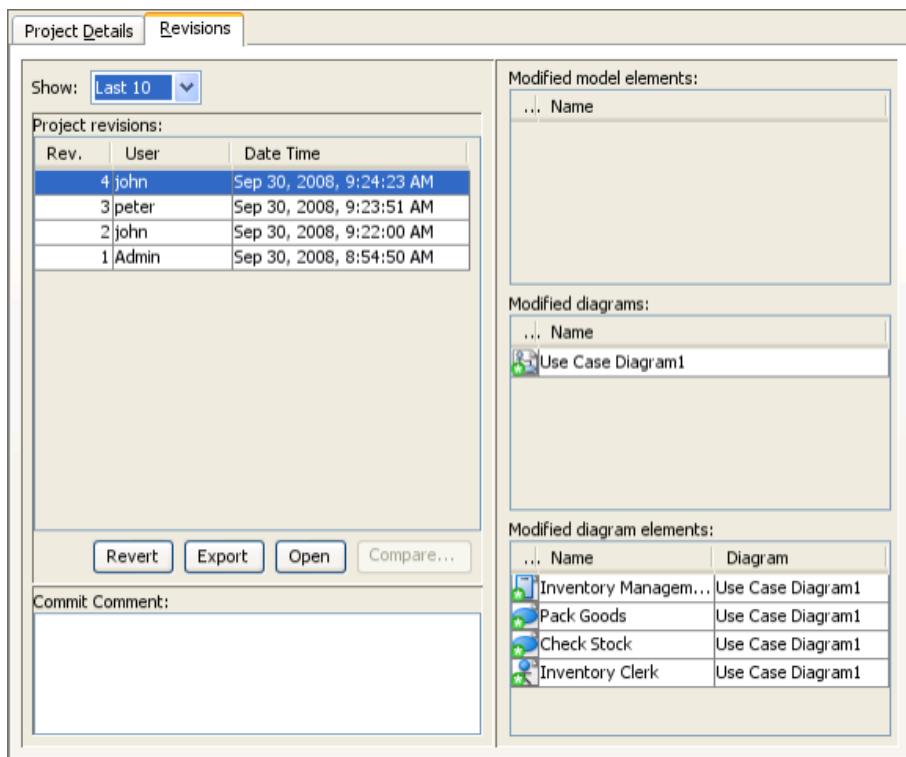


Figure 2-8 The Revisions tab in Teamwork Client dialog box

Region	Description
Show:	To list certain amount of revisions, such as all, last 10, last 20, etc.
Revision list	To display the revisions of chosen project.
[Revert]	By reverting a selected revision, this means to undone changes made in that revision.
[Export]	To export selected revision(s) as project files.
[Open]	To open selected revision.
[Compare]	To compare the differences between revisions.

Table 2-4 Description of Revisions tab in Teamwork Client dialog box

Importing projects to Subversion

1. Start VP-UML.
2. Select **Tools > Teamwork > Open Teamwork Client...** from the main menu, or **Open Teamwork Client** icon from toolbar to open teamwork client.

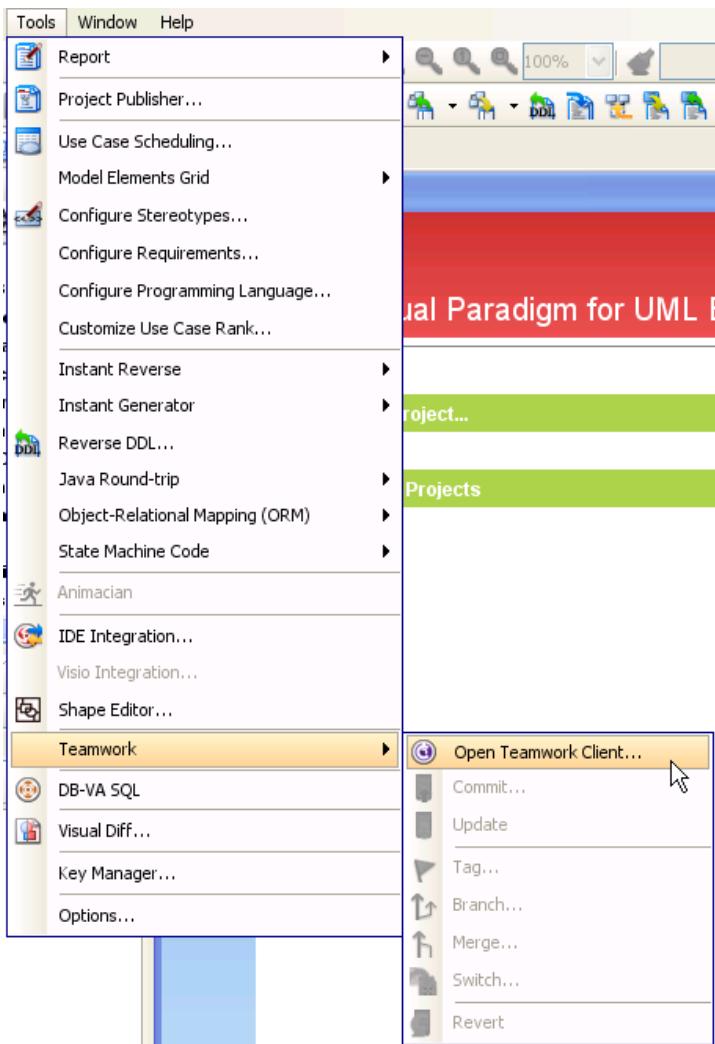


Figure 2-9 Open Teamwork Client from main menu

3. Select **Subversion** in **Server** field.

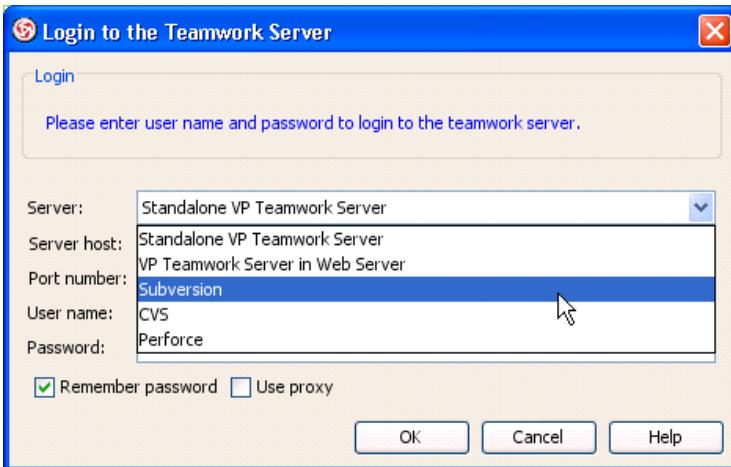


Figure 2-10 subversion teamwork server

4. Fill in the server and user information to login teamwork server.

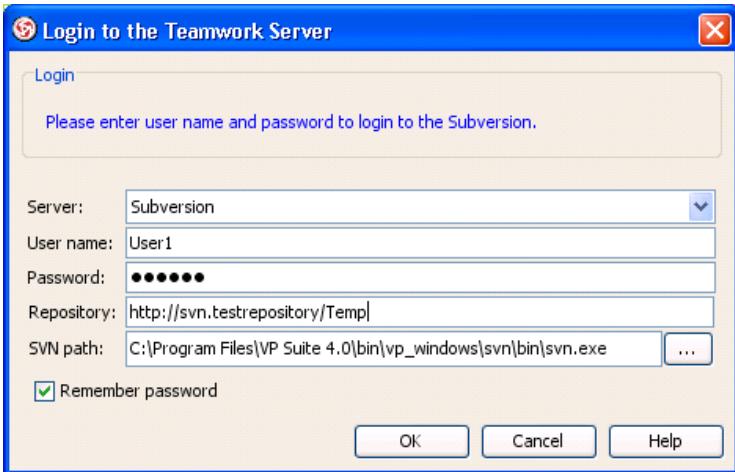


Figure 2-11 Login teamwork server

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be Subversion
User name	The user name for connecting to server
Password	The password for given user
Repository	The SVN repository path
SVN Path	The filepath of svn.exe. By default the one in installation folder will be picked up. For Mac OS X client, please download SVN from the following URL and install it manually: http://subversion.tigris.org/getting.html#osx
Remember password	Remember the login password so that you don't need to enter again when login next time

Table 2-5 Description of **Login** dialog box

5. Select **Project > Import Project to Repository** from the menu.

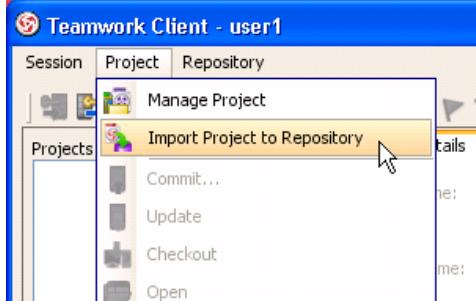


Figure 2-12 Import project from menu

Or click the **Import Project to Repository** button on toolbar.

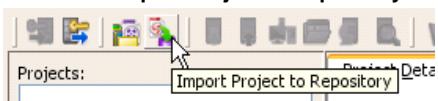


Figure 2-13 Import project from toolbar

6. In the **Import Project** dialog, fill in the **Project name**.

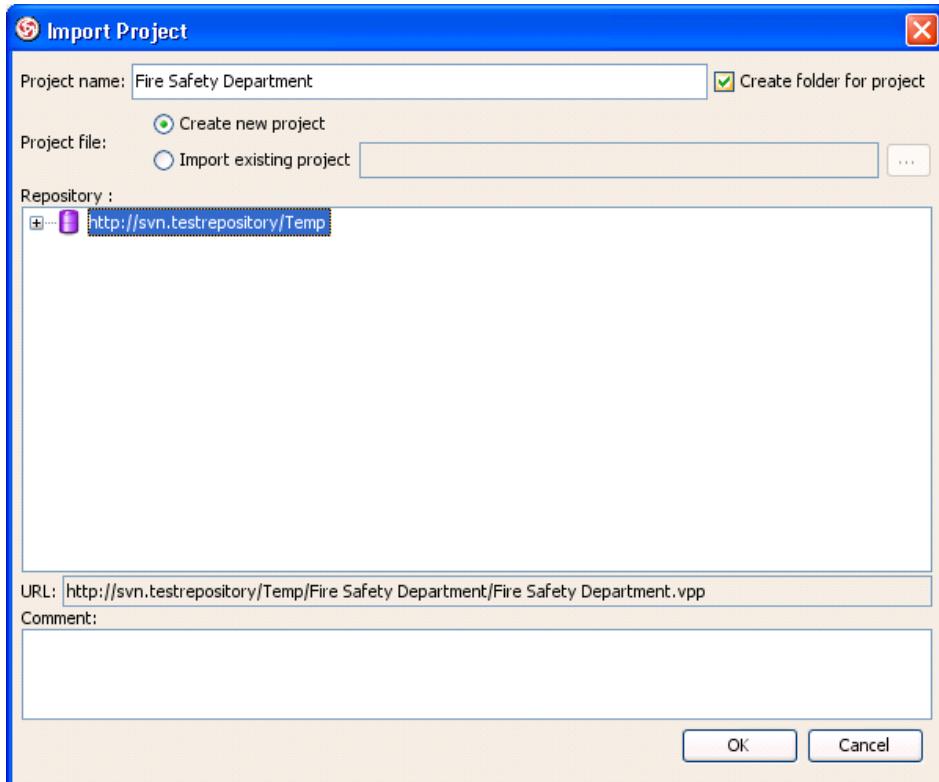


Figure 2-14 Fill project name in Import Project dialog

Field	Description
Project name	The name of teamwork project
Create folder for project	Create a folder in repository for storing the project. The folder will be of same name as the project.
Project file	Create new project - Start with a blank, new project Import existing project - Start by using an existing project file
Repository	A tree which shows the structure of repository
URL	The URL of the project in repository
Comment	The comment of project

Table 2-6 Description of Import Project dialog box

7. Choose **Create new project** or **Import existing project** from **Project file**. **Create new project** will import a blank project into teamwork server, **Import existing project** will import an existing VP-UML Project file (*.vpp) into teamwork server as first revision.



Figure 2-15 Specify project file

NOTE: Please make sure the user login to **Teamwork Client** has **Create Project** permission

8. Select the repository for import the project.

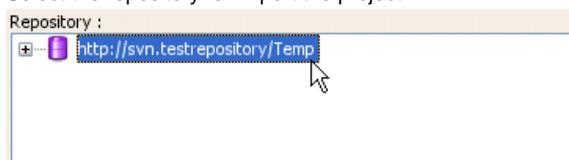


Figure 2-16 Select repository

9. Fill in the comment.

Comment:

Importing project

Figure 2-17 Input comment

10. Click **OK** button to import project.

Checkout project from Subversion

Checkout and open Subversion projects

1. Open Teamwork Client dialog.
2. Select Project > Manage Project from the menu.

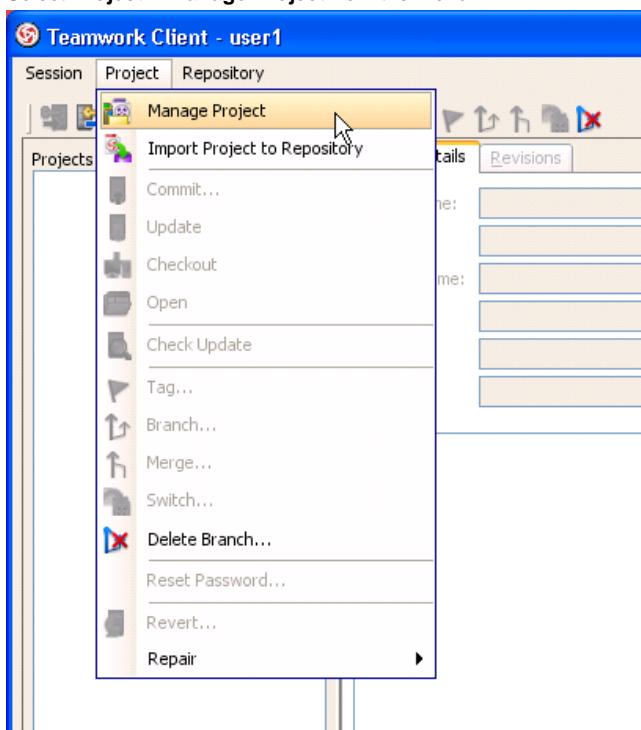


Figure 2-18 Manage project from main menu

Or click **Manage Project** icon on the toolbar.

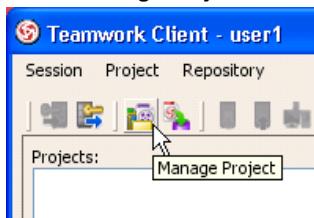


Figure 2-19 Manage project from toolbar

3. In **Manage Project** dialog, expand the repository node and select the project to checkout, click > button to add project. Click **OK** button to close the dialog.

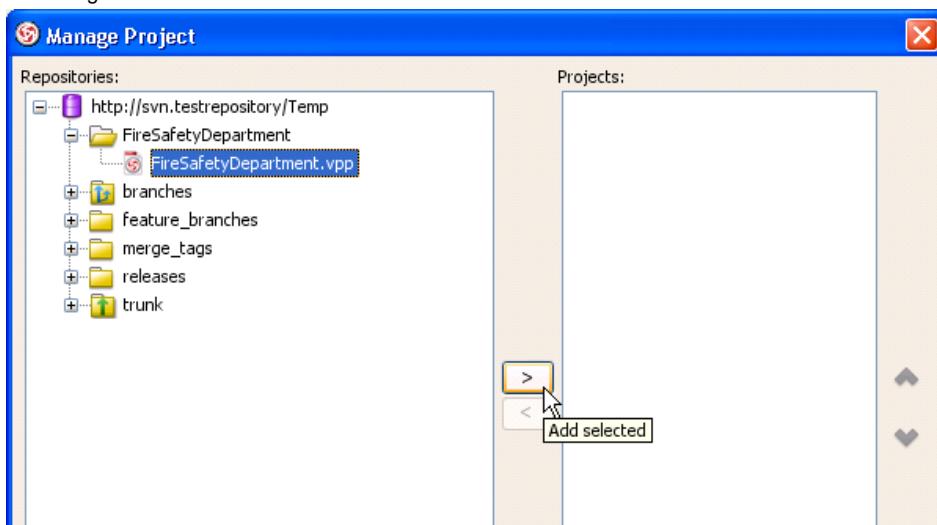


Figure 2-20 Manage projects

4. Select the project in project list.

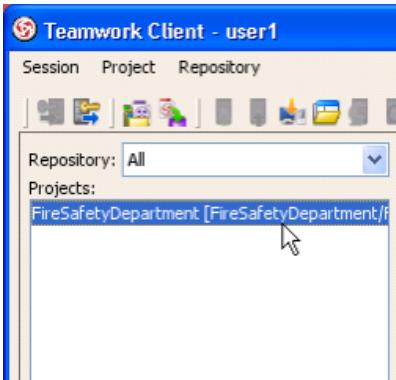


Figure 2-21 Select project from the list

5. Click **Open Project** button to checkout and open the project.



Figure 2-22 Open project

6. The project opened in VP-UML.

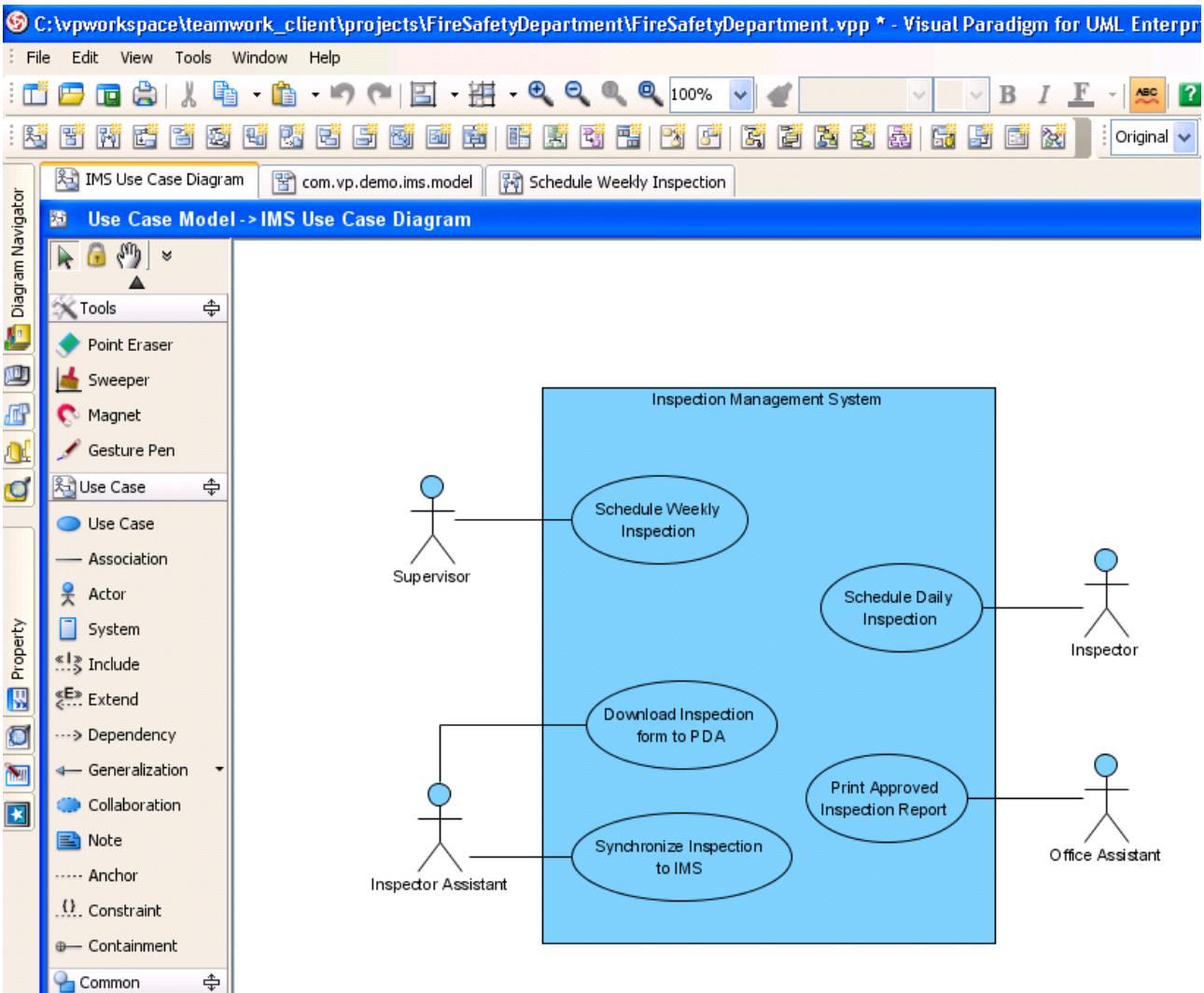


Figure 2-23 Project opened

Committing local modification to Subversion

1. Modify the project.

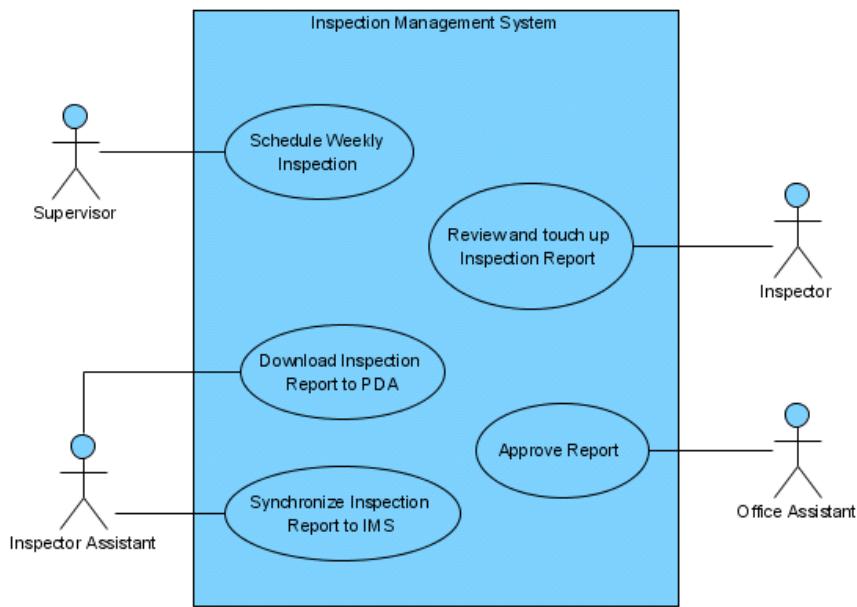


Figure 2-24 Modified project

2. Show **Teamwork Toolbar** by right click on the Toolbar, select **Teamwork** if not selected already.

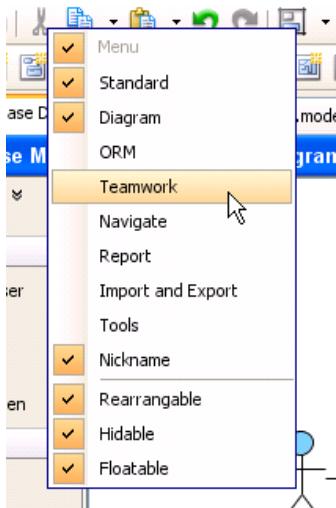


Figure 2-25 Show teamwork toolbar

3. Click the **Commit** button on **Teamwork Toolbar**.

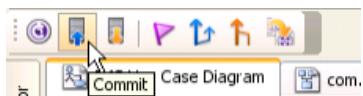


Figure 2-26 Commit project

4. In **Commit** dialog, you can review the changes for commit. On the left of dialog, you can see a list of changes shapes and model elements, click on it to view the detail changes on the right.

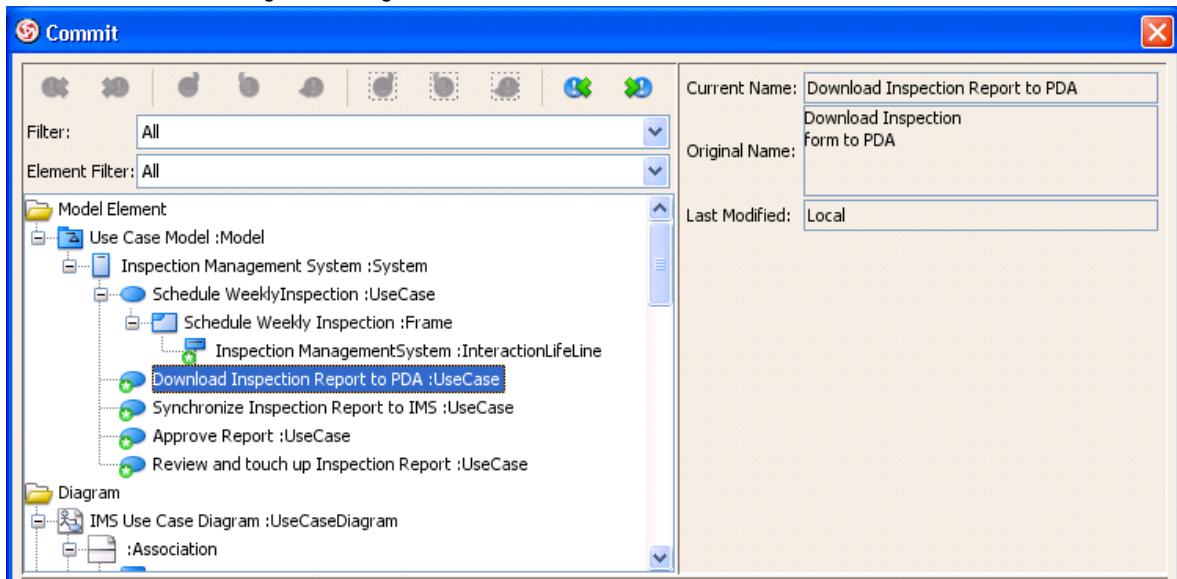


Figure 2-27 Review commit changes

5. On the bottom of dialog, click the **Preview** tab to visually preview the changes in diagram. The selected shape is highlighted in purple.

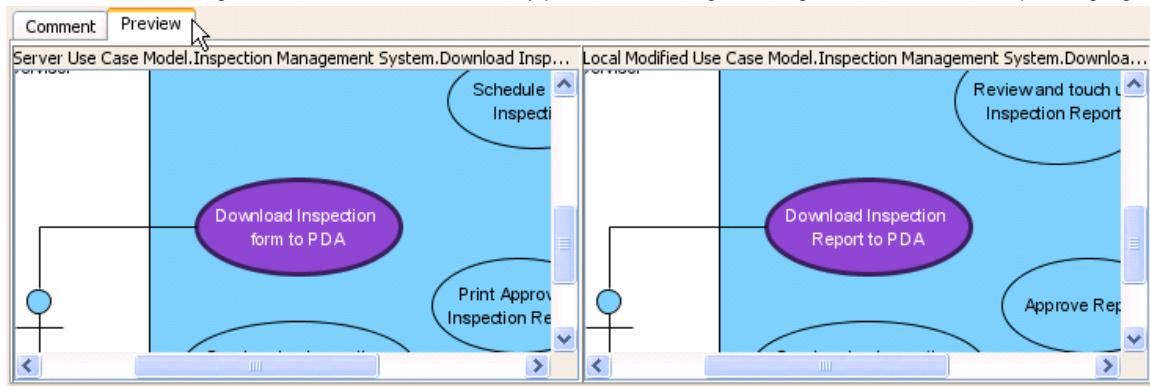


Figure 2-28 Preview commit changes

6. After review the changes, click the **Comment** tab and input the comment for commit. Click **OK** button to start commit.

Figure 2-29 Input commit comment

Resolving conflicts

1. Modify the project.

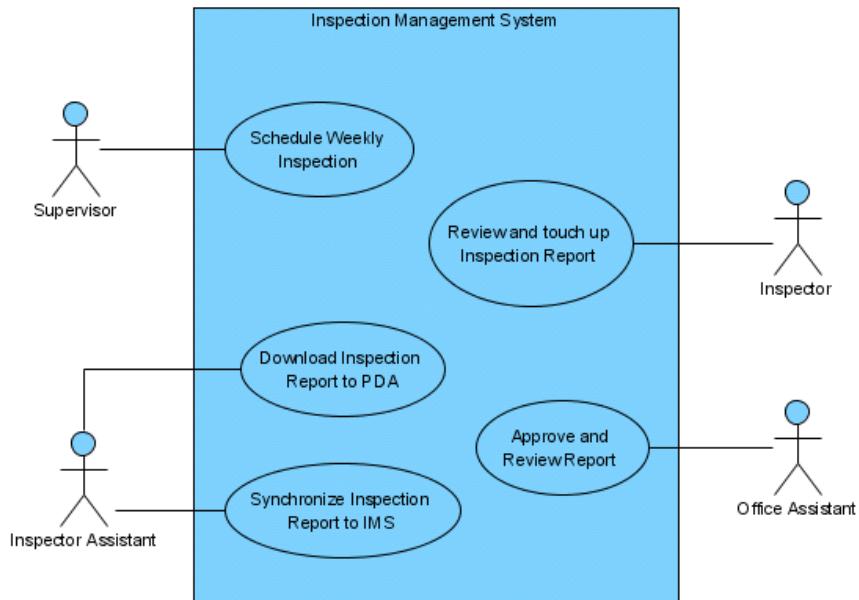


Figure 2-30 Modified project

2. Click **Commit** button on the **Teamwork Toolbar**.

3. In **Commit** dialog, you may found some shapes or model elements show with red icon. This indicate there is conflict when commit, it is caused by someone modified the same content and commit after you checkout. You can review the current value, original value (the value when you checkout), and conflict value (the value changed by other users). And dialog preview is disabled until you resolve the conflict.

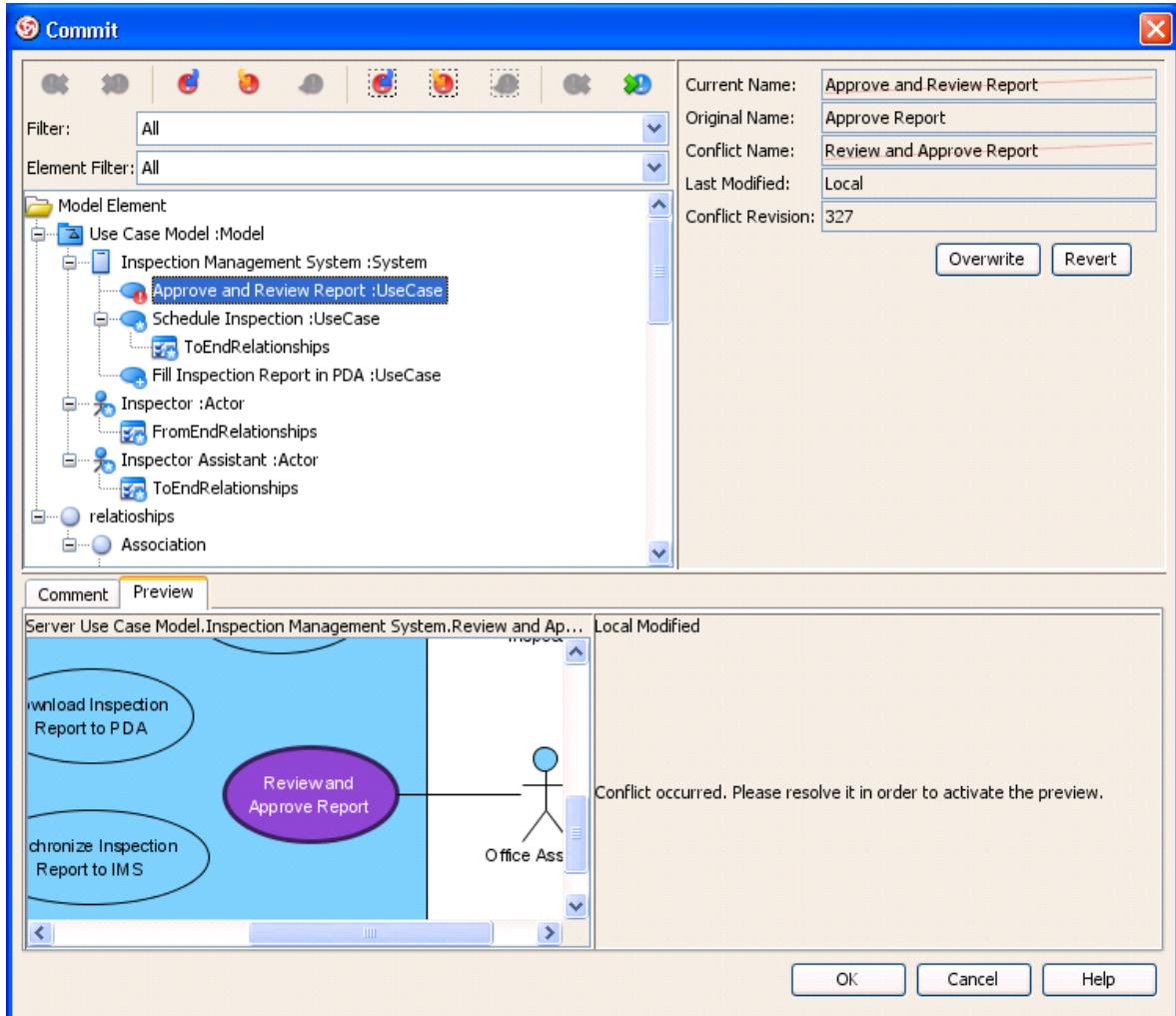


Figure 2-31 Commit with conflicts

Below is a description of buttons or fields in dialog box.

Button/Field	Description
	Move to the previous conflict in element tree.
	Move to the next conflict in element tree.
	Overwrite all conflicts.
	Revert all conflicts.
	Reset all conflicts.
	Overwrite the selected conflict in tree.
	Revert the selected conflict in tree.
	Reset the selected conflict in tree.
	Select the previous change in tree.
	Select the next change in tree.
Current Name	The value of name of the committing/updating copy.
Original Name	The value of name of the original copy.
Conflict Name	The value of name of the server copy.
Last Modified	The user who modified last time.
Conflict Revision	The revision that cause conflict with the committing/updating action.
[Overwrite]	Click on overwrite the conflict selected in tree.
[Revert]	Click on revert the conflict selected in tree.

4. Select the conflict element, click **Overwrite selected conflicts** button on the toolbar to overwrite other user's change.



Figure 2-32 Overwrite selected conflicts

Or click **Revert selected conflicts** to revert your own changes.

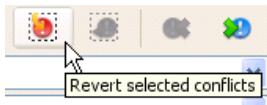


Figure 2-33 Revert selected conflicts

You can also click **Overwrite all conflicts** or **Revert all conflicts** to overwrite or revert all conflicts at once.



Figure 2-34 Overwrite/Revert all conflicts

5. After resolve conflict, the preview will be enabled to visualize the final result.

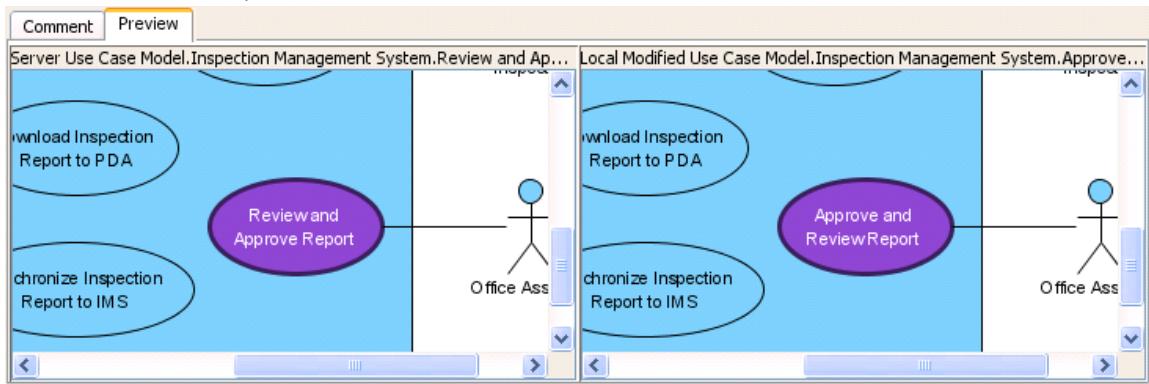


Figure 2-35 Preview resolved conflict

6. You can change your mind by click **Reset selected conflicts** or **Reset all conflicts** button. Then overwrite or revert the conflict again.

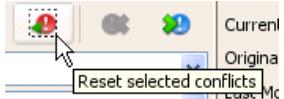


Figure 2-36 Reset conflicts

7. After all conflicts was resolved, you can now input comment and click **OK** button to commit.

Updating latest revision from Subversion

Updating modification from Subversion

1. Click **Update** button on **Teamwork Toolbar**.



Figure 2-37 Update from toolbar

2. Similar to commit, you can review the change and preview the diagram in **Update** dialog.

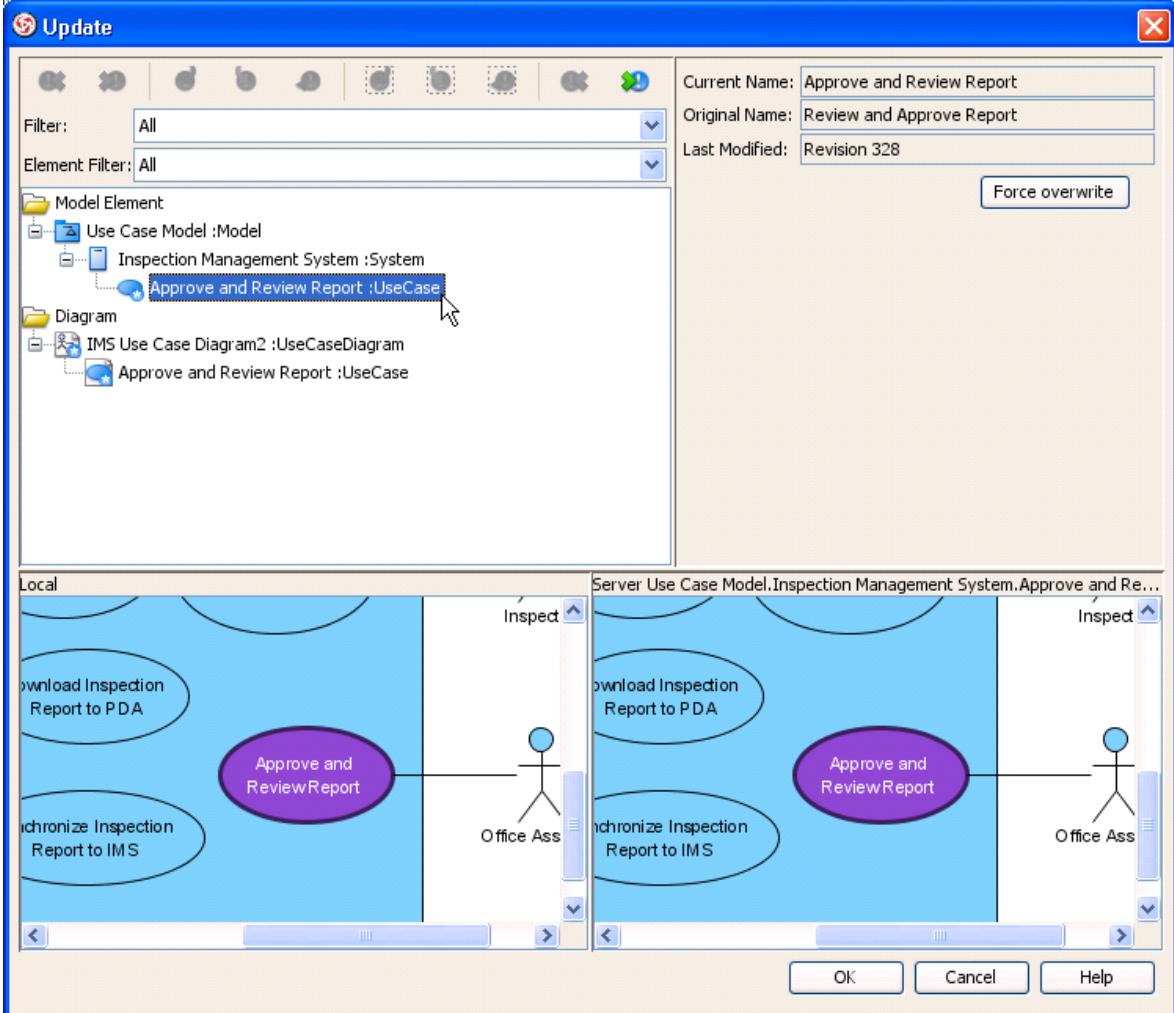


Figure 2-38 Update dialog

3. If there are conflicts, resolve it similar to commit, but the changes will apply to local project instead of commit to server immediately.

4. Click **OK** button to update. **VP-UML** will open the updated project.

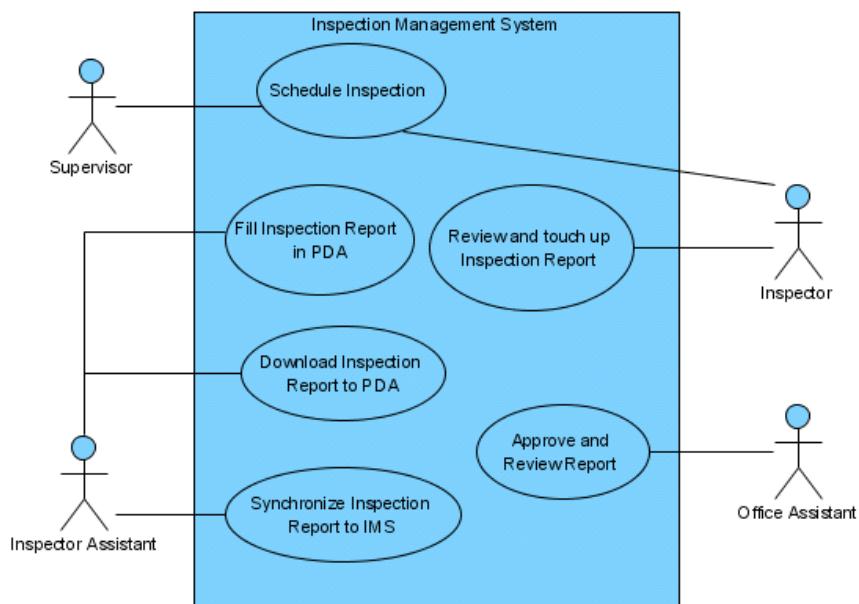


Figure 2-39 Updated project

Checking status

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Check Update** from menu.

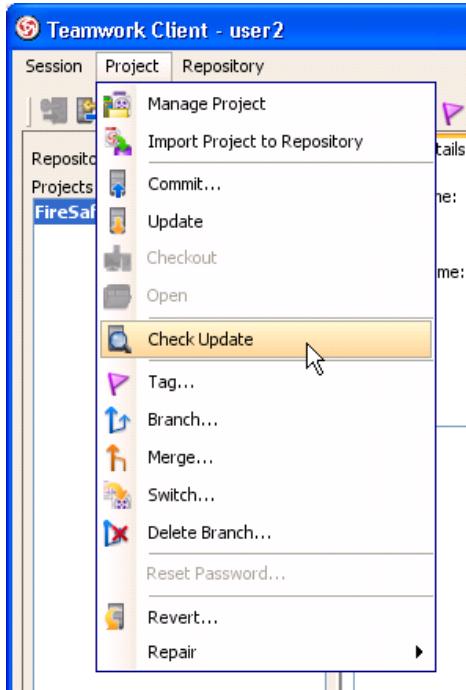


Figure 2-40 Update project from menu

Or click **Check Update** button from toolbar.



Figure 2-41 Update project from toolbar

4. You can see the status showing **Has update** or **Up-to-date**, it also indicate local project status (**local project not modified**) or (**local project modified**).

Status: Up-to-date (local project not modified)

Figure 2-42 Project status

Rolling back undesired changes by reverting changes

Roll back local and uncommitted modifications

1. Modify the project.

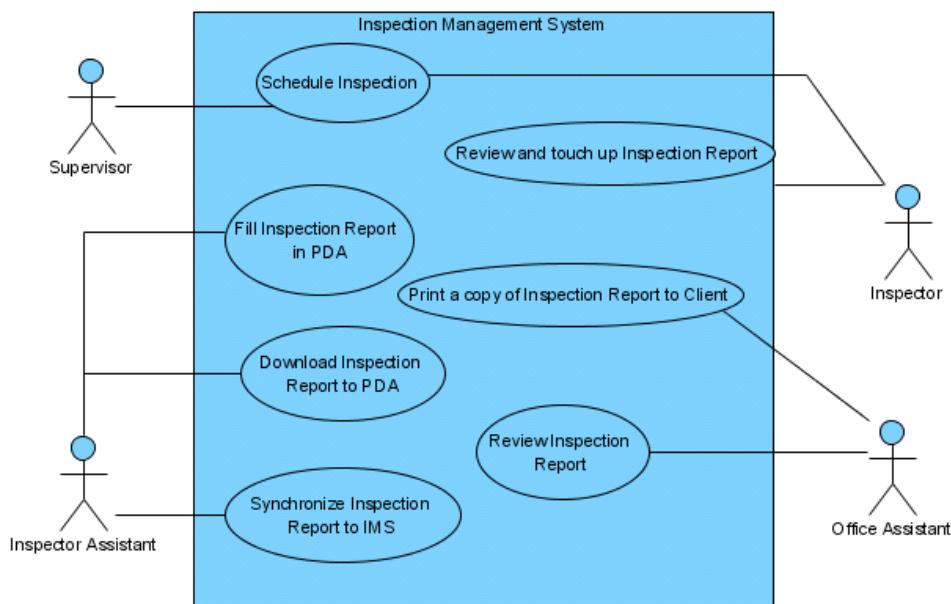


Figure 2-43 Modify project

2. Open Teamwork Client dialog.

3. Select the teamwork project in the list.

4. Select Project > Revert... from menu

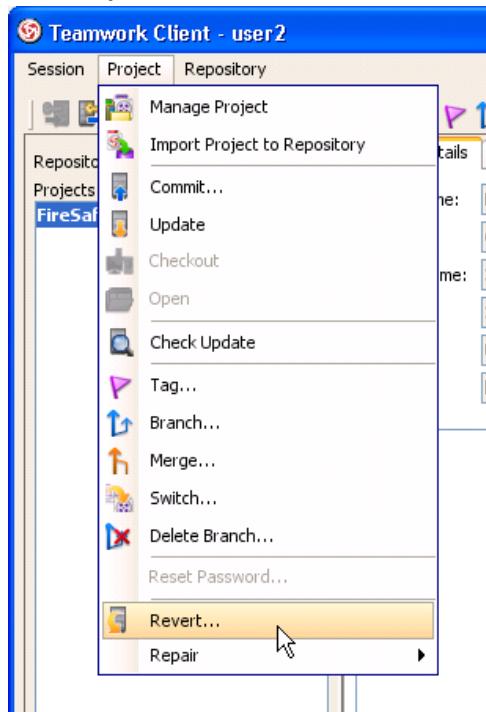


Figure 2-44 Revert from menu

Or click Revert... button from toolbar.



Figure 2-45 Revert from toolbar

5. Click Yes button from the confirmation dialog.

6. The reverted project opened in VP-UML.

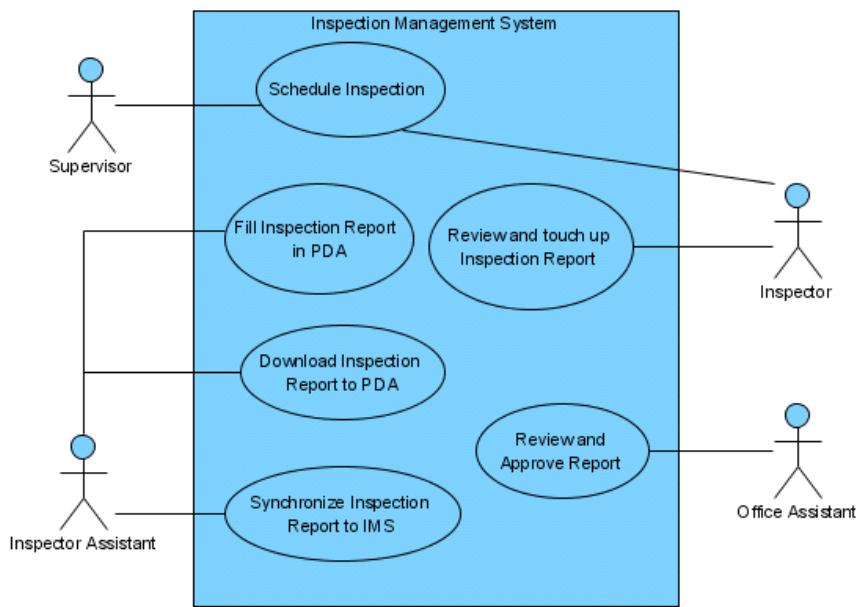


Figure 2-46 Reverted project

Roll back modifications of a past revision

If there were undesired changes made in past revision(s), such as wrong deletion of elements, you can undo the changes by reverting the revision(s) involved. Here are the steps:

1. Open project.

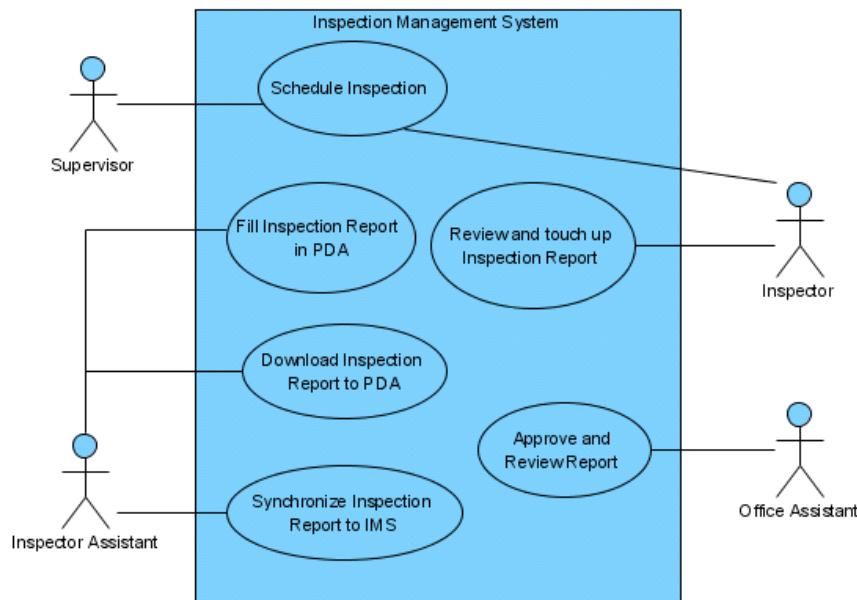


Figure 2-47 Open project

2. Open **Teamwork Client** dialog.
3. Select the teamwork project in the list.

4. Click the **Revisions** tab on the right of the project list.

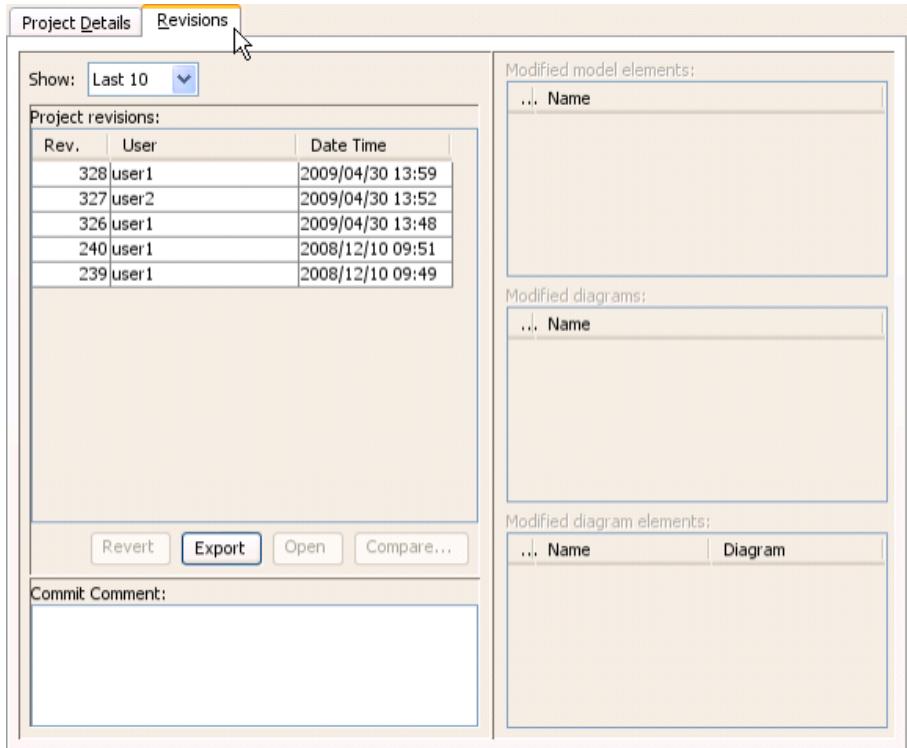


Figure 2-48 Revision tab

5. Select the revision(s) to have its changes rolled back.

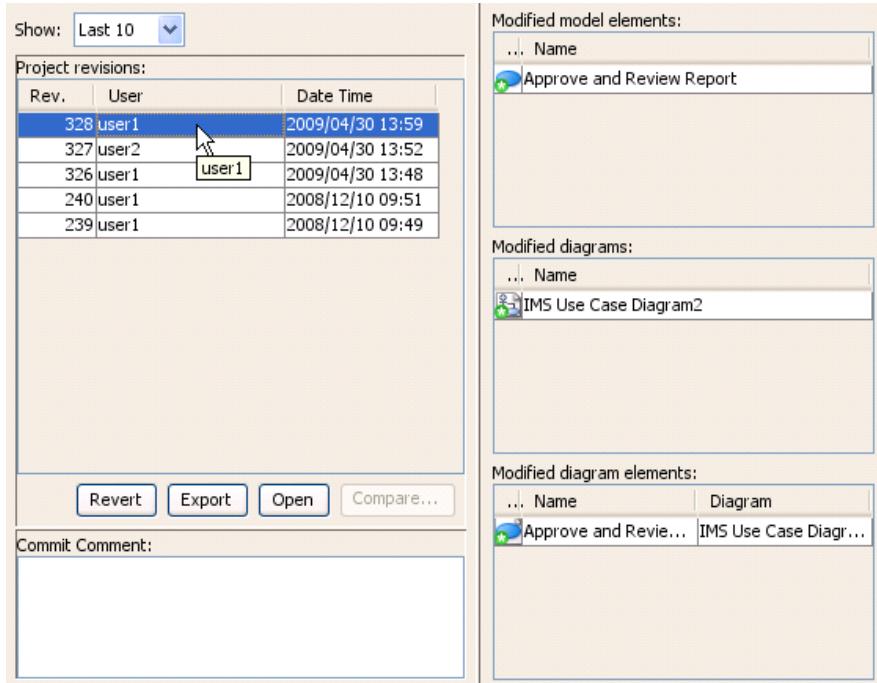


Figure 2-49 Select revision

6. Click **Revert** button.



Figure 2-50 Revert button

NOTE: You may make multiple selection by selecting one revision, pressing the **Control** key, and selecting the rest. Note that a non-consecutive revision selection is not revertable.

7. The **Commit** dialog show a list of shapes and model elements reverted for review the changes and preview diagram, input the comment and click **OK** button to commit the revert.

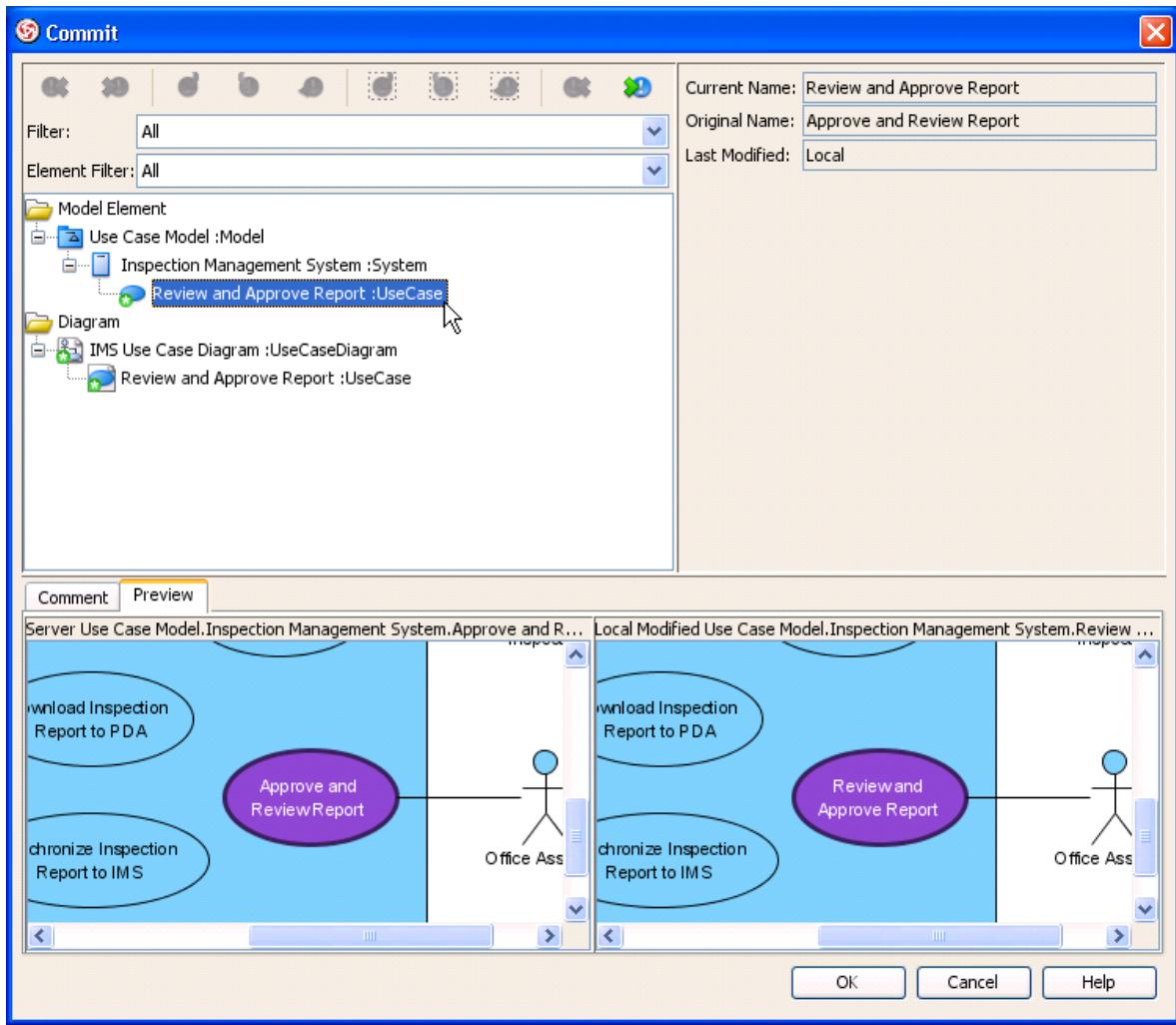


Figure 2-51 Commit dialog with reverted changes

8. A new revision with the reverted changes was created in server, and opened in VP-UML.

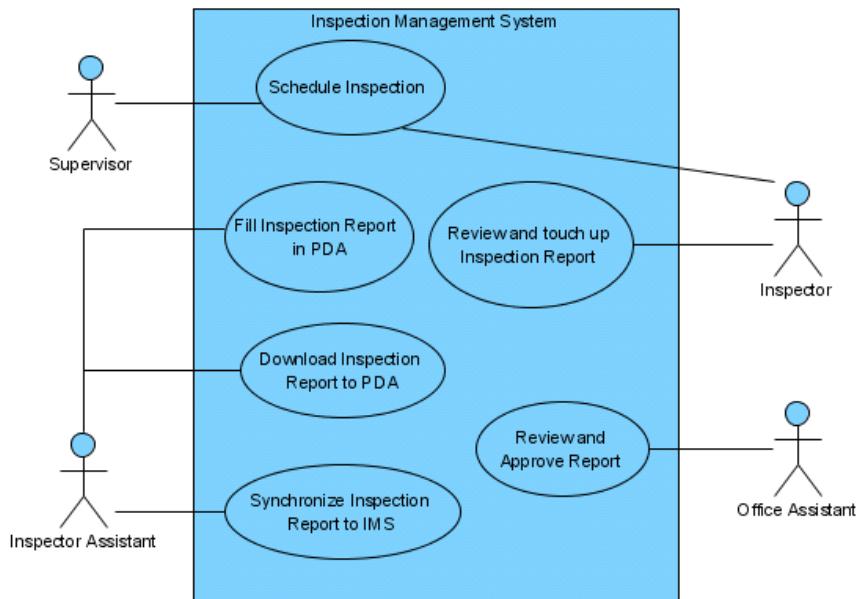


Figure 2-52 Reverted project

Browsing change histories (old revisions)

Checkout old revisions

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select the revision to open.
5. Click **Open** button.

The screenshot shows the 'Revisions' tab in the Teamwork Client. On the left, a table lists 'Project revisions' with columns for Rev., User, Date, and Time. Revision 326 by user1 is selected. Below the table are buttons for 'Revert', 'Export', 'Open' (which is highlighted with a yellow box), and 'Compare...'. A 'Commit Comment:' field contains 'Modify Use Case Diagram'. On the right, three sections show 'Modified model elements', 'Modified diagrams', and 'Modified diagram elements', each listing several items with icons and names.

Figure 2-53 Open button

6. Selected revision opened in **VP-UML**.

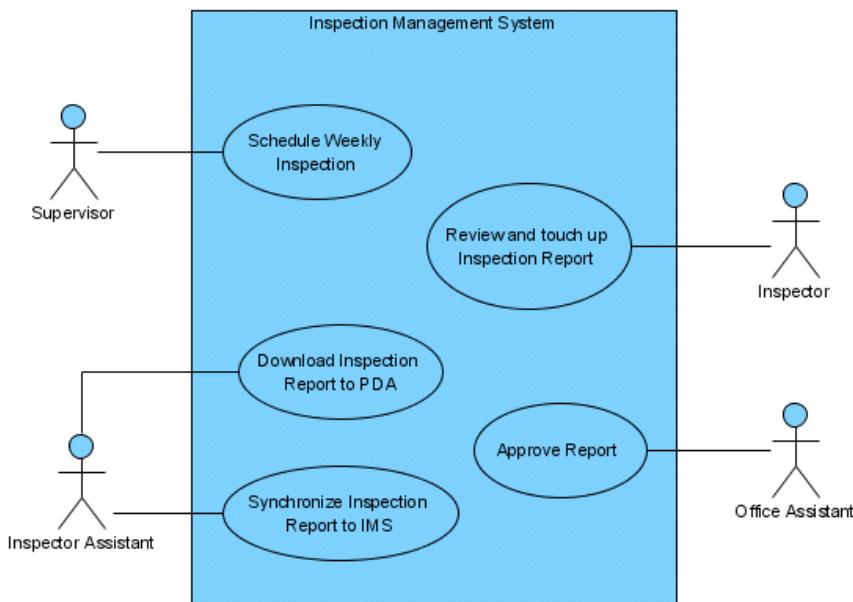


Figure 2-54 Open selected revision

Showing differences between revisions visually

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select a revision for compare.

5. Select another revision by **Ctrl+Click**.

Rev.	User	Date Time
329	user2	2009/04/30 14:09
328	user1	2009/04/30 13:59
327	user2	2009/04/30 13:52
326	user1	2009/04/30 13:48
240	user1	2008/12/10 09:51
239	user1	2008/12/10 09:49

Figure 2-55 Select two revisions

6. Click the **Compare...** button.



Figure 2-56 Compare button

7. Similar to **Commit** and **Update** dialog, the **Compare Projects** dialog show a list of differences between the selected revisions. You can also view the differences visually in diagram on the preview tab.

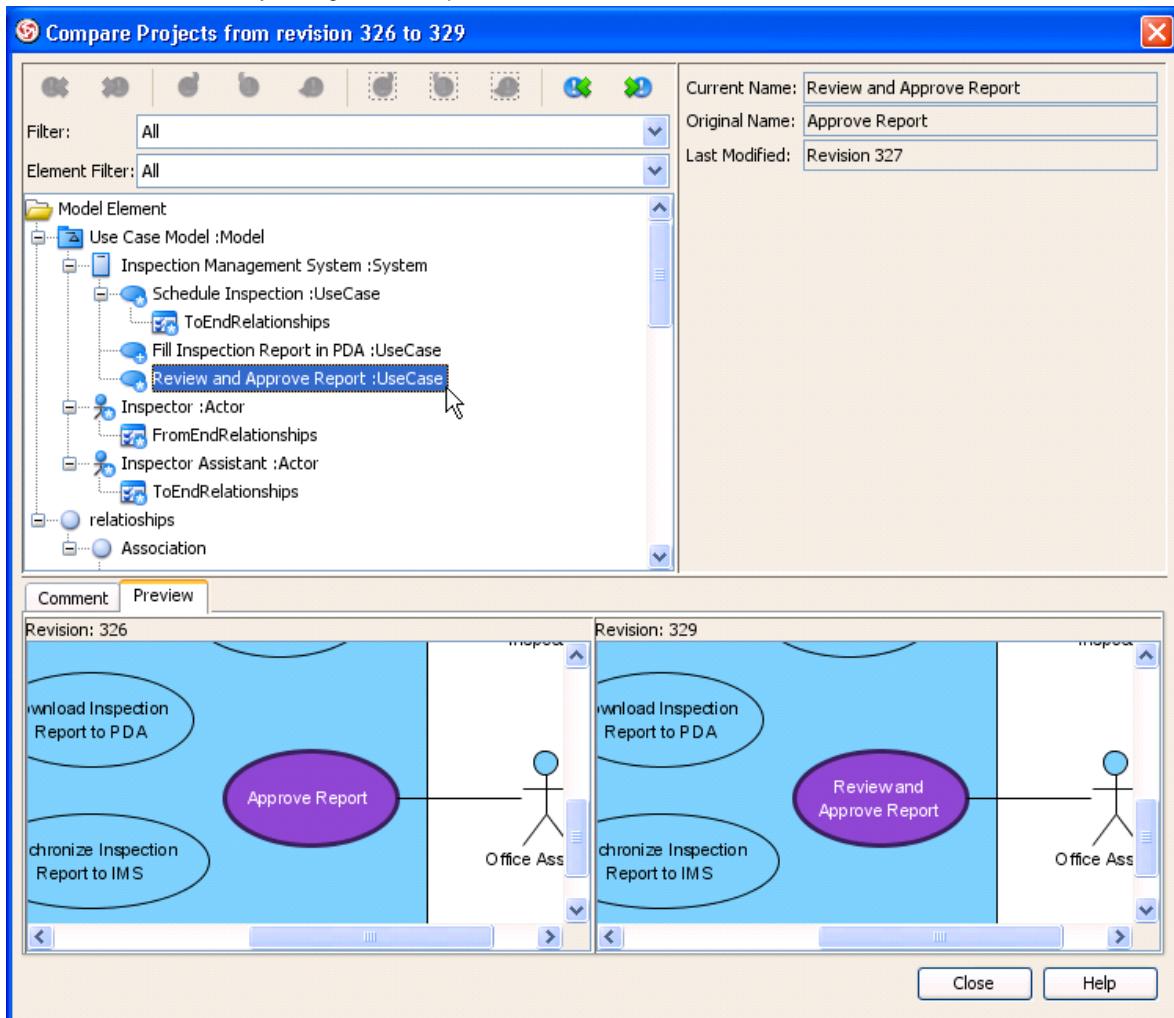


Figure 2-57 Compare differences

Export multiple revisions to local

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.

4. Select multiple revisions for export, by **Ctrl+Click** or **Shift+Click**.

Rev.	User	Date Time
329	user2	2009/04/30 14:09
328	user1	2009/04/30 13:59
327	user2	2009/04/30 13:52
326	user1	2009/04/30 13:48
240	user1	2008/12/10 09:51
239	user1	2008/12/10 09:49

Figure 2-58 Select multiple revisions

5. Click the **Export** button.



Figure 2-59 Export buttons

6. Select **Export selected revisions...** from the popup.

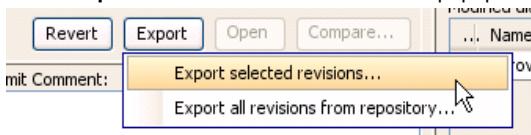


Figure 2-60 Export selected revisions

7. Select the directory to save the exported project revisions.

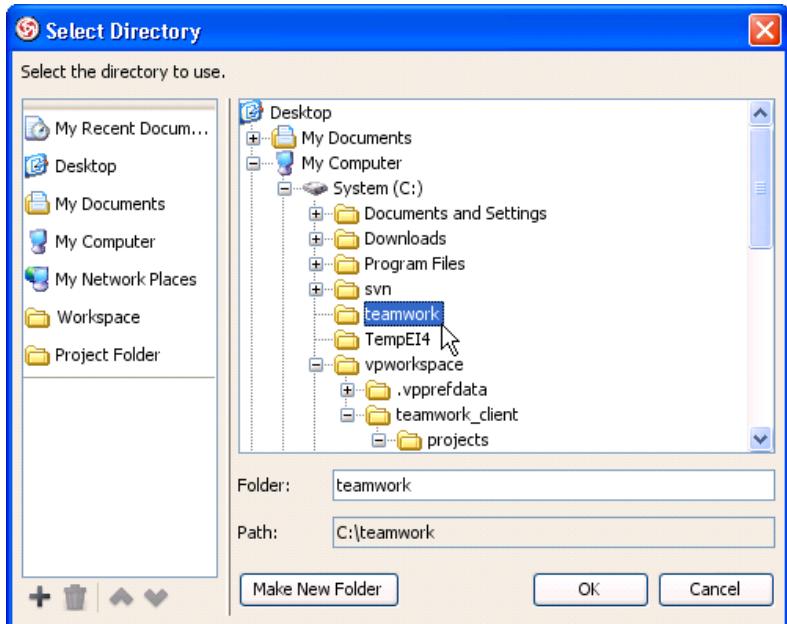


Figure 2-61 Select directory

8. You'll found the selected revisions was exported to the select directory.

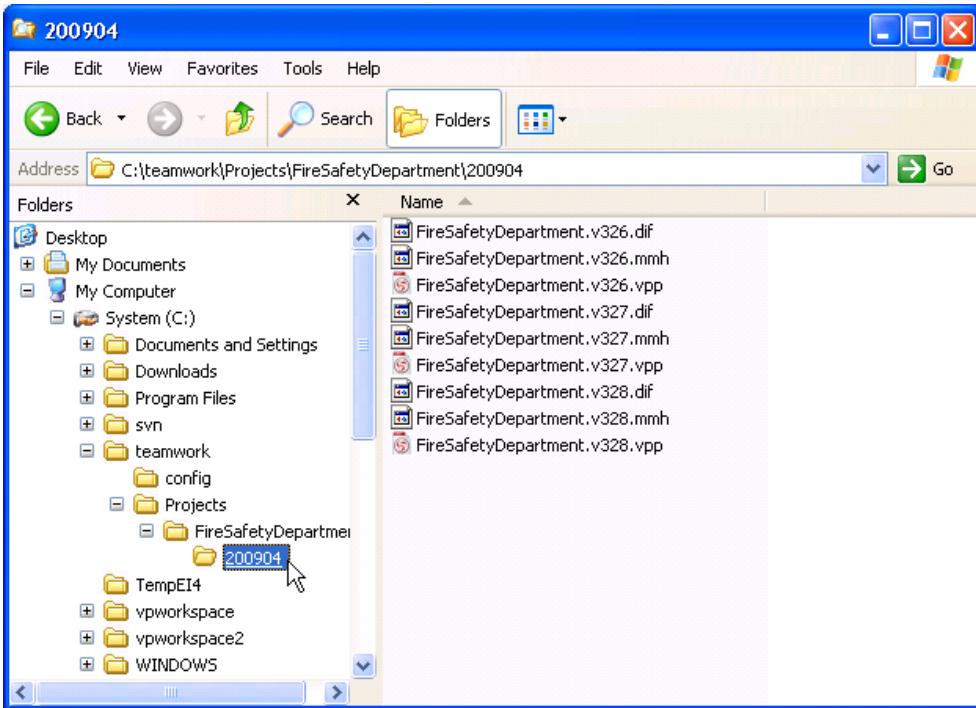


Figure 2-62 Exported revisions

Isolating last long modifications with branches

Creating branch

1. Open Teamwork Client dialog.
2. Select the teamwork project in the list.
3. Select Project > Branch... from menu

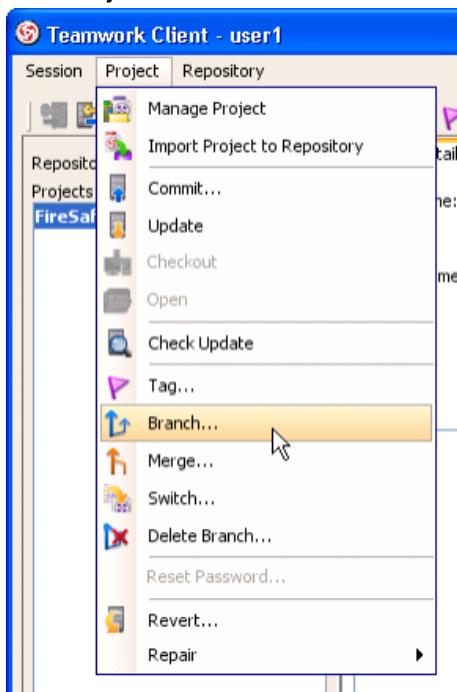


Figure 2-63 Branch from menu

Or click Branch... button from toolbar.



Figure 2-64 Branch from toolbar

4. Fill in the Branch Name in Create Branch dialog.

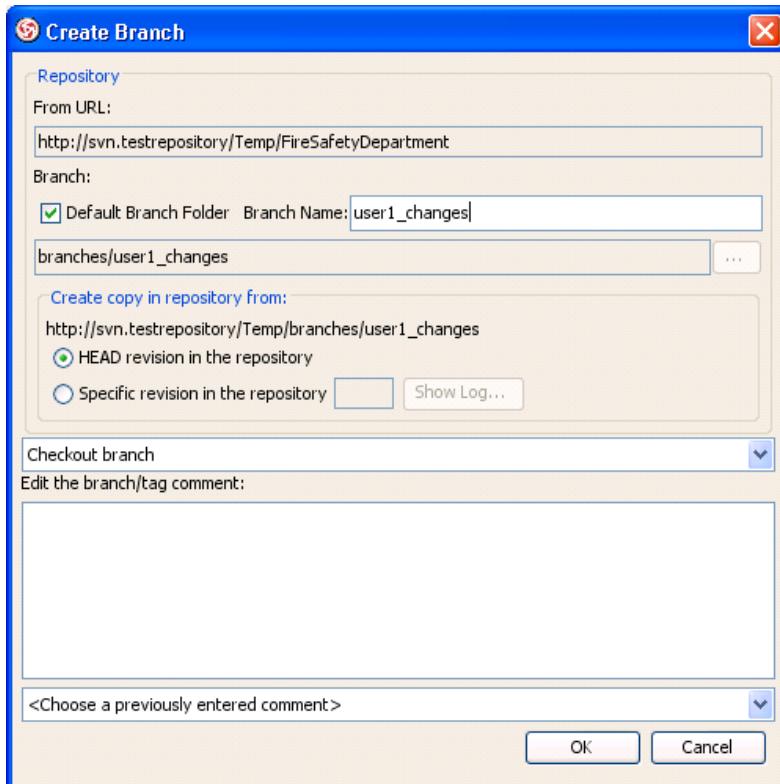


Figure 2-65 Create Branch dialog

5. Select **Stay in trunk** in the list.



Figure 2-66 Stay in trunk

6. If the folder is not exists in the server, it will display a warning dialog for asking create or not.

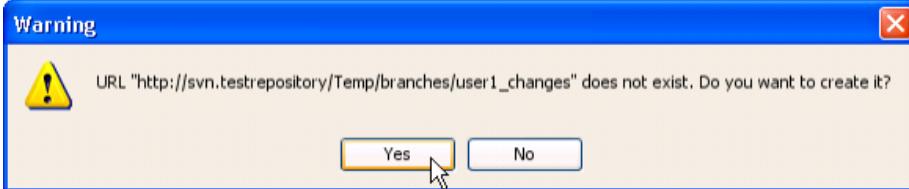


Figure 2-67 Create folder warning

Switch local copy between branches

1. Select **Project > Switch...** from menu

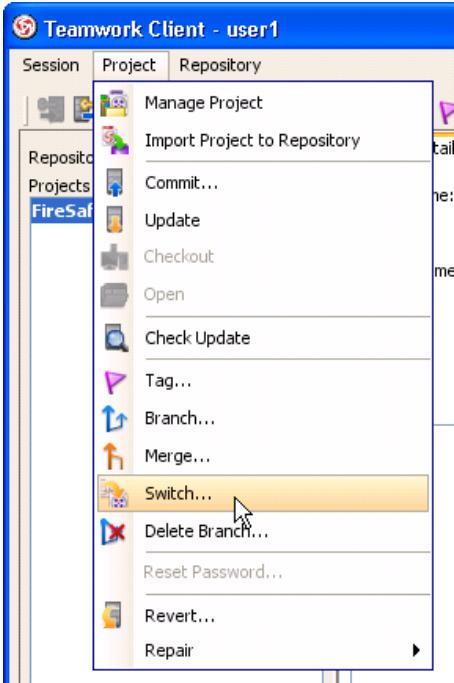


Figure 2-68 Switch from menu

Or click **Switch...** button from toolbar.

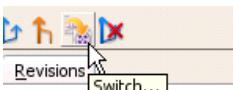


Figure 2-69 Switch from toolbar

2. Click ... to show the server hierarchy.

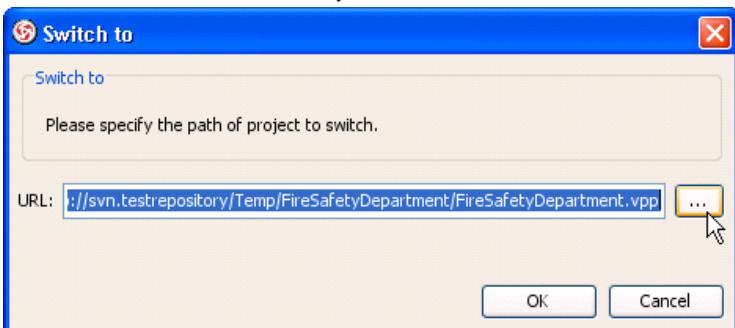


Figure 2-70 Select branch

3. Select branch path **branches/user1_changes/FireSafetyDepartment/FireSafetyDepartment.vpp**.

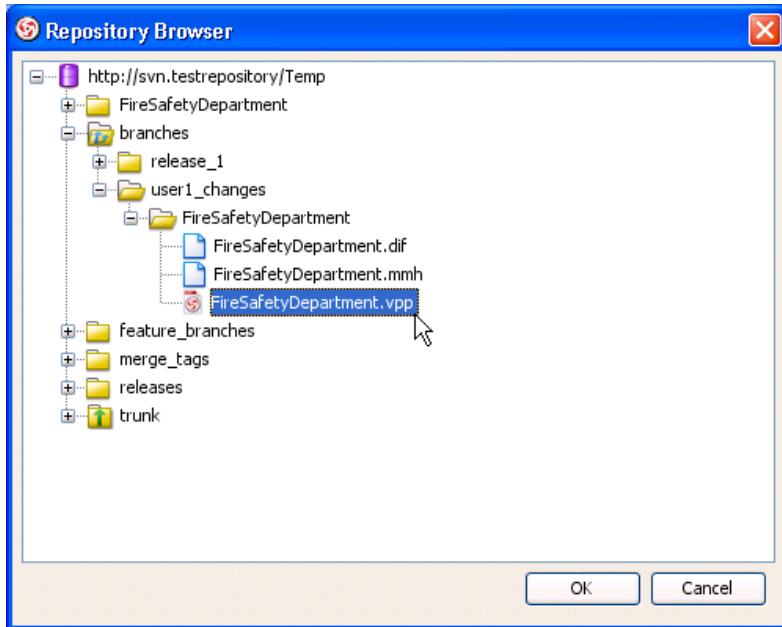


Figure 2-71 Select branch path

4. Press **OK** to open.
 5. The branch is opened in **VP-UML**.

Merging from trunk to branch

1. Open, modify and commit trunk project.

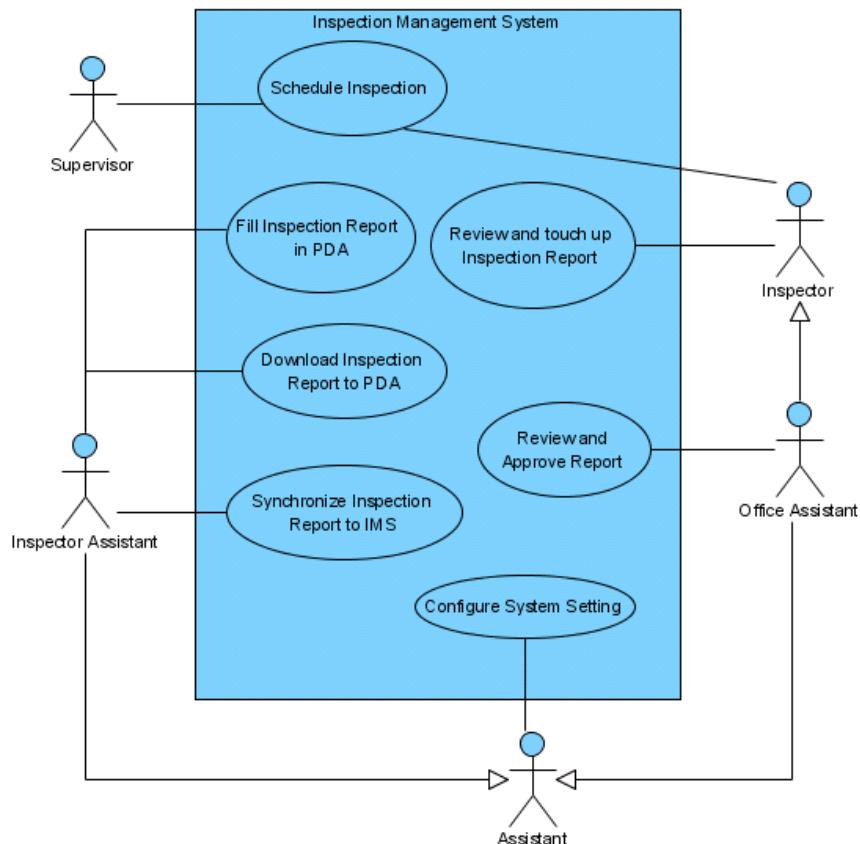


Figure 2-72 Modified trunk project

2. Open branch project.

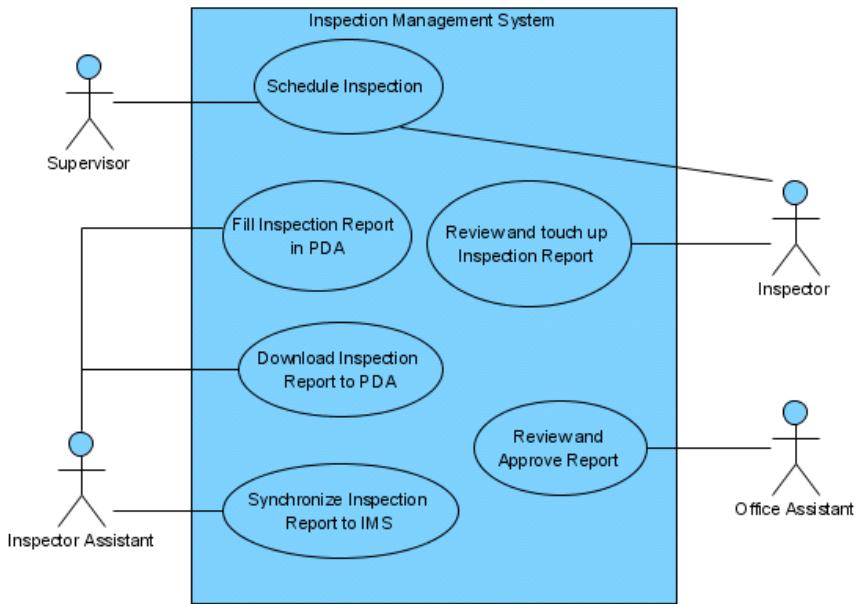


Figure 2-73 Branch project

3. Select **Project > Merge...** from menu.

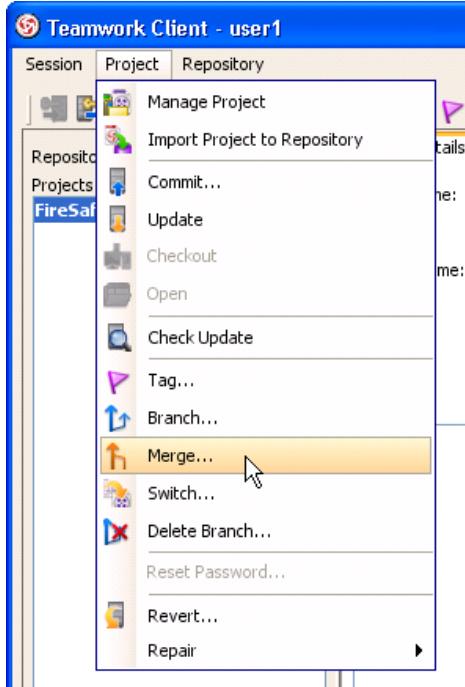


Figure 2-74 Merge from menu

Or click **Merge...** button from toolbar.



Figure 2-75 Merge from toolbar

4. Select the trunk path ([http://svn.testrepository\(Temp/FireSafetyDepartment/FireSafetyDepartment.vpp](http://svn.testrepository(Temp/FireSafetyDepartment/FireSafetyDepartment.vpp)) as **From** in the **Merge** dialog, and press **OK**.

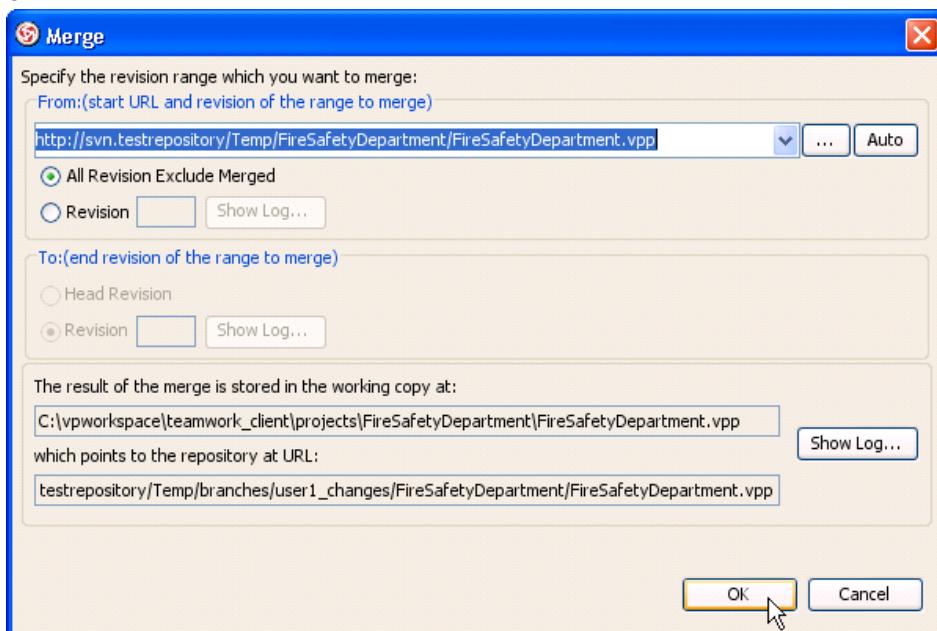


Figure 2-76 Merge dialog

5. The **Merge Project** dialog show a list of changes similar to **Commit** and **Update** dialog.

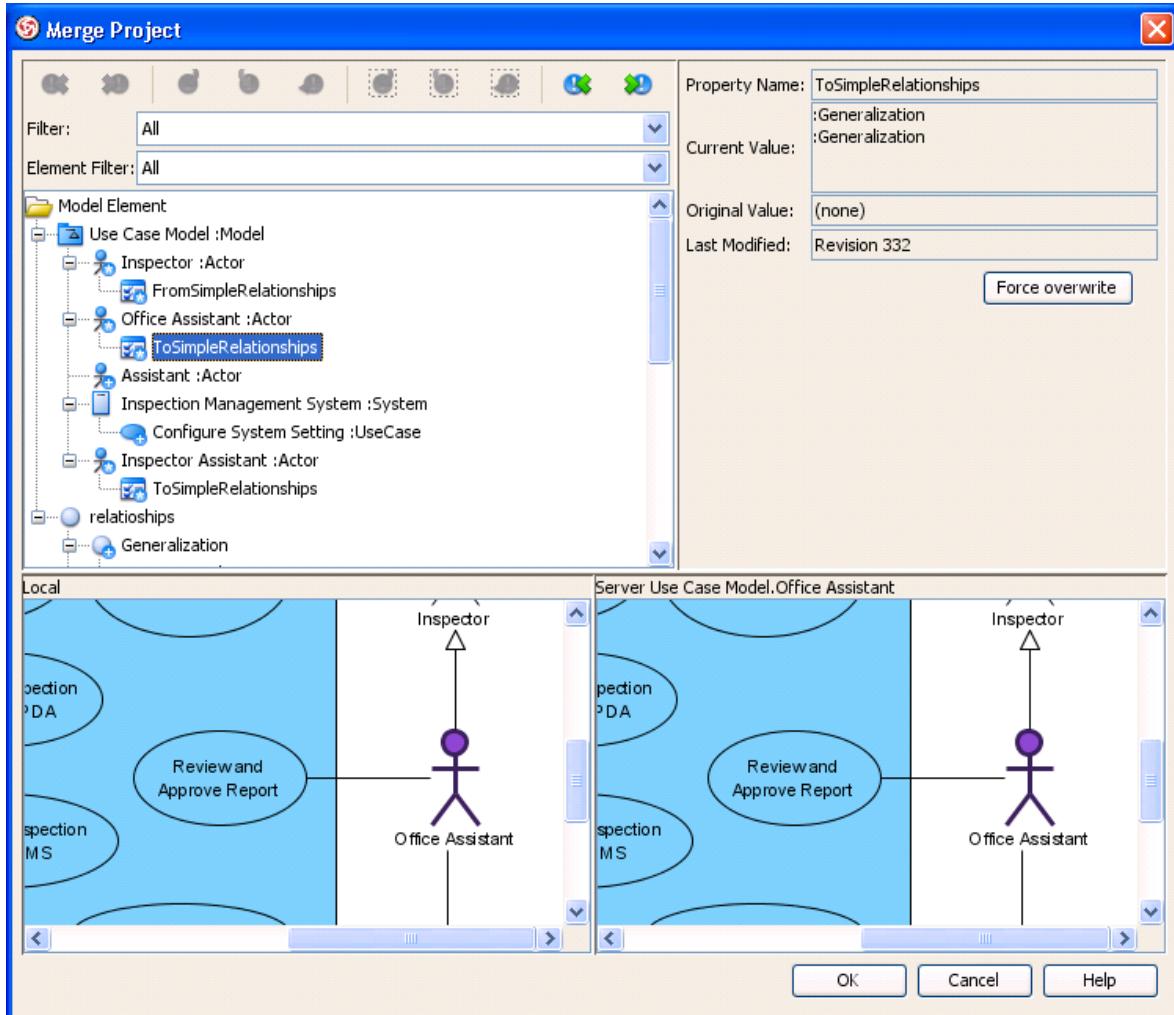


Figure 2-77 Merge project dialog

6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.

7. The branch project now contains the changes from trunk.

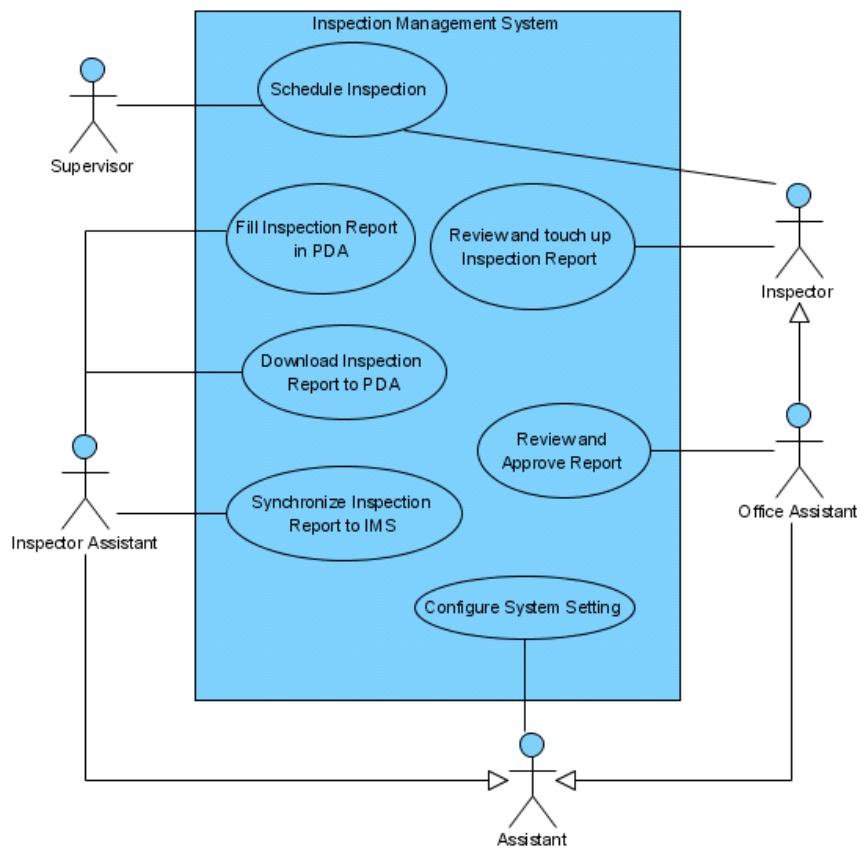


Figure 2-78 Branch project merged from trunk

Merging from branch to trunk

1. Open, modify and commit branch project.

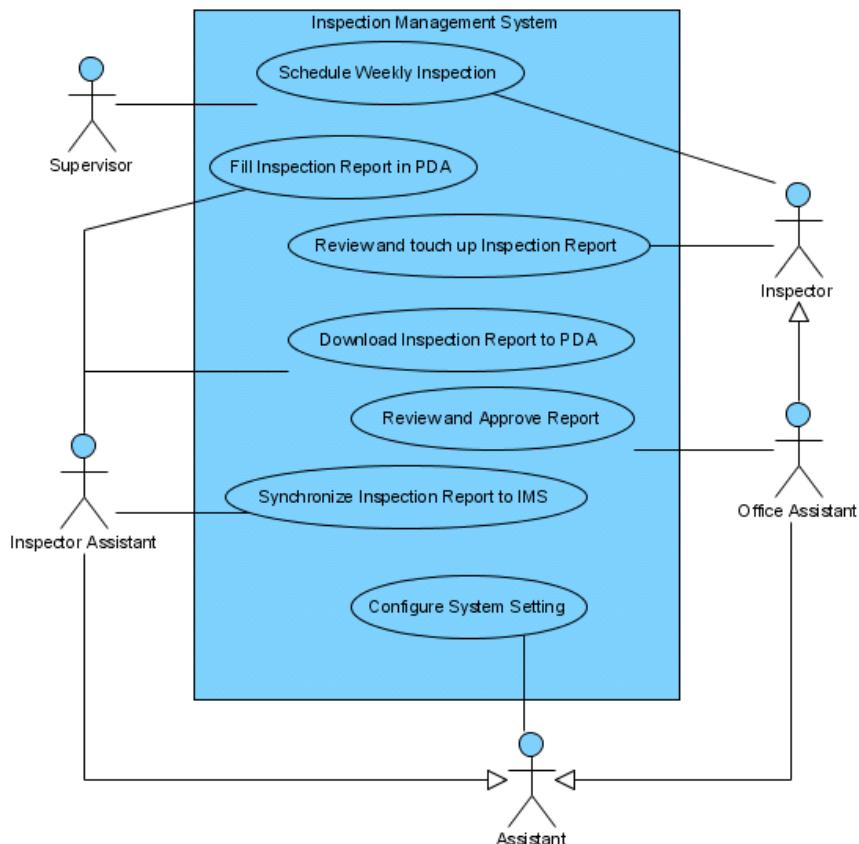


Figure 2-79 Modified branch project

2. Open trunk project.

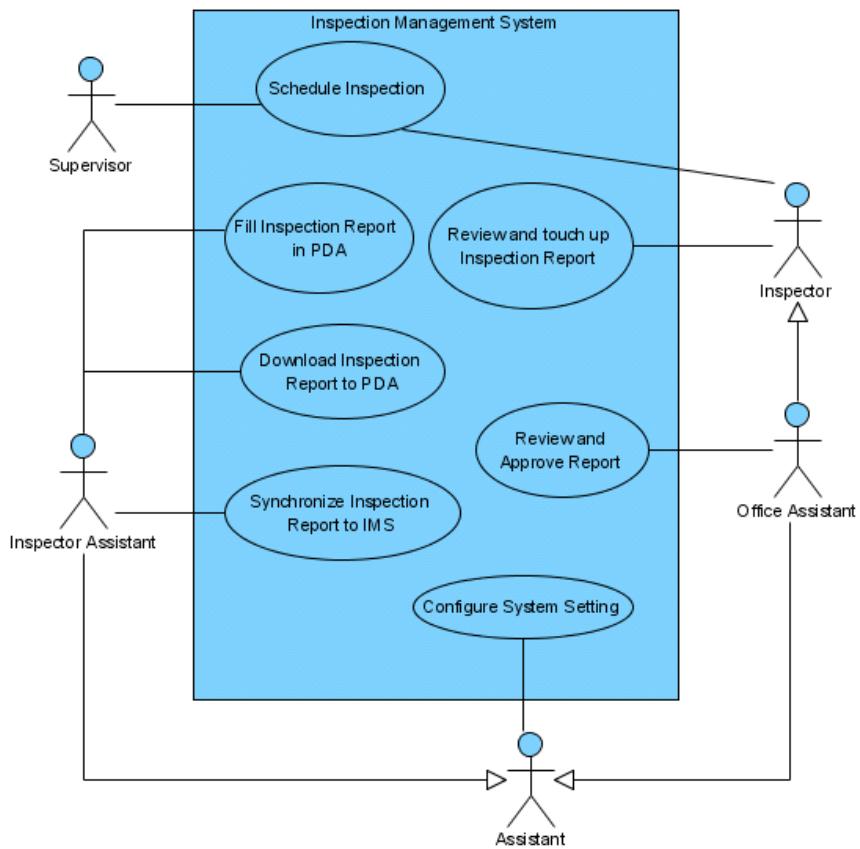


Figure 2-80 Trunk project

3. Select **Project > Merge...** from menu, or click **Merge...** button from toolbar.
4. Select the branch path([http://svn.testrepository\(Temp/branches/user1_changes/FireSafetyDepartment/FireSafetyDepartment.vpp](http://svn.testrepository(Temp/branches/user1_changes/FireSafetyDepartment/FireSafetyDepartment.vpp)) as **From** and press **OK**.
5. The **Merge Project** dialog show a list of changes similar to **Commit** and **Update** dialog.
6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.

7. The trunk project now contains the changes from branch.

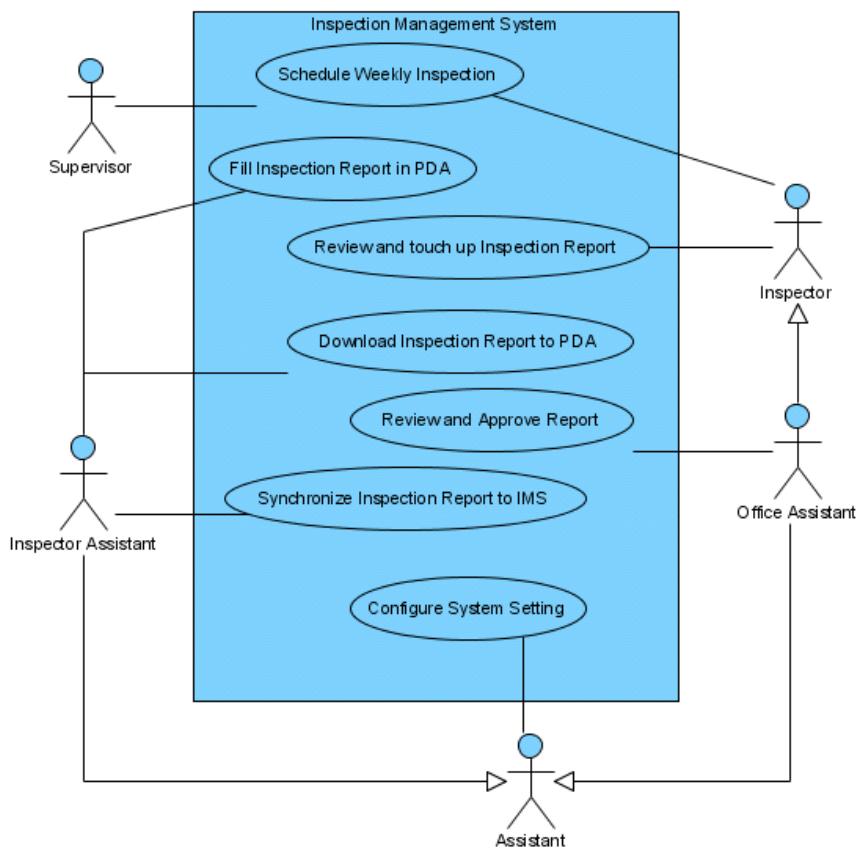


Figure 2-81 Trunk project merged from branch

Delete branch

1. Select Project > Delete Branch... from menu

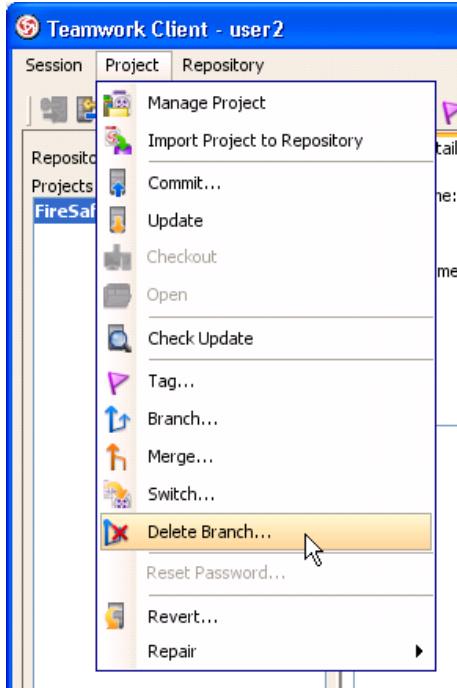


Figure 2-82 Delete Branch from menu

Or click Delete Branch ... button from toolbar.



Figure 2-83 Delete Branch from toolbar

2. Expand the repository and folder node, select the branch to delete. Click OK button to continue.

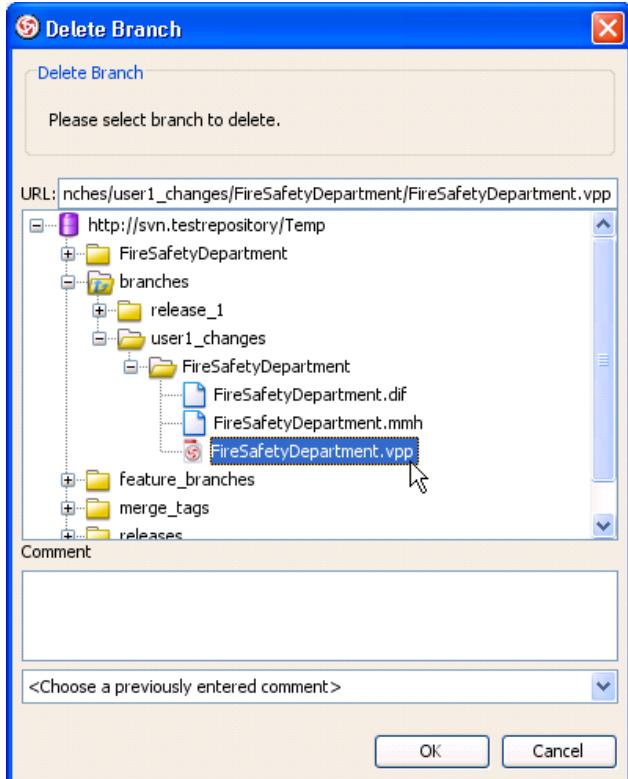


Figure 2-84 Delete branch dialog

3. Click Yes button on confirmation dialog.

Marking release or milestone with tags

Tags and branches are almost the same, with the only difference - tags are read only. Creating read only tags ensure you are able open the release version project later.

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Tag...** from menu

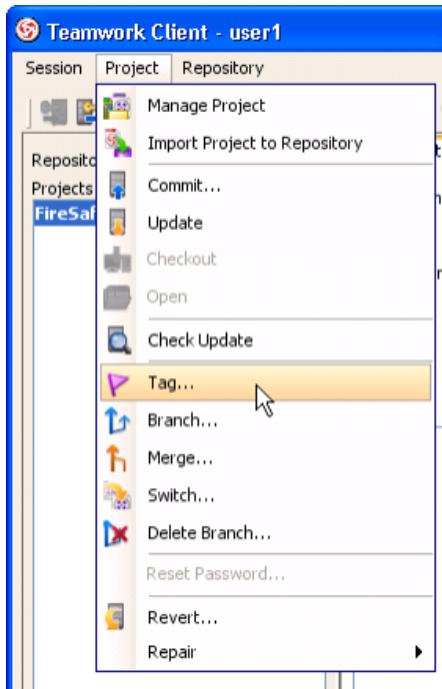


Figure 2-85 Tag from menu

Or click **Branch...** button from toolbar.



Figure 2-86 Tag from toolbar

4. Fill in the Tag Name in **Create Tag** dialog.

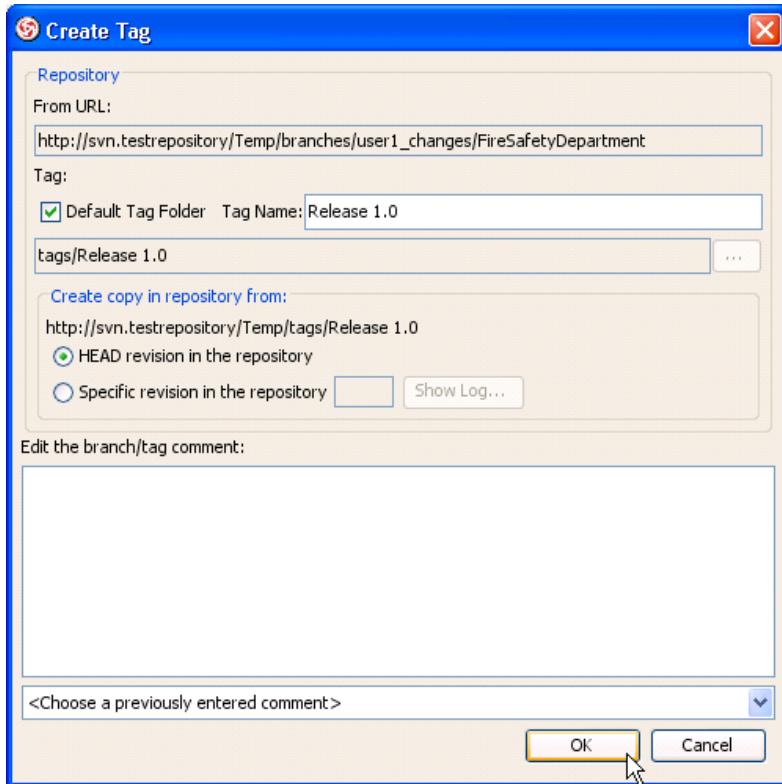


Figure 2-87 Create Tag dialog

5. Click **OK** button, the tag will be created.

6. If the path doesn't exists, it will prompt for ask create.



Figure 2-88 Create folder warning

Working with teamwork client dialog box

To perform teamwork operations such as to commit or update project, you can click the buttons in application menus or toolbars. Besides, you can perform via the Teamwork Client dialog box. Teamwork Client dialog box enables you to perform teamwork operations, manage teamwork projects, as well to review and checkout revisions of particular project.

Opening the teamwork client

1. Open the Teamwork Client by either of the ways below:
 - Select **Tools > Teamwork > Open Teamwork Client...** from the main menu.

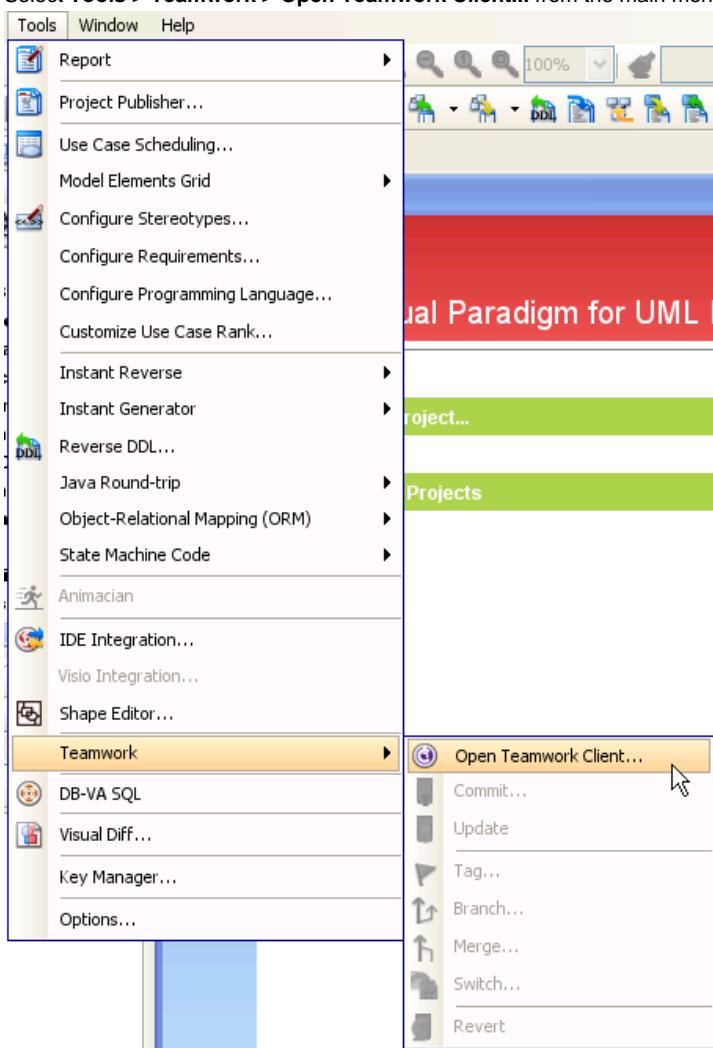


Figure 3-1 Open teamwork client through menu

- Click on the button **Open Teamwork Client** in toolbar.

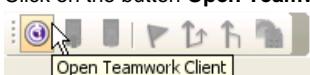


Figure 3-2 Open teamwork client through toolbar

2. In the login dialog box, fill in the server and user information to login to server.

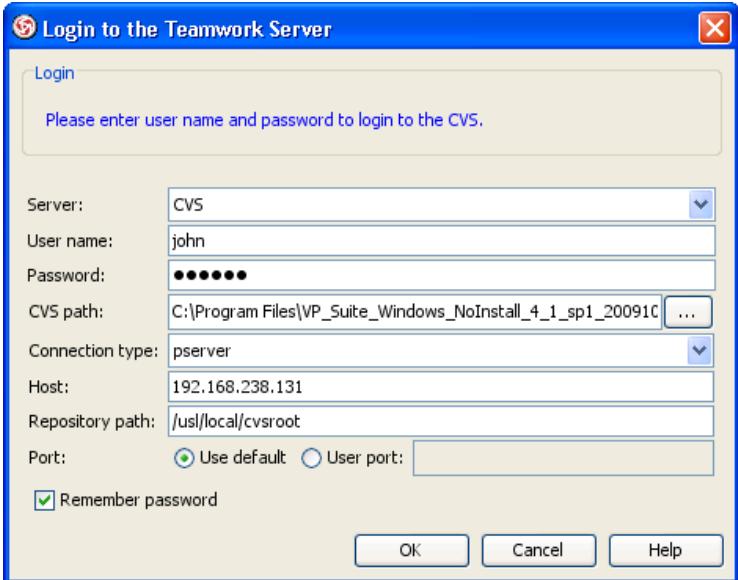


Figure 3-3 Login CVS

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be CVS
Server host	The server host, which can be host name or IP address
Port number	The port of connecting to server
User name	The user name for connecting to server
Password	The password for given user
Remember password	Check to memorize the password so that you don't need to enter again
Use proxy	Check if proxy is needed for connecting to server. When checked, you will need to enter proxy host.

Table 3-1 Description of Login dialog box

3. If this is the first time you connect to the CVS server, you will be asked to select the projects to manage. By managing a project, you can check it out, open and edit it. In the **Manage Project** dialog box, select the projects to manage.

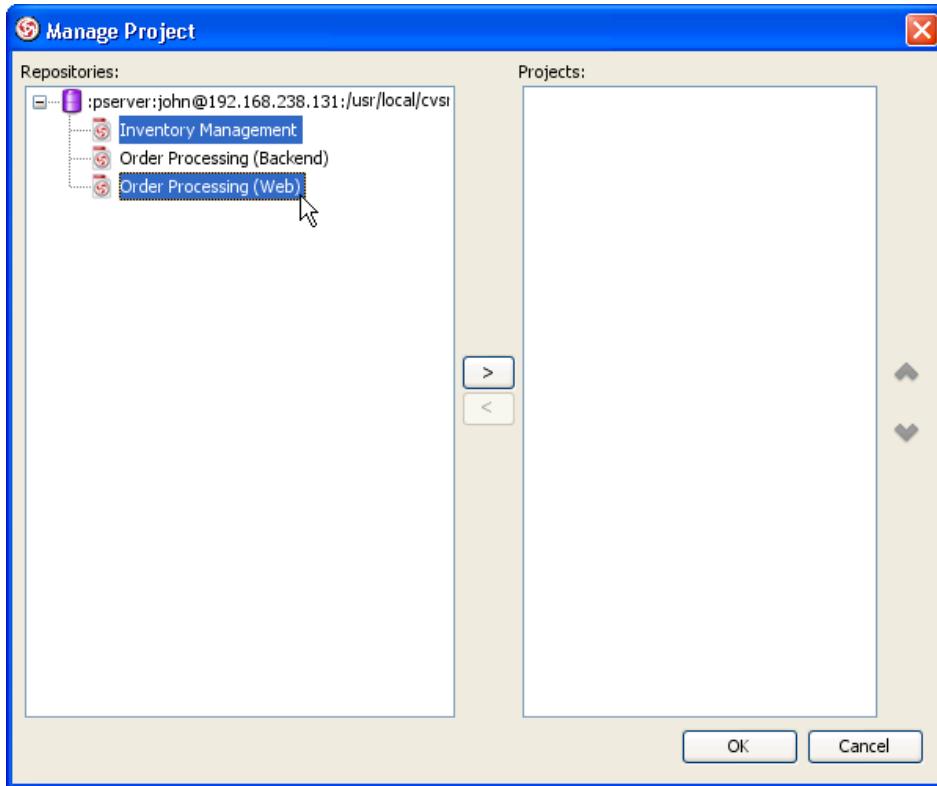


Figure 3-4 Select projects to manage

4. Click on the **>** button in the middle of the dialog box to add them into project selection.
 5. Click **OK** to proceed. This end up showing the **Teamwork Client** dialog box.

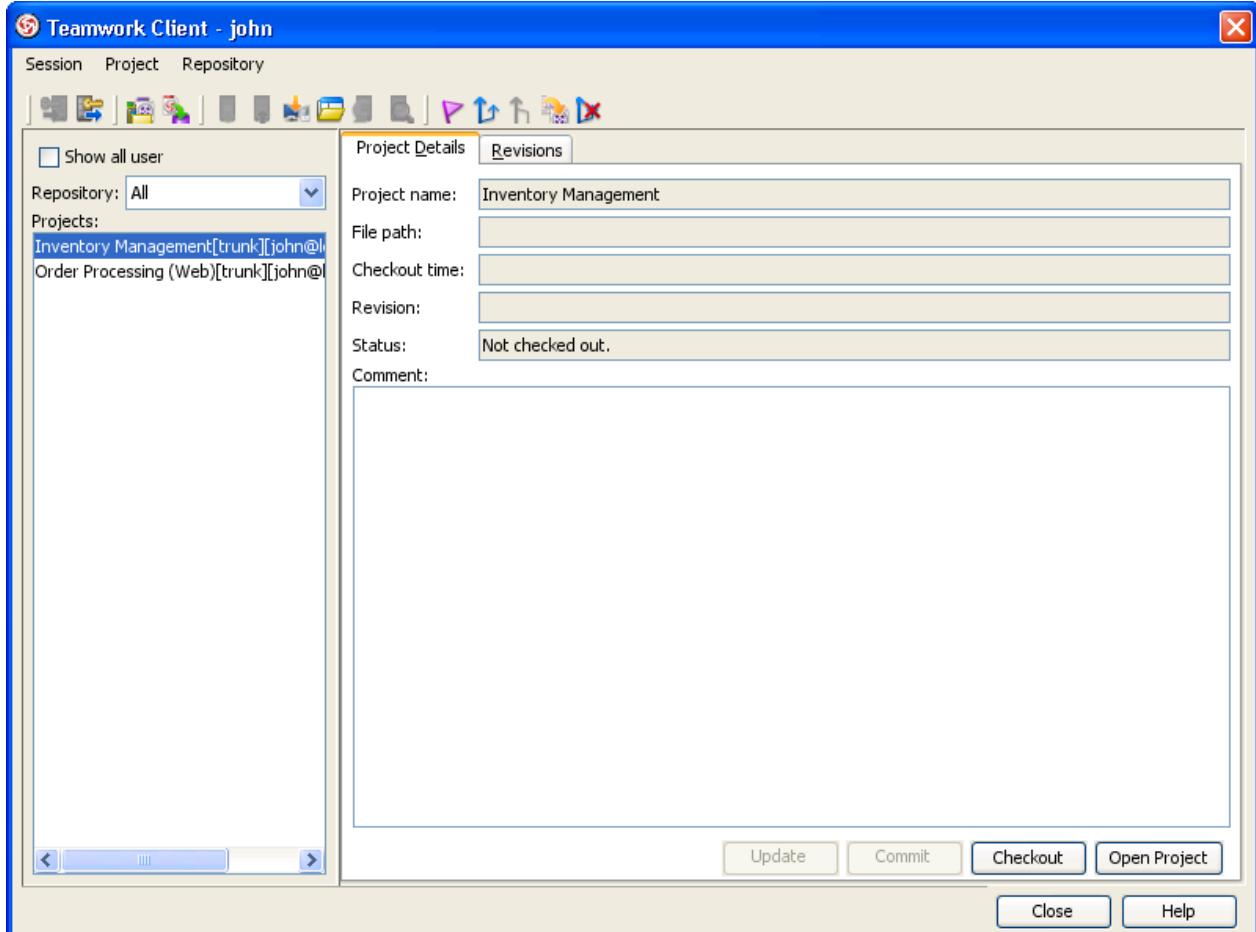


Figure 3-5 The Teamwork Client dialog box

The interface

General

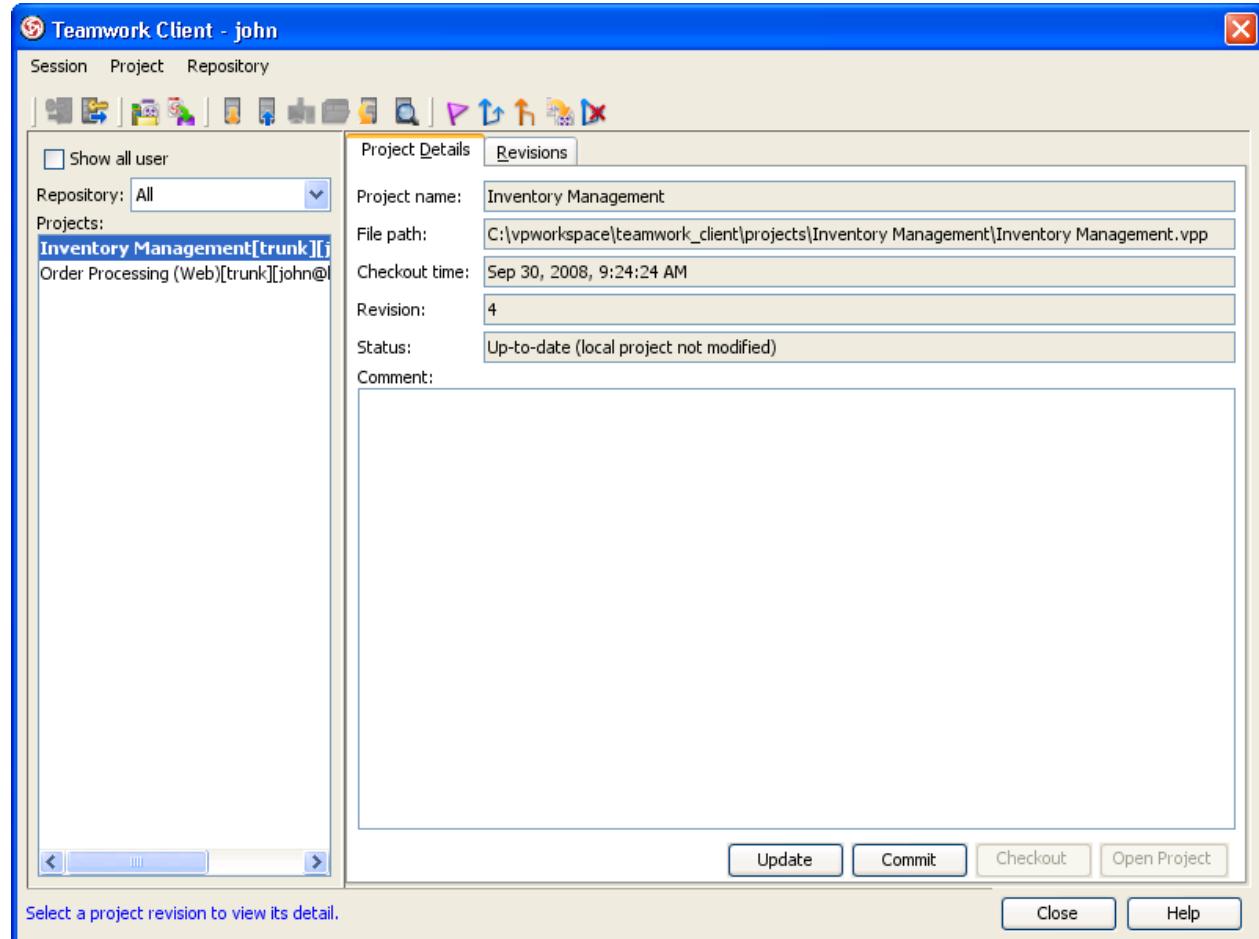


Figure 3-6 The Teamwork Client dialog box

Region	Description
	To logout from CVS server.
	To login to CVS server.
	To manage projects, which enables you to add or remove projects from the list of managed projects.
	To create a project in server by importing one into it. You may create a blank new project, or import an existing project file to start with.
	To update changes from server.
	To commit changes to server.
	To checkout a project from server.
	To open the project selected in Projects list. If the project is not checked out yet, it will be checked out automatically.
	To remove changes made in a revision.
	To check for changes
	To create a tag for a trunk/branch.
	To create a branch for a trunk/branch.
	To merge changes from trunk/branch.
	To switch to another trunk/branch/tag.



To remove a branch. Note that removed branch will not delete the, but just make it not accessible.

Show all user To show the projects manageable by all users.

Repository To update the **Projects** list by showing only projects in certain repository.

Projects The list of manageable project.

Project Details Details of selected project, including the name, revision and status.

Revisions Revisions of selected project.

Table 3-2 Description of the Teamwork Client dialog box

Project details

The screenshot shows the 'Project Details' tab selected in a dialog box. The form contains the following fields:

Project name:	Inventory Management
File path:	C:\vpworkspace\teamwork_client\projects\Inventory Management\Inventory Management.vpp
Checkout time:	Sep 30, 2008, 9:24:24 AM
Revision:	4
Status:	Up-to-date (local project not modified)
Comment:	(Large text area)

At the bottom are four buttons: Update, Commit, Checkout, and Open Project.

Figure 3-7 The Project Details tab in Teamwork Client dialog box

Field	Description
Project name	Name of the selected project.
File path	The file path of the project in your system.
Checkout time	The time when you checkout the project.
Revision	The revision of your local copy.
Status	The status of local project.
Comment	The comment of project entered by the administrator when creating project.

Table 3-3 Description of Project Details tab in Teamwork Client dialog box

Revisions

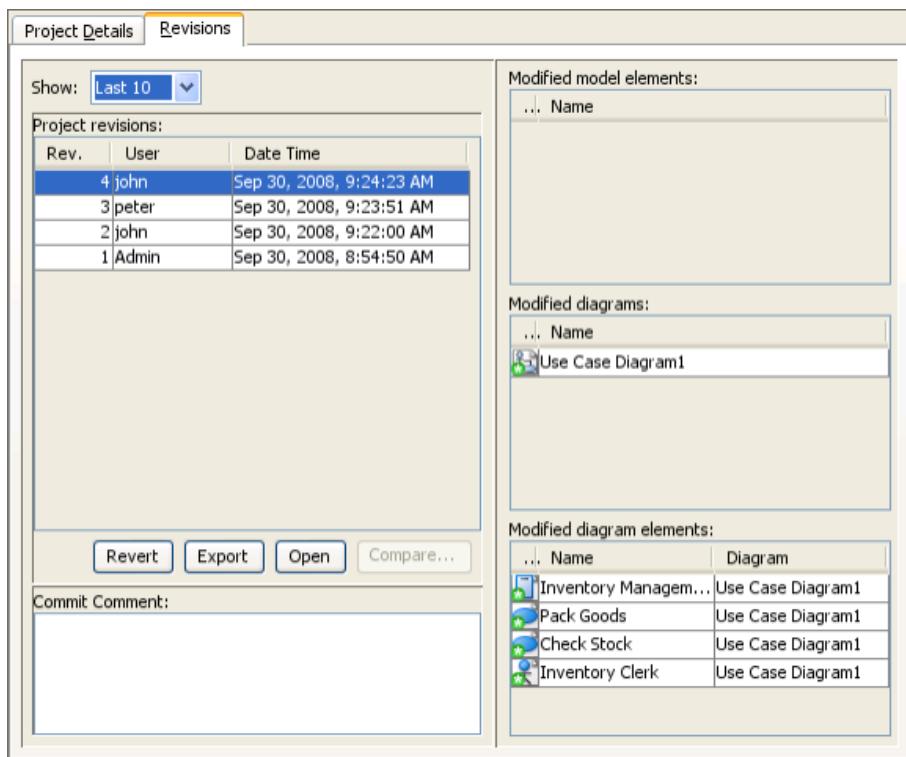


Figure 3-8 The Revisions tab in Teamwork Client dialog box

Region	Description
Show:	To list certain amount of revisions, such as all, last 10, last 20, etc.
Revision list	To display the revisions of chosen project.
[Revert]	By reverting a selected revision, this means to undone changes made in that revision.
[Export]	To export selected revision(s) as project files.
[Open]	To open selected revision.
[Compare]	To compare the differences between revisions.

Table 3-4 Description of Revisions tab in Teamwork Client dialog box

Importing projects to CVS server

1. Start VP-UML.
2. Select **Tools > Teamwork > Open Teamwork Client...** from the main menu, or **Open Teamwork Client** icon from toolbar to open teamwork client.

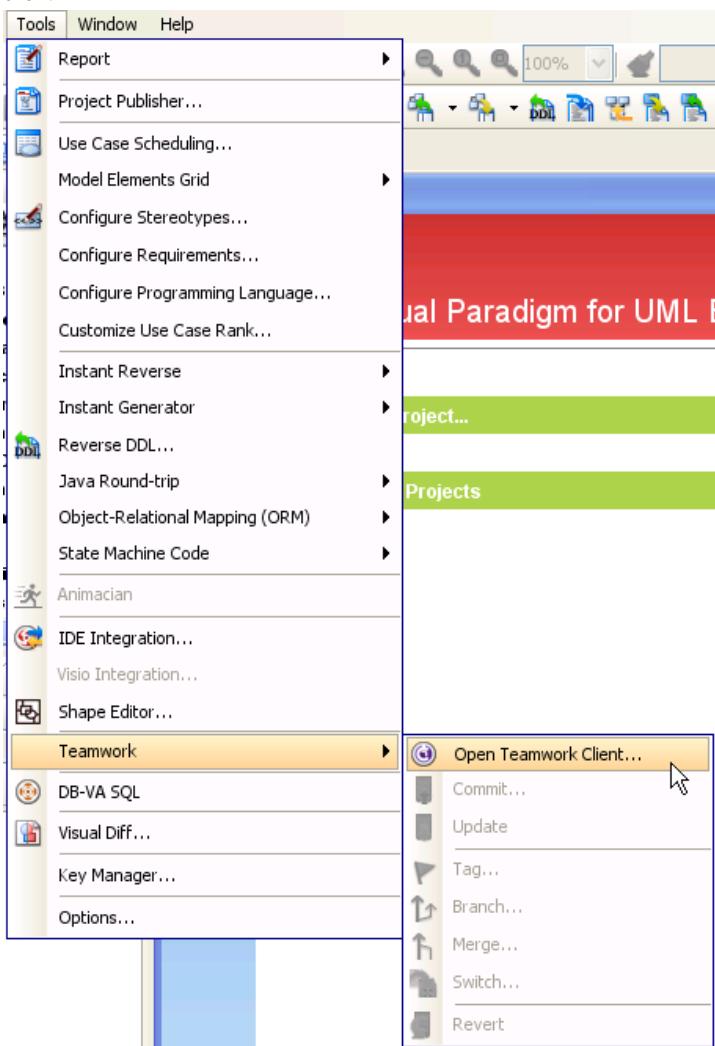


Figure 3-9 Open Teamwork Client from main menu

3. Select CVS Server, fill in the server and user information to login cvs server.

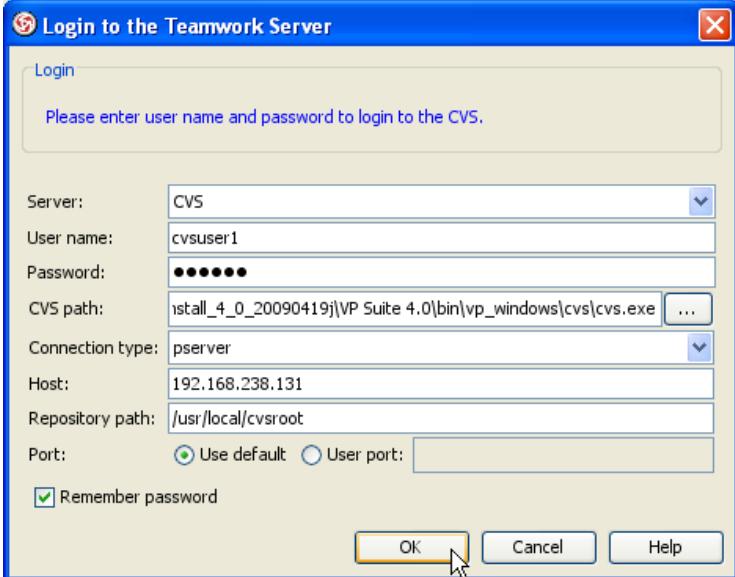


Figure 3-10 Login cvs server

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be CVS
User name	The user name for connecting to server
Password	The password for given user
CVS path	The filepath of cvs.exe. By default, the one in installation folder will be picked up
Connection type	The type of connection with CVS server, which can be pserver, ext, ssh
Host	The host name of server
Repository path	The path of CVS repository
Port	The port of connecting to CVS server. To specify one, check User port and enter the port number
Remember password	Check to memorize the password so that you don't need to enter again

Table 3-5 Description of Login dialog box

4. Select **Project > Import Project to Repository** from the menu.

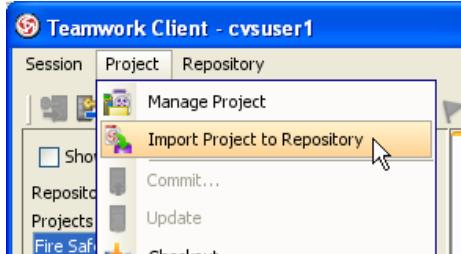


Figure 3-11 Import project from menu

Or click the **Import Project to Repository** button on toolbar.



Figure 3-12 Import project from toolbar

5. In the **Import Project** dialog, fill in the **Project name**.

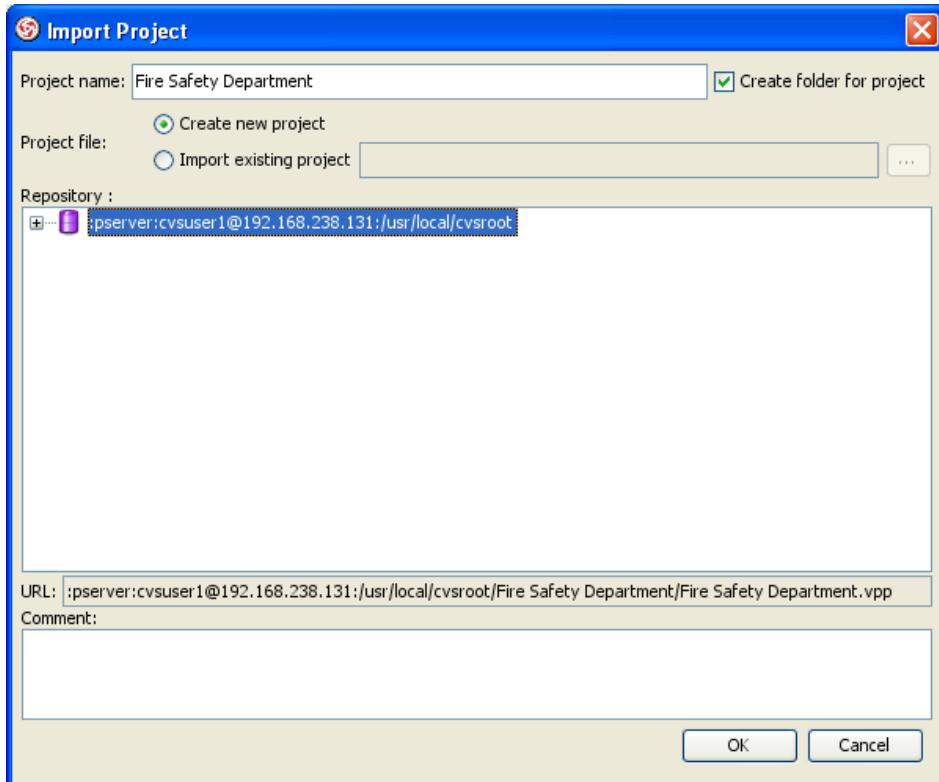


Figure 3-13 Fill project name in Import Project dialog

Field	Description
Project name	The name of teamwork project
Create folder for project	Create a folder in repository for storing the project. The folder will be of same name as the project.
Project file	Create new project - Start with a blank, new project Import existing project - Start by using an existing project file
Repository	A tree which shows the structure of repository
URL	The URL of the project in repository
Comment	The comment of project

Table 3-6 Description of Import Project dialog box

6. Choose **Create new project** or **Import existing project** from **Project file**. **Create new project** will import a blank project into teamwork server, **Import existing project** will import an existing VP-UML Project file (*.vpp) into teamwork server as first revision.



Figure 3-14 Specify project file

NOTE: Please make sure the user login to **Teamwork Client** has **Create Project** permission

7. Select the repository for import the project.

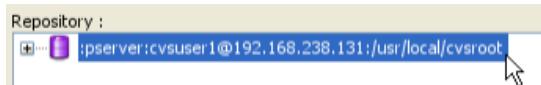


Figure 3-15 Select repository

8. Fill in the comment.

Comment:
Importing project

Figure 3-16 Input comment

9. Click **OK** button to import project.

Checkout project from CVS server

Checkout and open CVS server projects

1. Open Teamwork Client dialog.
2. Select Project > Manage Project from the menu.

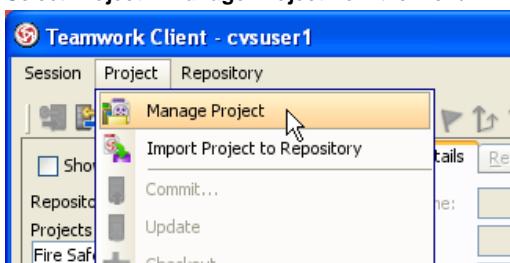


Figure 3-17 Manage project from main menu

Or click **Manage Project** icon on the toolbar.

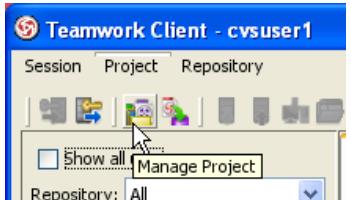


Figure 3-18 Manage project from toolbar

3. In **Manage Project** dialog, expand the repository node and select the project to checkout, click > button to add project. Click **OK** button to close the dialog.

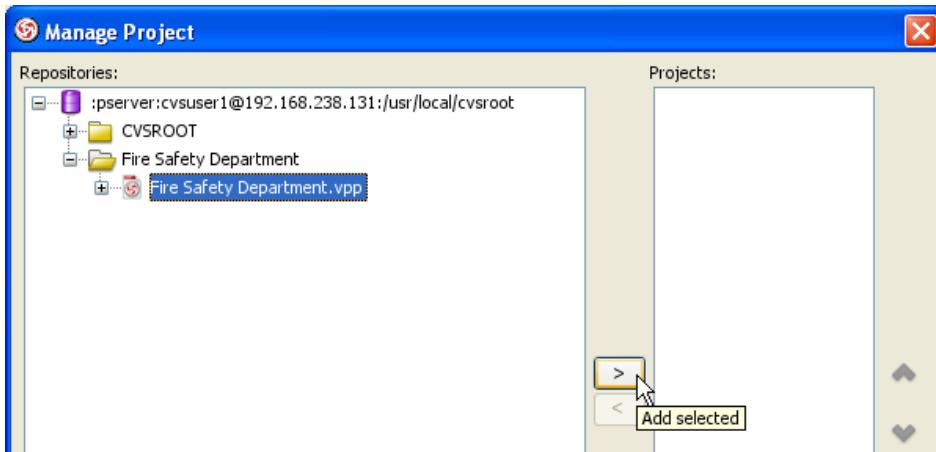


Figure 3-19 Manage projects

4. Select the project in project list.

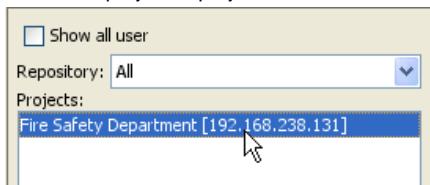


Figure 3-20 Select project from the list

5. Click **Open Project** button to checkout and open the project.

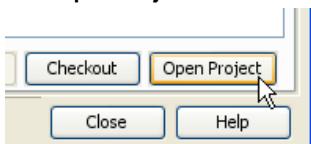


Figure 3-21 Open project

6. The project opened in VP-UML.

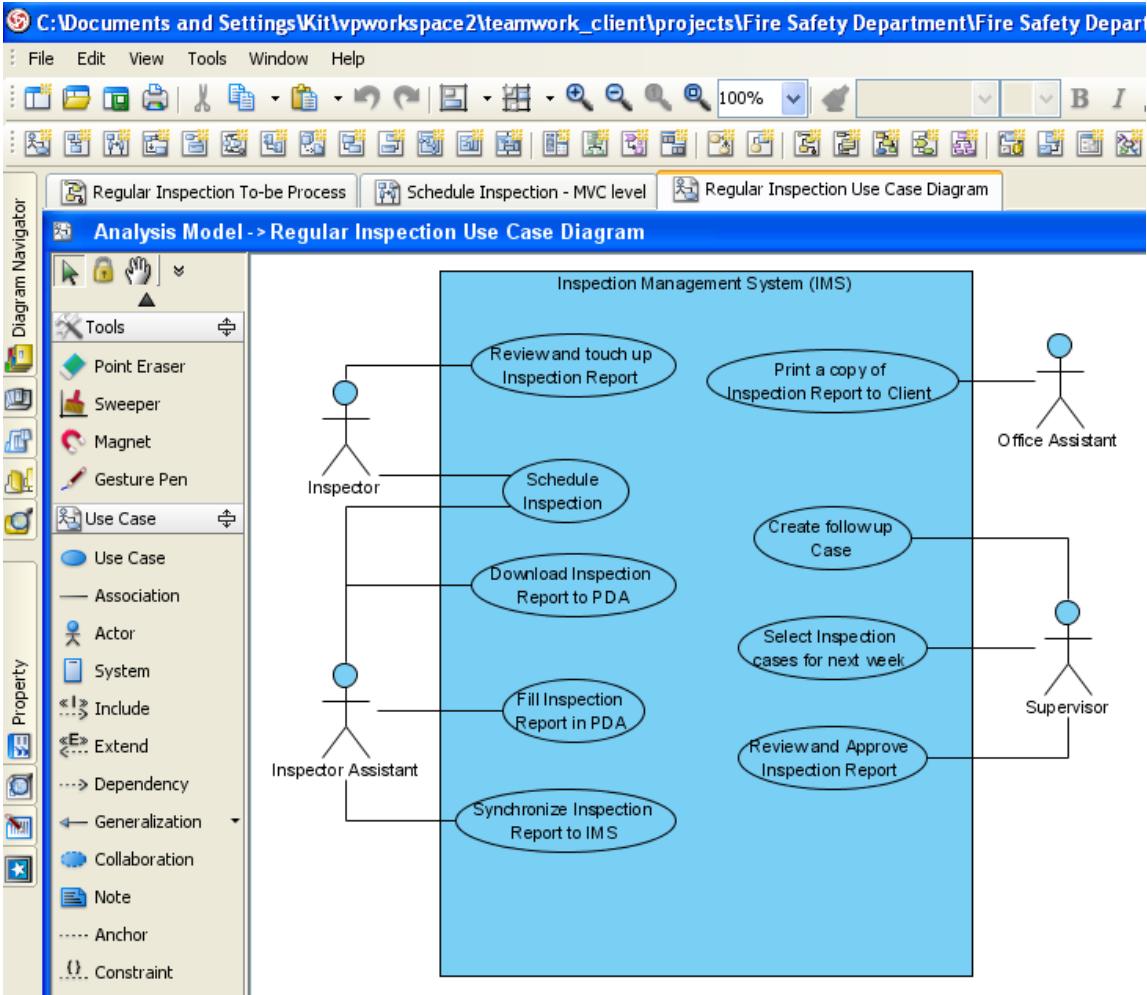


Figure 3-22 Project opened

Committing local modification to CVS server

1. Modify the project.

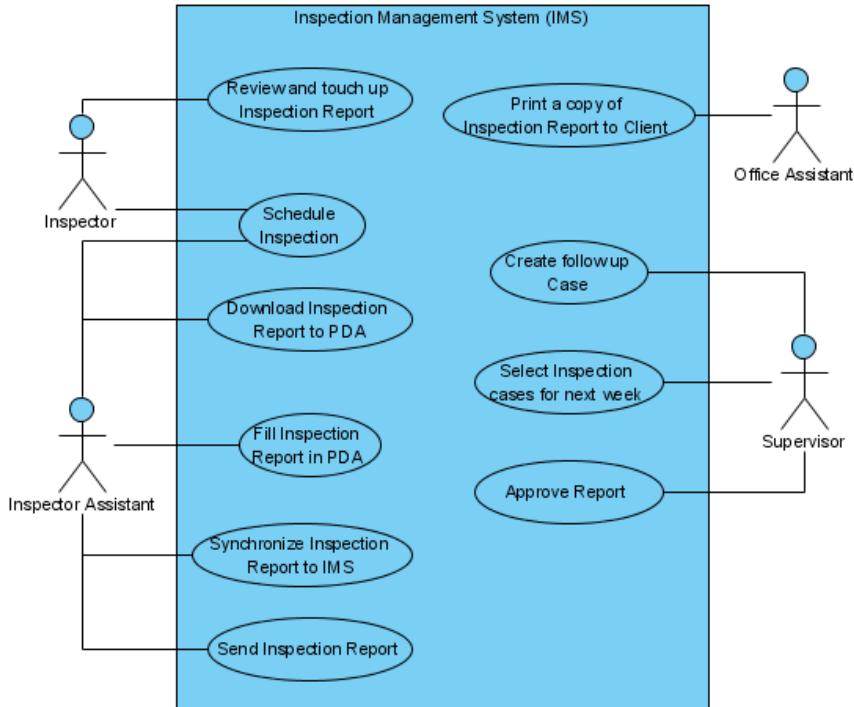


Figure 3-23 Modified project

2. Show **Teamwork Toolbar** by right click on the Toolbar, select **Teamwork** if not selected already.

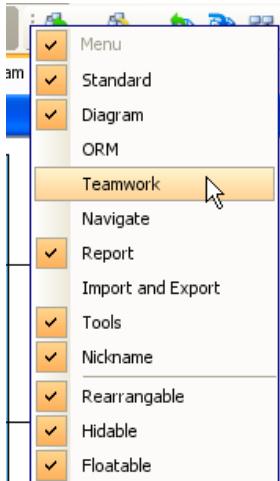


Figure 3-24 Show teamwork toolbar

3. Click the **Commit** button on **Teamwork Toolbar**.



Figure 3-25 Commit project

4. In **Commit** dialog, you can review the changes for commit. On the left of dialog, you can see a list of changes shapes and model elements, click on it to view the detail changes on the right.

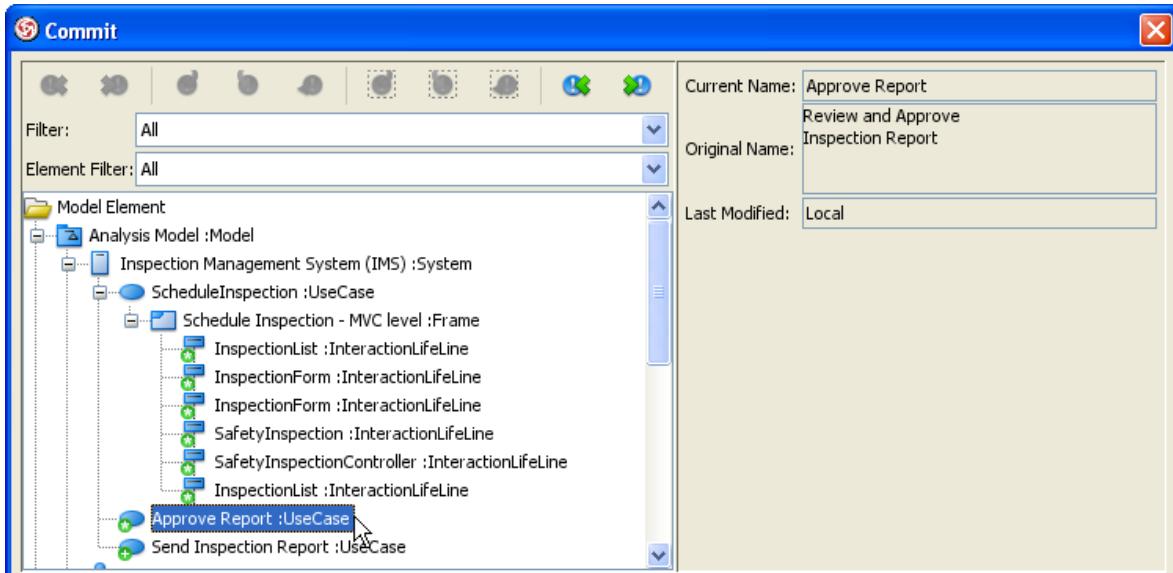


Figure 3-26 Review commit changes

5. On the bottom of dialog, click the **Preview** tab to visually preview the changes in diagram. The selected shape is highlighted in purple.

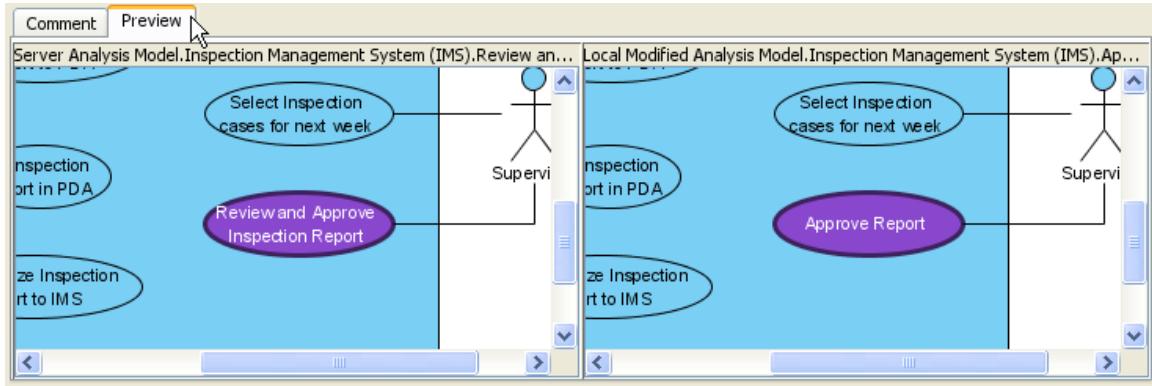


Figure 3-27 Preview commit changes

6. After review the changes, click the **Comment** tab and input the comment for commit. Click **OK** button to start commit.

Project name:	Fire Safety Department	Checkout time:	29 Apr 2009, 04:32 PM
Checkout revision:	1	Latest server revision:	1
Edit the commit comment: Modify Use Case Diagram			
<Choose a previously entered comment>			

Figure 3-28 Input commit comment

Resolving conflicts

1. Modify the project.

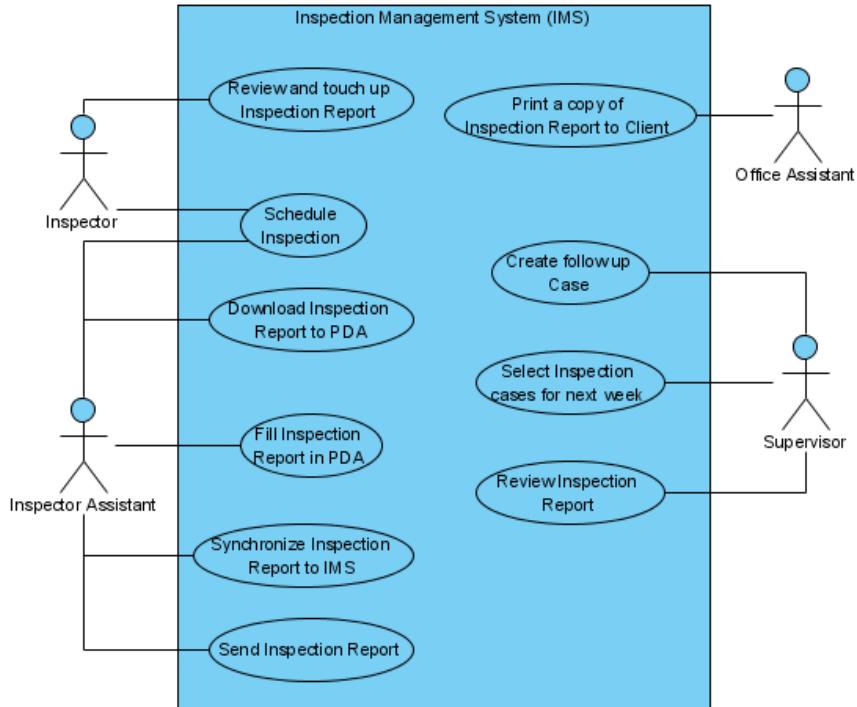


Figure 3-29 Modified project

2. Click **Commit** button on the **Teamwork Toolbar**.

3. In **Commit** dialog, you may found some shapes or model elements show with red icon. This indicate there is conflict when commit, it is caused by someone modified the same content and commit after you checkout. You can review the current value, original value (the value when you checkout), and conflict value (the value changed by other users). And dialog preview is disabled until you resolve the conflict.

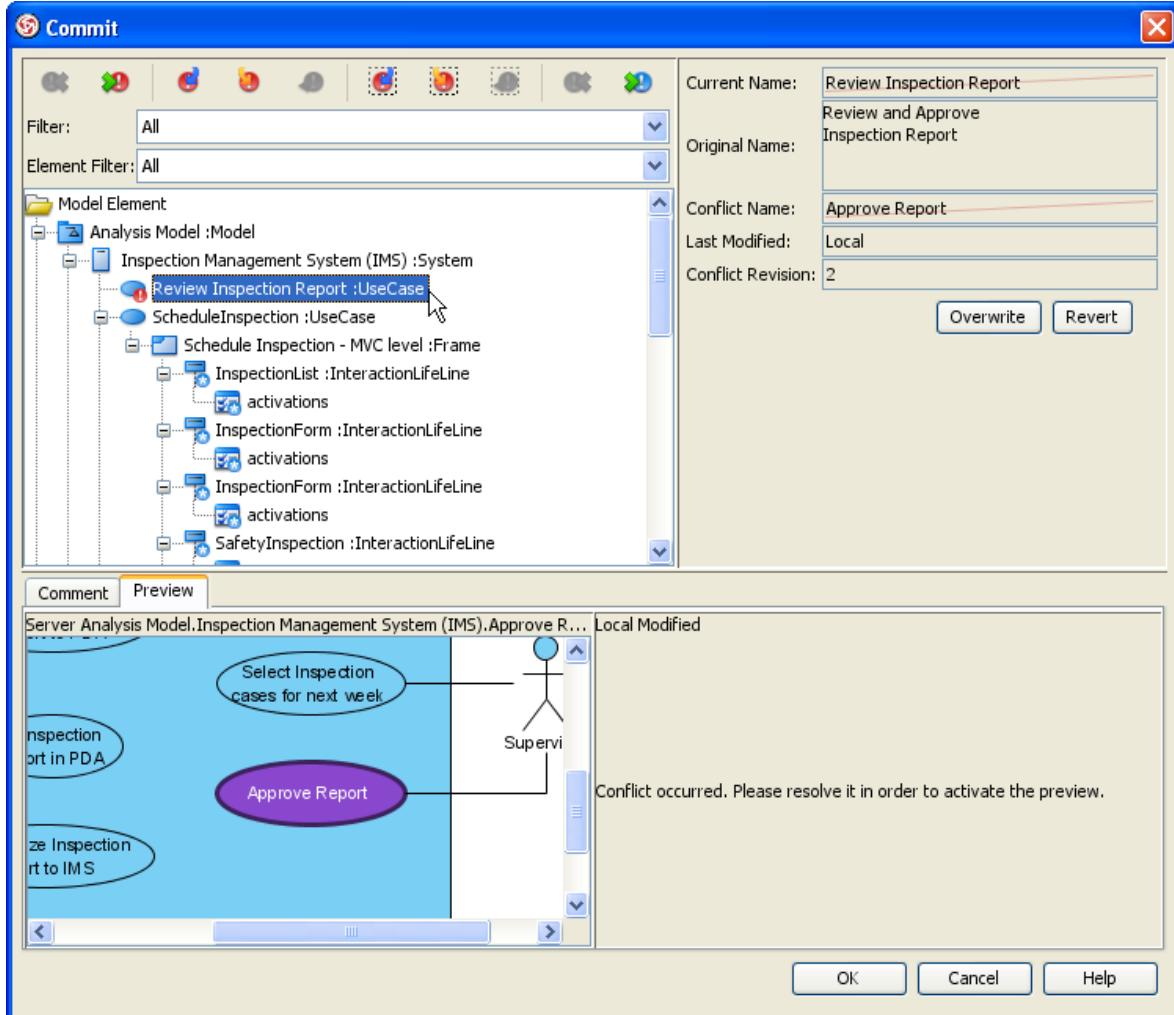


Figure 3-30 Commit with conflicts

Below is a description of buttons or fields in dialog box.

Button/Field	Description
	Move to the previous conflict in element tree.
	Move to the next conflict in element tree.
	Overwrite all conflicts.
	Revert all conflicts.
	Reset all conflicts.
	Overwrite the selected conflict in tree.
	Revert the selected conflict in tree.
	Reset the selected conflict in tree.
	Select the previous change in tree.
	Select the next change in tree.
Current Name	The value of name of the committing/updating copy.
Original Name	The value of name of the original copy.
Conflict Name	The value of name of the server copy.
Last Modified	The user who modified last time.
Conflict Revision	The revision that cause conflict with the committing/updating action.
[Overwrite]	Click on overwrite the conflict selected in tree.
[Revert]	Click on revert the conflict selected in tree.

4. Select the conflict element, click **Overwrite selected conflicts** button on the toolbar to overwrite other user's change.



Figure 3-31 Overwrite selected conflicts

Or click **Revert selected conflicts** to revert your own changes.



Figure 3-32 Revert selected conflicts

You can also click **Overwrite all conflicts** or **Revert all conflicts** to overwrite or revert all conflicts at once.



Figure 3-33 Overwrite/Revert all conflicts

5. After resolve conflict, the preview will be enabled to visualize the final result.

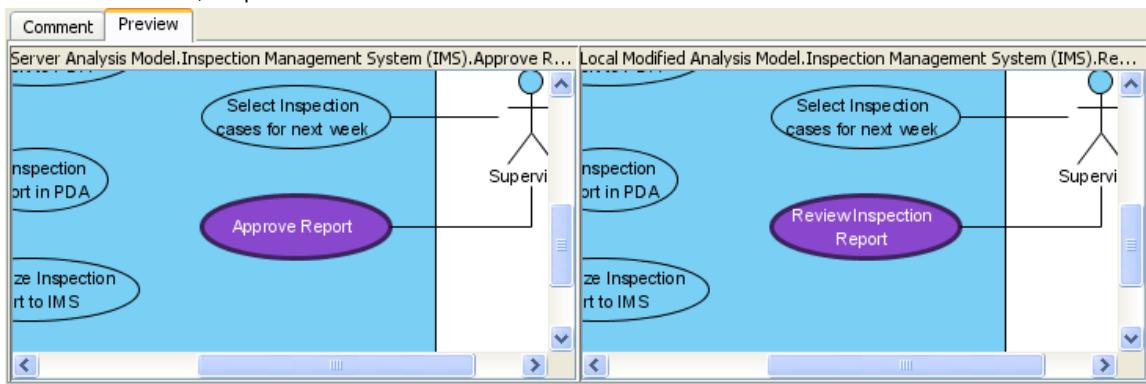


Figure 3-34 Preview resolved conflict

6. You can change your mind by click **Reset selected conflicts** or **Reset all conflicts** button. Then overwrite or revert the conflict again.



Figure 3-35 Reset conflicts

7. After all conflicts was resolved, you can now input comment and click **OK** button to commit.

Updating latest revision from CVS server

Updating modification from CVS server

1. Click **Update** button on **Teamwork Toolbar**.



Figure 3-36 Update from toolbar

2. Similar to commit, you can review the change and preview the diagram in **Update** dialog.

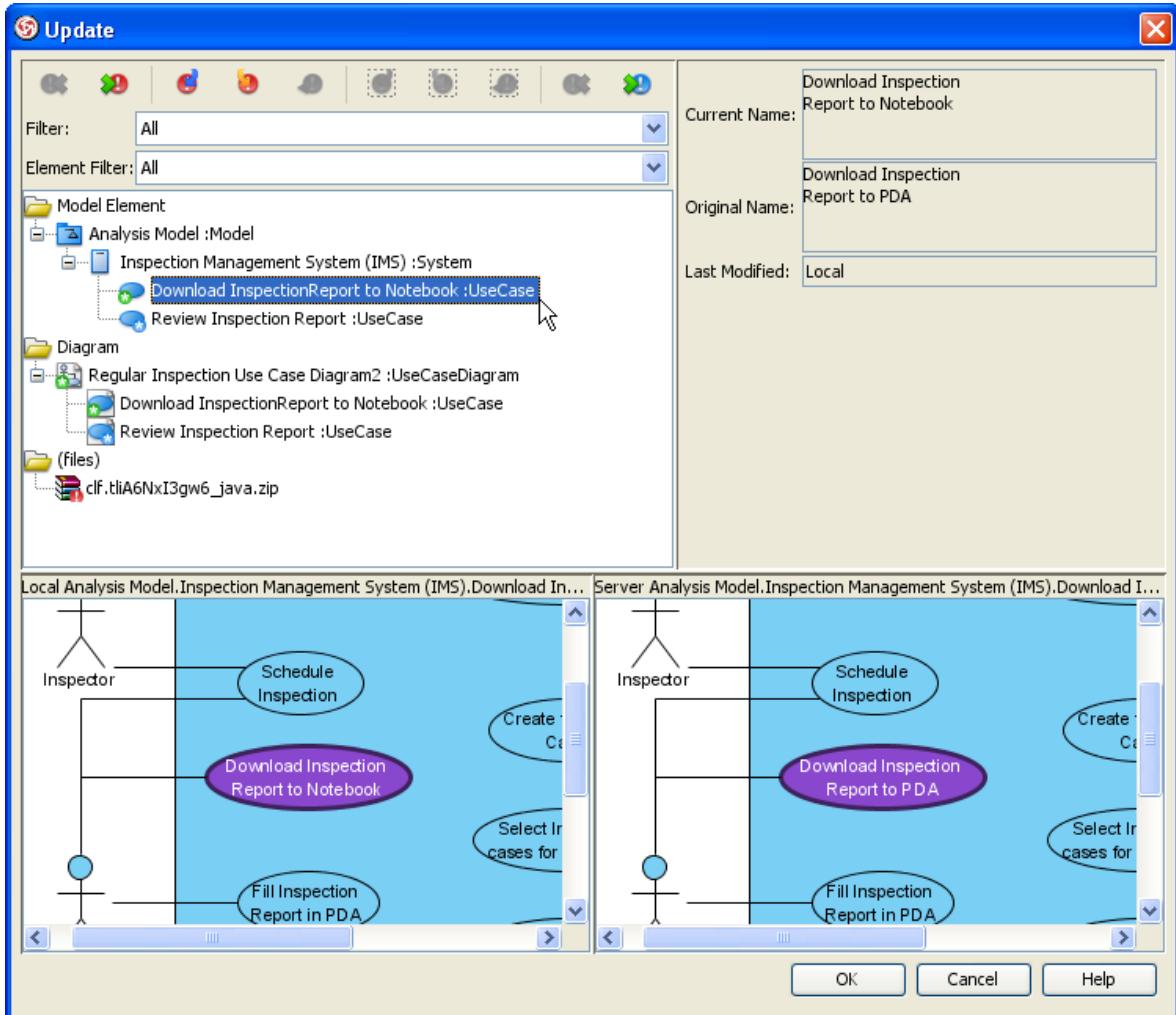


Figure 3-37 Update dialog

3. If there are conflicts, resolve it similar to commit, but the changes will apply to local project instead of commit to server immediately.

4. Click **OK** button to update. **VP-UML** will open the updated project.

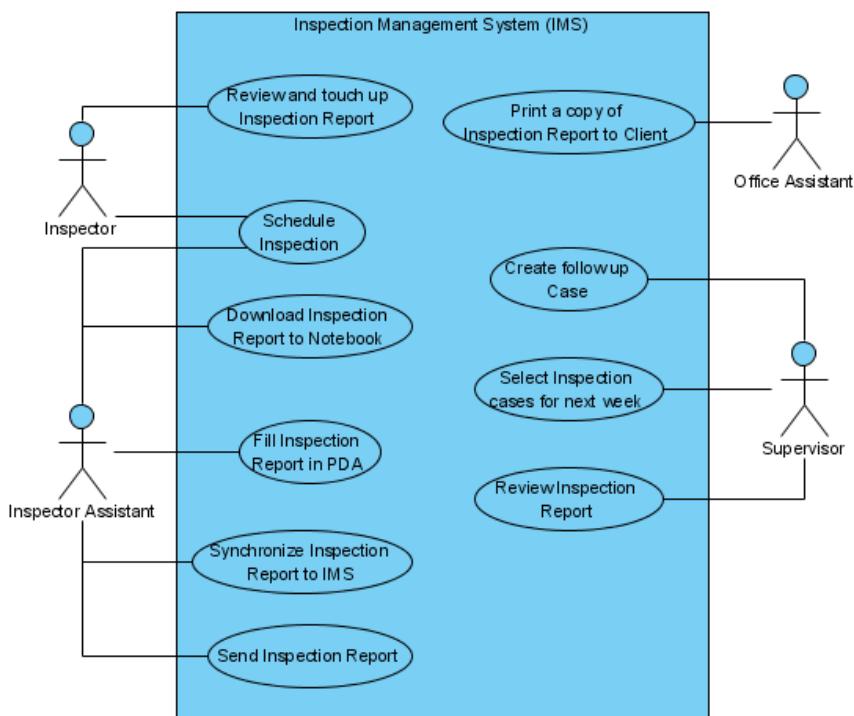


Figure 3-38 Updated project

Checking status

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Check Update** from menu.

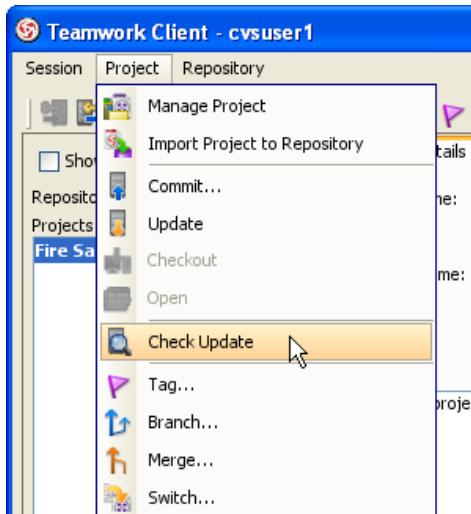


Figure 3-39 Update project from menu

Or click **Check Update** button from toolbar.



Figure 3-40 Update project from toolbar

4. You can see the status showing **Has update** or **Up-to-date**, it also indicate local project status (**local project not modified**) or (**local project modified**).

Status: Has update (local project not modified)

Figure 3-41 Project status

Reverting changes

Roll back local and uncommitted modifications

1. Modify the project.

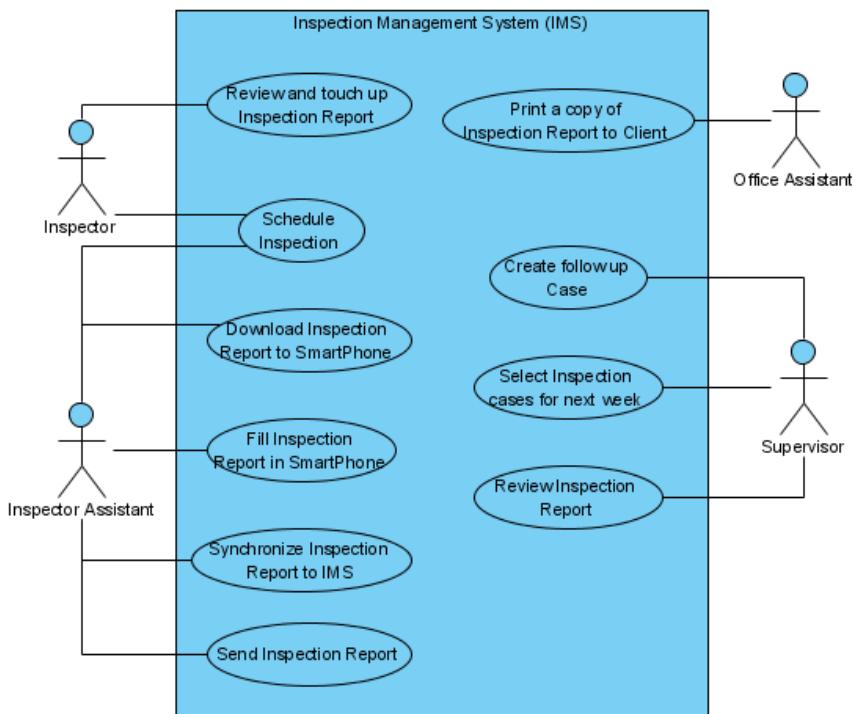


Figure 3-42 Modify project

2. Open **Teamwork Client** dialog.
3. Select the teamwork project in the list.
4. Select **Project > Revert...** from menu

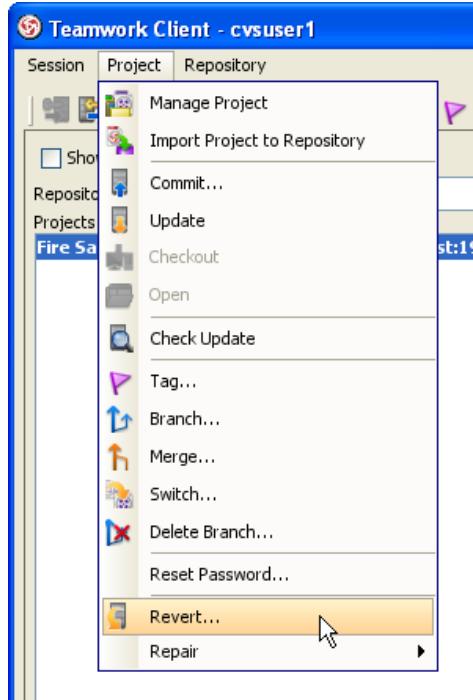


Figure 3-43 Revert from menu

Or click **Revert...** button from toolbar.



Figure 3-44 Revert from toolbar

5. Click **Yes** button from the confirmation dialog.
6. The reverted project opened in **VP-UML**.

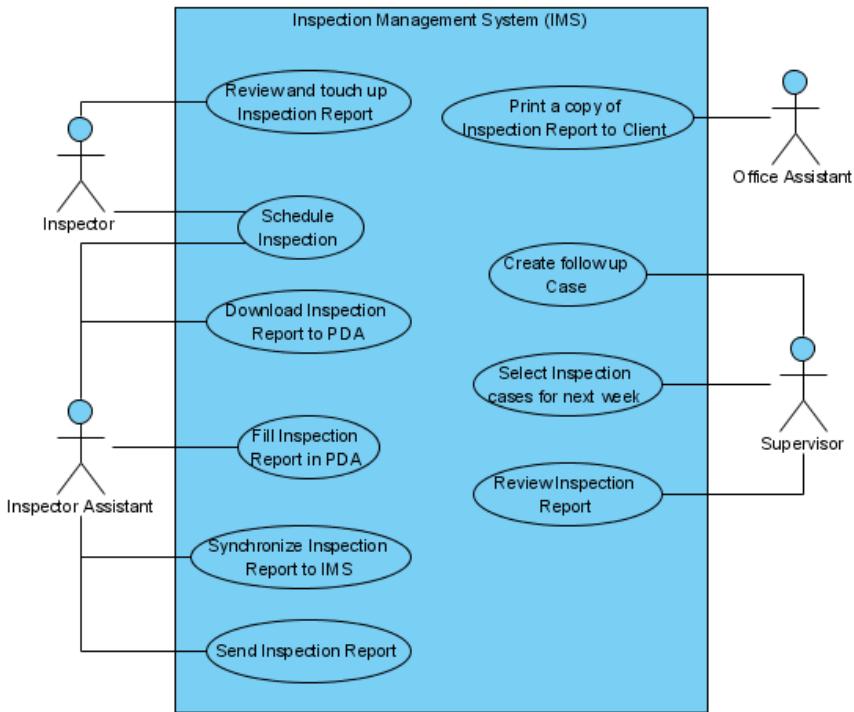


Figure 3-45 Reverted project

Roll back modifications of a past revision

If there were undesired changes made in past revision(s), such as wrong deletion of elements, you can undo the changes by reverting the revision(s) involved. Here are the steps:

1. Open project.

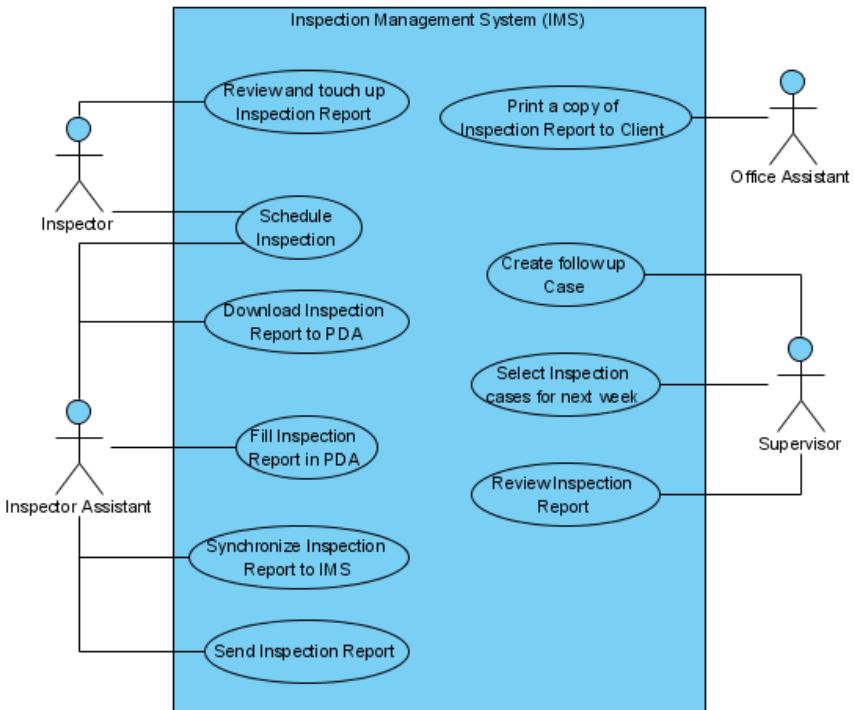


Figure 3-46 Open project

2. Open **Teamwork Client** dialog.
3. Select the teamwork project in the list.

4. Click the **Revisions** tab on the right of the project list.

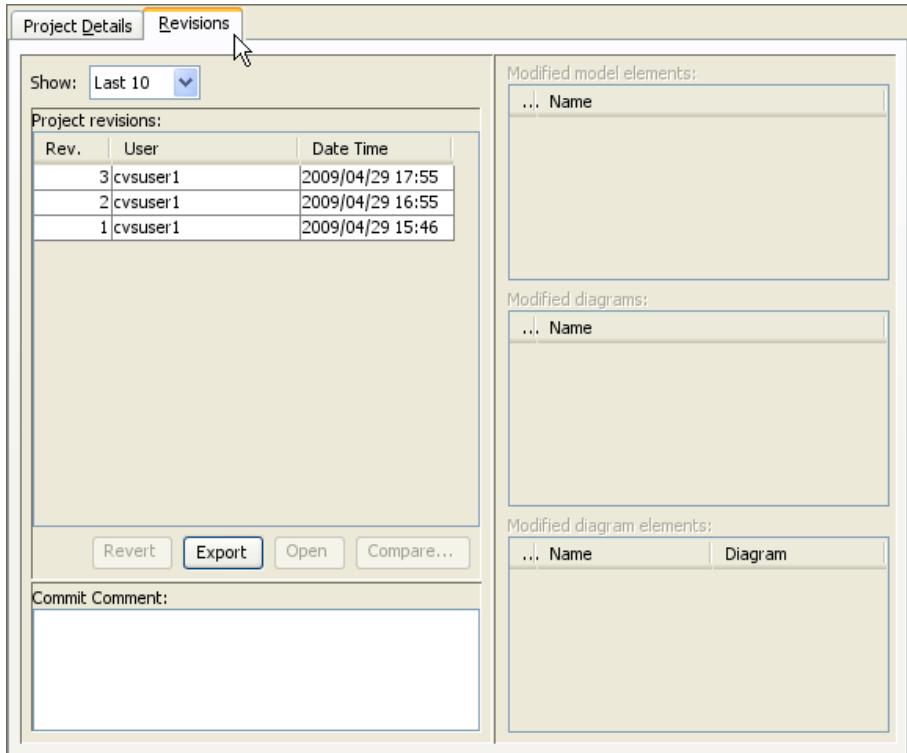


Figure 3-47 Revision tab

5. Select the revision(s) to have its changes rolled back.

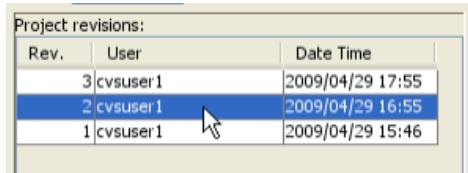


Figure 3-48 Select revision

6. Click **Revert** button.



Figure 3-49 Revert button

NOTE: You may make multiple selection by selecting one revision, pressing the **Control** key, and selecting the rest. Note that a non-consecutive revision selection is not revertable.

7. The **Commit** dialog show a list of shapes and model elements reverted for review the changes and preview diagram, input the comment and click **OK** button to commit the revert.

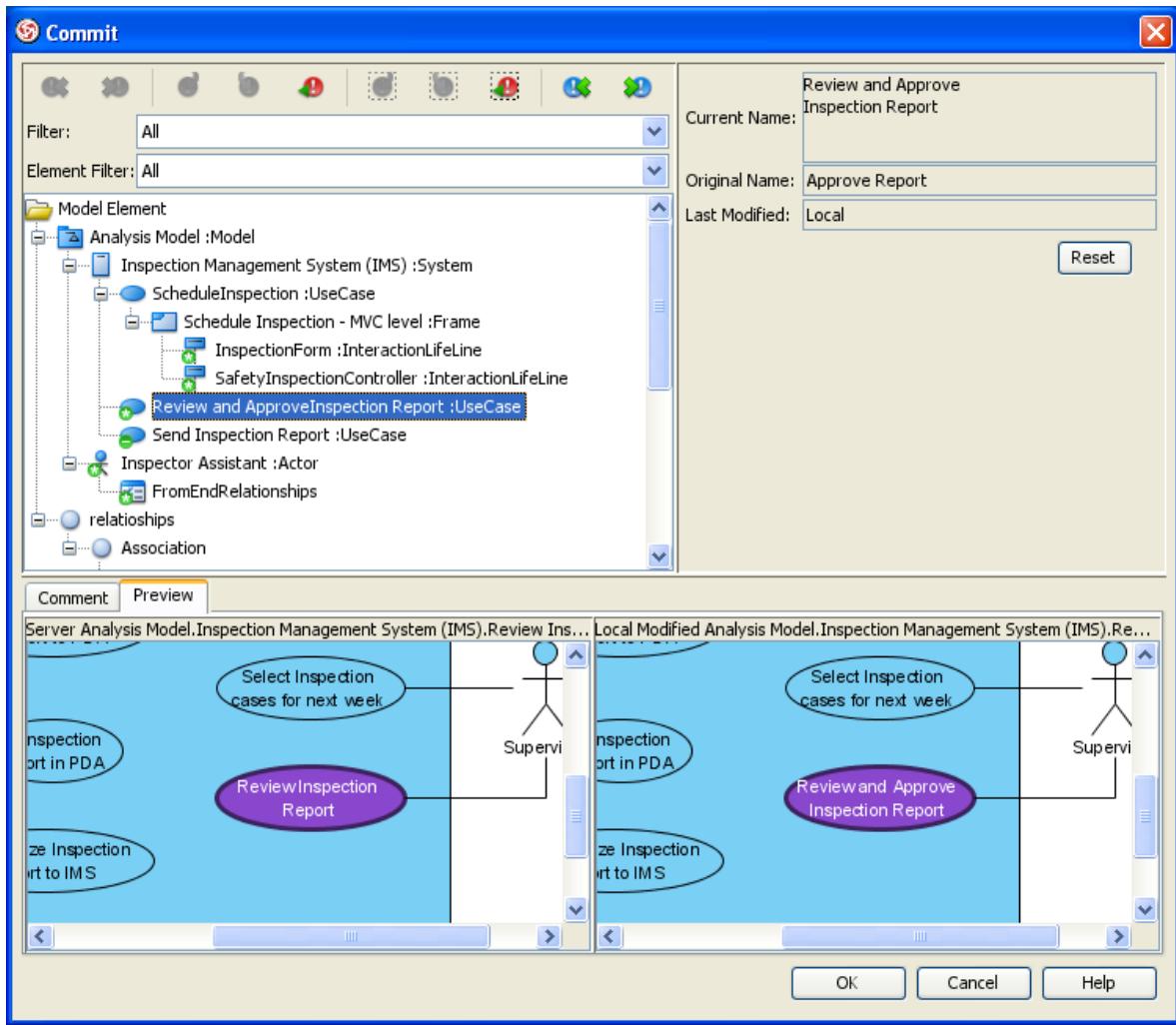


Figure 3-50 Commit dialog with reverted changes

8. A new revision with the reverted changes was created in server, and opened in VP-UML.

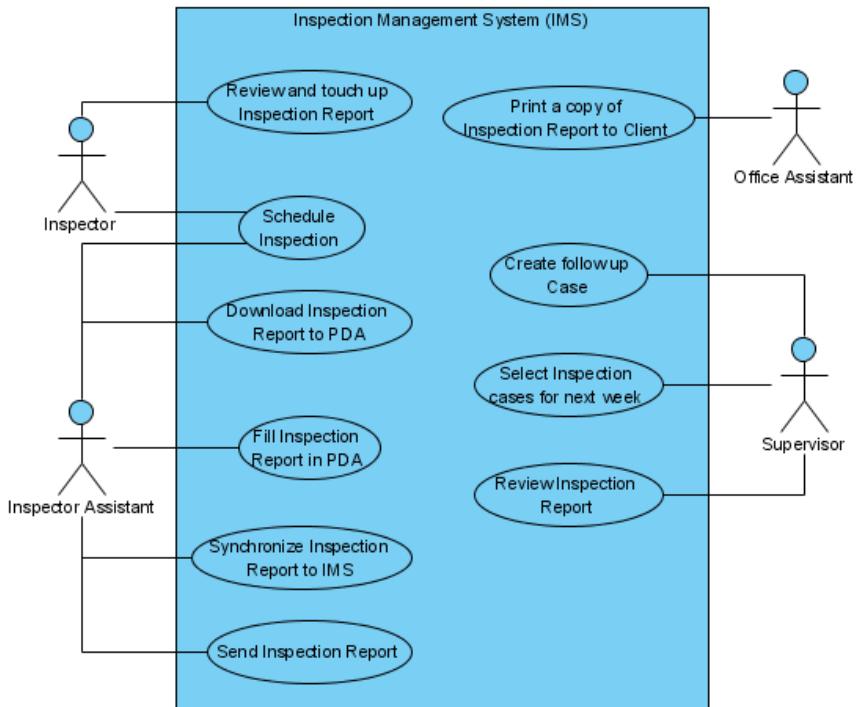


Figure 3-51 Reverted project

Browsing change histories (old revisions)

Checkout old revisions

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select the revision to open.
5. Click Open button.



Figure 3-52 Open button

6. Selected revision opened in **VP-UML**.

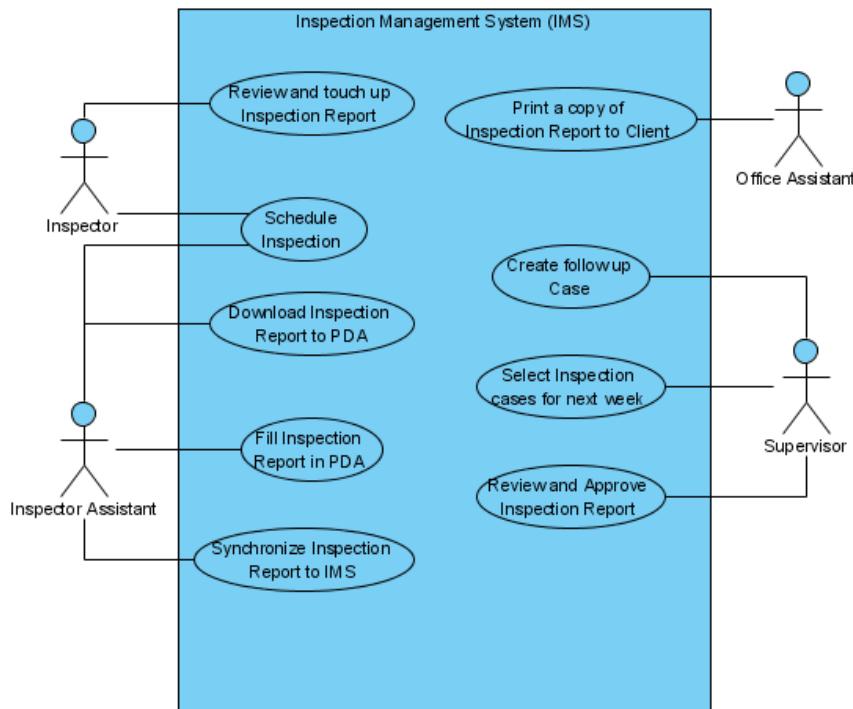


Figure 3-53 Open selected revision

Showing differences between revisions visually

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select a revision for compare.
5. Select another revision by **Ctrl+Click**.

Project revisions:		
Rev.	User	Date Time
4	cvsuser1	2009/04/30 09:28
3	cvsuser1	2009/04/29 17:55
2	cvsuser1	2009/04/29 16:55
1	cvsuser1	2009/04/29 15:46

Figure 3-54 Select two revisions

6. Click the **Compare...** button.

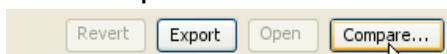


Figure 3-55 Compare button

7. Similar to **Commit** and **Update** dialog, the **Compare Projects** dialog show a list of differences between the selected revisions. You can also view the differences visually in diagram on the preview tab.

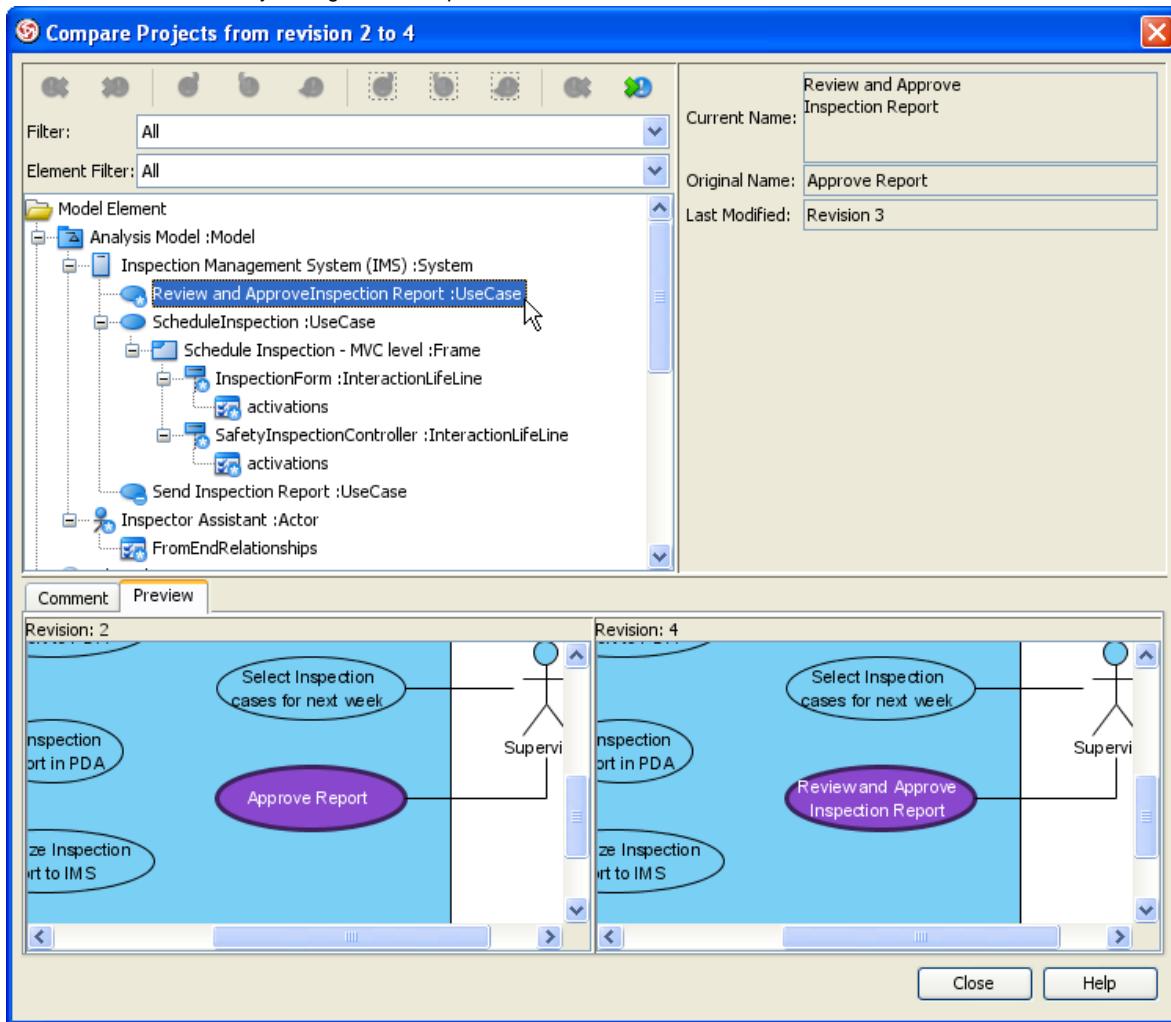


Figure 3-56 Compare differences

Export multiple revisions to local

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select multiple revisions for export, by **Ctrl+Click** or **Shift+Click**.

Project revisions:		
Rev.	User	Date Time
4	cvsuser1	2009/04/30 09:28
3	cvsuser1	2009/04/29 17:55
2	cvsuser1	2009/04/29 16:55
1	cvsuser1	2009/04/29 15:46

Figure 3-57 Select multiple revisions

5. Click the **Export** button.



Figure 3-58 Export buttons

6. Select **Export selected revisions...** from the popup.

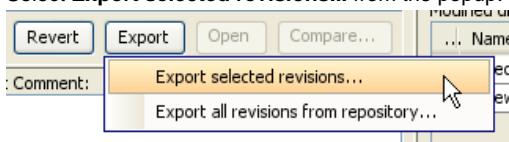


Figure 3-59 Export selected revisions

7. Select the directory to save the exported project revisions.

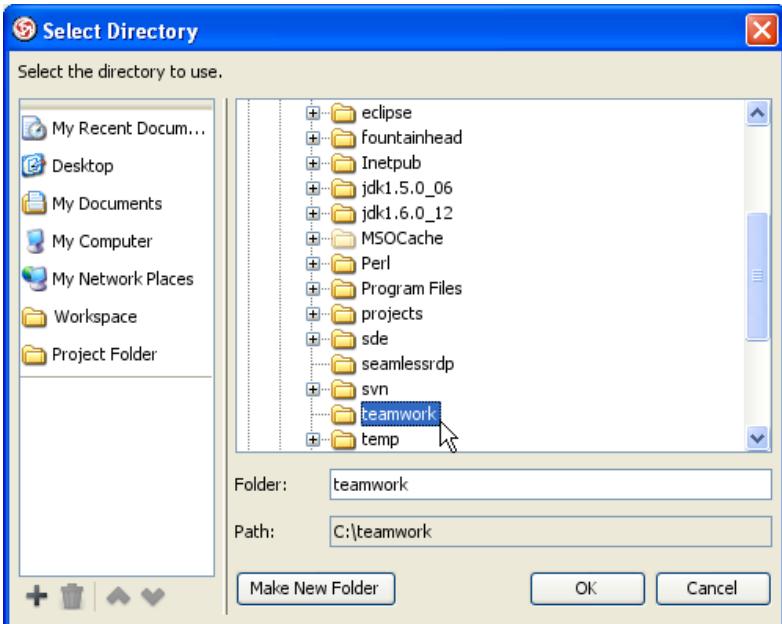


Figure 3-60 Select directory

8. You'll find the selected revisions was exported to the select directory.

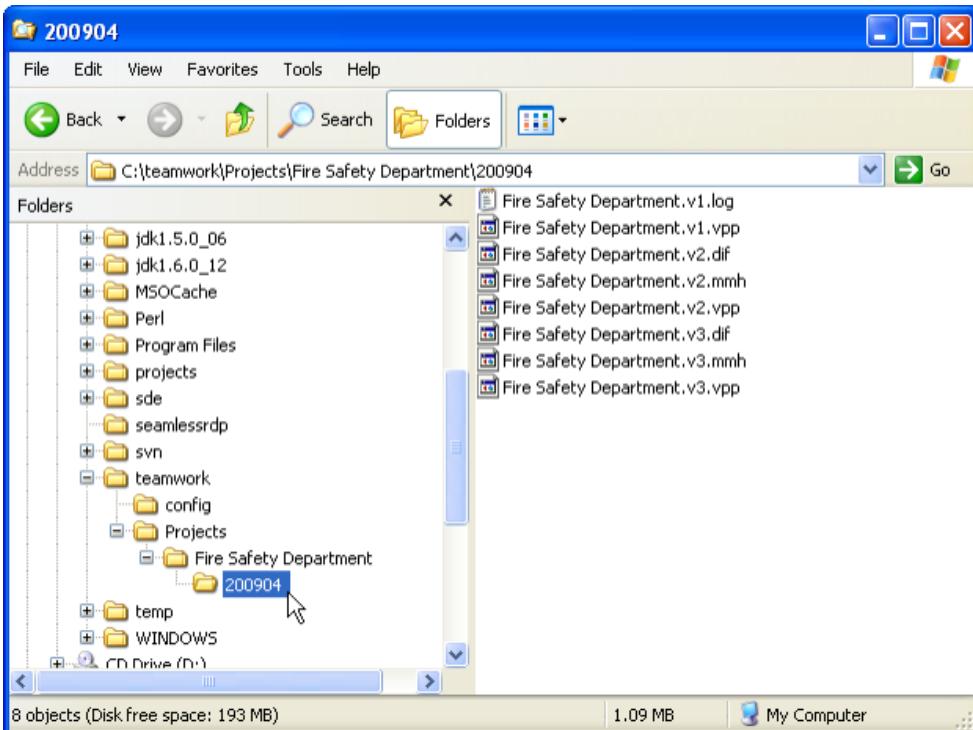


Figure 3-61 Exported revisions

Isolating last long modifications with branches

Creating branch

1. Open Teamwork Client dialog.
2. Select the teamwork project in the list.
3. Select Project > Branch... from menu

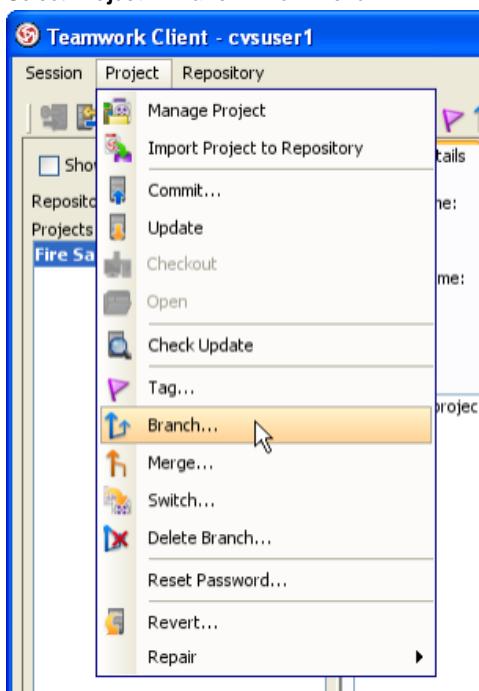


Figure 3-62 Branch from menu

Or click Branch... button from toolbar.



Figure 3-63 Branch from toolbar

4. Fill in the **Branch Name** in Create Branch dialog.

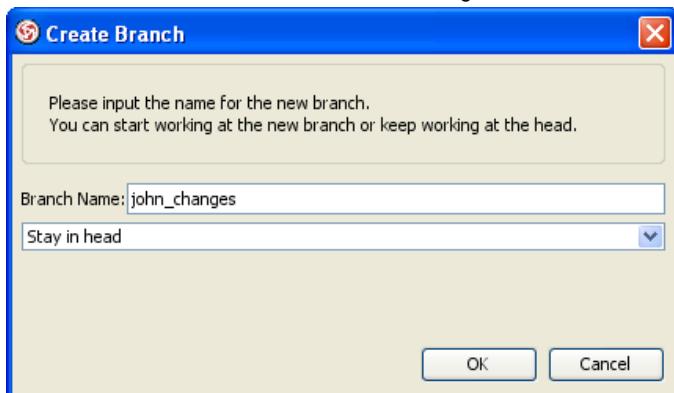


Figure 3-64 Create Branch dialog

5. Select **Stay in head** in the list.

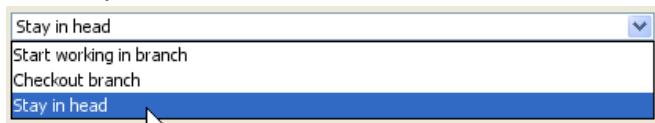


Figure 3-65 Stay in head

Switch local copy between branches

1. Select Project > Switch... from menu

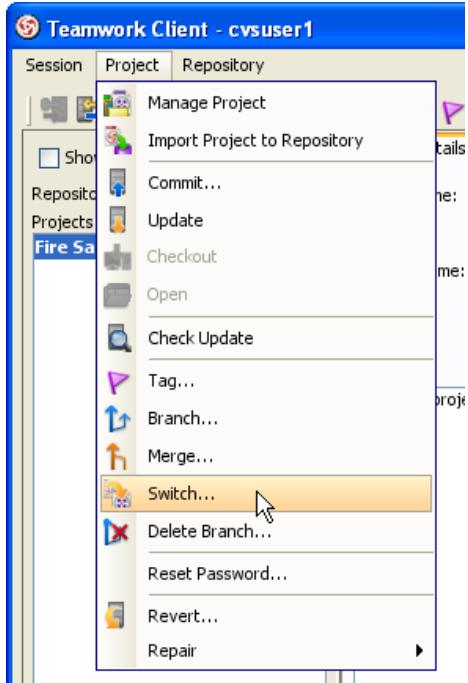


Figure 3-66 Switch from menu

Or click **Switch...** button from toolbar.



Figure 3-67 Switch from toolbar

2. Select the branch to switch.



Figure 3-68 Select branch

3. The branch is opened in VP-UML.

Merging from HEAD to branch

1. Open, modify and commit HEAD project.

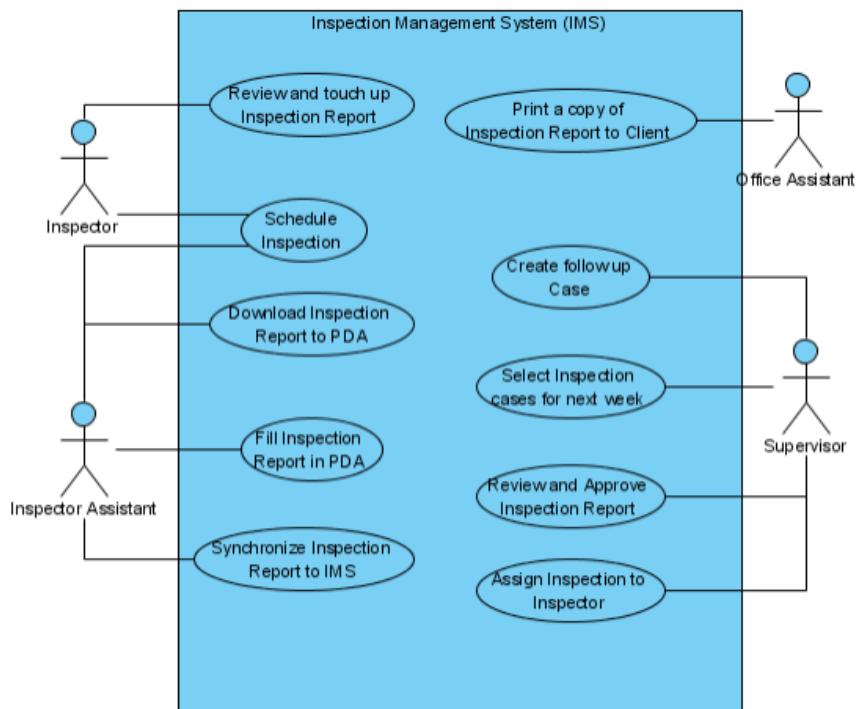


Figure 3-69 Modified HEAD project

2. Open branch project.

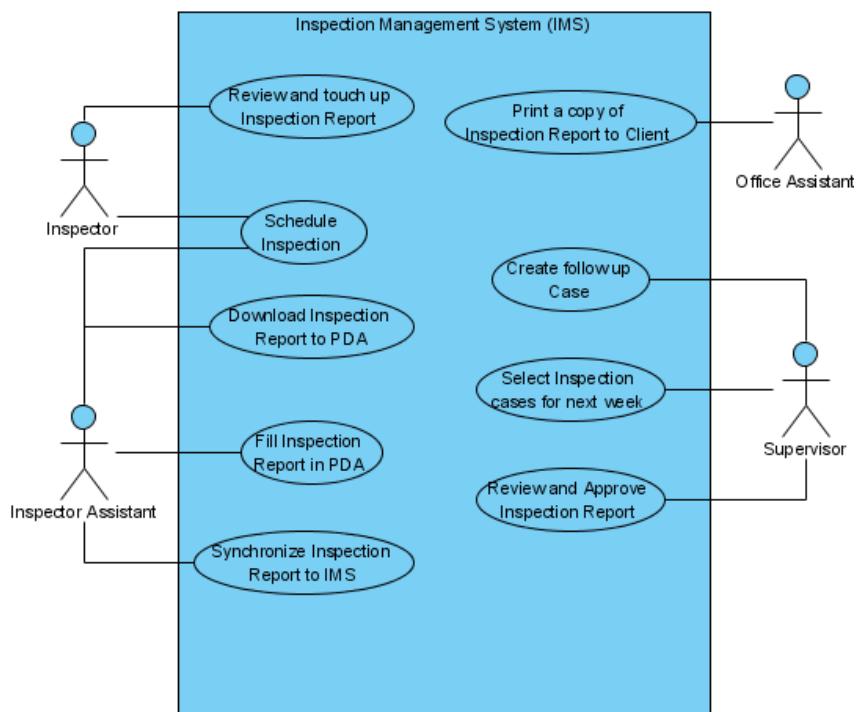


Figure 3-70 Branch project

3. Select **Project > Merge...** from menu.

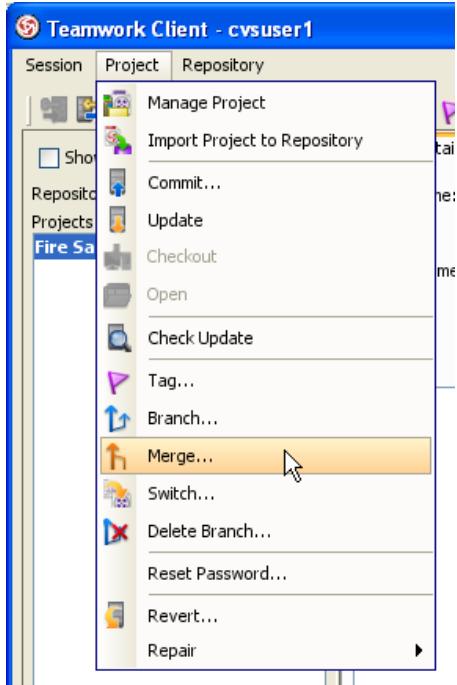


Figure 3-71 Merge from menu

Or click **Merge...** button from toolbar.



Figure 3-72 Merge from toolbar

4. Select from HEAD in the **Merge** dialog.

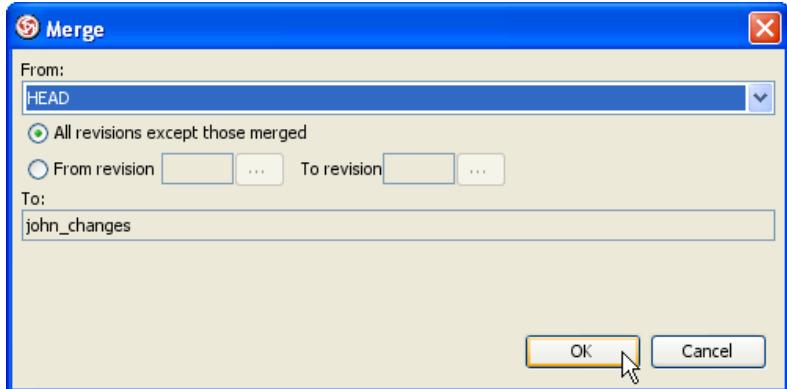


Figure 3-73 Merge dialog

5. The **Merge Project** dialog shows a list of changes similar to **Commit** and **Update** dialog.

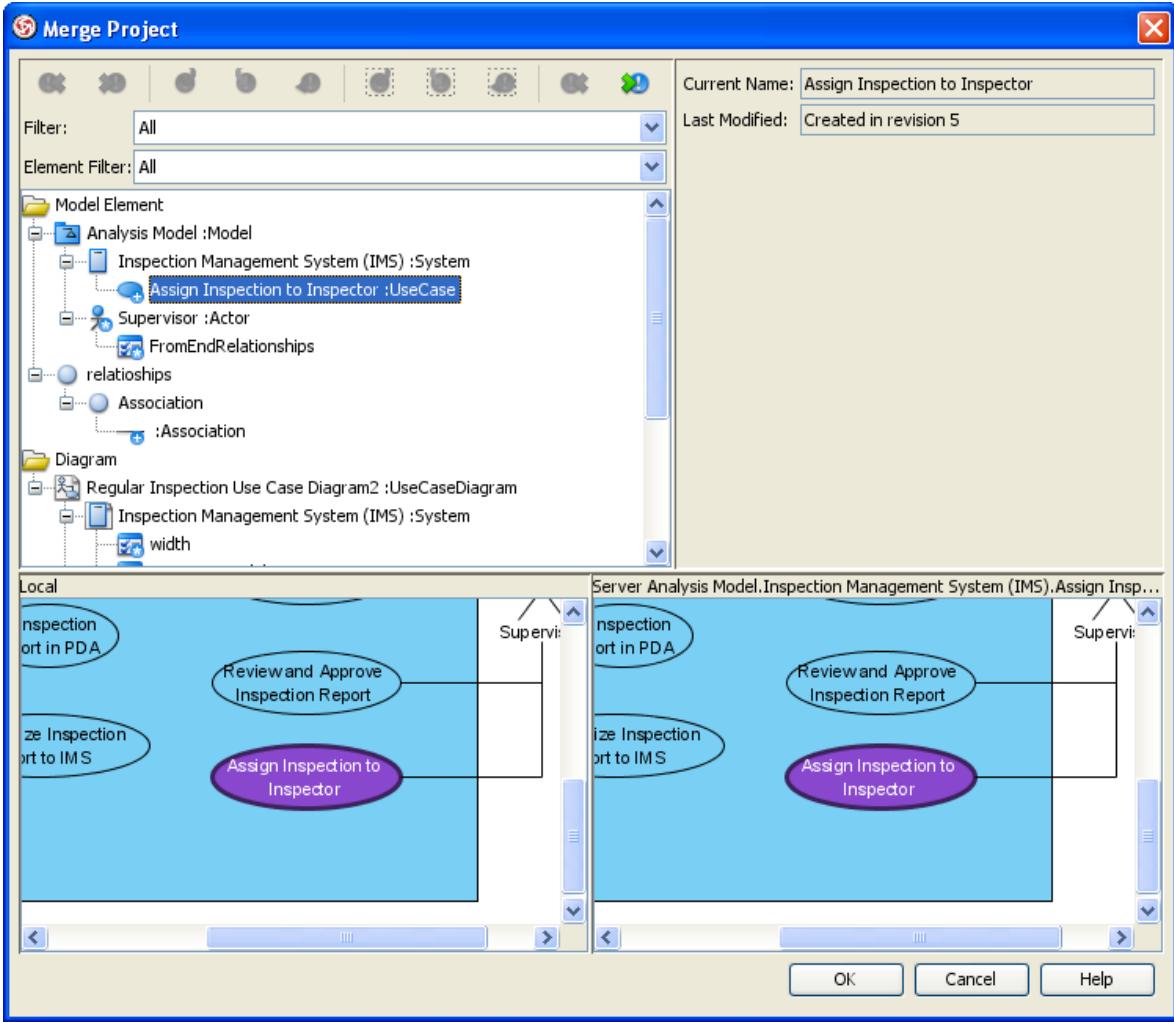


Figure 3-74 Merge project dialog

6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.
 7. The branch project now contains the changes from HEAD.

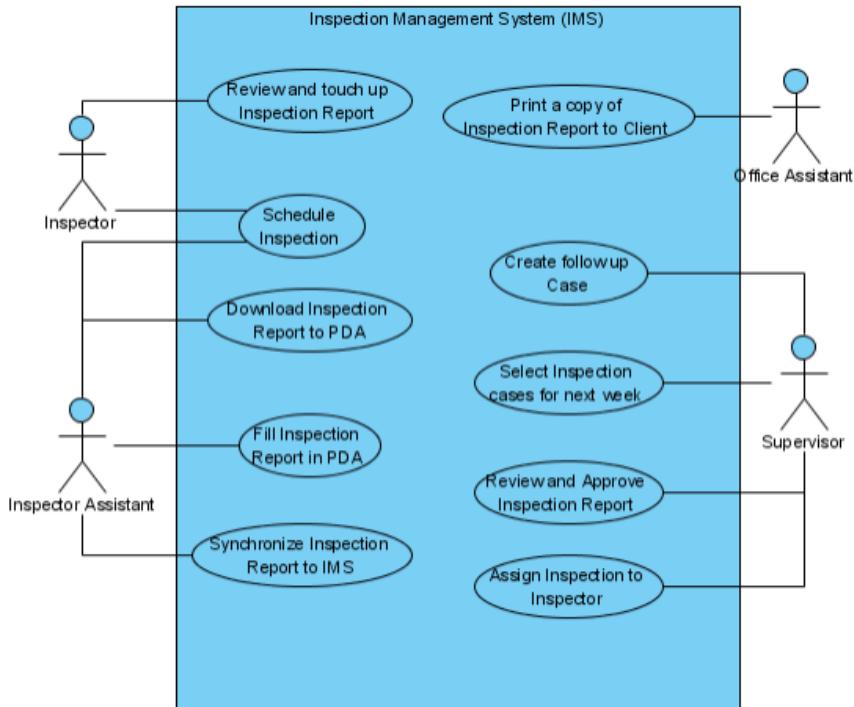


Figure 3-75 Branch project merged from HEAD

Merging from branch to HEAD

1. Open, modify and commit branch project.

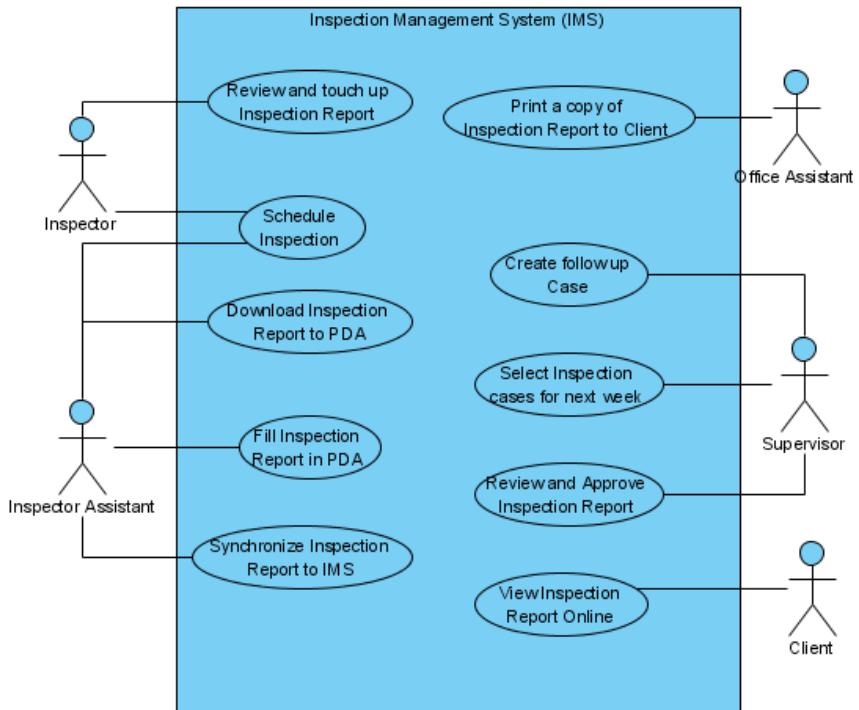


Figure 3-76 Modified branch project

2. Open HEAD project.

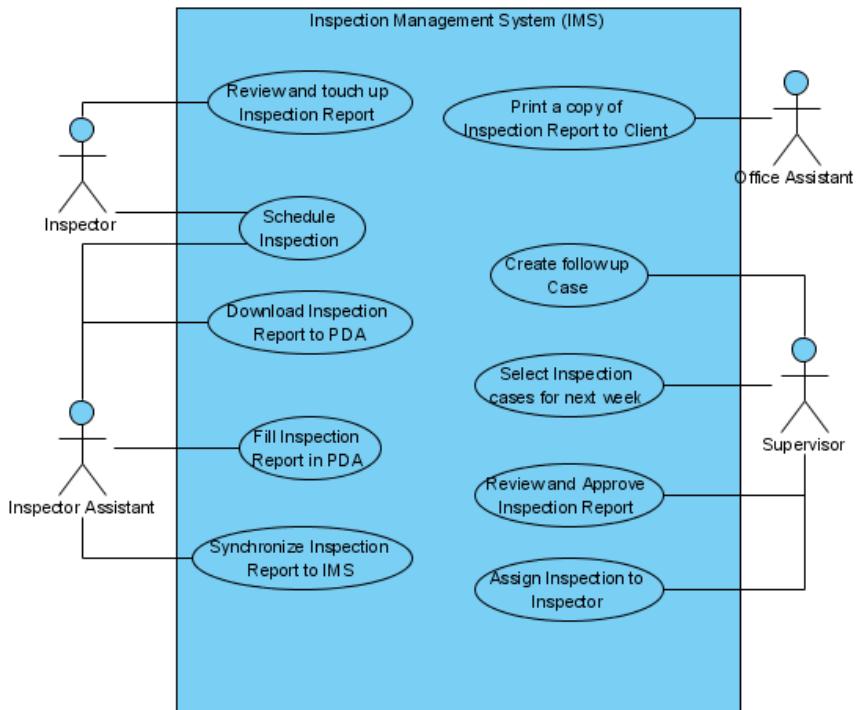


Figure 3-77 HEAD project

3. Select **Project > Merge...** from menu, or click **Merge...** button from toolbar.

4. Select a branch name in **From** combo box.

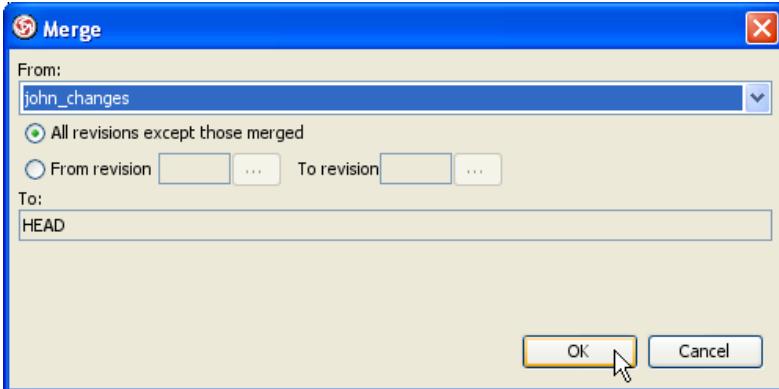


Figure 3-78 Merge dialog

5. The **Merge Project** dialog show a list of changes similar to **Commit** and **Update** dialog.
 6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.
 7. The HEAD project now contains the changes from branch.

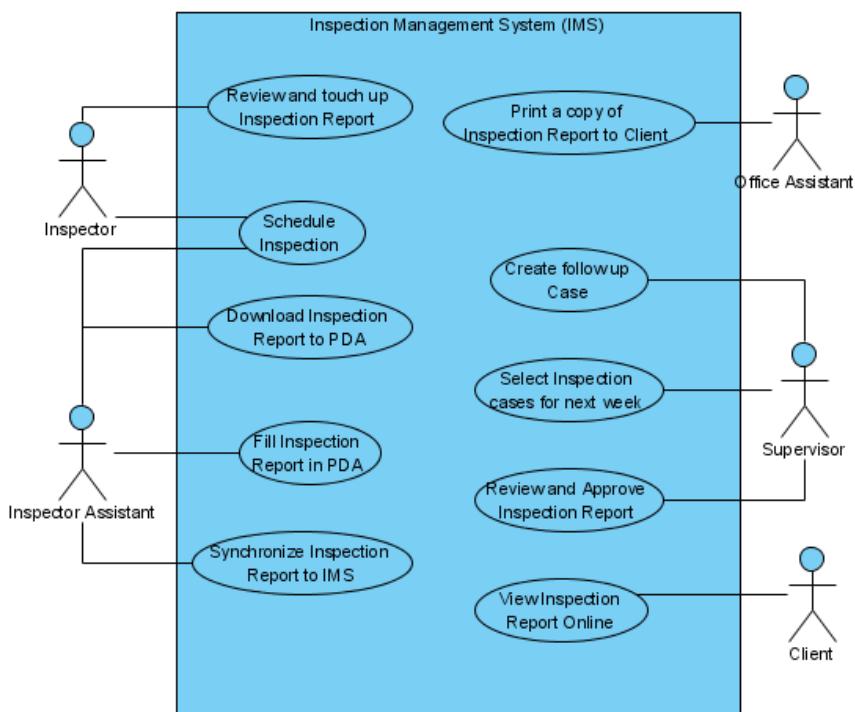


Figure 3-79 HEAD project merged from branch

Delete branch

1. Select Project > Delete Branch... from menu

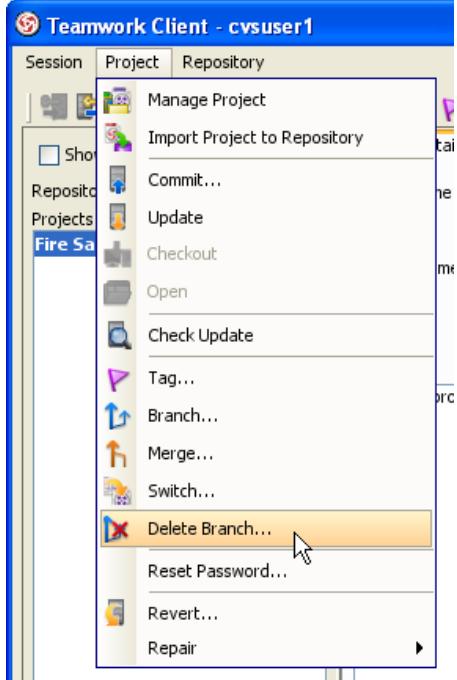


Figure 3-80 Delete Branch from menu

Or click Delete Branch ... button from toolbar.



Figure 3-81 Delete Branch from toolbar

2. Expand the repository and folder node, select the branch to delete. Click OK button to continue.

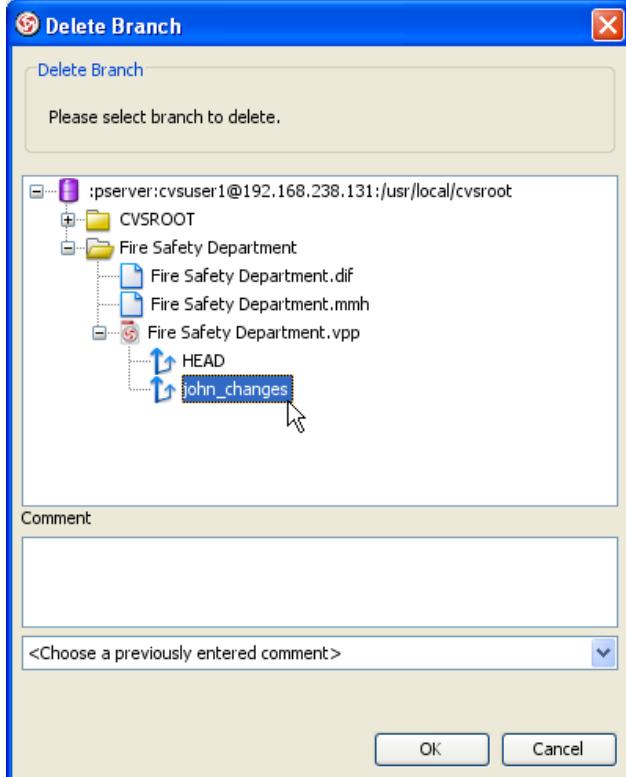


Figure 3-82 Delete branch dialog

3. Click Yes button on confirmation dialog.

Marking release or milestone with tags

Tags and branches are almost the same, with the only difference - tags are read only. Creating read only tags ensure you are able open the release version project later.

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Tag...** from menu

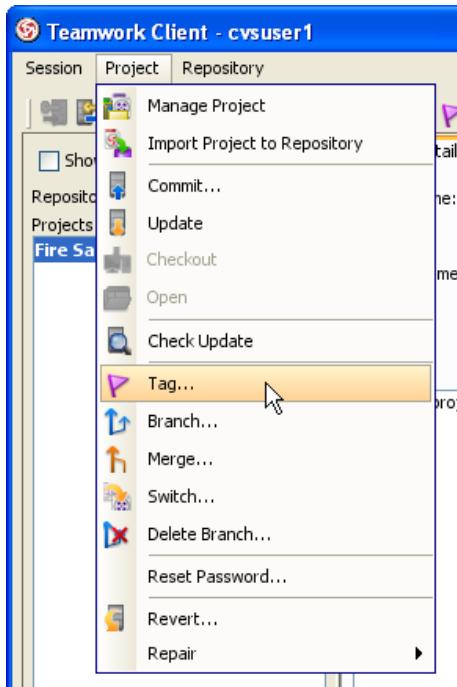


Figure 3-83 Tag from menu

Or click **Branch...** button from toolbar.



Figure 3-84 Tag from toolbar

4. Fill in the **Tag Name** in **Create Tag** dialog.

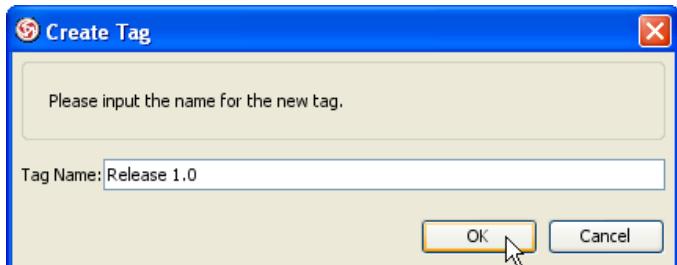


Figure 3-85 Create Tag dialog

5. Click **OK** button, the tag will be created.

Working with teamwork client dialog box

To perform teamwork operations such as to commit or update project, you can click the buttons in application menus or toolbars. Besides, you can perform via the Teamwork Client dialog box. Teamwork Client dialog box enables you to perform teamwork operations, manage teamwork projects, as well to review and checkout revisions of particular project.

Opening the teamwork client

1. Open the Teamwork Client by either of the ways below:
 - Select **Tools > Teamwork > Open Teamwork Client...** from the main menu.

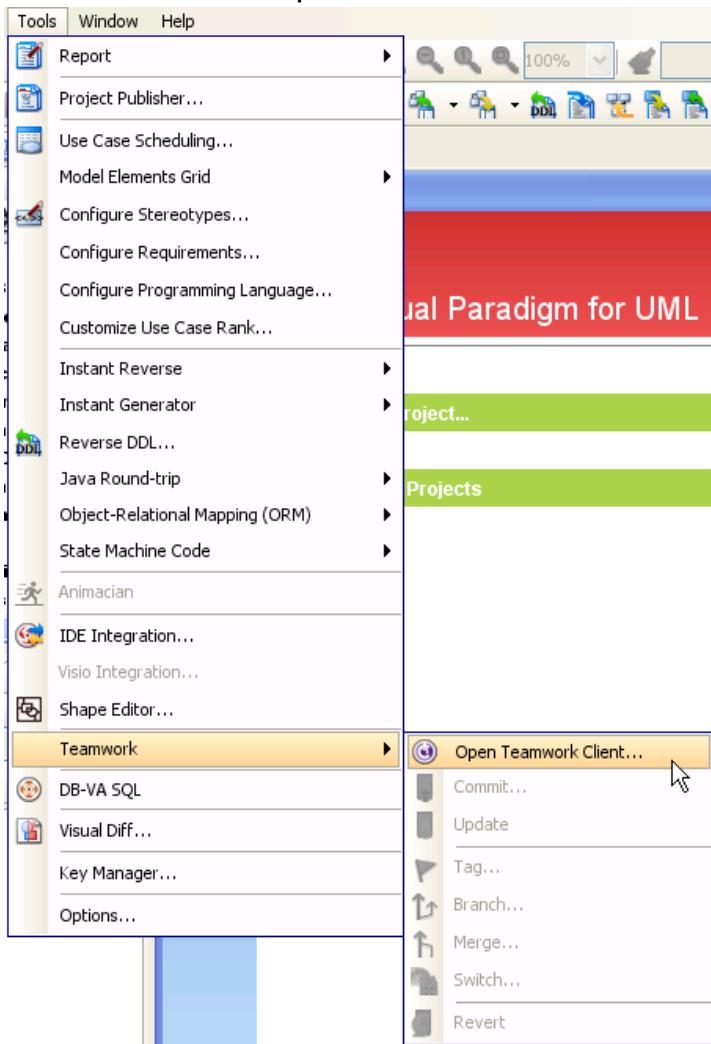


Figure 4-1 Open teamwork client through menu

- Click on the button **Open Teamwork Client** in toolbar.



Figure 4-2 Open teamwork client through toolbar

2. In the login dialog box, fill in the server and user information to login to server.

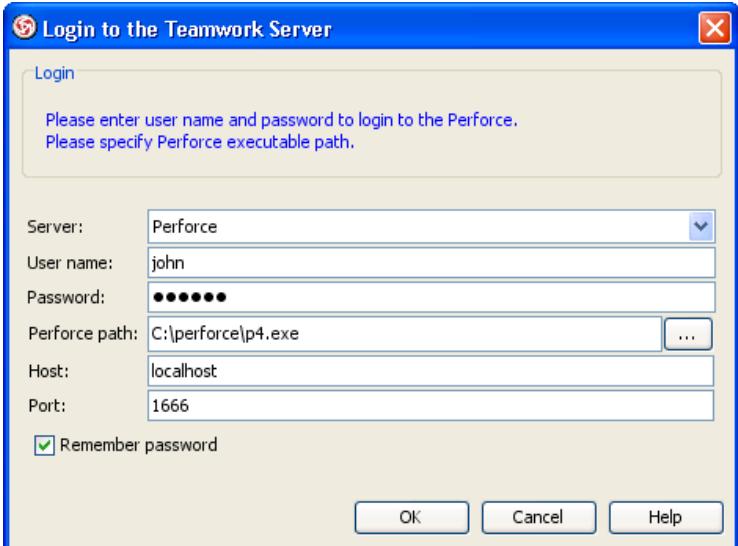


Figure 4-3 Login Perforce

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be Perforce
Server host	The server host, which can be host name or IP address
Port number	The port of connecting to server
User name	The user name for connecting to server
Password	The password for given user
Remember password	Check to memorize the password so that you don't need to enter again
Use proxy	Check if proxy is needed for connecting to server. When checked, you will need to enter proxy host.

Table 4-1 Description of **Login** dialog box

3. If this is the first time you connect to the Perforce server, you will be asked to select the projects to manage. By managing a project, you can check it out, open and edit it. In the **Manage Project** dialog box, select the projects to manage.

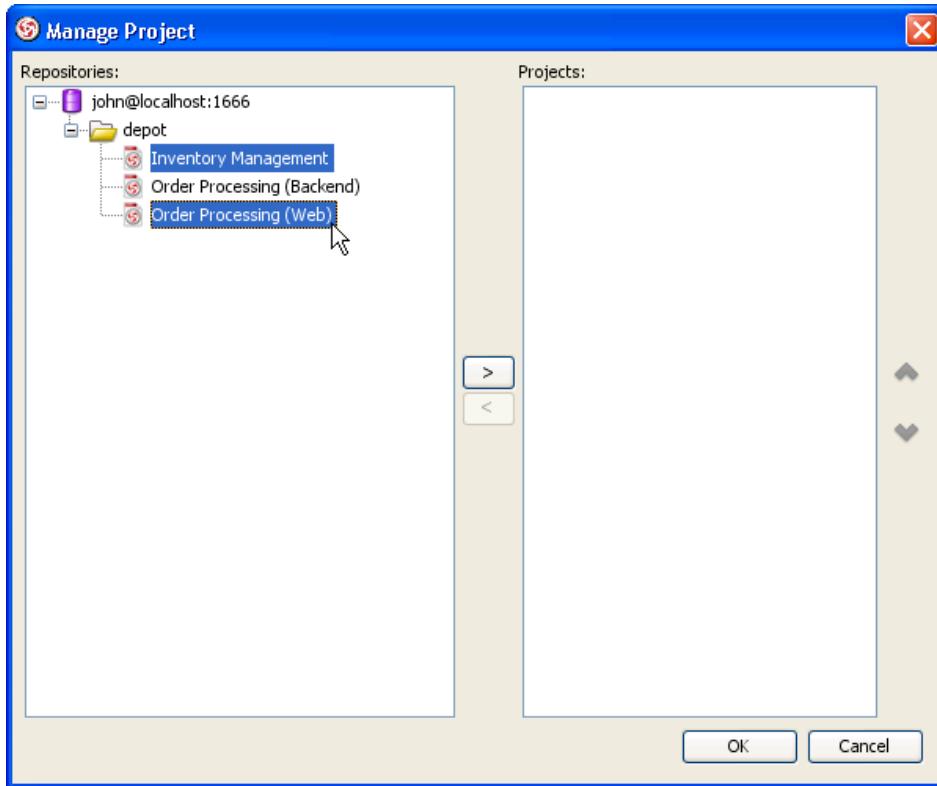


Figure 4-4 Select projects to manage

4. Click on the > button in the middle of the dialog box to add them into project selection.
 5. Click **OK** to proceed. This end up showing the **Teamwork Client** dialog box.

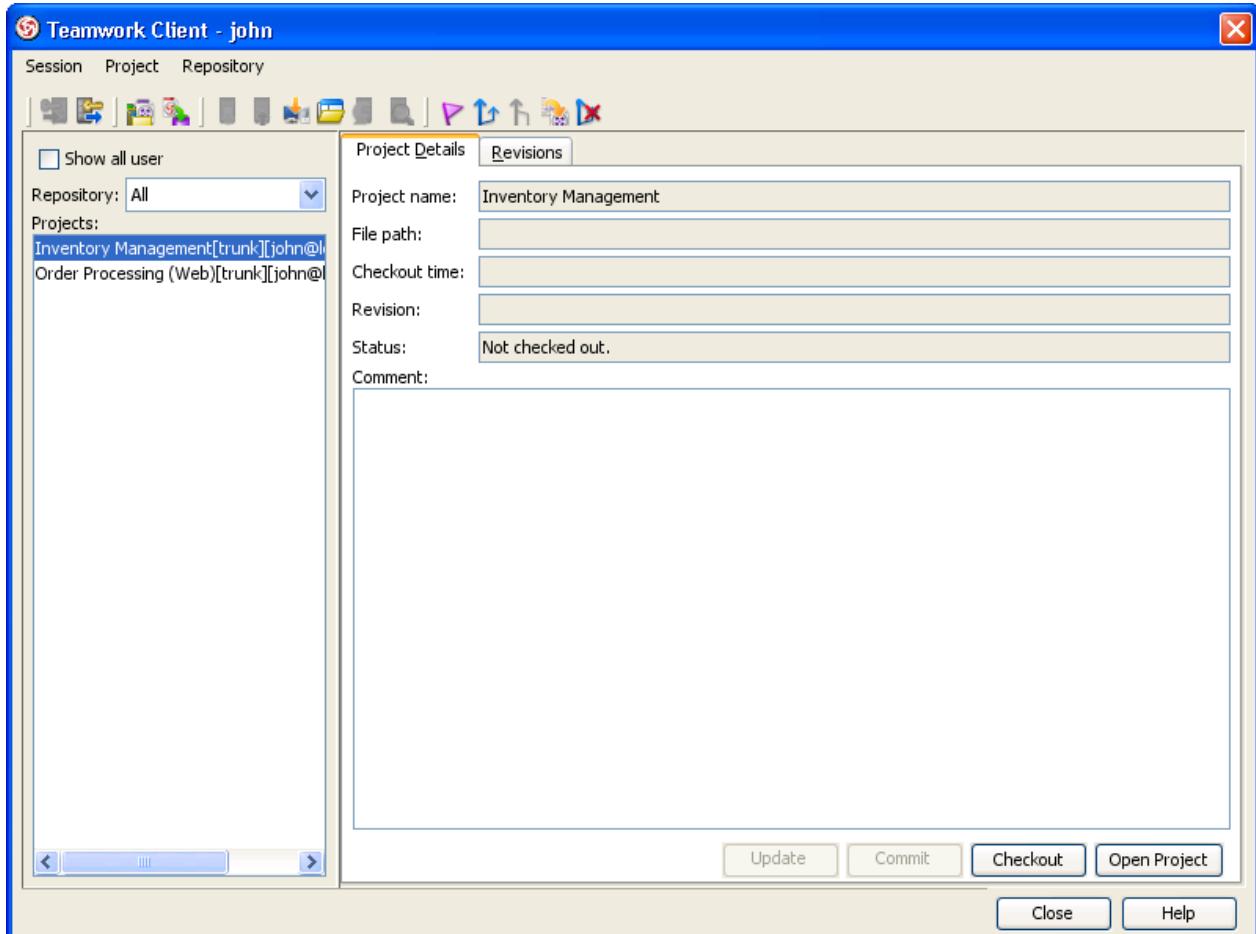


Figure 4-5 The Teamwork Client dialog box

The interface

General

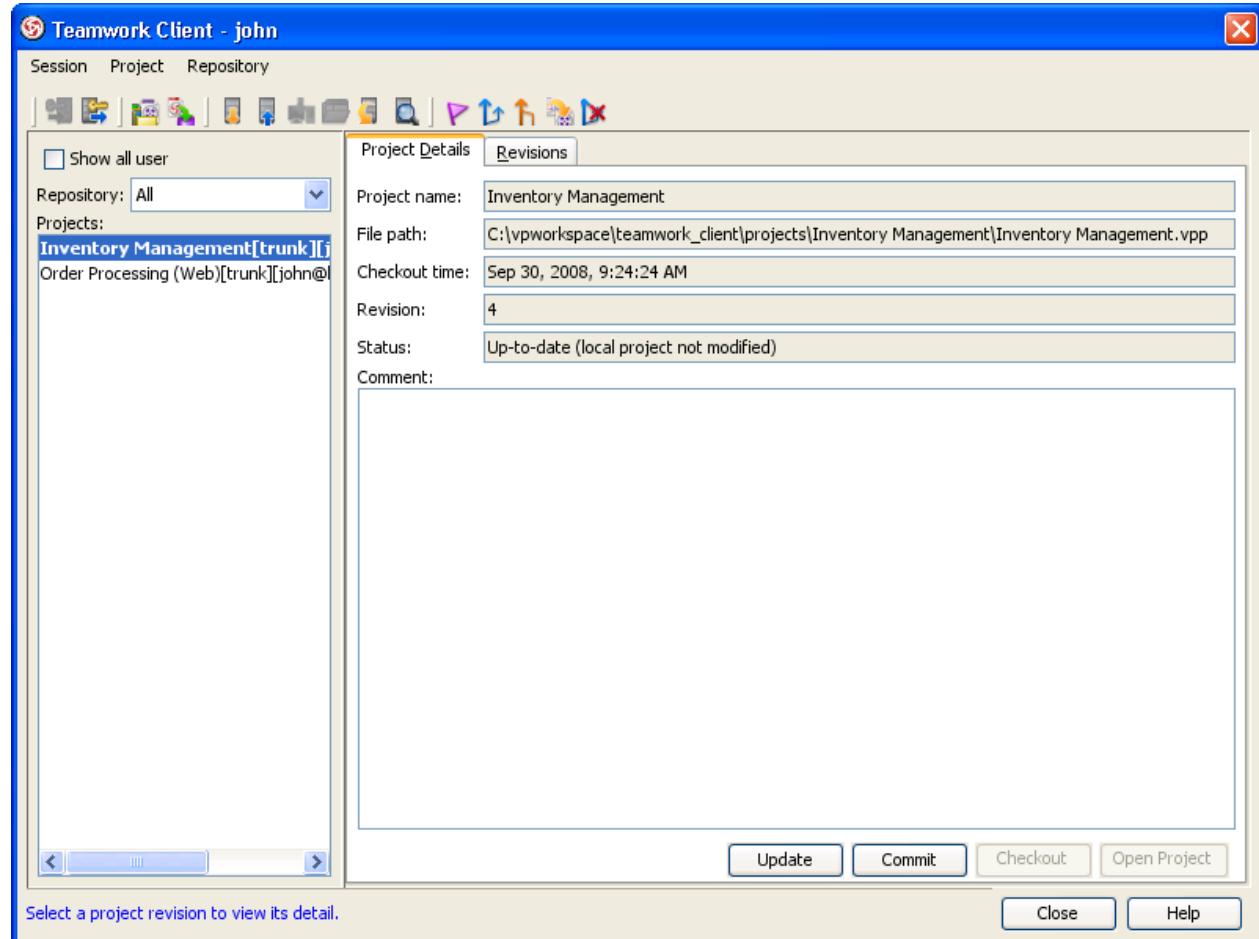


Figure 4-6 The Teamwork Client dialog box

Region	Description
	To logout from Perforce server.
	To login to Perforce server.
	To manage projects, which enables you to add or remove projects from the list of managed projects.
	To create a project in server by importing one into it. You may create a blank new project, or import an existing project file to start with.
	To update changes from server.
	To commit changes to server.
	To checkout a project from server.
	To open the project selected in Projects list. If the project is not checked out yet, it will be checked out automatically.
	To remove changes made in a revision.
	To check for changes
	To create a tag for a trunk/branch.
	To create a branch for a trunk/branch.
	To merge changes from trunk/branch.
	To switch to another trunk/branch/tag.



To remove a branch. Note that removed branch will not delete the, but just make it not accessible.

Show all user To show the projects manageable by all users.

Repository To update the **Projects** list by showing only projects in certain repository.

Projects The list of manageable project.

Project Details Details of selected project, including the name, revision and status.

Revisions Revisions of selected project.

Table 4-2 Description of the Teamwork Client dialog box

Project details

The screenshot shows the 'Project Details' tab selected in a dialog box. The form contains the following fields:

Project name:	Inventory Management
File path:	C:\vpworkspace\teamwork_client\projects\Inventory Management\Inventory Management.vpp
Checkout time:	Sep 30, 2008, 9:24:24 AM
Revision:	4
Status:	Up-to-date (local project not modified)
Comment:	(Large text area)

At the bottom of the dialog are four buttons: 'Update', 'Commit', 'Checkout', and 'Open Project'.

Figure 4-7 The Project Details tab in Teamwork Client dialog box

Field	Description
Project name	Name of the selected project.
File path	The file path of the project in your system.
Checkout time	The time when you checkout the project.
Revision	The revision of your local copy.
Status	The status of local project.
Comment	The comment of project entered by the administrator when creating project.

Table 4-3 Description of Project Details tab in Teamwork Client dialog box

Revisions

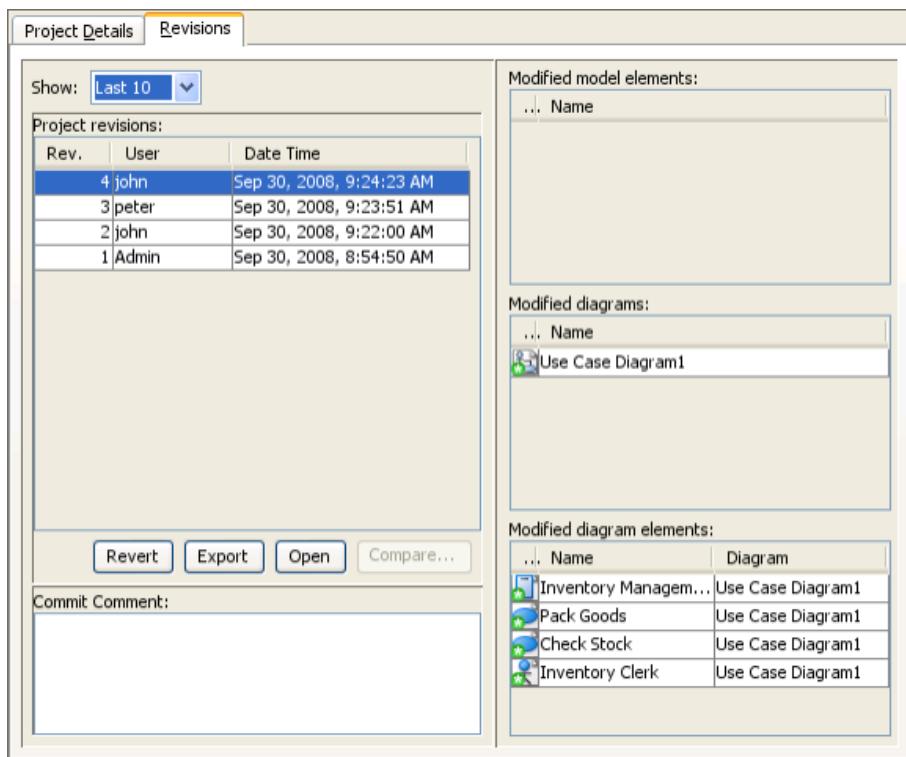


Figure 4-8 The Revisions tab in Teamwork Client dialog box

Region	Description
Show:	To list certain amount of revisions, such as all, last 10, last 20, etc.
Revision list	To display the revisions of chosen project.
[Revert]	By reverting a selected revision, this means to undone changes made in that revision.
[Export]	To export selected revision(s) as project files.
[Open]	To open selected revision.
[Compare]	To compare the differences between revisions.

Table 4-4 Description of Revisions tab in Teamwork Client dialog box

Importing projects to performe

1. Start VP-UML.
2. Select **Tools > Teamwork > Open Teamwork Client...** from the main menu, or **Open Teamwork Client** icon from toolbar to open teamwork client.

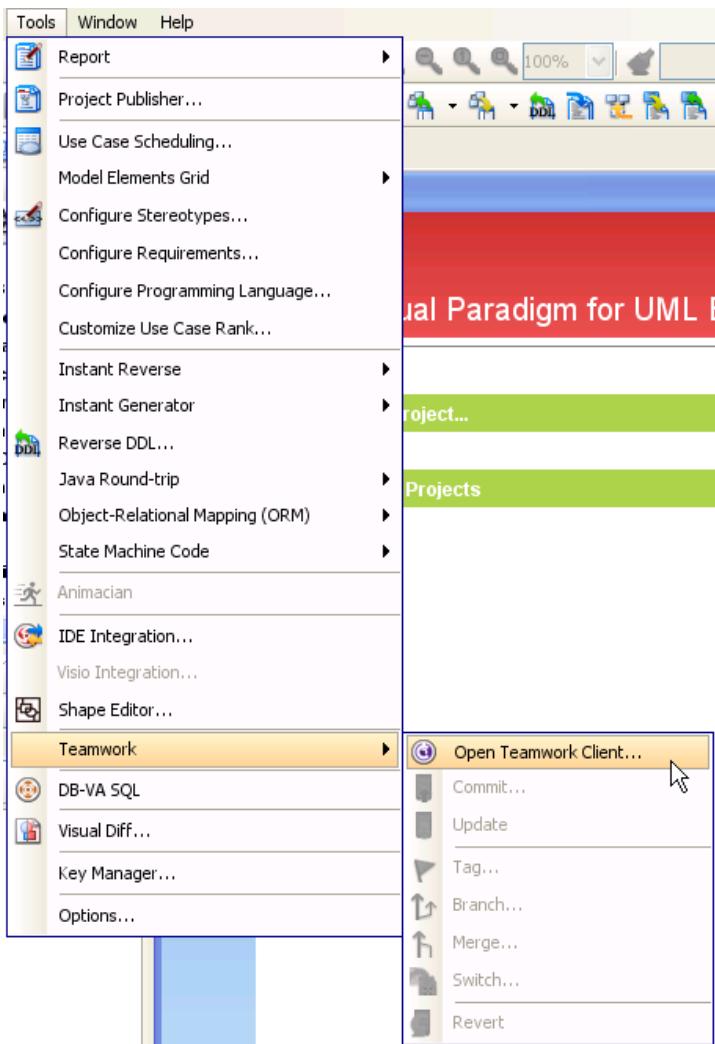


Figure 4-9 Open Teamwork Client from main menu

3. Select **Perforce** in **Server** field.

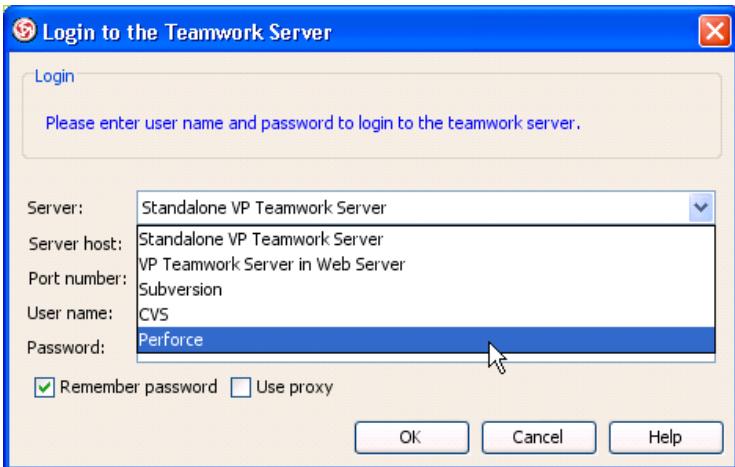


Figure 4-10 perforce teamwork server

4. Fill in the server and user information to login teamwork server.

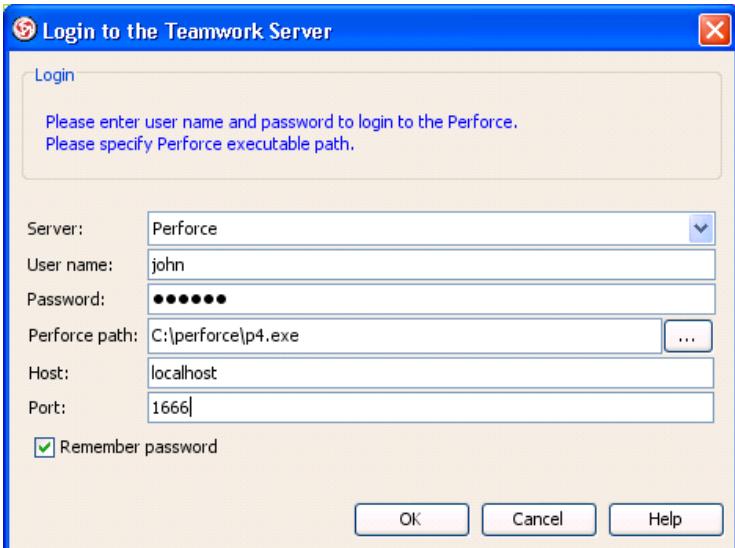


Figure 4-11 Login teamwork server

Below is a description of fields in the **Login** dialog box:

Field	Description
Server	The type of server, which should be Perforce
User name	The user name for connecting to server
Password	The password for given user
Perforce path	The filepath of p4.exe. You are required to supply a valid path in order to connect to Perforce server
Host	The host of Perforce server
Port	The port number of connecting to server
Remember password	Check to memorize the password so that you don't need to enter again

Table 4-5 Description of **Login** dialog box

5. Select **Project > Import Project to Repository** from the menu.

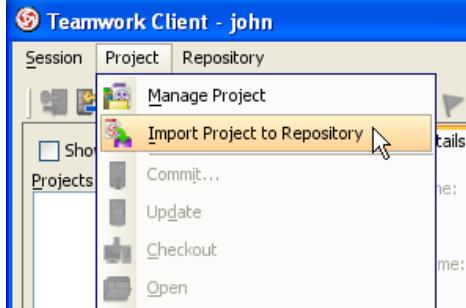


Figure 4-12 Import project from menu

Or click the **Import Project to Repository** button on toolbar.



Figure 4-13 Import project from toolbar

6. In the **Import Project** dialog, fill in the **Project name**.

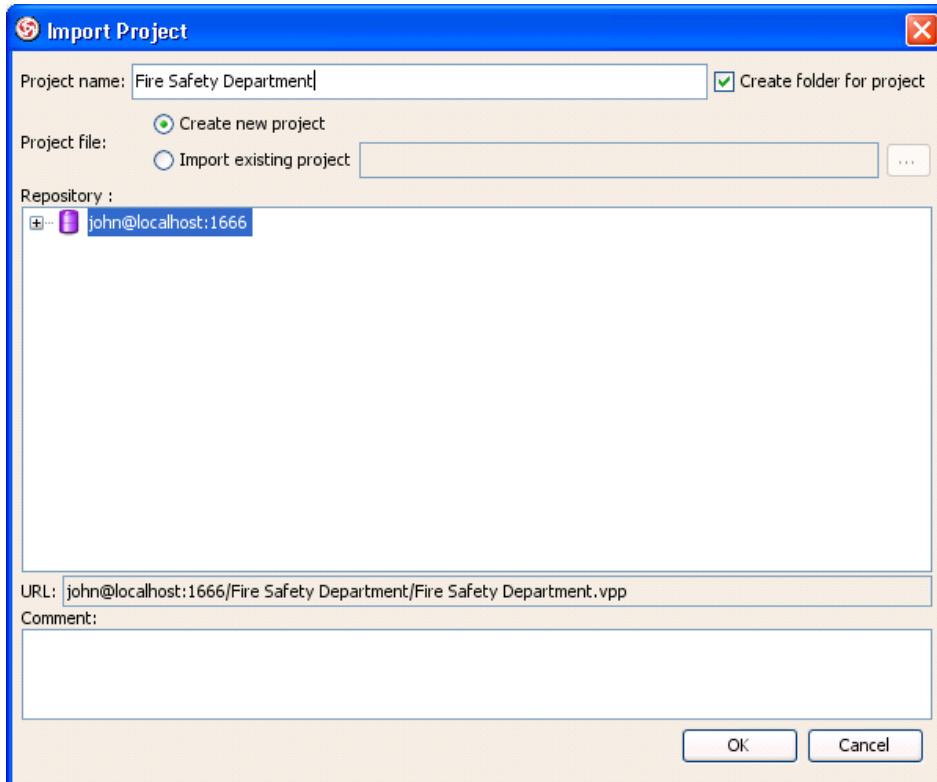


Figure 4-14 Fill project name in Import Project dialog

Field	Description
Project name	The name of teamwork project
Create folder for project	Create a folder in repository for storing the project. The folder will be of same name as the project.
Project file	Create new project - Start with a blank, new project Import existing project - Start by using an existing project file
Repository	A tree which shows the structure of repository
URL	The URL of the project in repository
Comment	The comment of project

Table 4-6 Description of Import Project dialog box

7. Choose **Create new project** or **Import existing project** from **Project file**. **Create new project** will import a blank project into teamwork server, **Import existing project** will import an existing VP-UML Project file (*.vpp) into teamwork server as first revision.



Figure 4-15 Specify project file

8. Select the **depot** node for import the project.

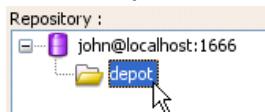


Figure 4-16 Select repository

9. Fill in the comment.

Figure 4-17 Input comment

10. Click **OK** button to import project.

Checkout project from perforce

Checkout and open perforce projects

1. Open Teamwork Client dialog.
2. Select Project > Manage Project from the menu.

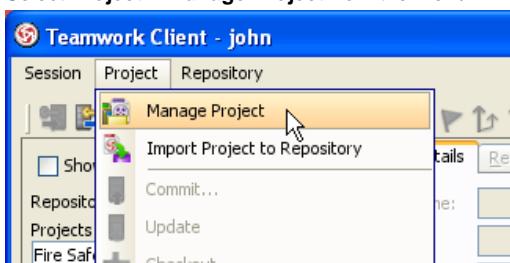


Figure 4-18 Manage project from main menu

Or click **Manage Project** icon on the toolbar.

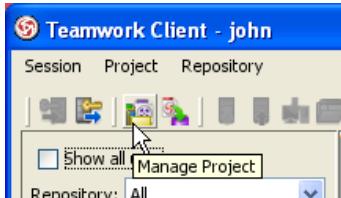


Figure 4-19 Manage project from toolbar

3. In **Manage Project** dialog, expand the repository node and select the project to checkout, click > button to add project. Click **OK** button to close the dialog.

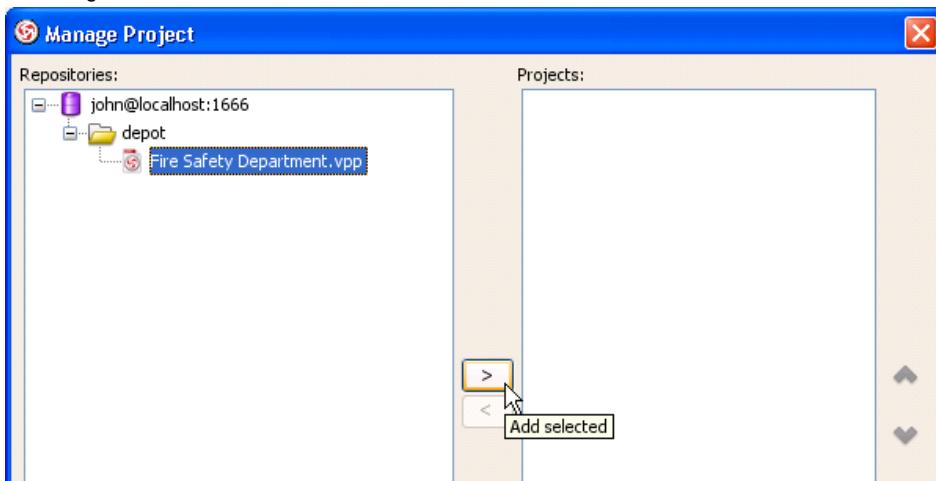


Figure 4-20 Manage projects

4. Select the project in project list.

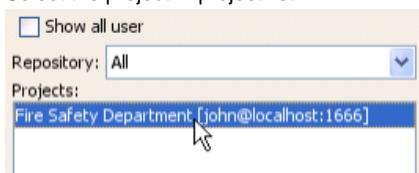


Figure 4-21 Select project from the list

5. Click **Open Project** button to checkout and open the project.

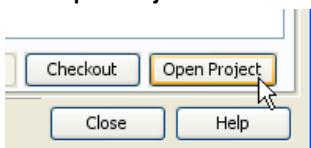


Figure 4-22 Open project

6. The project opened in VP-UML.

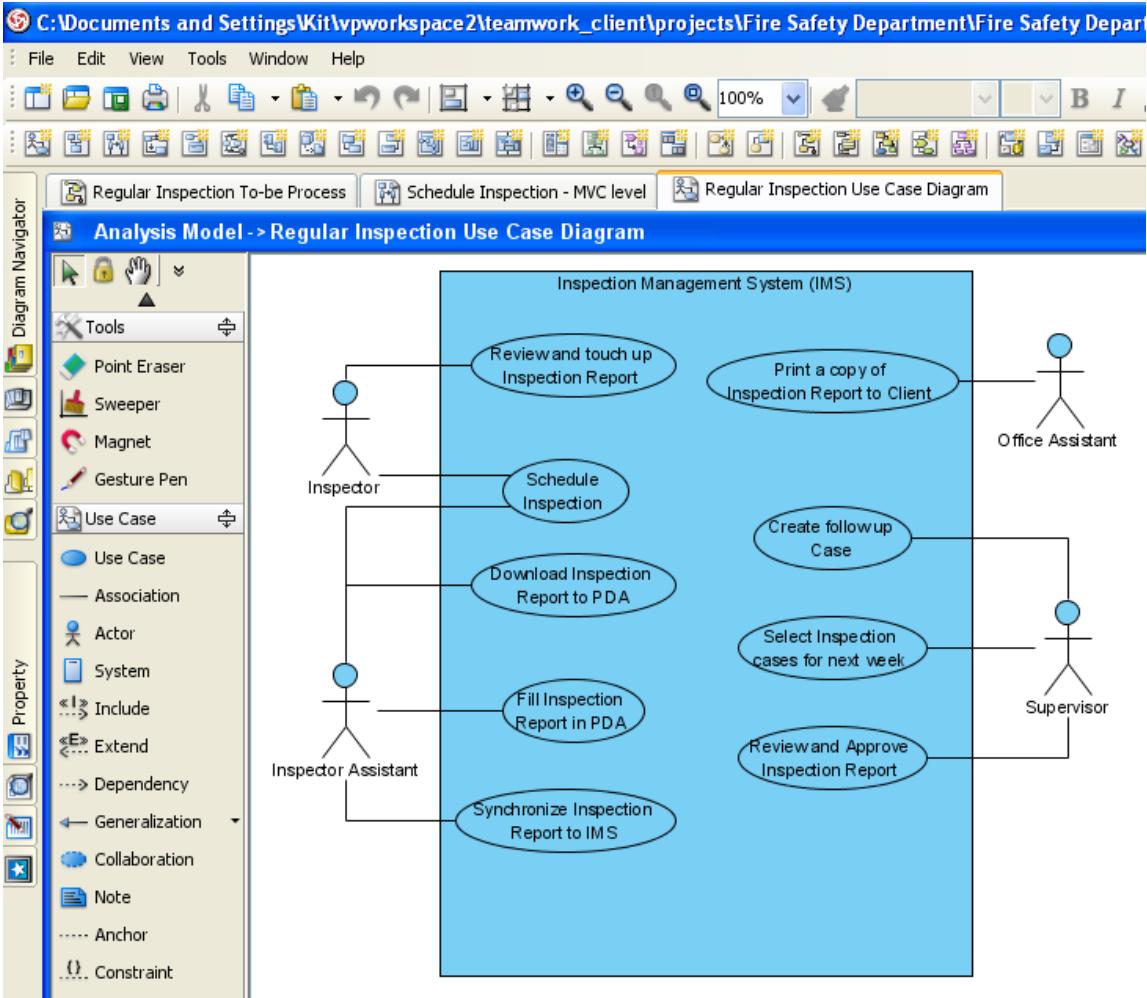


Figure 4-23 Project opened

Committing local modification to performe

1. Modify the project.

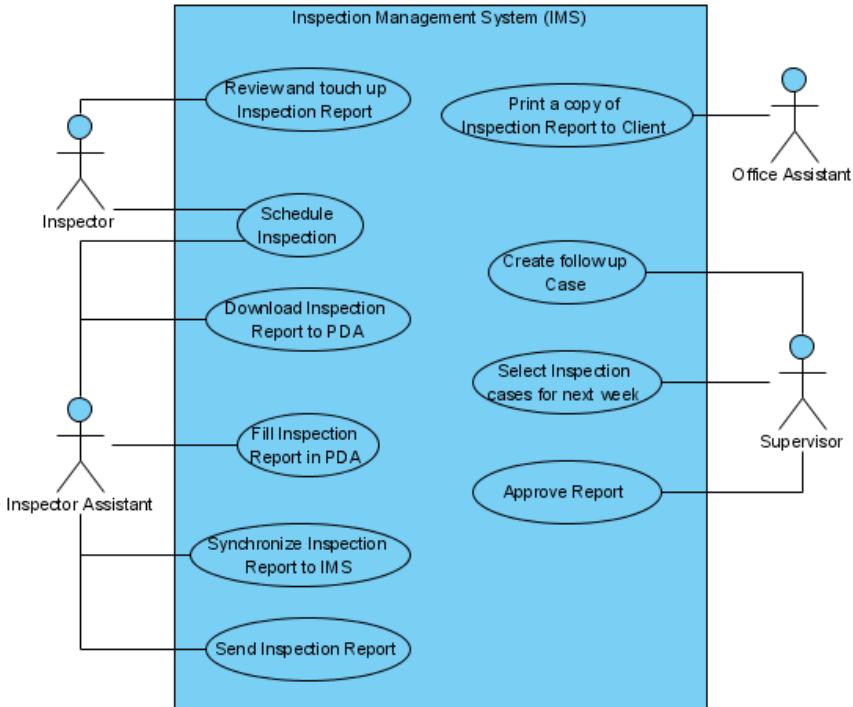


Figure 4-24 Modified project

2. Show **Teamwork Toolbar** by right click on the Toolbar, select **Teamwork** if not selected already.

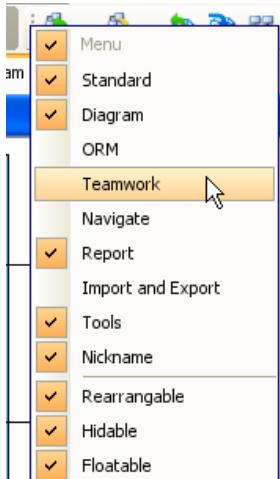


Figure 4-25 Show teamwork toolbar

3. Click the **Commit** button on **Teamwork Toolbar**.



Figure 4-26 Commit project

4. In **Commit** dialog, you can review the changes for commit. On the left of dialog, you can see a list of changes shapes and model elements, click on it to view the detail changes on the right.

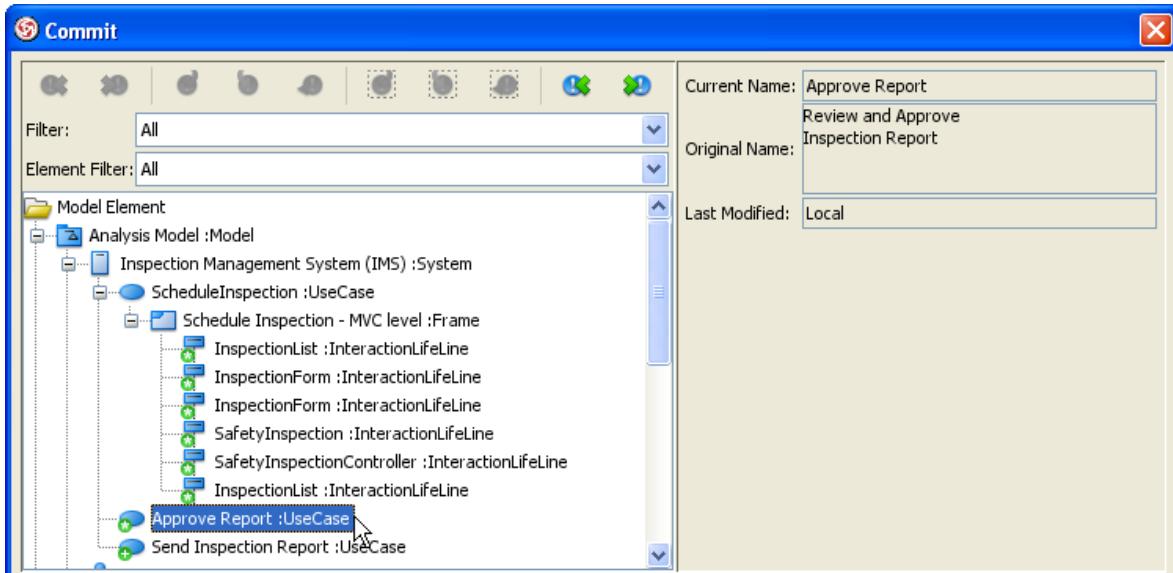


Figure 4-27 Review commit changes

5. On the bottom of dialog, click the **Preview** tab to visually preview the changes in diagram. The selected shape is highlighted in purple.

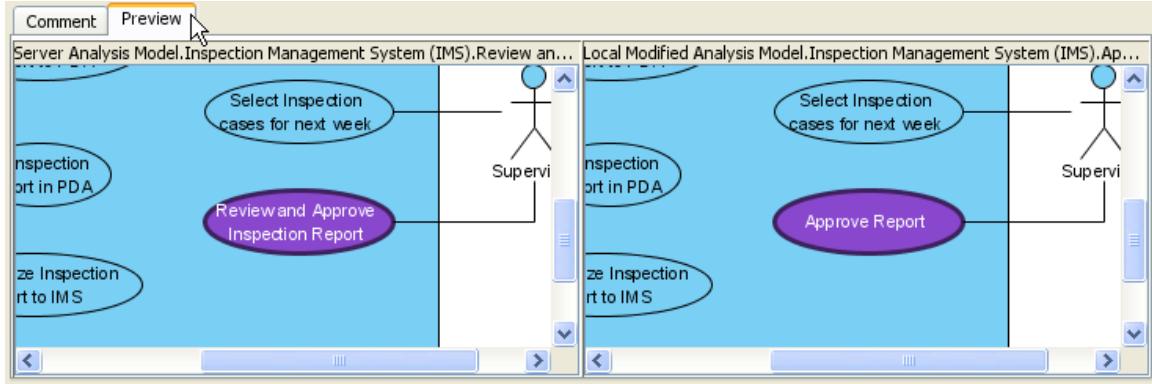


Figure 4-28 Preview commit changes

6. After review the changes, click the **Comment** tab and input the comment for commit. Click **OK** button to start commit.

The screenshot shows the 'Comment' tab of the 'Commit' dialog. It includes the following fields:

Project name:	Fire Safety Department	Checkout time:	29 Apr 2009, 04:32 PM
Checkout revision:	1	Latest server revision:	1

Below these fields is a text area for 'Edit the commit comment:' containing the text 'Modify Use Case Diagram'.

At the bottom, there is a preview area with the text '<Choose a previously entered comment>' and three buttons: **OK**, **Cancel**, and **Help**.

Figure 4-29 Input commit comment

Resolving conflicts

1. Modify the project.

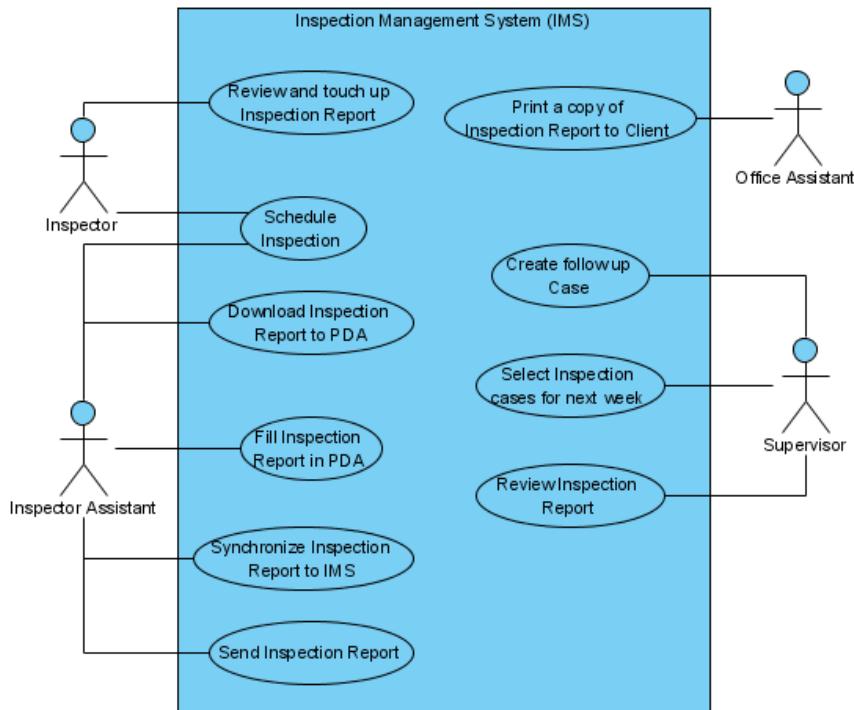


Figure 4-30 Modified project

2. Click **Commit** button on the **Teamwork Toolbar**.

3. In **Commit** dialog, you may found some shapes or model elements show with red icon. This indicate there is conflict when commit, it is caused by someone modified the same content and commit after you checkout. You can review the current value, original value (the value when you checkout), and conflict value (the value changed by other users). And dialog preview is disabled until you resolve the conflict.

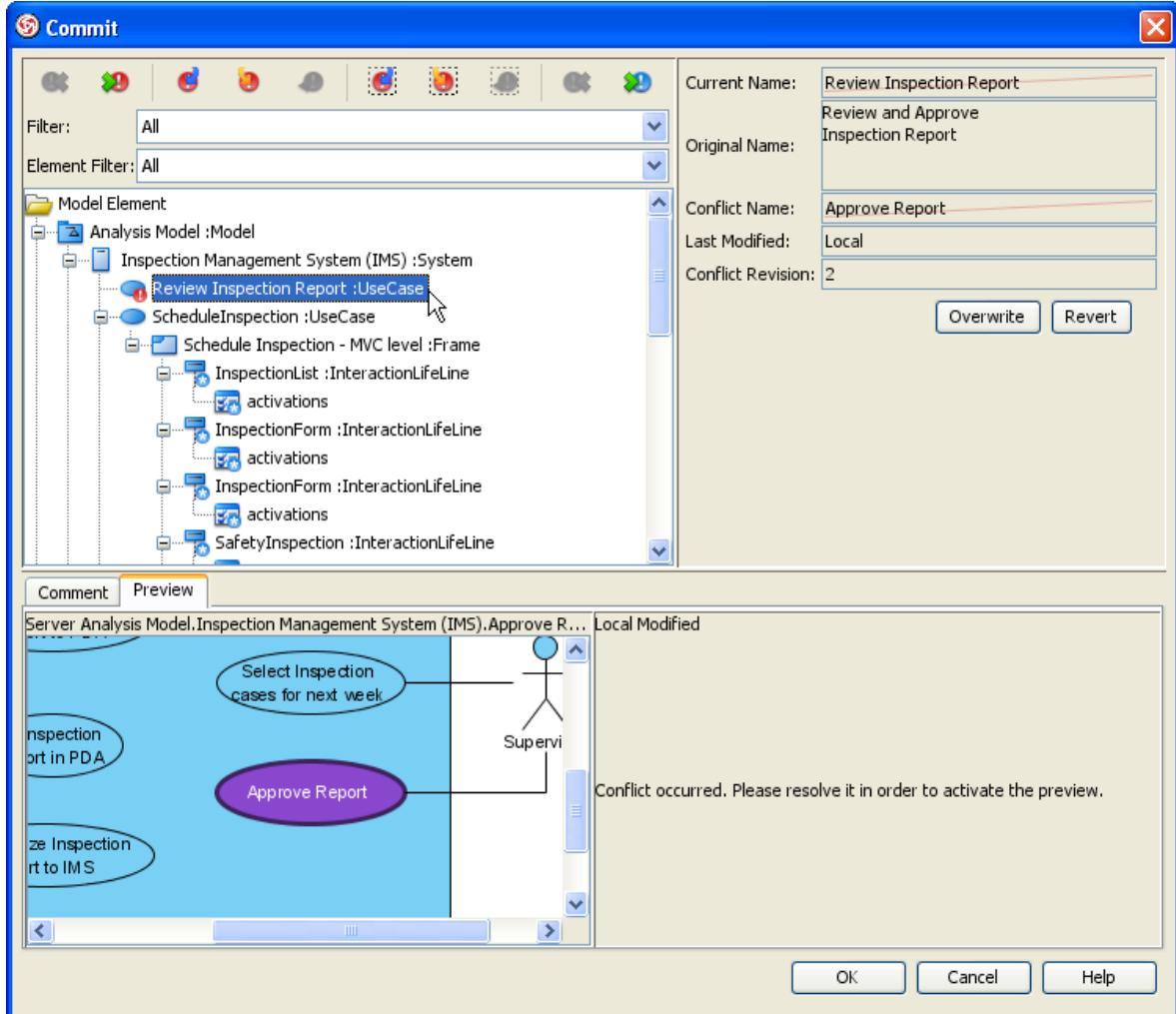


Figure 4-31 Commit with conflicts

Below is a description of buttons or fields in dialog box.

Button/Field	Description
	Move to the previous conflict in element tree.
	Move to the next conflict in element tree.
	Overwrite all conflicts.
	Revert all conflicts.
	Reset all conflicts.
	Overwrite the selected conflict in tree.
	Revert the selected conflict in tree.
	Reset the selected conflict in tree.
	Select the previous change in tree.
	Select the next change in tree.
Current Name	The value of name of the committing/updating copy.
Original Name	The value of name of the original copy.
Conflict Name	The value of name of the server copy.
Last Modified	The user who modified last time.
Conflict Revision	The revision that cause conflict with the committing/updating action.
[Overwrite]	Click on overwrite the conflict selected in tree.
[Revert]	Click on revert the conflict selected in tree.

4. Select the conflict element, click **Overwrite selected conflicts** button on the toolbar to overwrite other user's change.



Figure 4-32 Overwrite selected conflicts

Or click **Revert selected conflicts** to revert your own changes.



Figure 4-33 Revert selected conflicts

You can also click **Overwrite all conflicts** or **Revert all conflicts** to overwrite or revert all conflicts at once.



Figure 4-34 Overwrite/Revert all conflicts

5. After resolve conflict, the preview will be enabled to visualize the final result.

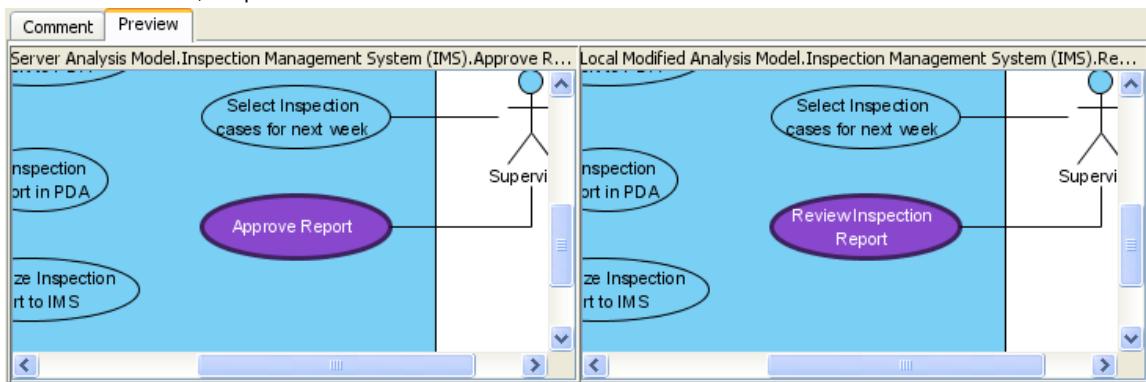


Figure 4-35 Preview resolved conflict

6. You can change your mind by click **Reset selected conflicts** or **Reset all conflicts** button. Then overwrite or revert the conflict again.



Figure 4-36 Reset conflicts

7. After all conflicts was resolved, you can now input comment and click **OK** button to commit.

Updating latest revision from perforce

Updating modification from perforce

1. Click **Update** button on **Teamwork Toolbar**.



Figure 4-37 Update from toolbar

2. Similar to commit, you can review the change and preview the diagram in **Update** dialog.

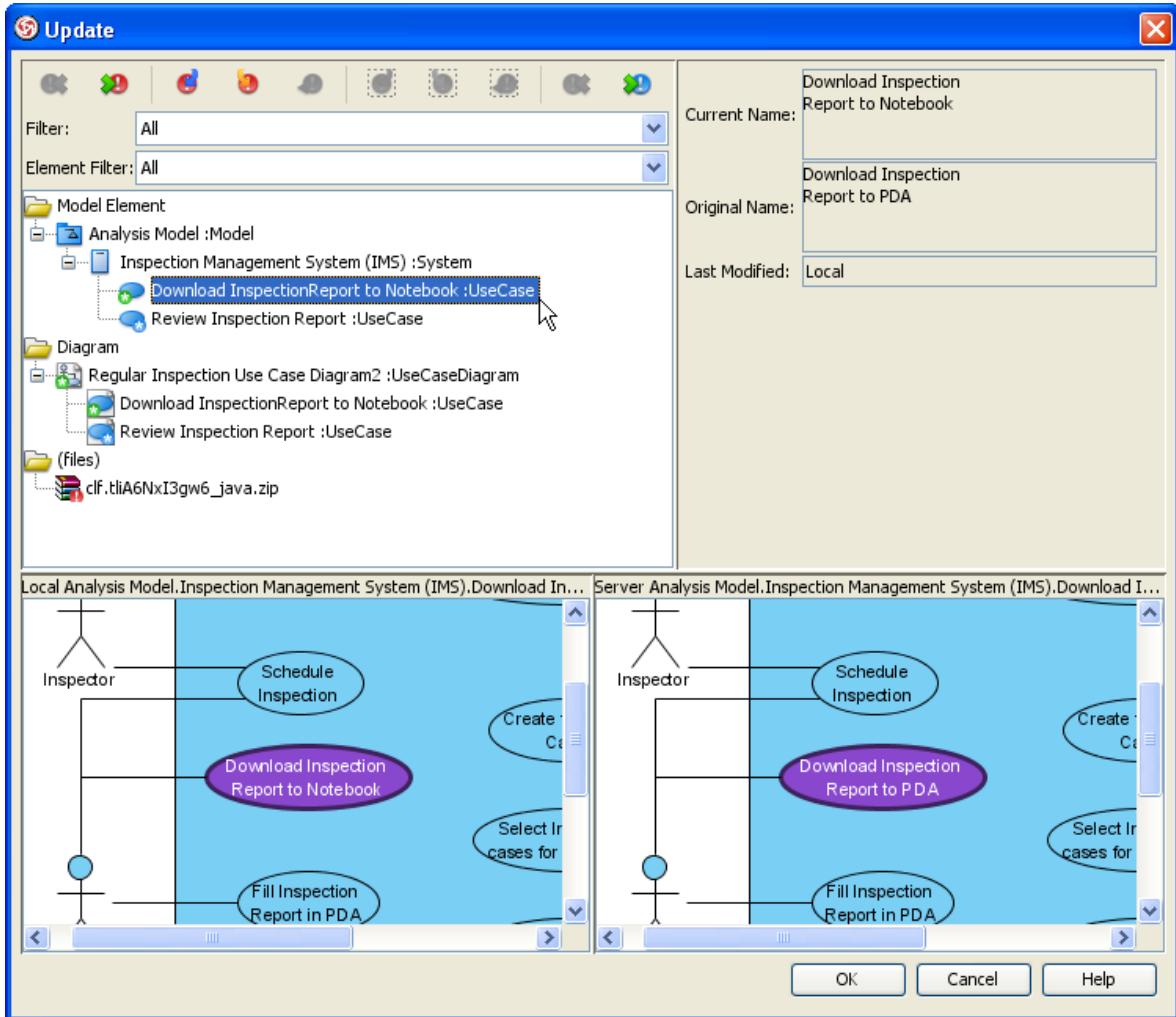


Figure 4-38 Update dialog

3. If there are conflicts, resolve it similar to commit, but the changes will apply to local project instead of commit to server immediately.

4. Click **OK** button to update. **VP-UML** will open the updated project.

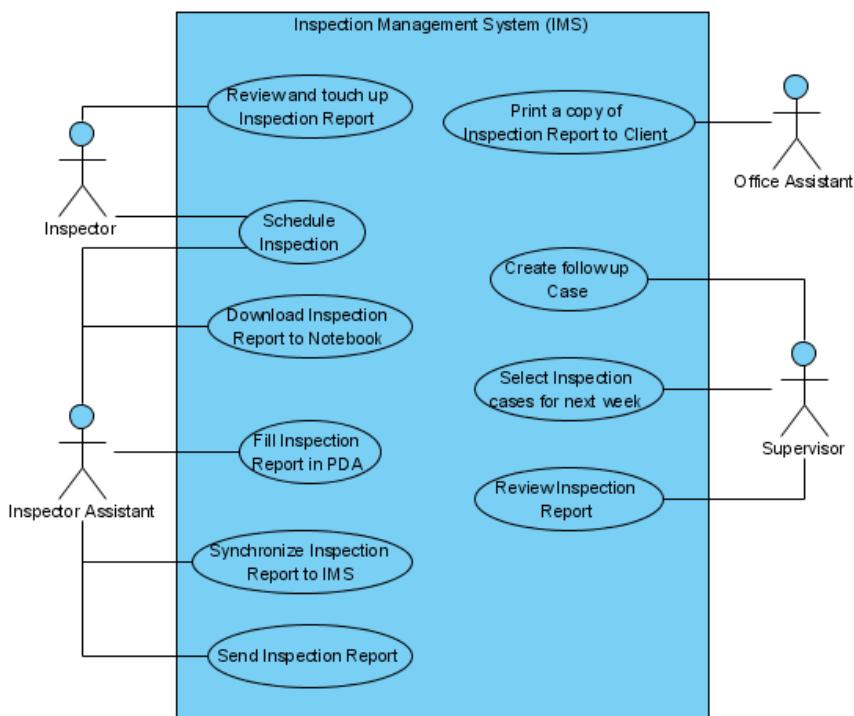


Figure 4-39 Updated project

Checking status

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Check Update** from menu.

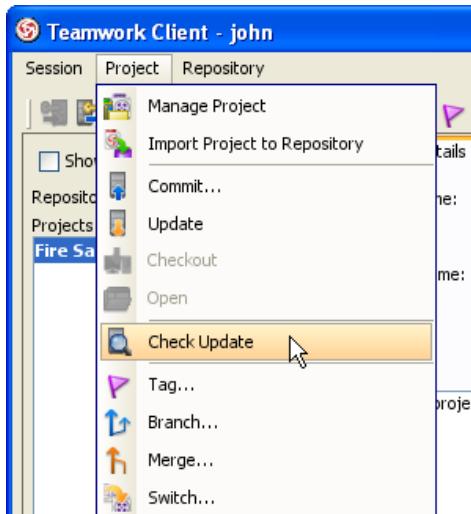


Figure 4-40 Update project from menu

Or click **Check Update** button from toolbar.



Figure 4-41 Update project from toolbar

4. You can see the status showing **Has update** or **Up-to-date**, it also indicate local project status (**local project not modified**) or (**local project modified**).

Status: Has update (local project not modified)

Figure 4-42 Project status

Reverting changes

Roll back local and uncommitted modifications

1. Modify the project.

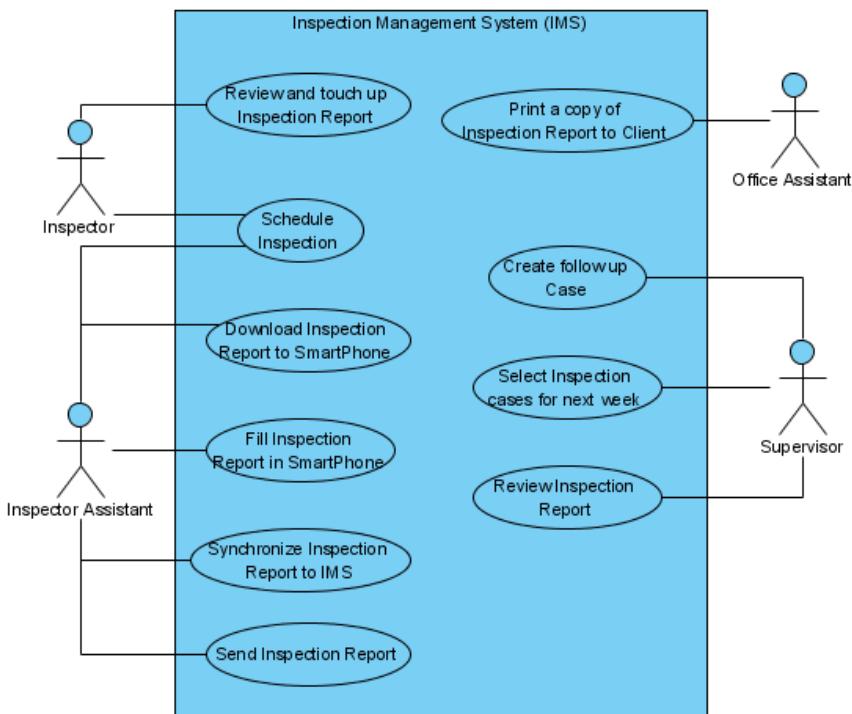


Figure 4-43 Modify project

2. Open **Teamwork Client** dialog.
3. Select the teamwork project in the list.
4. Select **Project > Revert...** from menu

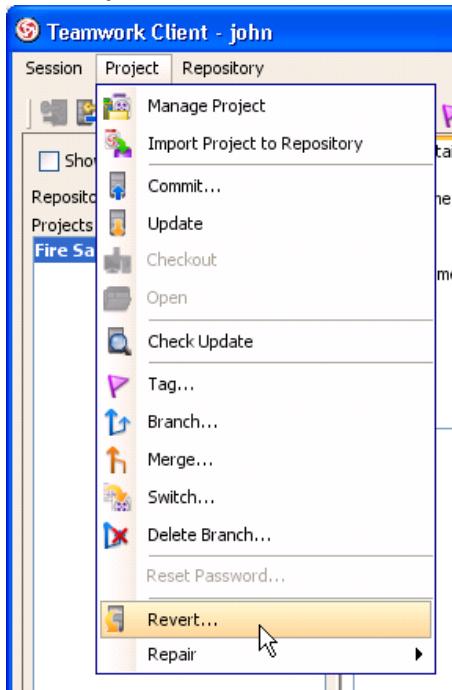


Figure 4-44 Revert from menu

Or click **Revert...** button from toolbar.



Figure 4-45 Revert from toolbar

5. Click **Yes** button from the confirmation dialog.
6. The reverted project opened in **VP-UML**.

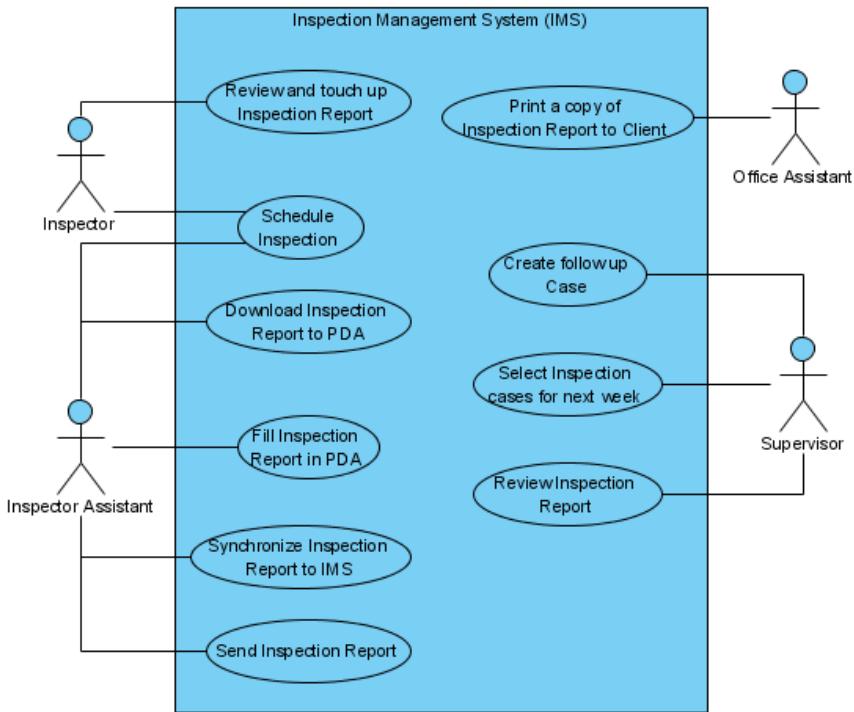


Figure 4-46 Reverted project

Roll back modifications of a past revision

If there were undesired changes made in past revision(s), such as wrong deletion of elements, you can undo the changes by reverting the revision(s) involved. Here are the steps:

1. Open project.

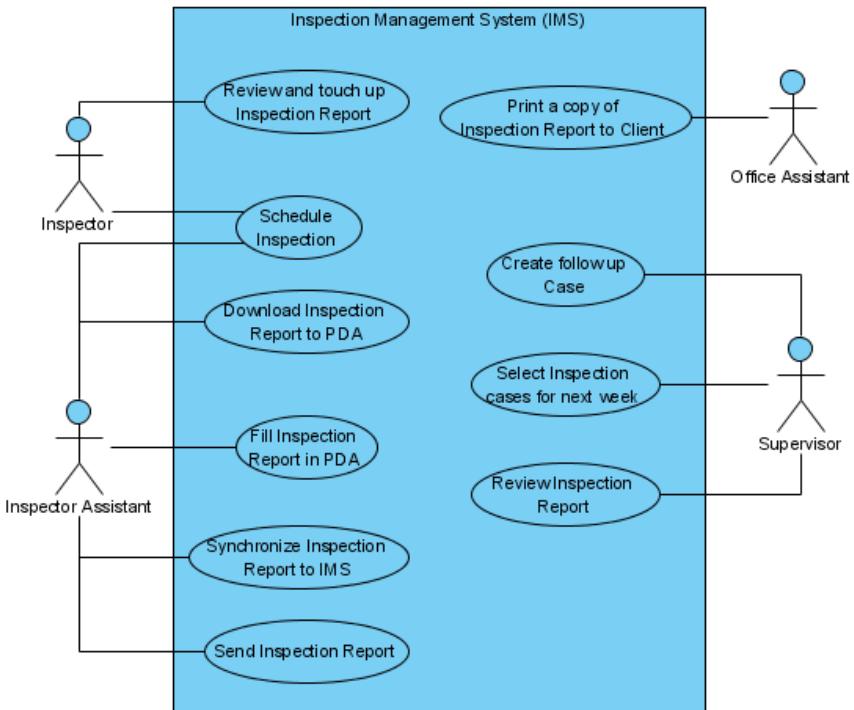


Figure 4-47 Open project

2. Open **Teamwork Client** dialog.
3. Select the teamwork project in the list.

4. Click the **Revisions** tab on the right of the project list.

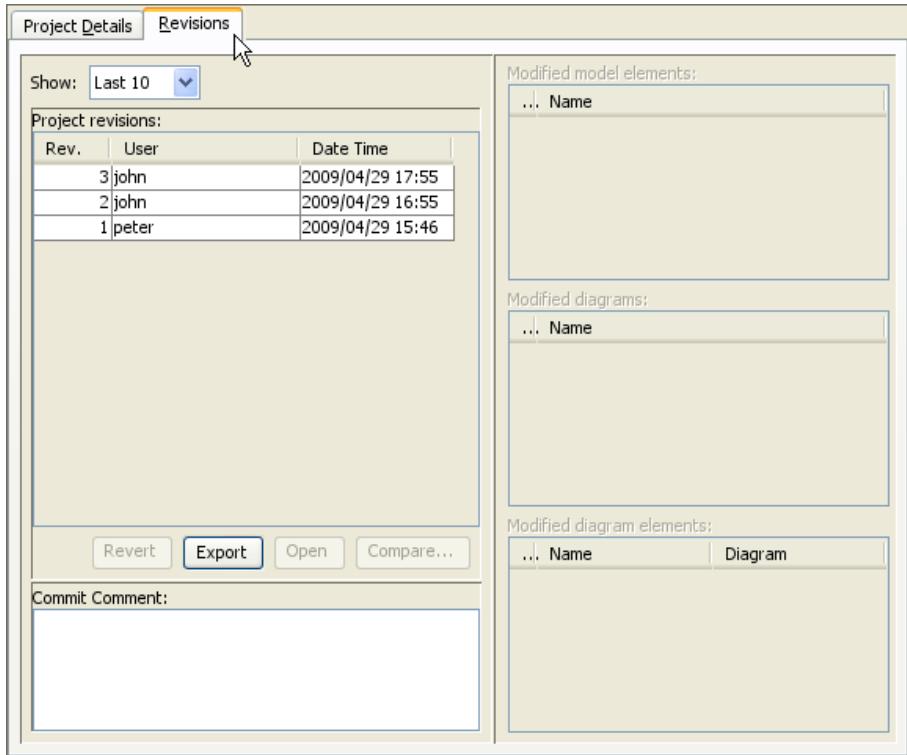


Figure 4-48 Revision tab

5. Select the revision(s) to have its changes rolled back.

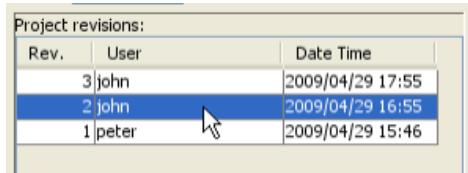


Figure 4-49 Select revision

6. Click **Revert** button.



Figure 4-50 Revert button

NOTE: You may make multiple selection by selecting one revision, pressing the **Control** key, and selecting the rest. Note that a non-consecutive revision selection is not revertable.

7. The **Commit** dialog show a list of shapes and model elements reverted for review the changes and preview diagram, input the comment and click **OK** button to commit the revert.

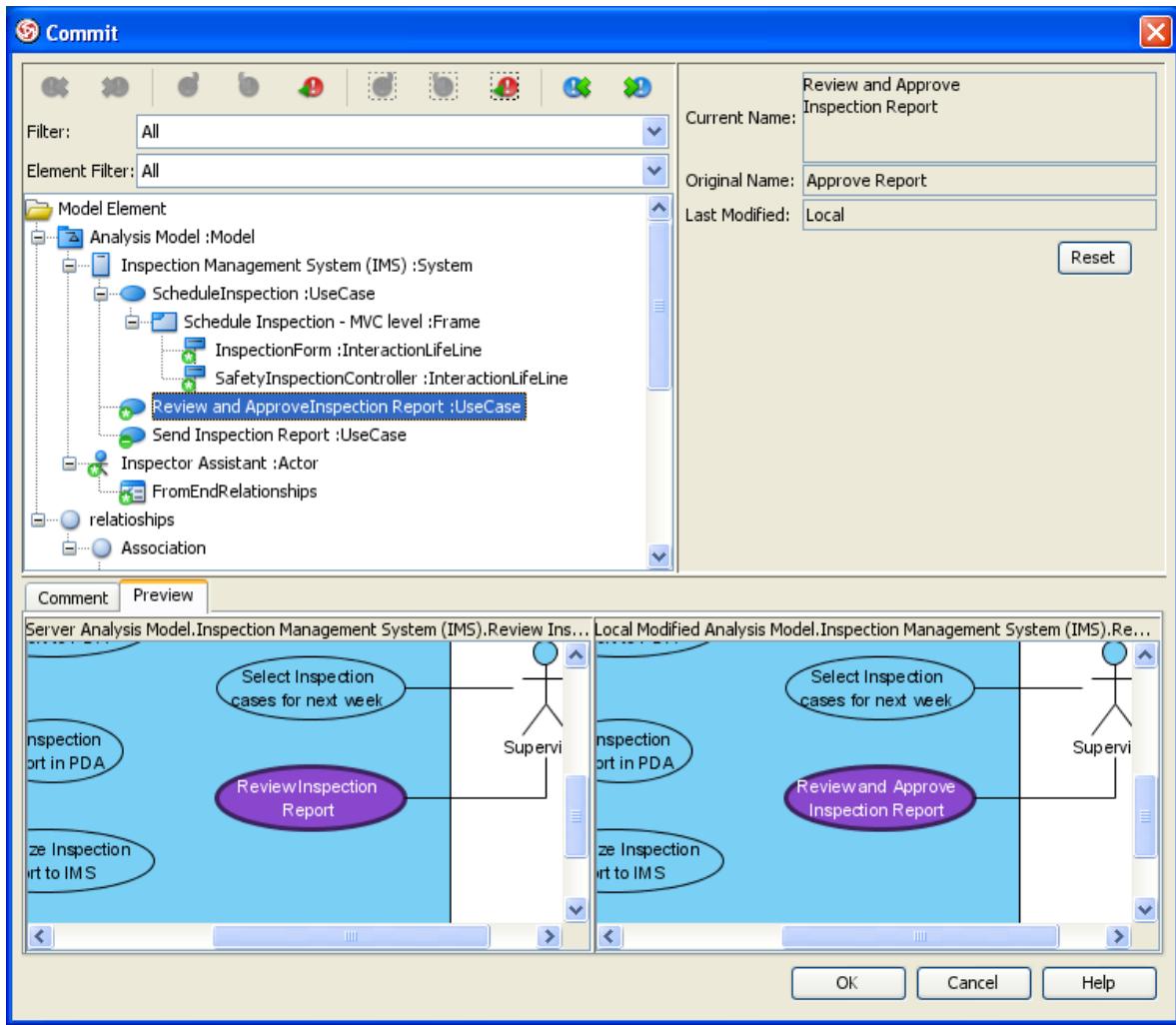


Figure 4-51 Commit dialog with reverted changes

8. A new revision with the reverted changes was created in server, and opened in VP-UML.

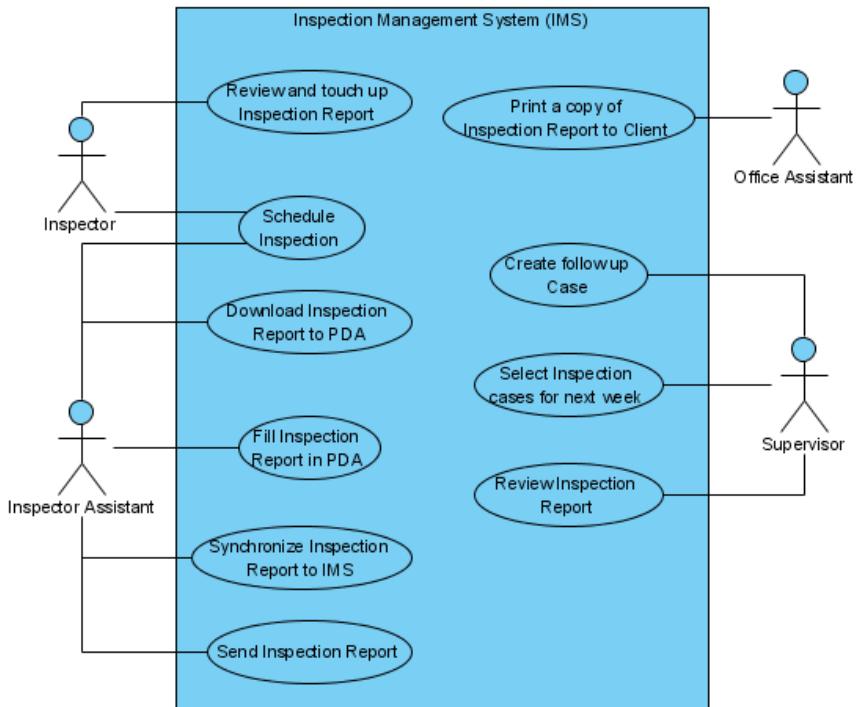


Figure 4-52 Reverted project

Browsing change histories (old revisions)

Checkout old revisions

1. Open Teamwork Client dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select the revision to open.
5. Click Open button.



Figure 4-53 Open button

6. Selected revision opened in VP-UML.

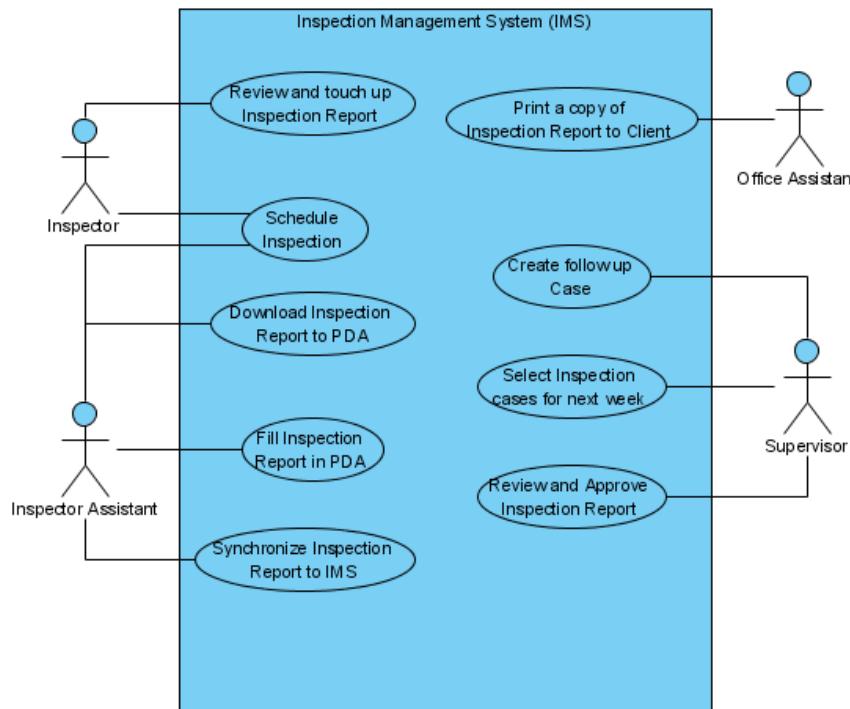


Figure 4-54 Open selected revision

Showing differences between revisions visually

1. Open Teamwork Client dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select a revision for compare.
5. Select another revision by **Ctrl+Click**.

Project revisions:		
Rev.	User	Date Time
4	john	2009/04/30 09:28
3	john	2009/04/29 17:55
2	john	2009/04/29 16:55
1	peter	2009/04/29 15:46

Figure 4-55 Select two revisions

6. Click the **Compare...** button.



Figure 4-56 Compare button

7. Similar to **Commit** and **Update** dialog, the **Compare Projects** dialog show a list of differences between the selected revisions. You can also view the differences visually in diagram on the preview tab.

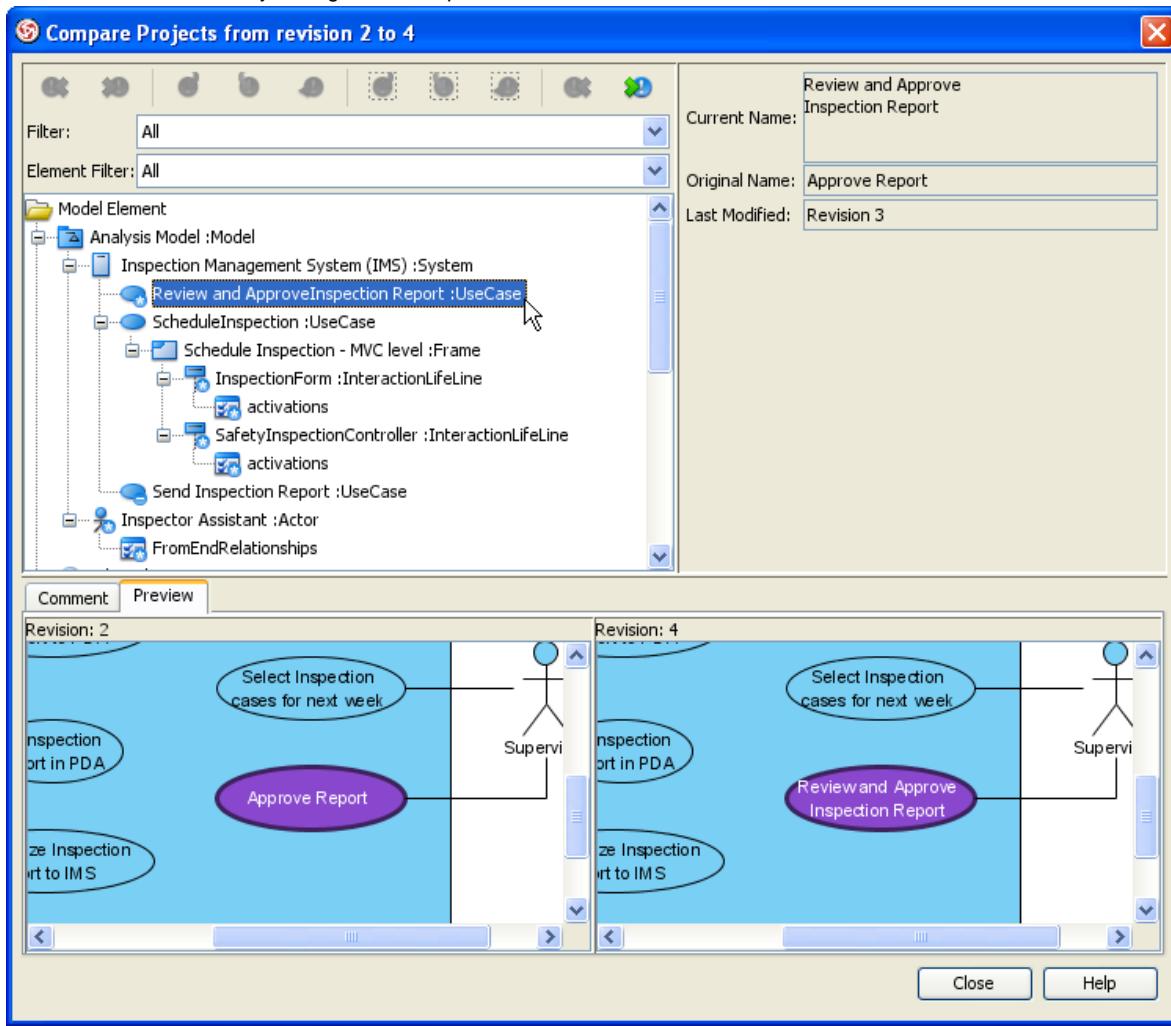


Figure 4-57 Compare differences

Export multiple revisions to local

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Click the **Revisions** tab.
4. Select multiple revisions for export, by **Ctrl+Click** or **Shift+Click**.

Project revisions:		
Rev.	User	Date Time
4	john	2009/04/30 09:28
3	john	2009/04/29 17:55
2	john	2009/04/29 16:55
1	peter	2009/04/29 15:46

Figure 4-58 Select multiple revisions

5. Click the **Export** button.



Figure 4-59 Export buttons

6. Select **Export selected revisions...** from the popup.

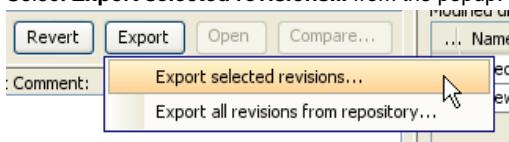


Figure 4-60 Export selected revisions

7. Select the directory to save the exported project revisions.

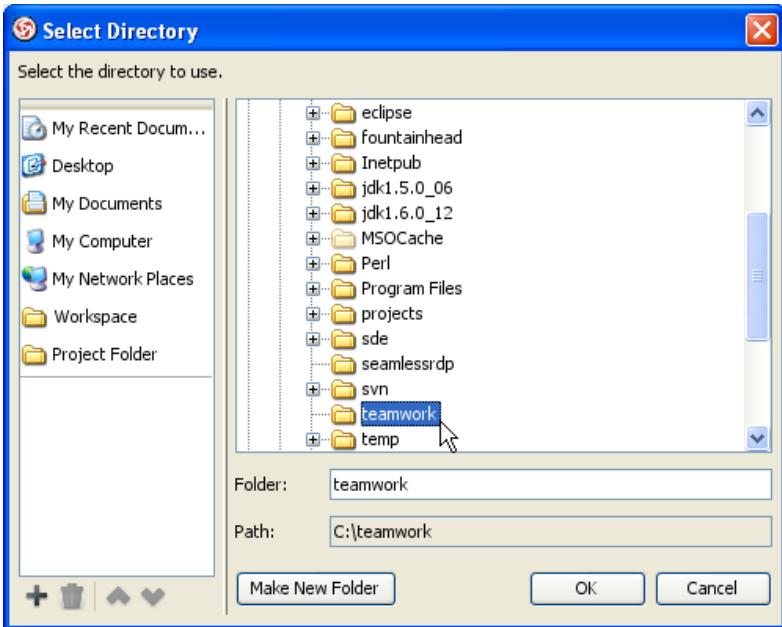


Figure 4-61 Select directory

8. You'll find the selected revisions was exported to the select directory.

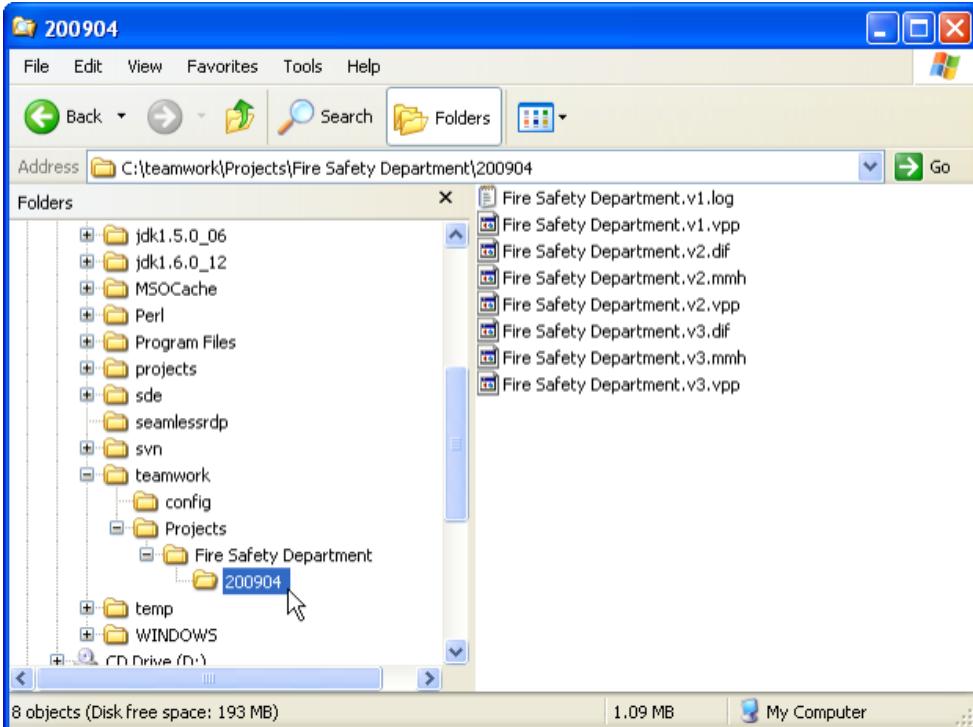


Figure 4-62 Exported revisions

Isolating last long modifications with branches

Creating branch

1. Open Teamwork Client dialog.
2. Select the teamwork project in the list.
3. Select Project > Branch... from menu

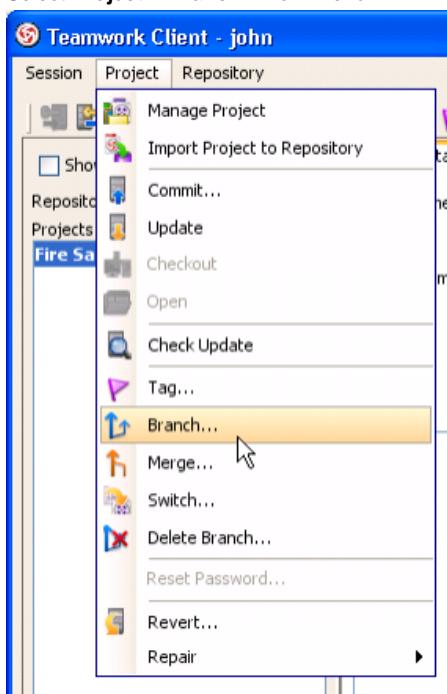


Figure 4-63 Branch from menu

Or click Branch... button from toolbar.



Figure 4-64 Branch from toolbar

4. Fill in the Branch Name in Create Branch dialog.

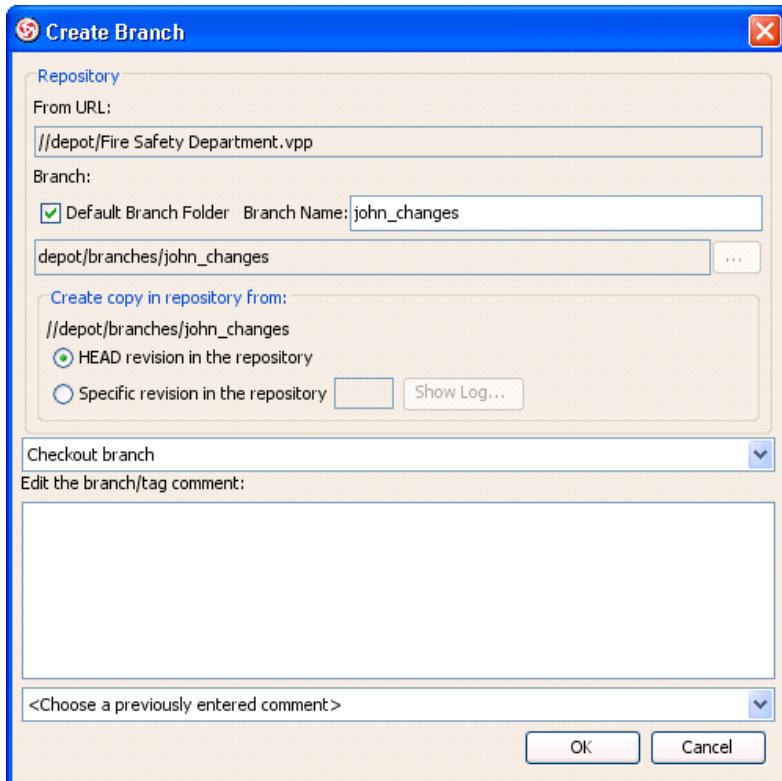


Figure 4-65 Create Branch dialog

5. Select **Stay in trunk** in the list.



Figure 4-66 Stay in trunk

Switch local copy between branches

1. Select **Project > Switch...** from menu

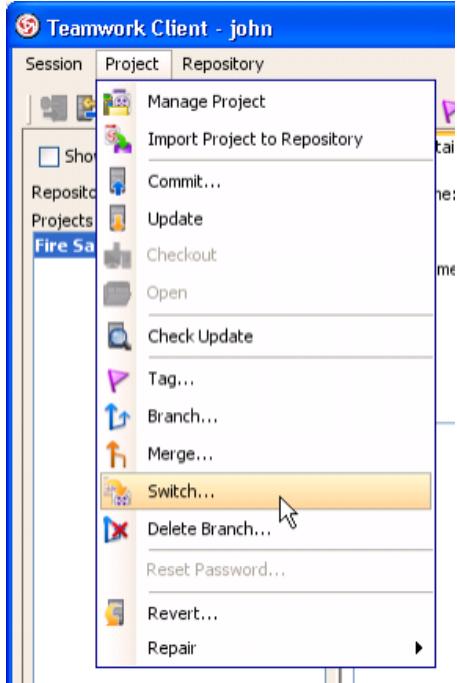


Figure 4-67 Switch from menu

Or click **Switch...** button from toolbar.



Figure 4-68 Switch from toolbar

2. Click ... to browser the path.

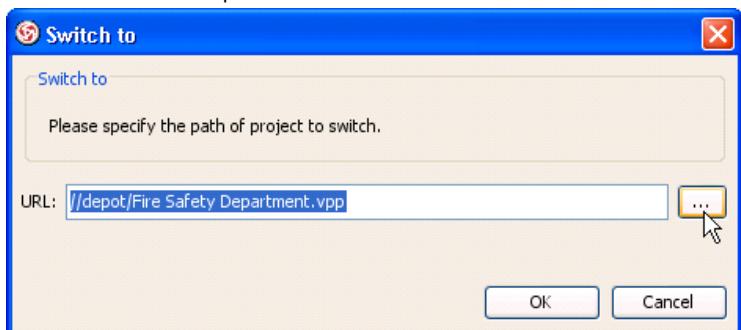


Figure 4-69 Select branch

3. Select the branch to switch, and click **OK**.

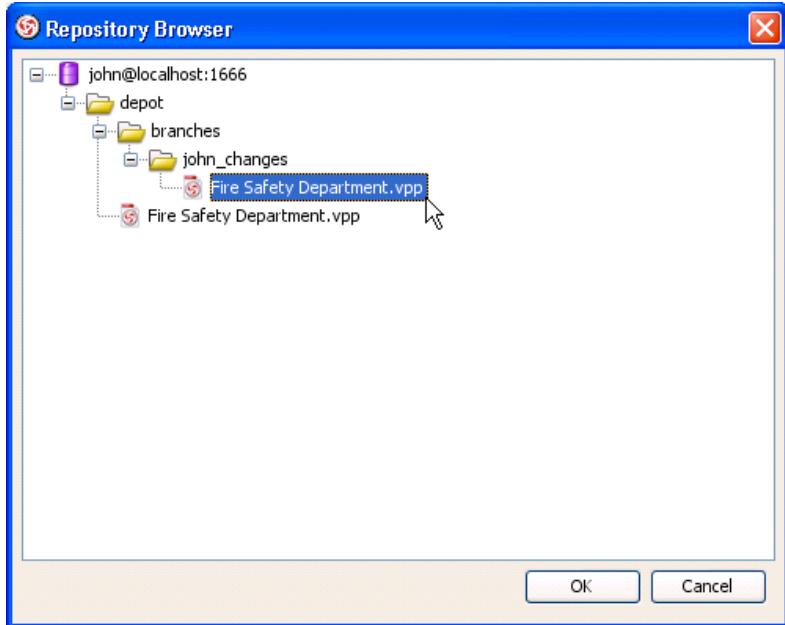


Figure 4-70 Select branch path

4. The branch is opened in **VP-UML**.

Merging from trunk to branch

1. Open, modify and commit trunk project.

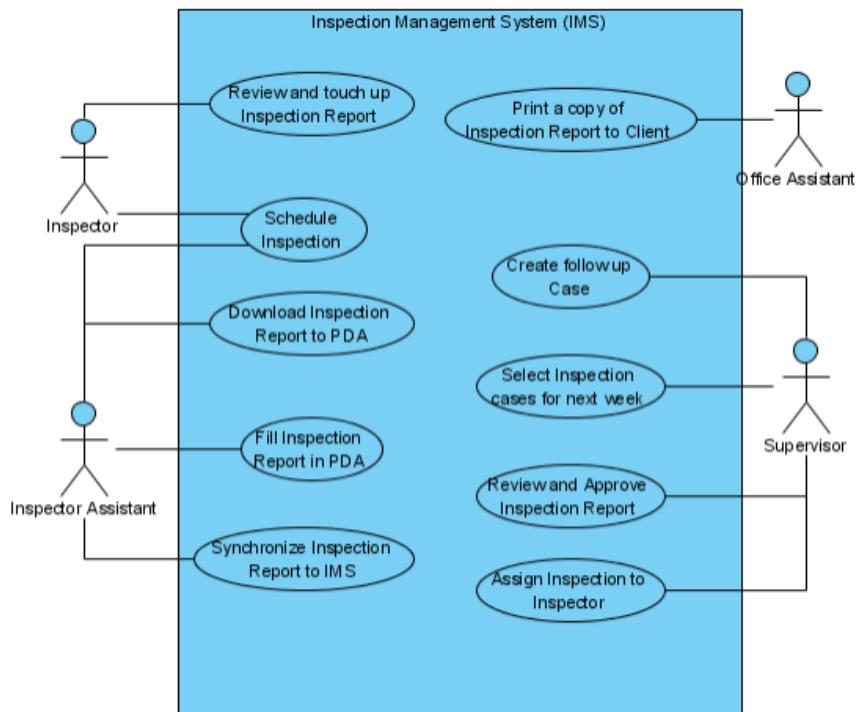


Figure 4-71 Modified trunk project

2. Open branch project.

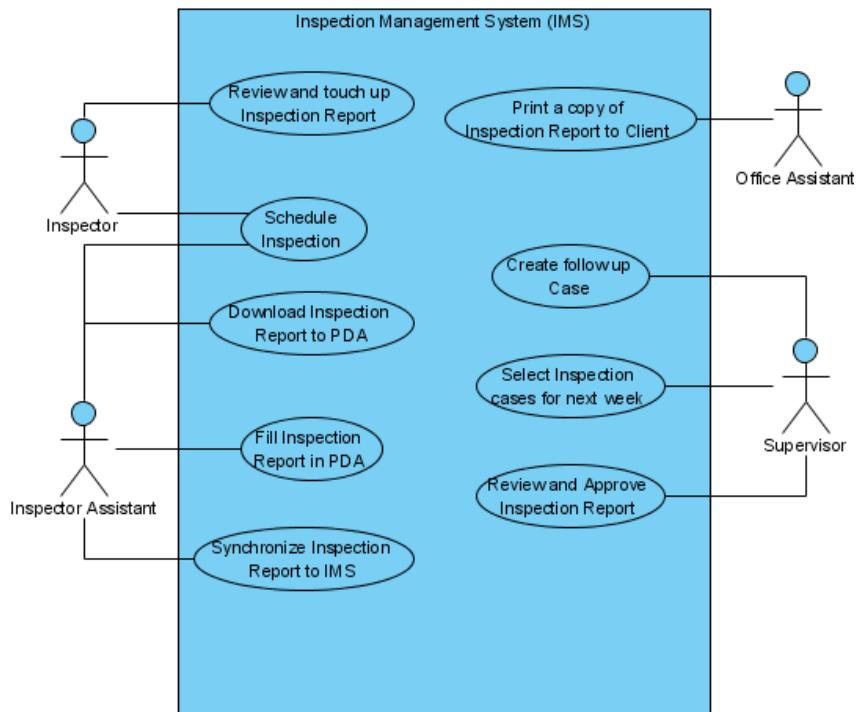


Figure 4-72 Branch project

3. Select **Project > Merge...** from menu.

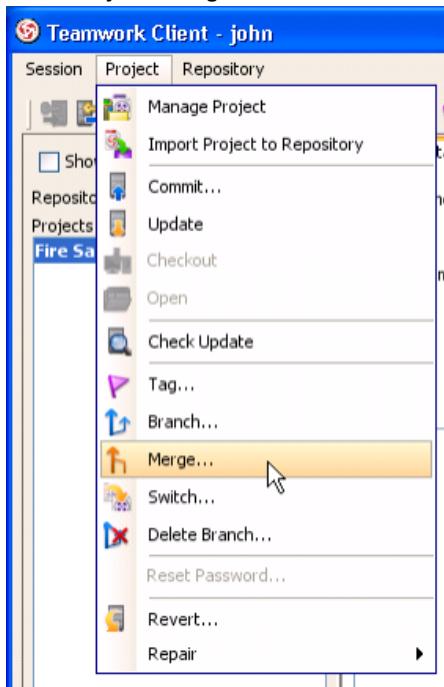


Figure 4-73 Merge from menu

Or click **Merge...** button from toolbar.



Figure 4-74 Merge from toolbar

4. Select the trunk path(*depot/FireSafetyDepartment.vpp*) as **From** in the **Merge** dialog, and press **OK**.

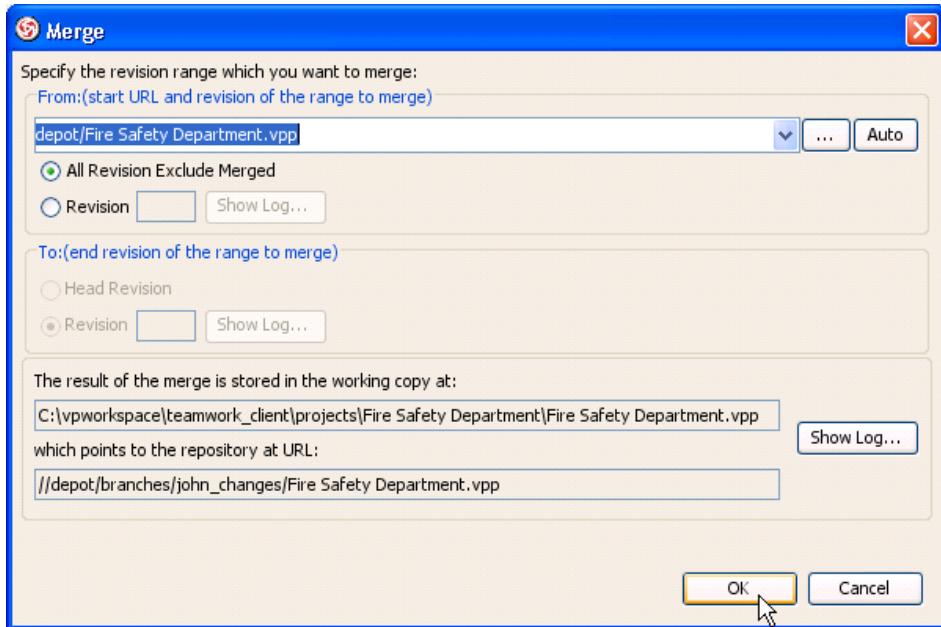


Figure 4-75 Merge dialog

5. The **Merge Project** dialog show a list of changes similar to **Commit** and **Update** dialog.

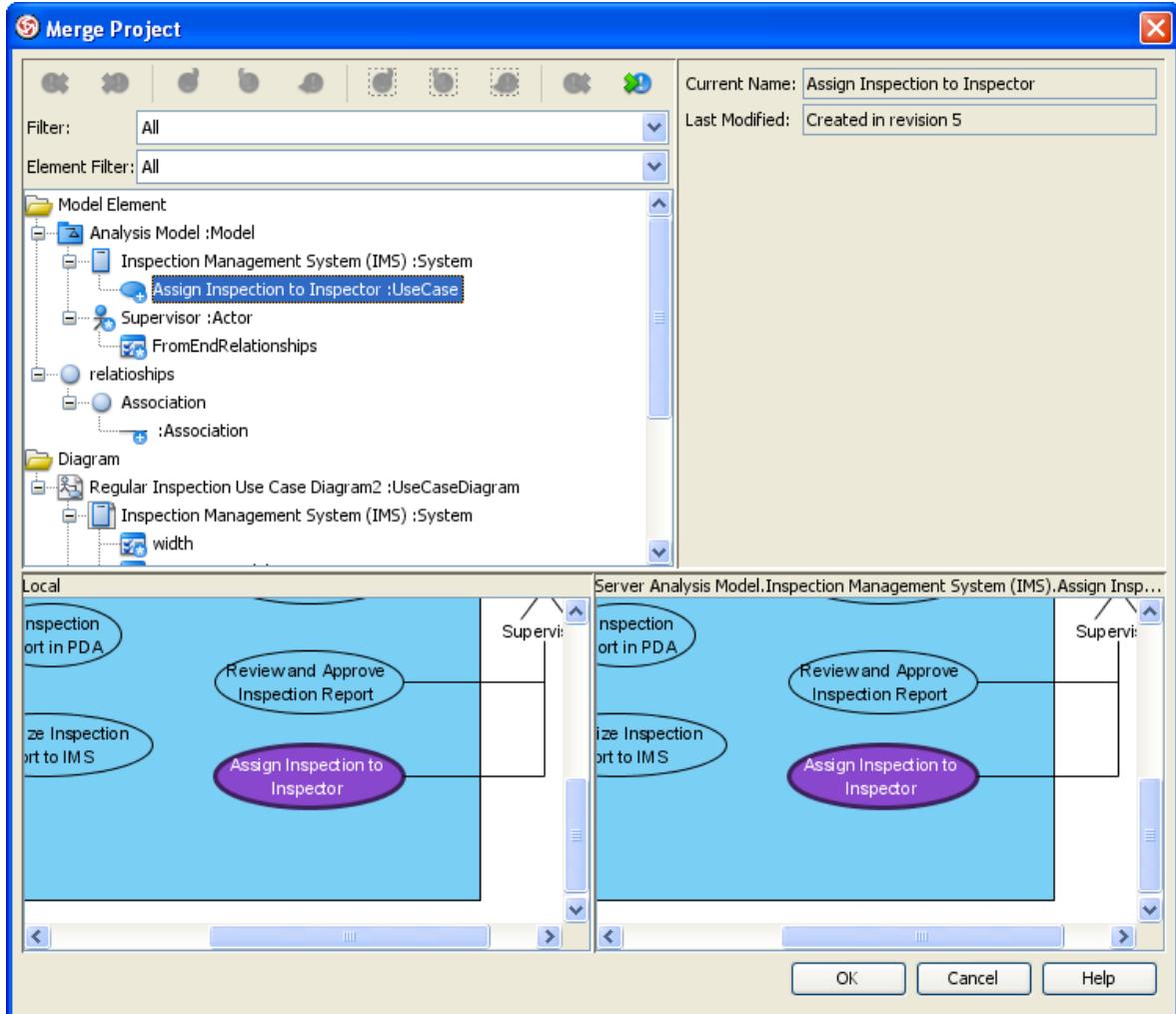


Figure 4-76 Merge project dialog

6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.

7. The branch project now contains the changes from trunk.

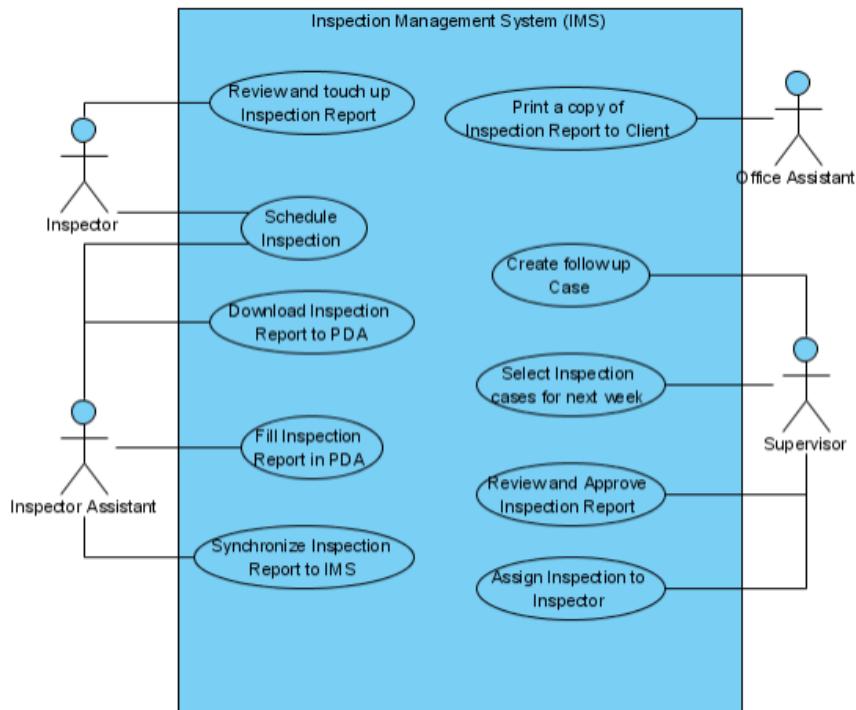


Figure 4-77 Branch project merged from trunk

Merging from branch to trunk

1. Open, modify and commit branch project.

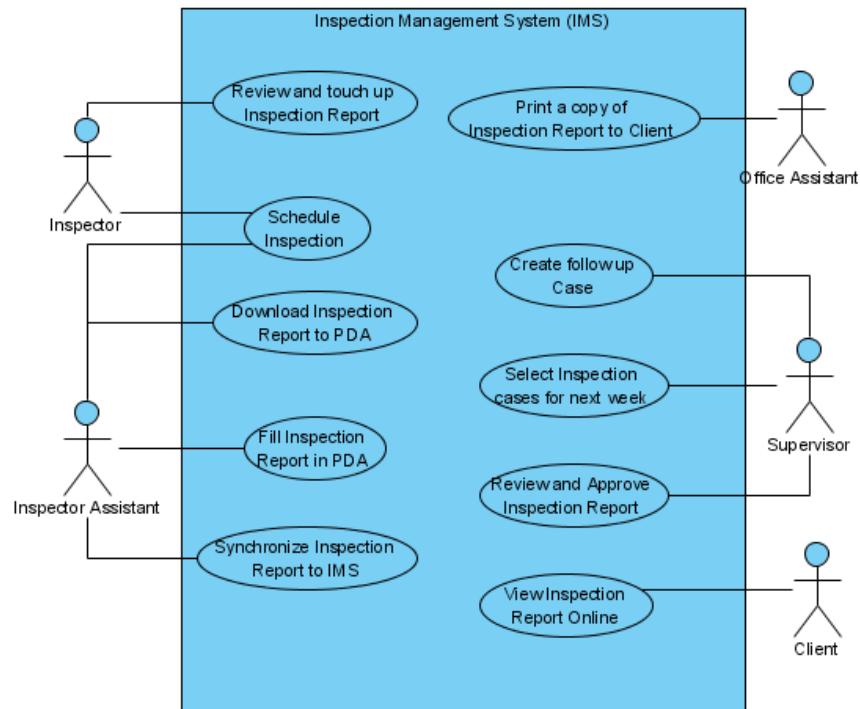


Figure 4-78 Modified branch project

2. Open trunk project.

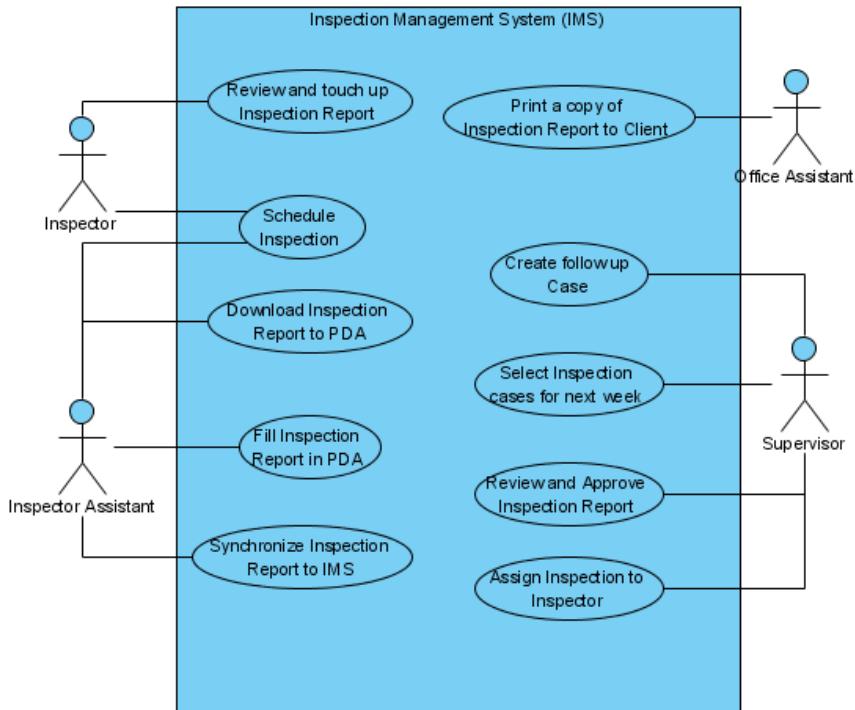


Figure 4-79 Trunk project

3. Select **Project > Merge...** from menu, or click **Merge...** button from toolbar.
 4. Select the branch path(e.g //depot/branches/john_changes/FireSafetyDepartment.vpp) as **From** in the **Merge** dialog, and press **OK**.

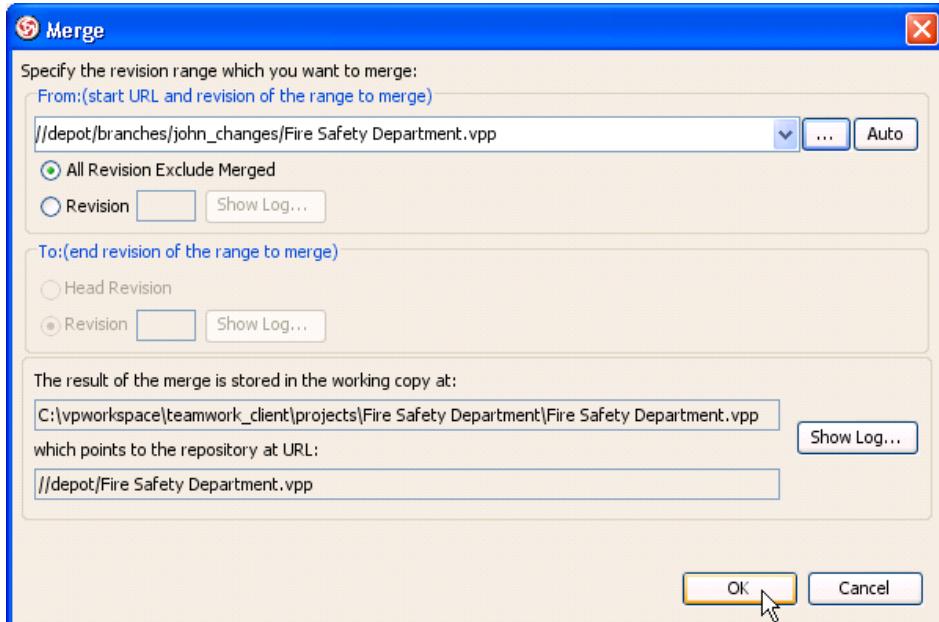


Figure 4-80 Merge dialog

5. The **Merge Project** dialog show a list of changes similar to **Commit** and **Update** dialog.
 6. Finally, review the changes in **Commit** dialog. Input comment and click **OK** to commit.

7. The trunk project now contains the changes from branch.

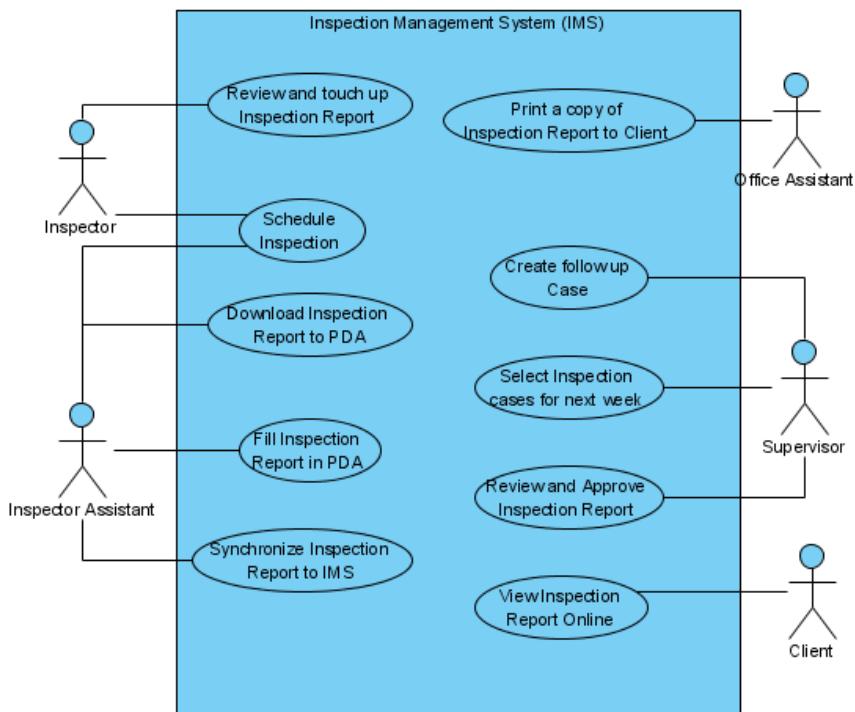


Figure 4-81 Trunk project merged from branch

Delete branch

- Select Project > Delete Branch... from menu

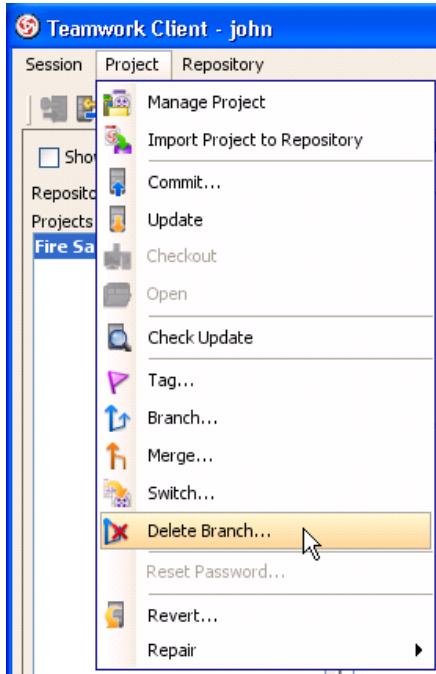


Figure 4-82 Delete Branch from menu

Or click Delete Branch ... button from toolbar.



Figure 4-83 Delete Branch from toolbar

2. Expand the repository and folder node, select the branch to delete. Click **OK** button to continue.

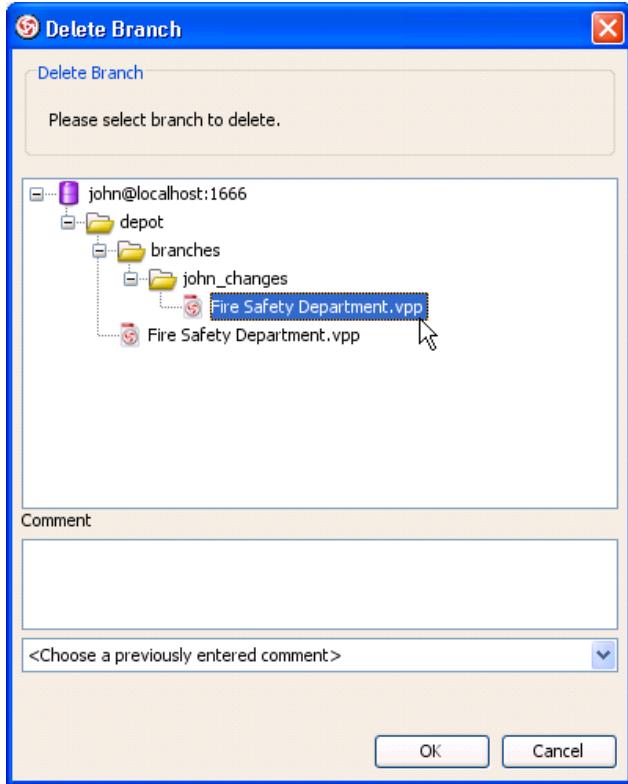


Figure 4-84 Delete branch dialog

3. Click **Yes** button on confirmation dialog.

Marking release or milestone with tags

Tags and branches are almost the same, with the only difference - tags are read only. Creating read only tags ensure you are able open the release version project later.

1. Open **Teamwork Client** dialog.
2. Select the teamwork project in the list.
3. Select **Project > Tag...** from menu

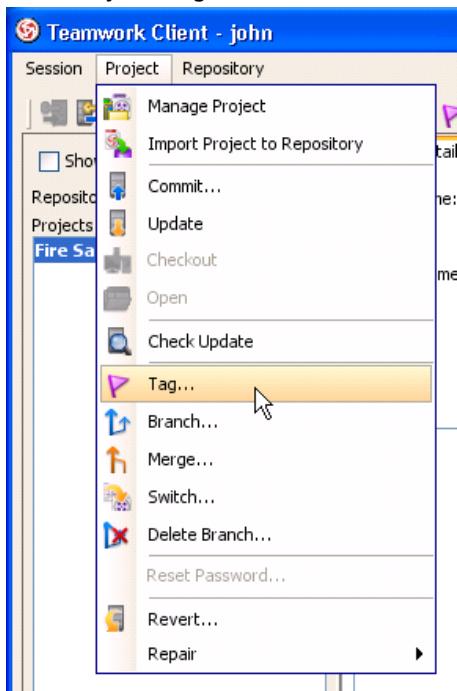


Figure 4-85 Tag from menu

Or click **Branch...** button from toolbar.

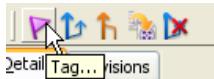


Figure 4-86 Tag from toolbar

4. Fill in the Tag Name in **Create Tag** dialog.

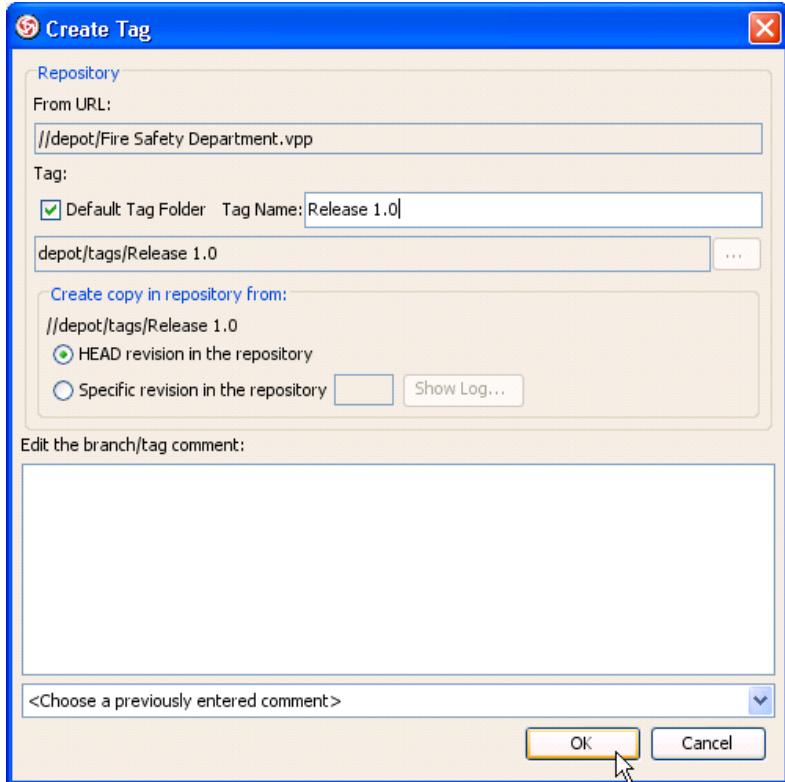


Figure 4-87 Create Tag dialog

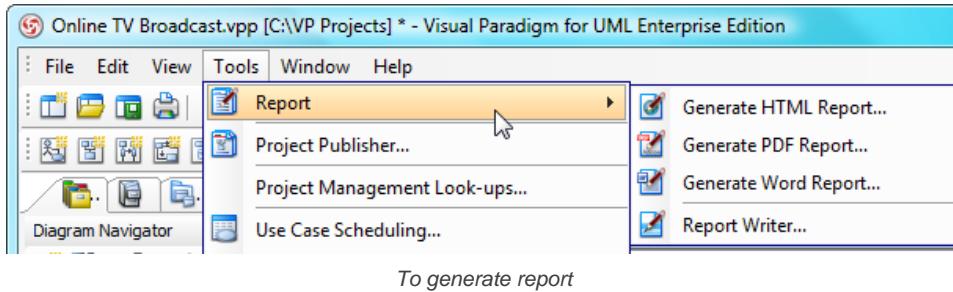
5. Click **OK** button, the tag will be created.

Generating 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. They differ in file format, but have the same layout. In this chapter, we will go through the core steps in report generation.

To generate a report:

1. Select **Tools > Report** from the main menu. Then select **Generate HTML Report...**, **Generate PDF Report...** or **Generate Word Report...** depending on the type of report you want to generate.

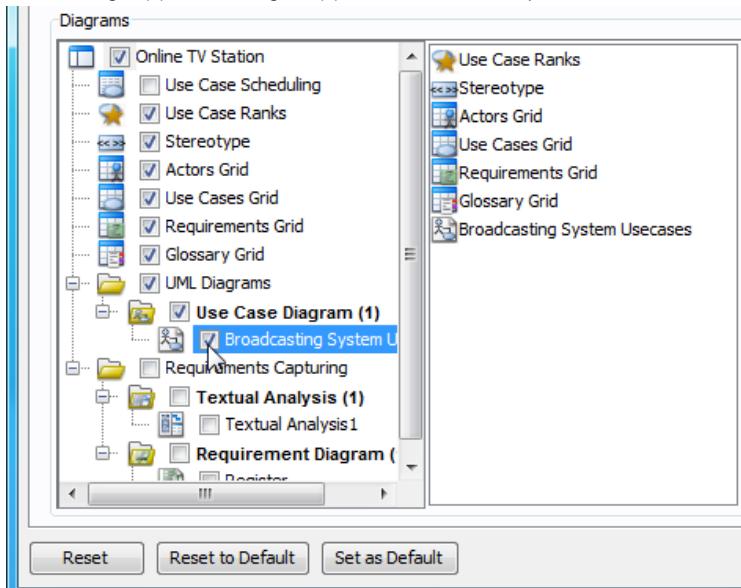


To generate report

2. In the **Generate HTML/PDF/Word** dialog box, fill in the output path where the report should be generated to.

NOTE: For HTML report, specify the folder of the HTML files to be generated.
For PDF report, specify the file path of PDF file (*.pdf) to be generated.
For MS Word report, specify the file path of the document file (*.docx) to be generated.

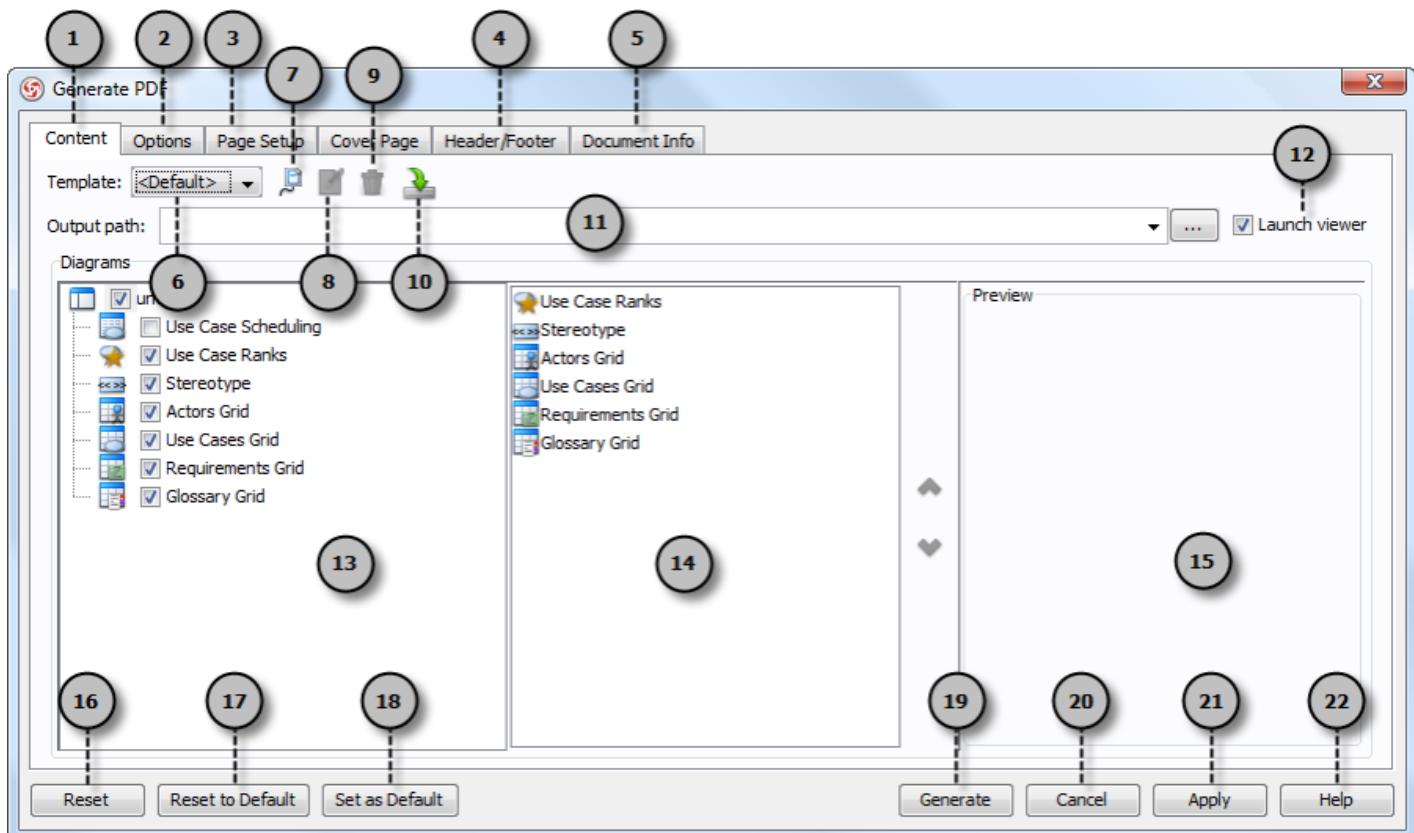
3. Select the grid(s) and/or diagram(s) to be included in report.



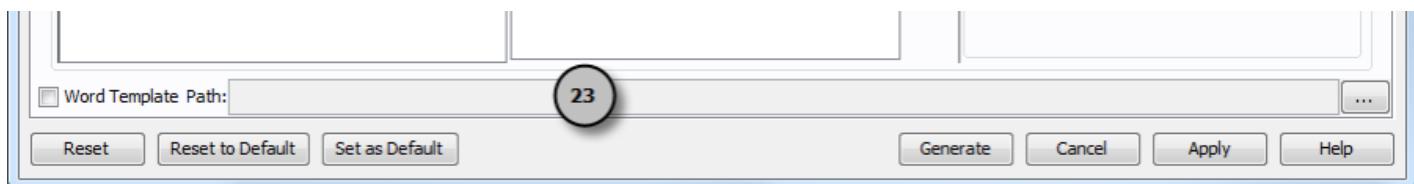
Select diagram to include in report

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 report generation dialog box



Overview of report generation dialog box



The bottom part of report generation dialog box for MS Word report

No.	Name	Description
1	Content	The main page of report generation that lets you select the diagram(s) to generate report.
2	Options	Configurable options for detailed report configuration.
3	Page Setup	The setup of layout of report.
4	Header/Footer	The content of header and footer.
5	Document info	To define document info.
6	Template	You can define a template for report 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 with generation with the template. For details about report customization, read the next chapter.
7	Use external template	Click to link to an external template file.
8	Edit template	Click to edit the template selected in the drop down menu of Template .
9	Delete template	Click to delete the template selected in the drop down menu of Template .
10	Import template	Click to import a template file into the current project.
11	Output path	The output path of report to be generated.
12	Launch viewer	Check to open the report automatically after generation.
13	Available diagram list	The list of diagrams in opening project.
14	Selected diagram list	The list of diagrams selected to generate report.
15	Preview	Preview of diagram being selected in the list of selected diagram

16 Reset	Reset changes made in this dialog box
17 Reset to default	Reset changes made in this dialog box to default settings.
18 Set as default	Set the settings in this dialog box to default.
19 Generate	Click to generate report.
20 Cancel	Click to close the report dialog box.
21 Apply	Click to apply the changes made in report, causing reopening of this dialog box to restore the applied settings.
22 Help	Click to read the help contents.
23 Word template path	Available only to MS Word report generation, this option enables you to specify the path of MS Word document file that you want the generator to use as template. Report generator will append the template file content in front of generated report. 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 <i>Generating MS Word report with template (MS Word report only)</i> .

Description of report generation dialog box

Generating MS Word report with template (MS Word report only)

At the bottom of the MS Word report generation dialog box, 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 report. During report 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 report.

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
ActorID	1

Relationships

Unnamed Association

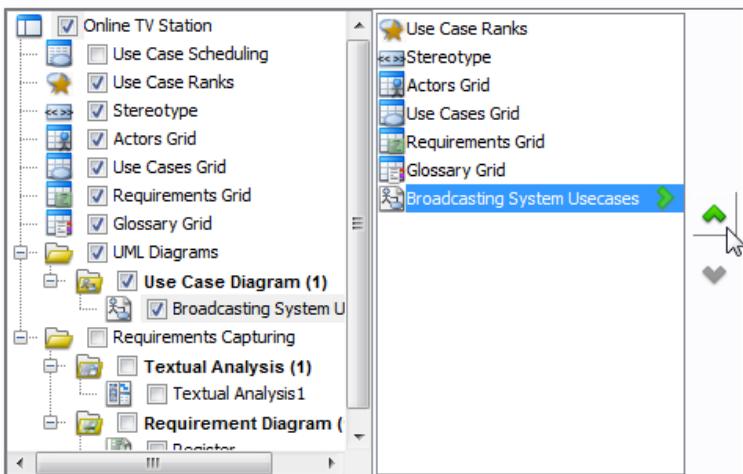
To	Name	Value
	End Model Element	Compose Mail
	Author	Peter
	Create Date Time	Feb 9, 2010 8:30:27 AM
	Last Modified	Feb 9, 2010 8:38:14 AM

A generated report with style defined in template applied

Sorting diagrams in report

By default, diagrams show in report follows the order defined in diagram tree in the **Generate PDF** dialog box. 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 dialog box, and click or to sort.

Diagrams



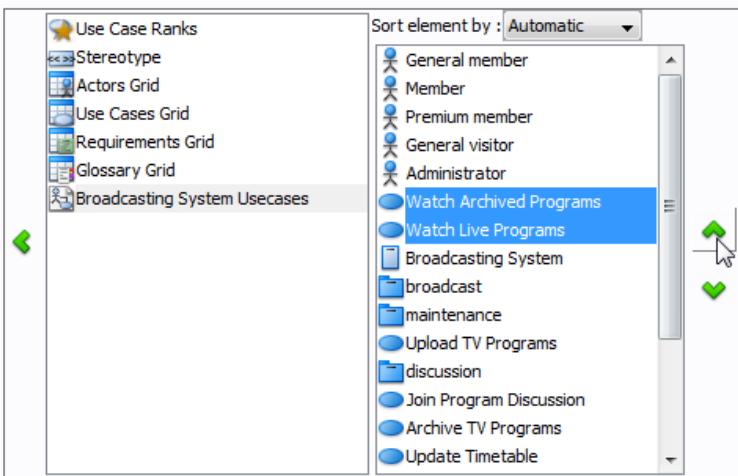
Reorder diagram

Sorting shapes in report

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.

Diagrams



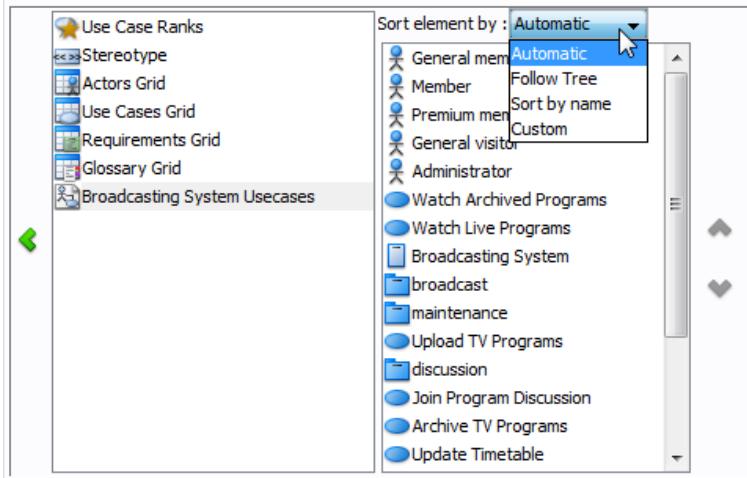
Sort shapes

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.

Diagrams



Select the way to sort shape

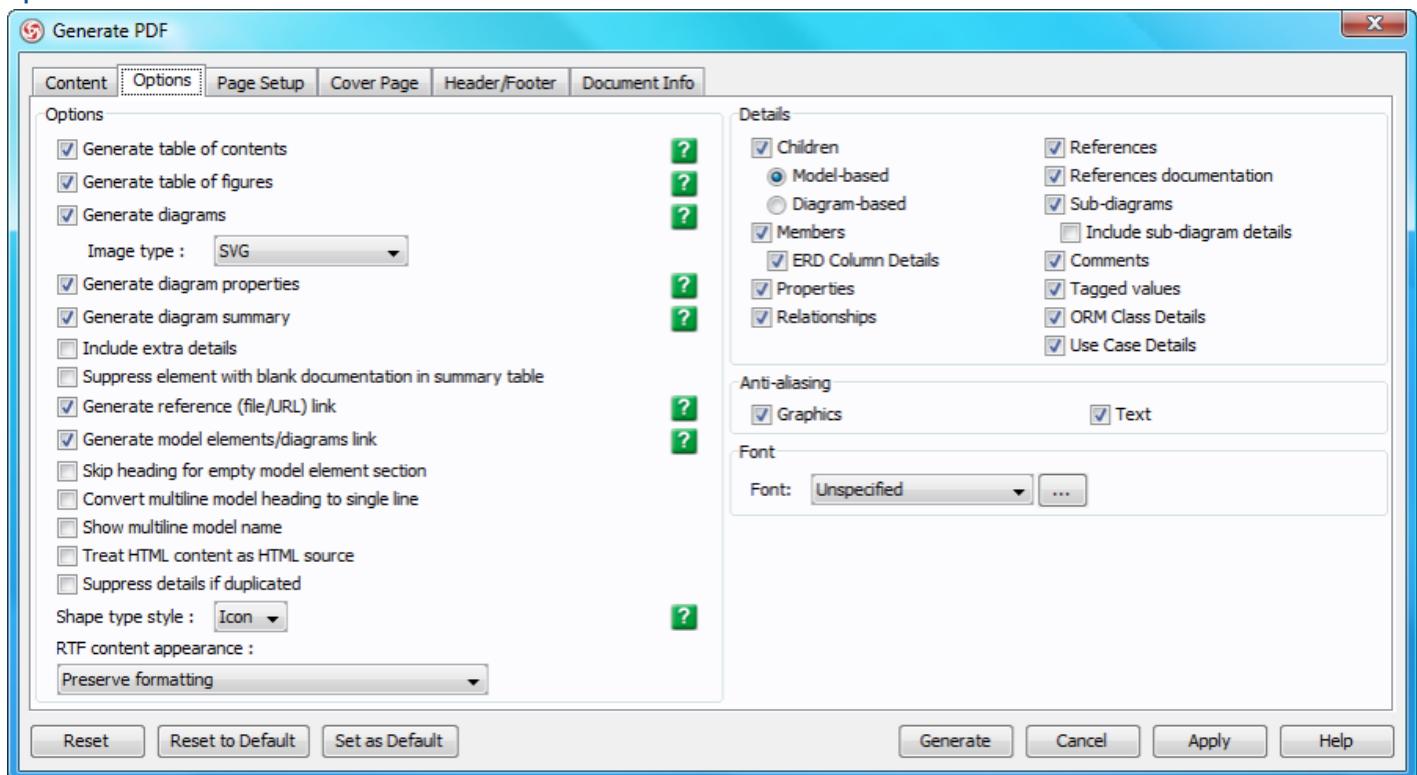
Way of sorting	Description
Automatic	The way of sorting elements is automatically managed. The order is often based on the flow and/or position of elements, which is the most logical order, following most users' understanding of that kind of diagram.
Sort by Tree	Sort diagram elements by following the 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

Configuring report

Report generation can be configured to make the output more close 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

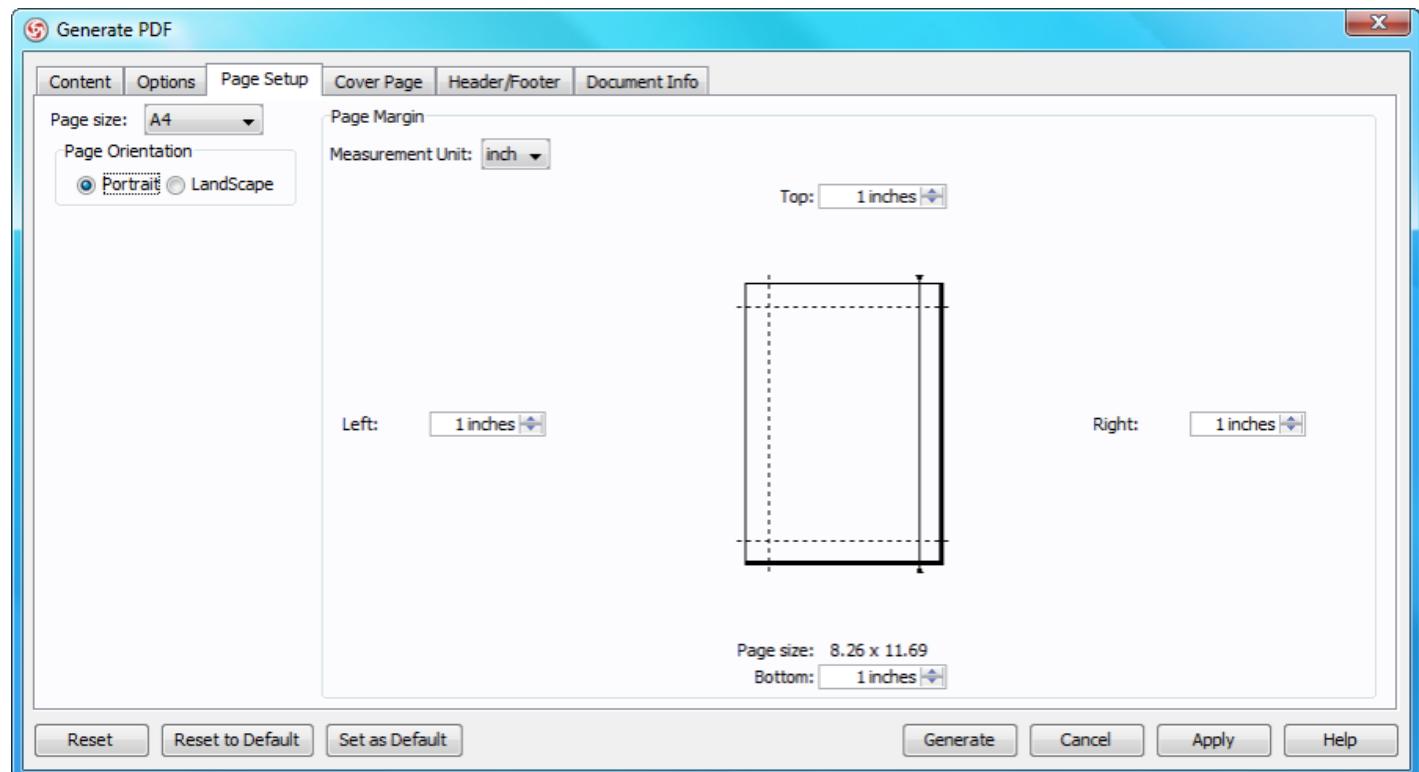
Option	Description
Generate table of contents	If this option is selected, table of content for this document will be generated to the report
Generate table of figures	If this option is selected, table of figures for this document will be generated to the report
Generate diagrams	If this option is selected, the image of the selected diagrams will be generated to the report. For PDF report, you can select the type of diagram. Here are the possible selections: <ul style="list-style-type: none">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 report content.
Generate diagram properties	If this option is selected, the properties of the selected diagrams will be generated to the report.
Generate diagram summary	If the option is selected, the summary of the selected diagrams will be generated to the report.
Include extra details	If the option is selected, information like ID and stereotypes will be generated to the summary table of report.
Suppress element with blank documentation in summary table	If the option is selected, diagram elements without documentation 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 treat as HTML source.
Suppress details if duplicated	If this option is selected, duplicated details will be suppressed.
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 report - using same font size for RTF content in whole report Display in plain text - using plain text for RTF content
Copy reference files	If this option is selected, referenced files will be copy to output folder of report. With this option, you can copy the whole report folder to another machine and read there, without having broken file linkage for references.
Details	Select a kinds 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, documentation, abstract, leaf are example of properties. Relationships - Association, dependency are example of relationships References - File, diagram, folder, URL, shape are possible kinds of reference References documentation - Determine whether to generate the referenced shape/diagram's documentation 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 report 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	Determines the font family of report content. This option is only available to PDF report.
Encoding	Determines the encoding of HTML file to be generated. This option is only available to HTML report.

A description of general options

Page Setup

Page setup controls the layout of report. You can adjust a report size, page orientation and margin.



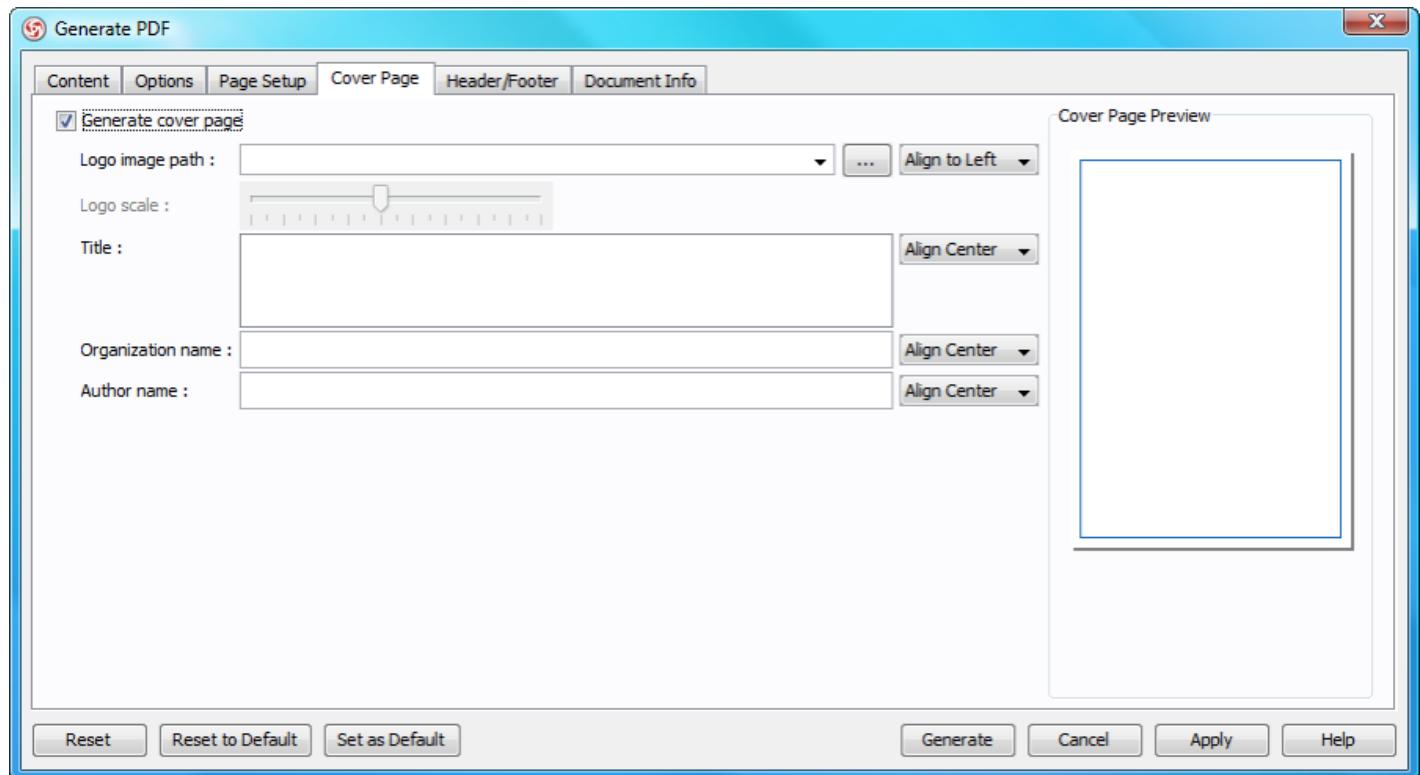
Page setup

Option	Description
Page size	To select the paper size of the generated report.
Page orientation	This option is used to select the orientation of the report (portrait/landscape). This option is only available to PDF and MS Word report.
Page margin	To specify the page margins of the report. This option is only available to PDF and MS Word report.

A description of options of page setup

Cover Page (Front Page for HTML report generation)

Cover page is the first page of report. You can add your company logo there, and enter the report title, organization name and author name. Notice that in HTML report generation, the tab **Cover Page** is named as **Front Page**.



Cover page

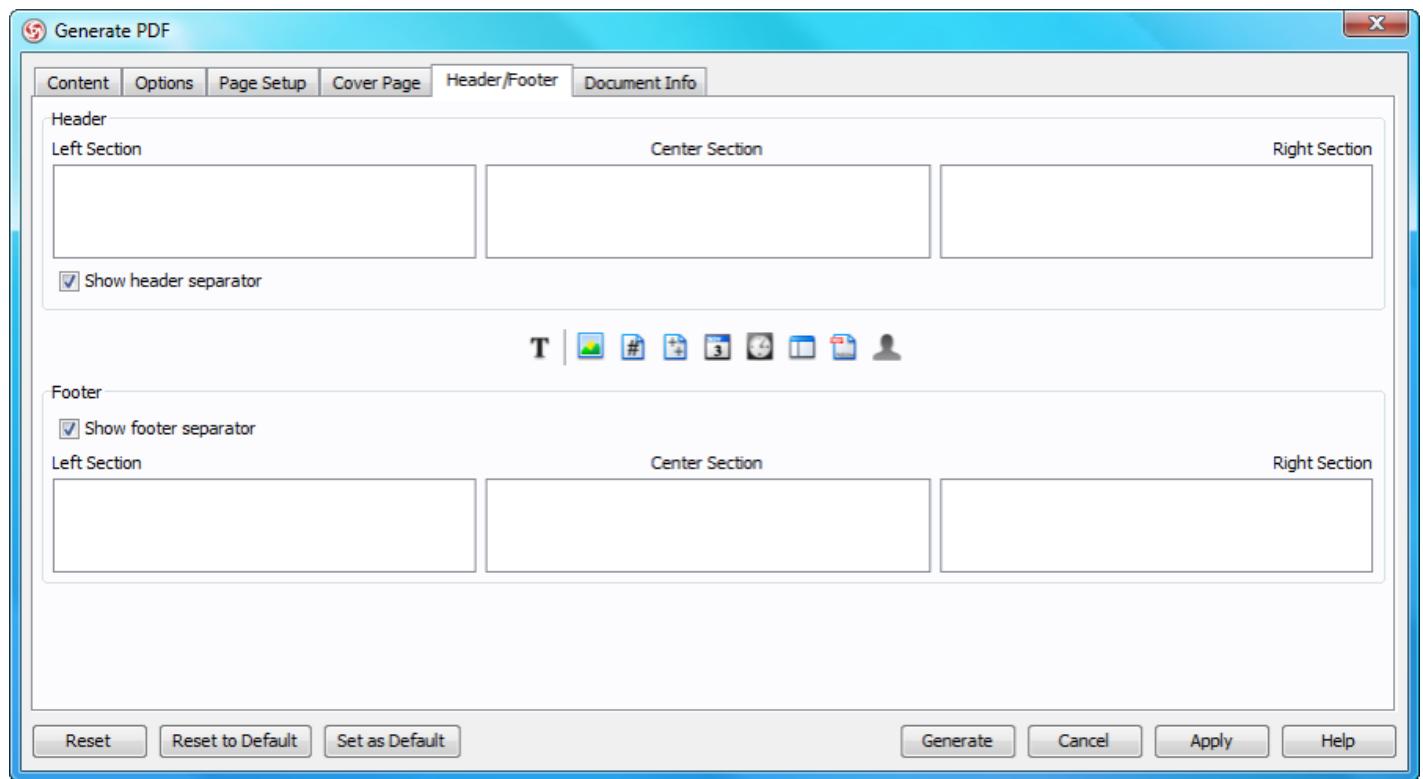
Option	Description
Generate cover page (PDF and MS Word)	If this option is selected, there will be a cover page generated to the report.
Generate front page (HTML)	
Logo image path	An image that appears at the report. You are expected to supply the file path of the image file. The drop down menu at the 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 report.
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 appear in the top and bottom of every page in report. For MS Word report, there are two text boxes for you to edit the header and footer. For PDF report, there are six boxes, three for each of header and footer. Each of the text box represent 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.



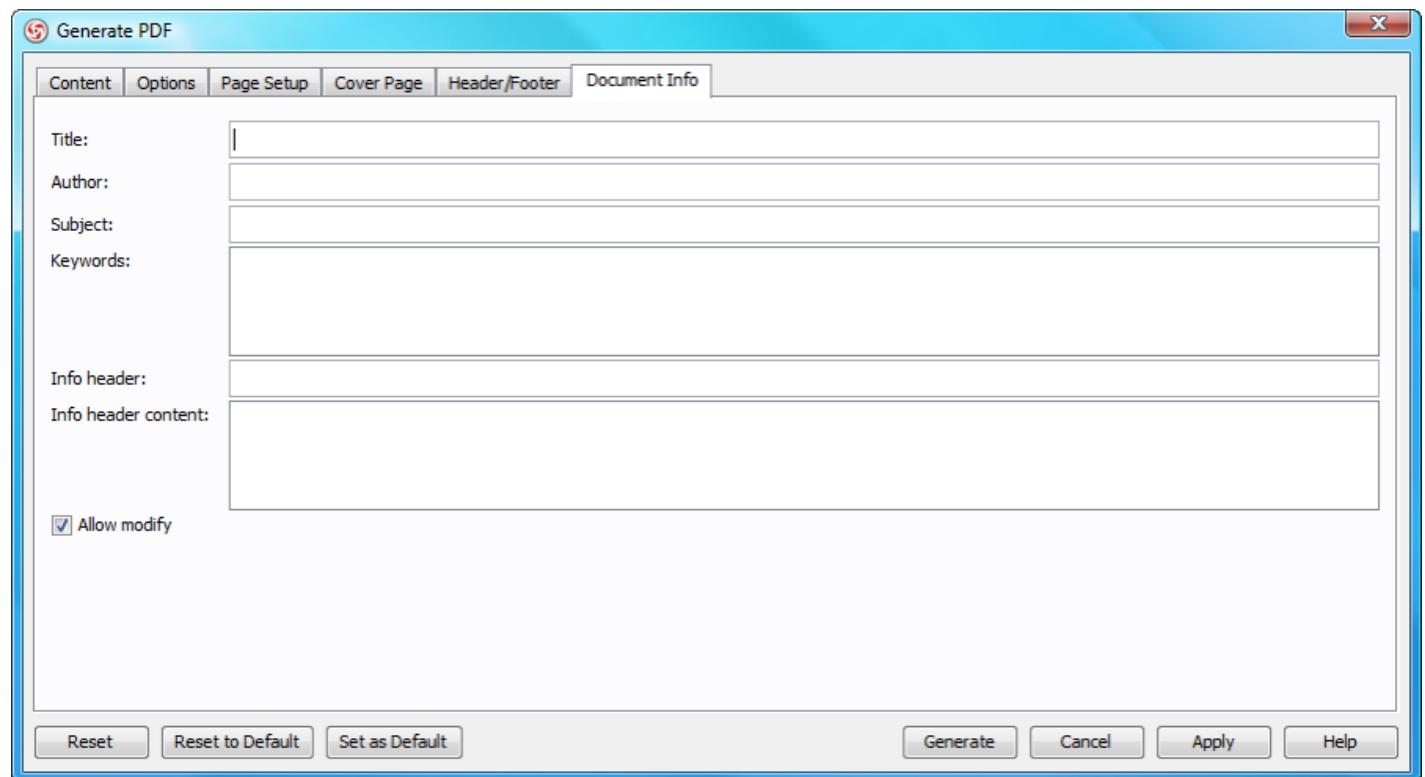
Header/Footer

Variable	Name	Description
	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 report.
	Align to center	Align content to the center of header/footer. This option is available only to MS Word report.
	Align to right	Align content to the right of header/footer. This option is available only to MS Word report.
	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 report is generated to the position where the text cursor is placing.
	Insert time	Insert the time of when the report 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 report file name	Insert the name of report 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 report, document info refers to the meta information of HTML document. For PDF and MS Word report, document info refers to the possible document properties that can be defined.



Document info

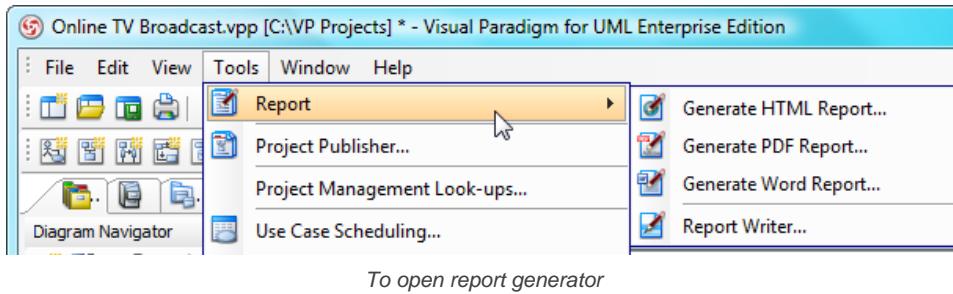
Option	Description
Title	The title of report.
Author	The author of the report. This option is only available to PDF report.
Subject	The subject of the report.
Keywords	The keywords of the report.
Info header	The info header of the report. This option is only available to PDF report.
Info header content	The info header content of the report. This option is only available to PDF report.
Allow modify	Select to allow modification on the report. This option is only available to PDF report.

A description of document info

Customizing report

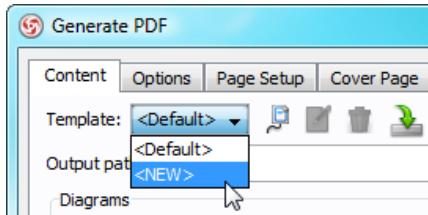
Instead of generating a report in a way that VP-UML defined for you, you can develop report templates to customize the report content, to make output match your needs. With report customization, you can select model elements and properties to generate to report. You also can add custom text content. To customize report:

1. Select **Tools > Report** from the main menu. Then select **Generate HTML Report...**, **Generate PDF Report...** or **Generate Word Report...** depending on the type of report you want to generate.



NOTE: Report templates are shared among HTML, PDF and Word reports. If you just want to design template, but have no preference on the type of report to generate yet, just select either HTML, PDF or Word.

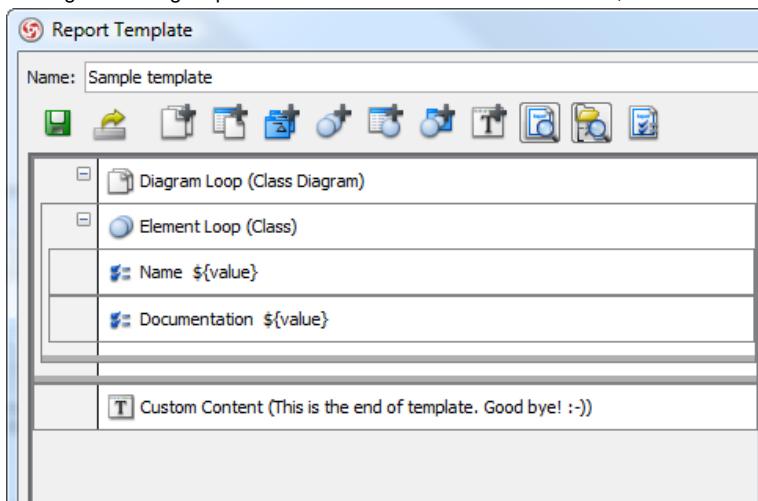
2. Press on the **Template** drop down menu and select **<New>** from the popup menu to open the report template editor and start editing template.



To create a template

3. In the **Report Template** dialog box, enter the template name at the top of dialog box.

4. Construct the report template. A report 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
	Export	To export the opening template to a report template file (.vpr). You can import the file to other machines for reusing it.

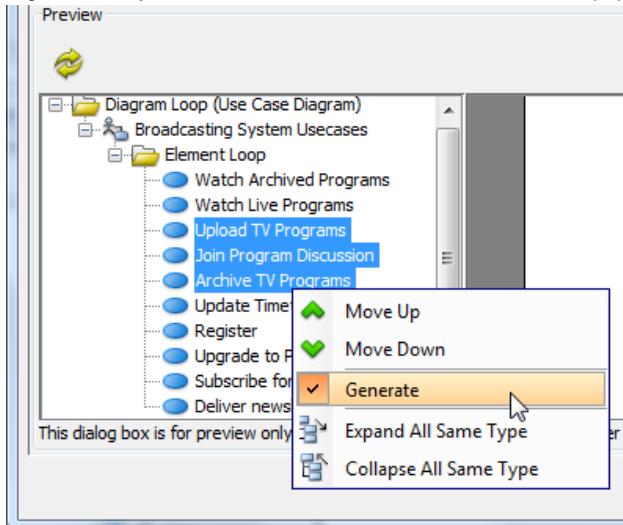
	Add diagram loop	To add a component to template editor, indicating the need of looping specific type(s) of diagram.
	Add diagram summary	To add a component to template editor, indicating the need of looping specific type(s) of diagram for constructing a tabular diagram summary.
	Add element loop (Model)	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 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 report content. Rendering of report 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 report structure.
	Options	Configure report options.

Description of different tools in template editor

Ignoring specific diagrams/shapes

A report 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 report by pressing or from the toolbar.
3. In the report structure tree, select the diagrams or shapes that you want to ignore in generated report.
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 report 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

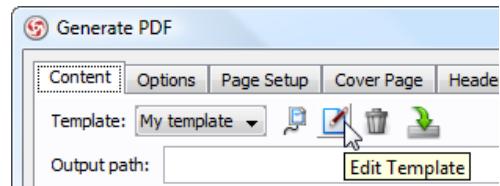
In the report structure tree, right click on the diagram(s) or shape(s) that you want to ignore when generating report and de-select **Generate** from the popup menu.

Reordering diagrams/shapes

In the report 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

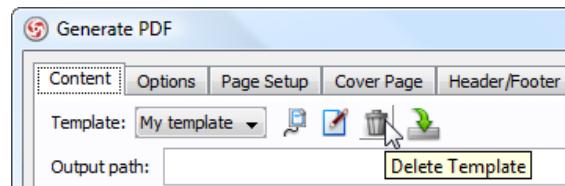
To edit an existing report template, select it in the generate dialog box such as **Generate PDF**, then click on the edit button.



To edit a template

To delete a template

To delete an existing report template, select it in the generate dialog box such as **Generate PDF**, then click on the delete button.

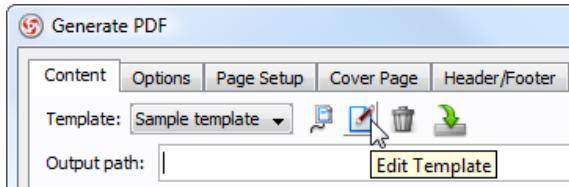


To delete a template

Export/import report template

You can apply a report template on another project by exporting it as a report template file, and importing it to the target project. To export and import template:

1. Open the report 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 report dialog box by selecting **Tools > Generate HTML/PDF/Word Report** from the main menu.
5. Click on the button.
6. In the **Import** dialog box, select the report template file and click **Open**.

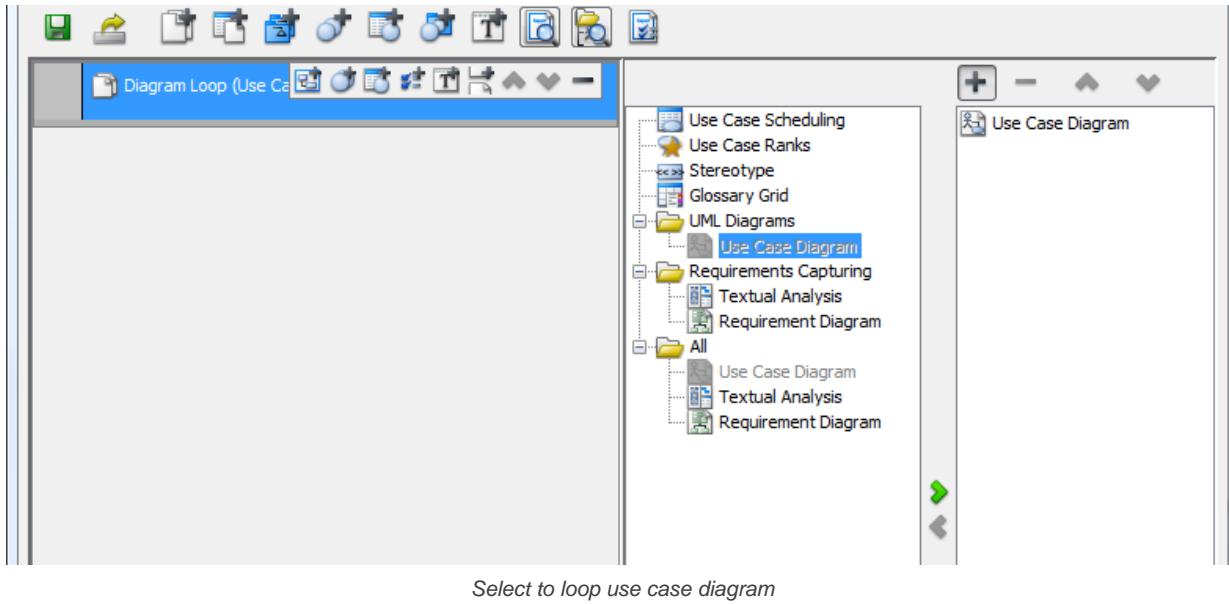
Diagram loop

A diagram loop is a component in a report 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 to be determined by the children components of loop.

Looping diagrams in project

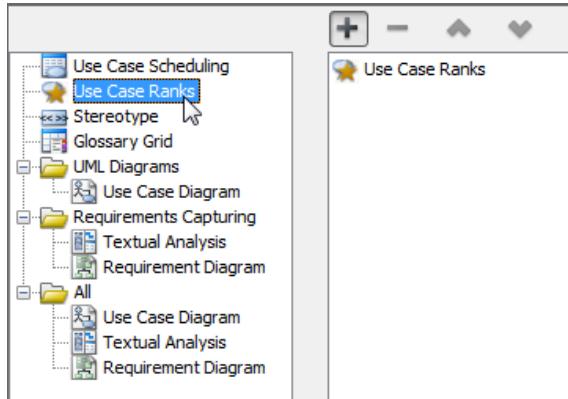
To loop specific type(s) of diagrams in project, create diagram loop at template root. To create diagram loop at template root:

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 loop and click  to confirm the selection.



Including use case scheduling,ranks, stereotypes and various grid in report

Use case scheduling, use case ranks, stereotypes and various grid fall into the category of diagram. If you want to print them to report, follow the steps as described in the previous section, and to select the appropriate content to include at the final step.

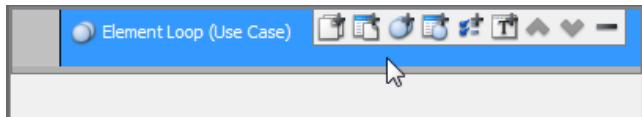


Choosing to include information of use case ranks in template

Looping sub-diagrams of specific element

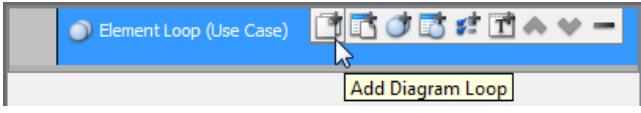
Instead of looping diagrams in a project, you can also place diagram loop under an element loop to loop for sub-diagrams of specific type(s) of model element. To create diagram loop under an element loop:

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

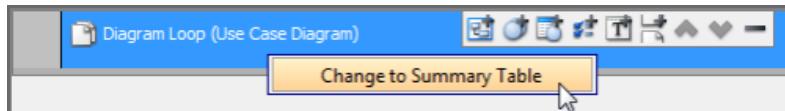
NOTE: Make sure you are clicking on the button from the toolbar of element loop. If you are clicking on the button at editor toolbar, you will create a loop at template root.

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

A diagram summary is a component in a report template, indicating the need of looping (i.e. walking through) specific type(s) of diagram, and presenting their properties in tabular form.

You can convert a diagram loop to diagram summary by right clicking on a diagram loop and selecting **Change to Summary Table** from popup menu.



To convert a diagram loop to diagram summary

NOTE: Conversion can be done only when diagram loop has no children or has solely property values as children.

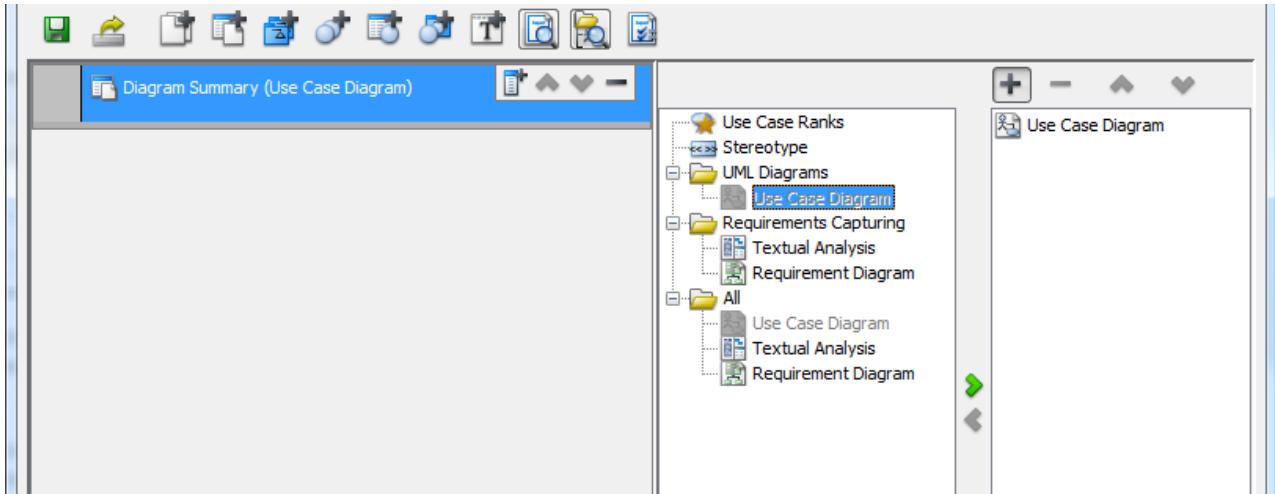
Diagram summary

A diagram summary is a component in a report 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 documentation. 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

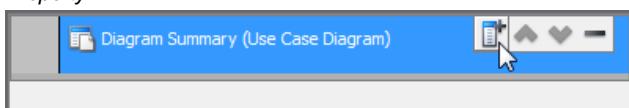
To loop specific type(s) of diagrams in project for listing their properties in summary a table, create diagram summary at template root. To create diagram summary at template root:

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 loop and click  to confirm the selection.



Select to loop use case diagram

3. If you generate report with a template like this, you will obtain a table of diagram names, provided that there exists diagram(s) of the chosen type(s). 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 add into the table as column(s). For more details about the use of property, read the chapter [Property](#).

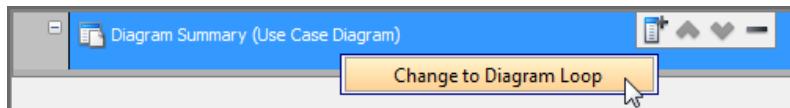


To add a property column in diagram summary

Switching from diagram summary to diagram loop

A diagram loop is a component in a report template, indicating the need of looping specific type(s) of diagram.

You can convert a diagram summary to diagram loop by right clicking on a diagram summary and selecting **Change to Diagram Loop** from popup menu.



To convert a diagram summary to diagram loop

Element loop

An element loop is a component in a report 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:

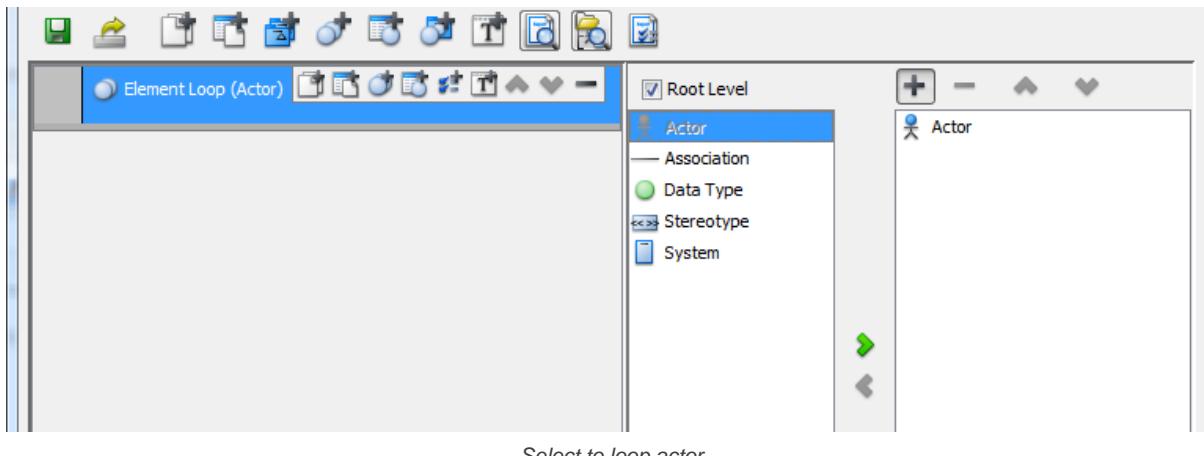
1. In the template editor, click on any of the buttons in the toolbar depending on your need.

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 loop on the right hand side of the template editor and click to confirm the selection.

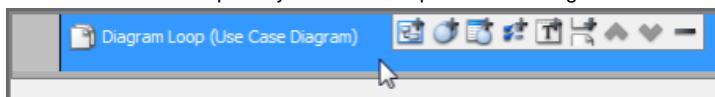


NOTE: You can switch between a root level and a all-level loop by changing the option **Root Level** above the element list. Notice that changing the value of **Root Level** is not about changing available element selection, but also the end result, whether to access only root level elements or not.

Looping diagram elements in a diagram

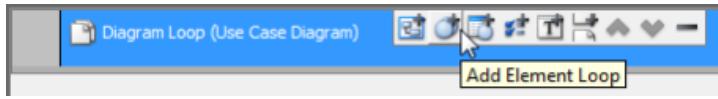
Instead of looping model elements in a project, you can also place element loop under a diagram loop to loop for diagram elements in specific type(s) of diagram. To create element loop under a diagram loop:

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

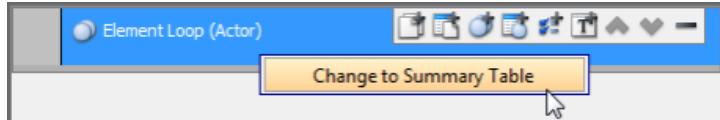
NOTE: Make sure you are clicking on the button from the toolbar of diagram loop. If you are clicking on the button at editor toolbar, you will create a loop at template root.

3. On the right hand side of the template editor, select the type of diagram element to loop and click to confirm the selection.

Switching from element loop to element summary

An element summary is a component in a report template, indicating the need of looping (i.e. walking through) specific type(s) of diagram/model element, and presenting their properties in tabular form.

You can convert an element loop to element summary by right clicking on a element loop and selecting **Change to Summary Table** from popup menu.



To convert an element loop to element summary

NOTE: Conversion can be done only when element loop has no children or has solely property values as children.

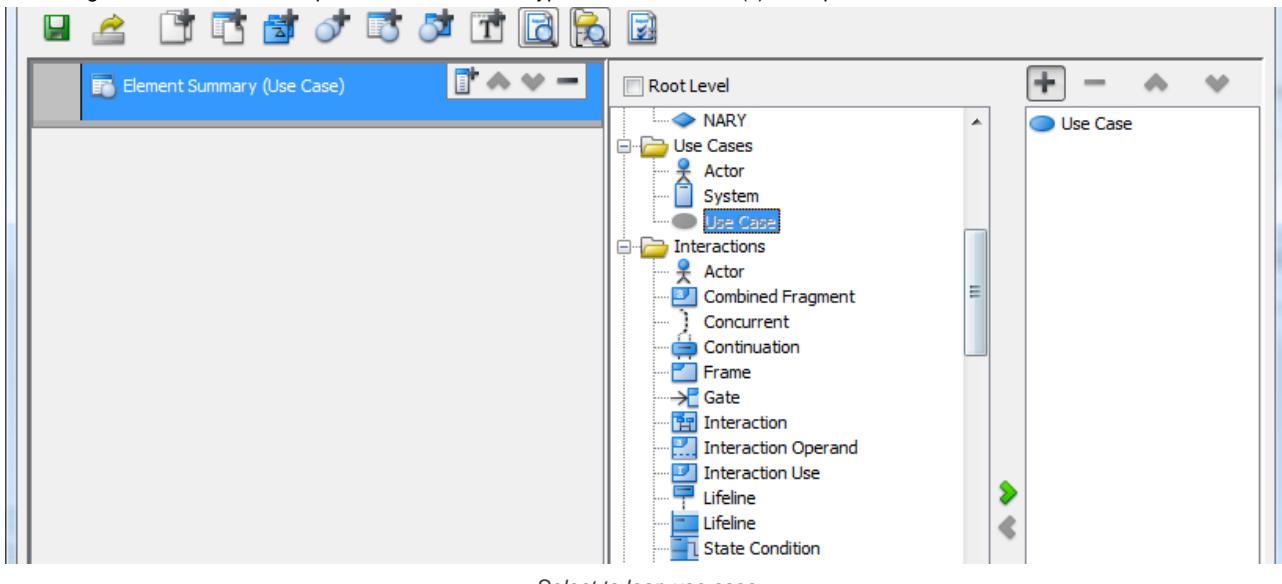
Element summary

An element summary is a component in a report 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 summary a table, create element summary at template root. To create element summary at template root:

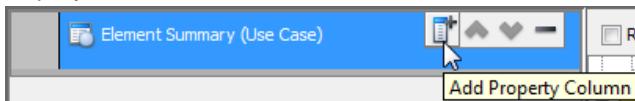
1. In the template editor, click on the  on toolbar.
2. On the right hand side of the template editor, select the type of model element(s) to loop and click  to confirm the selection.



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 report with a template like this, you will obtain a table of element names, provided that there exists element(s) of the chosen type(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 add into the table as column(s). For more details about the use of property, read the chapter **Property**.

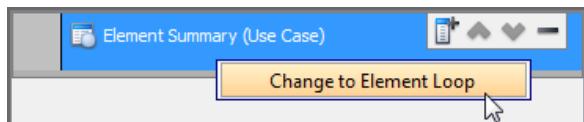


To add a property column in element summary

Switching from element summary to element loop

An element loop is a component in a report template, indicating the need of looping specific type(s) of model/diagram element.

You can convert an element summary to element loop by right clicking on an element summary and selecting **Change to Element Loop** from popup menu.



To convert an element summary to element loop

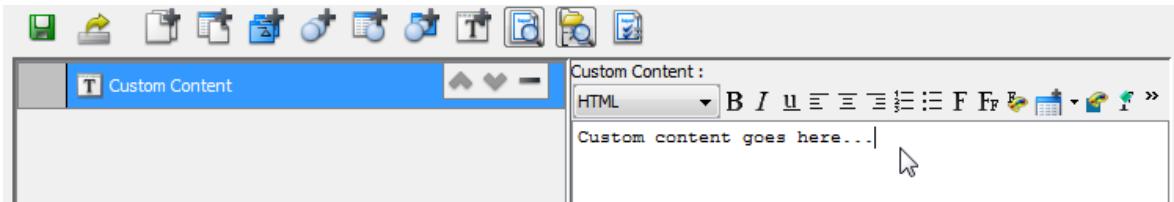
Custom content

Custom content is a component in a report template, which acts as a placeholder of user written text. For example, to add a section of acknowledgment at the beginning of report 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.



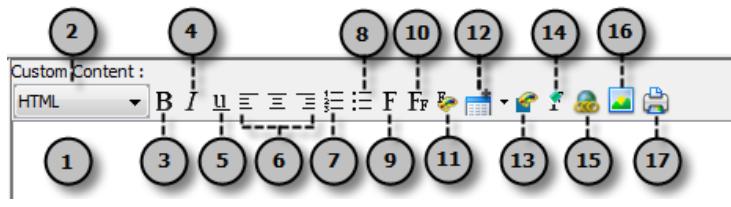
Adding custom content

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, 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	HTML	HTML - Read and edit the real content. HTML Source - Read and edit the HTML source of content.
3	Bold	Set the highlighted text to bold.
4	Italic	Set the highlighted text to italic.
5	Underline	Underline the highlighted text

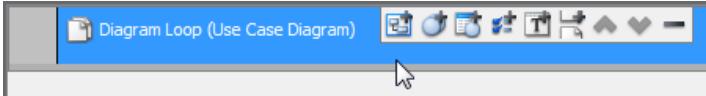
6 Alignments	Set the alignment of highlighted text to right, center or left.
7 Ordered list	Add a numbered list.
8 Un-ordered list	Add a list with bullet points.
9 Font	Select the font family of highlighted text.
10 Font size	Select the size of highlighted text.
11 Font color	Select the color of highlighted text.
12 Table	Add a table.
13 Background color	Select the background color of highlighted text.
14 Clear formats	Clear formats of whole editor to convert the content to plain text.
15 Link	Add a hyperlink.
16 Image	Add an image.
17 Print	Print the custom content.

A description of custom content editor

Diagram image

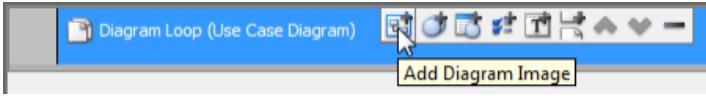
Diagram image is a component in a report template, which represents a placeholder of the image of a diagram under a diagram loop. You must place a diagram image under diagram loop. To add a diagram image:

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

Property value

Property value is an element component in a report 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

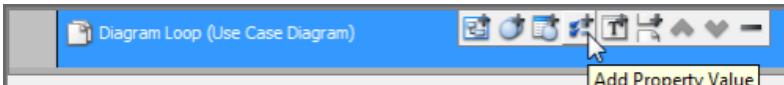
To add property value into a loop or summary:

1. Select the loop or summary that you want to add property value under it.



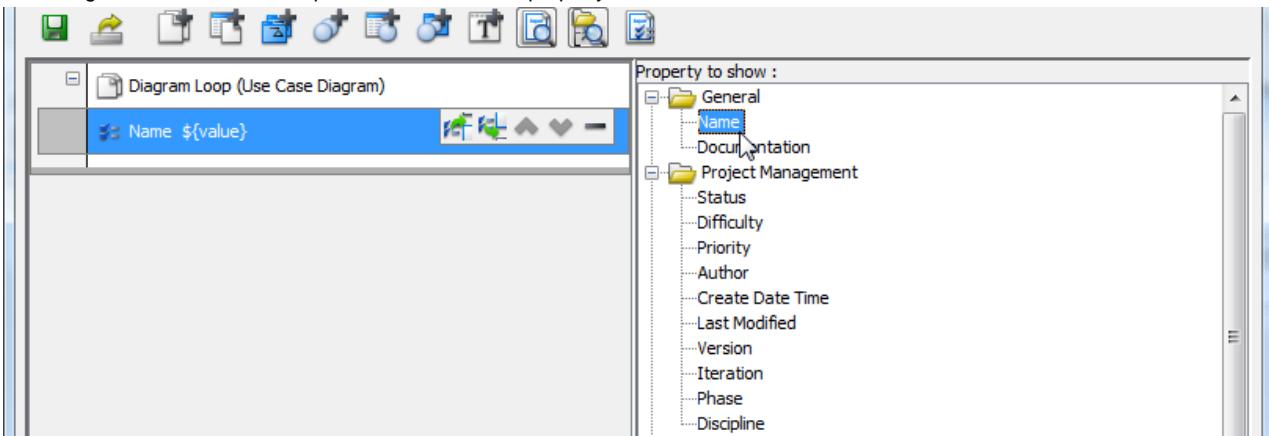
Selecting a diagram loop

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.

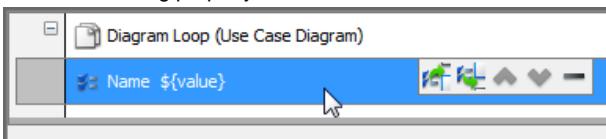


Selecting a property to access

Adding a property below another property

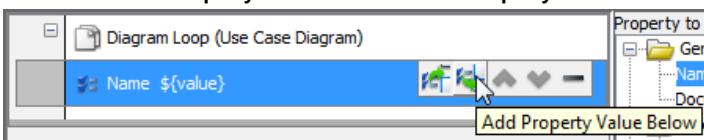
To add property below another property:

1. Select an existing property value.



Selecting a property value

2. Click on the **Add Property Value Below** or **Add Property Value Above** button.



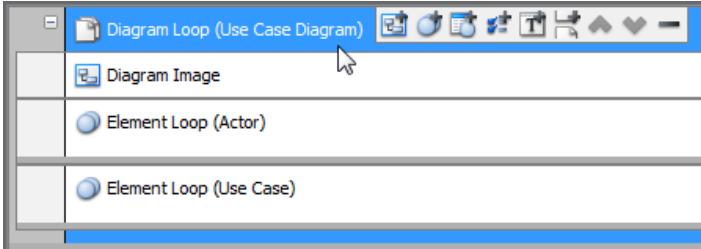
To add a property value below an existing one

3. On the right hand side of the template editor, select the property to access.

Page break

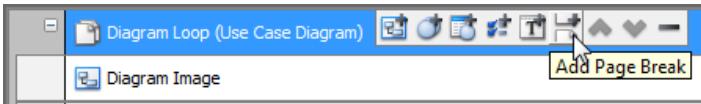
Page Break is where the report should start the contents which follows in a new page. It can be placed within a loop, either a diagram or element loop. To create a Page Break:

1. Select the loop that you want to insert a page break.



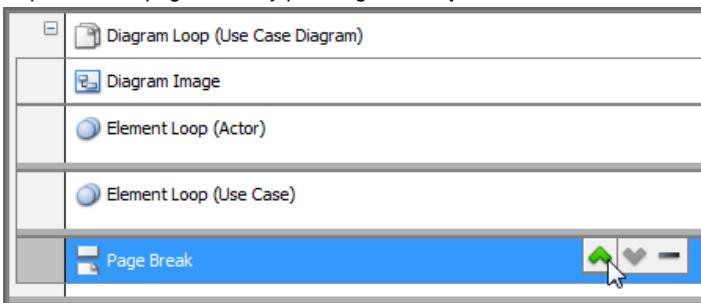
Selecting the loop for adding a page break

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.



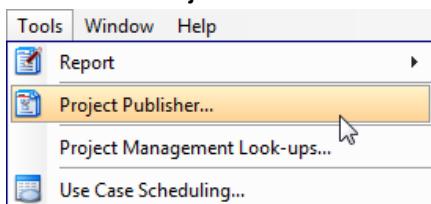
Repositioning a page break

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:

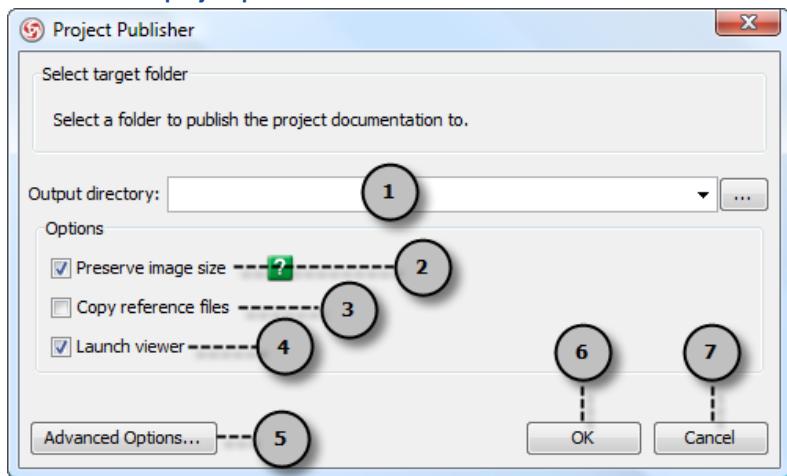
1. Select **Tools > Project Publisher...** from main menu.



To launch project publisher

2. In the **Project Publisher** dialog box, specify the output directory where you want to save the published content.
3. You can optionally configure the publisher by adjust the options or options. For details, refer to the sections below.
4. Click **OK** button to start publishing.

An overview of project publisher



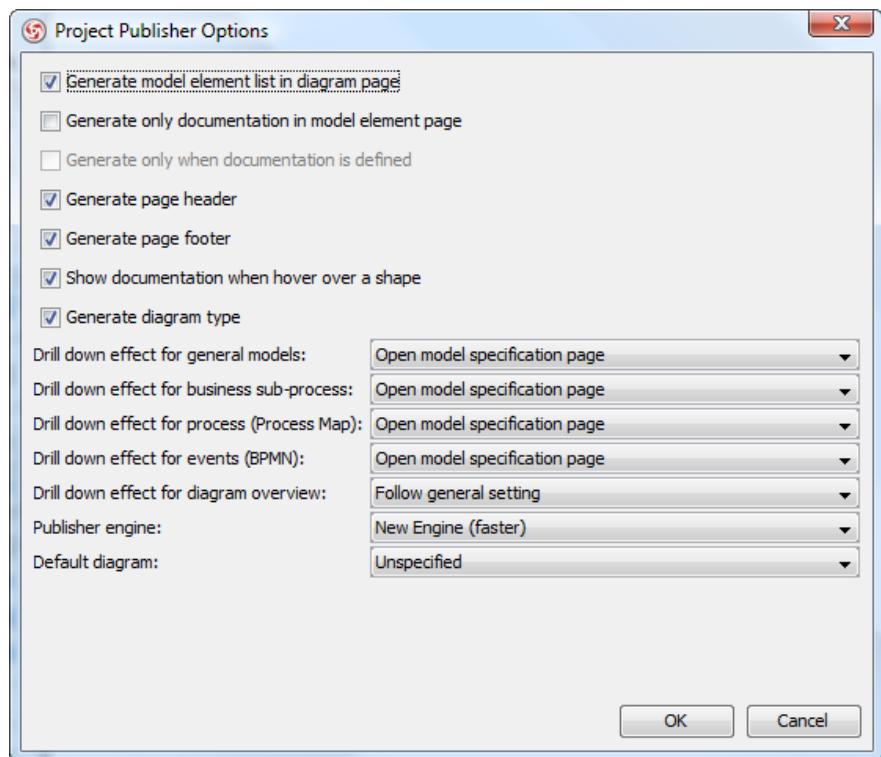
Overview of project publisher

No.	Name	Description
1	Output directory	The folder where you want to publish the content to.
2	Preserve image size	When checked, published content will show images in exact width and height.
3	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 any referenced file when browsing the published content in other machine easily.
4	Launch viewer	When checked, the system will launch the web browser and open the published Web contents.
5	Advanced options	Click to configure advanced publisher options. For details about the options, read the next section.
6	OK	Click to publish.
7	Cancel	Click to close the dialog box without publishing.

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.



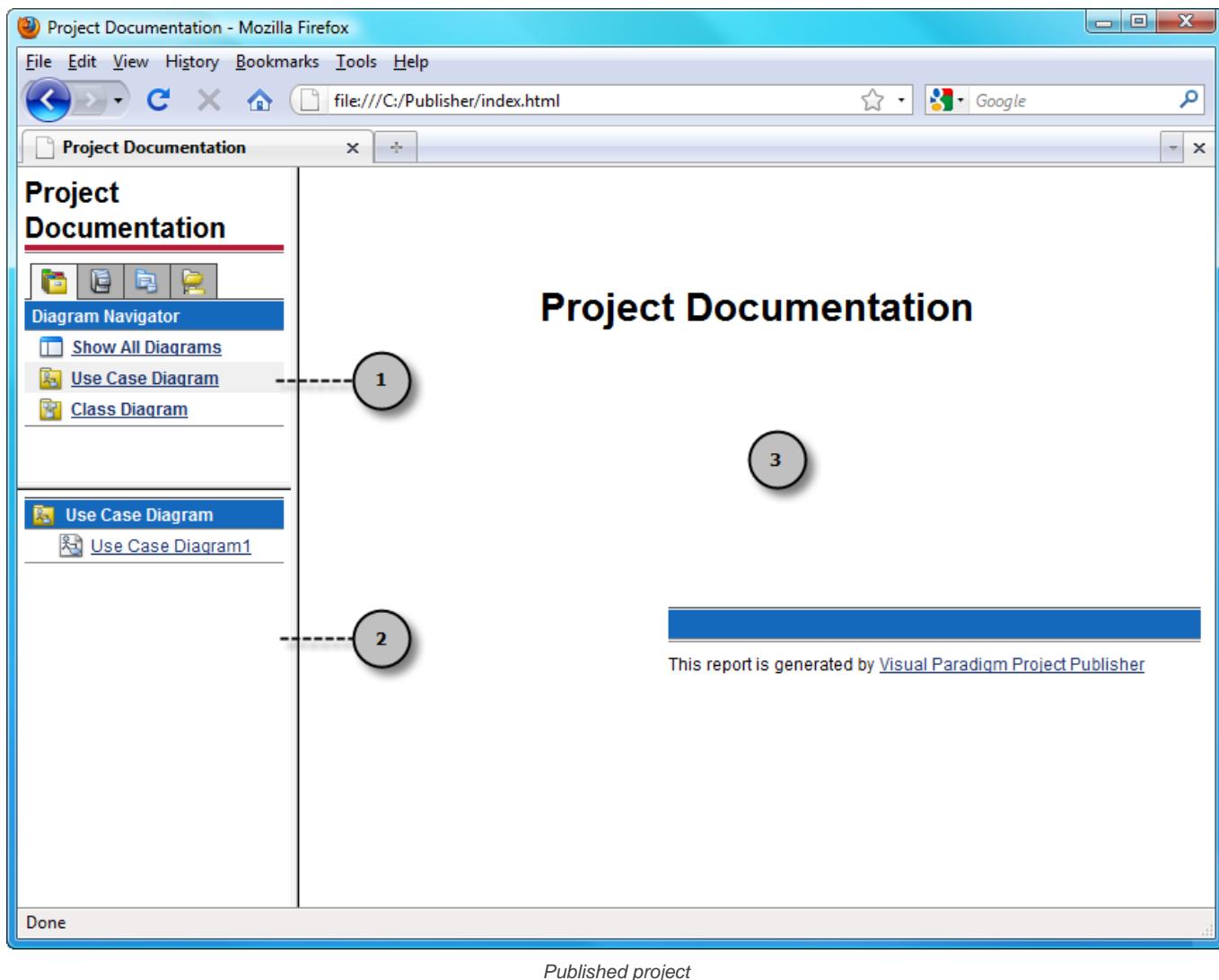
Project publisher advanced options

Option	Description
Generate model element list in diagram page	Check to generate a list of model element in a diagram page.
Generate only documentation in model element page	Check to generate only documentation in model element page, and exclude other contents.
Generate only when documentation is defined	Generate element page only when that element has documentation. 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 pre-defined header.
Generate page footer	Check to generate pre-defined footer.
Show documentation when hover over a shape	Check to show element's documentation when moving the mouse pointer over a shape in an image of diagram.
Generate diagram type	Check to generate the diagram type in addition to diagram name.
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.
Publisher engine	Choose the engine for publishing. You are advised to use the new engine.
Default diagram	Choose the diagram that first appear when opening the published content. If unspecified, a default page with project information will be presented.

Description of project publisher advanced options

Using the published content

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.



No.	Name	Description
1	Navigator pane	
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. For more details about the possible contents please refer to the Navigator Pane section.
3	Content pane	It shows the details of the item (diagram, model or package/class) you clicked in the Menu Pane or Content Pane.

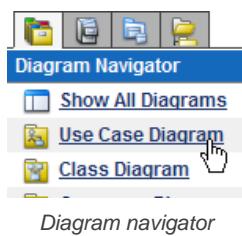
Description of the interface of published Web content

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 angle.

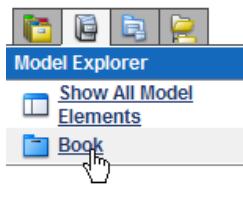
Diagram Navigator

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.



Model Explorer

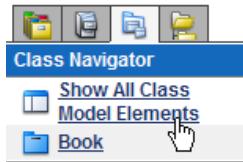
Shows the package models in the project. You can click on a package to view its child models in the **Menu Pane**, or click **Show All Models** to view all model elements.



Model Navigator

Class Navigator

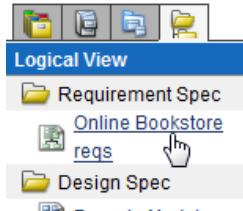
Shows the Package models in the project. You can click on a package to view its child packages/classes in the **Menu Pane**, or click **Show All Models** to view all packages/classes.



Class navigator

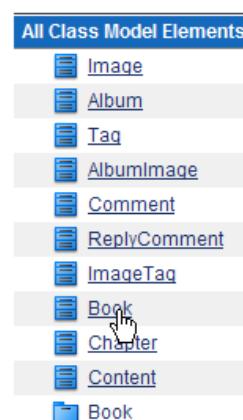
Logical view

Echos the logical view defined in project. You can click on a diagram to open it.



Logical view

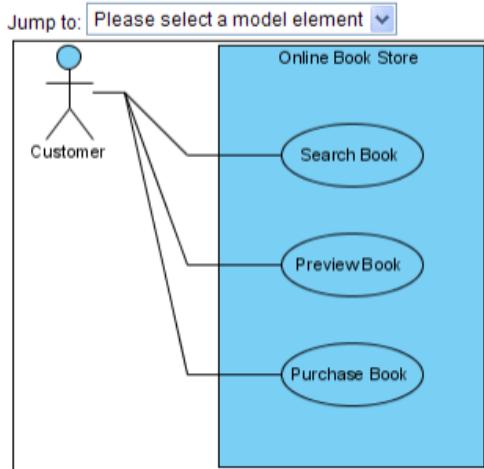
To view the details of an item (diagram, model or package/class), click on its link in the **Menu Pane** and its details will be shown in the **Content Pane**.



Menu navigator

Diagram Content

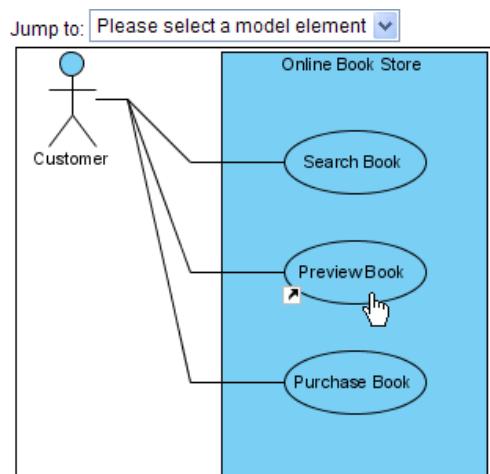
Use Case Diagram - Online Book Store - General



Model Elements

Name	Documentation
<i>Diagram content</i>	

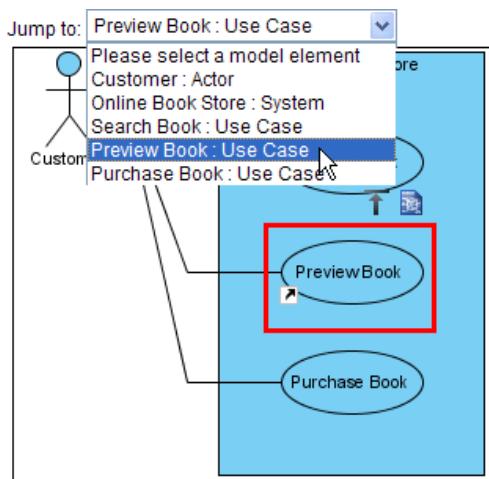
The diagram type, name, description, together with a full size image of the diagram are shown in the Content Pane. The image is mapped to different clickable regions for each shape, so you can click on a shape in the image to view its details.



Shape link to descriptions

Using jump to

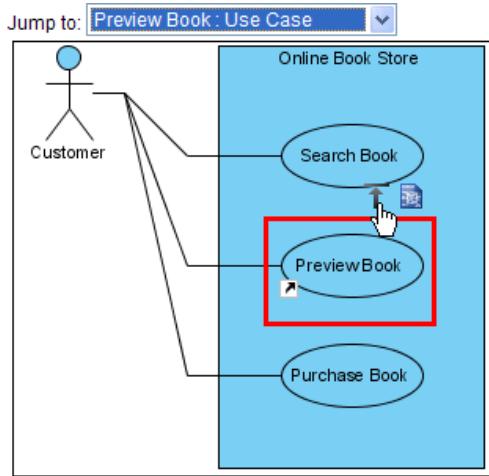
The **Jump to** drop down menu in the diagram content page lists all shapes in the diagram. You can select a shape to jump to. The content page will scroll to the selected shape and the shape will be highlighted by a red border.



Jump to

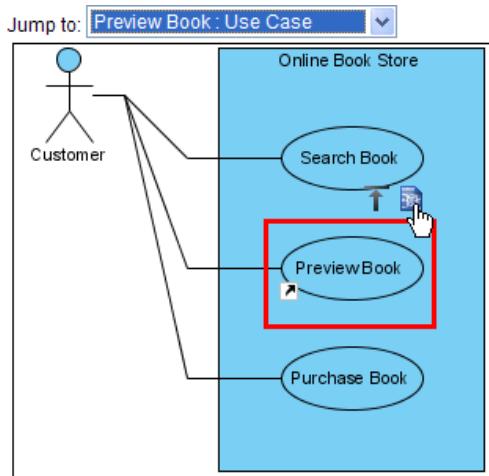
Besides, there will be two shortcut buttons above the selected shape.

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 documentation 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	Documentation
 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 Documentation

[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 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 Documentation

[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																																		
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>Role</td><td></td></tr><tr><td>From</td><td><table border="1"><thead><tr><th>Element</th><th>Customer : Actor</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table></td></tr><tr><td></td><td><table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>Role</td><td></td></tr><tr><td>To</td><td><table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table></td></tr><tr><td>Abstract</td><td>false</td></tr><tr><td>Leaf</td><td>false</td></tr><tr><td>Visibility</td><td>Unspecified</td></tr><tr><td>Derived</td><td>false</td></tr></tbody></table></td></tr></tbody></table>	Name	Value	Role		From	<table border="1"><thead><tr><th>Element</th><th>Customer : Actor</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table>	Element	Customer : Actor	Multiplicity	Unspecified	Navigable	true		<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>Role</td><td></td></tr><tr><td>To</td><td><table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table></td></tr><tr><td>Abstract</td><td>false</td></tr><tr><td>Leaf</td><td>false</td></tr><tr><td>Visibility</td><td>Unspecified</td></tr><tr><td>Derived</td><td>false</td></tr></tbody></table>	Name	Value	Role		To	<table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table>	Element	Preview Book : Use Case	Multiplicity	Unspecified	Navigable	true	Abstract	false	Leaf	false	Visibility	Unspecified	Derived	false
Name	Value																																		
Role																																			
From	<table border="1"><thead><tr><th>Element</th><th>Customer : Actor</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table>	Element	Customer : Actor	Multiplicity	Unspecified	Navigable	true																												
Element	Customer : Actor																																		
Multiplicity	Unspecified																																		
Navigable	true																																		
	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>Role</td><td></td></tr><tr><td>To</td><td><table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table></td></tr><tr><td>Abstract</td><td>false</td></tr><tr><td>Leaf</td><td>false</td></tr><tr><td>Visibility</td><td>Unspecified</td></tr><tr><td>Derived</td><td>false</td></tr></tbody></table>	Name	Value	Role		To	<table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table>	Element	Preview Book : Use Case	Multiplicity	Unspecified	Navigable	true	Abstract	false	Leaf	false	Visibility	Unspecified	Derived	false														
Name	Value																																		
Role																																			
To	<table border="1"><thead><tr><th>Element</th><th>Preview Book : Use Case</th></tr></thead><tbody><tr><td>Multiplicity</td><td>Unspecified</td></tr><tr><td>Navigable</td><td>true</td></tr></tbody></table>	Element	Preview Book : Use Case	Multiplicity	Unspecified	Navigable	true																												
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

Installing report writer

The Report Writer is a sophisticated tool for report creation. Users can output the existing project as reports by documenting their project within VP-UML. VP-UML offers seamless integration of UML modeling tool with word processors to provide a unified documenting environment. By dragging the models from VP-UML to Report Writer, data is extracted from models and content is created in Report Writer.

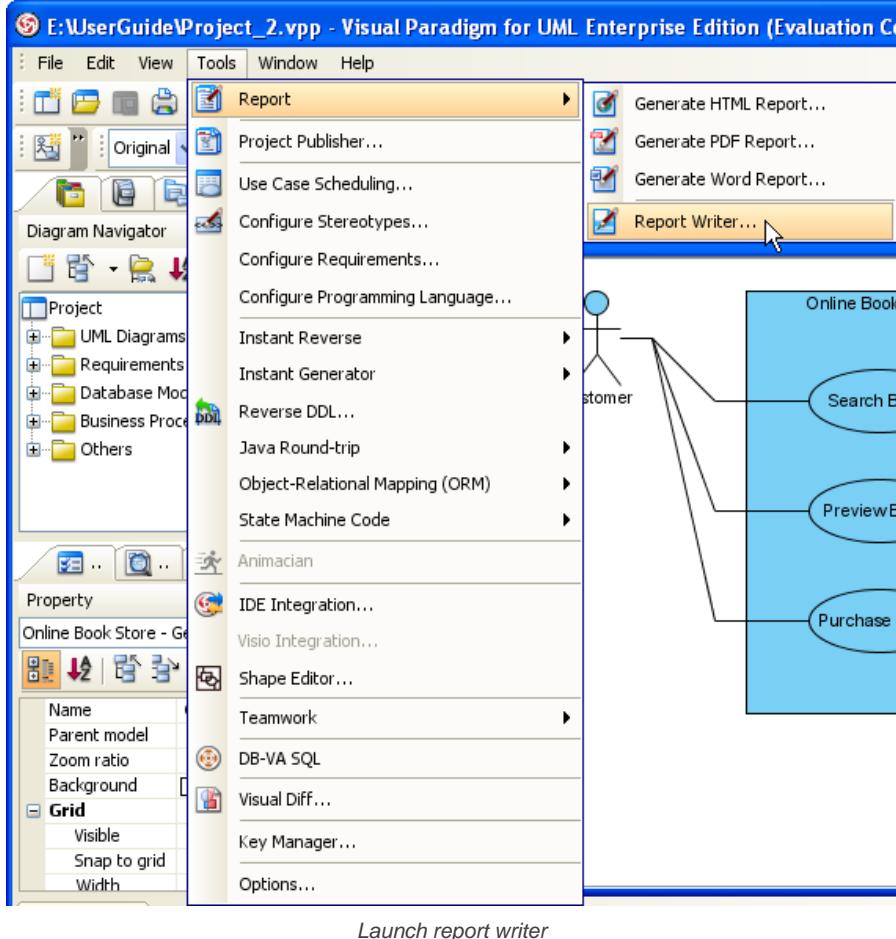
Retaining the conformance between documentation and design is a tedious task. Report Writer maintains the consistency between them. If you create a new model, the content will be appended to the existing one. If you remove a model, the generated element will be removed. If you re-edit the models, the content will be refreshed.

Users can also apply their own style for the generated element, to the Report Writer more flexible.

Launching report writer

To launch Report Writer:

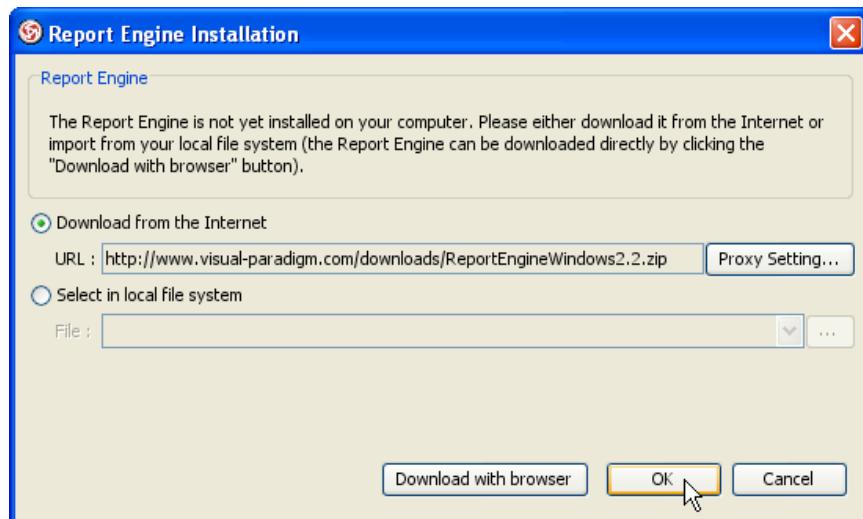
1. Select **Tools > Report > Report Writer...** from main menu.



Launch report writer

Installing report engine

If it is the first time you have started the Report Writer, the **Report Engine Installation** dialog box will be displayed asking for the installation of report engine.



Report engine installation dialog

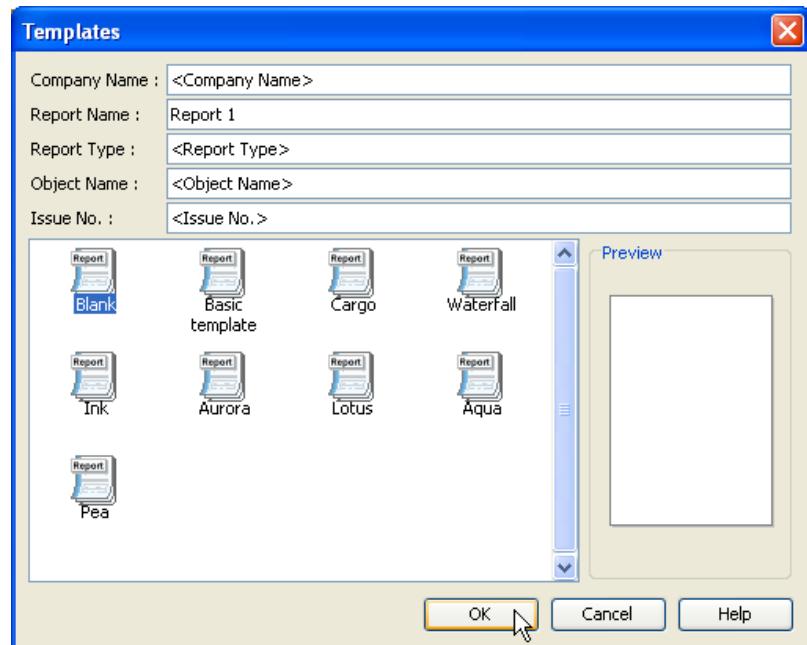
To install Report Engine, perform one of the following actions:

- Select the option **Download from Internet** and click **OK**. This downloads the report engine from the Internet and automatically proceeds with report engine installation once the download has been completed.
- Select the option **Select in local file system**. Locate the report engine and then click **OK** to start the installation. The report engine can be obtained by clicking **Download with browser**.

Report writer will be available when installation is done.

Creating report

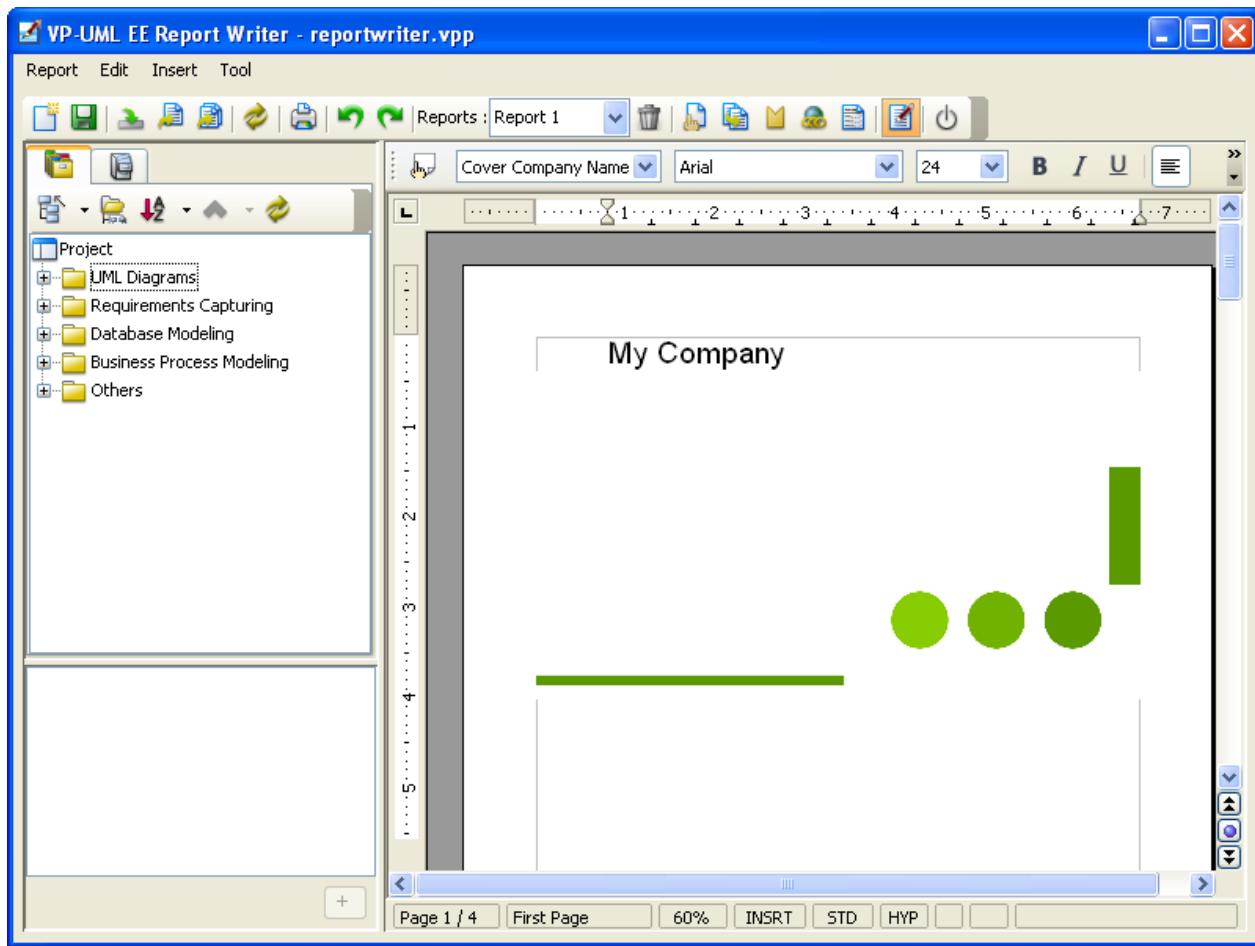
Upon launching Report Writer for the first time, the **Templates** dialog box will be displayed and ask for the information of the new report. Enter the report information and select a desired report theme for the report, preview of the selected theme is shown on the preview pane. Click **OK** to start report writer when everything is ready.



Templates dialog

Introduction to report writer user interface

When Report Writer is launched you are taken to the Report Writer environment where you can create and edit your reports. Three distinct panes are presented on the screen: the **Project Explorer**, **Template Pane** and the **Writer Pane**.



Report writer

Diagram navigator

The **Diagram Navigator** displays all diagrams within the project in a form of a project tree and organizes them by their diagram type. Through the use of a folding tree structure you can browse the names of these diagrams by either expanding or collapsing the folders and perform sorting by diagram type and name.

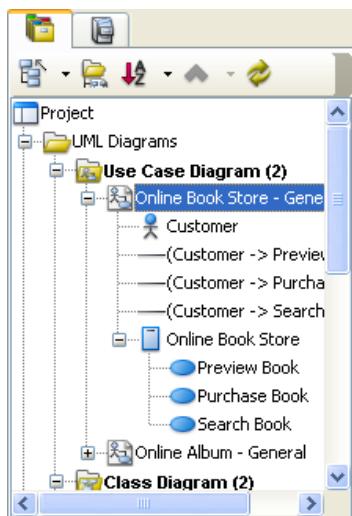


Diagram navigator

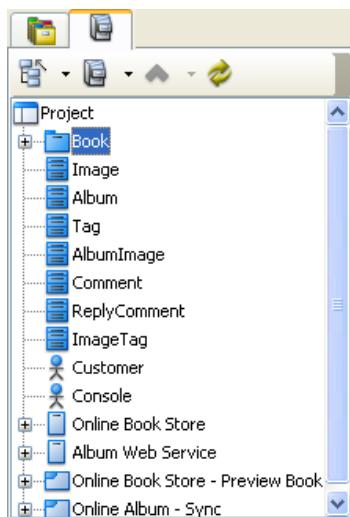
Details of each button in Diagram Navigator are show as below:

Button	Description
	To collapse all the nodes within the project tree.
	To expand all the nodes within the project tree.
	To show only diagrams but not models in the tree.
	To sort diagrams within the project tree by alphabetical order of their names.
	To sort diagrams within the project tree by their diagram type.
	To sort diagrams within the project tree by their diagram type.

Details of diagram navigator

Model tree

The **Model Pane** displays models within the project in a form of a project tree. Notice that not all the model elements are displayed, and only the elements that are available for generating report content are shown.



Model tree

Details of each button in Model Tree are shown as below:

Button	Description
	To collapse all the nodes within the project tree.
	To expand all the nodes within the project tree.
	To display the models within the project without sorting. Ordering of models will be based on their order of creation.
	To sort models within the project tree by alphabetical order of their names.
	To sort models within the project tree by their model type.
	To refresh the project tree within the Model Pane.

Details of model tree

Template pane

The **Template Pane** displays all the templates available for the model or diagram selected in **Project Explorer**.



Template pane

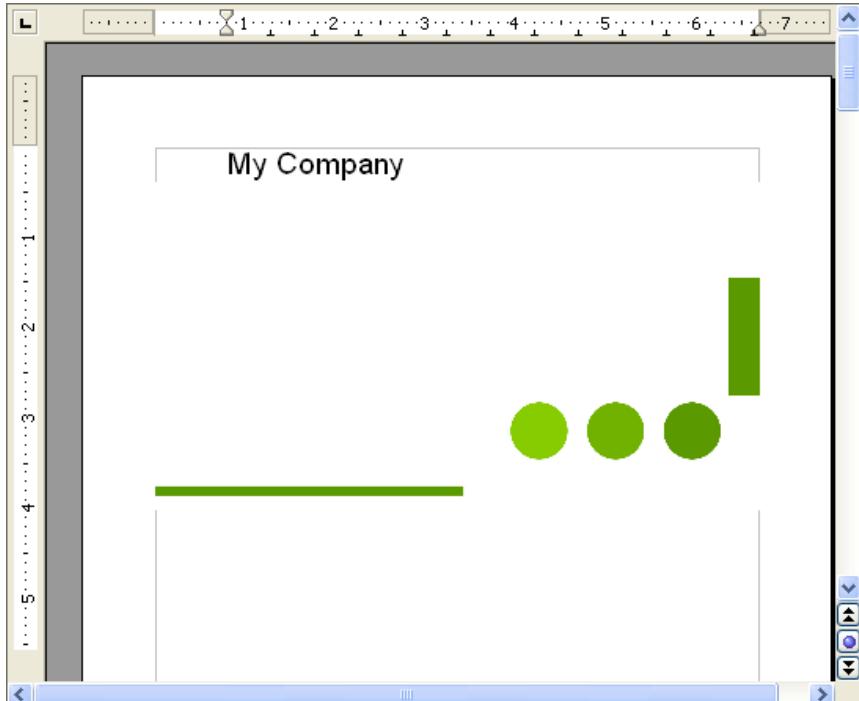
Each template represents the corresponding report content of a particular model or diagram. By dragging a template into the Writer Pane, the report content will be printed on the report. There are three types of template: Text, Image and Table. Each of them has its own appearance in the report content.

Button	Description
ABC	The generated element block is mainly composed of text. It is mainly used in the documentation template of elements.
	The generated element block is mainly composed of images. It is used in the diagram template for UML Diagrams.
	The generated element block is mainly composed of tables. Most of the content-related templates use this type of template.

Details of template pane

Writer pane

Writer Pane embeds a word processor to provide a report editing environment.



Writer pane

Toolbar

Toolbar is the horizontal bars placed below the menu bar. They store all the frequently used commands that appear as a row of buttons.

Button	Description
	To create a new report.
	To save modified reports.
	To import an external document (either an .sxw or a .doc file) as a report.
	To export the current report as an .sxw or .doc file.
	To export all the reports within the current project.
	To update the content within the current report from the VP-UML models.
	To print the current report by supplying the printer name.
	To undo the last action you performed.
	To redo the last action you performed.
Reports : <input type="button" value="Report 1"/>	To select a report from the current project for editing.
	To remove the existing report(s).
	To display the stylist dialog box for modifying the style.
	To copy the style settings defined in another report.
	To display bookmarks that outlines the boundary for each generated element.
	To insert a hyperlink.
	To insert index or table
	To include documentation of model when generating content.
	To close Report Writer and go back to VP-UML.

Details of toolbar

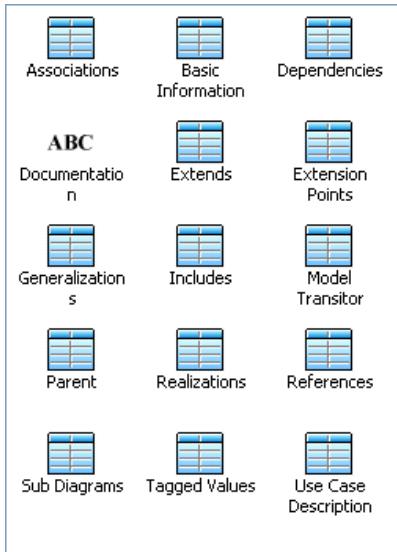
Constructing report

Creating a generated element

The term "Generated element" here means a block of report content generated by Report Writer and consists of details of a particular diagram or model element.

To create a generated element block:

1. Click to select the desired model element from either the **Diagram Navigator** or **Model Navigator** for content generation.
2. The supported templates for the selected model element are shown on the Template Pane. Each template represents a way in presenting the selected model element on the report. For example, **Children** template of a *System* represents a list of children placed inside a particular System.



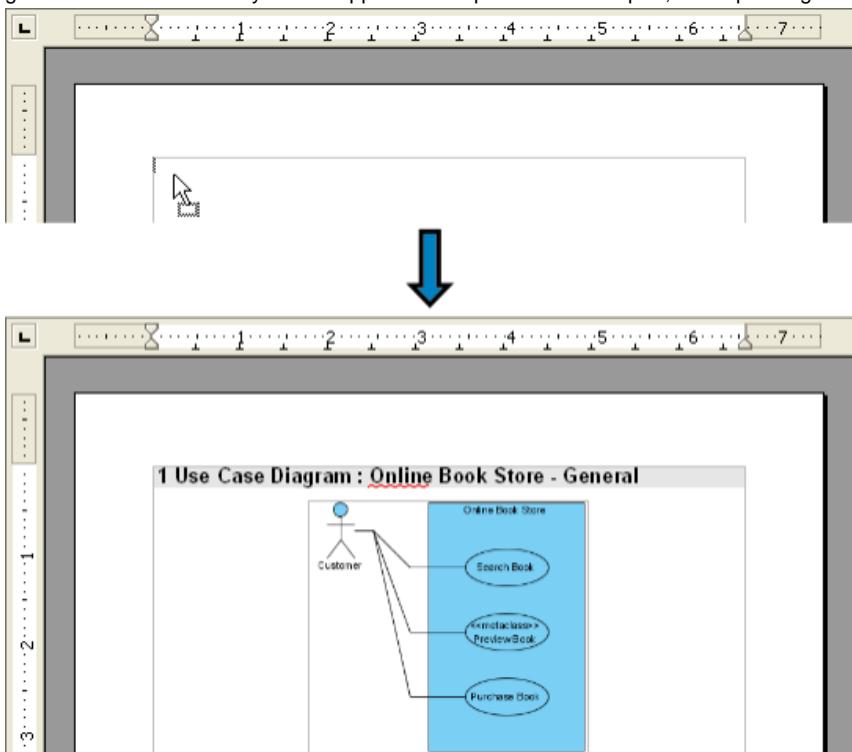
Template pane

3. Drag the desired template from the Template Pane and drop it onto the report.



Drag template from template pane

4. When the cursor drags over the Writer Pane, a tiny straight line will appear in the report indicating the position of the expected position of the generated element. Once you've dropped the template onto the report, corresponding content will be generated element to the dropped position.



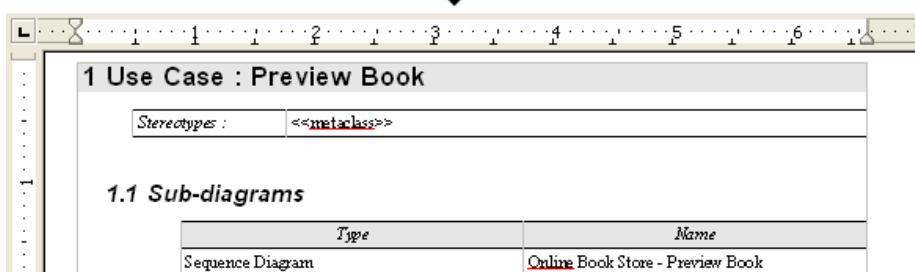
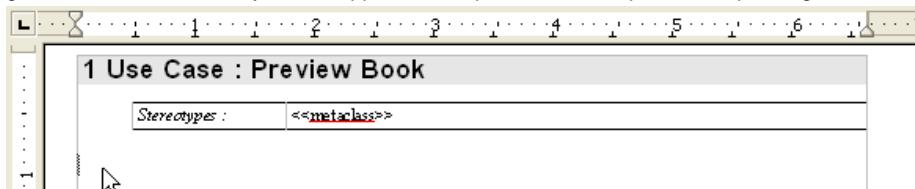
Drop template to report

5. Same as diagram, model element's template also able to drag to report.



Drag template from template pane

6. When the cursor drags over the Writer Pane, a tiny straight line will appear in the report indicating the position of the expected position of the generated element. Once you've dropped the template onto the report, corresponding content will be generated element to the dropped position.

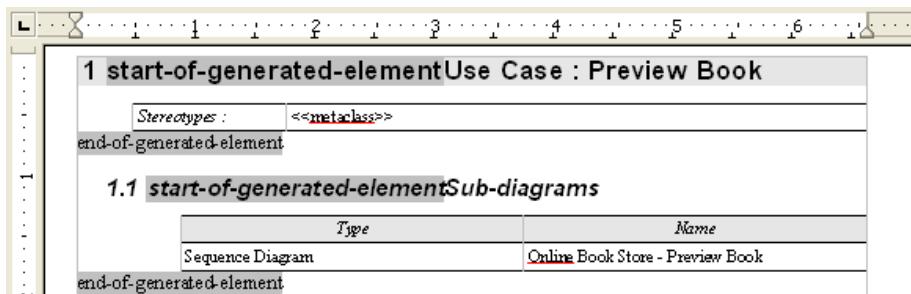


Drop template to report

Showing the bookmarks

Creating a new generated element within the boundary of an existing one is dangerous because the content may be messed up during a report update process. To avoid this, you can display bookmarks to indicate the start and end position of each generated element, and to prevent dropping a new one within the scope of the existing generated element.

To show/hide bookmarks, either check/uncheck **Tool > Show Bookmarks** from main menu to show/hide bookmarks.



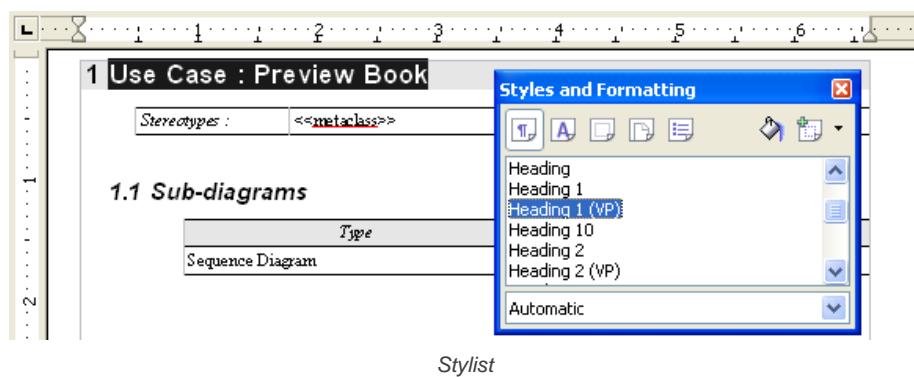
Showing bookmarks

Applying style to report

A style in Report Writer is a collection of formatting attributes that describe the nature of paragraphs. The generated element highly adopts the predefined styles in Report Writer therefore users can customize the related styles to bring consistency to the whole document. There are two ways for applying style to report.

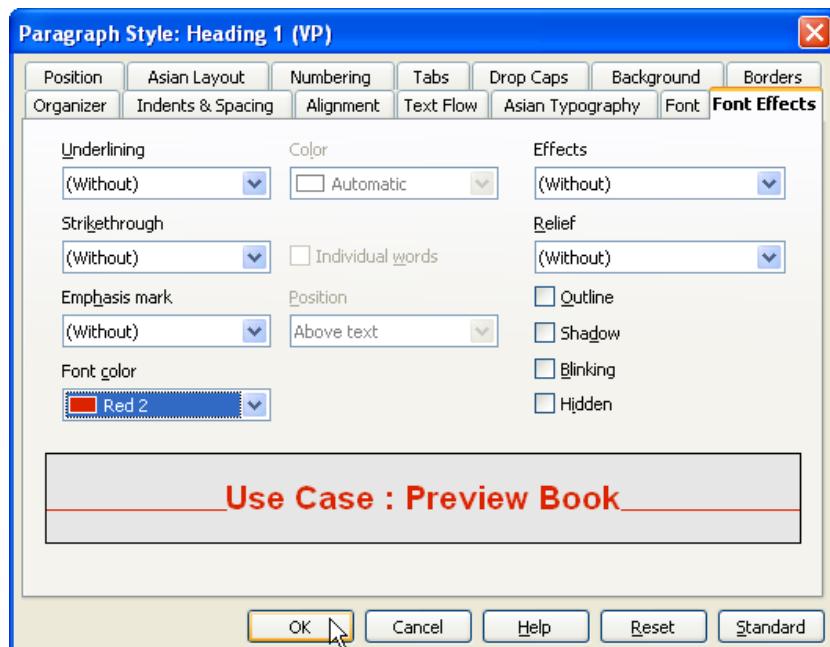
Style configuration

The **Stylist** dialog box allows you to configure the pre-defined styles. To display the **Stylist** dialog box, select **Tool > Show Stylist** from main menu. The **Stylist** dialog box appear.



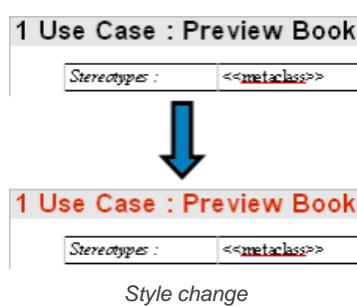
Stylist

To edit the style, right-click on the highlighted style and choose **Modify...** from the popup menu. This displays the dialog box for the selected style. You can now adjust it with your own preference. When everything is ready, please click **OK** to commit the settings and exit the dialog.



Edit style

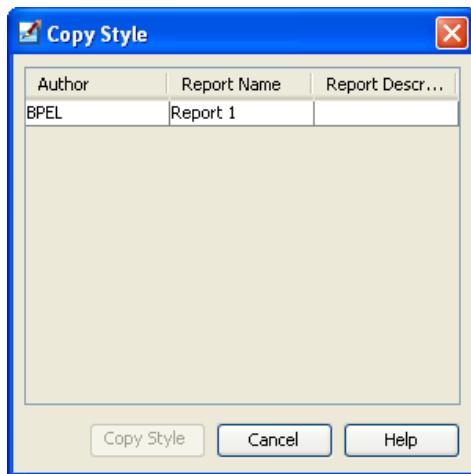
The changes will take effect immediately and you will notice the style is applied to those generated elements using the same style.



Style change

Loading style from other report

The **Copy Style** dialog box allows you to copy the style from existing report. To display the Copy Style dialog box, select **Tool > Copy Style** from main menu. The **Copy Style** dialog box appear.



Copy style dialog

Select a desired report for getting the style configuration and click **Copy Style**. The style configuration in the current report is replaced by the style configuration of the selected report. All the predefined styles will be overwritten.

Updating table of contents

There is a predefined Table of Contents in each of the report template. Here is the pre-built Table of Contents structure:

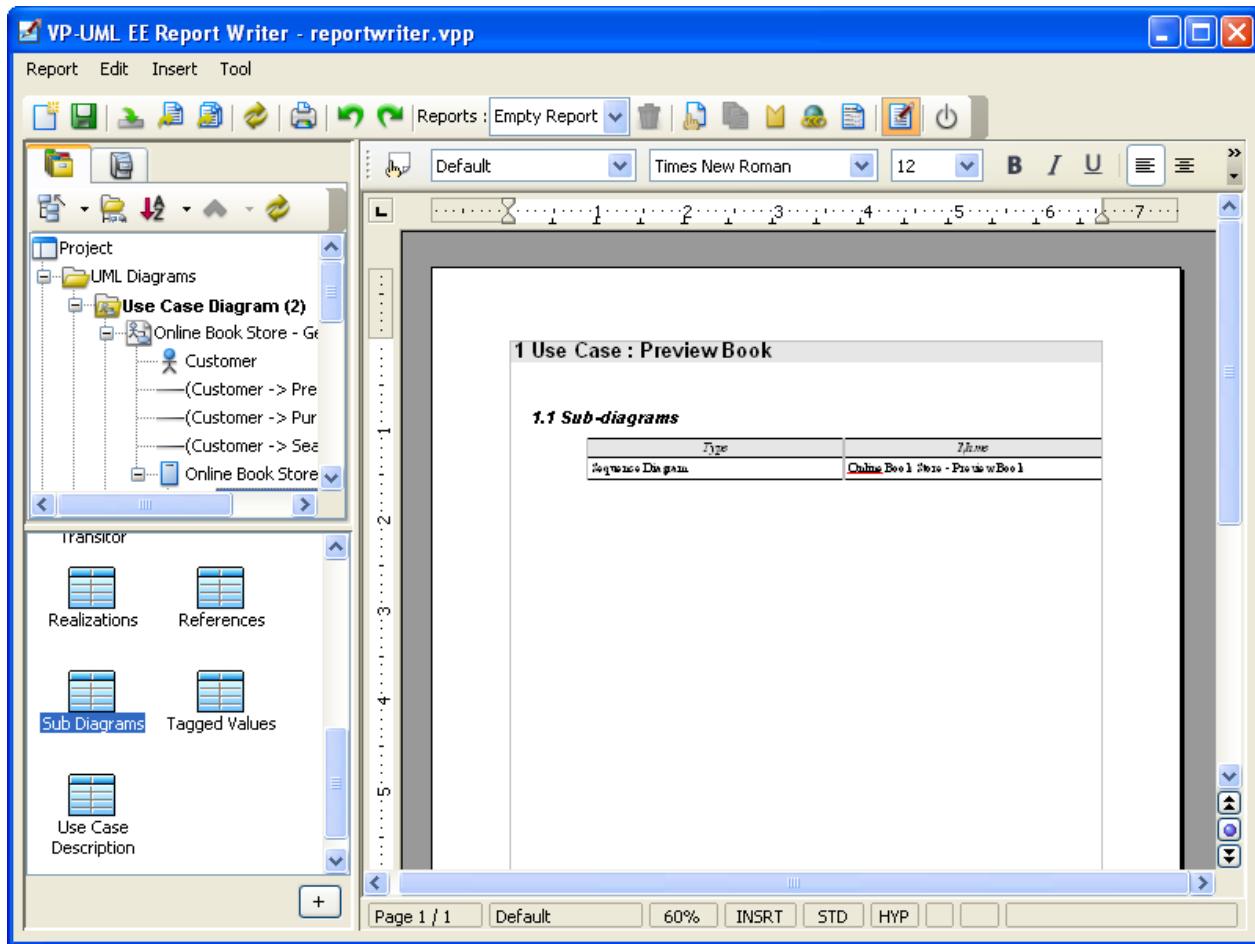
Level	Paragraph Style
0	Heading 1 (VP)
1	Heading 2 (VP)
2	Heading 3 (VP)
3-9	None

Details of predefined table of contents

To update the Table of Contents, right-click on the caption **Table of Contents** and select **Update Index/Table** from popup menu.

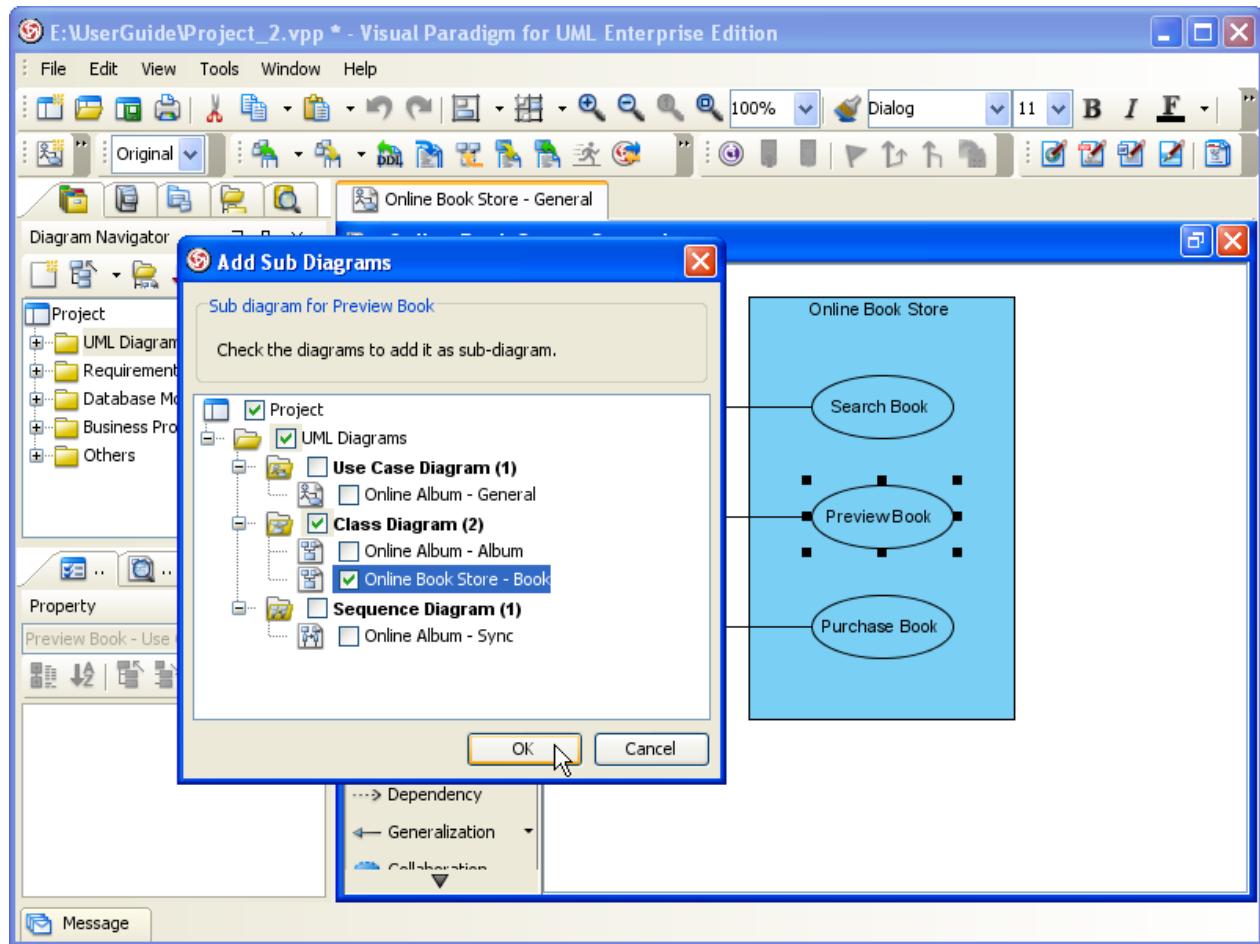
Updating report content

In reality, software design keeps evolving from time to time. Originally, users needed to modify the related documents manually to ensure that it is fully conformed to the latest design. Report Writer binds closely with the VP-UML project, and hence generated elements can then be updated without affecting the user-defined content.



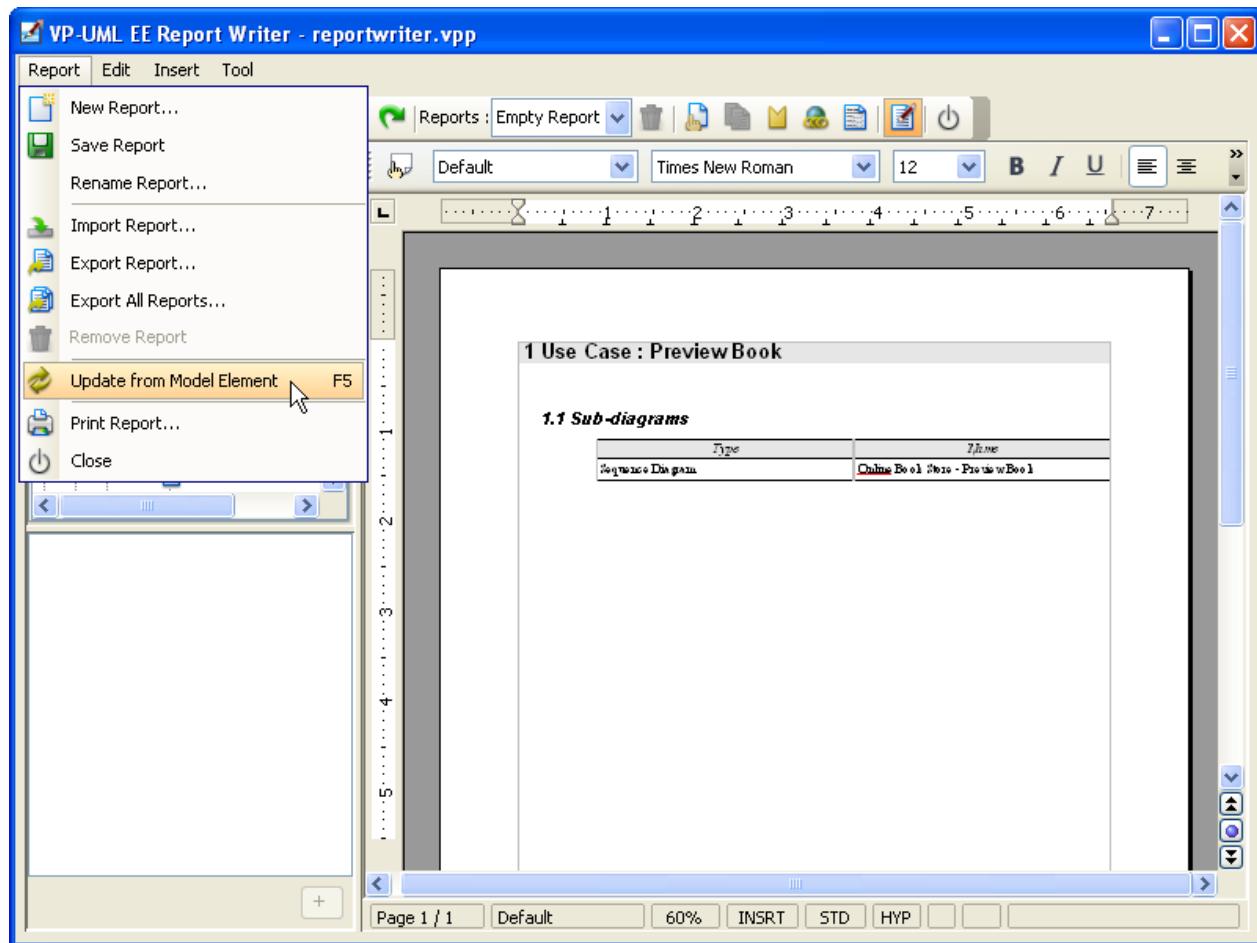
Report writer

Edit project's content which used by report.

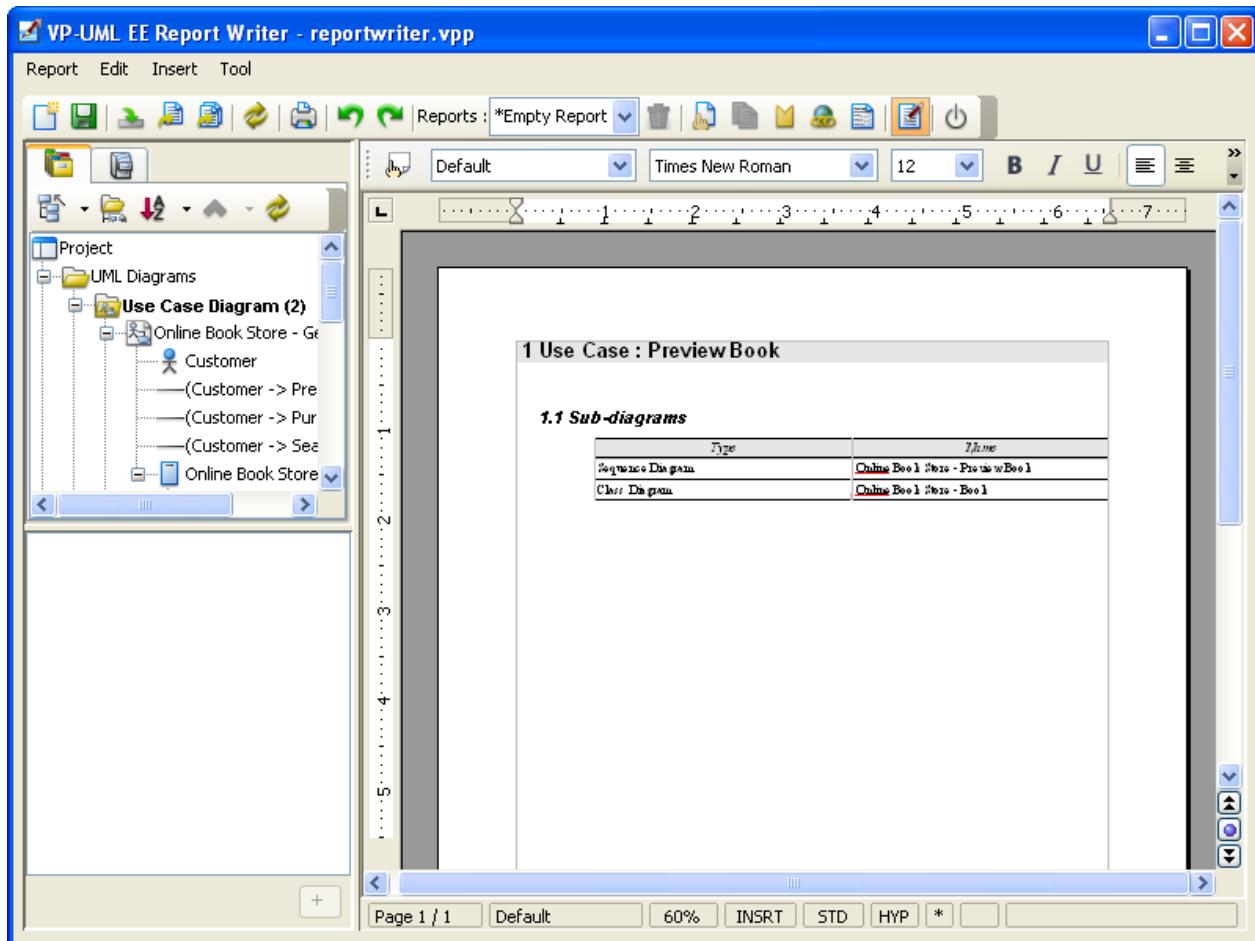


Edit project

To update a report, choose the desired report for updating from the drop-down menu and click the **Update from Model** button from the toolbar or select **Report > Update from Model** from main menu.



Update from model element



Updated report

NOTE: Please do not click on the Writer Pane while the update process is undergoing, as it may affect the accuracy of the content. It can also damage the generated element, so that updating cannot be performed anymore unless the damaged block is removed manually.

NOTE: The update process will replace ALL the contents within each generated element without notification. Therefore please insert the content carefully and ensure that it is not located inside the scope of any generated elements.

Export and import report

You can export report as file and edit it outside VP-UML. Supported format includes Microsoft Word and OpenOffice.org Text Document.

Exporting current report

1. Click on the **Export Report...** button on the toolbar or select **Report > Export Report...** from main menu. This display the **Save** dialog box.
2. In the **Save** dialog box, enter the file name and select the format for exporting.
3. Click **Save** to export the report.

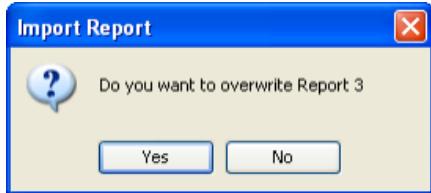
Exporting all report(s)

1. Click the **Export All Reports...** button on the toolbar or select **Report > Export All Reports...** from main menu. This display the **Save** dialog box.
2. In the **Save** dialog box, enter the directory for storing the reports in the **File name** field and select the **Document Type**.
3. When everything is ready, click **Save** to export the report(s).

Importing a report

You can import a document back into Report Writer for data updating. To import a report:

1. Click the **Import Report...** button on the toolbar or select **Report > Import Report...** from main menu. This displays the **Open** dialog box.
2. Select a file and click **Open** to import the selected document into Report Writer.
3. If the document has previously been exported from Report Writer, a dialog will appear and ask for overwriting the existing one or not.



Confirm overwrite existing report

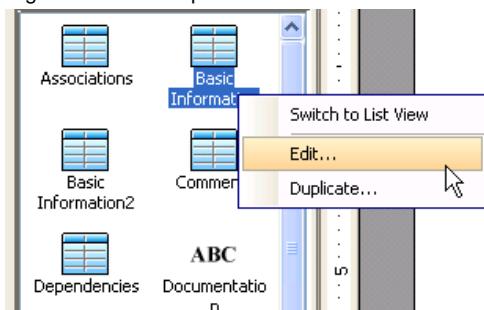
If you click **Yes**, the existing report will be replaced by the imported one. If you click **No**, the imported report will be stored into Report.

Customizing report writer template

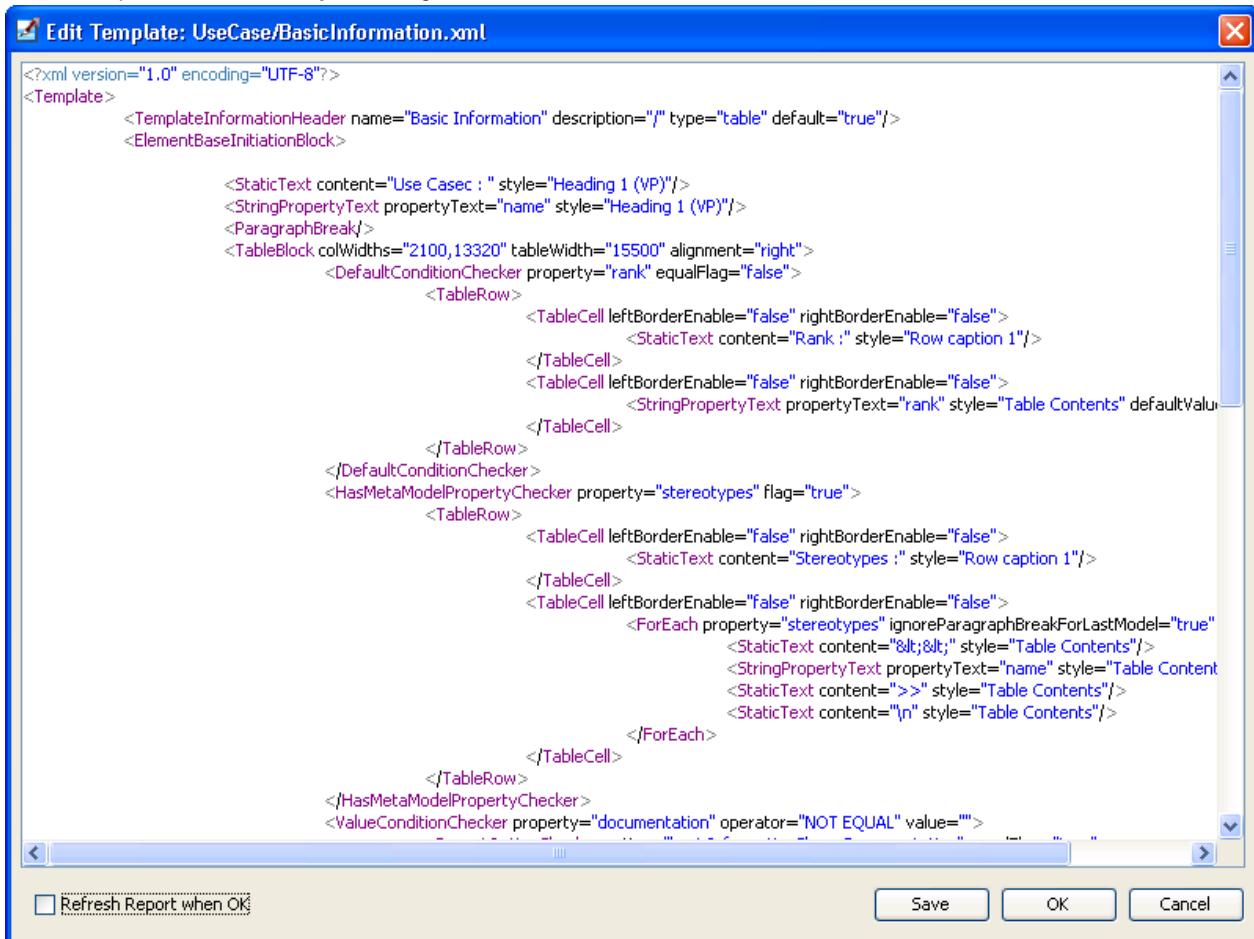
In report writer, the templates for model elements and diagrams, such as the basic information template, are formed by XML, and can be customized. By customizing a template, you can make report writer print additional customized text, to make it query other properties you wanted, or to remove part of the unneeded content, like a row in a table.

Customizing an existing template

1. Right click on a template and select **Edit...** from the popup menu.



2. Edit the template in the **Edit Template** dialog box.



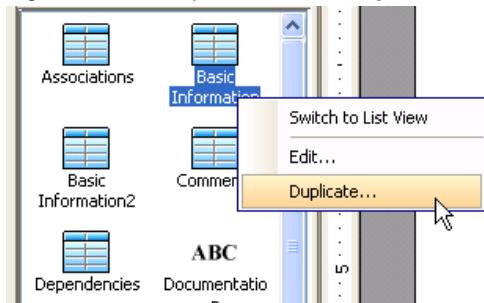
NOTE: You can save your work from time to time by pressing **Save**.

3. Click **OK** to confirm editing.

NOTE: If you have chosen **Refresh Report when OK**, the report will refresh itself to apply the changes you have made.

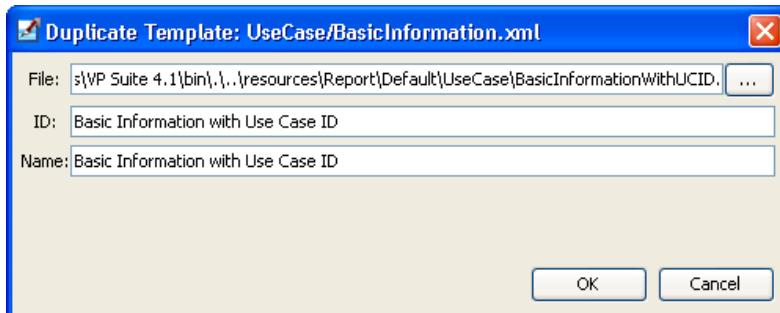
Creating a template from existing template

- Right click on a template and select **Duplicate...** from the popup menu.



To duplicate a template

- Fill in the **Duplicate Template** dialog box and click **OK**.



The duplicate template dialog box

Below is a description of the dialog box.

Field	Description
File	A template is an XML file which defines the structure of report content. By default, it will be stored in the resources folder of installation folder.
ID	The unique ID for template.
Name	The name of template, which will appear in the template list in template pane.

Details of duplicate template dialog box

- Click **OK** to confirm editing.

NOTE: If you have chosen **Refresh Report when OK**, the report will refresh itself to apply the changes you have made.

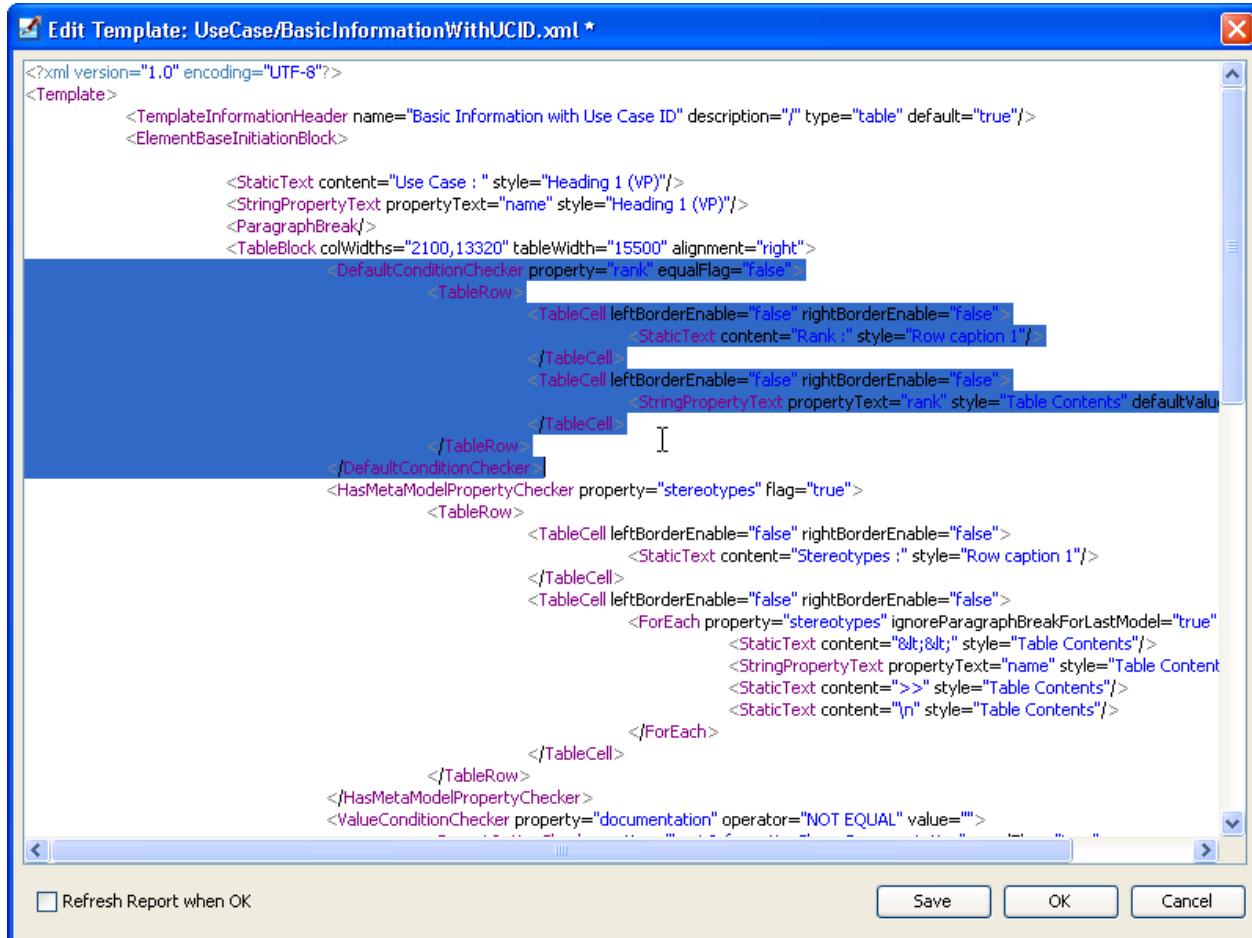
The template editor

The template editor is where you edit a template. The template is in XML format. Within the editor, you may make use of hotkey to perform operations like copy, paste and delete. Below is a description of common hotkeys.

Function	Hotkey
Copy	Ctrl-C
Paste	Ctrl-V
Delete	Delete
Cut	Ctrl-X

Hotkeys for template editor

As most of the templates are in tabular forms. We recommend users who want to add new rows to highlight and copy a trunk of row content...



Select existing content to copy and start editing

and edit the custom text and properties to query.

```
<TableBlock colWidths="2100,13320" tableWidth="15500" alignment="right">
    <DefaultConditionChecker property="useCaseID" equalFlag="false">
        <TableRow>
            <TableCell leftBorderEnable="false" rightBorderEnable="false">
                <StaticText content="ID :" style="Row caption 1"/>
            </TableCell>
            <TableCell leftBorderEnable="false" rightBorderEnable="false">
                <StringPropertyText propertyText="useCaseID" style="Table Contents"/>
            </TableCell>
        </TableRow>
    </DefaultConditionChecker>
```

1 Use Case : Process Order

ID :	UC-001
Stereotypes :	<<UseCase>>

Template content with respect to report content

Template schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.visual-paradigm.com/2008/ReportWriter1.1"
xmlns:rw="http://www.visual-paradigm.com/2008/ReportWriter1.1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
<xsd:element name="Template">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="rw:TemplateInformationHeader"/>
```

```

<xsd:choice>
  <xsd:element ref="rw:ElementBaseInitiationBlock"/>
  <xsd:element ref="rw:DiagramBaseInitiationBlock"/>
  <xsd:element ref="rw:DiagramElementBaseInitiationBlock"/>
  <xsd:element ref="rw:ProjectBaseInitiationBlock"/>
</xsd:choice>
<xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="TemplateInformationHeader">
<xsd:complexType>
  <xsd:attribute name="id" type="xsd:string" use="optional"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="description" type="xsd:string" use="optional"/>
  <xsd:attribute name="default" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="type" use="optional" default="text"/>
<xsd:simpleType>
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="text"/>
    <xsd:enumeration value="image"/>
    <xsd:enumeration value="table"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ElementBaseInitiationBlock">
<xsd:complexType>
<xsd:sequence>
  <!-- any element after the comment "CONTENTS"-->
  <xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="DiagramBaseInitiationBlock">
<xsd:complexType>
<xsd:sequence>
  <!-- any element after the comment "CONTENTS"-->
  <xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="DiagramElementBaseInitiationBlock">
<xsd:complexType>
<xsd:sequence>
  <!-- any element after the comment "CONTENTS"-->
  <xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="ProjectBaseInitiationBlock">
<xsd:complexType>
<xsd:sequence>
  <!-- any element after the comment "CONTENTS"-->
  <xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!-- CONTENTS -->
<!-- Breaks, Text -->
<xsd:element name="ParagraphBreak"/>
<xsd:element name="PageBreak"/>
<xsd:element name="StaticText">
<xsd:complexType>
  <xsd:attribute name="content" type="xsd:string" use="optional"/>
  <xsd:attribute name="isBold" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="isItalic" type="xsd:boolean" use="optional" default="false"/>
  <xsd:attribute name="fontFamily" type="xsd:string" use="optional" default="Times New Roman"/>
  <xsd:attribute name="alignment" use="optional" default="left"/>
<xsd:simpleType>
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="left"/>
    <xsd:enumeration value="center"/>
    <xsd:enumeration value="right"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="fontSize" type="xsd:integer" use="optional" default="12"/>
<xsd:attribute name="style" type="xsd:string" use="optional"/>

```

```

<xsd:attribute name="numberingLevel" type="xsd:short" use="optional"/>
<xsd:attribute name="foreColor" type="rw:color" use="optional"/>
<xsd:attribute name="indentation" type="xsd:double" use="optional"/>
</xsd:complexType>
</xsd:element>
<!-- Table -->
<xsd:element name="TableBlock">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="rw:TableRow" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="tableWidth" type="xsd:integer" use="optional" default="10000">
<xsd:annotation>
<xsd:documentation>Document is in A4 size (21000*27000)</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="colWidths" type="rw:integers" use="optional">
<xsd:annotation>
<xsd:documentation>
@colWidths is relative to @tableWidth.
For example, tableWidth="15000" colWidths="10, 20, 70", finally the columns widths will be:
1500 (15000*10/(10+20+70))
3000 (15000*20/(10+20+70))
10500 (15000*70/(10+20+70))
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="alignment" use="optional" default="center">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="left"/>
<xsd:enumeration value="center"/>
<xsd:enumeration value="right"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="rowBackgroundColors" type="rw:colors" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="TableRow">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="rw:TableCell" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="height" type="xsd:integer" use="optional"/>
<xsd:attribute name="backgroundColor" type="rw:color" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="TableCell">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="rw:TableCell" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="topBorderEnable" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="bottomBorderEnable" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="leftBorderEnable" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="rightBorderEnable" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="color" type="rw:color" use="optional"/>
<xsd:attribute name="splitted" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>
</xsd:element>
<!--
Iteration, retrieve a list of values
(IterationBlock, ForEach, ForEachSimpleRelationship, ForEachRelationshipEnd, ForEachSubDiagram, ForEachOwnerDiagram, ForEachDiagram)
-->
<xsd:element name="IterationBlock">
<xsd:annotation>
<xsd:documentation>
Retrieve children from Project/Model/Diagram/DiagramElement.
Project and Model will be retrieved children Model.
Diagram and DiagramElement will be retrieved children DiagramElement.
</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="elementType" type="xsd:string" use="optional">

```

```

<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelType (Model's modelType or DiagramElement's shapeType)</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter children by modelType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="elementTypes" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelTypes (Model's modelTypes or DiagramElement's shapeTypes)</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelTypes" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>Filter children by modelTypes. If @modelType is defined, @modelTypes will be ignored.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="name" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter children by name.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="filterHidden" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Filter hidden children DiagramElement.
For retrieve from Diagram/DiagramElement only.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="includeConnector" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @includeConnectors</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="includeConnectors" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Retrieve Shape+Connector from Diagram. Otherwise, only the Shape will retrieved.
For retrieve from Diagram only.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="byBounds" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Retrieve DiagramElement by Bounds.
sometimes, the parent shape "contains" another shape. But another shape is not child of the shape.
e.g. BPGroup won't be parent of the shapes.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="byBoundsInAllLevel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="allLevel" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Retrieve all models of the project. Otherwise, only the "root level models (models have no parent model)" will be retrieved.
For retrieve from Project only.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="breakString" type="xsd:string" use="optional"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBys</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">

```

```

<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:booleans" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="identifier" type="xsd:string" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="ForEach">
<xsd:annotation>
<xsd:documentation>Retrieve Model values from Model's property.</xsd:documentation>
</xsd:annotation>
</xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="property" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Retrieve from which property.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="breakString" type="xsd:string" use="optional"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBys</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
if sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property

```

```

</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:boolean" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ForEachSimpleRelationship">
<xsd:annotation>
<xsd:documentation>Retrieve SimpleRelationship from Model, or Connector from DiagramElement</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="type" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelType (Model's modelType or DiagramElement's shapeType)</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter relationships by modelType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelTypes" type="rw:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter relationships by modelTypes. If @modelType is defined, @modelTypes will be ignored.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="direction" use="optional">
<xsd:annotation>
<xsd:documentation>Filter relationship by direction.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="all"/>
<xsd:enumeration value="from"/>
<xsd:enumeration value="to"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="breakString" type="xsd:string" use="optional"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBys</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:string" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">

```

```

<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:booleans" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ForEachRelationshipEnd">
<xsd:annotation>
<xsd:documentation>Retrieve SimpleRelationship from Model, or Connector from DiagramElement</xsd:documentation>
</xsd:annotation>
</xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="type" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelType (Model's modelType or DiagramElement's shapeType)</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter relationships by modelType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelTypes" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>Filter relationships by modelTypes. If @modelType is defined, @modelTypes will be ignored.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="endPointer" use="optional">
<xsd:annotation>
<xsd:documentation>
Filter relationship by direction.
both = all, self = from, other = to
</xsd:documentation>
</xsd:annotation>
</xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="all"/>
<xsd:enumeration value="from"/>
<xsd:enumeration value="to"/>
<xsd:enumeration value="both"/>
<xsd:enumeration value="self"/>
<xsd:enumeration value="other"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>

```

```

<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="breakString" type="xsd:string" use="optional"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>Deprecated, replaced by @sortBys</xsd:annotation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:boolean" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:complexType>
</xsd:element>
<xsd:element name="ForEachSubDiagram">
<xsd:annotation>
<xsd:documentation>Retrieve sub diagrams from Model</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="diagramType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter diagram by diagramType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBys</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>

```

```

</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:booleans" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ForEachOwnerDiagram">
<xsd:annotation>
<xsd:documentation>Retrieve owner diagrams which the Model is shown on</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="diagramType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter diagram by diagramType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBy</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBy" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:strings" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property

```

```

</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:boolean" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ForEachDiagram">
<xsd:annotation>
<xsd:documentation>Retrieve diagrams from Project</xsd:documentation>
</xsd:annotation>
</xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="diagramType" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter diagram by diagramType.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="ignoreParagraphBreakForLastModel" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="sortBy" type="xsd:NMTOKEN" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @sortBys</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortBys" type="rw:string" use="optional">
<xsd:annotation>
<xsd:documentation>
Sort by following options:
name | modelType | property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortProperty" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortProperty is required to specified sort by which property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="sortValues" type="rw:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @sortValues can be specified for sort by values of the property
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="defaultPropertyValue" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>
If sort by property, @defaultPropertyValue can be specified for the default value of the model has no this property value
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSort" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @descendingSort</xsd:documentation>

```

```

</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="descendingSorts" type="rw:boolean" use="optional">
<xsd:annotation>
<xsd:documentation>
Sorting in descending order?
numbers of descendingSorts should be same as sortBys
e.g. @sortBy="name, modelType" @descendingSorts="true, false"
then, sort by name will be descending, sort by modelType will be ascending
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<!--
Conditional, if condition not match, children template node won't be executed
(DefaultConditionChecker, ValueConditionChecker, HasChildElementChecker, HasMetaModelPropertyChecker, HasParentModelChecker,
HasRelationshipChecker, ReportOptionChecker, HasDiagramChecker)
-->
<xsd:element name="DefaultConditionChecker">
<xsd:annotation>
<xsd:documentation>Check property value equals to its default value.</xsd:documentation>
</xsd:annotation>
</xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="equalFlag" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @flag</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="false"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="ValueConditionChecker">
<xsd:annotation>
<xsd:documentation>Check property value.</xsd:documentation>
</xsd:annotation>
</xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="propertyType" use="optional" default="string">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="string"/>
<xsd:enumeration value="int"/>
<xsd:enumeration value="boolean"/>
<xsd:enumeration value="model"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="operator" use="required">
<xsd:annotation>
<xsd:documentation>
"equal" is deprecated, replaced by "equals"
"not equal" is deprecated, replaced by "not equals"
>equals", "not equals" for string/int/boolean/model property
"less than", "equals or less than", "greater than", "equals or greater than" for string/int property
"like", "not like" for string property
</xsd:documentation>
</xsd:annotation>
</xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="equals"/>
<xsd:enumeration value="not equals"/>
<xsd:enumeration value="less than"/>
<xsd:enumeration value="equals or less than"/>
<xsd:enumeration value="greater than"/>
<xsd:enumeration value="equals or greater than"/>
<xsd:enumeration value="like"/>
<xsd:enumeration value="not like"/>
<xsd:enumeration value="equal"/>
<xsd:enumeration value="not equal"/>
</xsd:restriction>
</xsd:simpleType>

```

```

</xsd:attribute>
<xsd:attribute name="expectedValue" type="xsd:string" use="optional"/>
<xsd:attribute name="caseSensitive" type="xsd:boolean" use="optional" default="true">
<xsd:annotation>Case sensitive/insensitive for string property</xsd:annotation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="HasChildElementChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="type" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelType</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelType" type="xsd:string" use="optional"/>
<xsd:attribute name="elementTypes" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @modelTypes</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelTypes" type="xsd:string" use="optional"/>
<xsd:attribute name="includeConnector" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @includeConnectors</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="includeConnectors" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Retrieve Shape+Connector from Diagram. Otherwise, only the Shape will retrieved.  

For retrieve from Diagram only.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="allLevel" type="xsd:boolean" use="optional" default="false">
<xsd:annotation>
<xsd:documentation>
Retrieve all models of the project. Otherwise, only the "root level models (models have no parent model)" will be retrieved.  

For retrieve from Project only.
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="HasMetaModelPropertyChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="property" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="HasParentModelChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="true"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="HasRelationshipChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="type" type="xsd:string" use="optional">
<xsd:annotation>

```

```

<xsd:documentation>Deprecated, replaced by @modelType</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="modelType" type="xsd:string" use="optional"/>
<xsd:attribute name="modelTypes" type="xsd:string" use="optional"/>
<xsd:attribute name="direction" use="optional" default="all">
</xsd:annotation>
</xsd:documentation>
self_begins, self_ends are deprecated
self_begins = from
self_ends = to
</xsd:documentation>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="all"/>
<xsd:enumeration value="from"/>
<xsd:enumeration value="to"/>
<xsd:enumeration value="self_begins"/>
<xsd:enumeration value="self_ends"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="ReportOptionChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="equalFlag" type="xsd:boolean" use="optional" default="false">
</xsd:annotation>
<xsd:documentation>Deprecated, replaced by @flag</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="option" type="xsd:string" use="optional">
</xsd:annotation>
</xsd:documentation>
Check the option. Now, only one option is supported. Which is:
basicInformationShowsDocumentation
user can switch the option on Report Writer main menu: Tool &gt; Options &gt;
</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="HasDiagramChecker">
<xsd:complexType>
<xsd:sequence>
<!-- any element after the comment "CONTENTS"-->
<xsd:any minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="flag" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="type" type="xsd:string" use="optional">
</xsd:annotation>
<xsd:documentation>Deprecated, replaced by @diagramType</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="diagramType" type="xsd:string" use="optional"/>
</xsd:complexType>
</xsd:element>
<!--
Property, getting property value from model
(StringPropertyText, IntPropertyText, BooleanPropertyText, MetaModelElement, FromEnd, ToEnd, RelationshipEndEndRelationship,
RelationshipEndOppositeEnd, ParentModel, DiagramProperty)
-->
<xsd:element name="StringPropertyText">
<xsd:complexType>
<xsd:attribute name="propertyText" type="xsd:string" use="optional">
</xsd:annotation>
</xsd:documentation>Deprecated, replaced by @property</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="isHTML" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="forcePlainText" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="defaultValue" type="xsd:string" use="optional"/>
<xsd:attribute name="property" type="xsd:string" use="optional"/>

```

```

<xsd:attribute name="isBold" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="isItalic" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="fontFamily" type="xsd:string" use="optional" default="Times New Roman"/>
<xsd:attribute name="alignment" use="optional" default="left">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="left"/>
<xsd:enumeration value="center"/>
<xsd:enumeration value="right"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="fontSize" type="xsd:integer" use="optional" default="12"/>
<xsd:attribute name="style" type="xsd:string" use="optional"/>
<xsd:attribute name="numberingLevel" type="xsd:short" use="optional"/>
<xsd:attribute name="foreColor" type="rw:color" use="optional"/>
<xsd:attribute name="indentation" type="xsd:double" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="IntPropertyText">
<xsd:complexType>
<xsd:attribute name="propertyText" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @property</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="isHTML" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="property" type="xsd:string" use="optional"/>
<xsd:attribute name="isBold" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="isItalic" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="fontFamily" type="xsd:string" use="optional" default="Times New Roman"/>
<xsd:attribute name="alignment" use="optional" default="left">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="left"/>
<xsd:enumeration value="center"/>
<xsd:enumeration value="right"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="fontSize" type="xsd:integer" use="optional" default="12"/>
<xsd:attribute name="style" type="xsd:string" use="optional"/>
<xsd:attribute name="numberingLevel" type="xsd:short" use="optional"/>
<xsd:attribute name="foreColor" type="rw:color" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="BooleanPropertyText">
<xsd:complexType>
<xsd:attribute name="propertyText" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @property</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="property" type="xsd:string" use="optional"/>
<xsd:attribute name="isBold" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="isItalic" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="fontFamily" type="xsd:string" use="optional" default="Times New Roman"/>
<xsd:attribute name="alignment" use="optional" default="left">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="left"/>
<xsd:enumeration value="center"/>
<xsd:enumeration value="right"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="fontSize" type="xsd:integer" use="optional" default="12"/>
<xsd:attribute name="style" type="xsd:string" use="optional"/>
<xsd:attribute name="numberingLevel" type="xsd:short" use="optional"/>
<xsd:attribute name="foreColor" type="rw:color" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="MetaModelElement">
<xsd:annotation>
<xsd:documentation>Retrieve Model from property.</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:attribute name="mode" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="modelProperty" type="xsd:string" use="optional">

```

```

<xsd:annotation>
<xsd:documentation>Deprecated, replaced by @property</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="identifier" type="xsd:string" use="optional"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="FromEnd">
<xsd:annotation>
<xsd:documentation>Retrieve From RelationshipEnd from EndRelationship, or From model from SimpleRelationship.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="ToEnd">
<xsd:annotation>
<xsd:documentation>Retrieve To RelationshipEnd from EndRelationship, or To model from SimpleRelationship.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="RelationshipEndEndRelationship">
<xsd:annotation>
<xsd:documentation>Retrieve EndRelationship from RelationshipEnd.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="RelationshipEndOppositeEnd">
<xsd:annotation>
<xsd:documentation>Retrieve opposite end from RelationshipEnd.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="ParentModel">
<xsd:annotation>
<xsd:documentation>Retrieve parent model from Model.</xsd:documentation>
</xsd:annotation>
</xsd:element>
<xsd:element name="DiagramProperty">
<xsd:annotation>
<xsd:documentation>Retrieve diagram from property. e.g. BPSubProcess.diagramId refer its bpd.</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:attribute name="property" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:element>
<!--
Others
(ElementImage, FlowOfEventsTemplate)
-->
<xsd:element name="ElementImage">
<xsd:annotation>
<xsd:documentation>Showing image of Diagram/DiagramElement, or icon for model type.</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:attribute name="imageType" use="optional" default="icon">
<xsd:annotation>
<xsd:documentation>
"diagram" means show image for Diagram/DiagramElement
"icon" means show icon for the model type
</xsd:documentation>
</xsd:annotation>
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="diagram"/>
<xsd:enumeration value="icon"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="alignment" use="optional" default="left">
<xsd:simpleType>
<xsd:restriction base="xsd:NMTOKEN">
<xsd:enumeration value="left"/>
<xsd:enumeration value="center"/>
<xsd:enumeration value="right"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="style" type="xsd:string" use="optional"/>
<xsd:attribute name="width" type="xsd:integer" use="optional"/>
<xsd:attribute name="height" type="xsd:integer" use="optional"/>
<xsd:attribute name="maxWidth" type="xsd:integer" use="optional"/>
<xsd:attribute name="maxHeight" type="xsd:integer" use="optional"/>
</xsd:complexType>
</xsd:element>

```

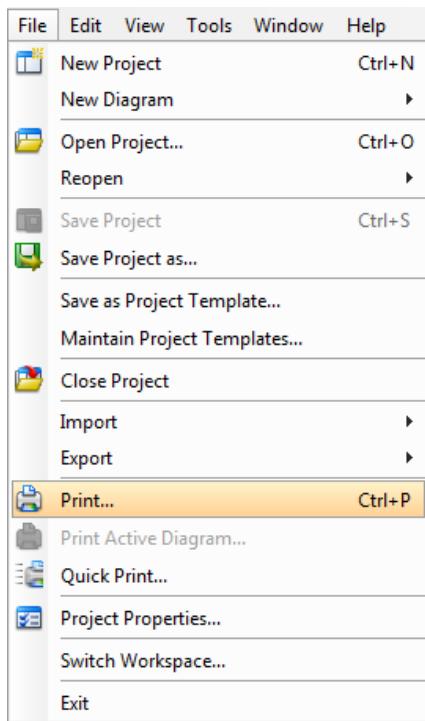
```

<xsd:element name="FlowOfEventsTemplate">
<xsd:annotation>
<xsd:documentation>Showing Flow Of Events for a Use Case.</xsd:documentation>
</xsd:annotation>
<xsd:complexType>
<xsd:attribute name="showProcedure" use="optional" default="follow model">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:enumeration value="follow model"/>
<xsd:enumeration value="true"/>
<xsd:enumeration value="false"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="tableWidth" type="xsd:integer" use="optional" default="15500"/>
<xsd:attribute name="showRowBorder" type="xsd:boolean" use="optional" default="false"/>
<xsd:attribute name="showColumnBorder" type="xsd:boolean" use="optional" default="true"/>
<xsd:attribute name="showName" type="xsd:boolean" use="optional" default="true">
<xsd:annotation>
<xsd:documentation>Show the name of Flow of event?</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
<xsd:attribute name="columnWidths" type="rw:integers" use="optional"/>
<xsd:attribute name="name" type="xsd:string" use="optional">
<xsd:annotation>
<xsd:documentation>Filter Flow of events by name.</xsd:documentation>
</xsd:annotation>
</xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:simpleType name="strings">
<xsd:annotation>
<xsd:documentation>Strings separated by ", ". e.g. "Class, UseCase, Actor" means "Class", "UseCase", "Actor"</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="integers">
<xsd:annotation>
<xsd:documentation>Integers separated by ", ". e.g. "10, 20, 70"</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="booleans">
<xsd:annotation>
<xsd:documentation>Booleans separated by ", ". e.g. "true, false, false"</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="color">
<xsd:annotation>
<xsd:documentation>
Color can be represented in following 2 forms:
1. "R, G, B" in decimal, e.g. "230, 230, 230" (it is a light gray color)
2. #FFFFFF (RGB in hexadecimal), e.g. #FF0000 (it is a red color)
Prefer using #FFFFFF. Because "R, G, B" is not suitable for colors.
</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="colors">
<xsd:annotation>
<xsd:documentation>
Colors separated by ", ". e.g. "#FF0000, #00FF00, #0000FF".
</xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string"/>
</xsd:simpleType>
</xsd:schema>

```

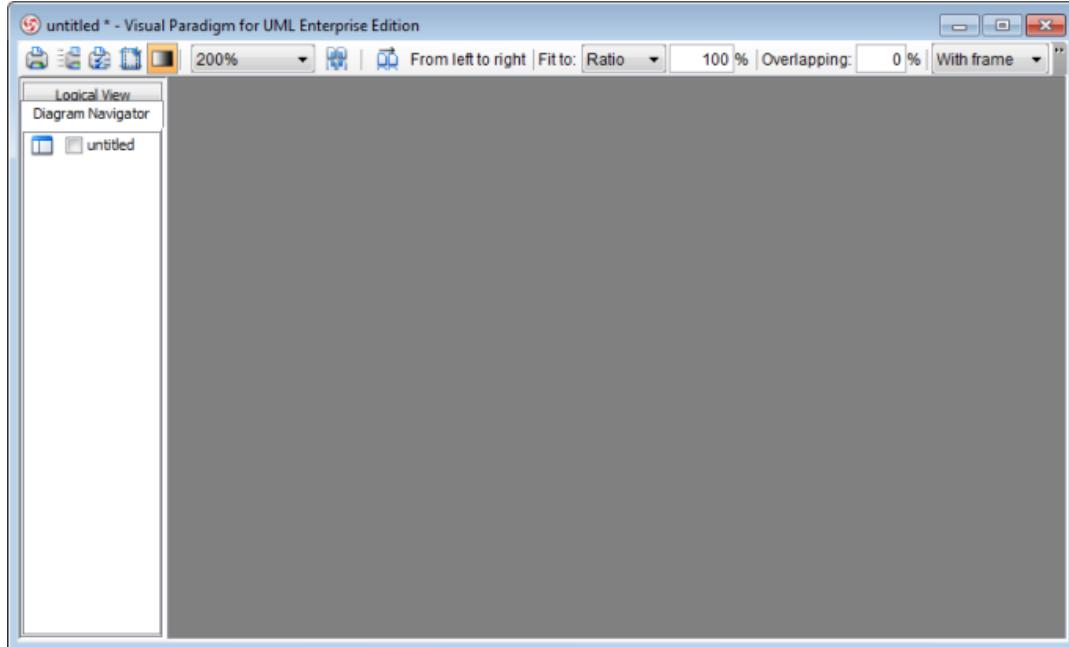
Printing diagrams

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 main menu.



To Perform Printing

An overview of Print window



Print preview dialog box

The toolbar of the print preview pane allows you to configure the print settings. The buttons and their description 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.
	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	Set the diagram size to fit to the specified ratio.
	Fit to Pages	Set the diagram to be printed on the specified number of pages.
	Overlapping	Set the percentage of the margins to overlap among adjacent pages.
	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.
	Help	Call the VP-UML help file.
	Close Print Preview	Close the print preview pane and return to the design area.

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.

- Select the desired diagram(s) for printing. The selected diagram(s) will be shown in the preview area.

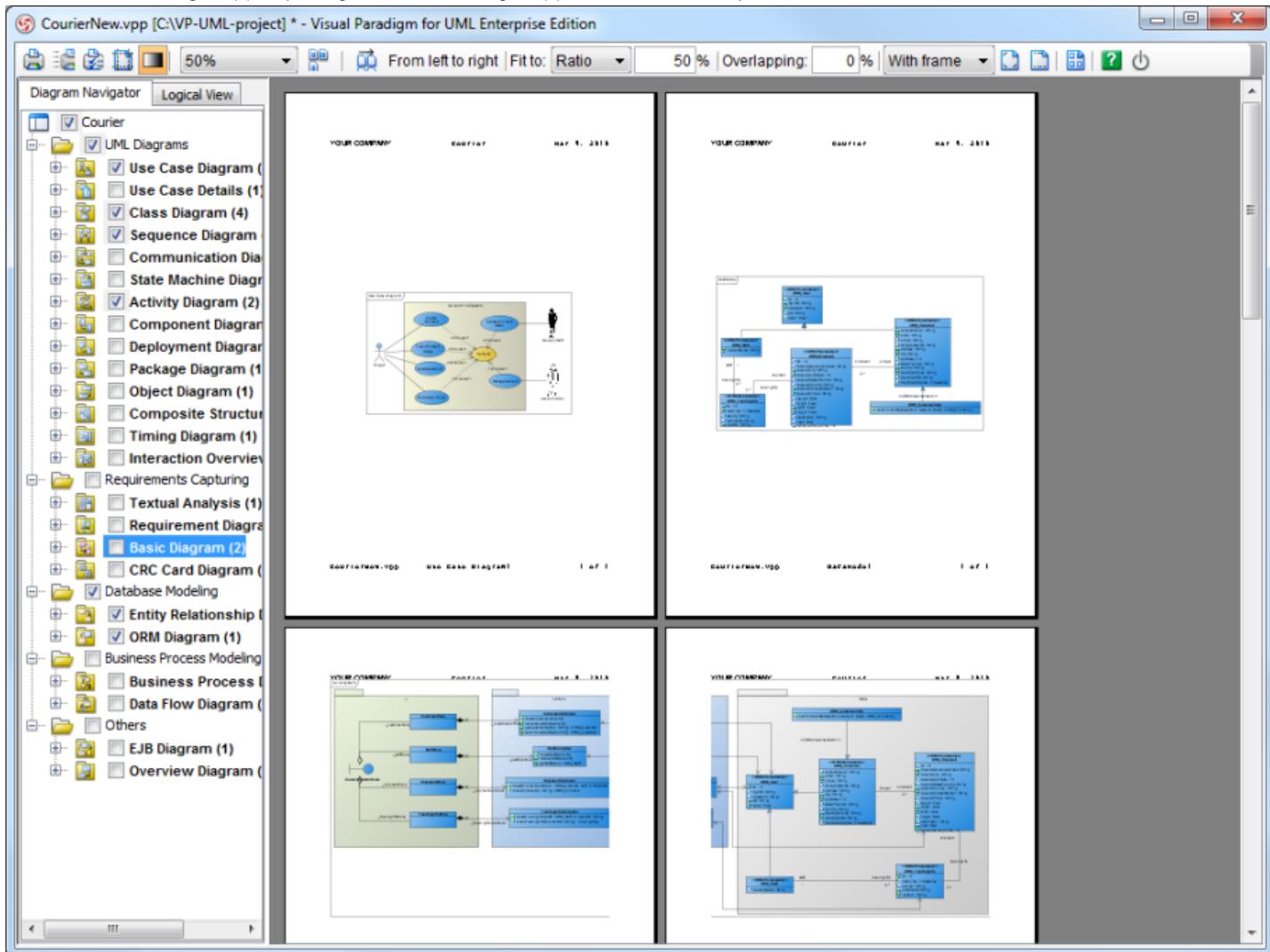
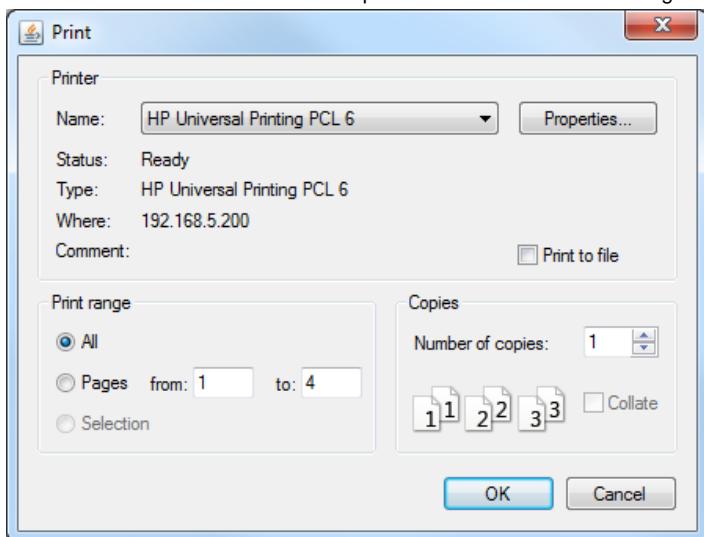


Diagram preview

- Click the **Print** button on the Print preview toolbar. The **Print** dialog box will be shown.

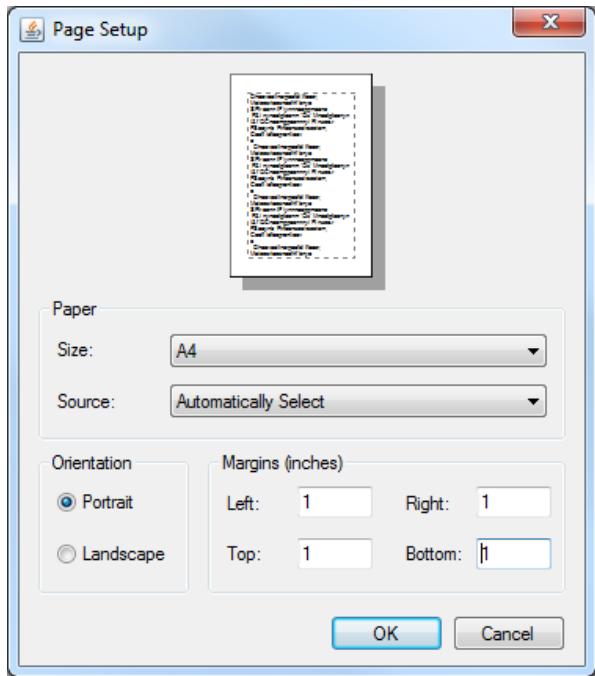


The **Print** dialog box

- 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.
- 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.

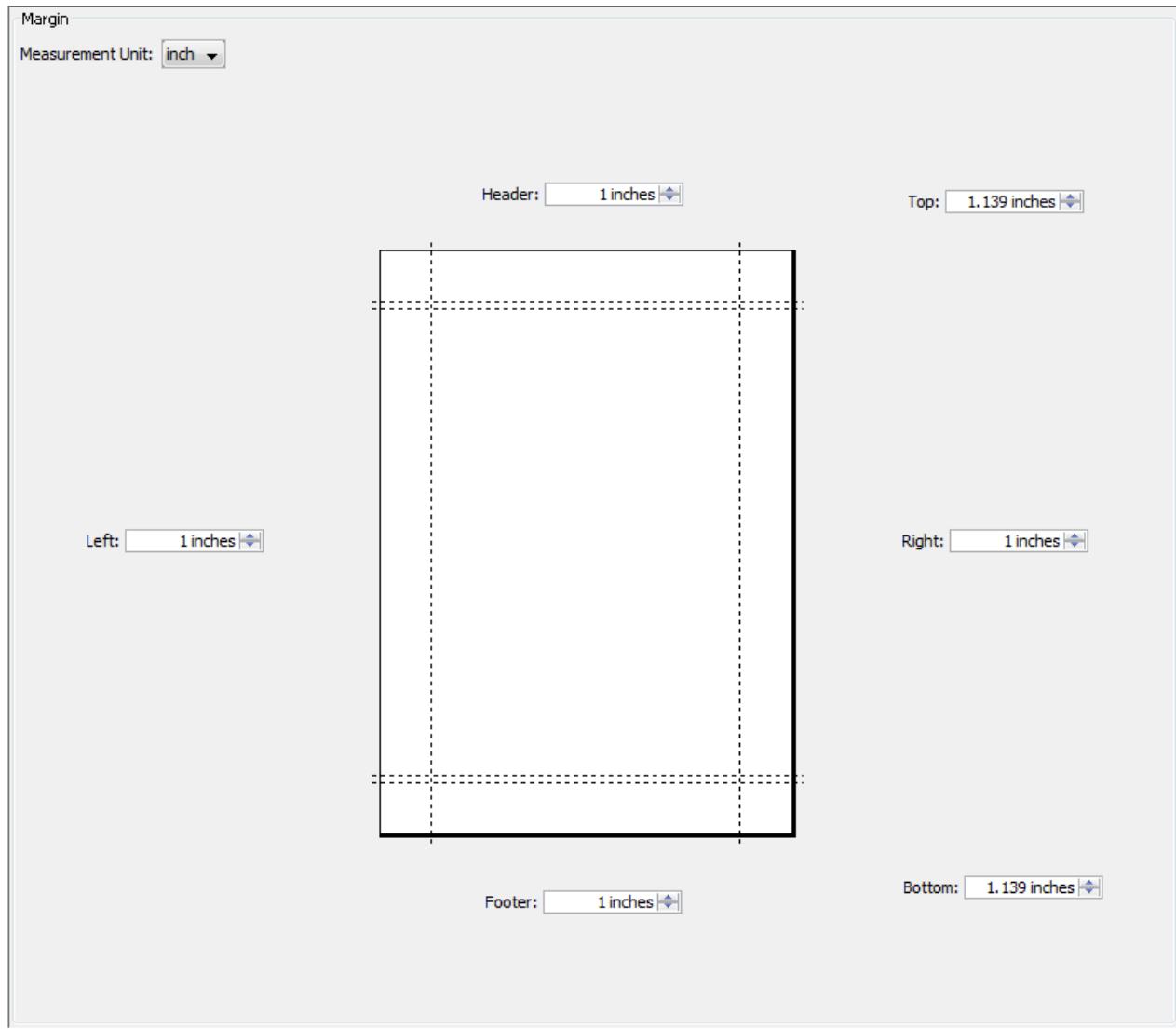


The **Page setup** dialog box

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** under **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.



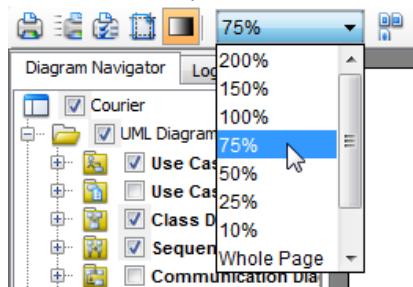
Adjusting margins

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

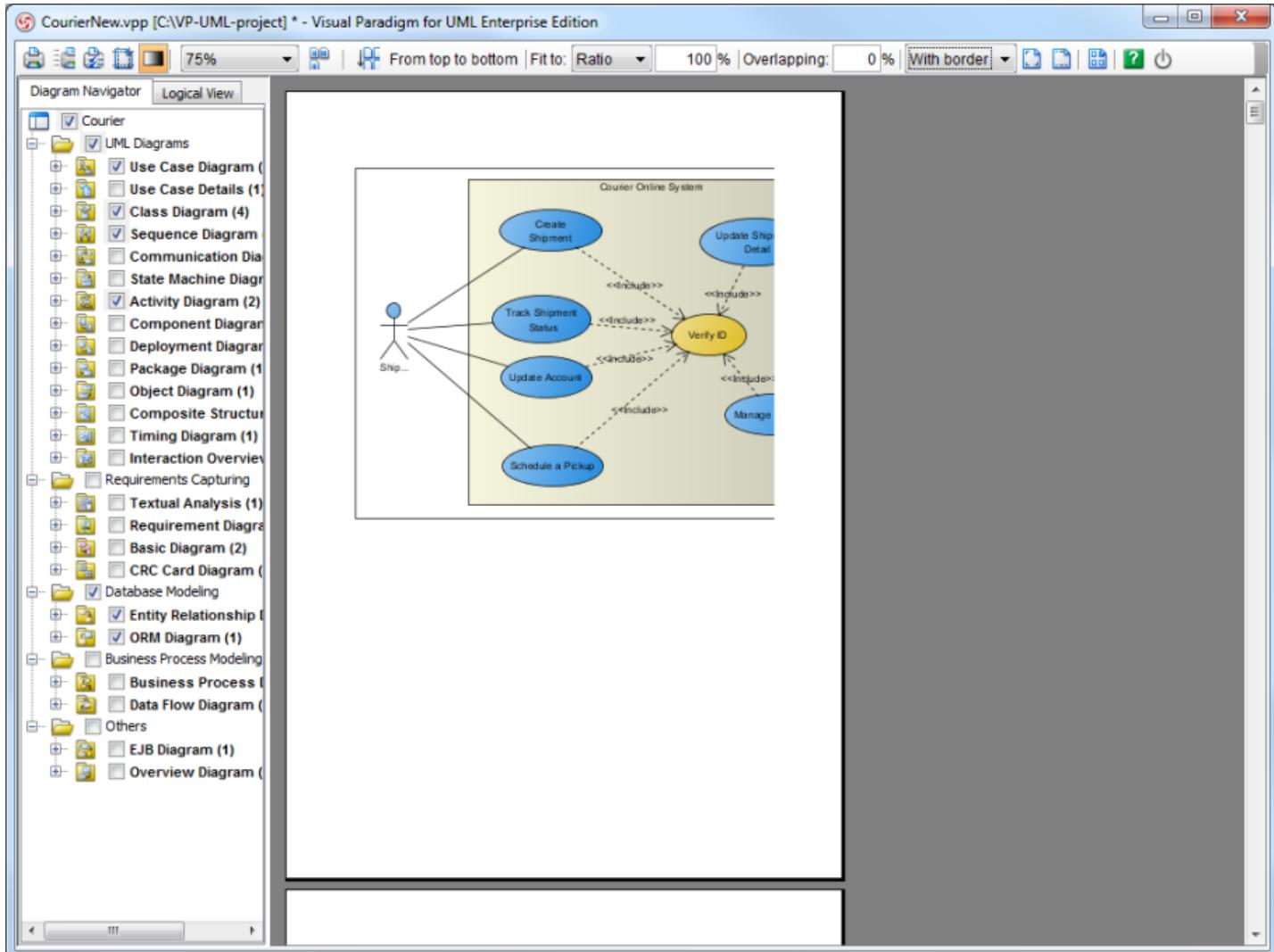
Diagrams can be zoomed in or zoomed out according to the user's preference.

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.



Preview in Preview dialog box

Selecting the preview layout

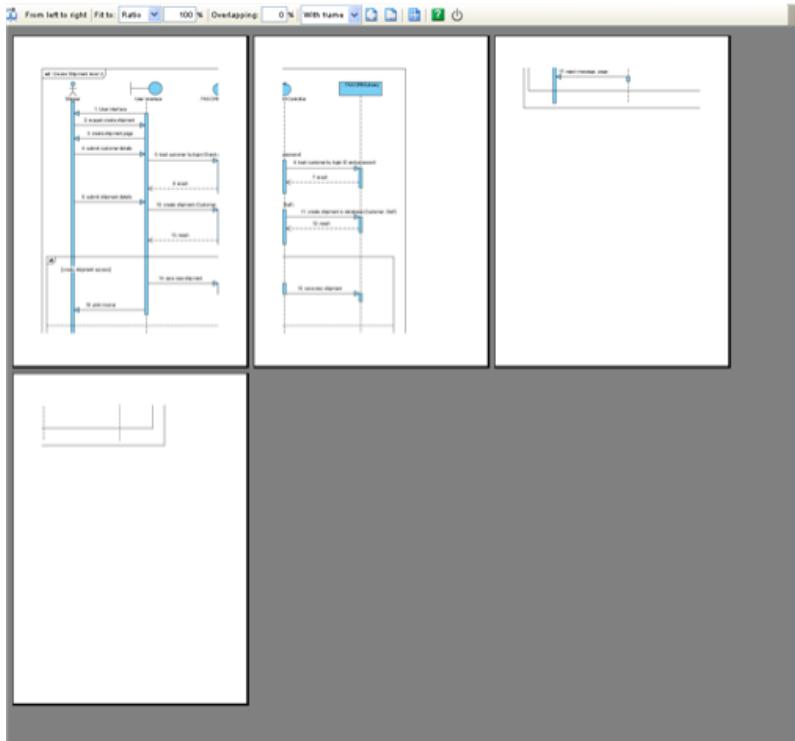
There are two layouts that you can choose for the print preview, the **Paper Base Layout** and the **Diagram Base Layout**.

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.

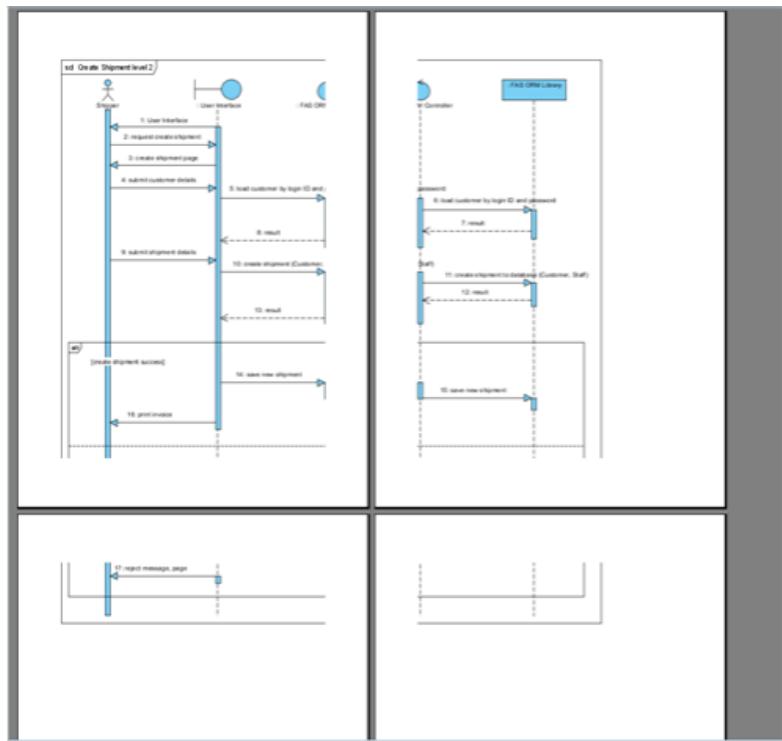
To select a layout of the preview, click the **Paper Base Layout** button or **Diagram Base Layout** button on the toolbar. A pop-up menu where you can choose the layout will appear.

The preview after applying the Paper Base Layout:



Preview in paper base layout

The preview after applying the Diagram Base Layout:



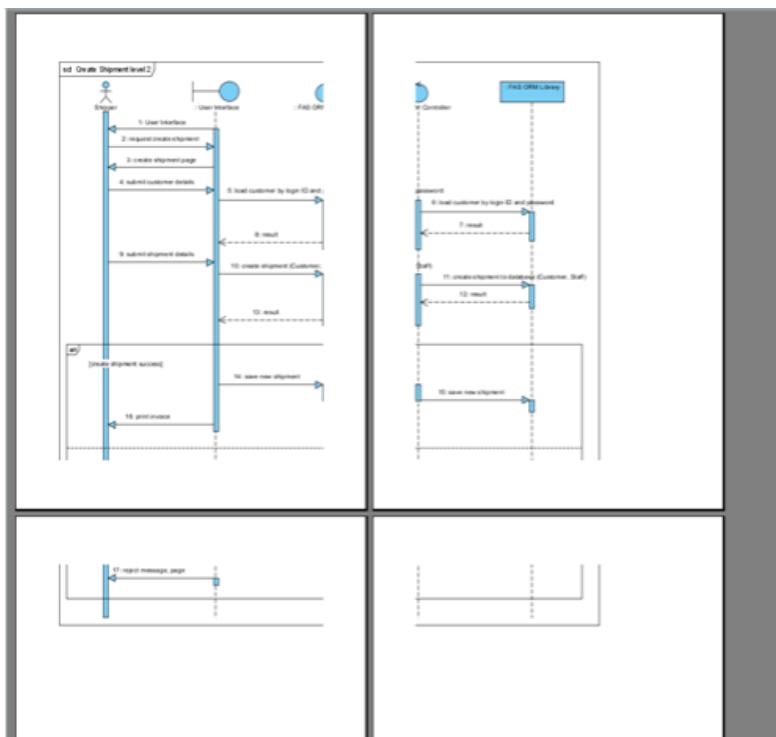
Preview in diagram base layout

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**:



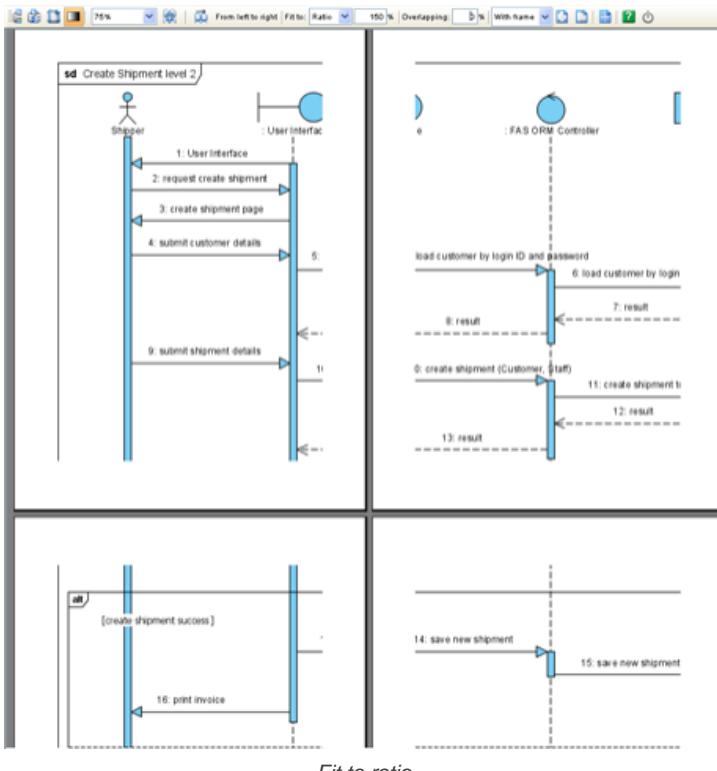
Printout order is left to right

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.



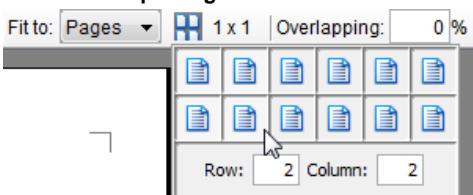
Fit to ratio

Fit to pages

Fit to Pages is used to split the diagram to a desired number of pages when printing.

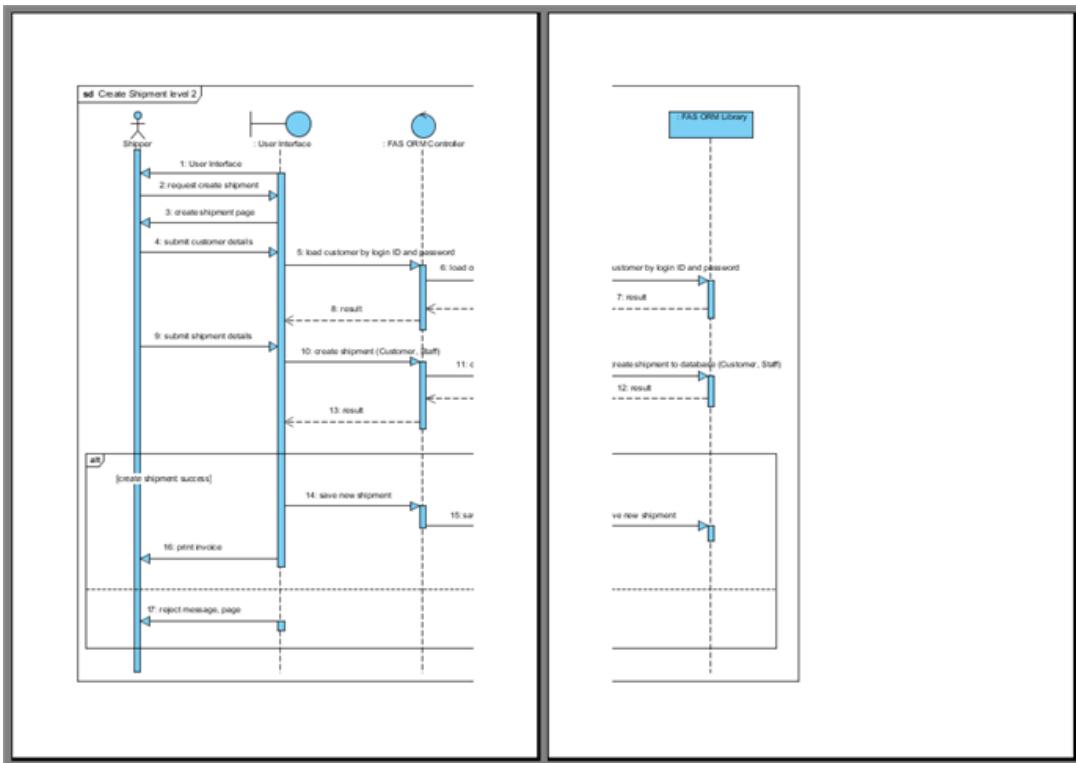
1. Click the **Fit to** drop-down menu and select **pages**.

2. Click the **Multiple Page Mode** button  on the toolbar. The page selector will appear.



Select multiple pages

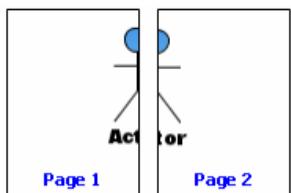
3. Click the row-column combination to select it (note that you can click and drag on the page selector to extend the selection). The diagram will be split into multiple pages by the rows and columns that you have selected.



Fit to page

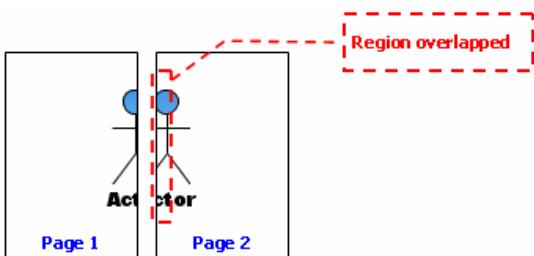
Setting the diagram overlap percentage

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 spans 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.



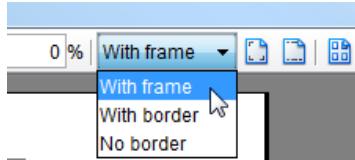
Multiple page with overlap

Printing with frame/Border option

You can print your diagram with a frame or border. There are three options available:

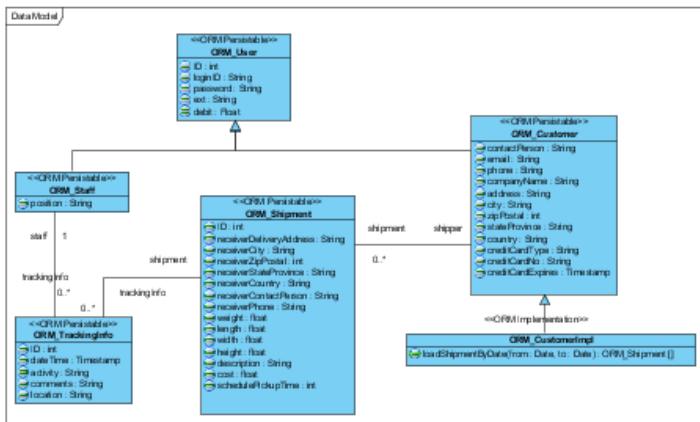
- With frame
 - With border
 - No border

Select **With frame**/ **With border**/ **No border** option from the drop-down menu.



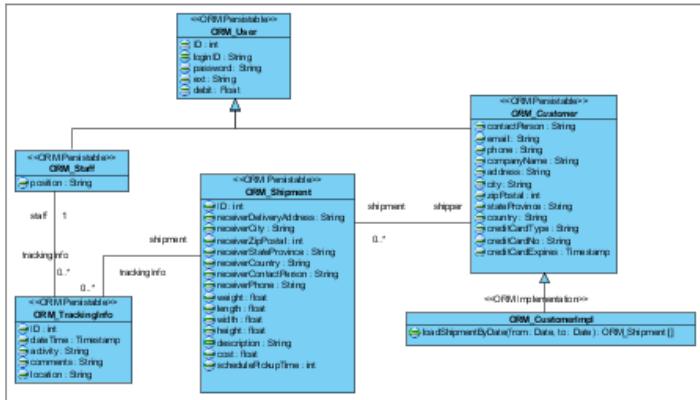
Select an option from drop-down menu

Output of printing with frame:



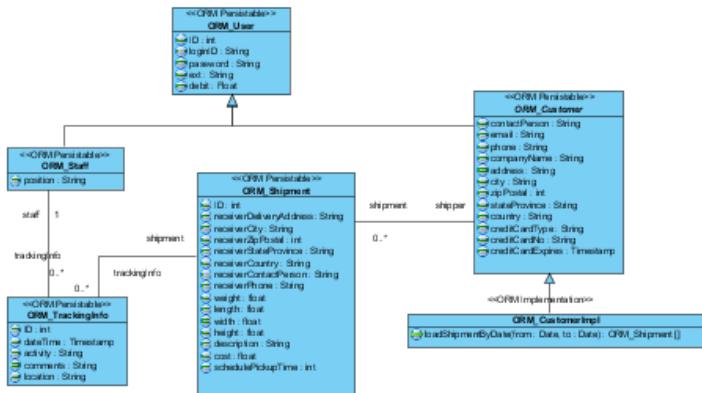
Printing with frame

Output of printing with border:



Printing with border

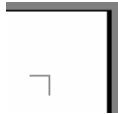
Output of printing with no border:



Printing with no border

Showing/Hiding clip marks on page

Clip marks act as an indication of the boundary of a page.

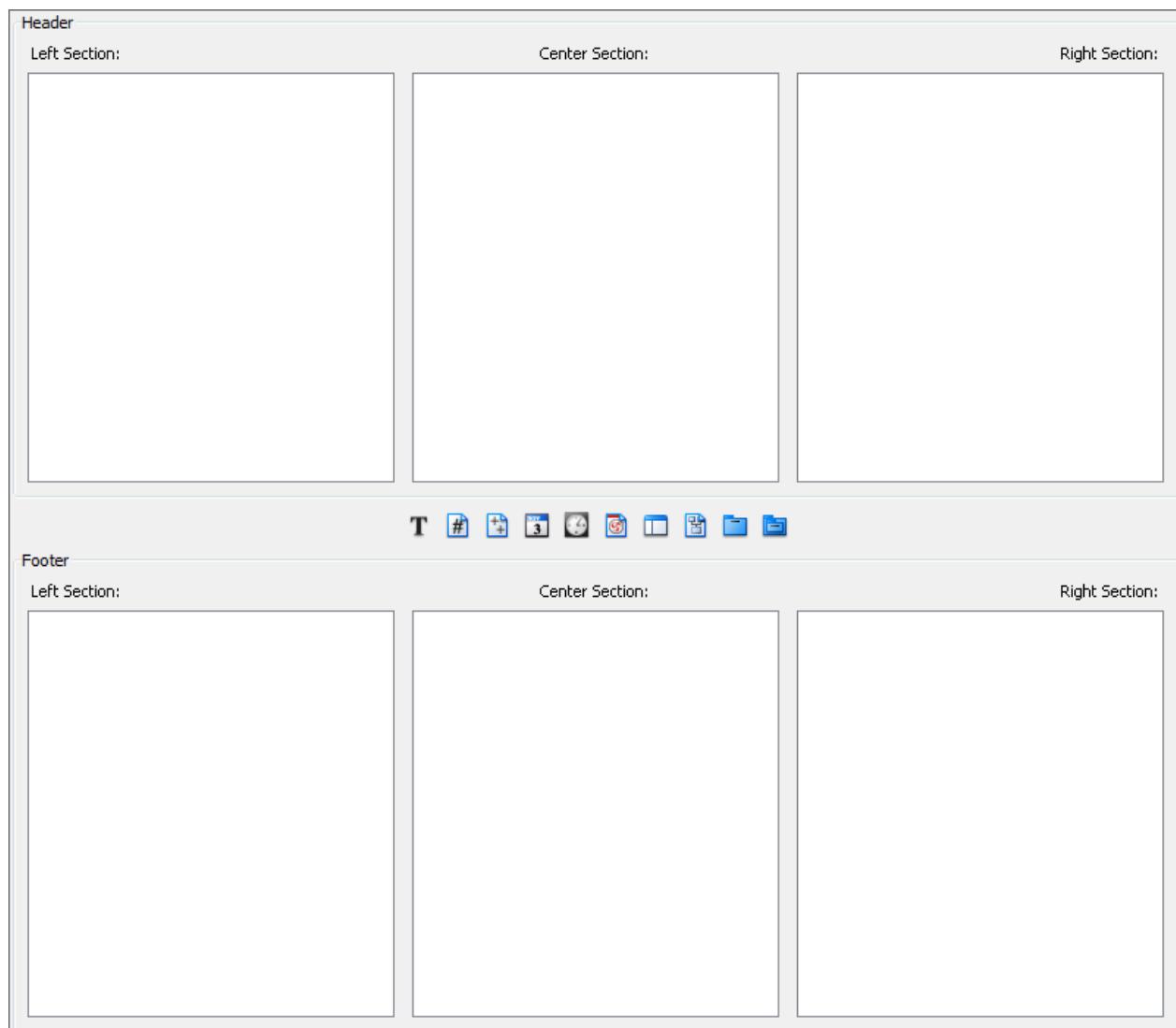


Clip marks

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

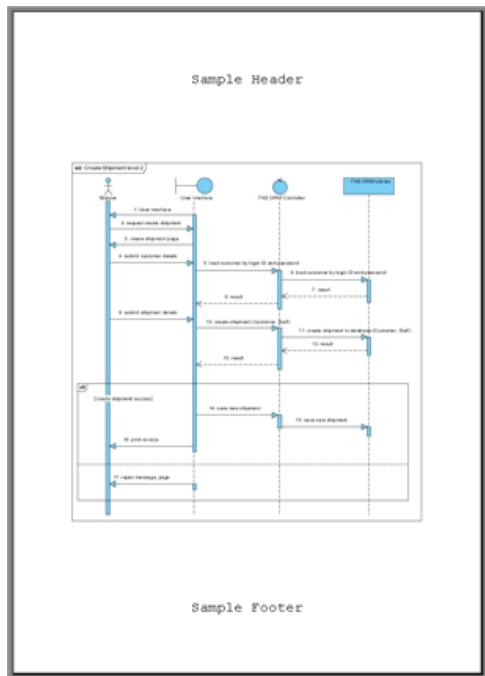
You can edit the header and the footer in the Header panel and the Footer panel respectively. Each of the panel consists of the Left Section, Center Section and the Right Section, which represents the position that the content will be located in the header/footer.

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 VP-UML 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 VP-UML 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:

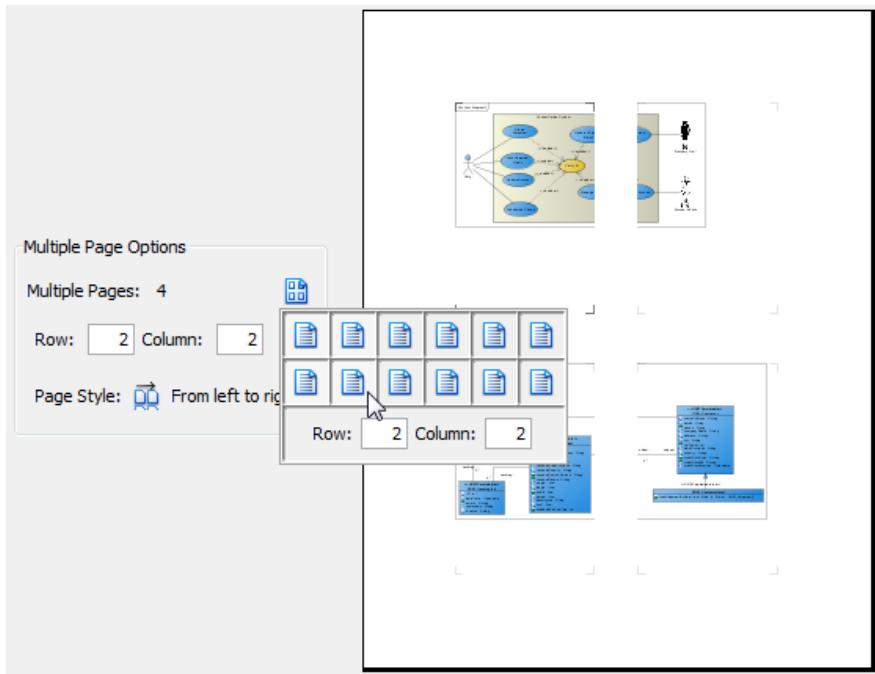


Page with header and footer

The multiple page mode

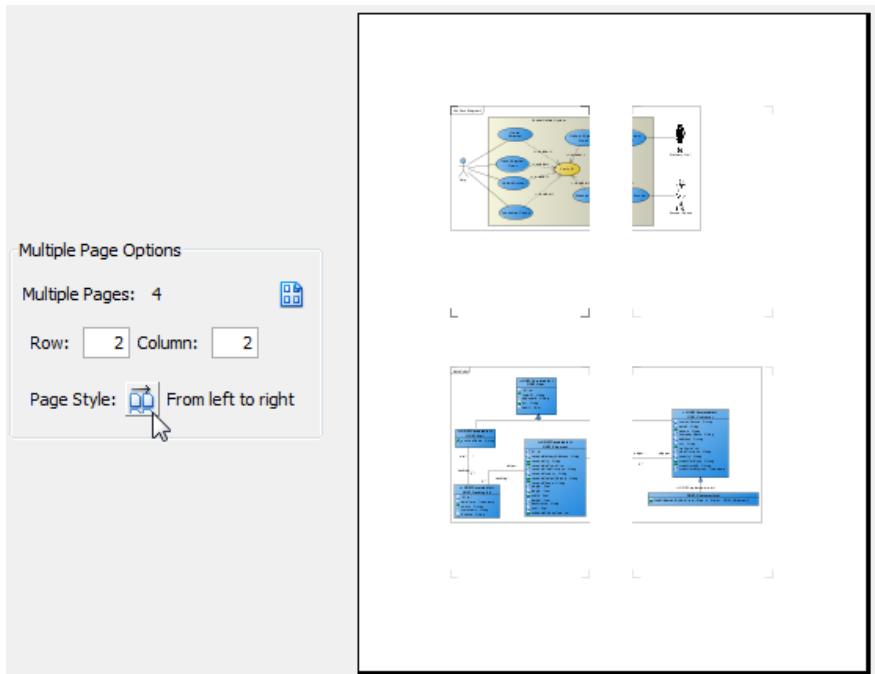
The Multiple Page Mode allows users to configure how the diagrams should be distributed in multiple pages. To switch to the Multiple Page Mode, click the **Multiple Page Mode** button on the toolbar.

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

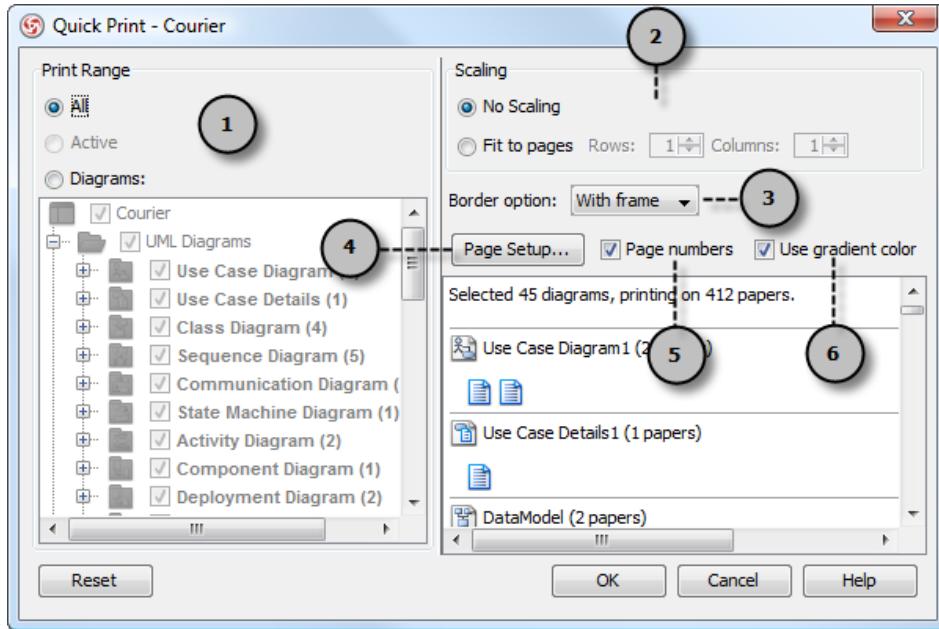
After you have finished configuring the multiple page settings, click the **Close Multiple Page Mode** button to close it.

Printing a Diagram with Quick Print

The Quick Print feature allows you to print diagrams without previewing them, hence speeding up the print job. To quick print, perform one of the following actions:

- Select **File > Quick Print...** from main menu
- Select **File > Print...** from main menu. This displays the Print Preview dialog box. Click  on the toolbar of the Print Preview dialog box.

In both cases, the Quick Print dialog box will show.



Quick print dialog

No.	Name	Description
1	Print Range	Click on either of the below 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 border option of printout.
4	Page Setup...	Page Setup allows you to specify the page size, the orientation as well as the margins of the pages.
5	Page numbers	Select to print diagrams with page number on it.
6	Use gradient color	Select to use gradient color in printout.

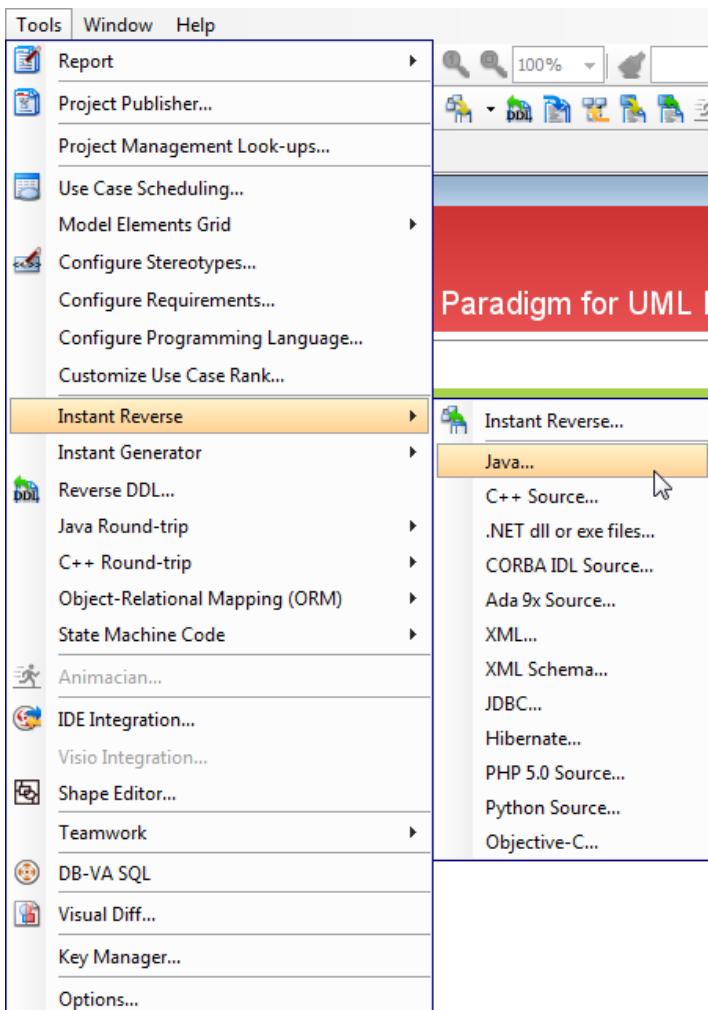
Details of quick print dialog

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.

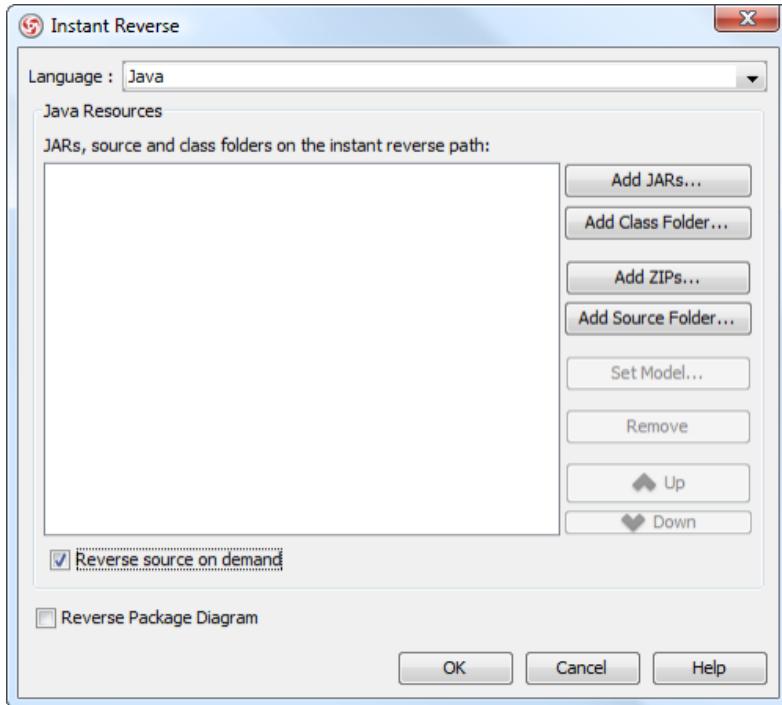
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > Java... from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, add the file or folder path of source by clicking on the appropriate **Add** button at the right hand side of the dialog box. There are four kinds of supported sources: Jar file, class folder, a zip of source or a folder of source files.



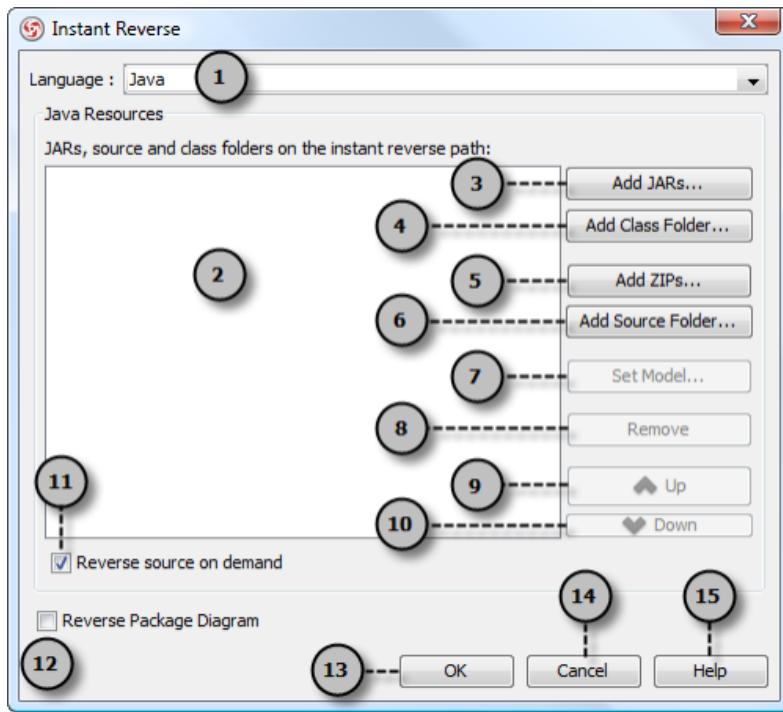
The **Instant Reverse** dialog

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.

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
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 dialog box

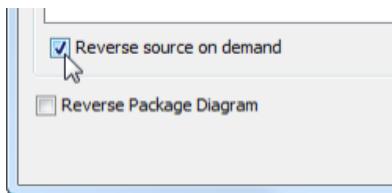
No.	Name	Description
1	Language	The programming language for reversing.
2	Source paths	A list of source paths to be reversed.
3	Add Jars	Add a path of Jar file for reversing.
4	Add class folder	Add a path of Java class folder for reversing.
5	Add zips	Add a path of a zipped source folder for reversing.
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 package diagram	Analyze the sources and form package diagram when reverse. For details about reversing package diagram, refer to the section below.
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 dialog box

On-demand reverse engineering

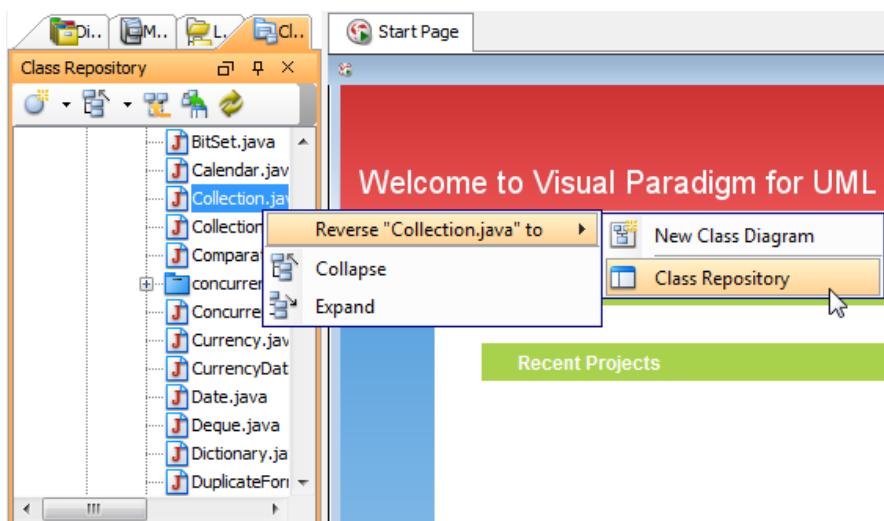
Consider if you have a zip file that contains million of Java source file, like the file src.zip of Java Development Kit (JDK), and now you want to make the class java.util.Collection appear as UML class so that you can extend it when developing your own collection framework. Now, if you try to reverse 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** dialog box.



The option **Reverse source on demand** that appear in reverse dialog box

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.

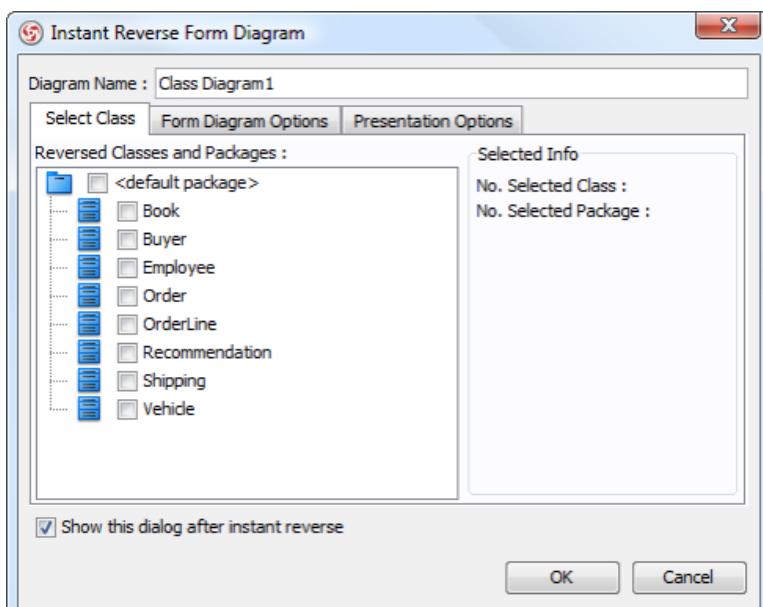


Reversing a java source file from index tree

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 formed.



The **Instant Reverse Form Diagram** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

NOTE: The **Form Diagram** dialog box 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 dialog, 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

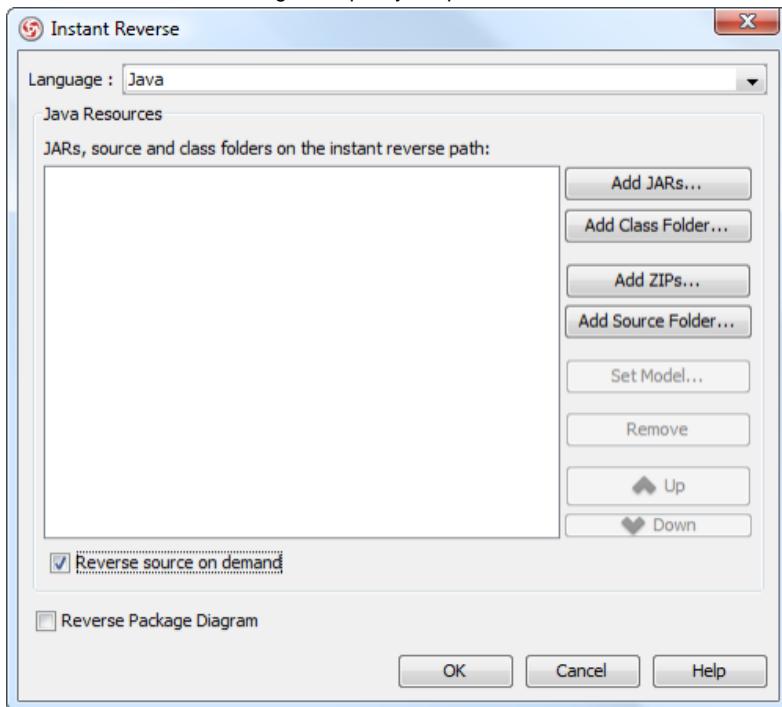
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in 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 > Instant Reverse > Java ...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The instant reverse dialog

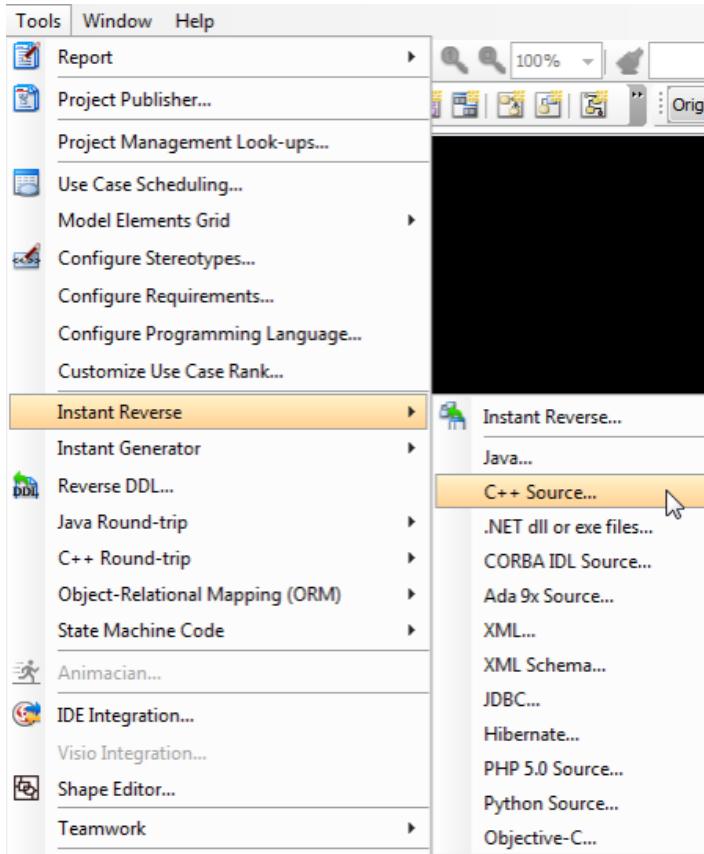
3. Check **Reverse Package Diagram** at the bottom of dialog box.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

Instant reverse C++ header 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 C++ header files.

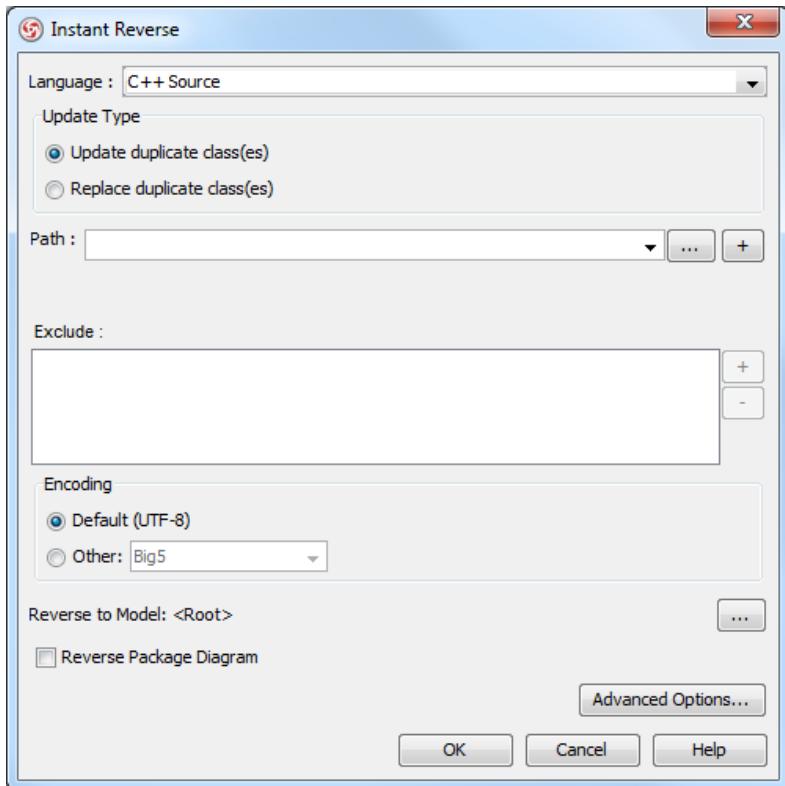
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > C++ header files... from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files. You can add multiple paths by clicking the **+** button.

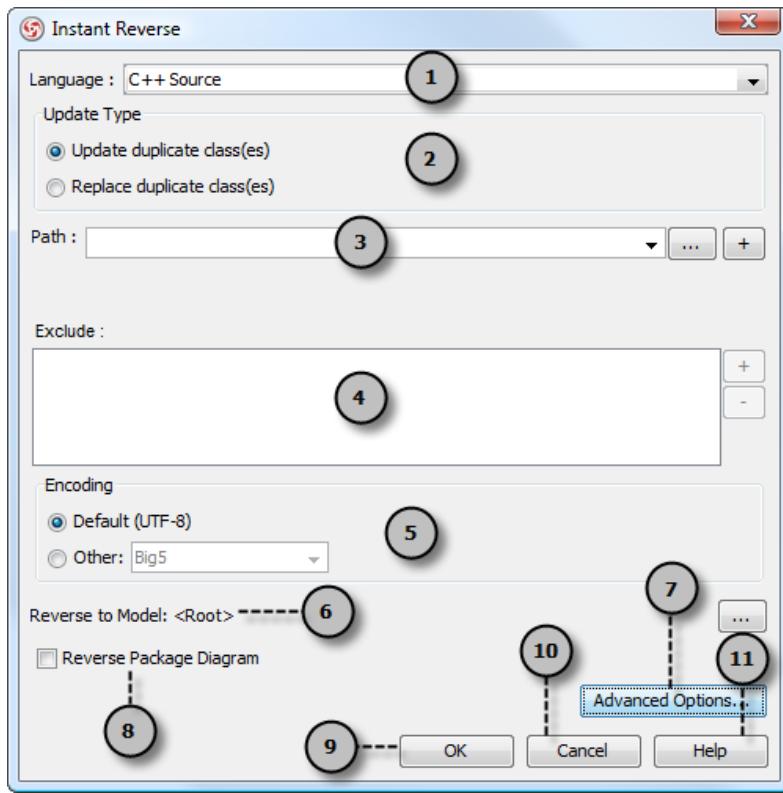


The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.
5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



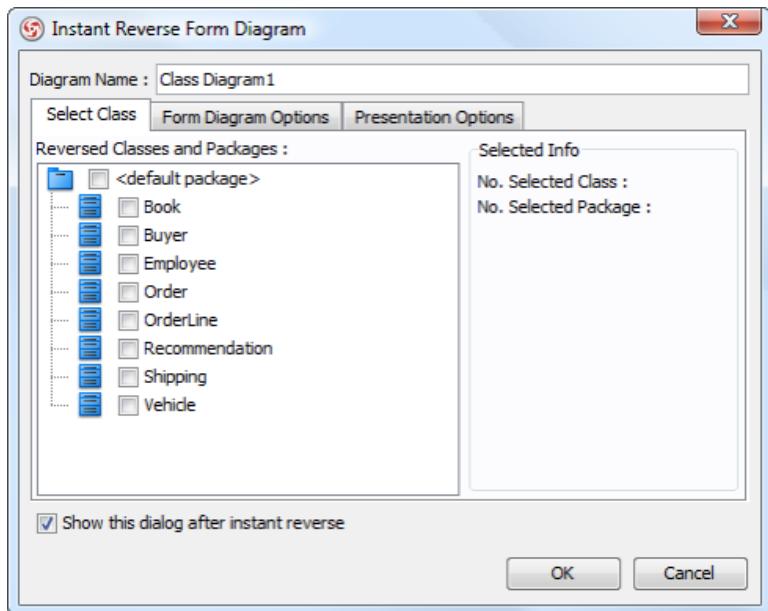
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Exclude	The paths to exclude when reverse. Click + to add a path to exclude, or click - to remove a chosen path.
5	Encoding	The encoding of source file.
6	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
7	Advanced options	Click to configure any detailed options related to reverse in a new dialog box.
8	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

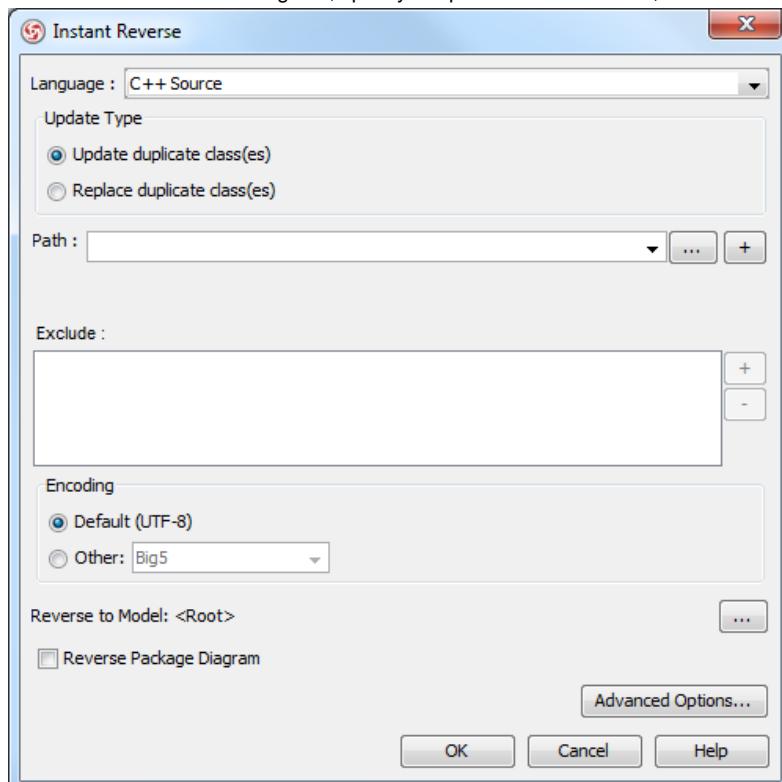
Presentation Options

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 > Instant Reverse > C++ header files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The instant reverse dialog

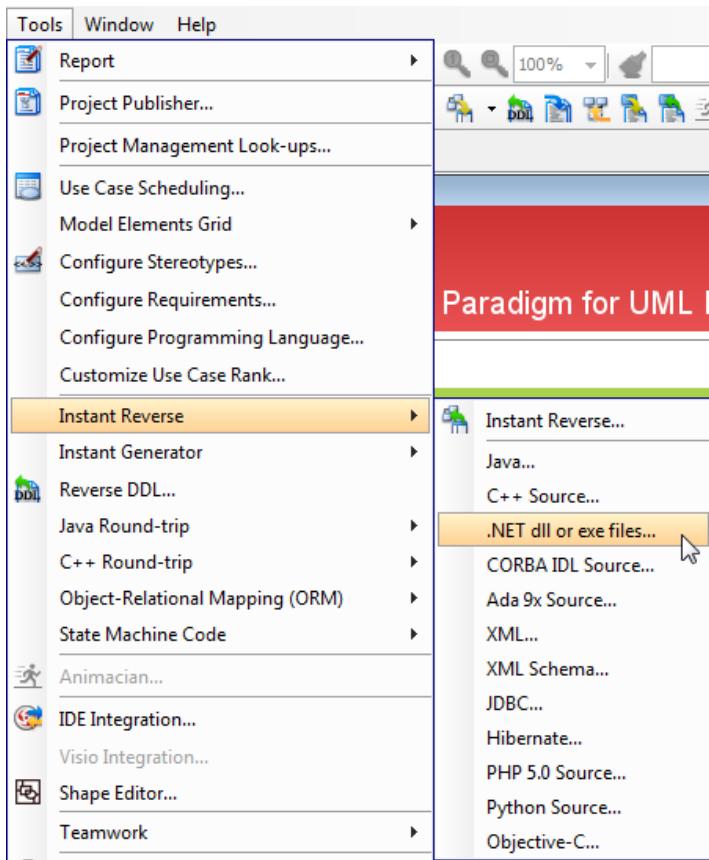
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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.

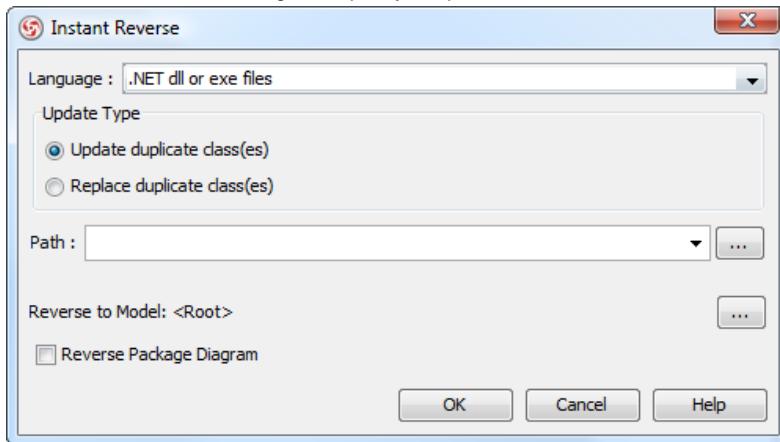
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > .NET dll or exe files... from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



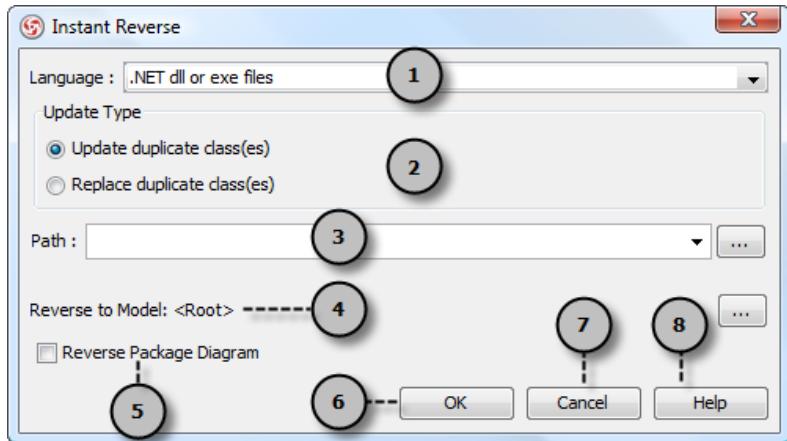
The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



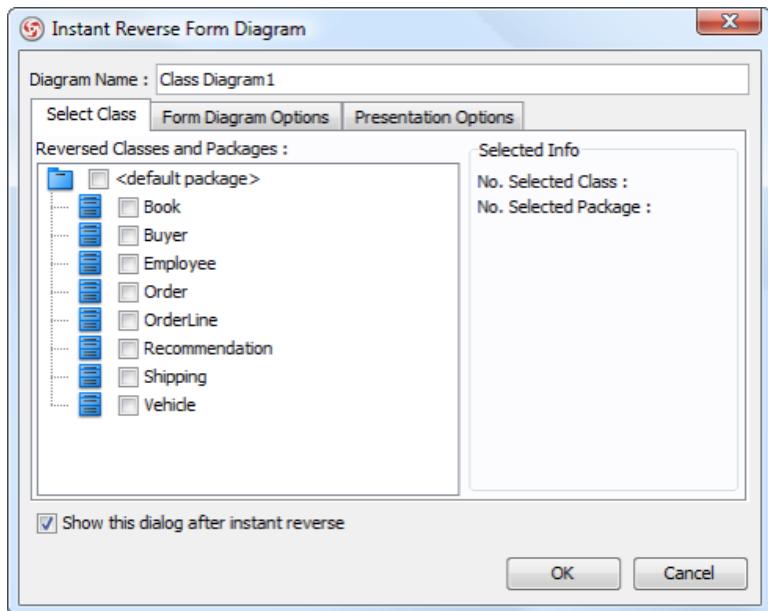
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

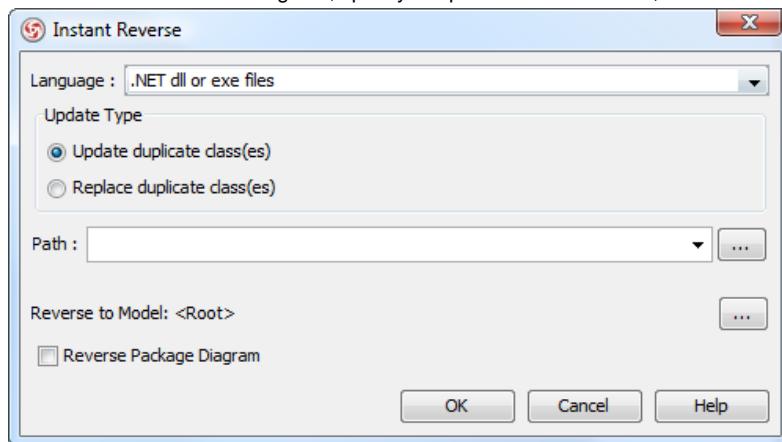
Presentation Options

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The instant reverse dialog

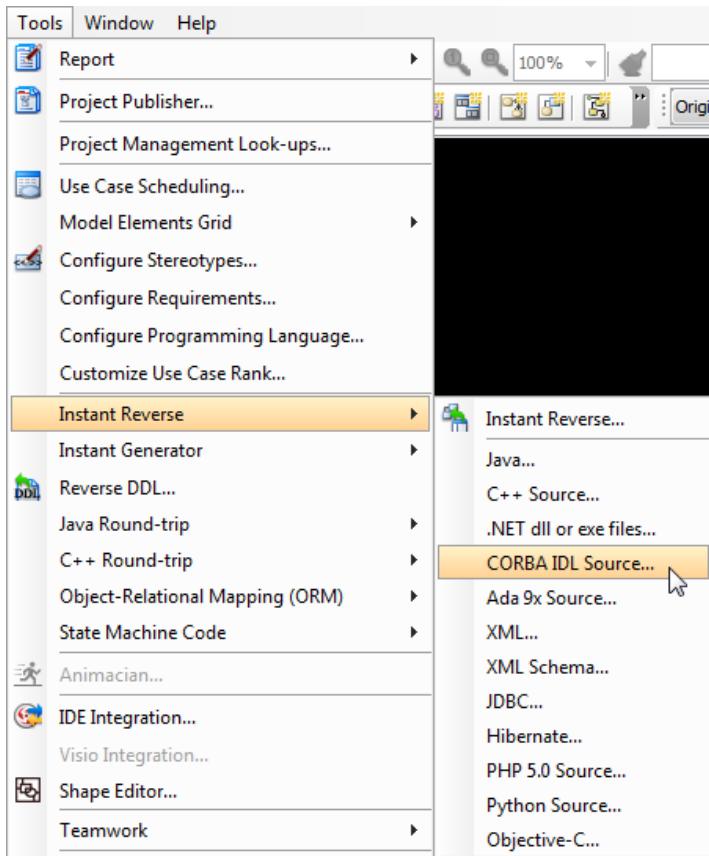
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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.

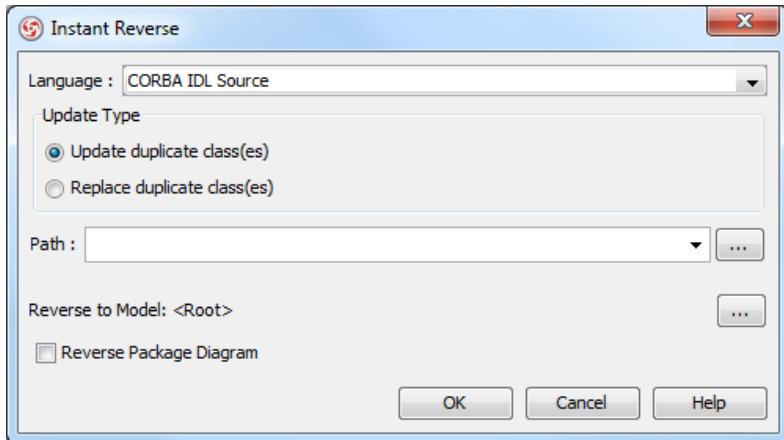
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > CORBA IDL Source files... from the main menu.



To perform instant reverse

2. In the Instant Reverse dialog box, specify the path of the source file, or the folder that contain those files.



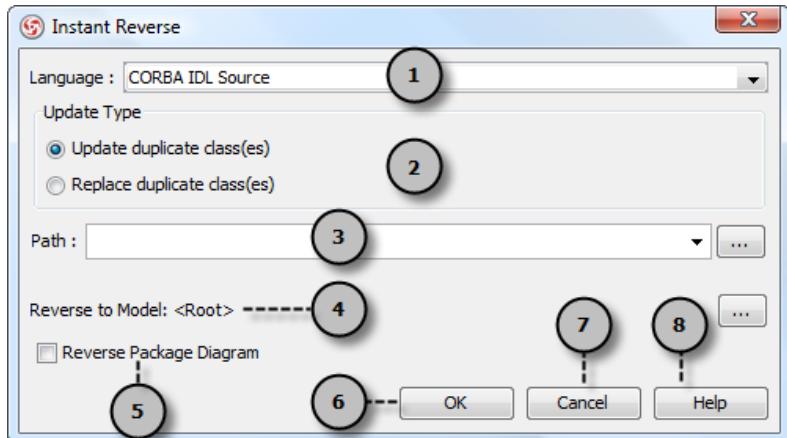
The Instant Reverse dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



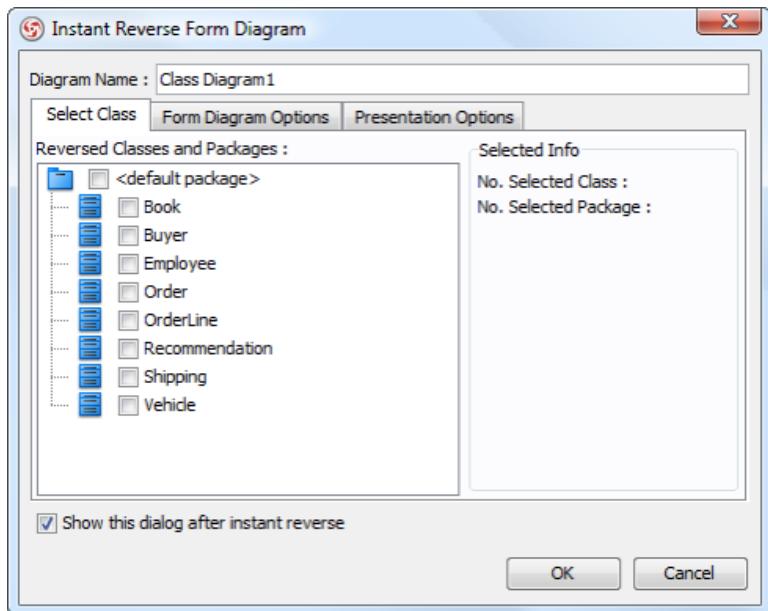
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

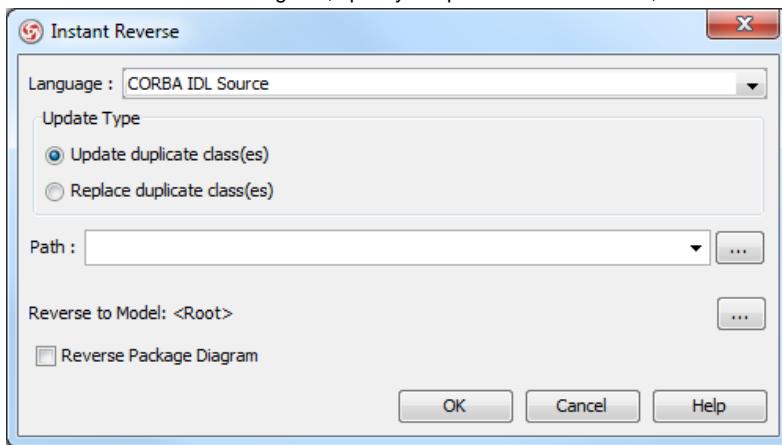
Presentation Options

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The *instant reverse* dialog

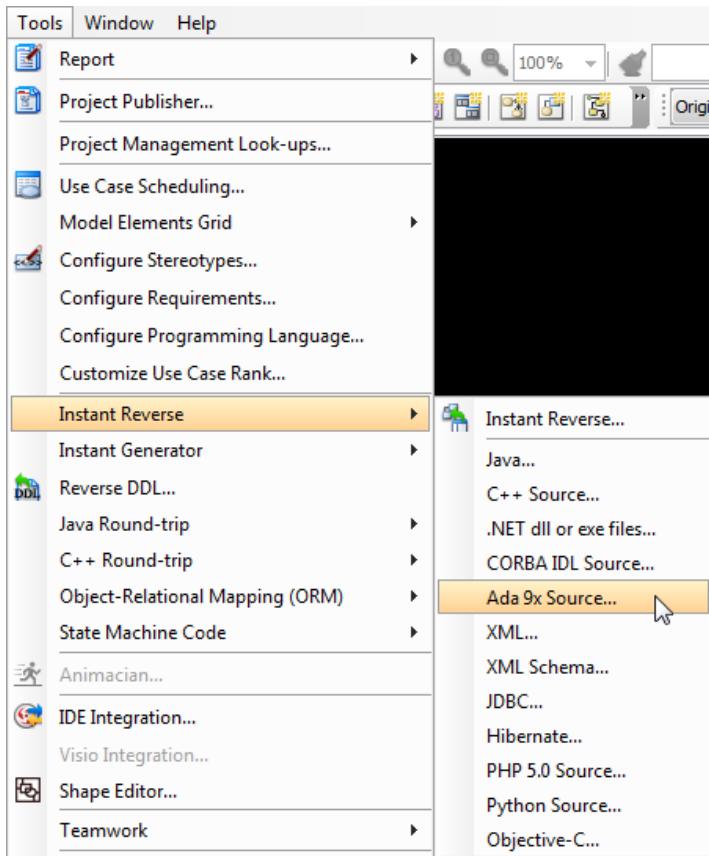
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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 extending determining the generic one. In this chapter, we will go through the instant reverse of Ada 9x Source files.

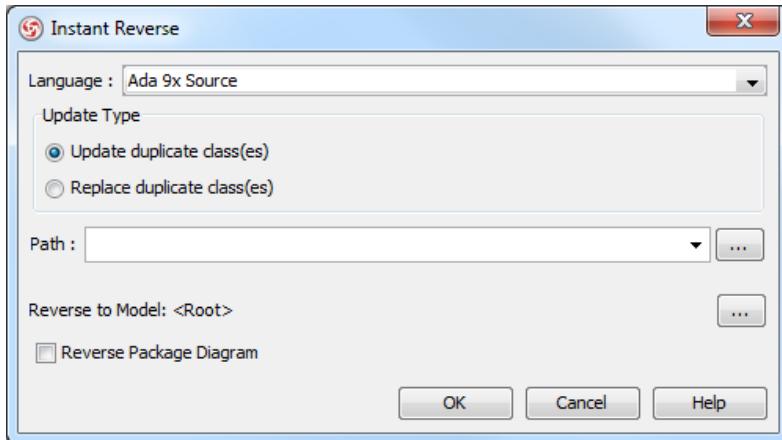
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > Ada 9x Source files... from the main menu.



To perform instant reverse

2. In the Instant Reverse dialog box, specify the path of the source file, or the folder that contain those files.



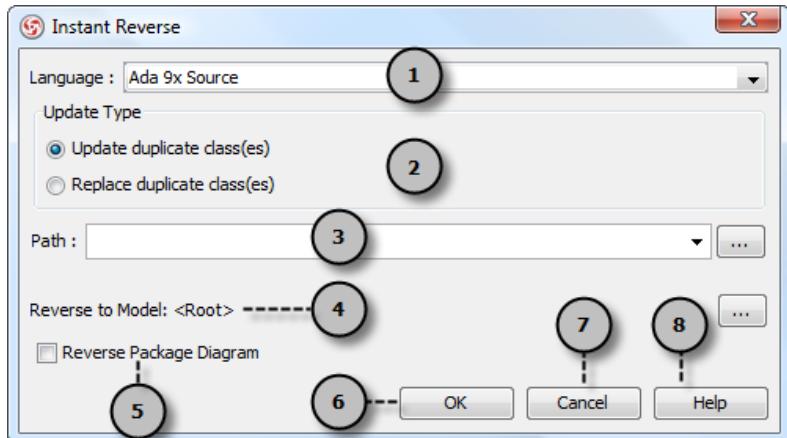
The Instant Reverse dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



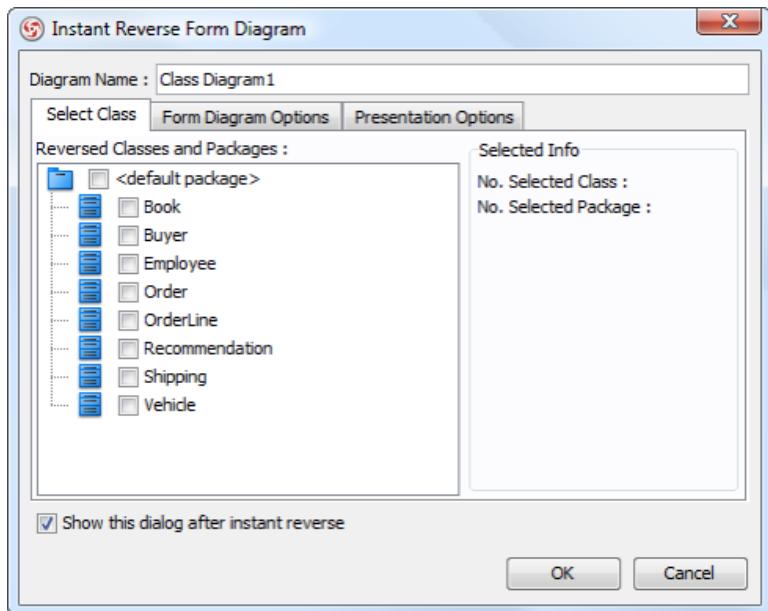
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

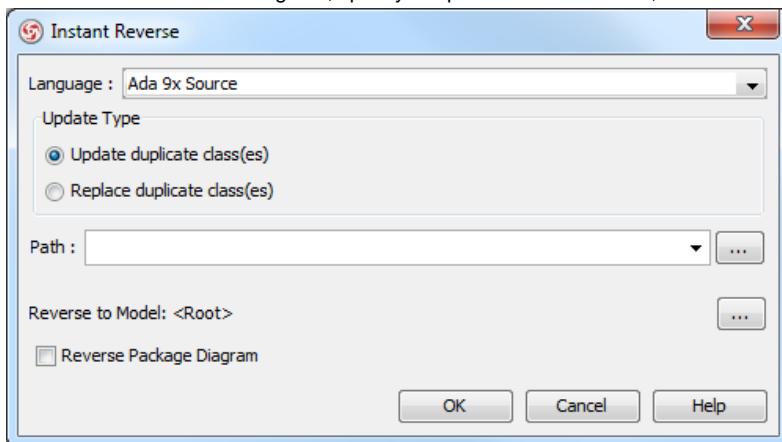
Presentation Options

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The *instant reverse* dialog

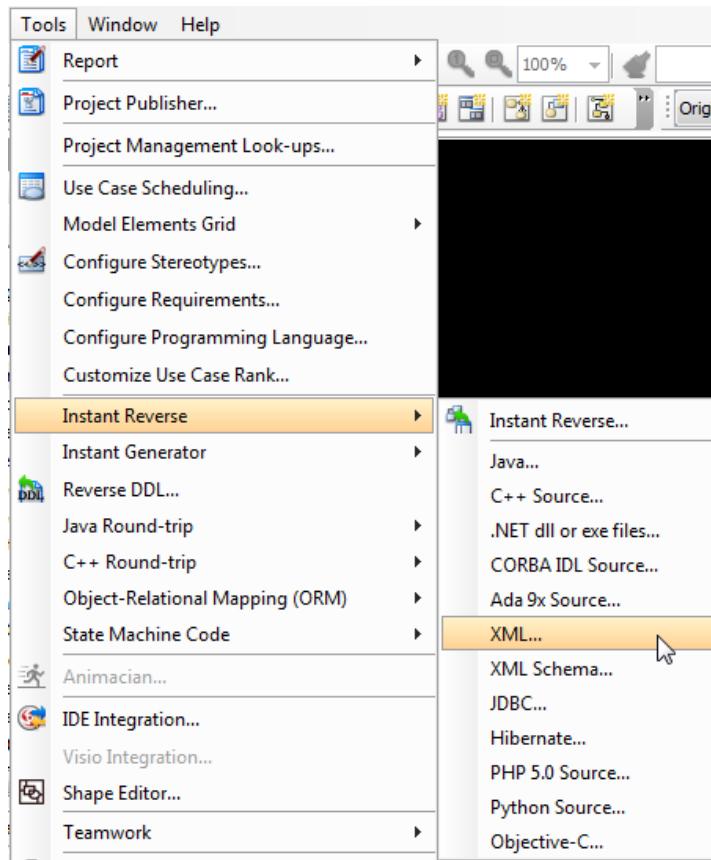
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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.

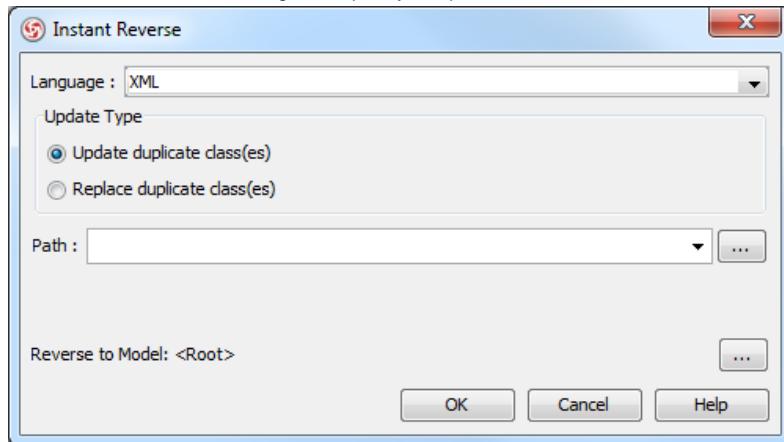
Reverse engineering UML classes from source files

1. Select **Tools > Instant Reverse > XML...** from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



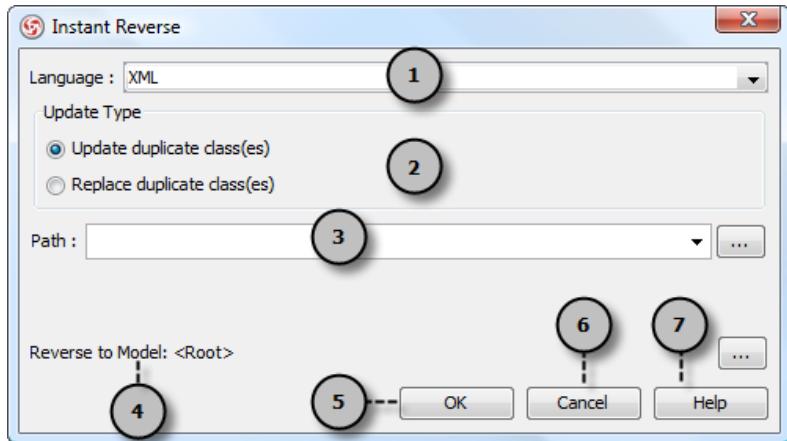
The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



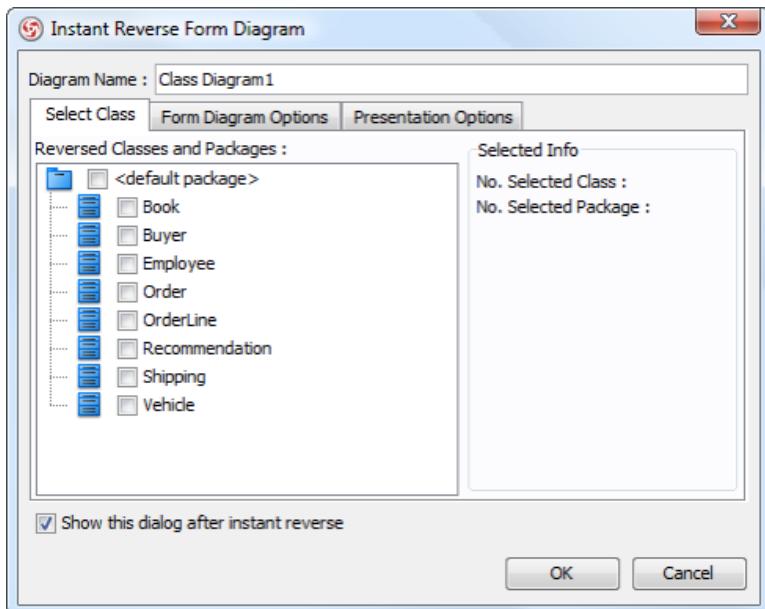
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

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.

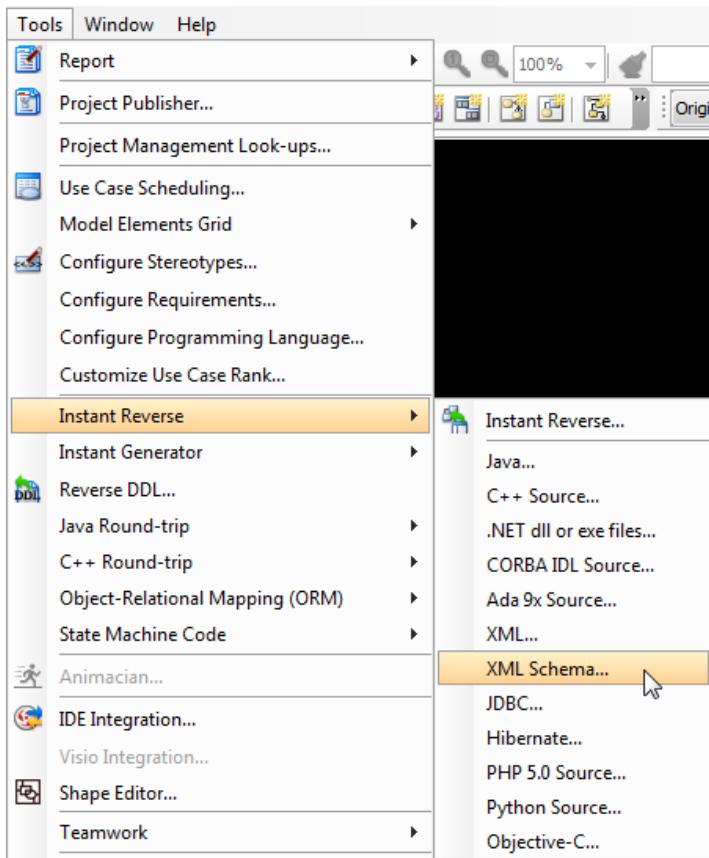
Description of presentation options

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.

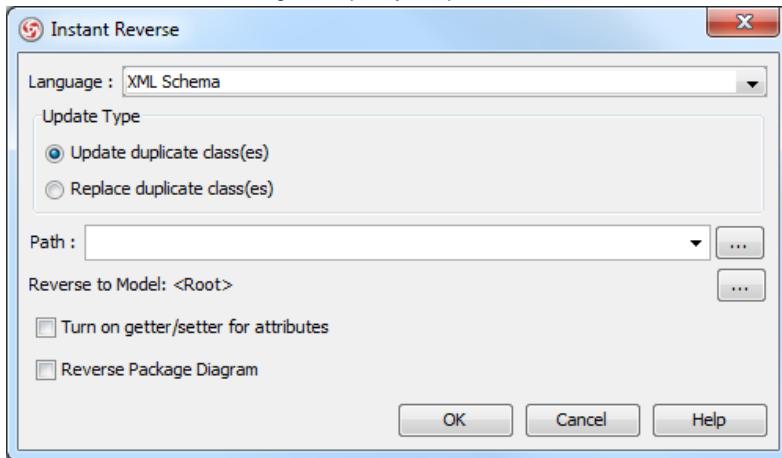
Reverse engineering UML classes from source files

1. Select **Tools > Instant Reverse > XML Schema...** from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



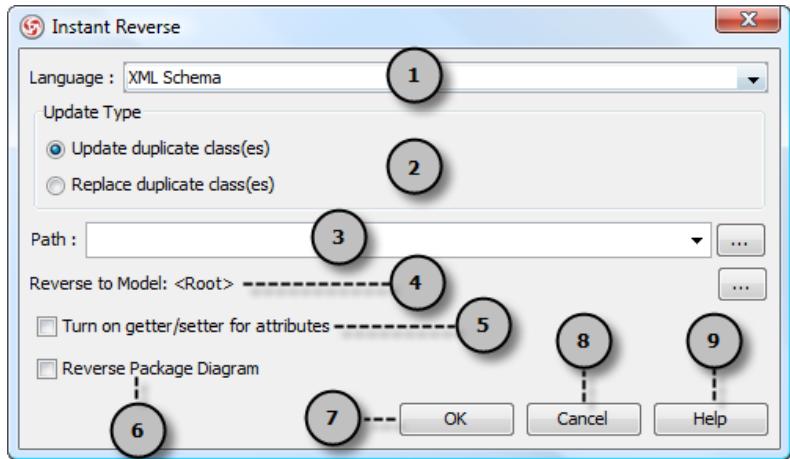
The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



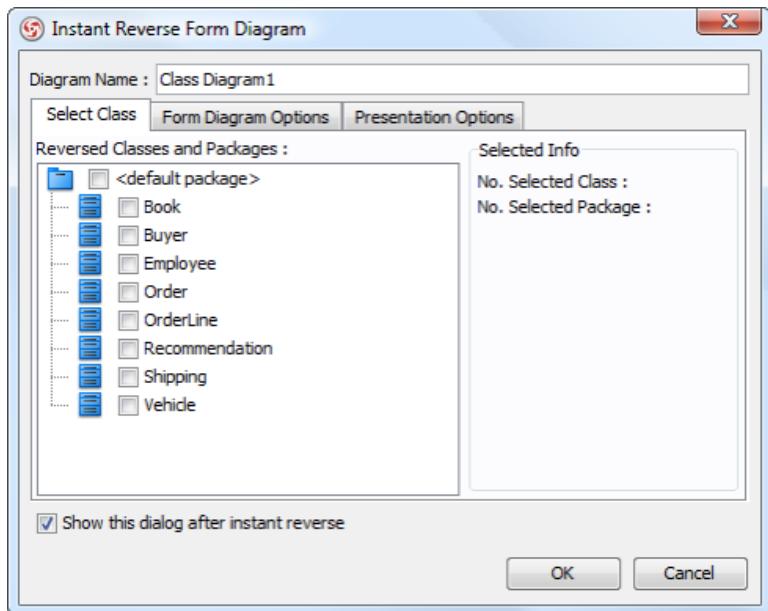
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Turn on getter/setter for attributes	Set all attributes' getter and setter options to be true.
6	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

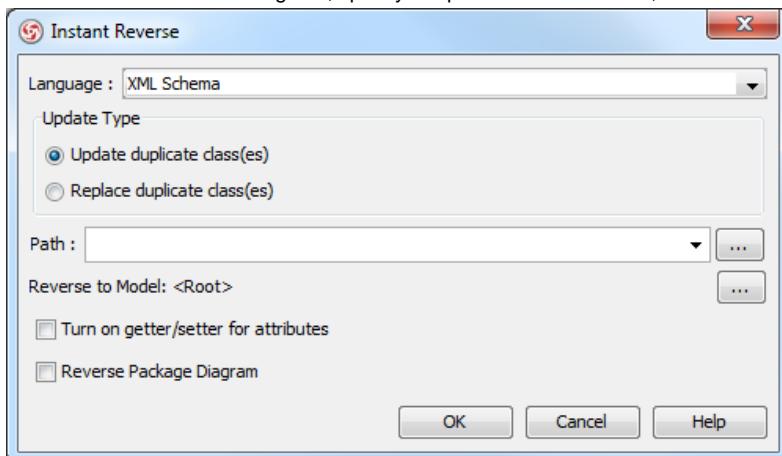
Presentation Options

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The *instant reverse* dialog

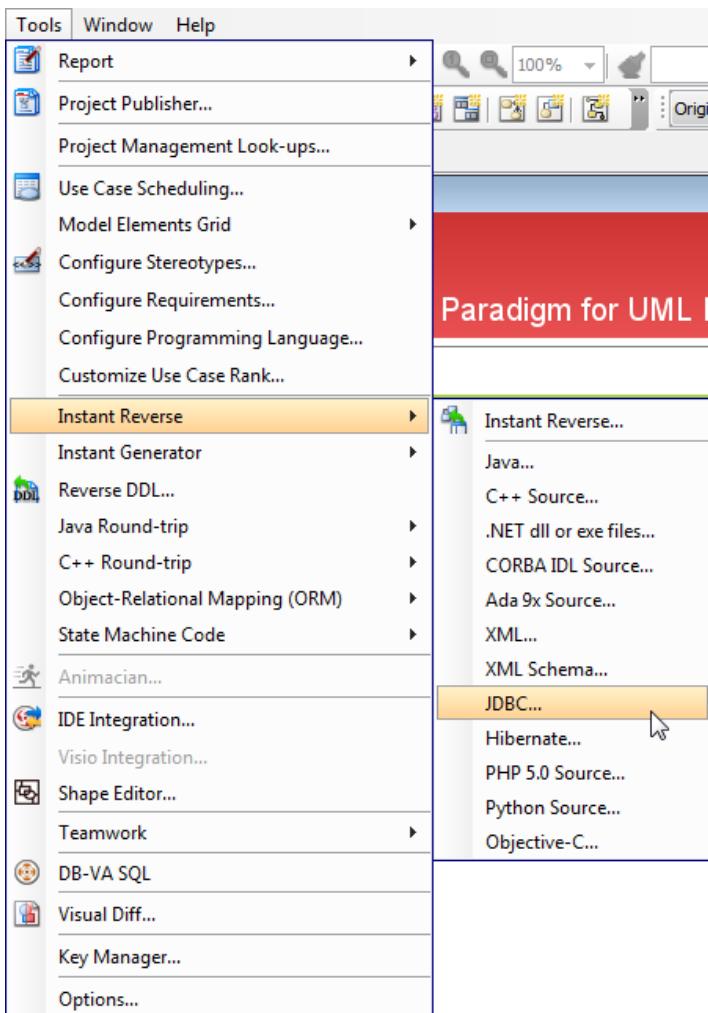
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

Instant reverse database through JDBC

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 database through JDBC.

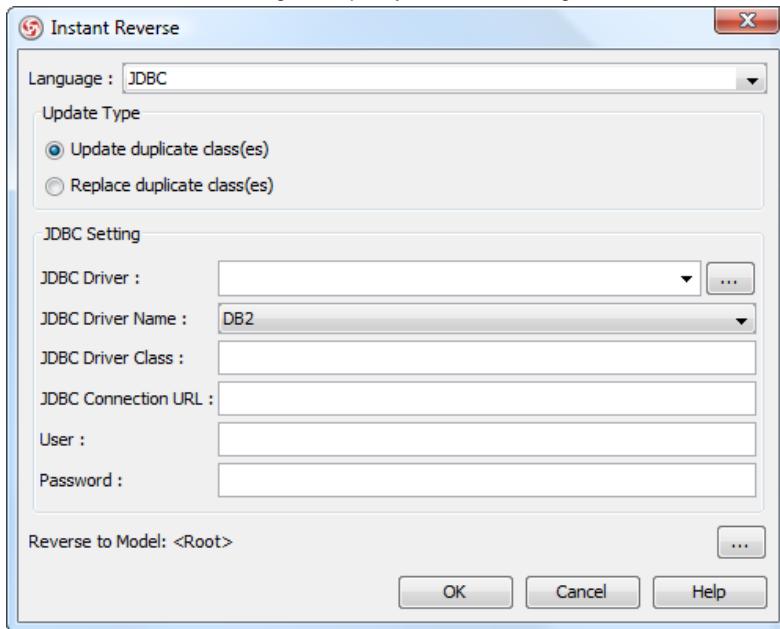
Reverse engineering UML classes from database

1. Select Tools > Instant Reverse > JDBC... from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the JDBC settings.

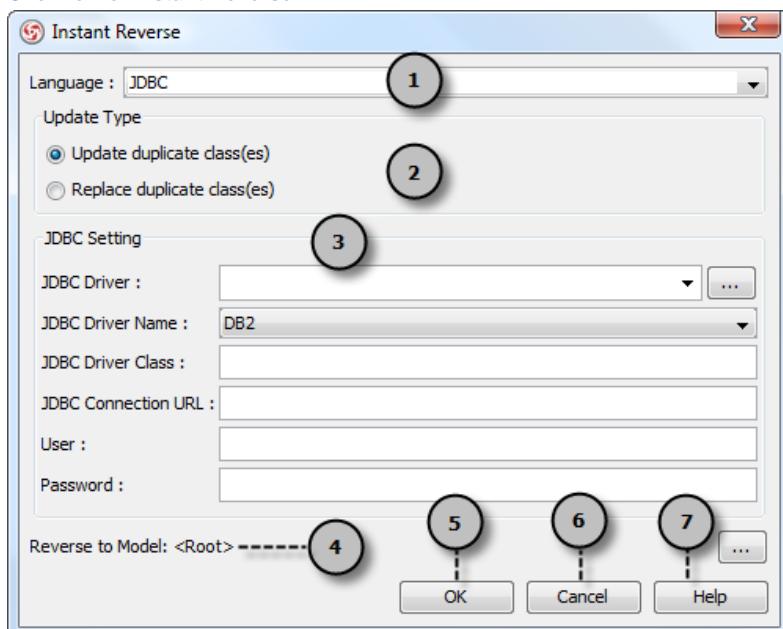


The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.
5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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 dialog box

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 JDBC settings The JDBC settings required to connect to database 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.

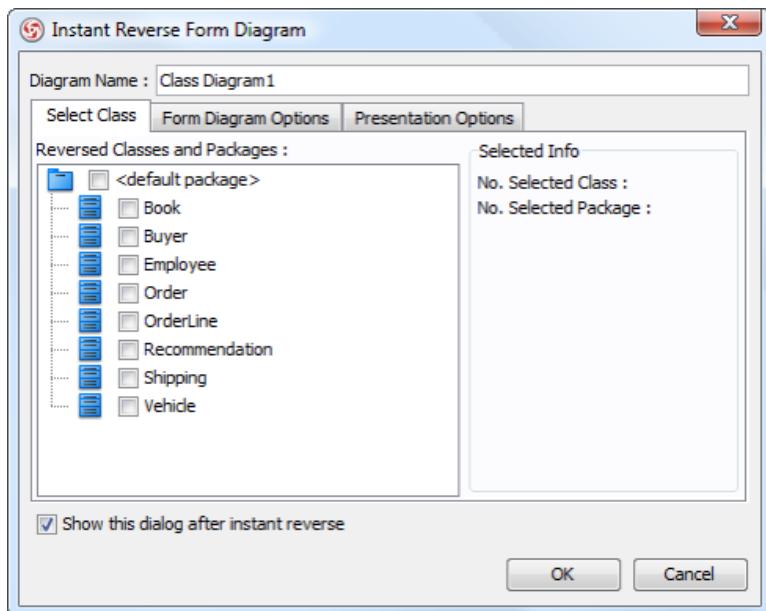
6 Cancel Click to close instant reverse.

7 Help Click to read the Help contents.

Overview of instant reverse dialog box

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* dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

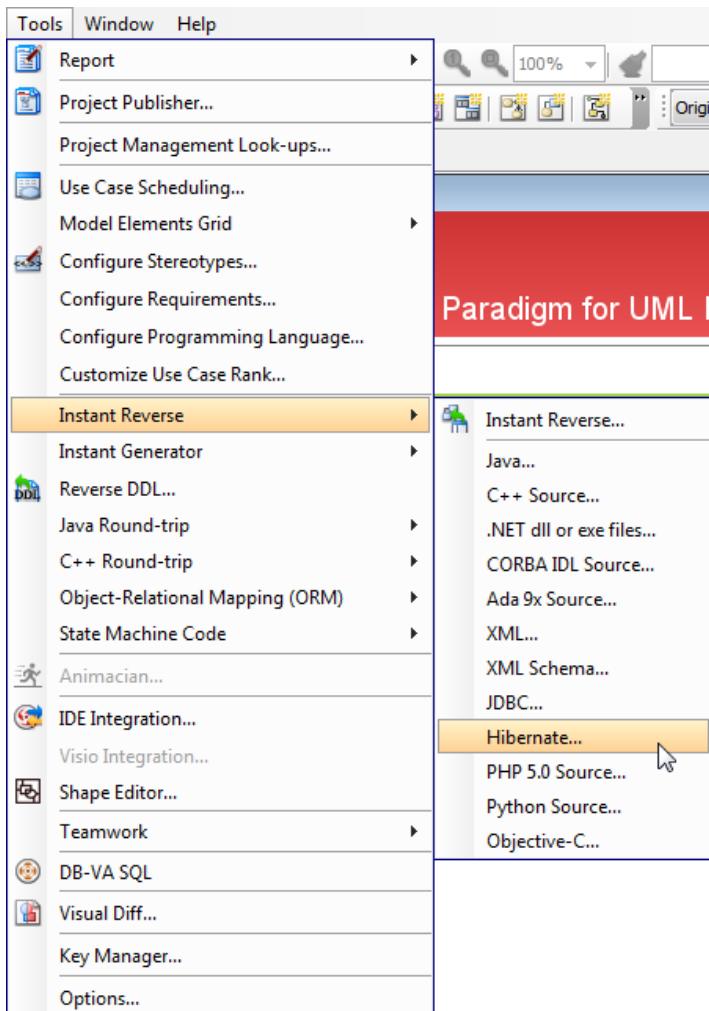
Description of presentation options

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.

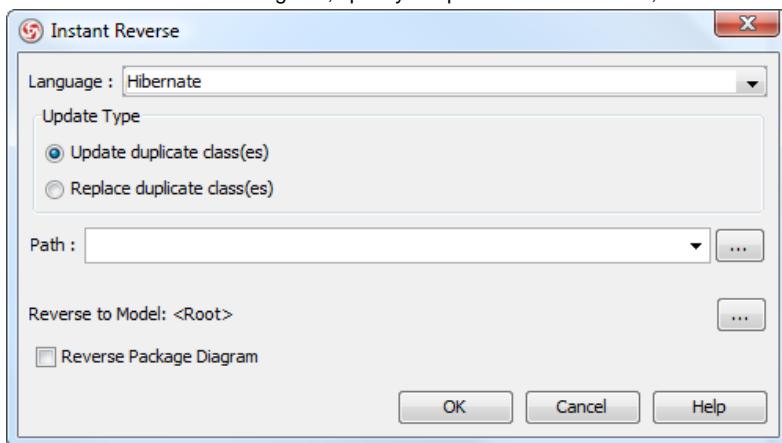
Reverse engineering UML classes from source files

1. Select **Tools > Instant Reverse > Hibernate...** from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.

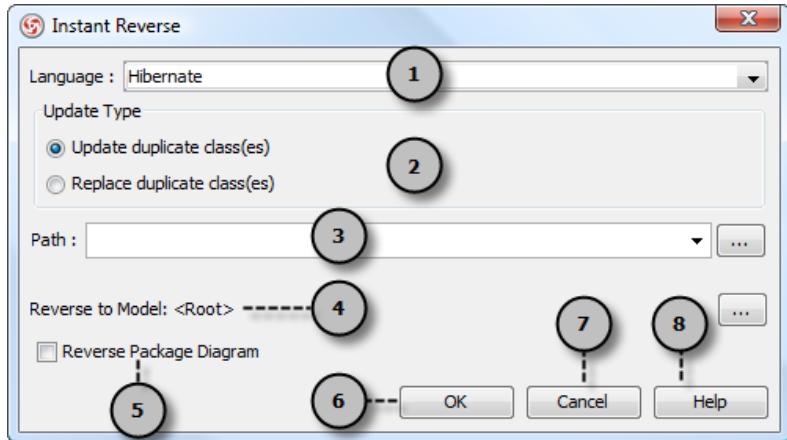


The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.
5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



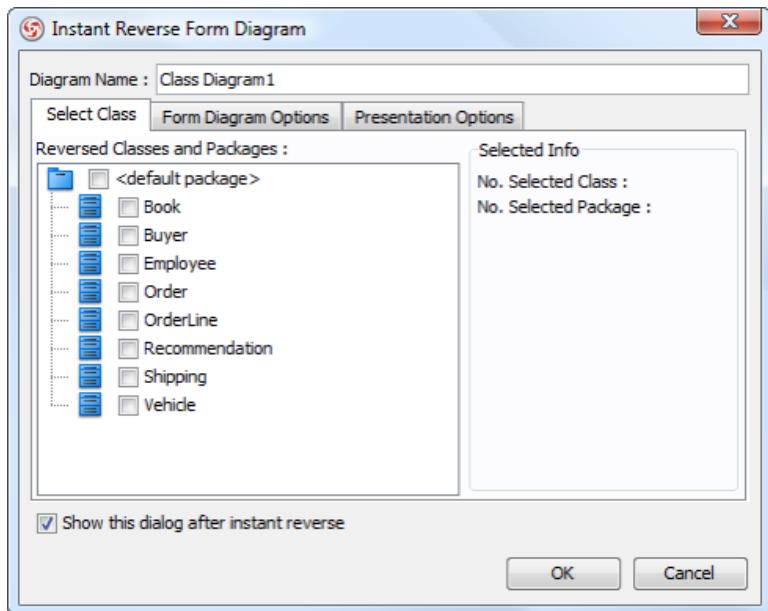
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

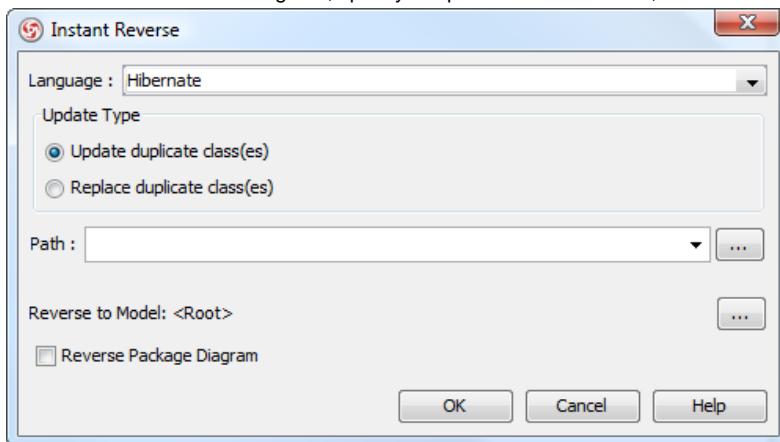
Presentation Options

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 > Instant Reverse > Hibernate...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The instant reverse dialog

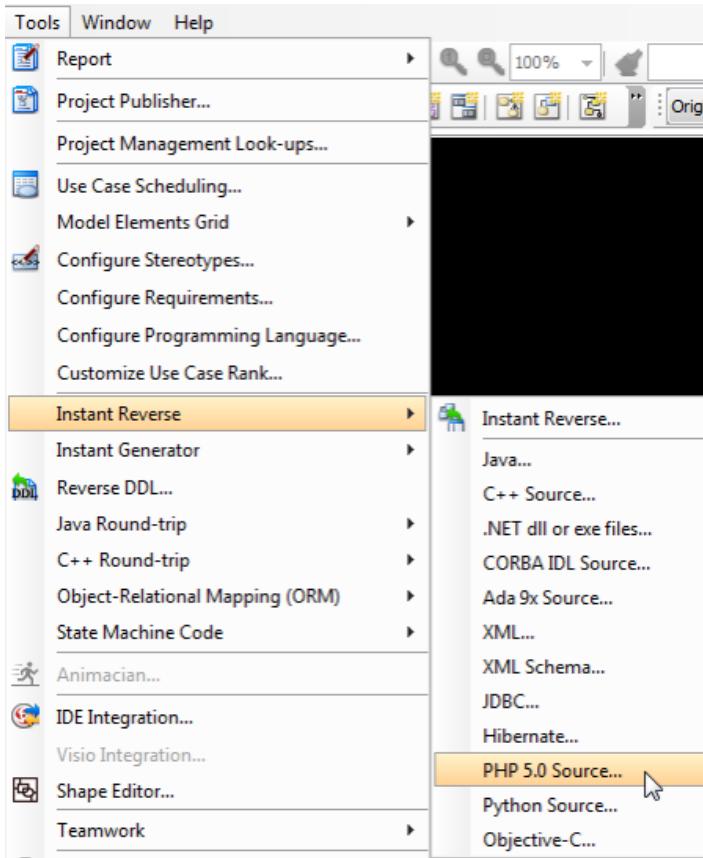
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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 Determiningeneric one. In this chapter, we will go through the instant reverse of PHP 5.0 Source files.

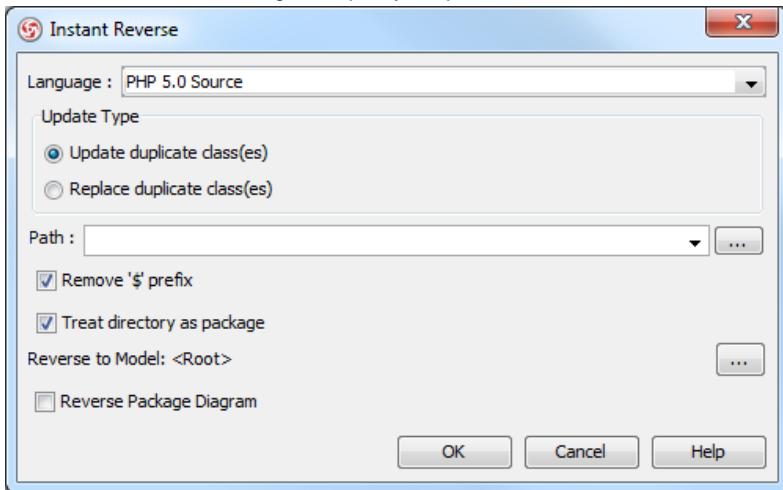
Reverse engineering UML classes from source files

1. Select Tools > Instant Reverse > PHP 5.0 Source files... from the main menu.



To perform instant reverse

2. In the Instant Reverse dialog box, specify the path of the source file, or the folder that contain those files.



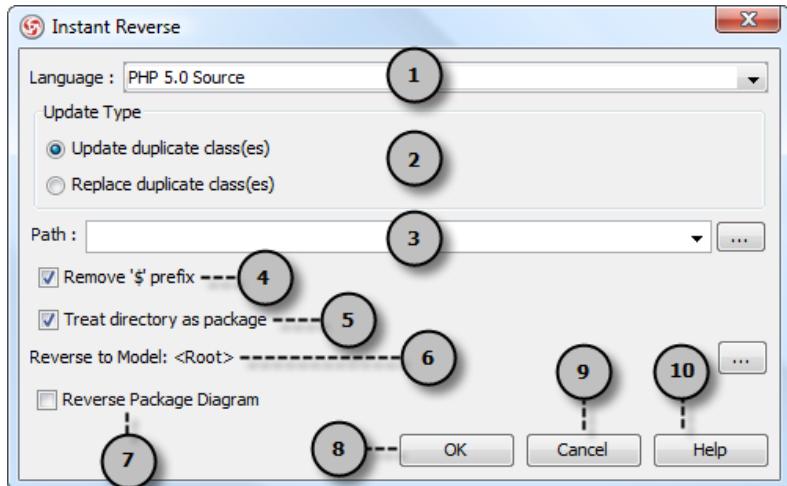
The Instant Reverse dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



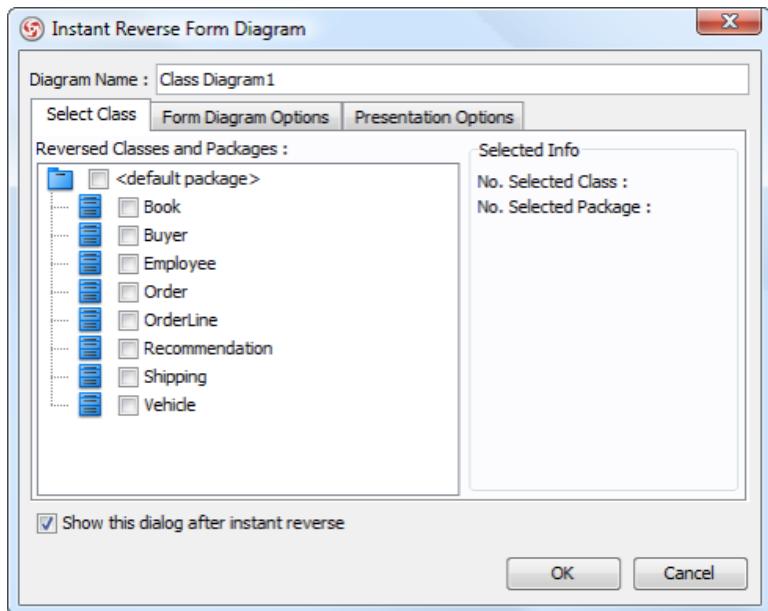
The instant reverse dialog box

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. You can click + to add multiple paths.
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 <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
7	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

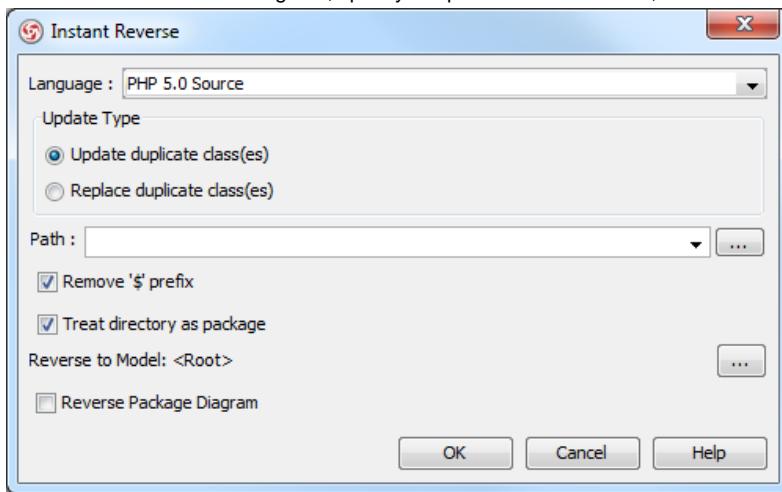
Presentation Options

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The *instant reverse* dialog

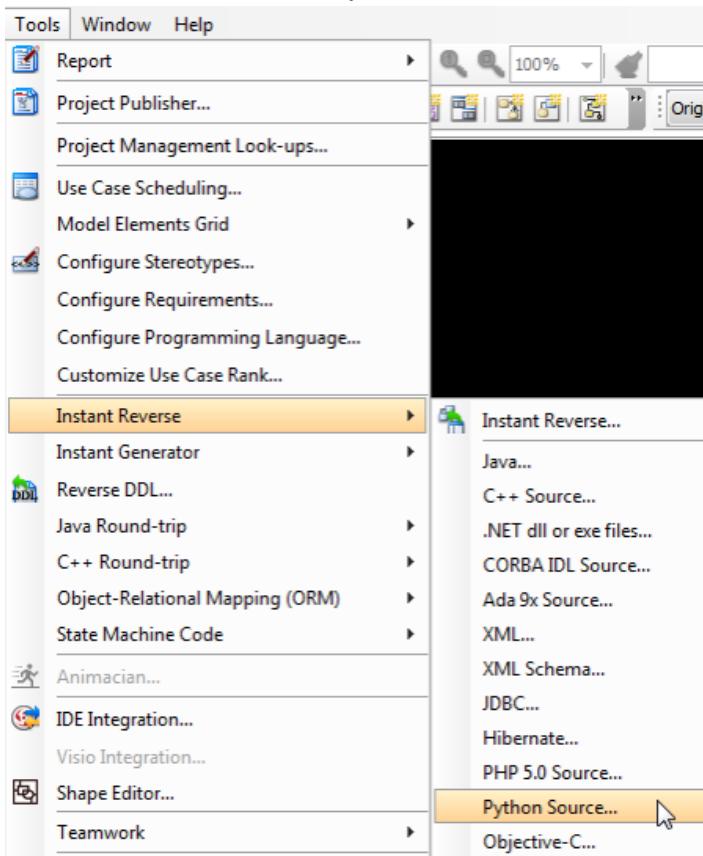
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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.

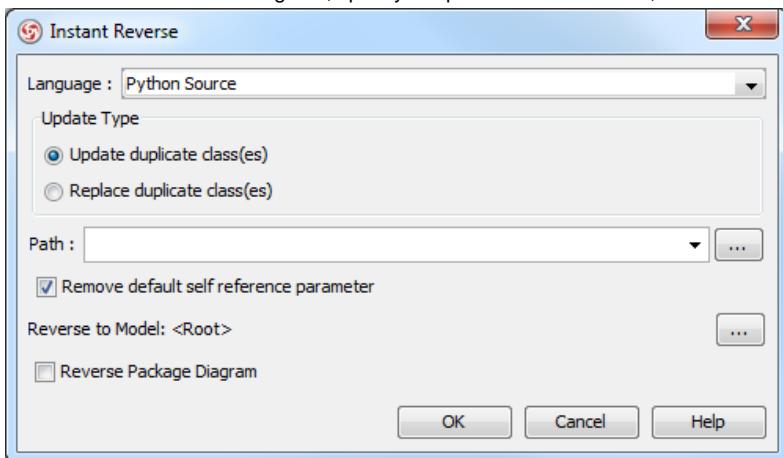
Reverse engineering UML classes from source files

1. Select **Tools > Instant Reverse > Python...** from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



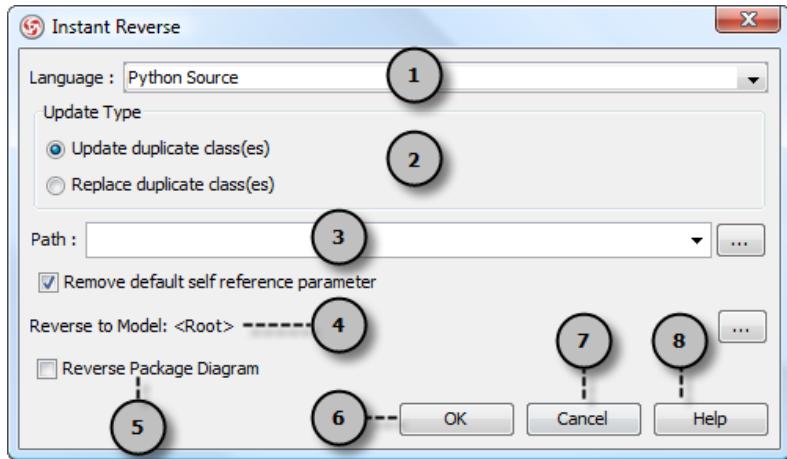
The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



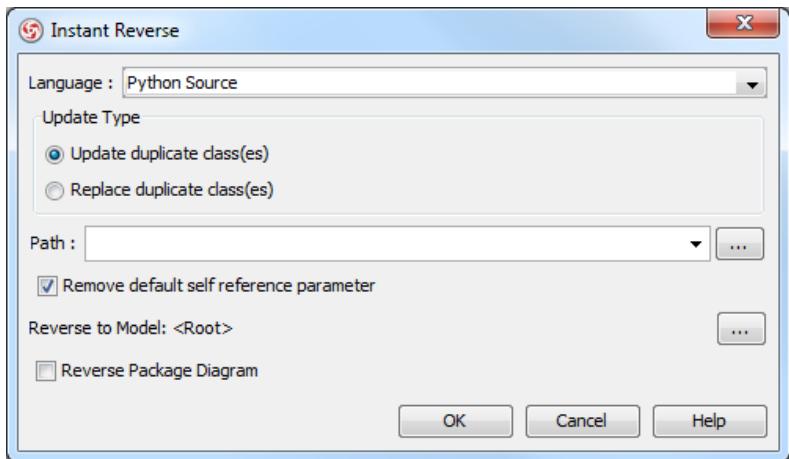
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> and so forth.
5	Reverse package diagram	Whether or not to reverse engineering 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.
-------------------	---

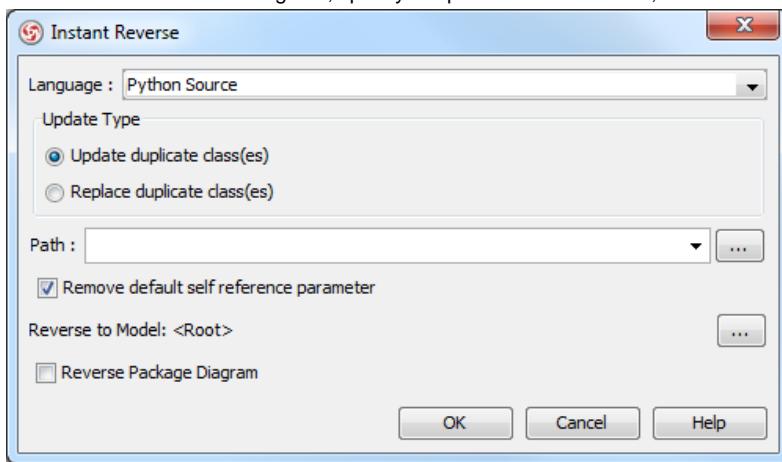
Type options	Fully-qualified - Show fully-qualified name of types. Name only - Show name of types. Relative - Show name of types relative to this class.
--------------	---

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 > Instant Reverse > .NET dll or exe files...** from the main menu.
2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



The **instant reverse** dialog

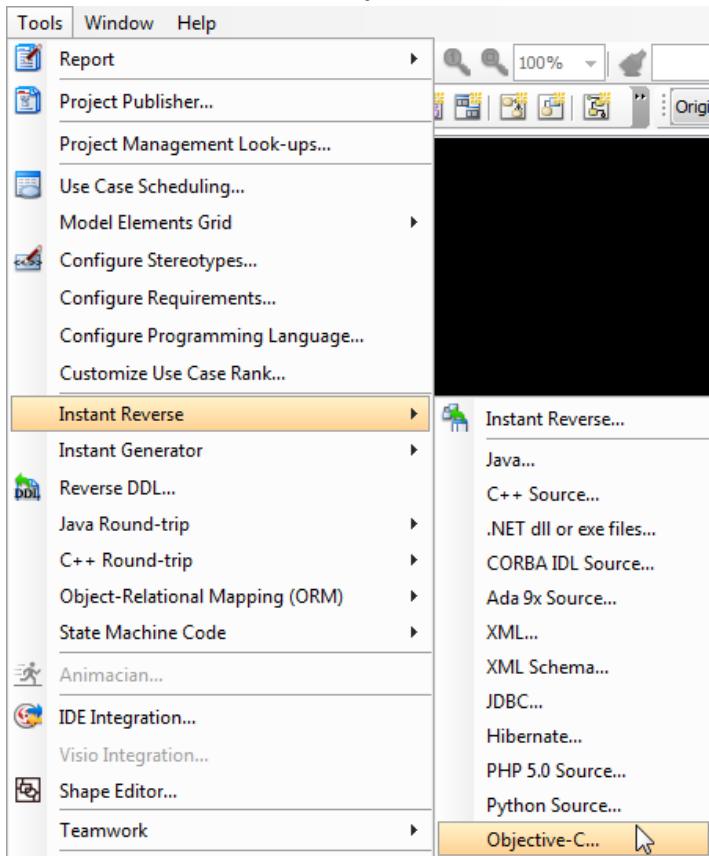
3. Check **Reverse Package Diagram**.
4. You can place reversed packages to specific model. To do this:
 1. Click on the ... button at the end of the **Reverse to Model** row.
 2. In the **Select Parent Model** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
5. Click **OK** to start reversing.

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.

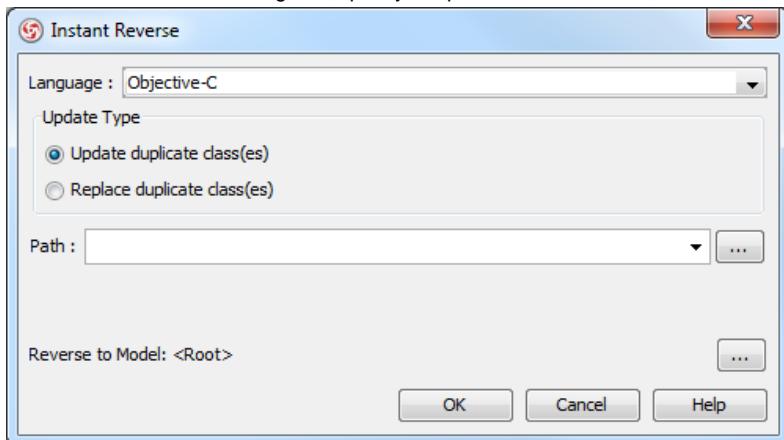
Reverse engineering UML classes from source files

1. Select **Tools > Instant Reverse > Objective-C...** from the main menu.



To perform instant reverse

2. In the **Instant Reverse** dialog box, specify the path of the source file, or the folder that contain those files.



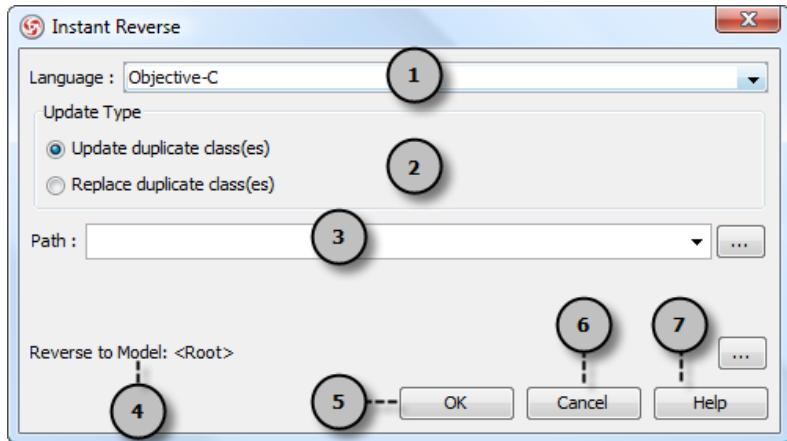
The **Instant Reverse** dialog

3. 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** dialog box, either select an existing model, or create one by clicking **New Model**.
 3. Click **OK** to confirm.
4. Click **OK** to start reversing.

5. Upon finishing, you will see the **Instant Reverse Form Diagram** dialog box 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



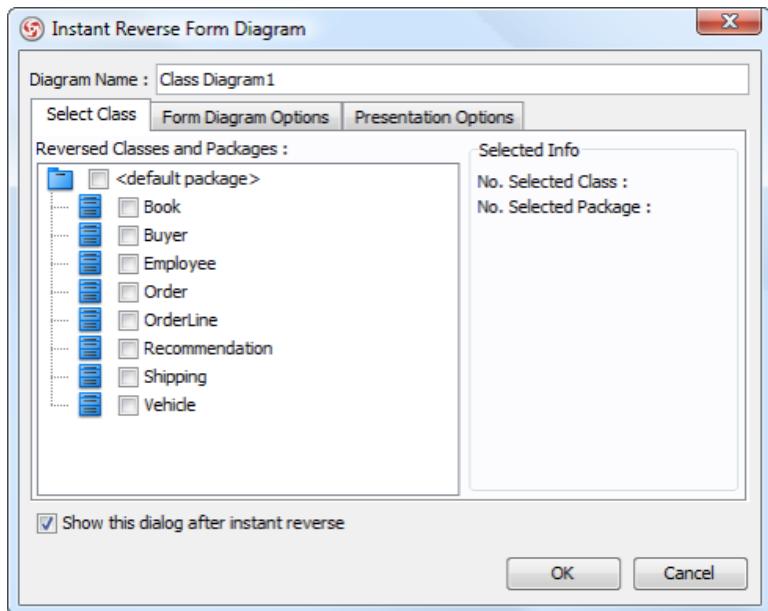
The instant reverse dialog box

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. You can click + to add multiple paths.
4	Reverse to model	Place reversed classes to specific model. For example, to place legacy code to a model named <i>Old</i> , to place system prototype to a model named <i>Prototype</i> 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 dialog box

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** dialog box

NOTE: If you do not want VP-UML to ask you for forming diagram next time you perform instant reverse, uncheck Show this dialog after instant reverse.

Below is a description of this dialog, 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.

Presentation Options

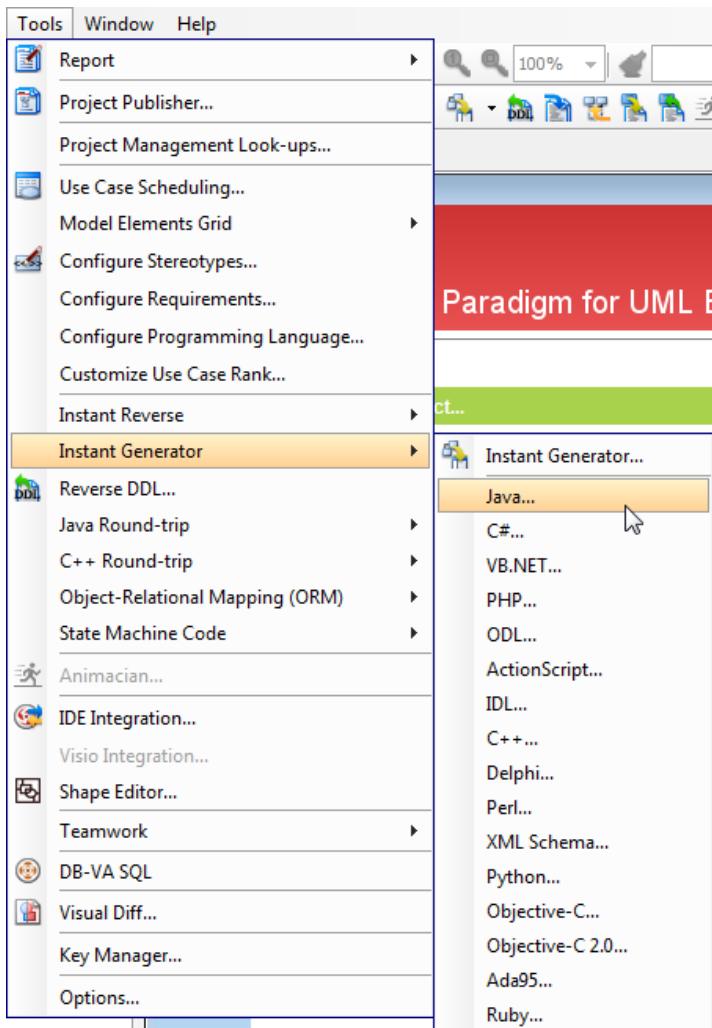
Option	Description
Attribute options	Show all - Show all attributes of classes in the new diagram. Public only - Show all public attributes of classes only in the new diagram. Hide all - Hide all attributes of classes in the new diagram. Initial values - Show initial values of attributes of classes in 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

Instant Generator for Java

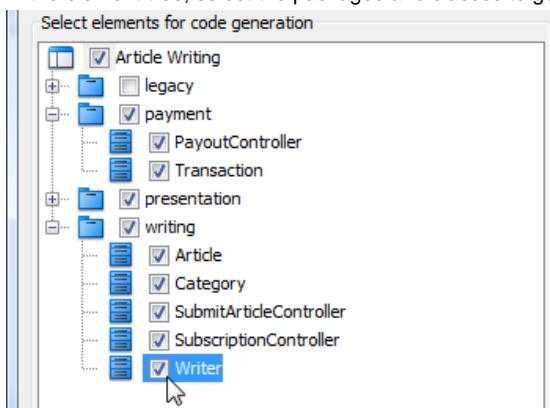
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Java...** from the main menu.



To open instant generator

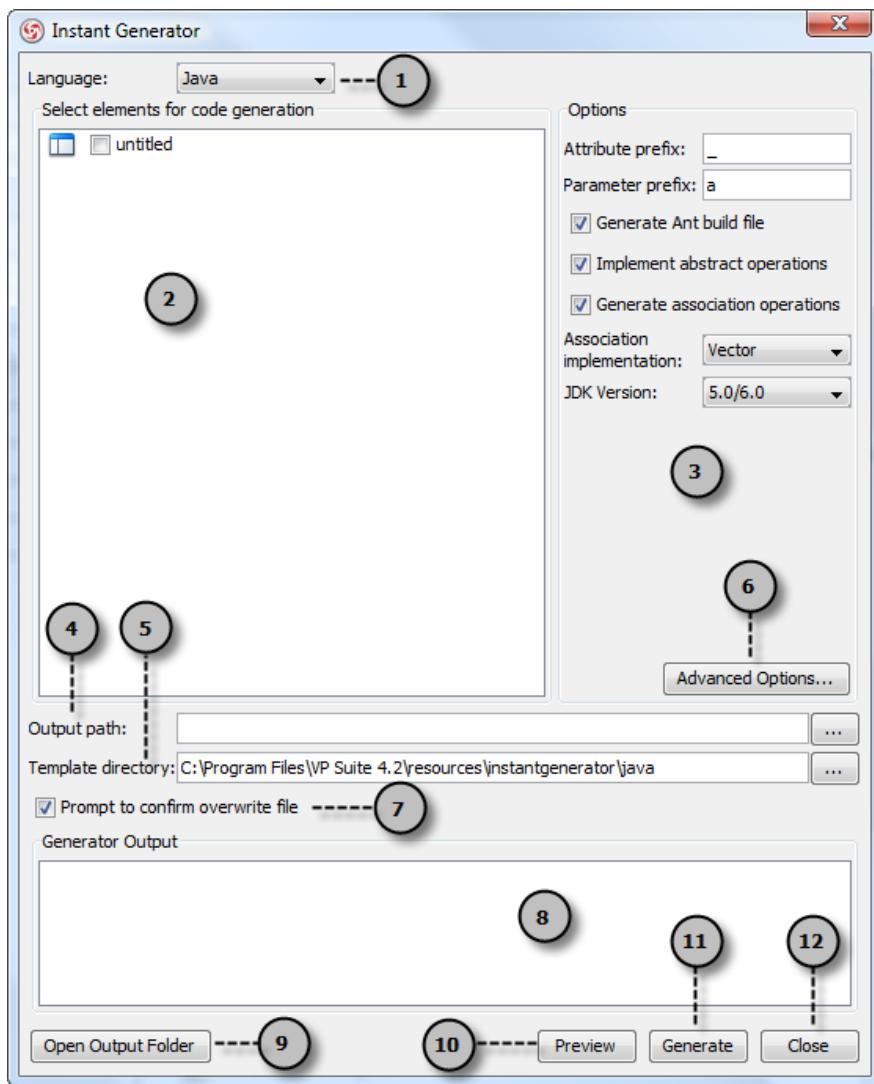
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

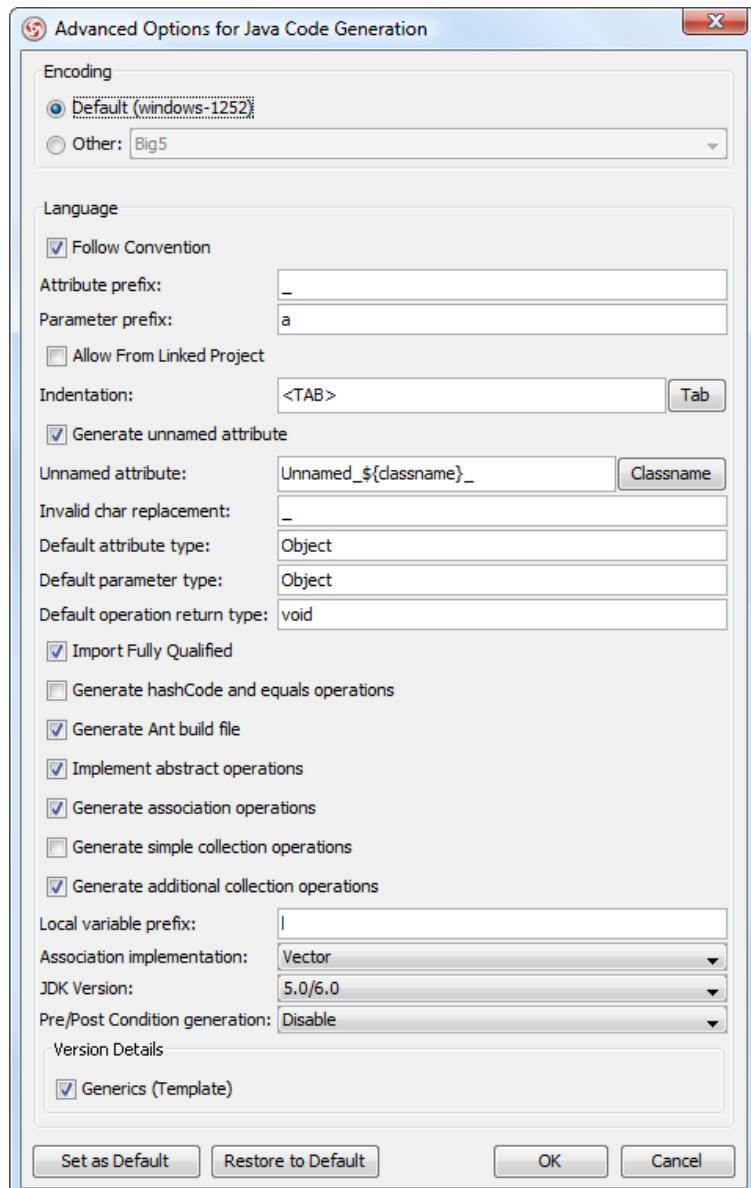


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

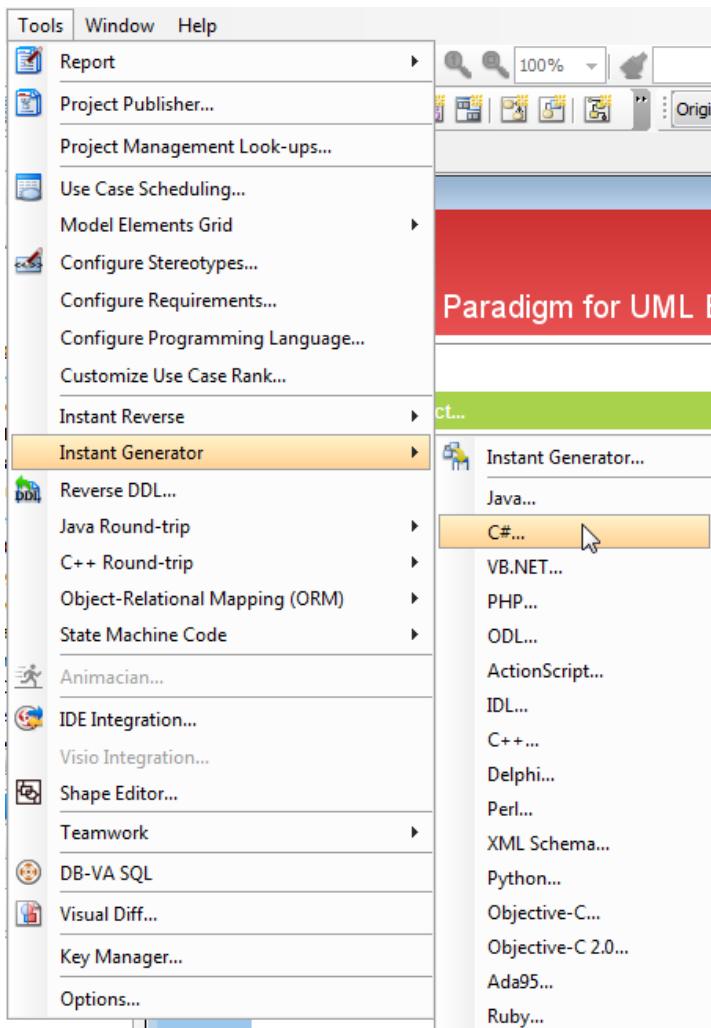
Option	Description
Encoding	The encoding of source file.
Follow Convention	Whether to apply camel case Java naming convention.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate 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 define 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

Instant Generator C# source code

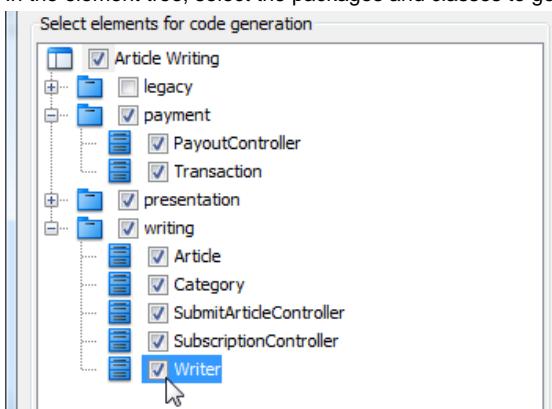
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > C# ...** from the main menu.



To open instant generator

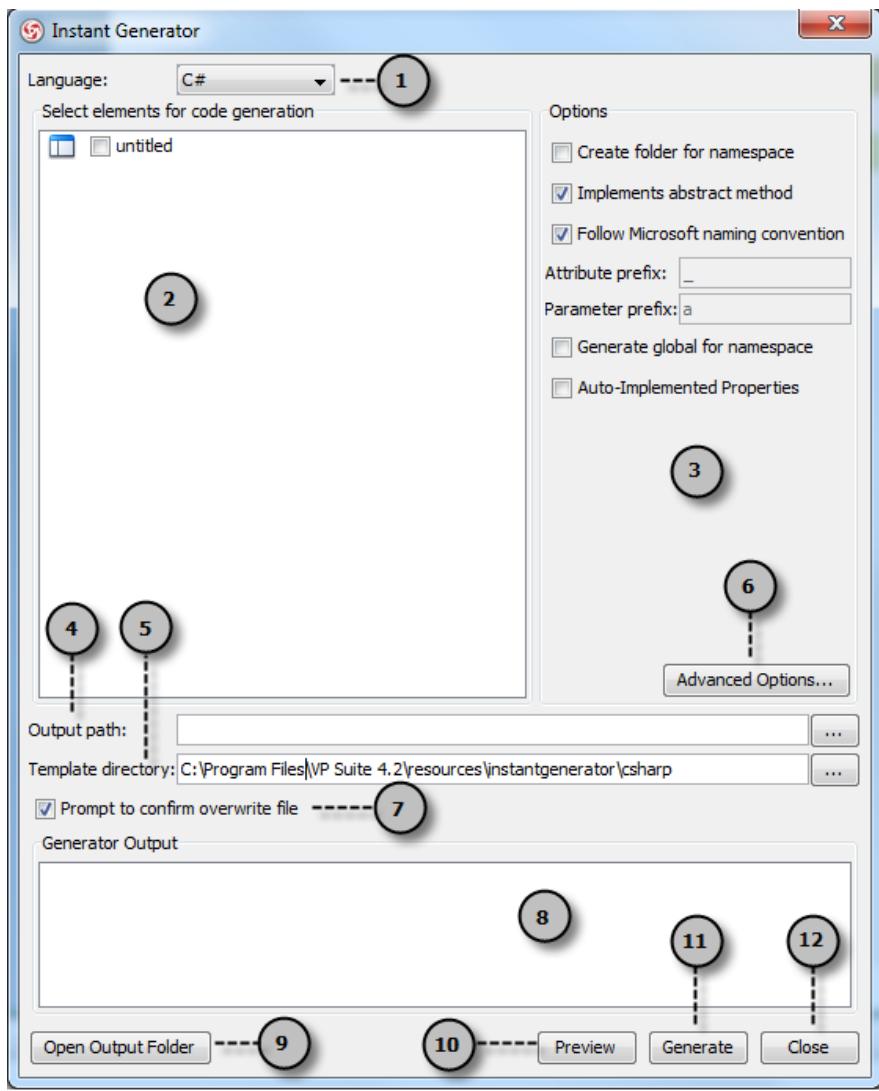
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

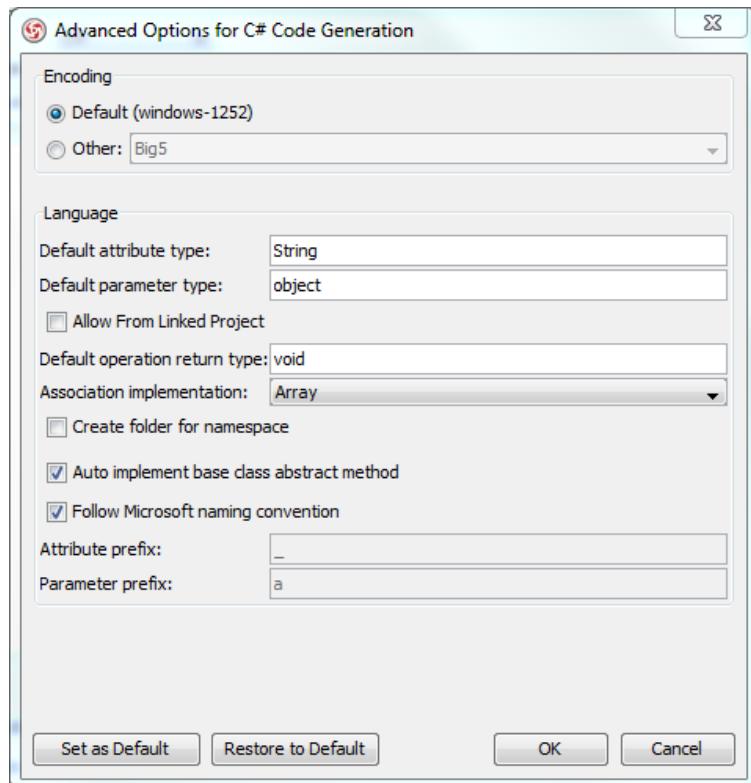


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

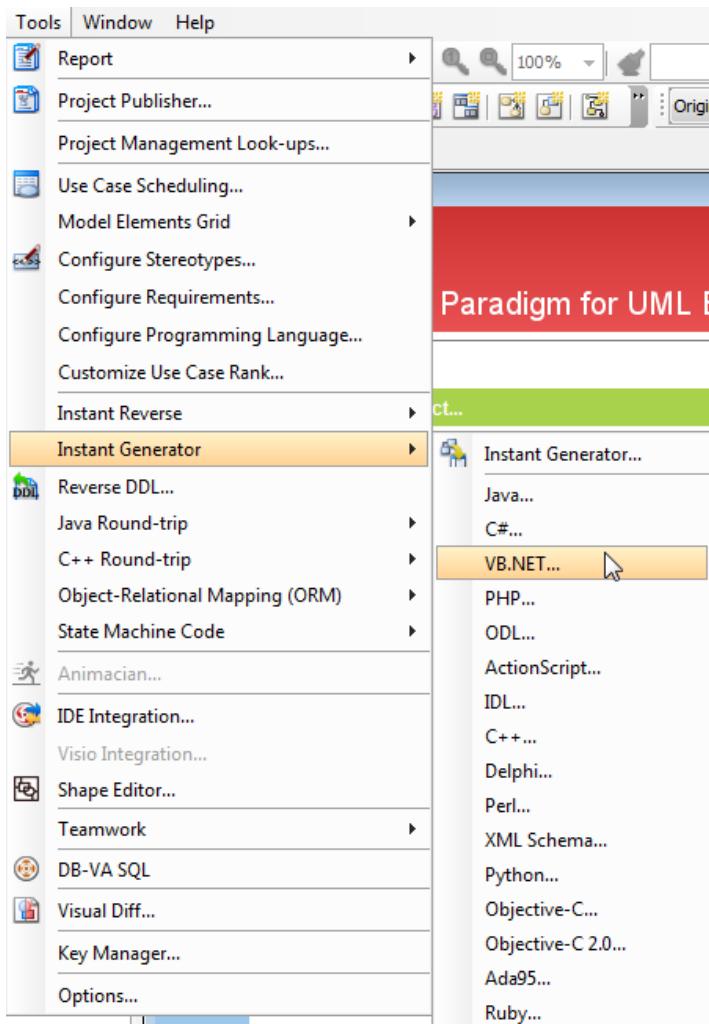
Option	Description
Encoding	The encoding of source file.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Default parameter type	Parameter type that will be used when parameter has no type specified.
Allow From Linked Project	Check to generate also classes 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

Instant Generator for VB.NET source code

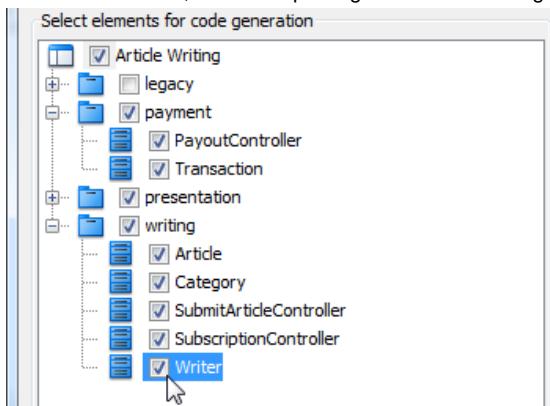
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > VB.NET ...** from the main menu.



To open instant generator

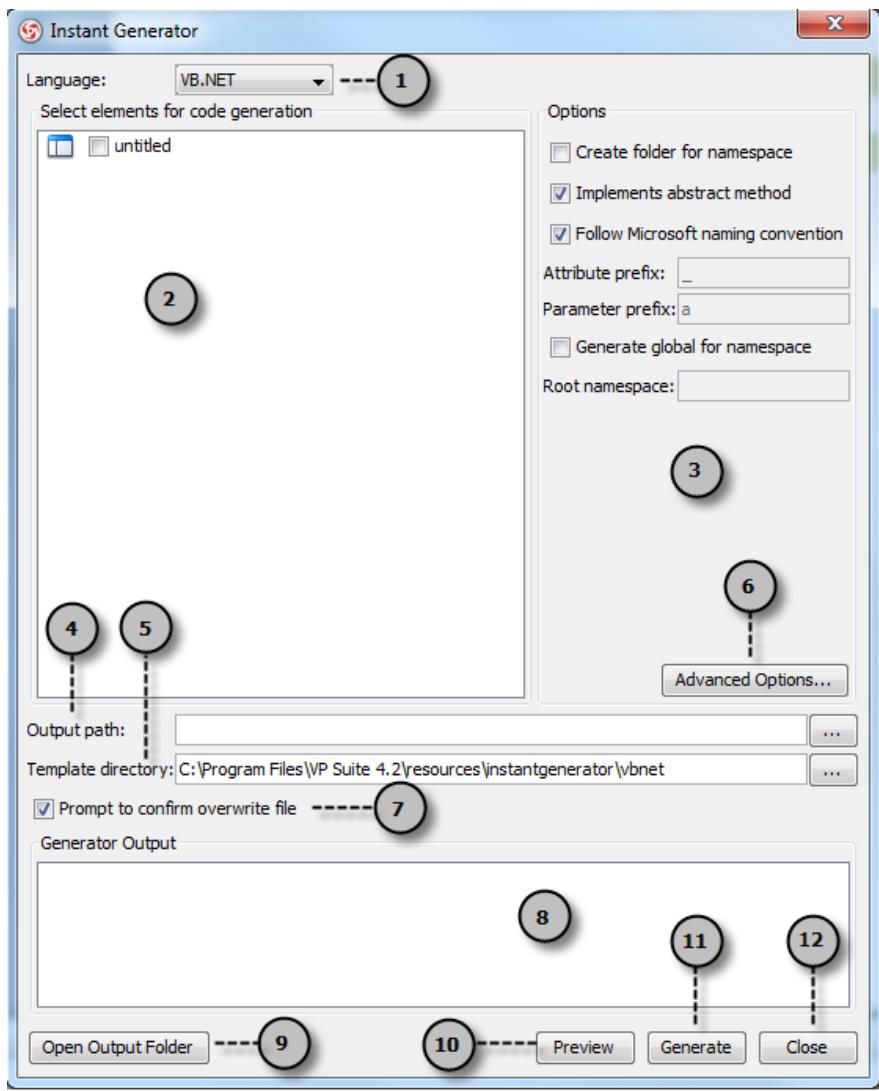
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

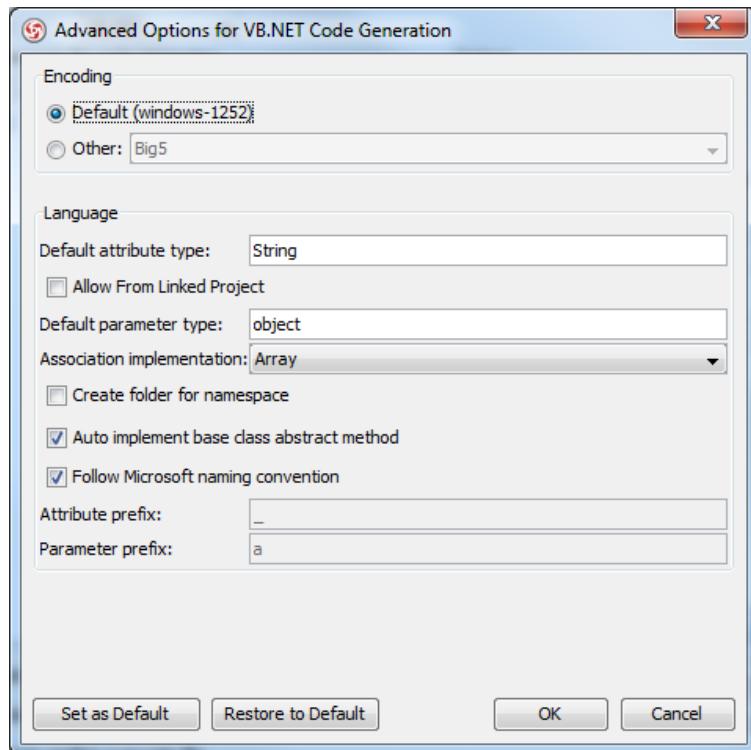


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

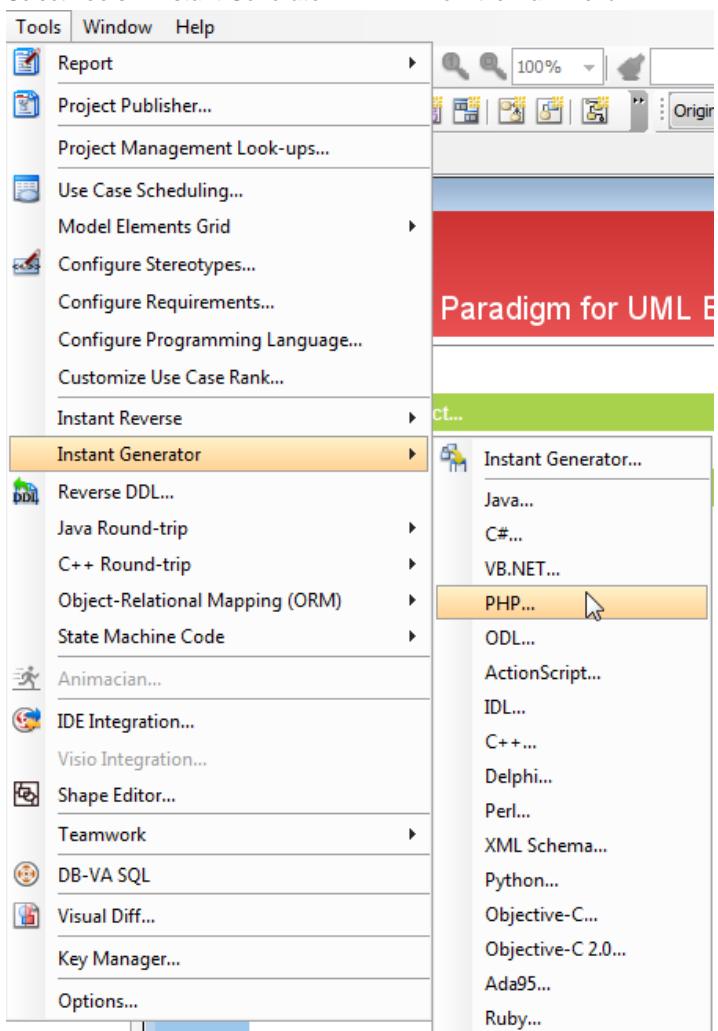
Option	Description
Encoding	The encoding of source file.
Default attribute type	Attribute type that will be used when attribute has no type specified.
Allow From Linked Project	Check to generate also classes in referenced project.
Default parameter type	Parameter type that will be used when parameter 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

Instant Generator for PHP source code

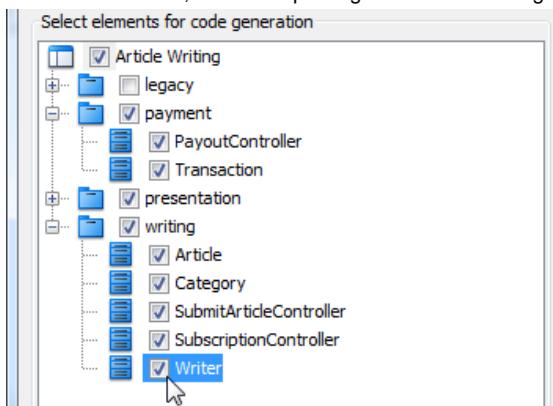
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > PHP ...** from the main menu.



To open instant generator

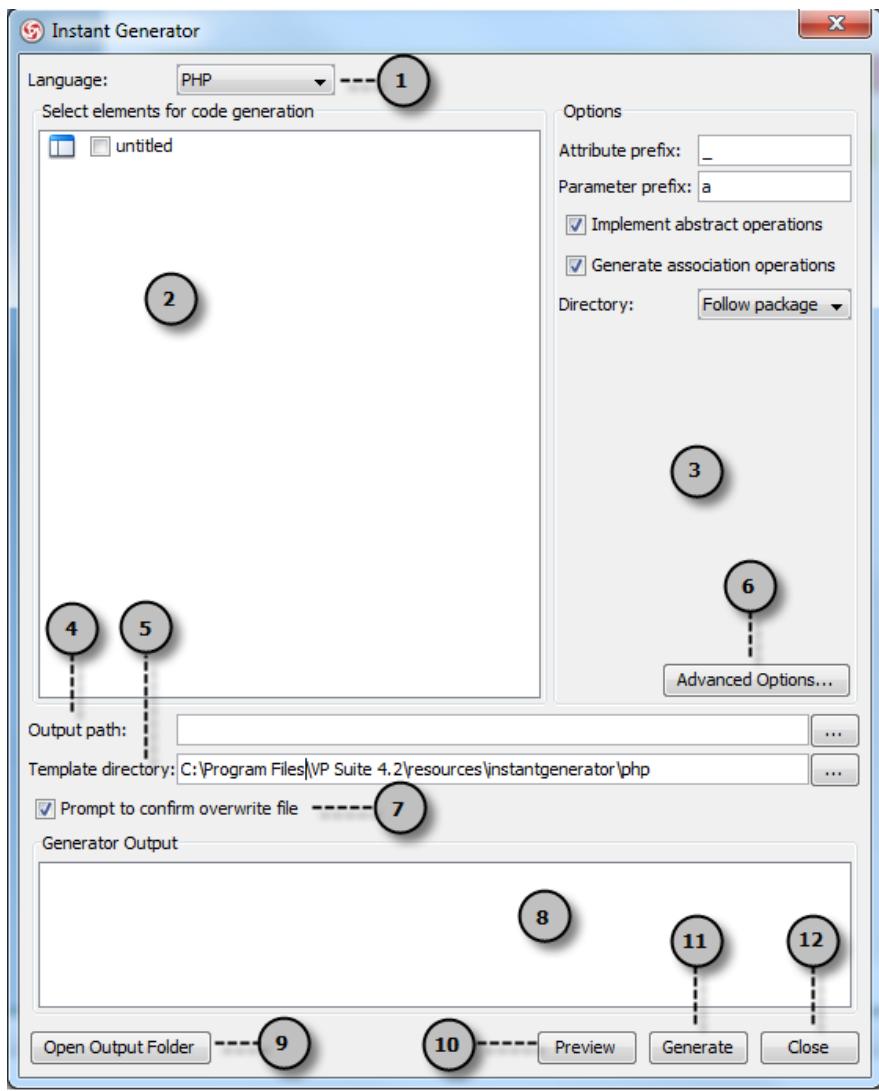
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

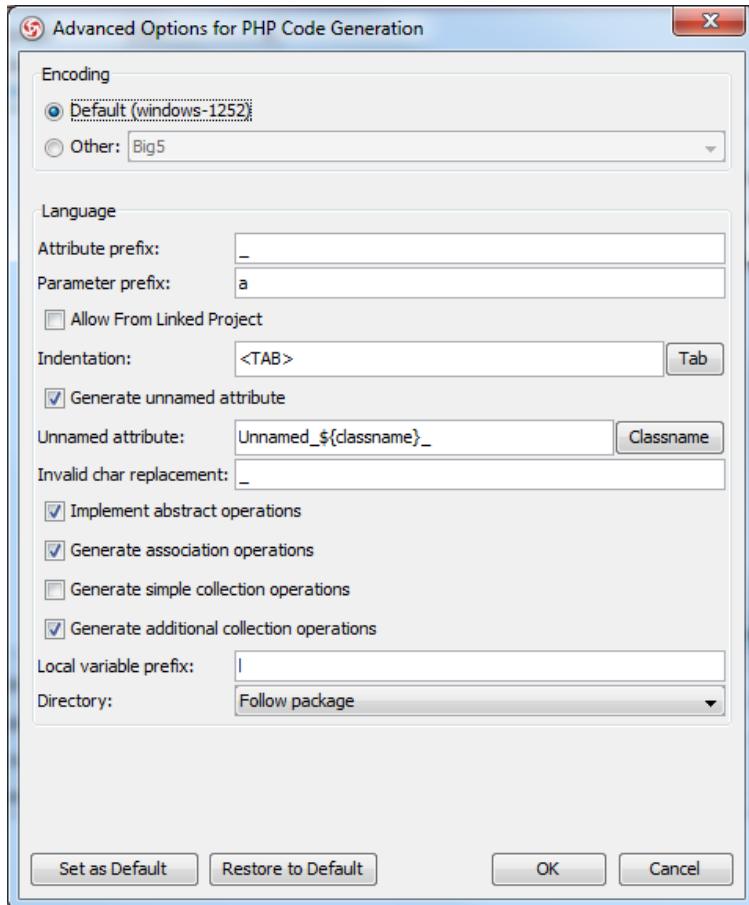


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

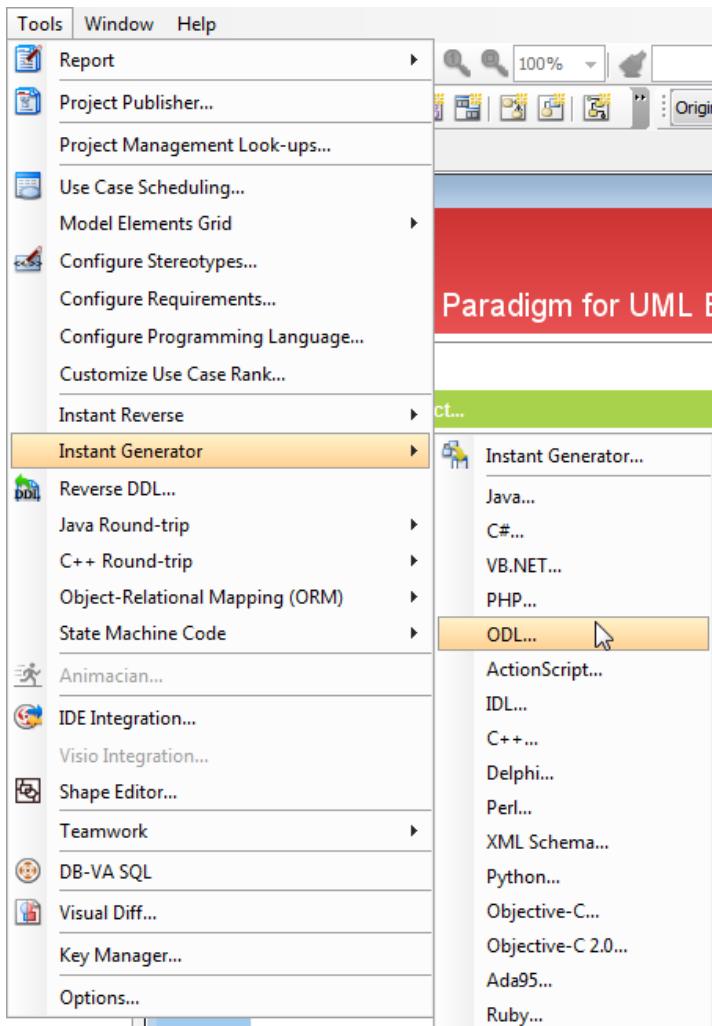
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) used for 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 those 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

Instant Generator for ODL source code

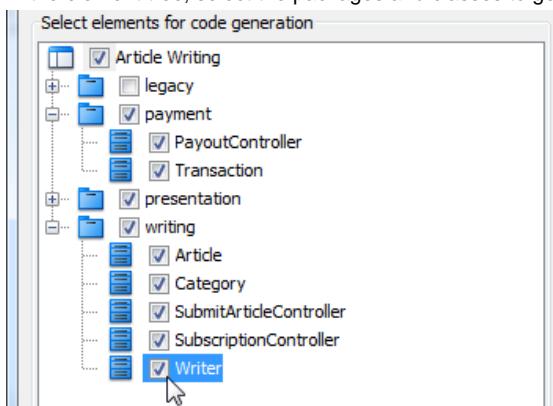
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > ODL ...** from the main menu.



To open instant generator

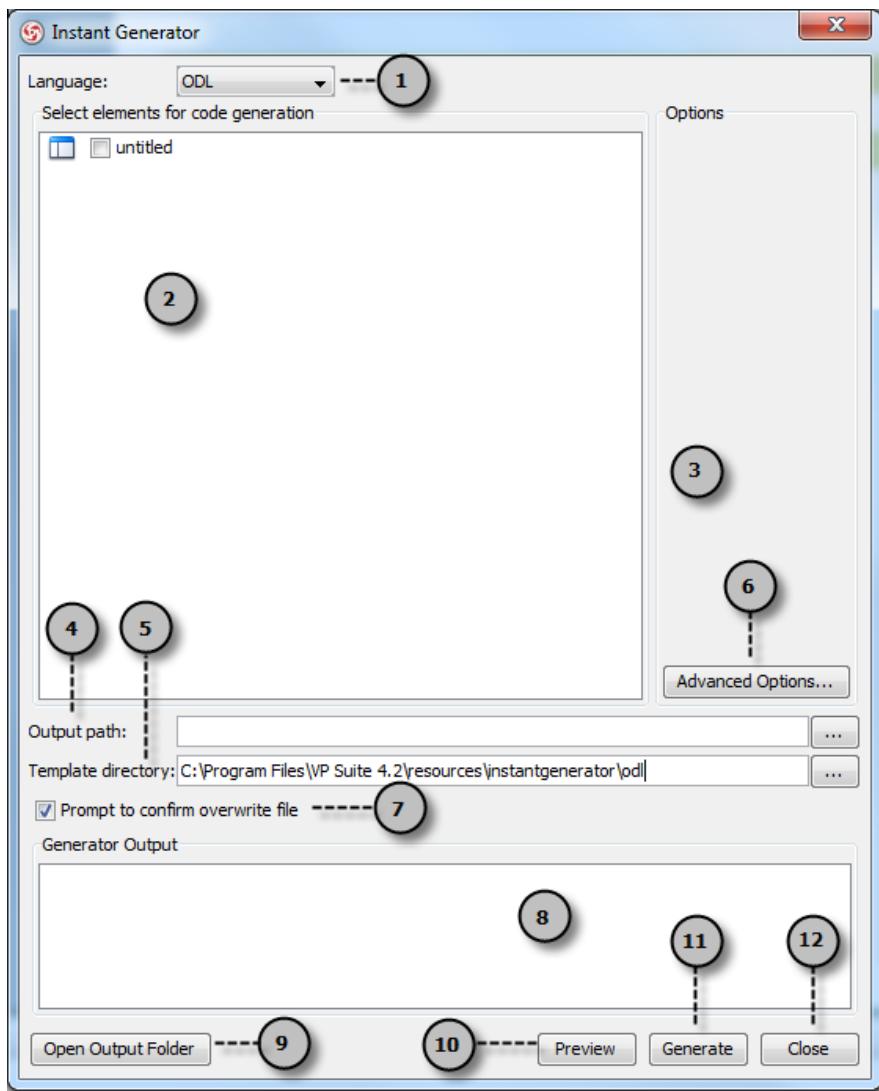
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

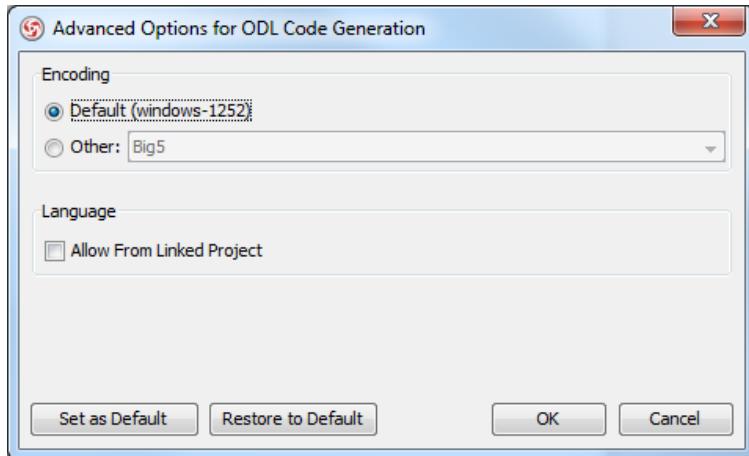


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

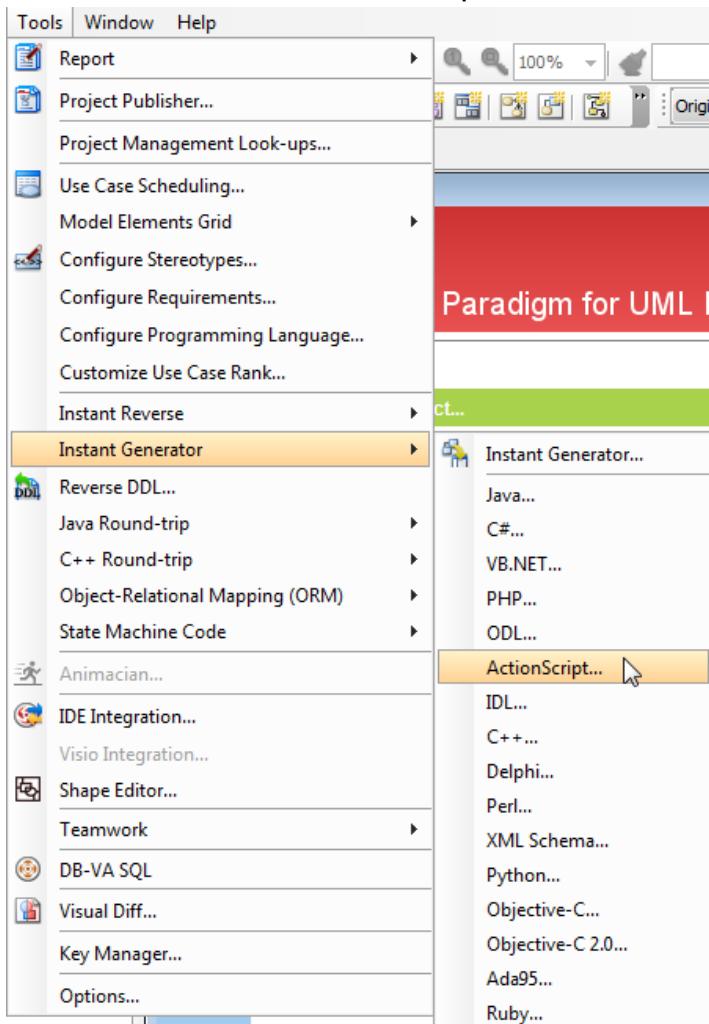
Option	Description
Encoding	The encoding of source file.
Allow From Linked Project	Check to generate also classes in referenced project.

A description of advanced options

Instant Generator for ActionScript source code

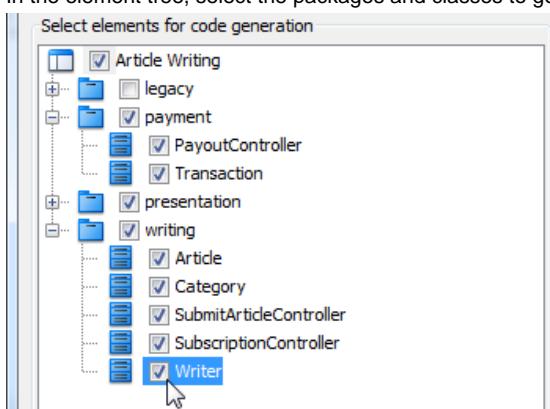
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > ActionScript ...** from the main menu.



To open instant generator

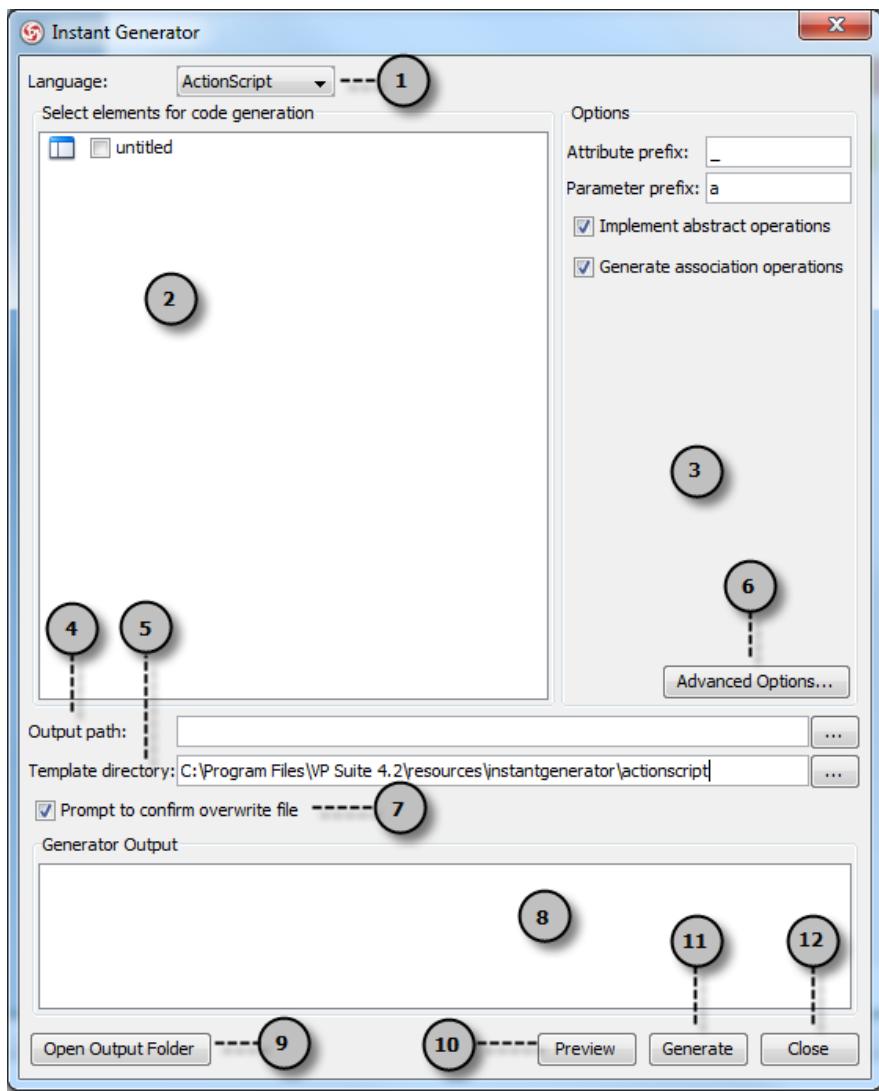
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

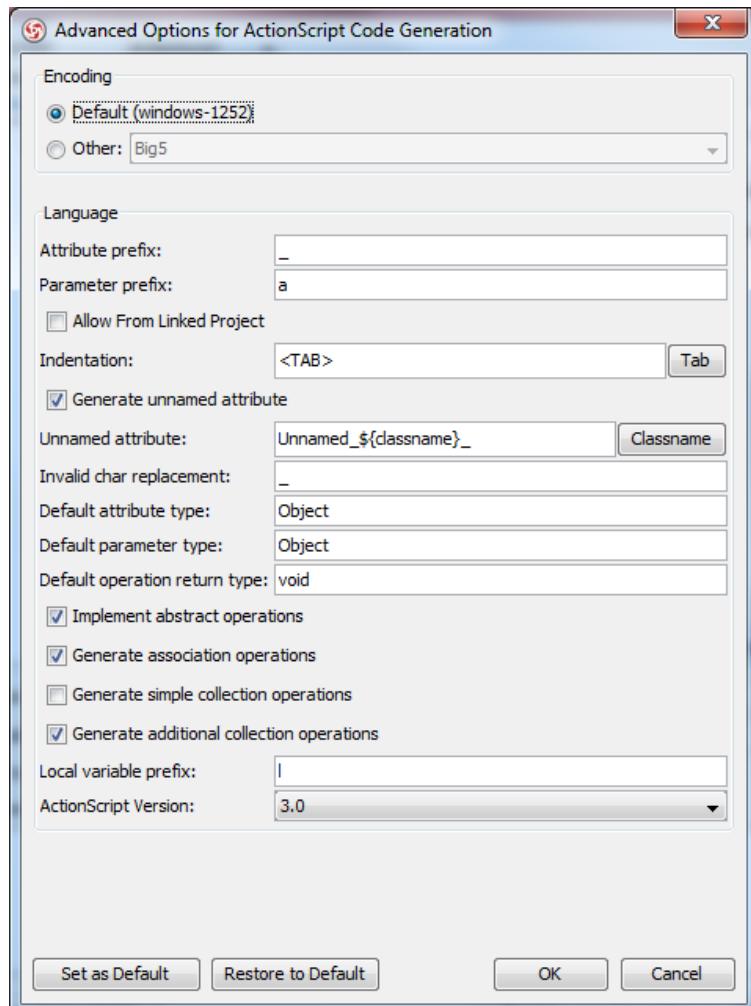


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

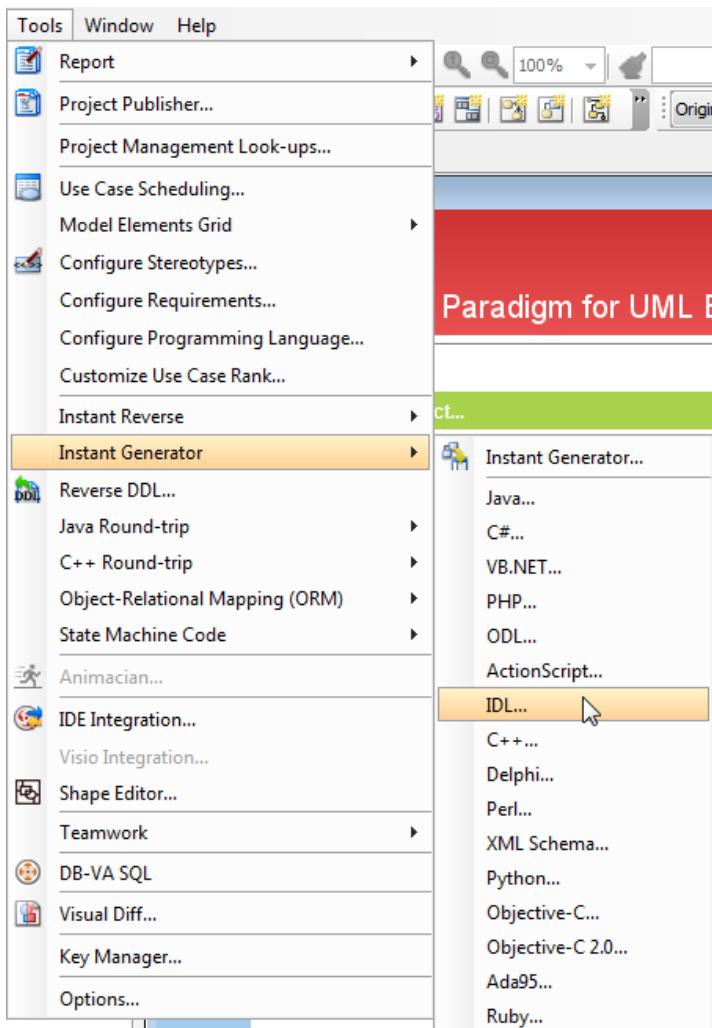
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used 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.
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

Instant Generator for IDL source code

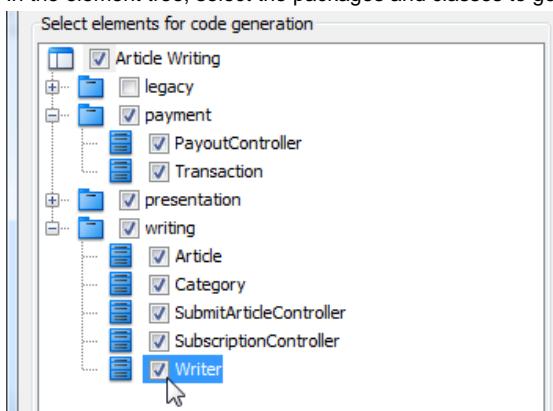
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > IDL ...** from the main menu.



To open instant generator

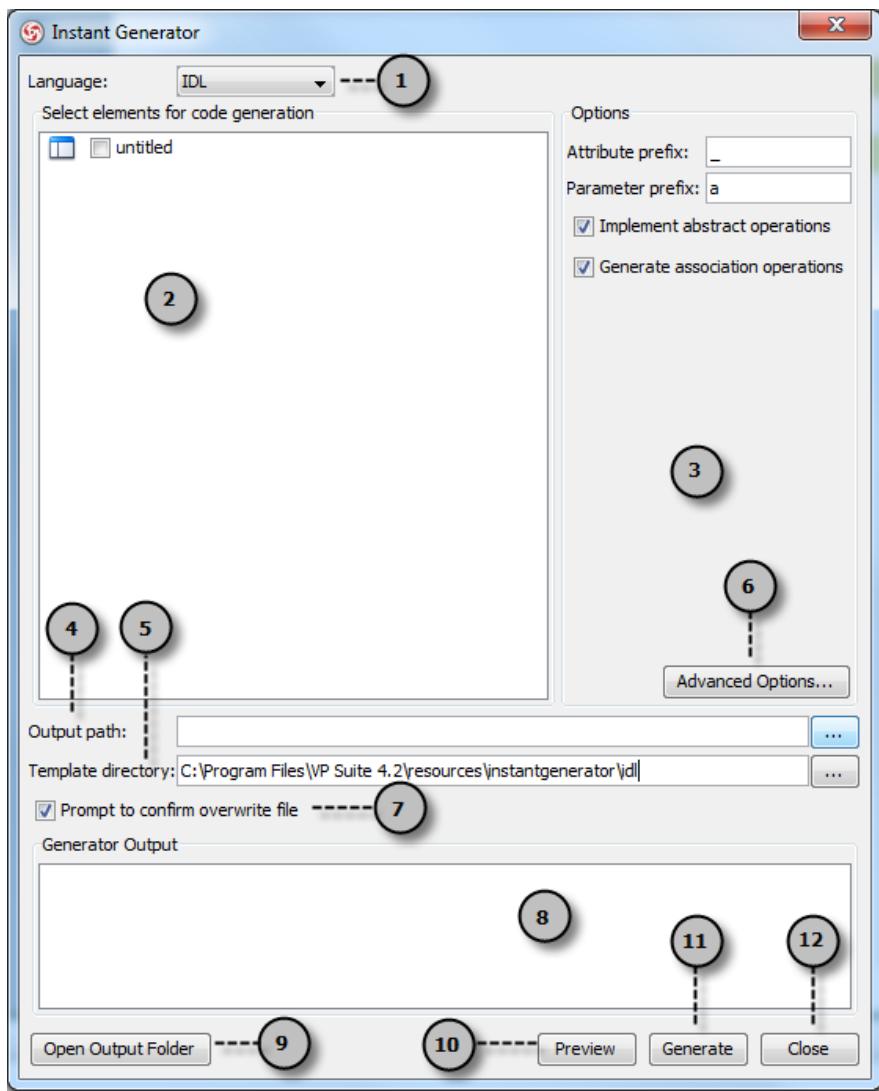
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

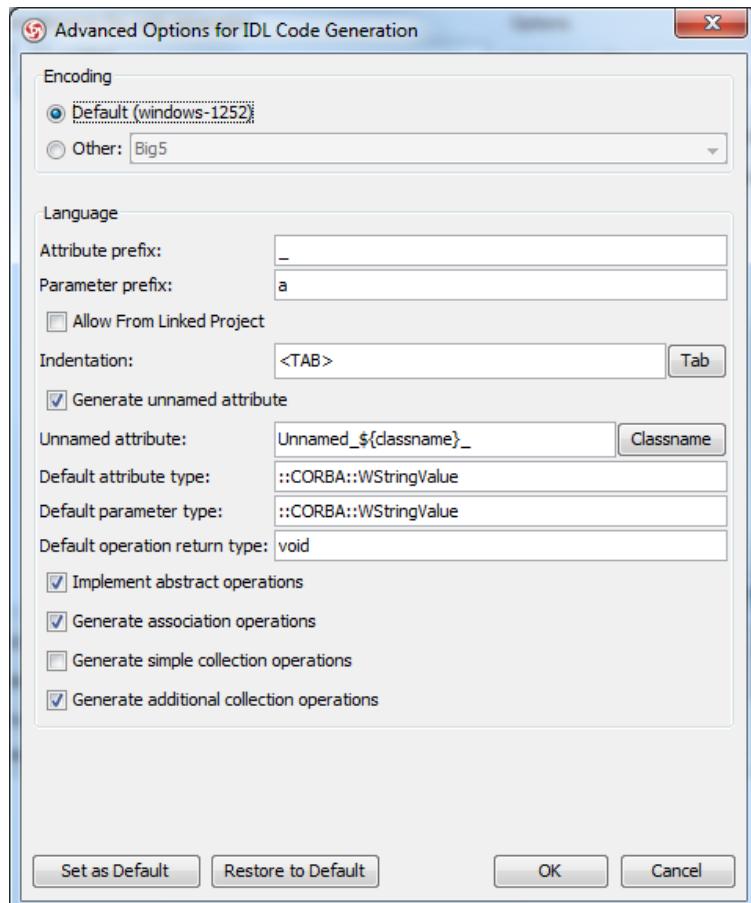


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

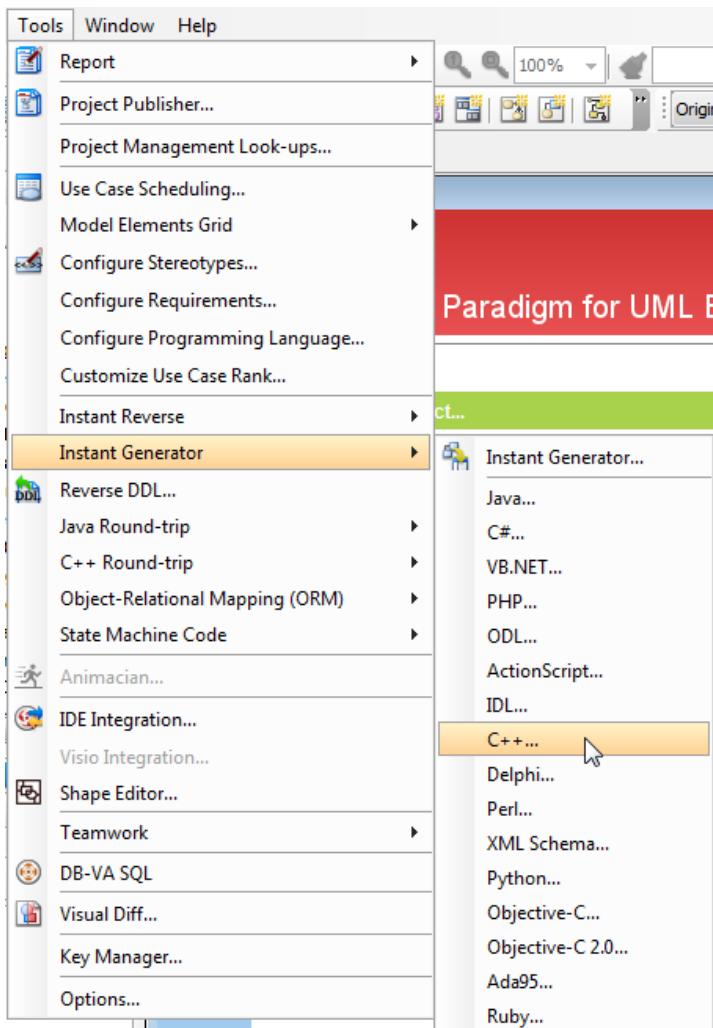
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used 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.
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

Instant Generator for C++ source code

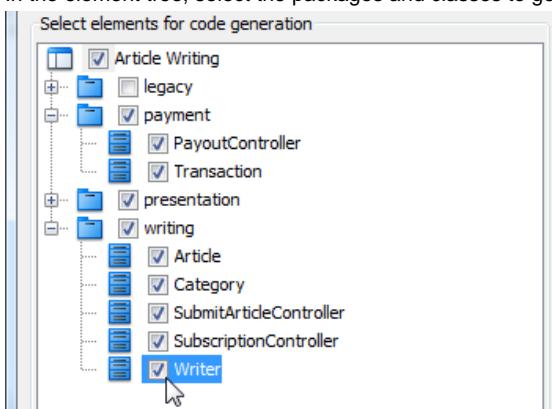
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > C++ ...** from the main menu.



To open instant generator

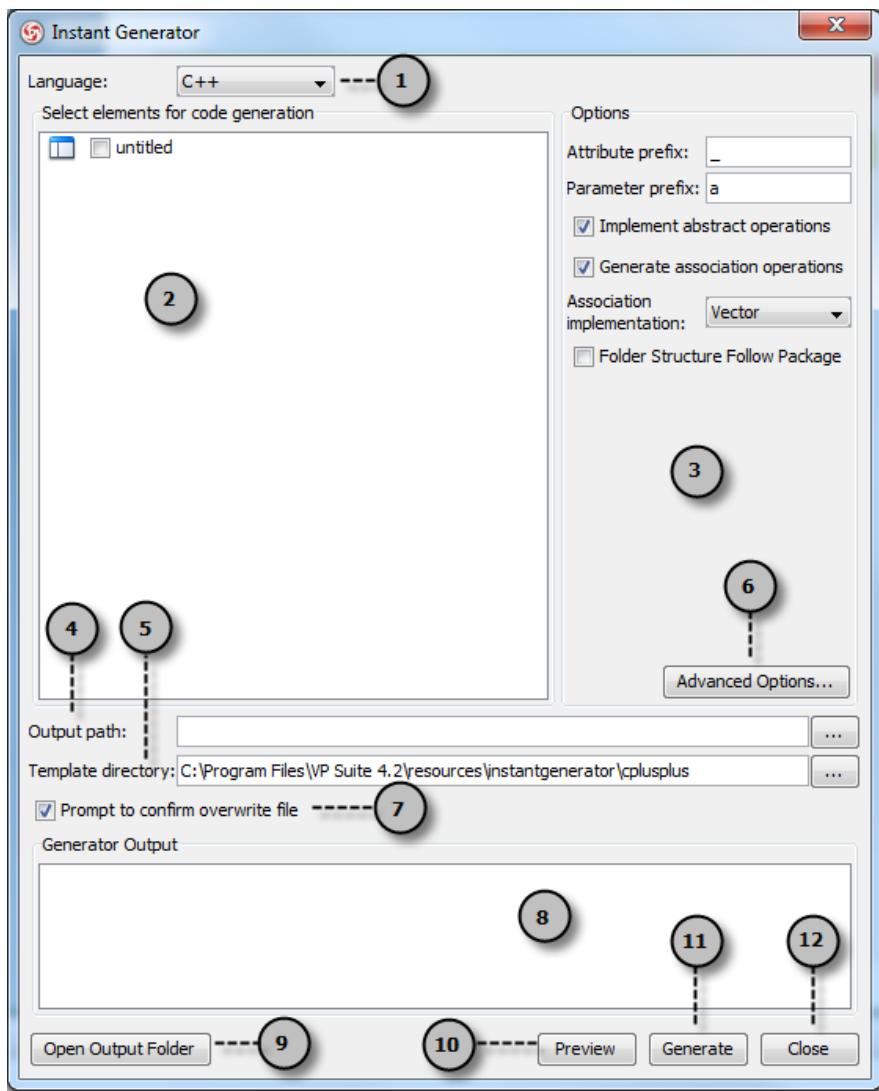
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

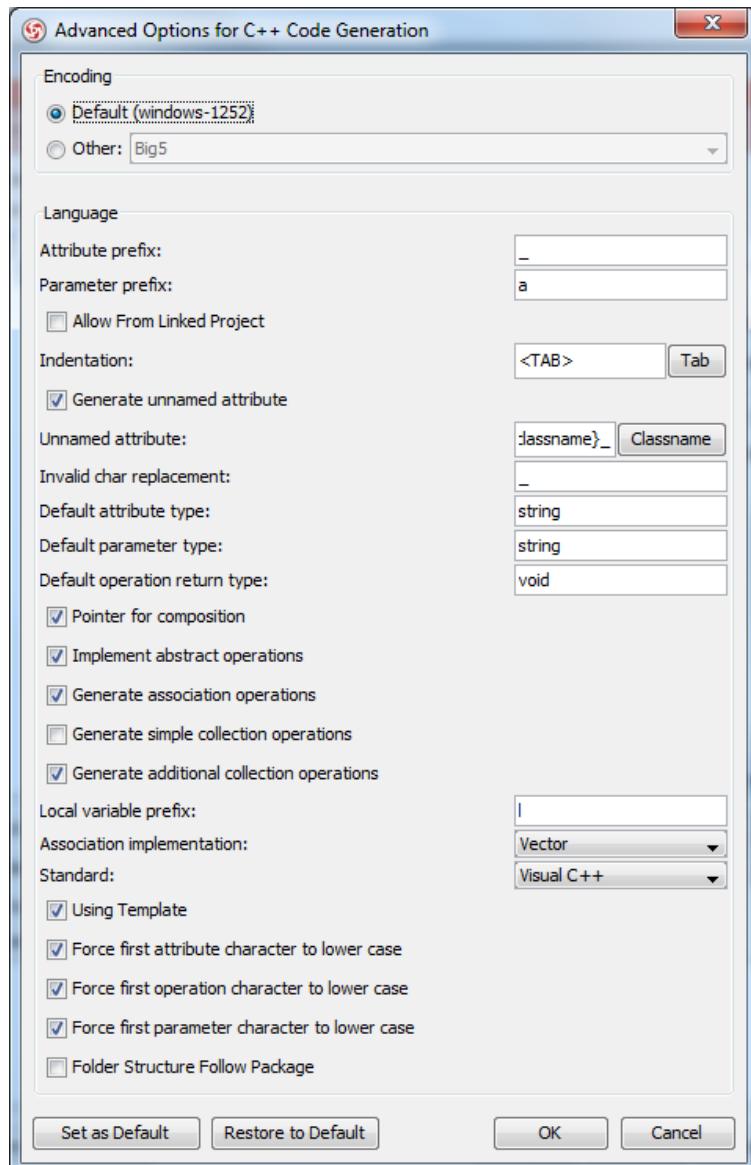


Overview of instant generator dialog box

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 VP-UML 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 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.

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.
Default attribute type	Attribute type that will be used when attribute has no type specified.

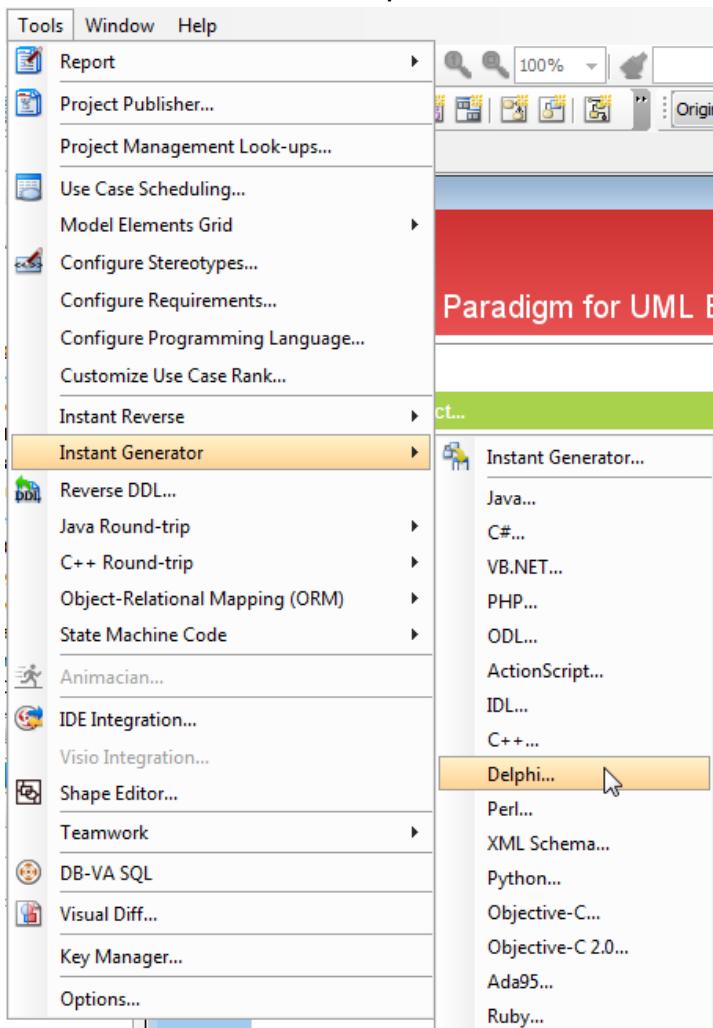
Default parameter type	Parameter type that will be used when parameter has no type specified.
Default operation return type	Operation return type that will be used when operation has no return type specified.
Pointer for composition	When checked, generate attribute for linking composited class using 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

Instant Generator for Delphi source code

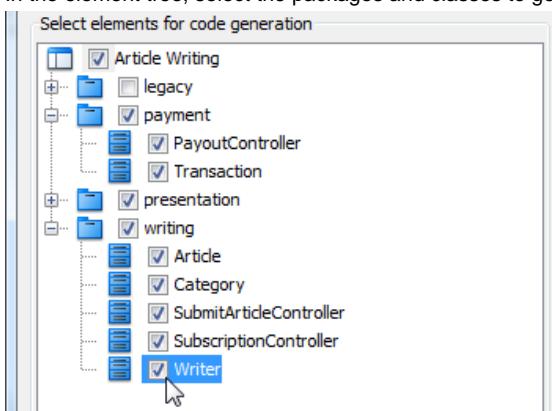
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Delphi ...** from the main menu.



To open instant generator

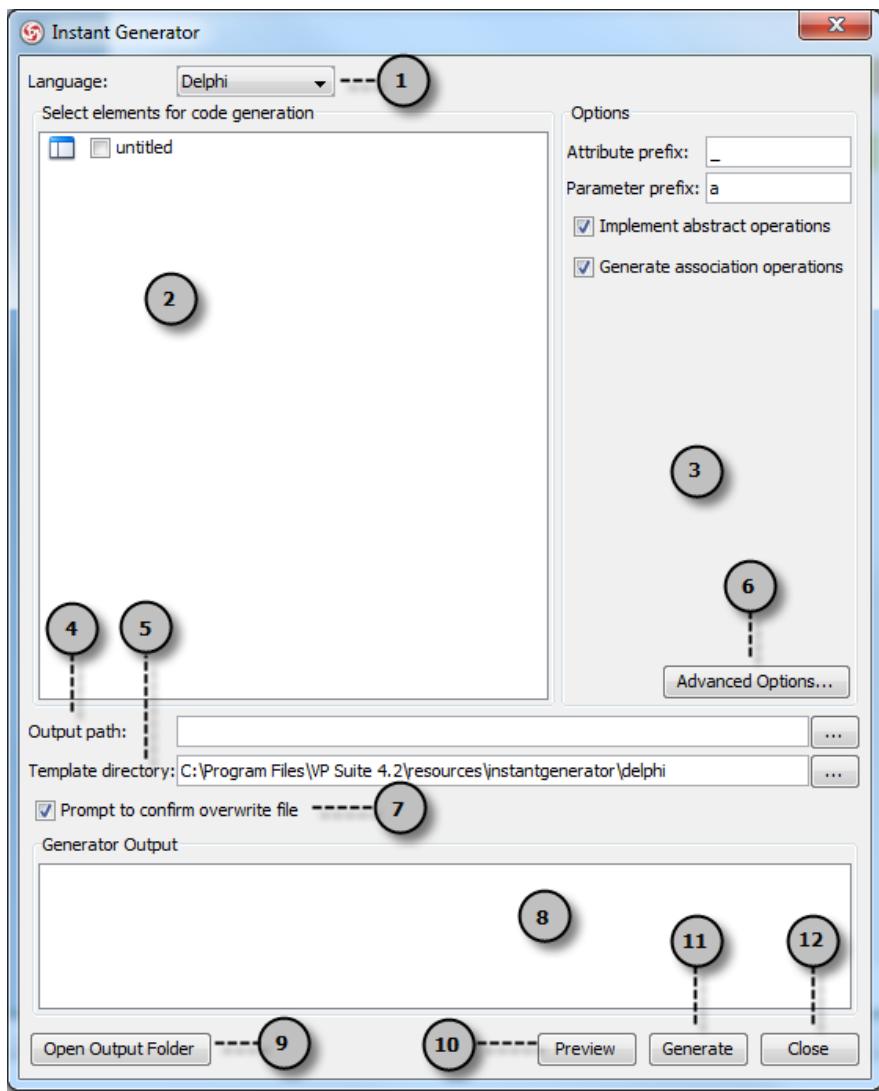
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

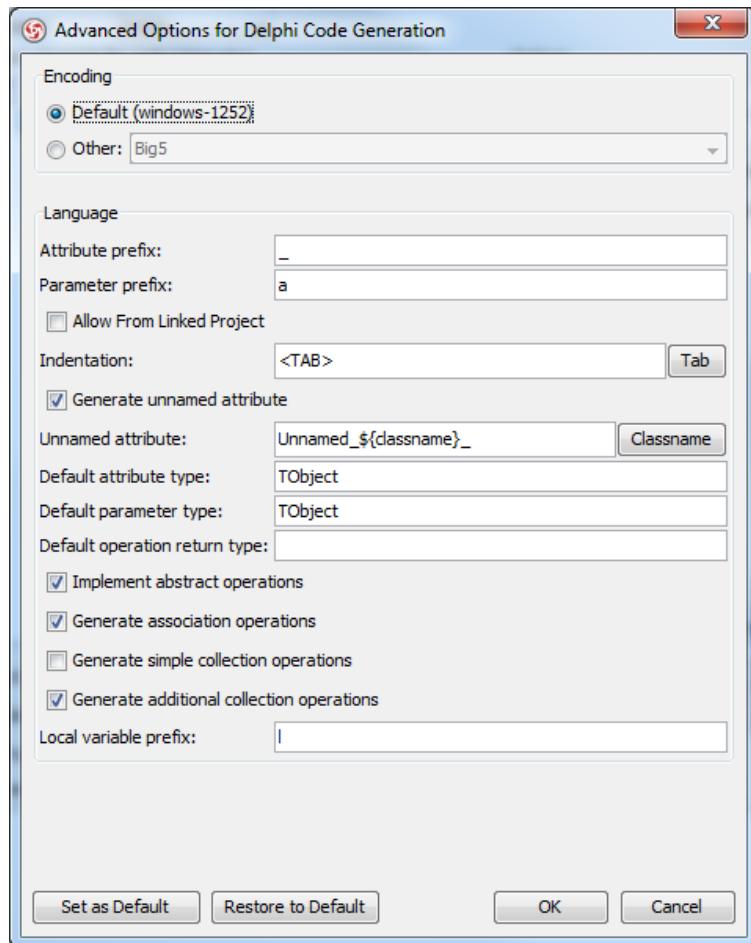


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

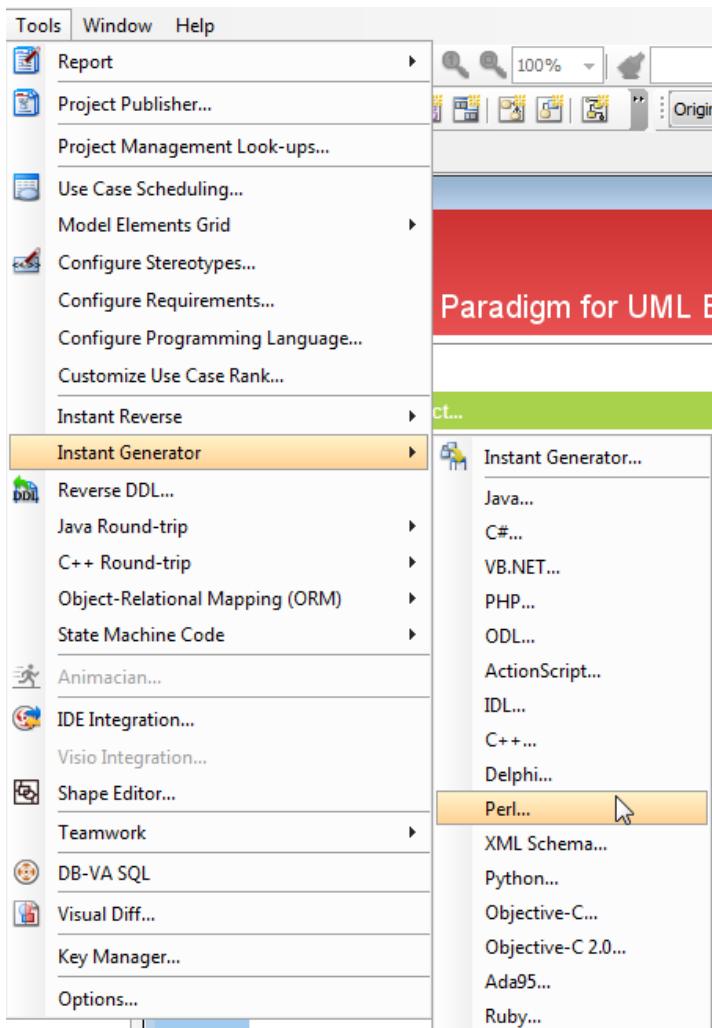
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used 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.
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

Instant Generator for Perl source code

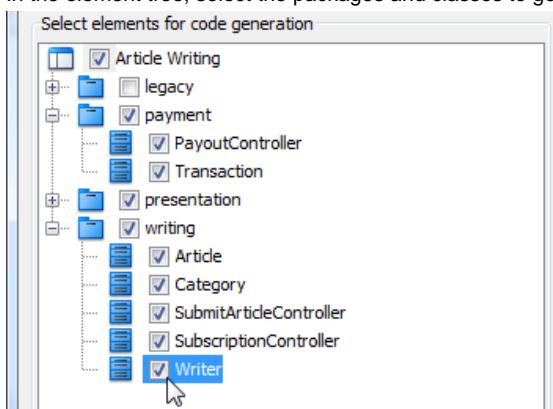
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Perl ...** from the main menu.



To open instant generator

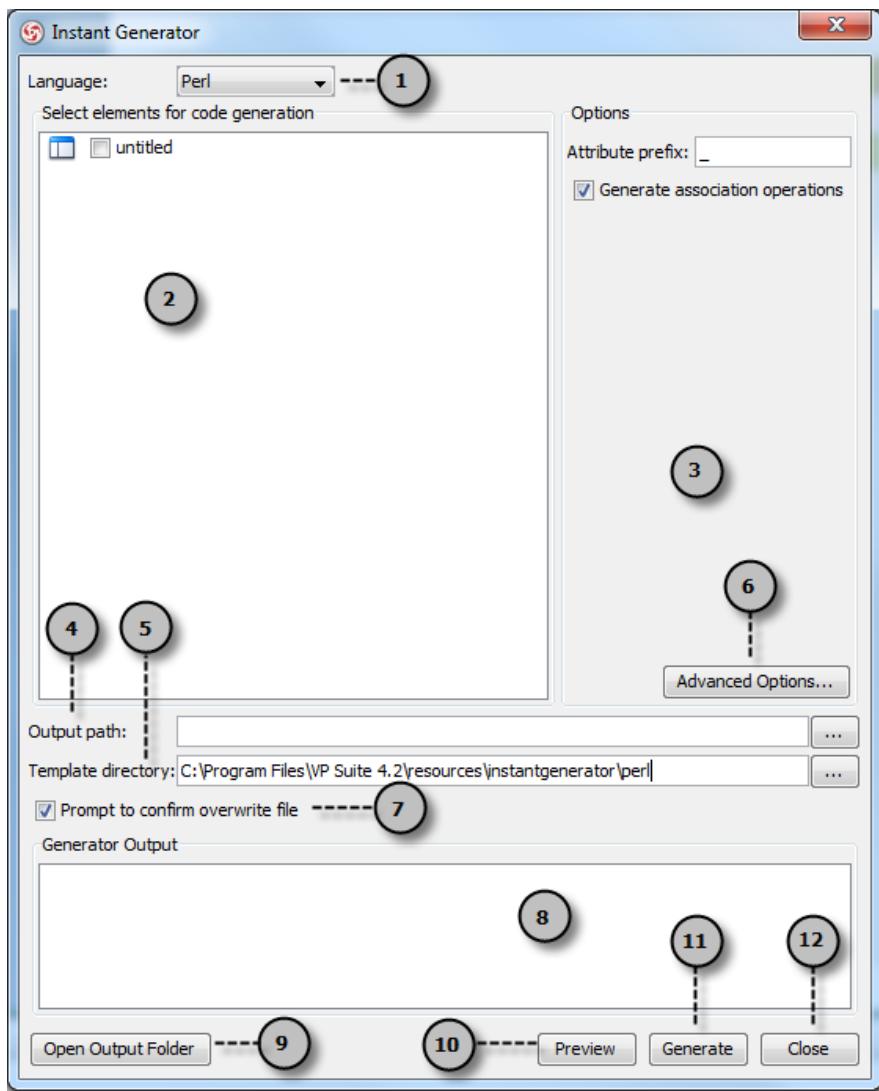
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

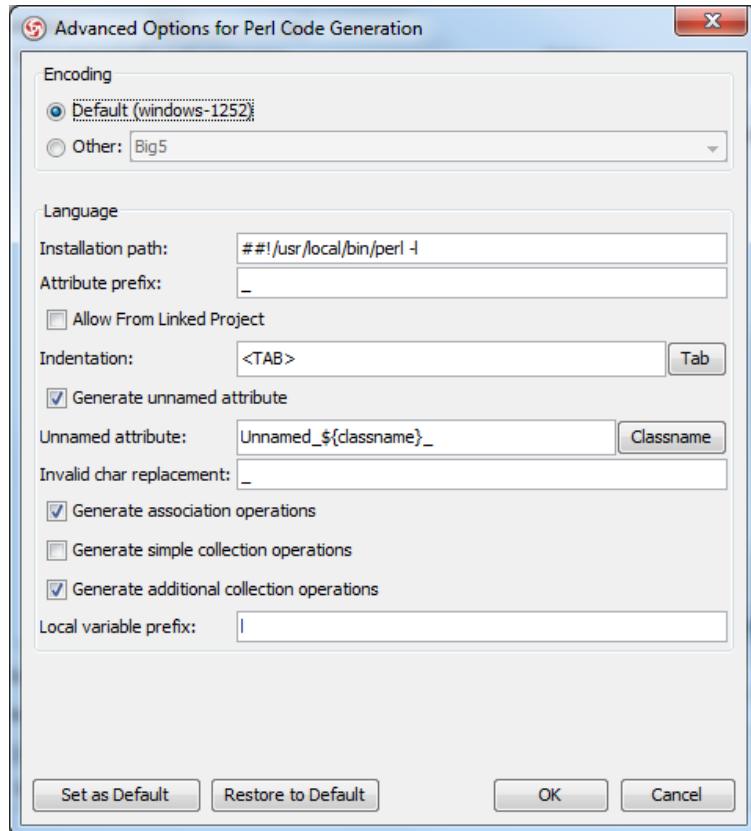


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

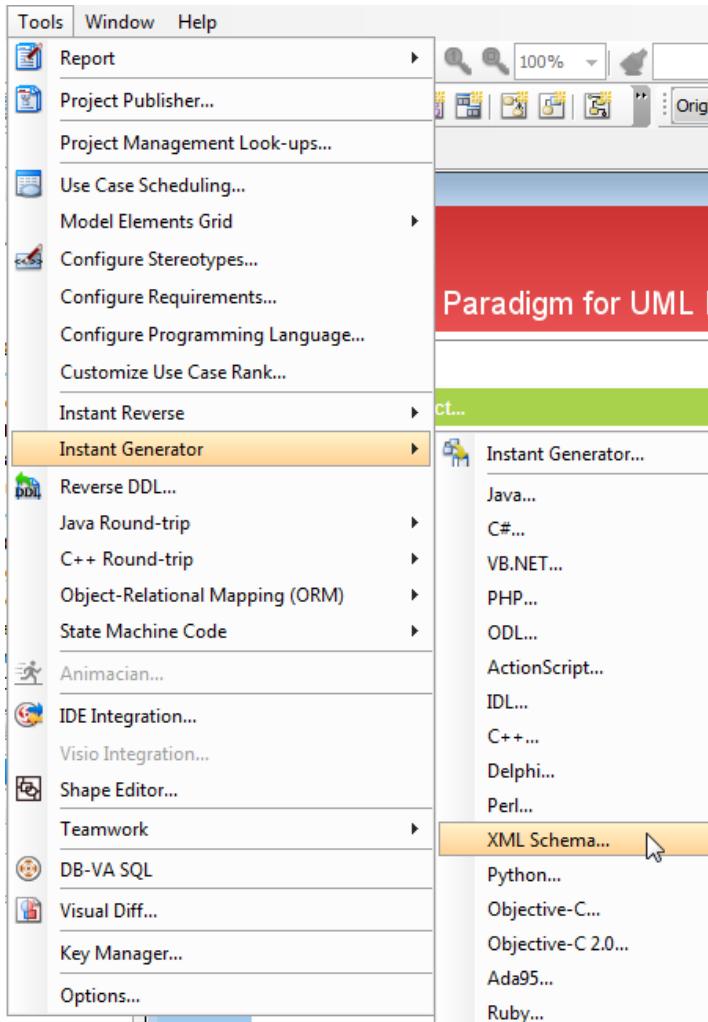
Option	Description
Encoding	The encoding of source file.
Installation path	The Perl installation path
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced 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

Instant Generator for XML Schema file

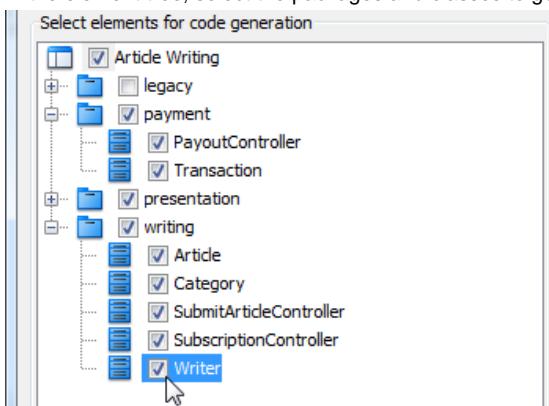
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > XML Schema ...** from the main menu.



To open instant generator

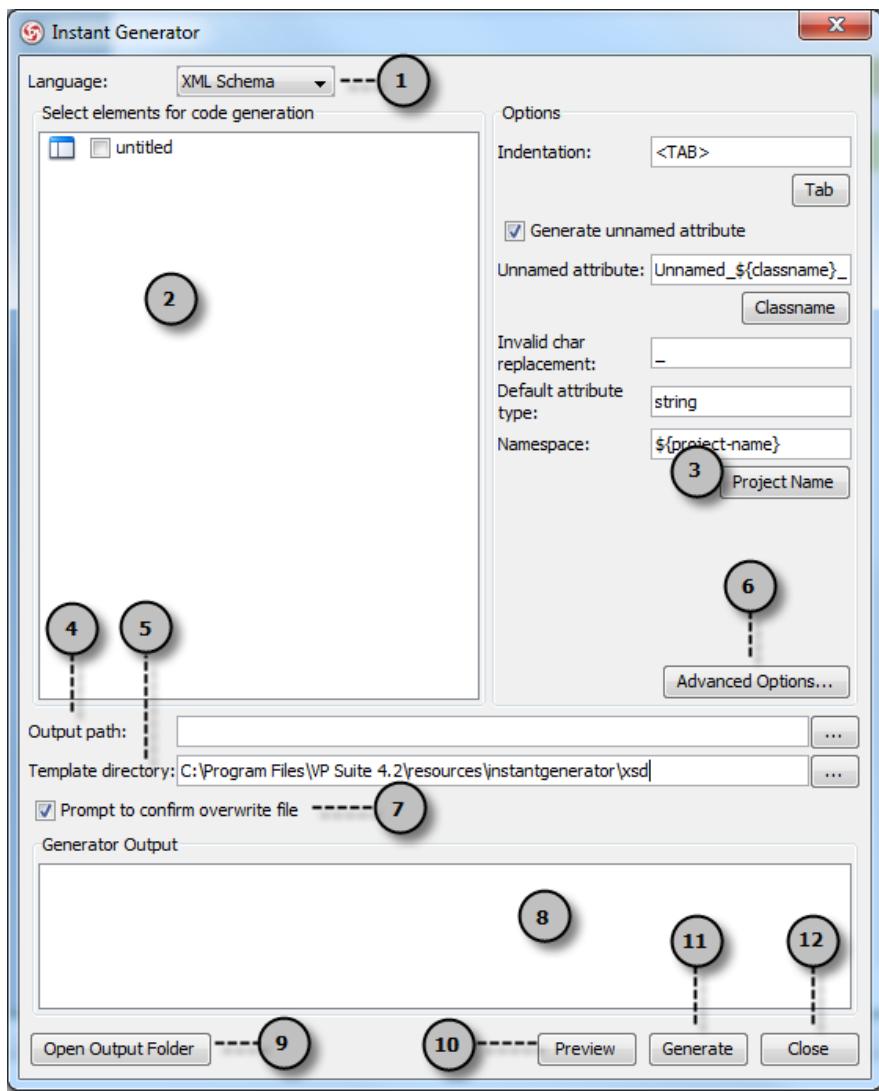
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

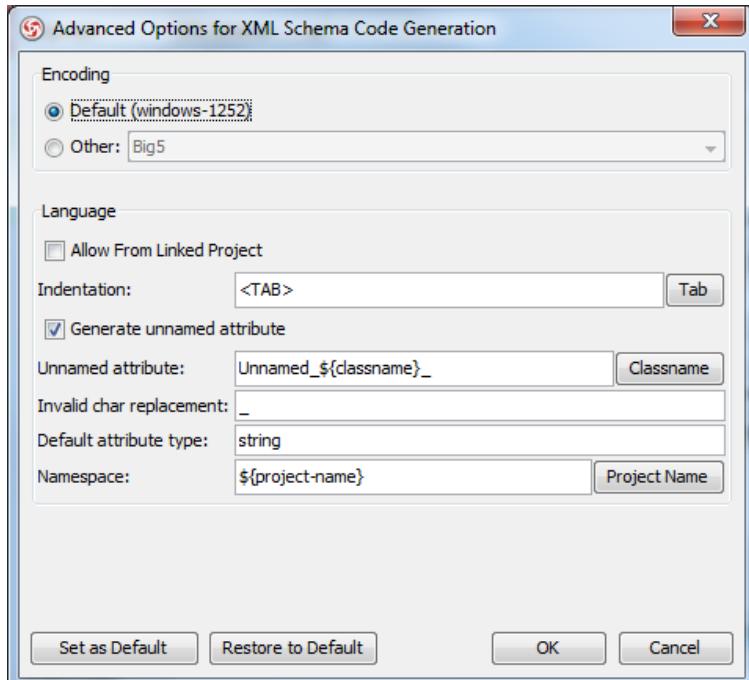


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

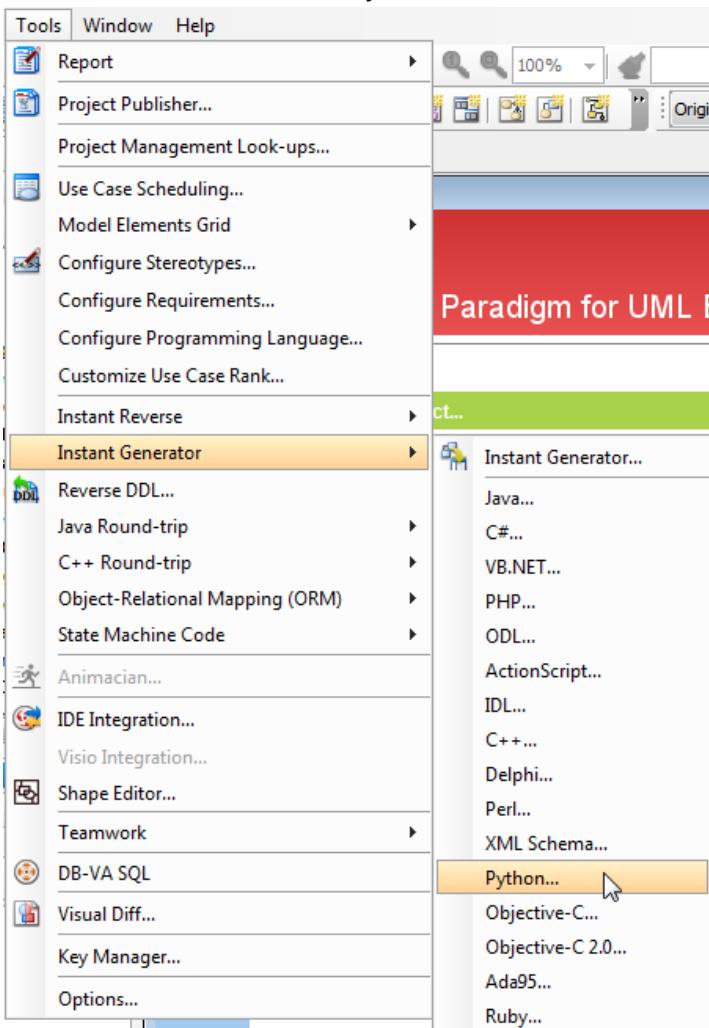
Option	Description
Encoding	The encoding of source file.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used for indentation. Default is Tab .
Generate unnamed attribute	When two classes are associated, checking this option will generate 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

Instant Generator for Python source code

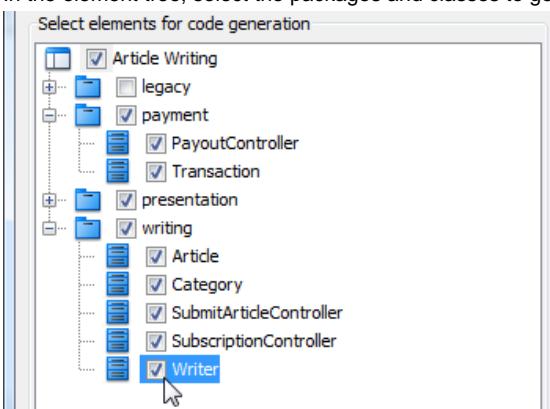
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Python ...** from the main menu.



To open instant generator

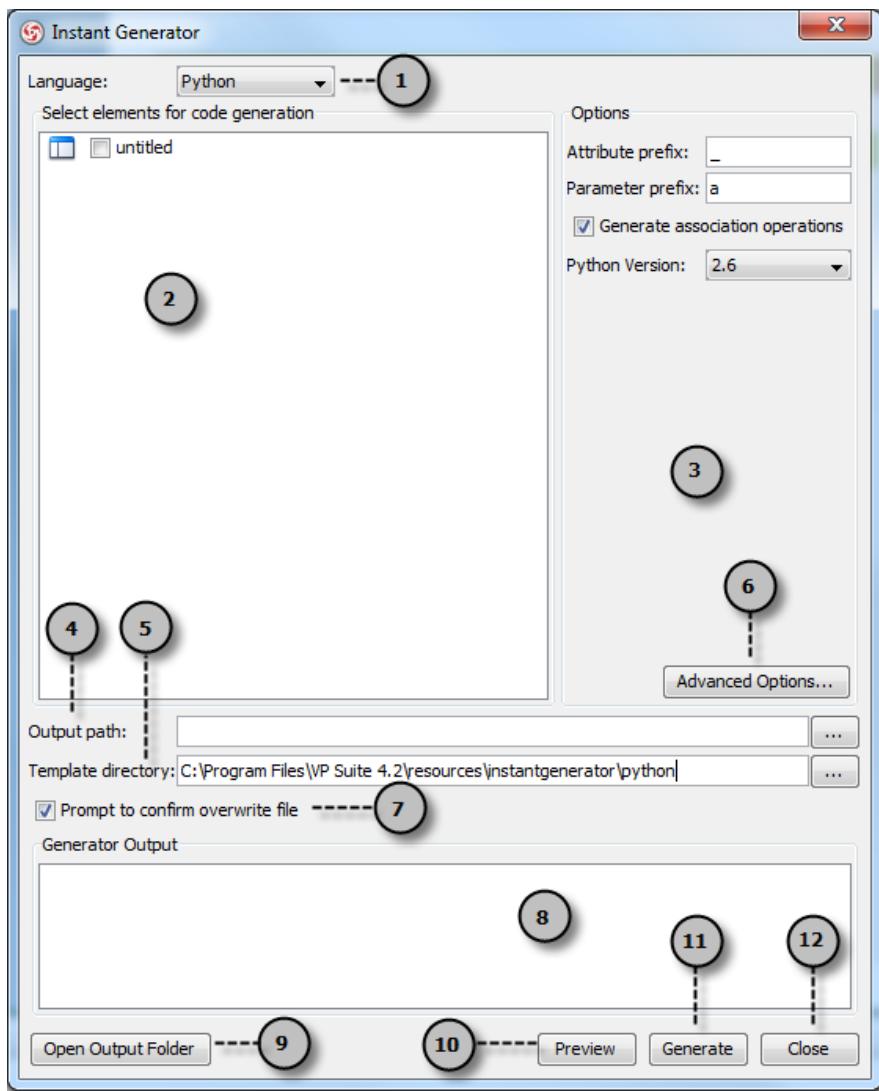
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

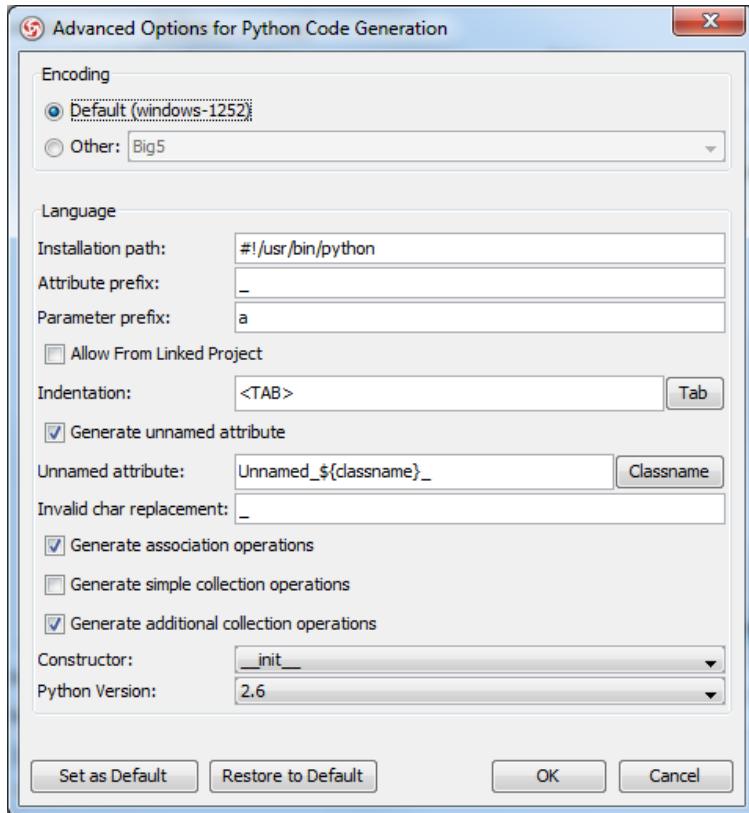


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

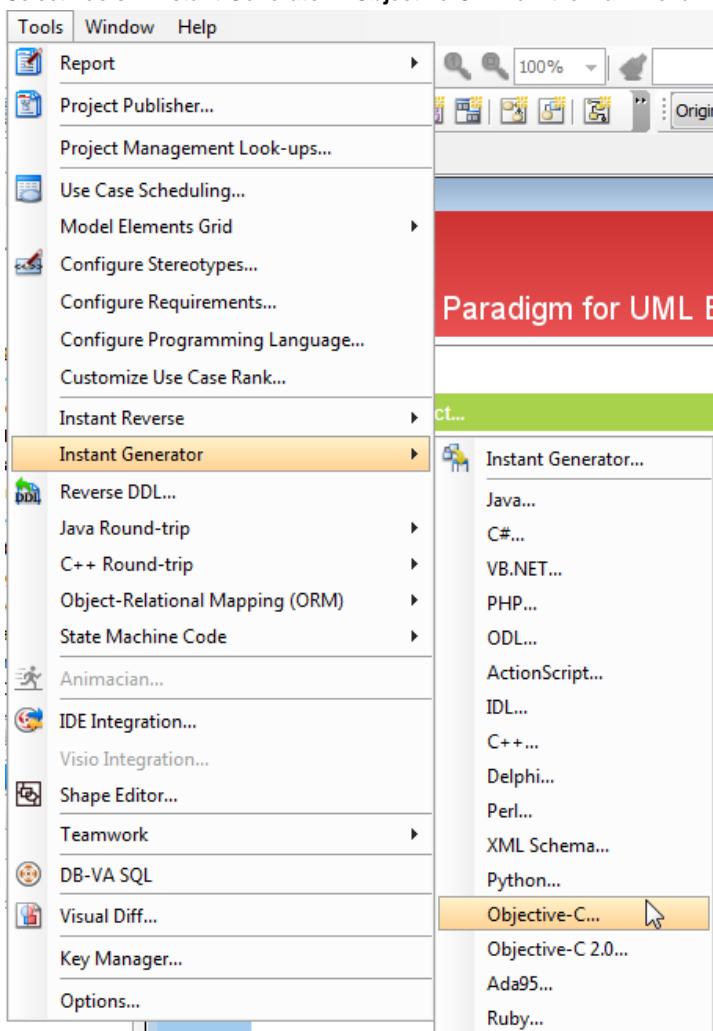
Option	Description
Encoding	The encoding of source file.
Installation path	The installation path of Python.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced 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

Instant Generator for Objective-C source code

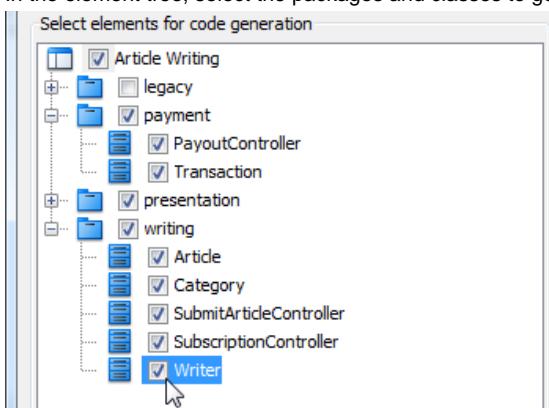
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Objective-C ...** from the main menu.



To open instant generator

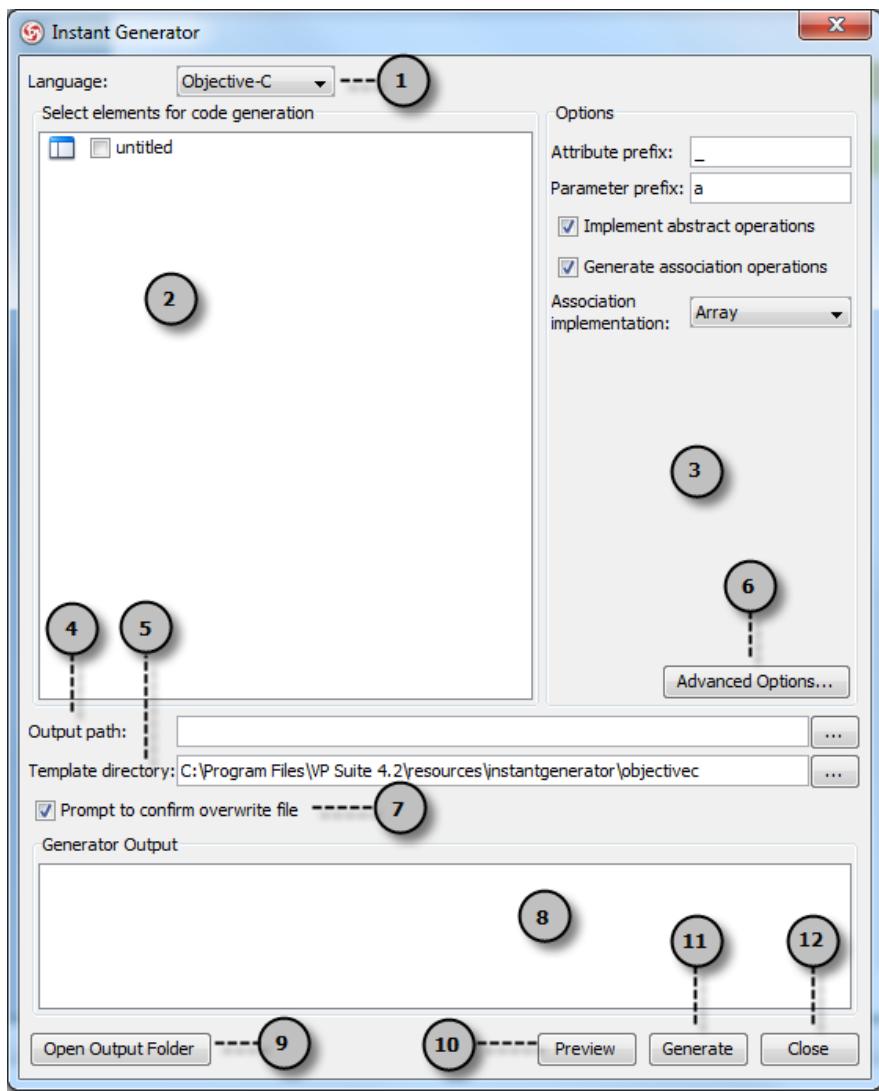
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

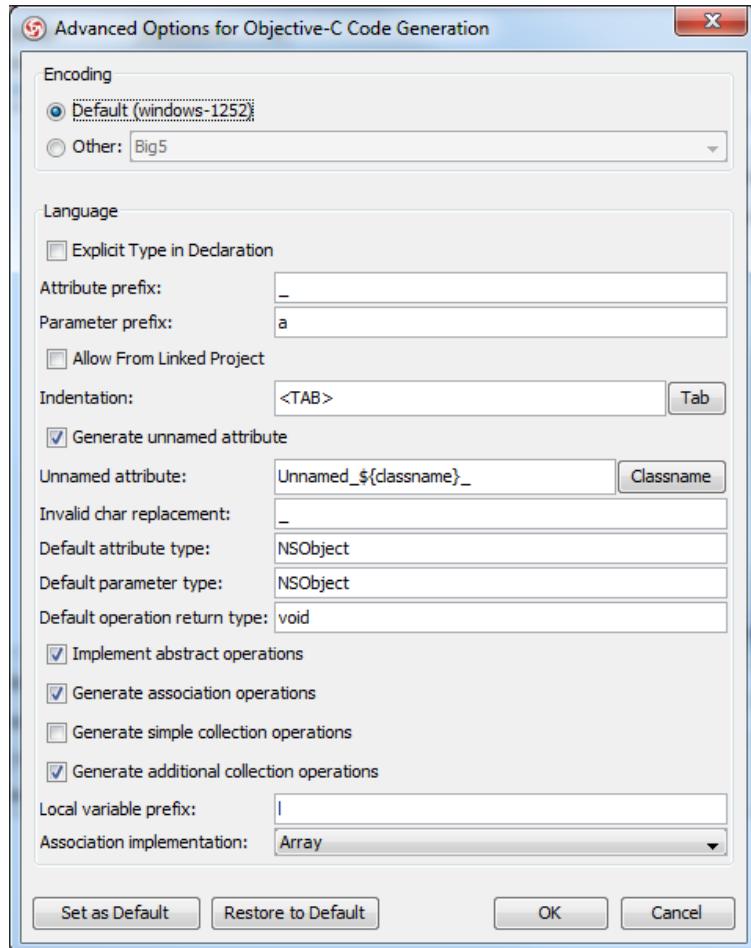


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

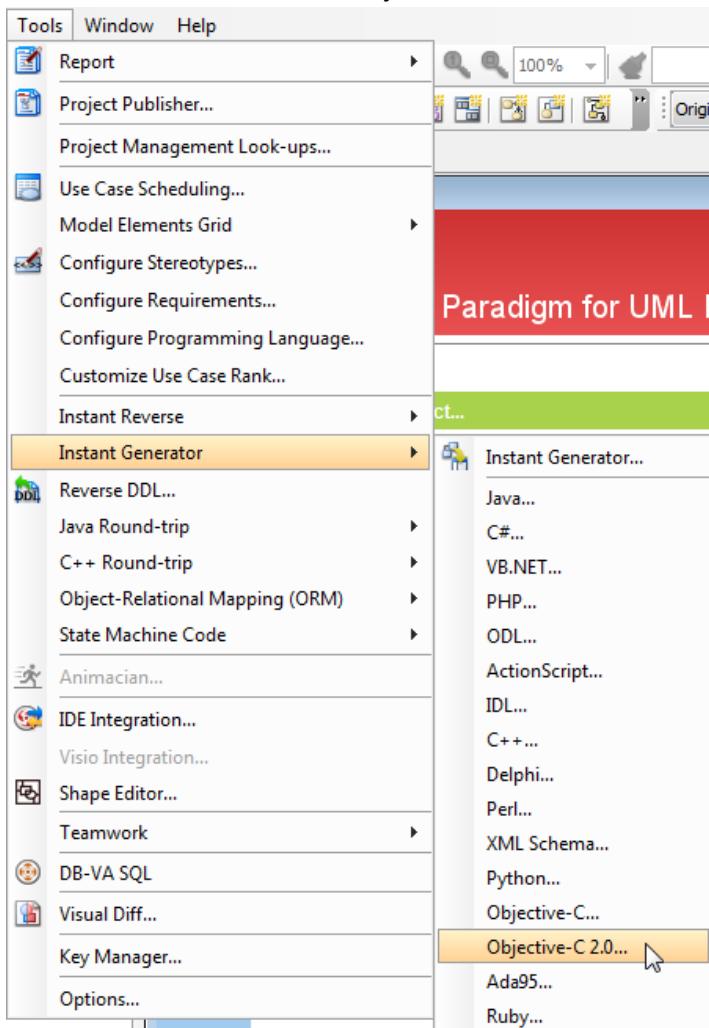
Option	Description
Encoding	The encoding of source file.
Explicit Type in Declaration	When checked, will generate attribute/parameter type with specified type or just id.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
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

Instant Generator for Objective-C 2.0 source code

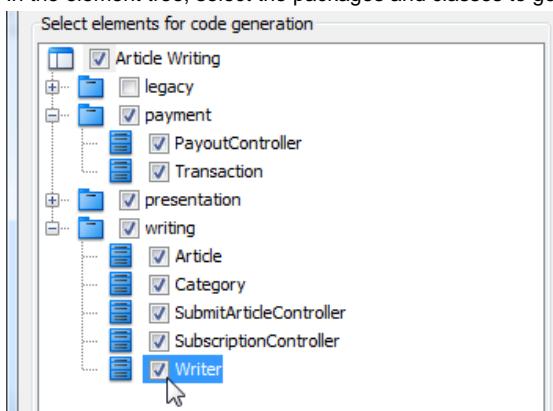
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Objective-C 2.0 ...** from the main menu.



To open instant generator

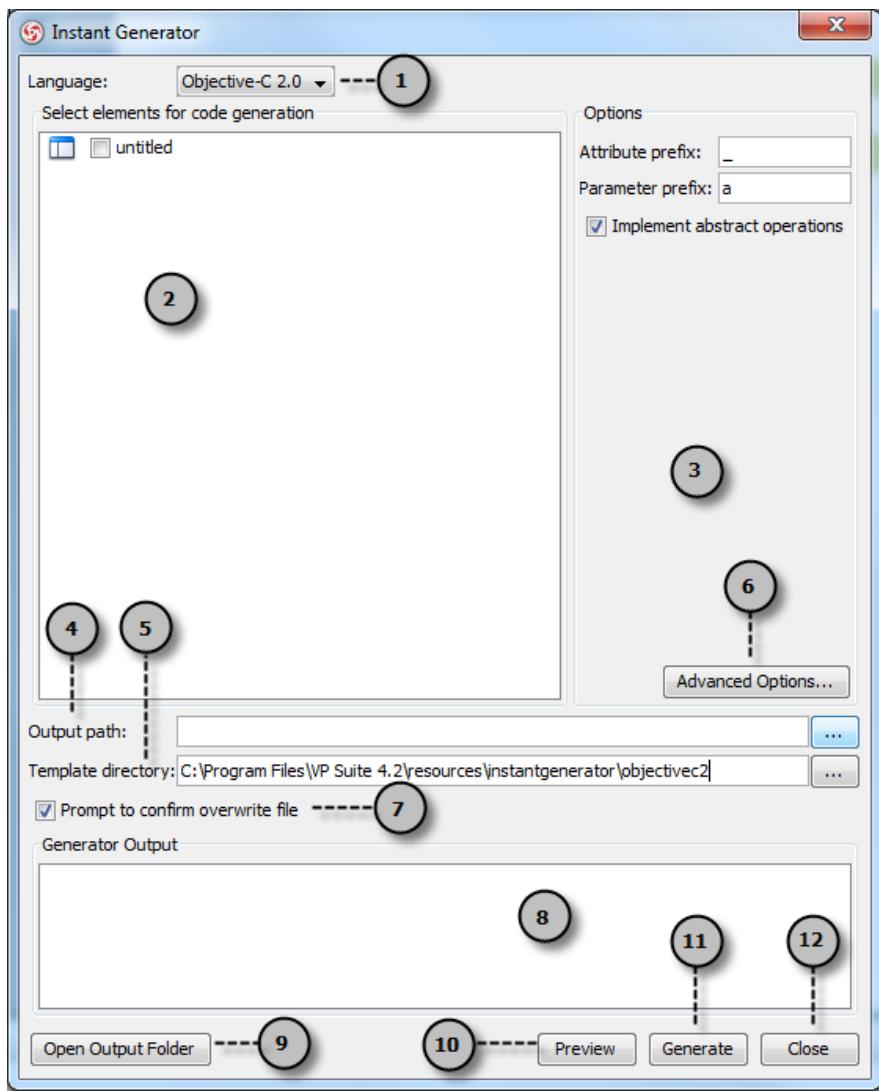
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

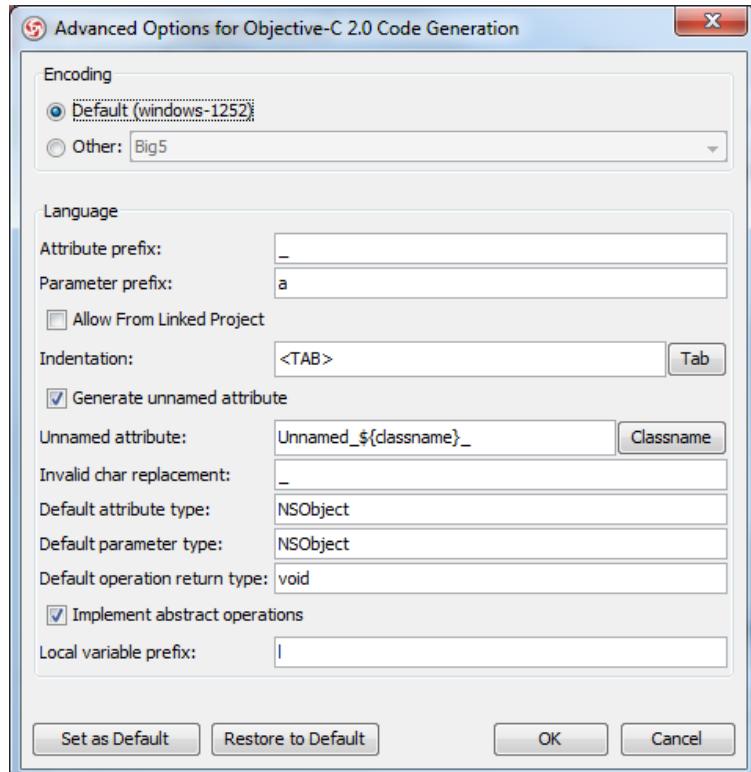


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

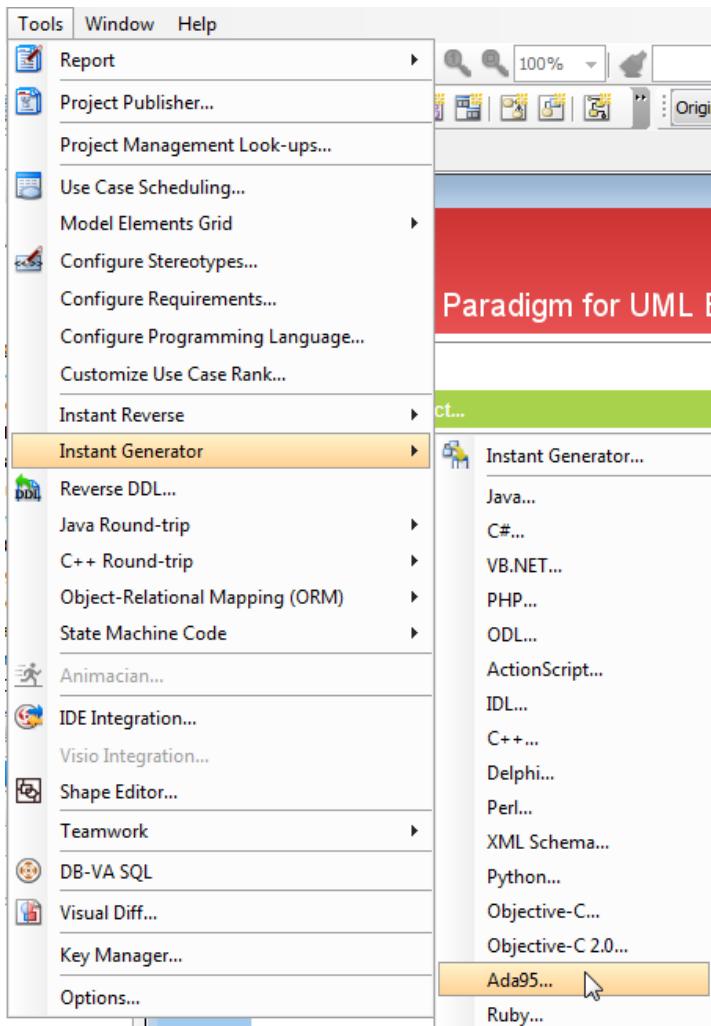
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
Allow From Linked Project	Check to generate also classes in referenced project.
Indentation	Character(s) being used 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.
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

Instant Generator for Ada95

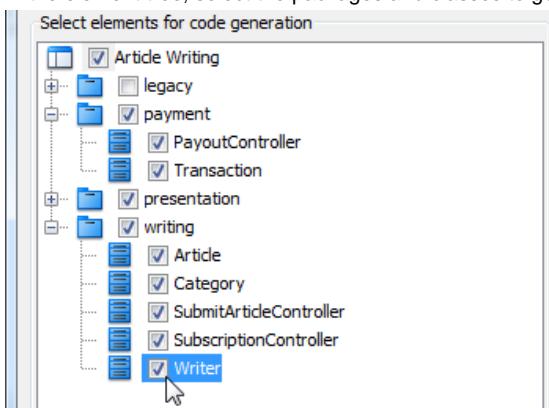
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Ada95 ...** from the main menu.



To open instant generator

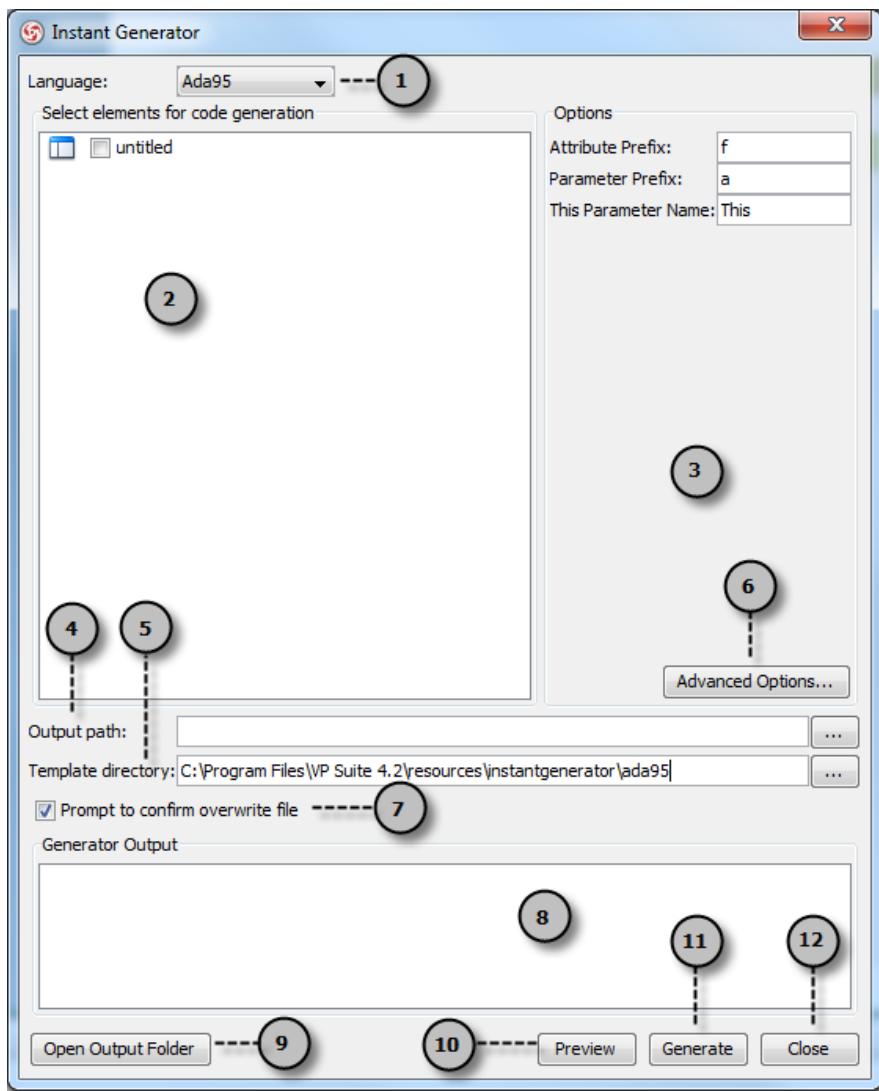
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

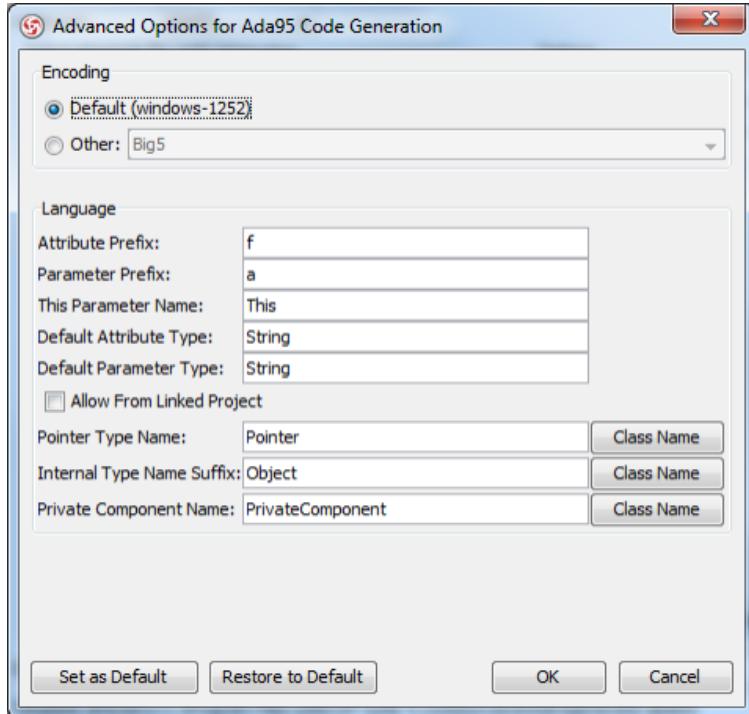


Overview of instant generator dialog box

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 VP-UML 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 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.

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.

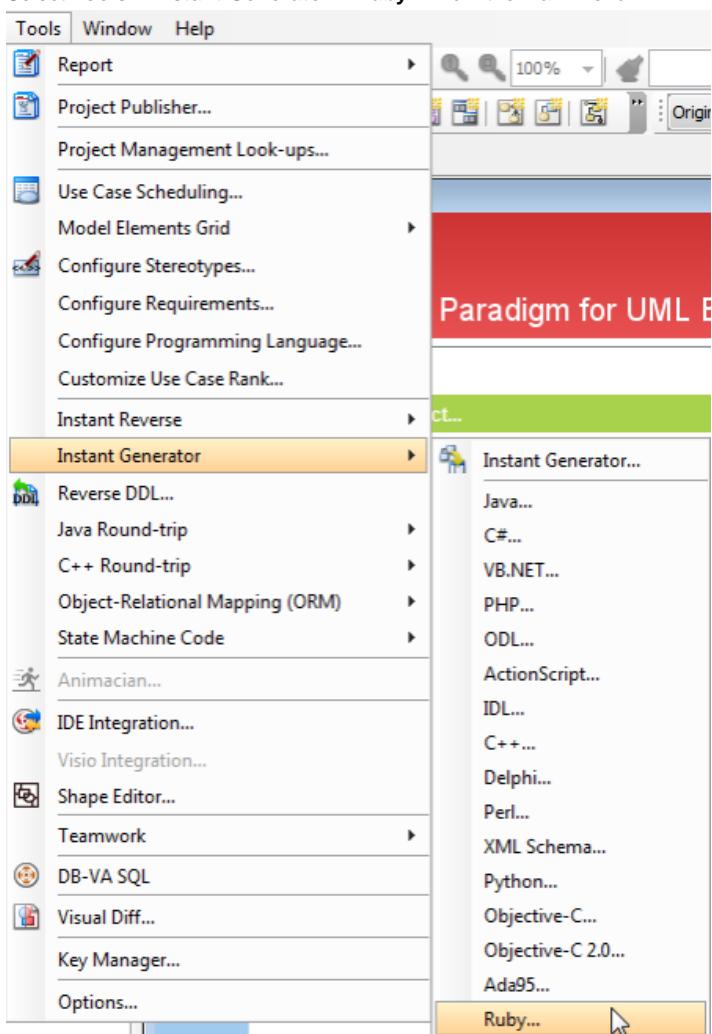
Option	Description
Encoding	The encoding of source file.
Attribute prefix	The text to append to attribute name as prefix.
Parameter prefix	The text to append to parameter name as prefix.
This Parameter Name	The name of the pointer which for accessing object itself.
Default attribute type	Attribute type 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

Instant Generator for Ruby

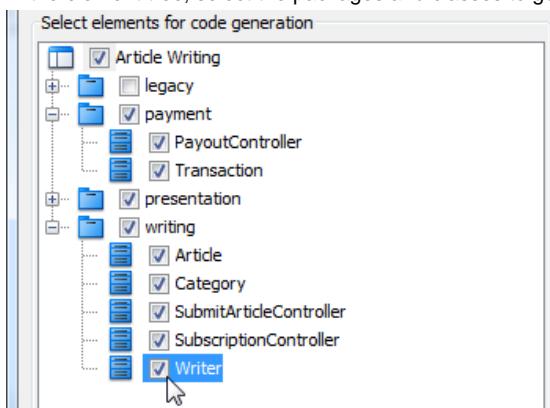
Instant generator is the process of producing source code from class model. Designers or software architects can build a high level domain class model, then pass to programmer to perform more lower-level system or application modeling, and eventually generate source code from implementation model. This 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 > Instant Generator > Ruby ...** from the main menu.



To open instant generator

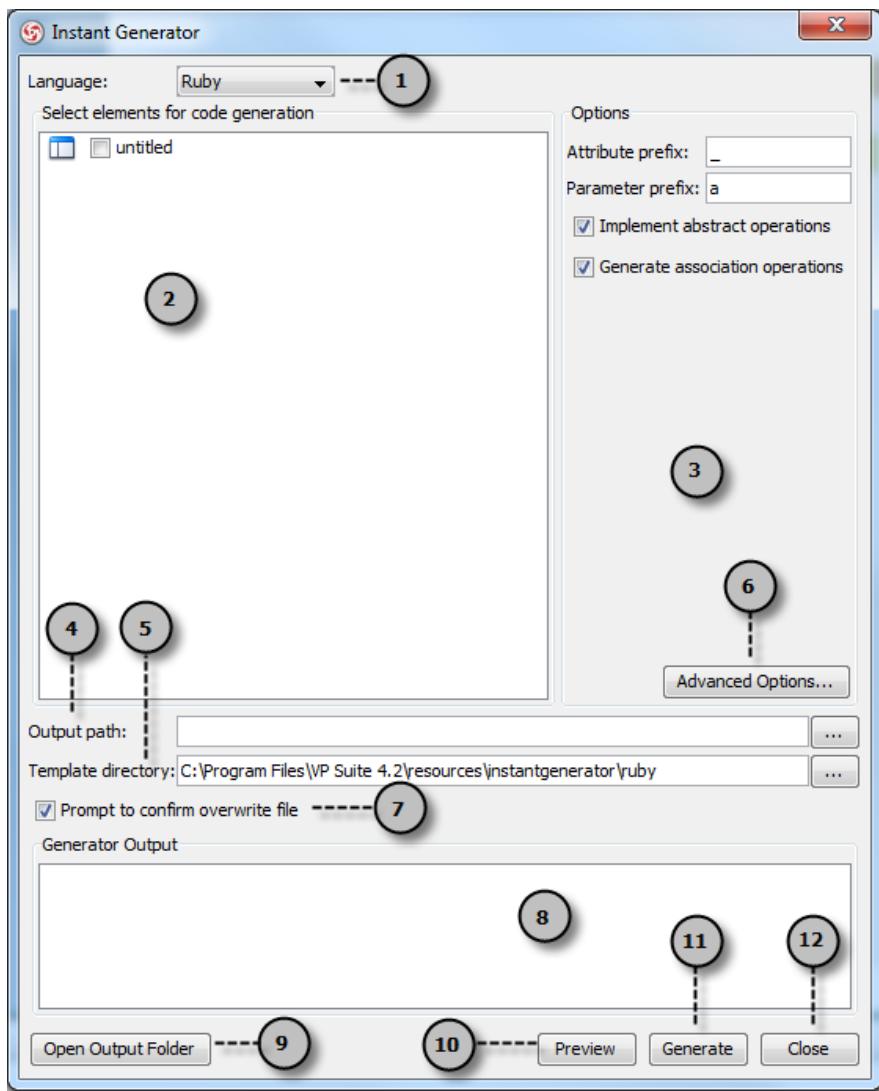
2. In the **Instant Generator** dialog box, fill in the **Path** field, which is the directory where you want the code to generate to.
3. In the element tree, select the packages and classes to generate code.



Select classes to generate code

4. Optionally configure the generator options. Read the section below for a description of options.
5. Click **Generate** to generate code.

Overview of Instant Generator

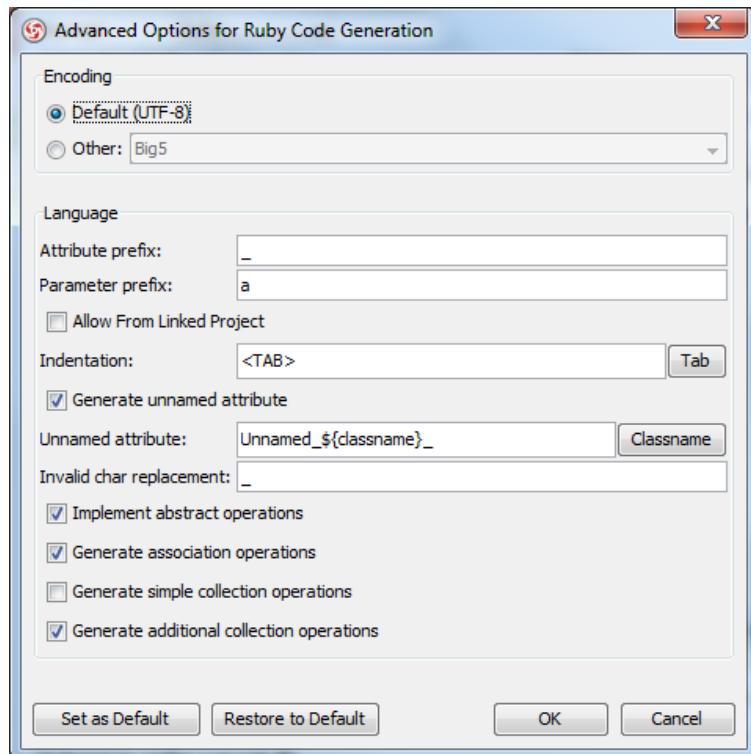


Overview of instant generator dialog box

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 VP-UML 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 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.

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

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.

Preparation

Text editor

The customization of template requires the use of a text editor. A suggestion of text editor would be JEdit, a powerful, yet free of charge text editor. More important, it provides syntax highlighting, which helps you read the template content easier by styling different parts with different colors. You can download JEdit from its official site at:

<http://www.jedit.org/>

To install JEdit:

1. Run the downloaded setup program.
2. Press **Next >** in the Welcome screen.
3. Accept the license agreement and press **Next >**.
4. Select the installation folder and press **Next >**.
5. Select the components to be installed. The editing of template does not require the API documentation, macros and batch files. Depending on your interest, you may decide to install them or not.
6. Select the Start Menu folder and press **Next >**.
7. Select whether to create desktop icons and quick launch icon and press **Next >**.
8. Confirm by pressing **Install**.

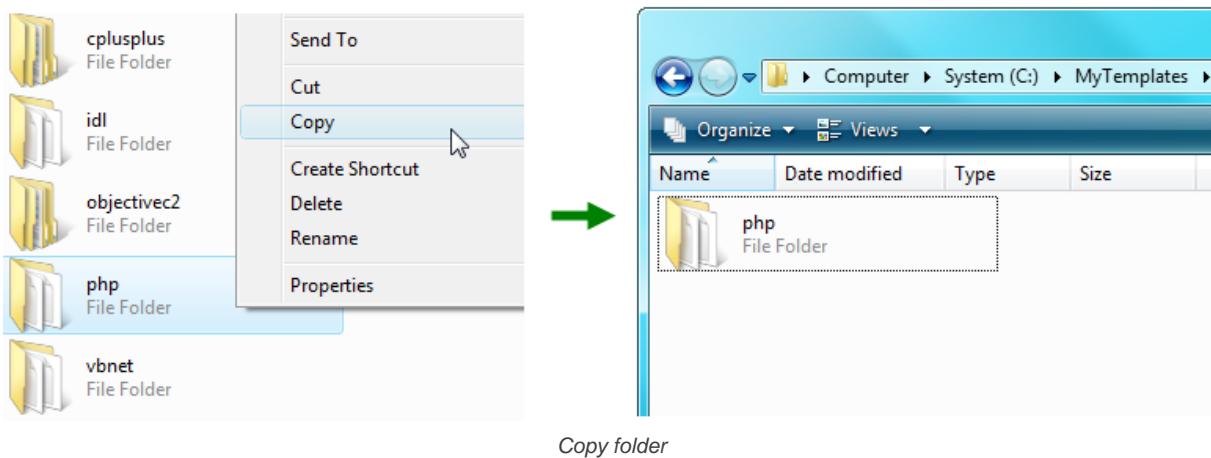
Setting up development environment

The template files are put under the **resources/instantgenerator** folder of VP Suite 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 VP Suite:

- Avoid the unexpected template removal by un-installing the VP Suite.
- Avoid accidental file replacement by running product updates.

To setup your development environment:

1. Create a folder as working directory.
2. Explore **%VP-Suite-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 at the working directory.

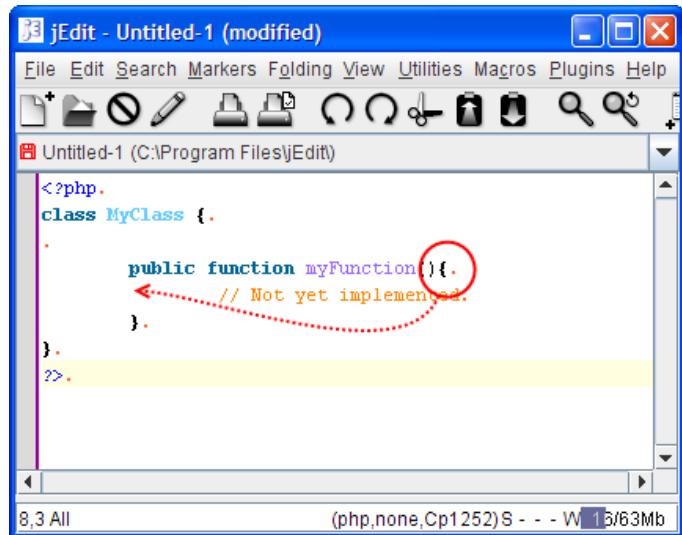


Customizing template

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>.

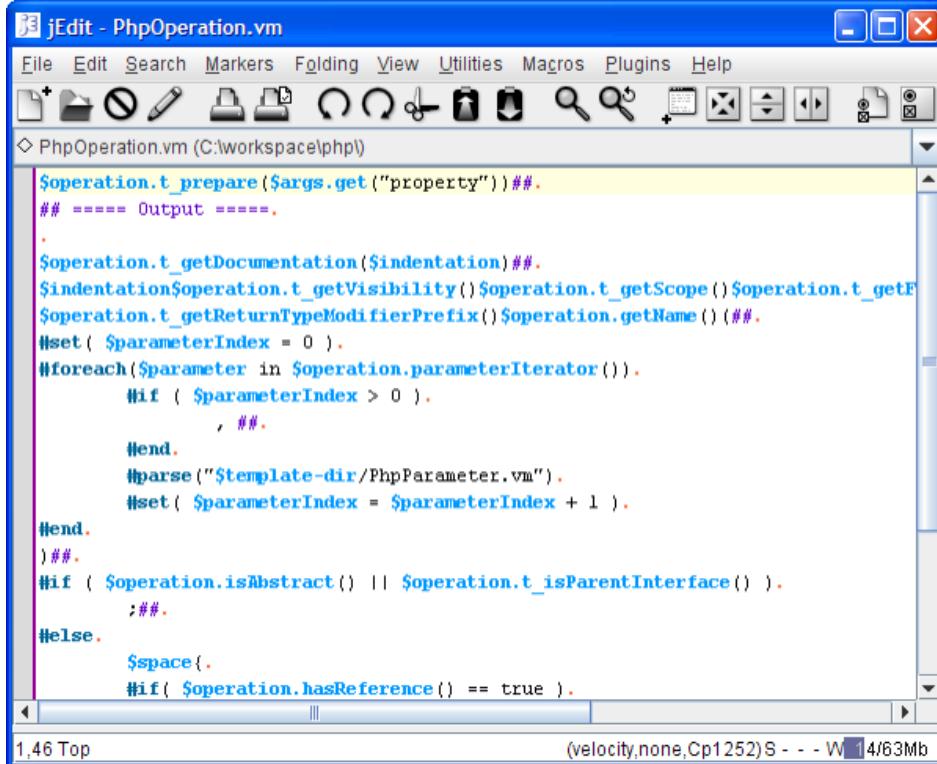
The following example demonstrates how to edit the PHP code generation template to reposition the brace of operation blocks to a new line.



A screenshot of the jEdit text editor. The title bar says "jEdit - Untitled-1 (modified)". The menu bar includes File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, and Help. The toolbar has icons for new file, open file, save, cut, copy, paste, find, and search. The main window shows PHP code:<?php.
class MyClass {
 public function myFunction(){
 // Not yet implemented.
 }
?>The brace for the myFunction() method is circled in red. A dotted line connects the circled brace to the explanatory comment above it.

Customization of operation in PHP class

1. Open the template you need to edit in text editor.



A screenshot of the jEdit text editor. The title bar says "jEdit - PhpOperation.vm". The menu bar includes File, Edit, Search, Markers, Folding, View, Utilities, Macros, Plugins, and Help. The toolbar has icons for new file, open file, save, cut, copy, paste, find, and search. The main window shows Velocity template code:\$operation.t_prepare(\$args.get("property"))##.
===== Output =====.

\$operation.t_getDocumentation(\$indentation)##.
\$indentation\$operation.t_getVisibility()\$operation.t_getScope()\$operation.t_getF
\$operation.t_getReturnTypeModifierPrefix()\$operation.getName()##.
#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).
 ##.

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 understand. 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.

```

jEdit - PhpOperation.vm
File Edit Search Markers Folding View Utilities Macros Plugins Help
+ - < > << >> <<< >>> <<<< >>>> <<<<< >>>>> <<<<<< >>>>>> <<<<<<< >>>>>>>
◊ PhpOperation.vm (C:\workspace\php)
    #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()")
        #else.
            ${indentation}${indentation}// Not yet implemented.
        #end.
        ${indentation}##.
#end.
.
18,9-16 Bot

```

Search for the open branch

3. Make change.

```

jEdit - PhpOperation.vm (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
+ - < > << >> <<< >>> <<<< >>>> <<<<<< >>>>>>
◊ PhpOperation.vm (C:\workspace\php)
    #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()")
        #else.
            ${indentation}${indentation}// Not yet implemented.
        #end.
        ${indentation}##.
#end.
.
19,2-9 Bot

```

Insert line breaks

4. Add a variable \$indentation to indicate the need of printing indentation before the open brace.

```

if ( $parameterIndex > 0 ) .
    ##.
endif.
#parse("$template-dir/PhpParameter.vm").
$set( $parameterIndex = $parameterIndex + 1 ).

)##.

if ( $operation.isAbstract() || $operation.t_isParentInterface() ) .
    ##.

else.
    $space.
    ${Indentation}().
    if( $operation.hasReference() == true ) .
        #parse("$template-dir/$operation.getReference().getRefTemplate()
    else.
        ${indentation}${indentation}// Not yet implemented.
    endif.
    ${indentation}##.

endif.
.
19,16-23 Bot

```

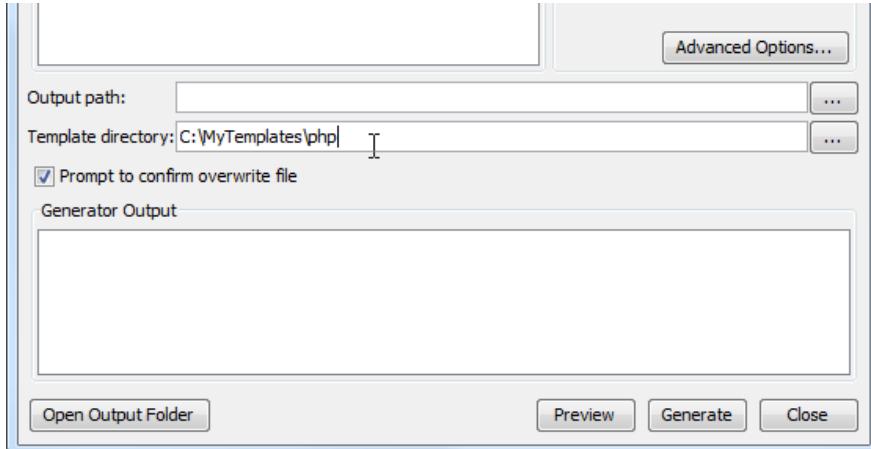
Add variable

5. Save the file.

Generate code with the customized template

To generate code with customized template:

1. In VP-UML, select **Tools > Instant Generator**, 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
	stereotypesIterator()	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
	getDIIName()	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 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=====
${...}
```

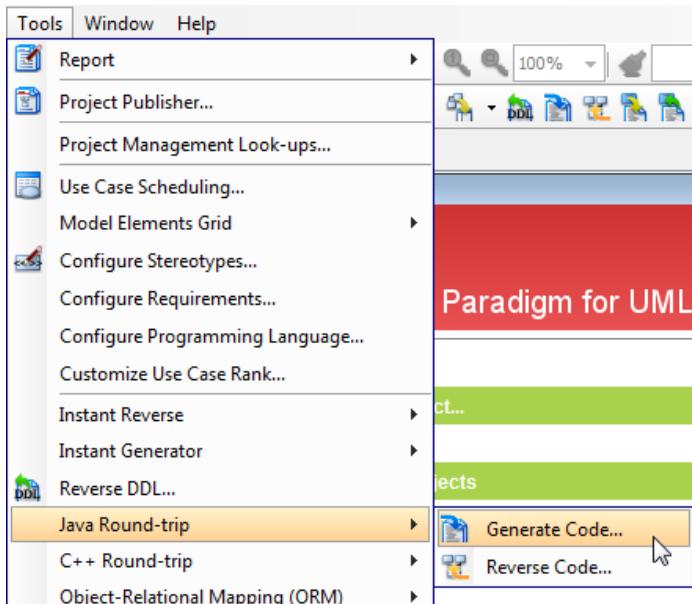
Generate/Update Java code

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 documentation on your model.

Generating/Updating code from whole project

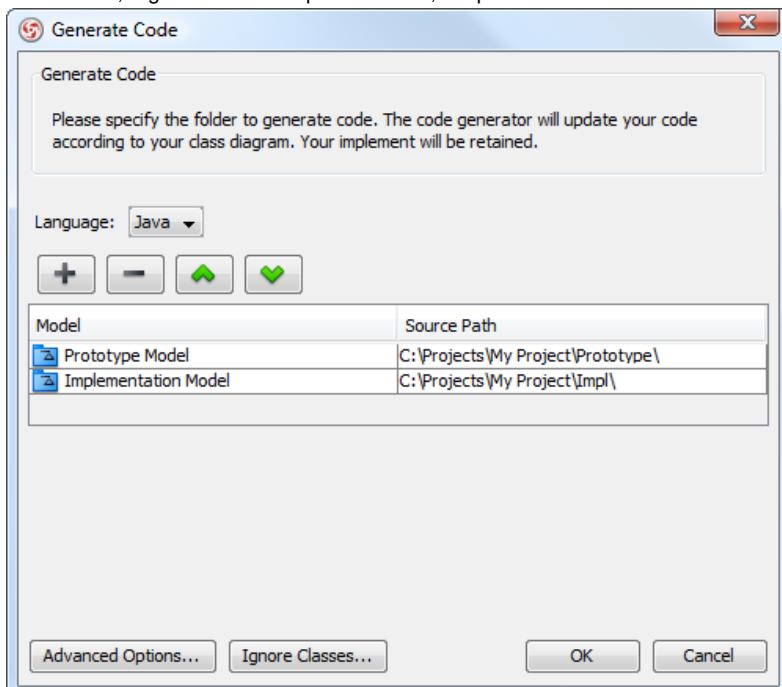
You can generate Java code from all classes in current project. To generate code from project:

1. Select **Tools > Java Round-trip > Generate Code...** from the main menu.



To generate code for a project

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

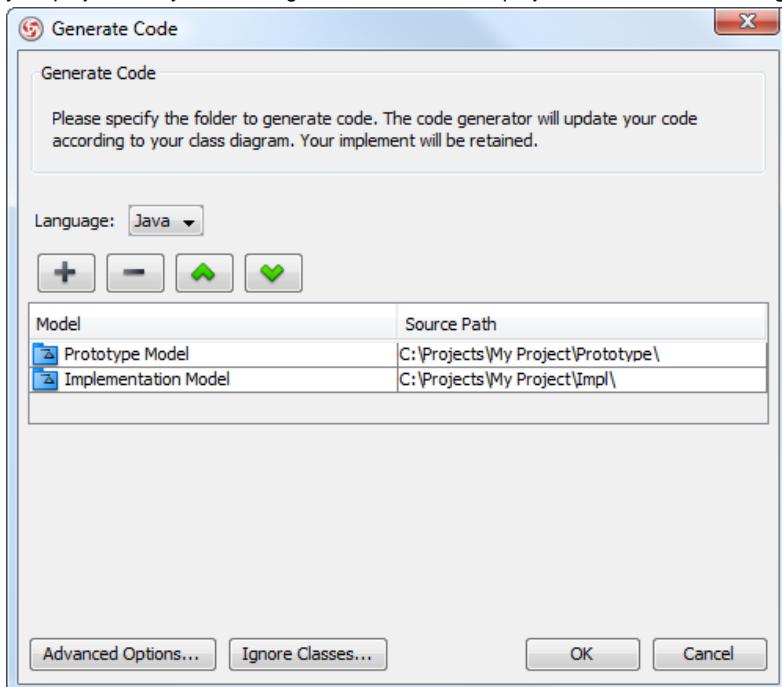
3. Optionally configure the advanced code generation options by clicking **Advanced Options....**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Documentation 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 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).

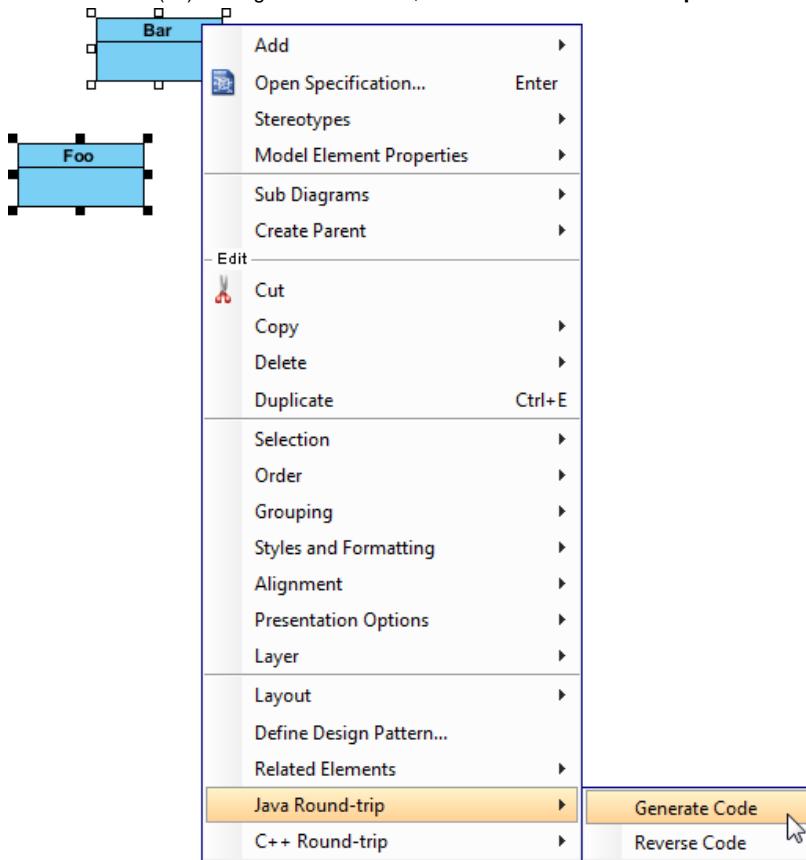
3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Documentation in model elements is generated as comment in code

Generating/Updating code from chosen classes

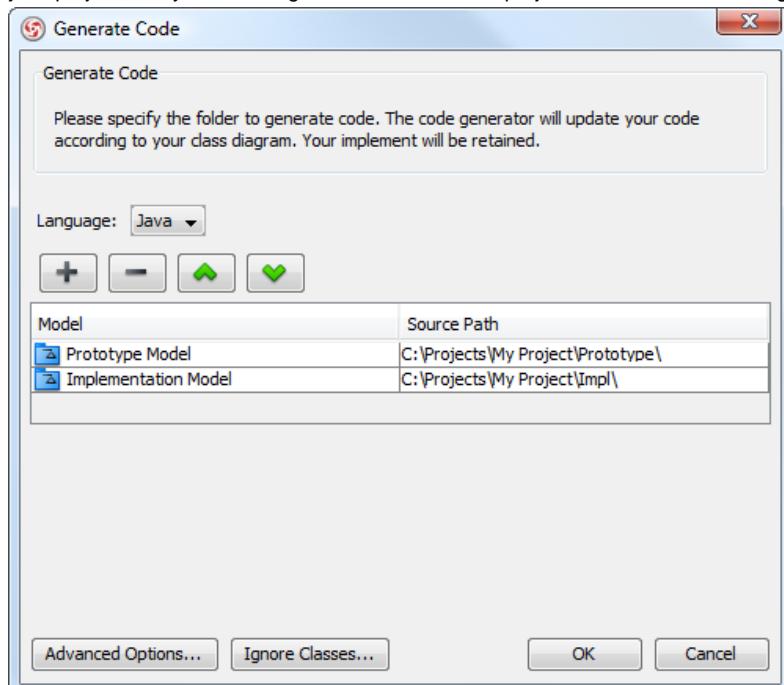
You can generate Java code from specific class or classes. To generate code from class/classes:

- Select the class(es) and right click on them, then select **Java Round-trip > Generate Code** from the popup menu.



To generate code for classes

- 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>**.



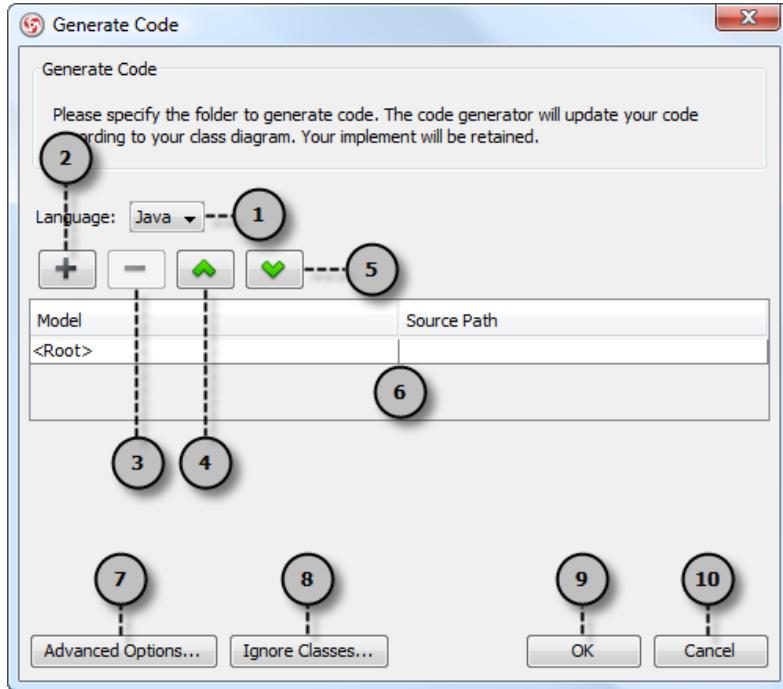
The mappings between models and source paths are defined

NOTE: If you have generated code for once, the **Generate Code** dialog box will not appear next time 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: Documentation in model elements is generated as comment in code

An overview of Generate Code dialog box



An overview of **Generate Code** dialog box

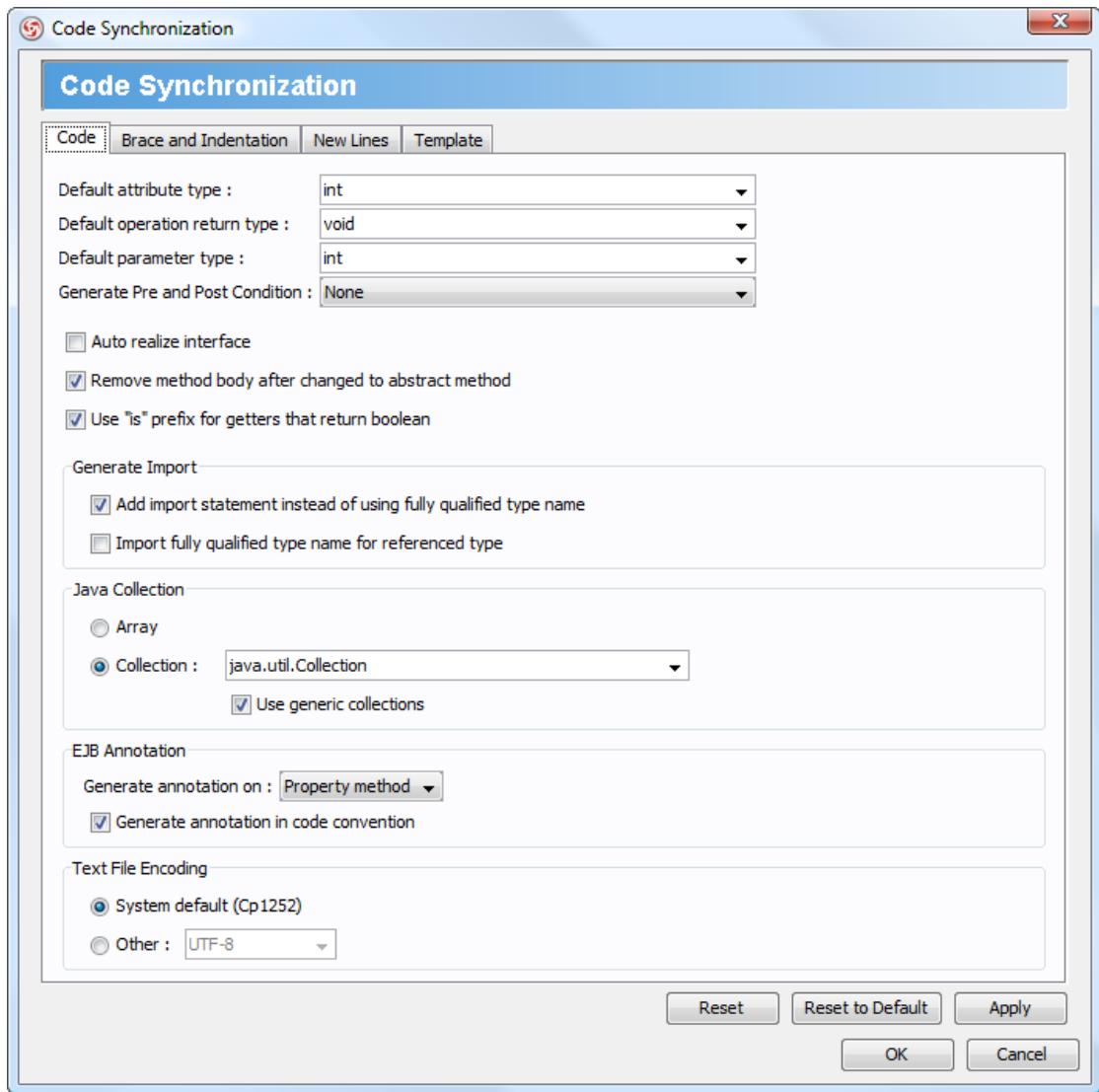
No.	Name	Description
1	Language	The programming language of the source code to generate.
2	Add model-to-source-path mapping	Click to add a new mapping between UML model and the source path where code will be generated to.
3	Remove model-to-source-path mapping	Click to remove chosen model-to-source-path 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 <i>Advanced Options</i> in this chapter.
8	Ignore classes	Click to organize the ignore list of classes to ignore in code generation. For details, read the section <i>To ignore classes in generation</i> 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

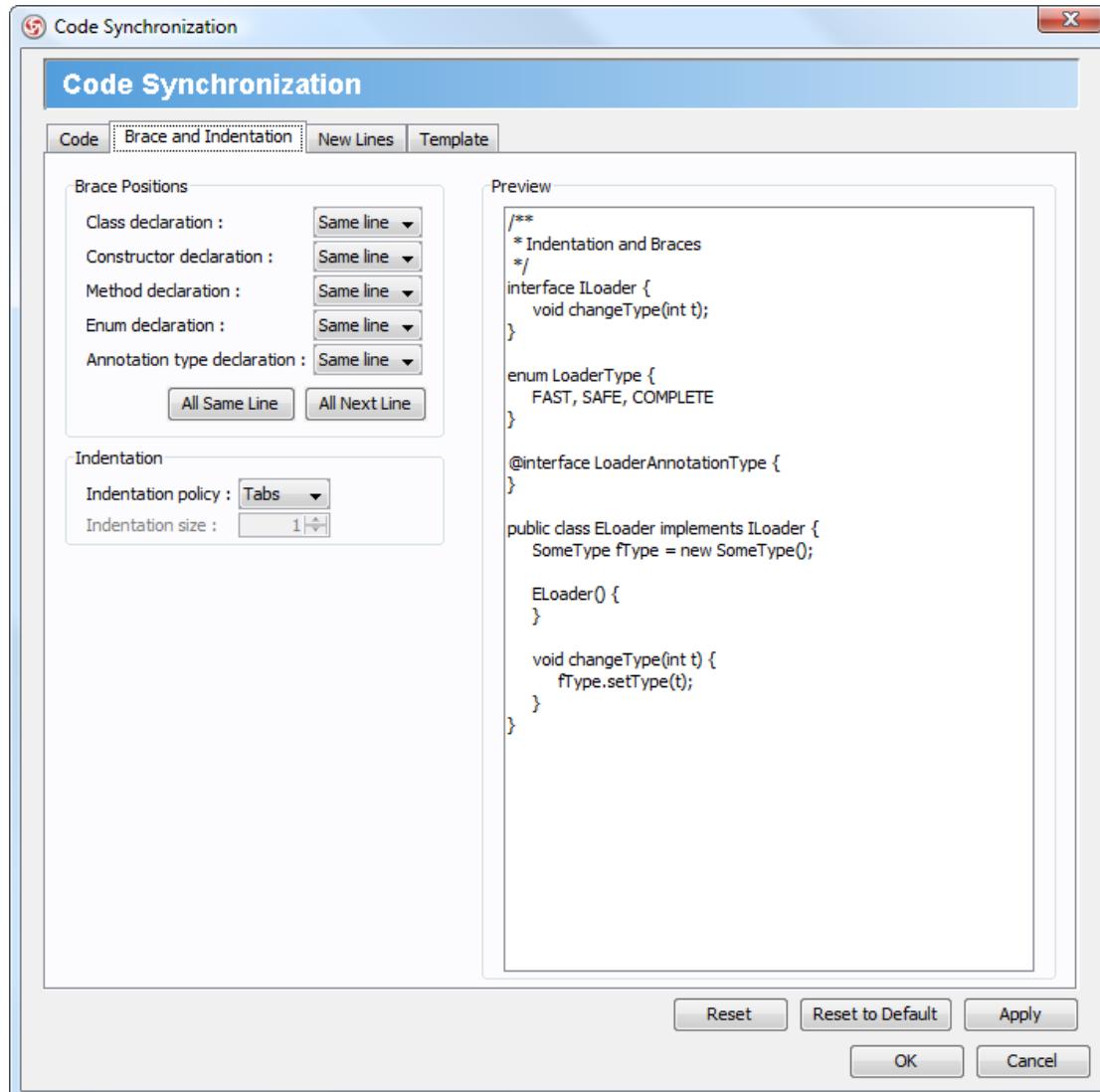
Option	Description
Default attribute type	(default int) Type that will be assigned to Attribute upon code generation when type is unspecified
Default operation return type	(default void) Return Type that will be assigned to operation upon code generation when return type is unspecified
Default parameter 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

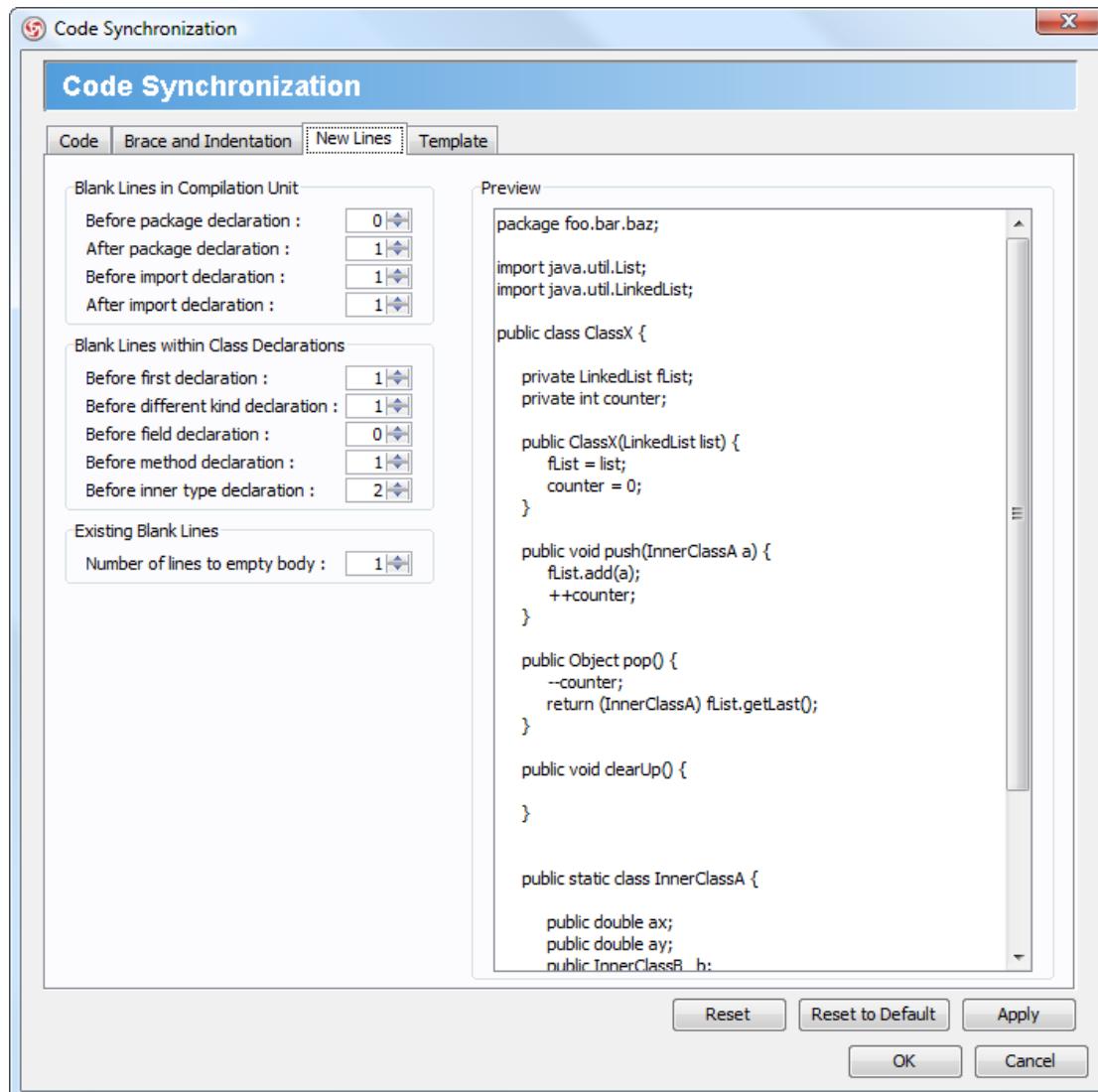
Option	Description
Class declaration	<ul style="list-style-type: none"> • Same line - (default) Brace for class declaration appear at the same line as the declaration • Next line - Brace for class declaration appear at the line after the declaration
Constructor declaration	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Same line - (default) Brace for enumeration appear at the same line as the declaration • Next line - Brace for enumeration to appear at the line after the declaration
Annotation type declaration	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

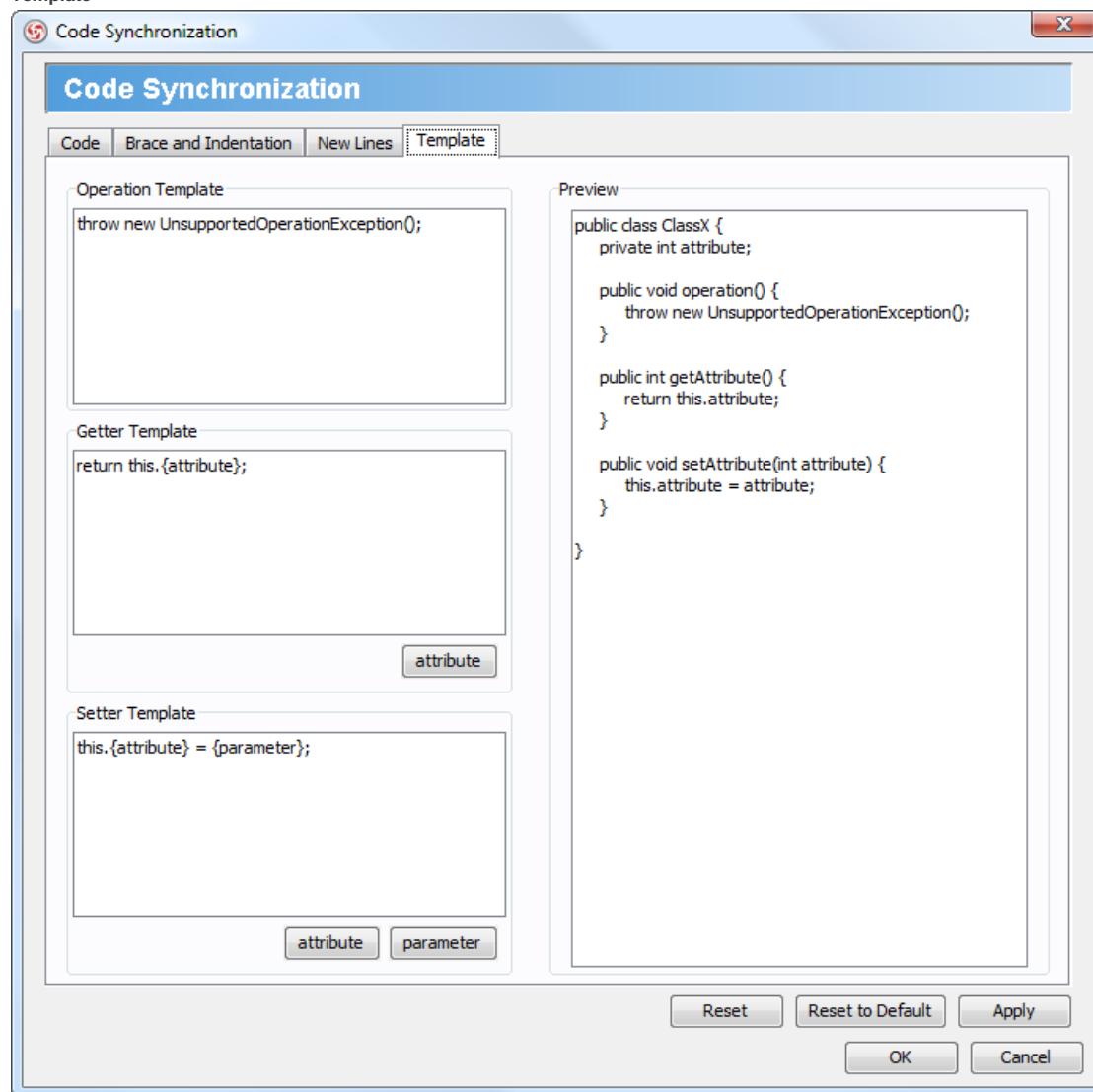
New Lines



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

Template



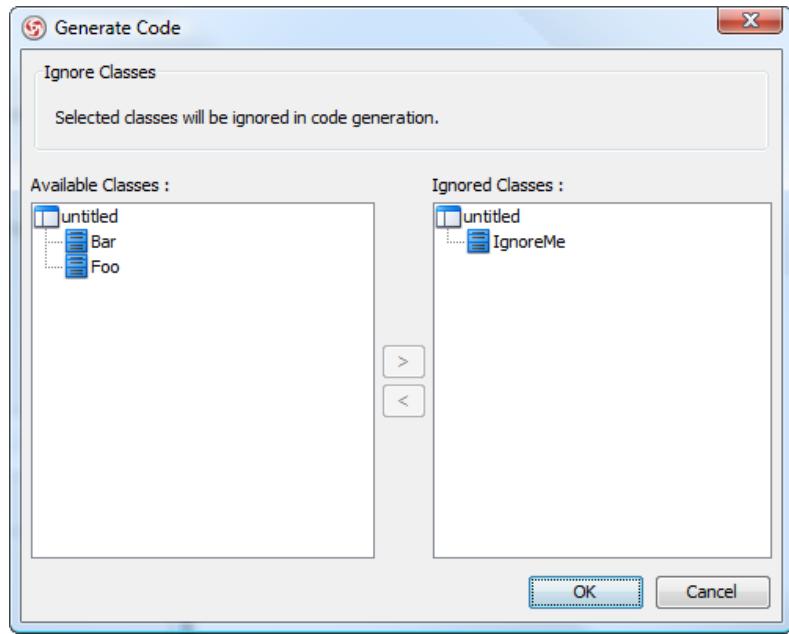
Template configuration

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

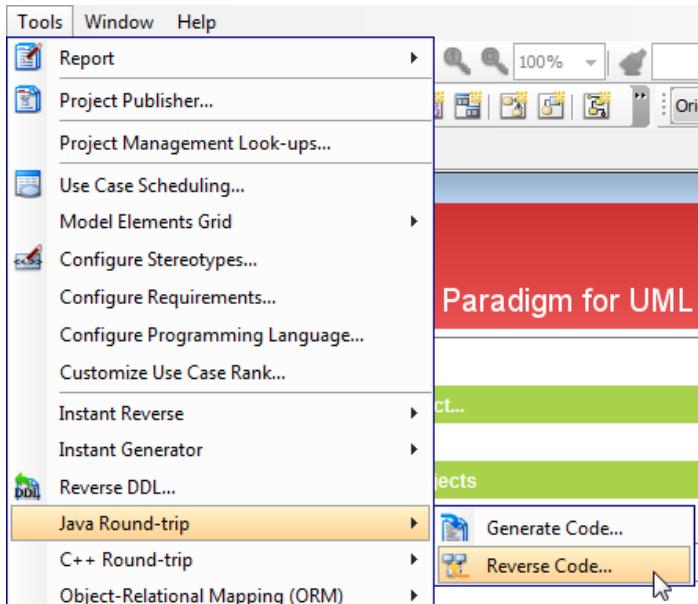
Generate/Update UML classes from Java code

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 documentation on your model.

Generate/Update UML classes from code

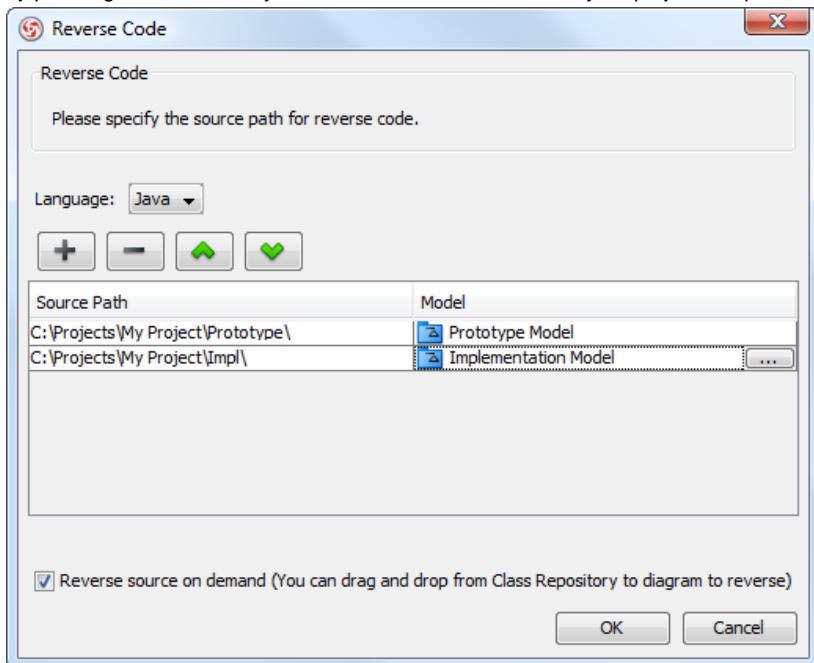
You can produce UML classes from source code, or to update from code all the reversed UML classes in project. To do this:

1. Select **Tools > Java Round-trip > Reverse Code...** from the main menu.



To reverse code to UML classes

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 phases 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>**.

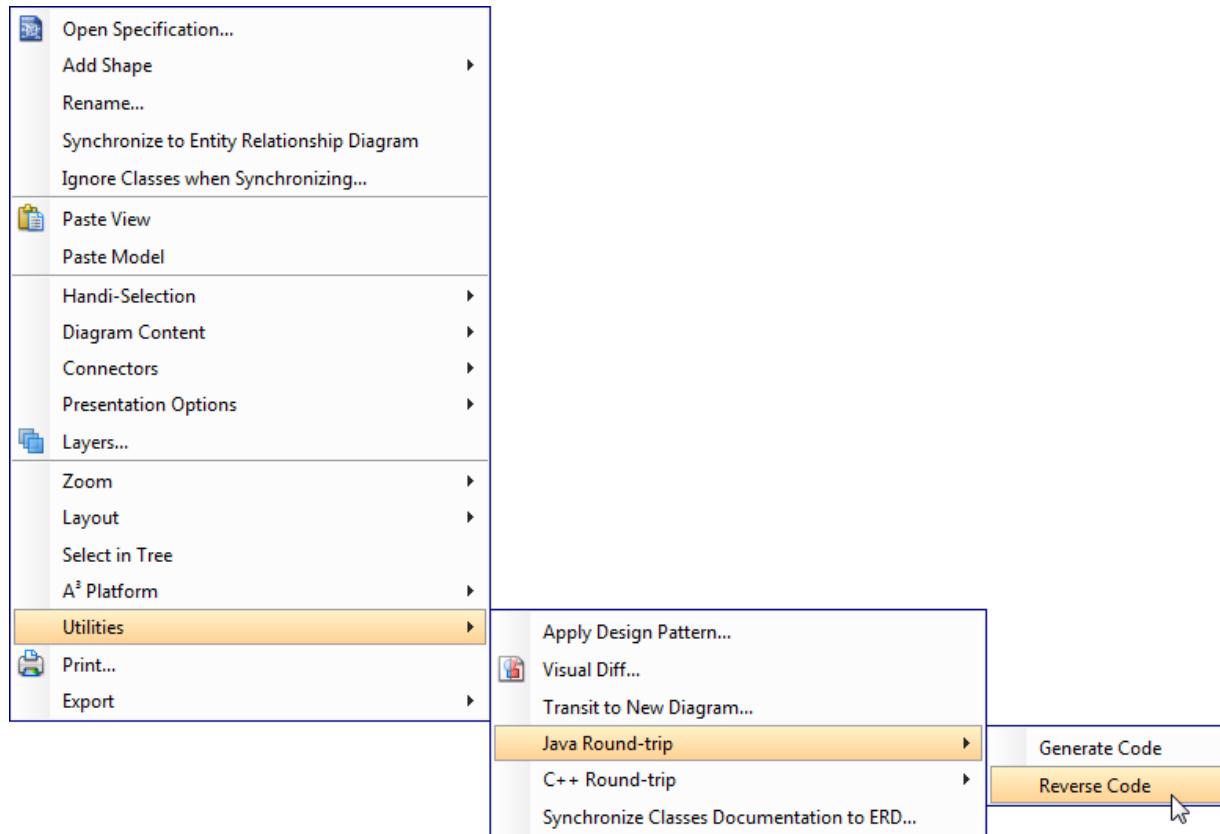


The mappings between source paths and model are defined

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.

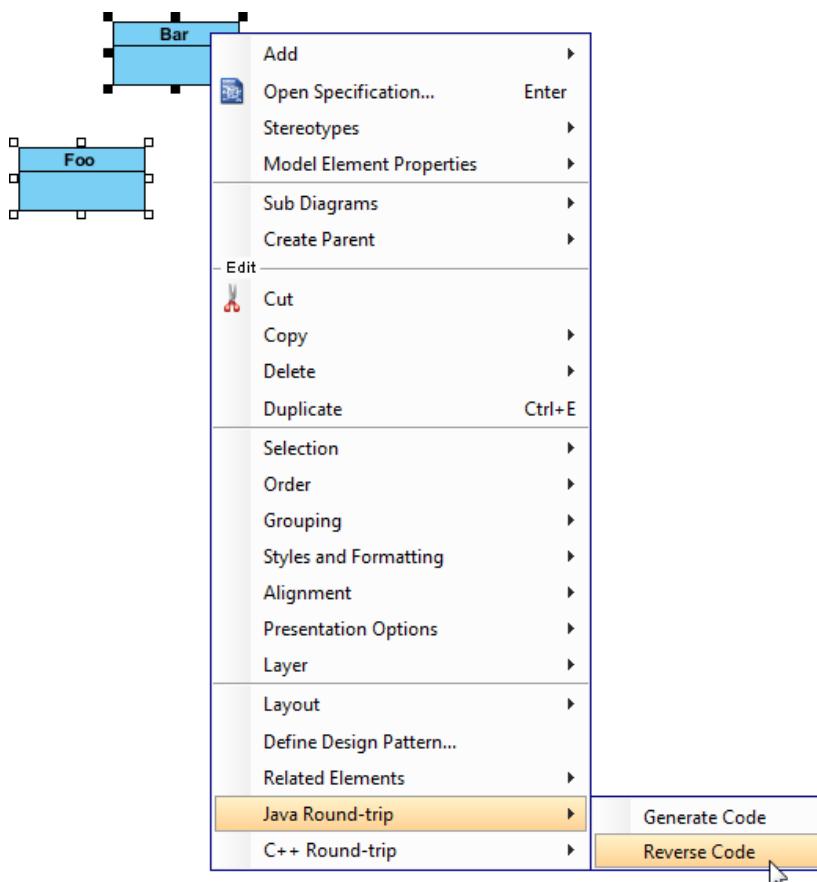


To update UML classes in a diagram from code

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.

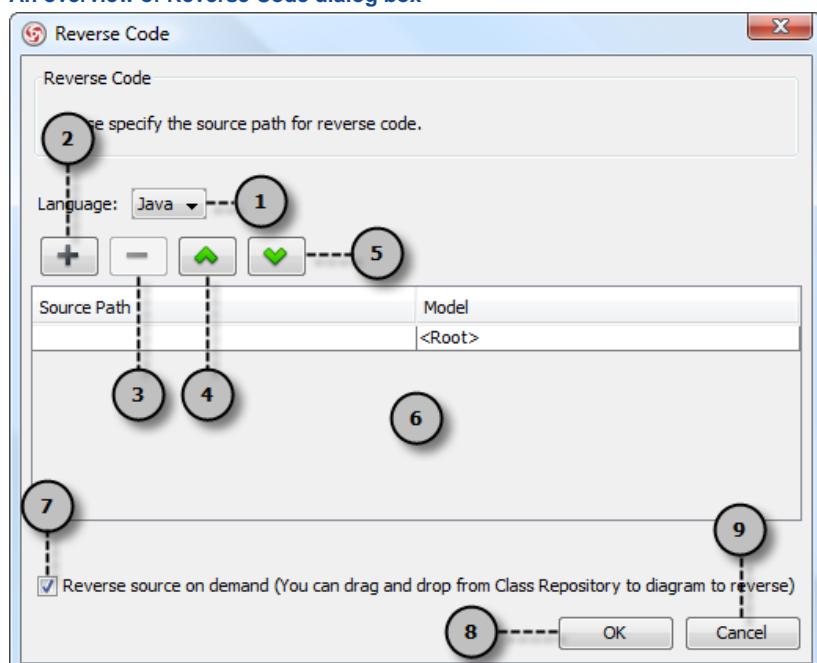
Updating specific UML classes from code

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.



To update specific UML class from code

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.

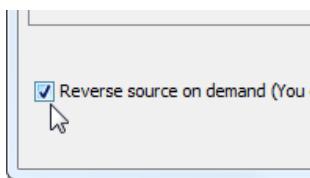
2 Add source-path-to-model mapping	Click to add a new mapping between source path where code will be reversed from and UML model.
3 Remove source-path-to-model mapping	Click to remove chosen source-path-to-model mapping.
4 Move source-path-to-model mapping up	Click to move chosen source-path-to-model 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

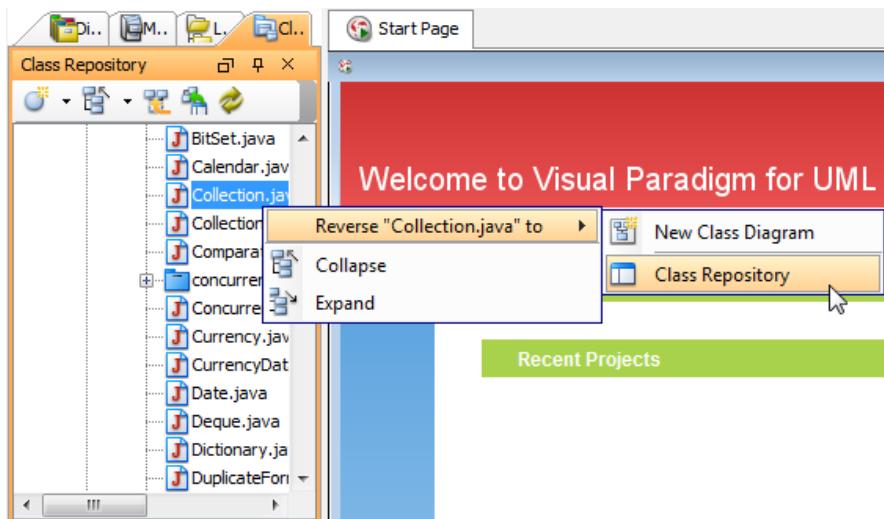
Consider if you have a project that contains million of Java source file, and now you want to re-develop just a few classes in it. If you try to reverse the whole project it will take you a long time to complete the reverse due to the amount of classes (and relationships) are just too many. With on-demand 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.



Reversing a java source file from index tree

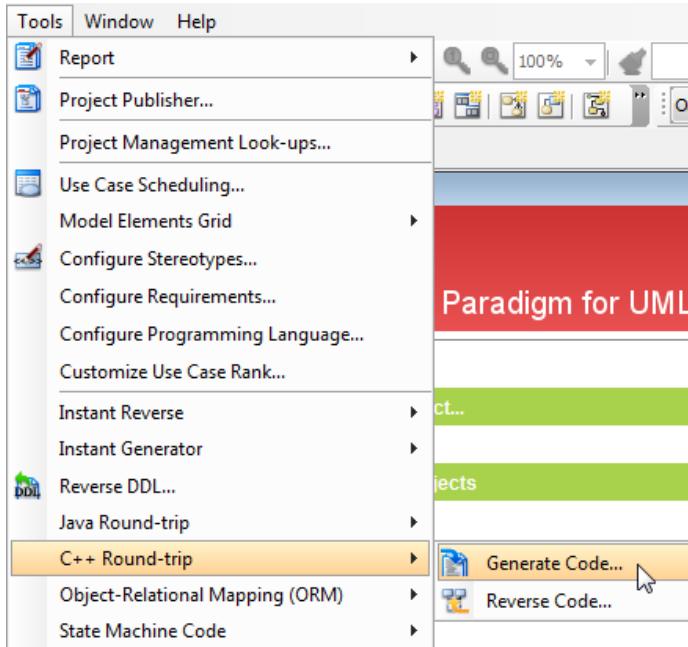
Generate/Update C++ code

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 documentation on your model.

Generating/Updating code from whole project

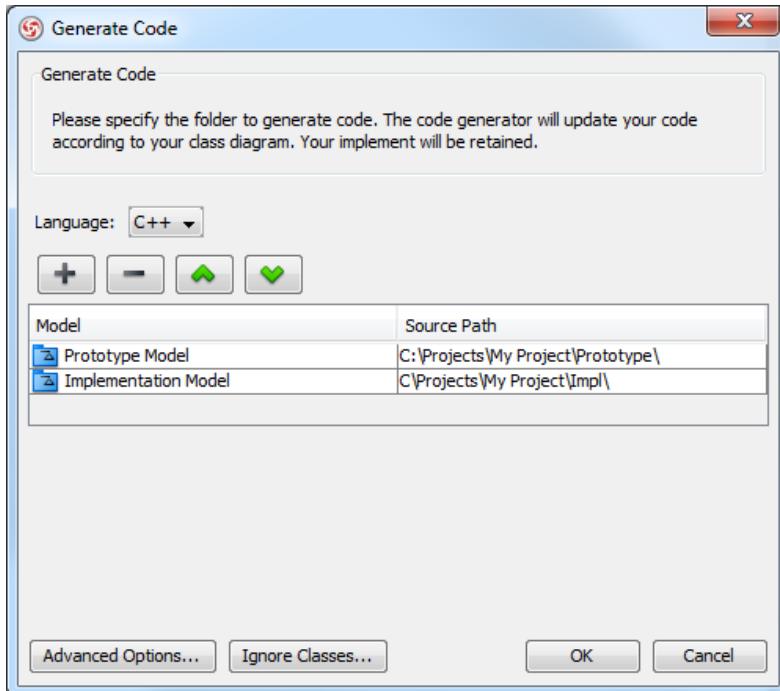
You can generate C++ code from all classes in current project. To generate code from project:

1. Select **Tools > C++ Round-trip > Generate Code...** from the main menu.



To generate code for a project

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

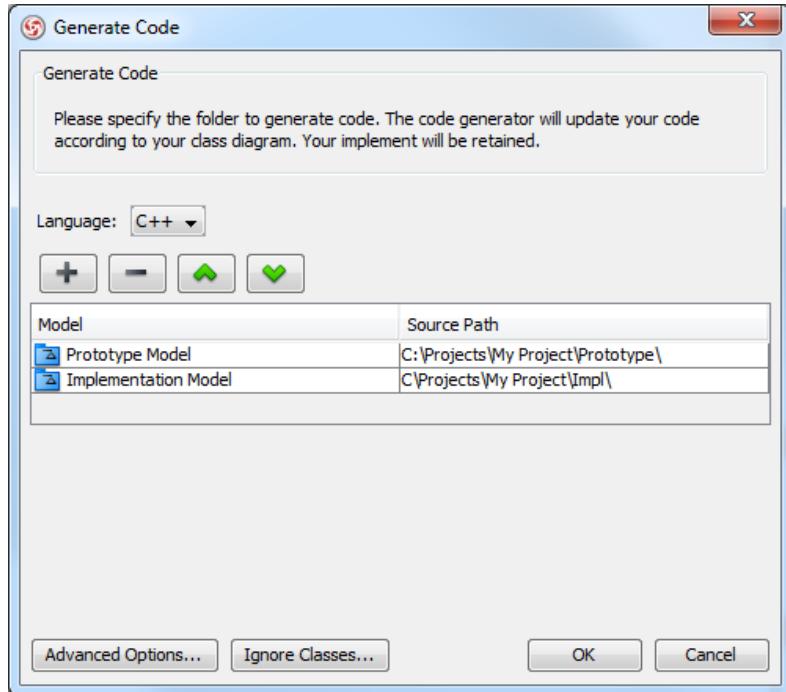
3. Optionally configure the advanced code generation options by clicking **Advanced Options...**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Documentation 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 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).

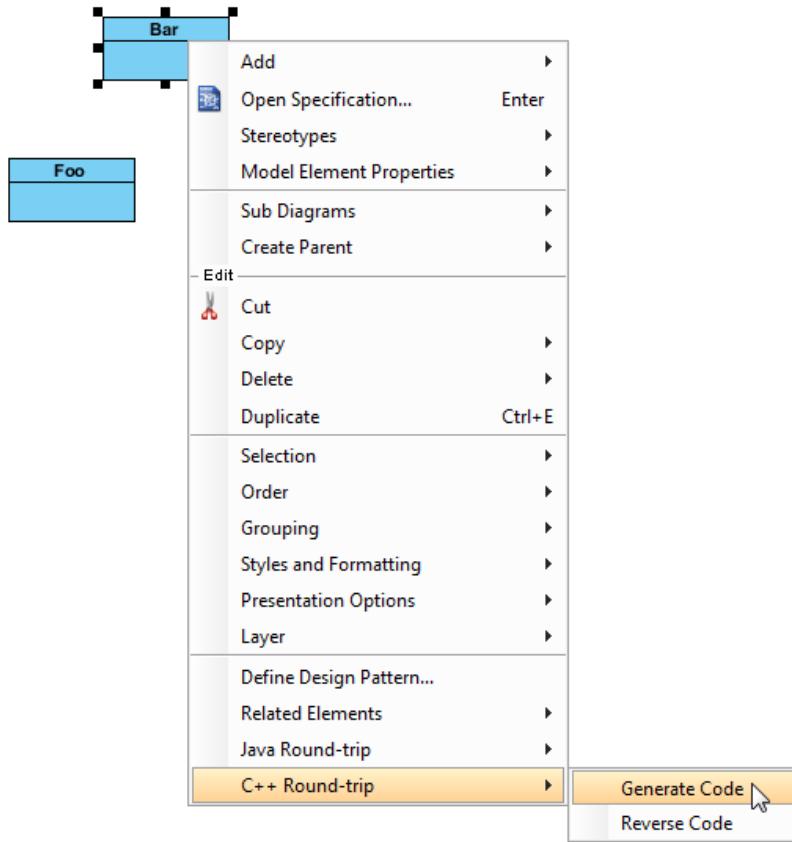
3. Optionally configure the advanced code generation options by clicking **Advanced Options....**. Read the section Advanced Options in this chapter for details about the options.
4. Click **OK** to proceed with generation.

NOTE: Documentation in model elements is generated as comment in code

Generating/Updating code from chosen classes

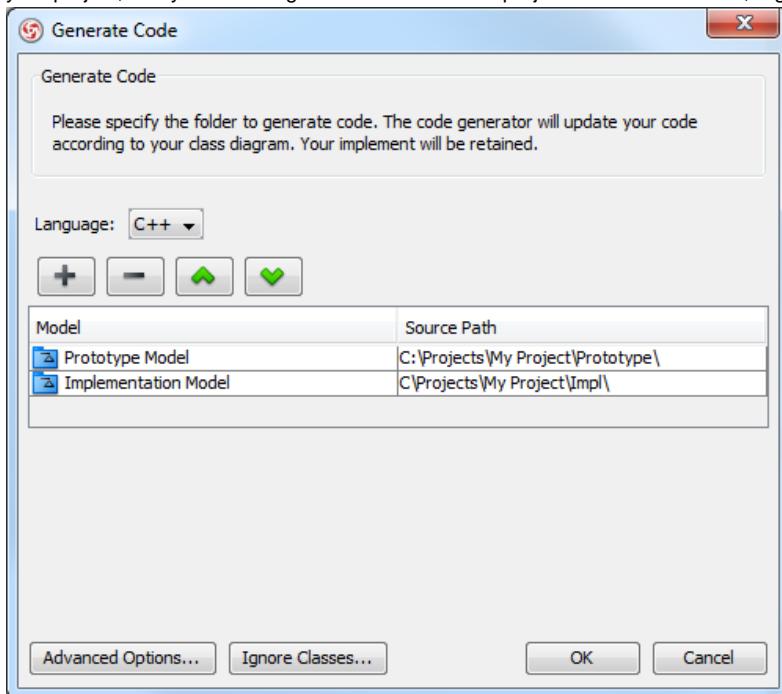
You can generate C++ code from specific class or classes. To generate code from class/classes:

- Select the class(es) and right click on them, then select **C++ Round-trip > Generate Code** from the popup menu.



To generate code for classes

- 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 at 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

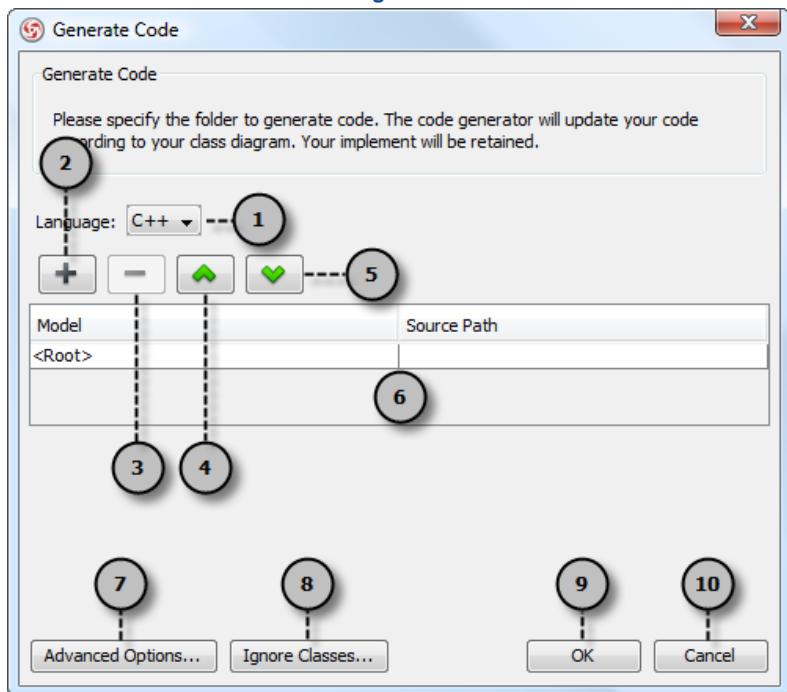
NOTE: If you have generated code for once, the **Generate Code** dialog box will not appear next time you generate/update code for any class selection. If you want to configure the model-to-source-path mapping or to configure options, you can run a code generation for project (refer to the previous section for detail).

- Optionally configure the advanced code generation options by clicking **Advanced Options....**. Read the section Advanced Options in this chapter for details about the options.

- Click **OK** to proceed with generation.

NOTE: Documentation in model elements is generated as comment in code

An overview of Generate Code dialog box



An overview of **Generate Code** dialog box

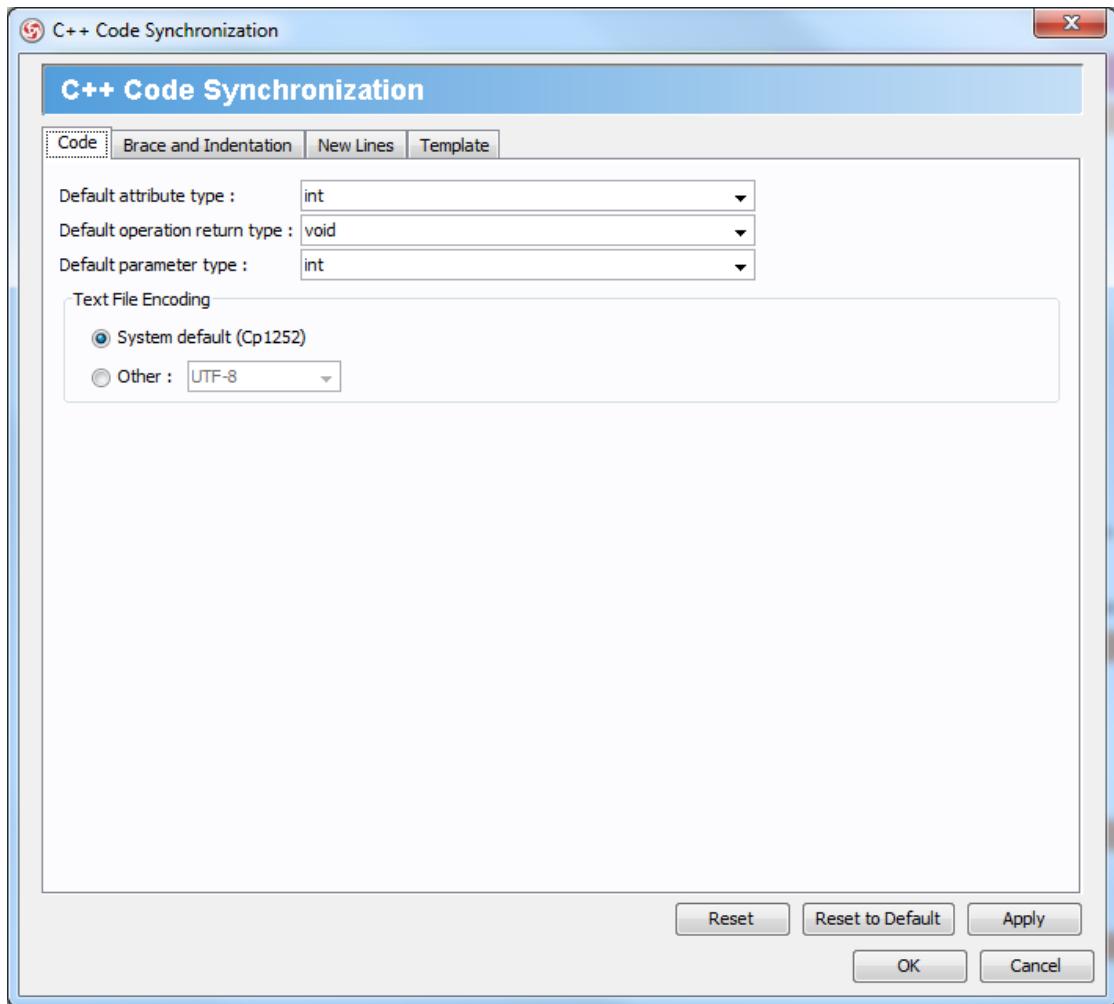
No.	Name	Description
1	Language	The programming language of the source code to generate.
2	Add model-to-source-path mapping	Click to add a new mapping between UML model and the source path where code will be generated to.
3	Remove model-to-source-path mapping	Click to remove chosen model-to-source-path 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 <i>Advanced Options</i> in this chapter.
8	Ignore classes	Click to organize the ignore list of classes to ignore in code generation. For details, read the section <i>To ignore classes in generation</i> 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

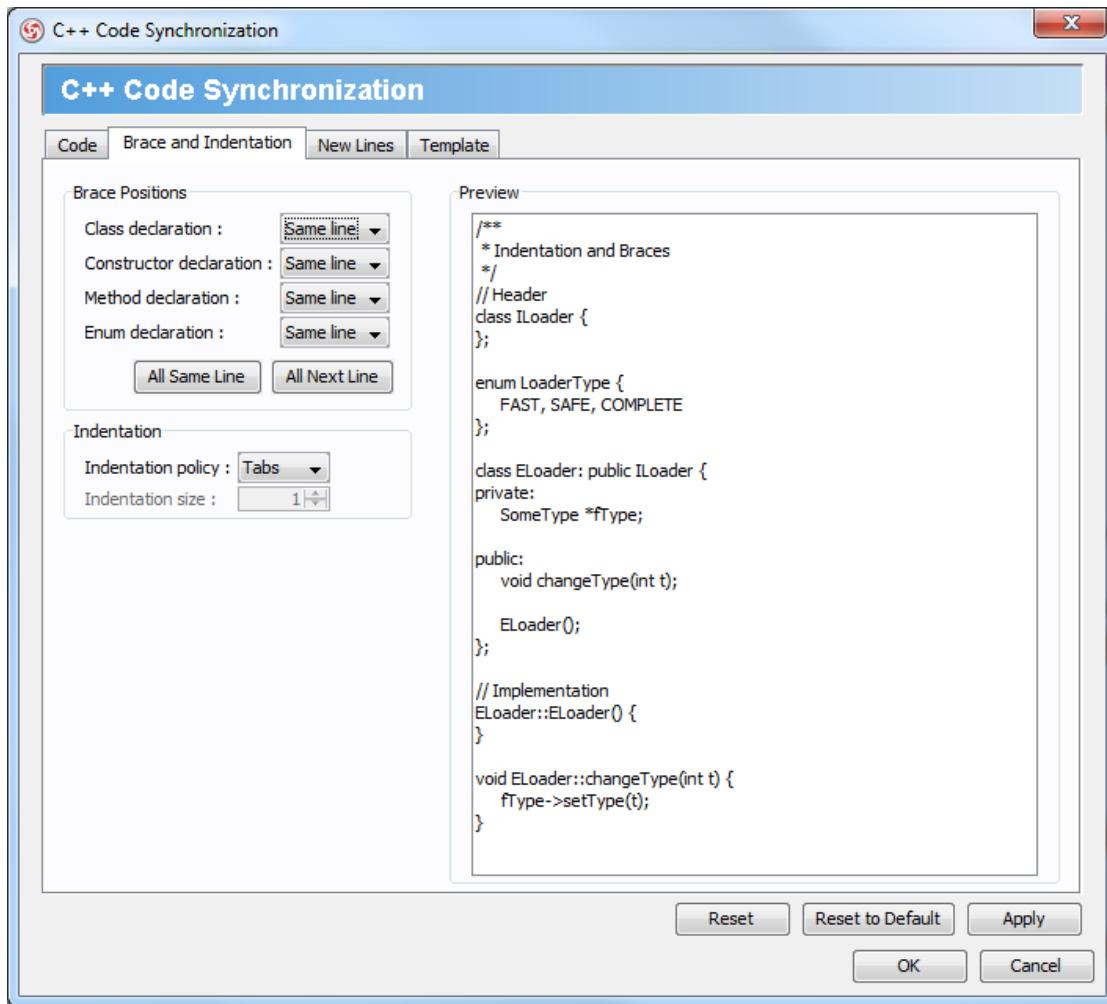


Code configuration

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	<ul style="list-style-type: none">• 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

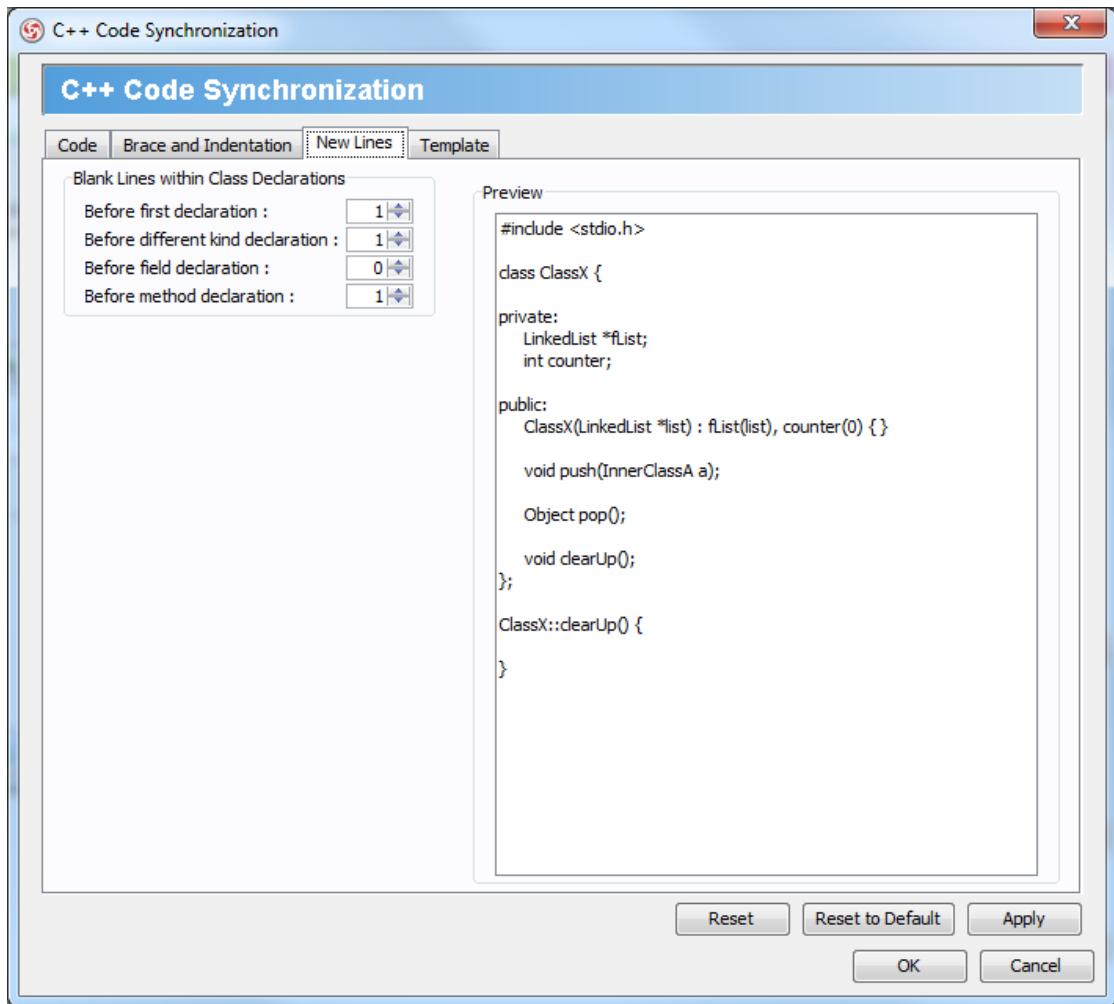


Brace and indentation configuration

Option	Description
Class declaration	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Same line - (default) Brace for enumeration appear at the same line as the declaration • Next line - Brace for enumeration or appear at the line after the declaration
Indentation policy	<ul style="list-style-type: none"> • 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

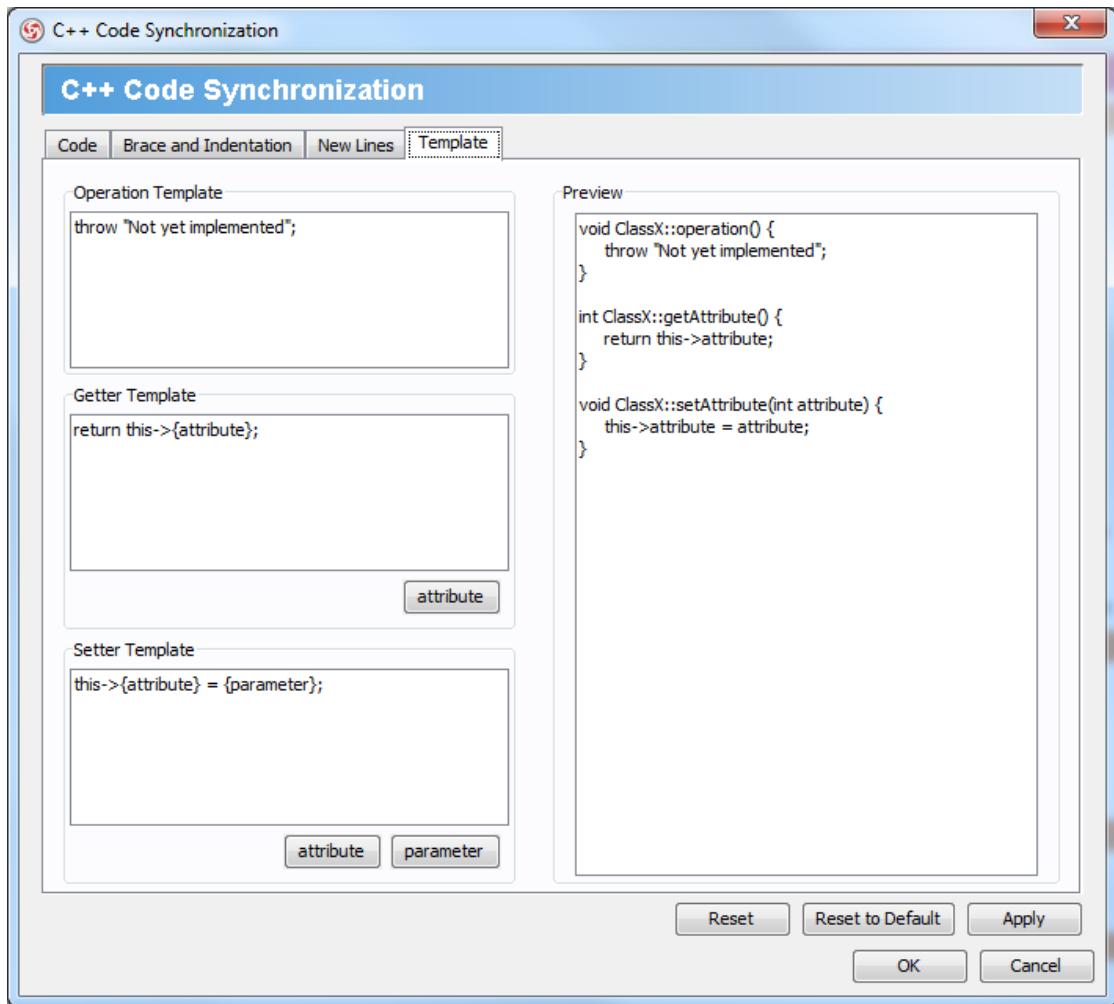


New lines configuration

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



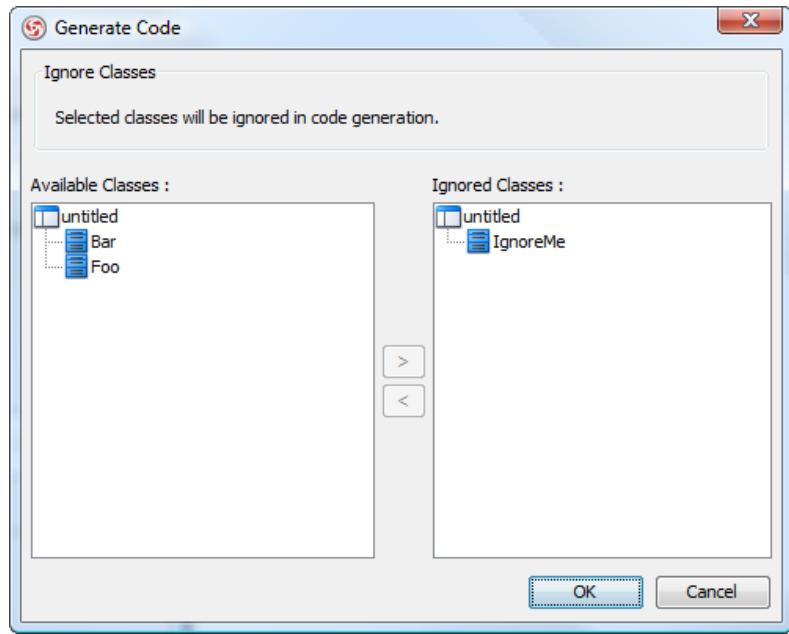
Template configuration

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

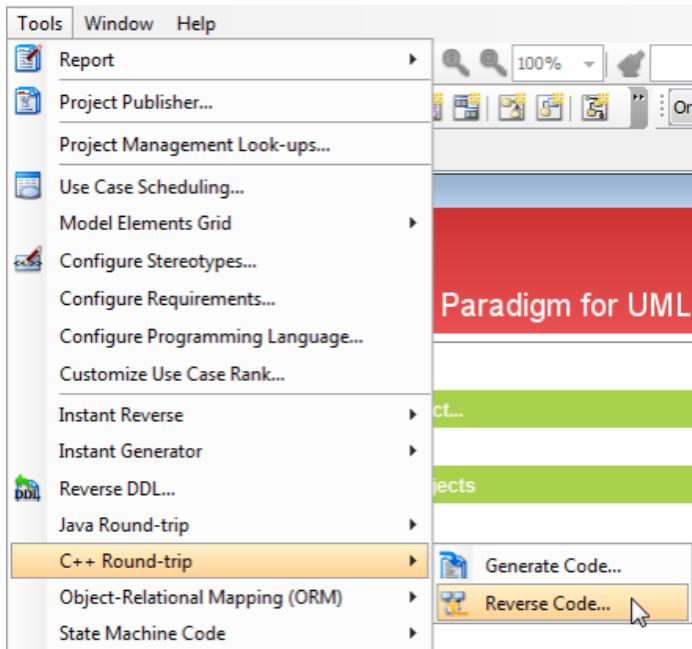
Generate/Update UML classes from C++ code

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

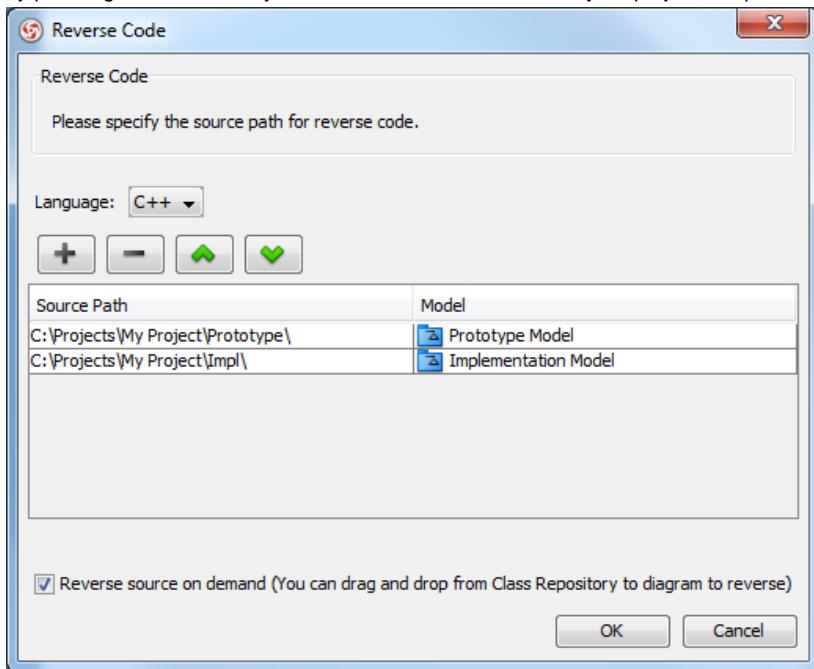
You can produce UML classes from source code, or to update from code all the reversed UML classes in project. To do this:

1. Select **Tools > C++ Round-trip > Reverse Code...** from the main menu.



To reverse code to UML classes

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 phases 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>**.

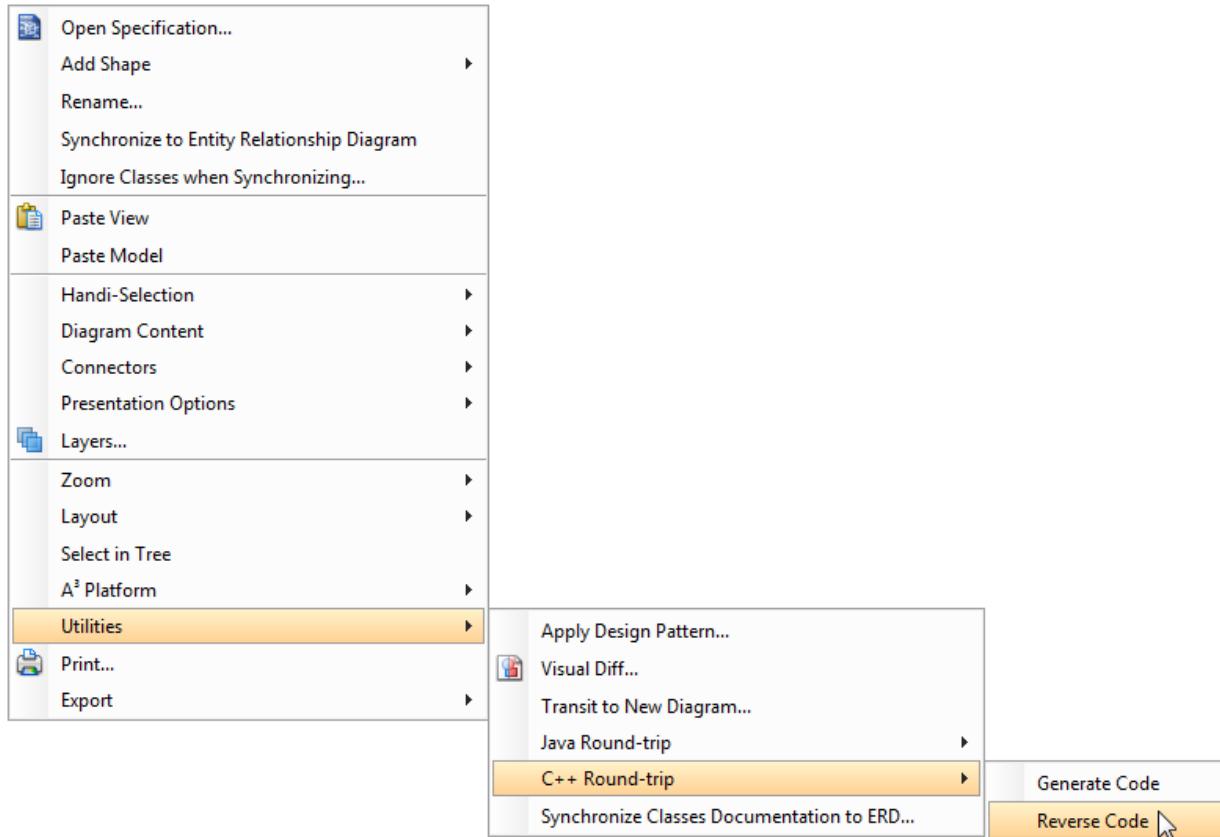


The mappings between source paths and model are defined

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 > C++ Round-trip > Reverse Code** from the popup menu.

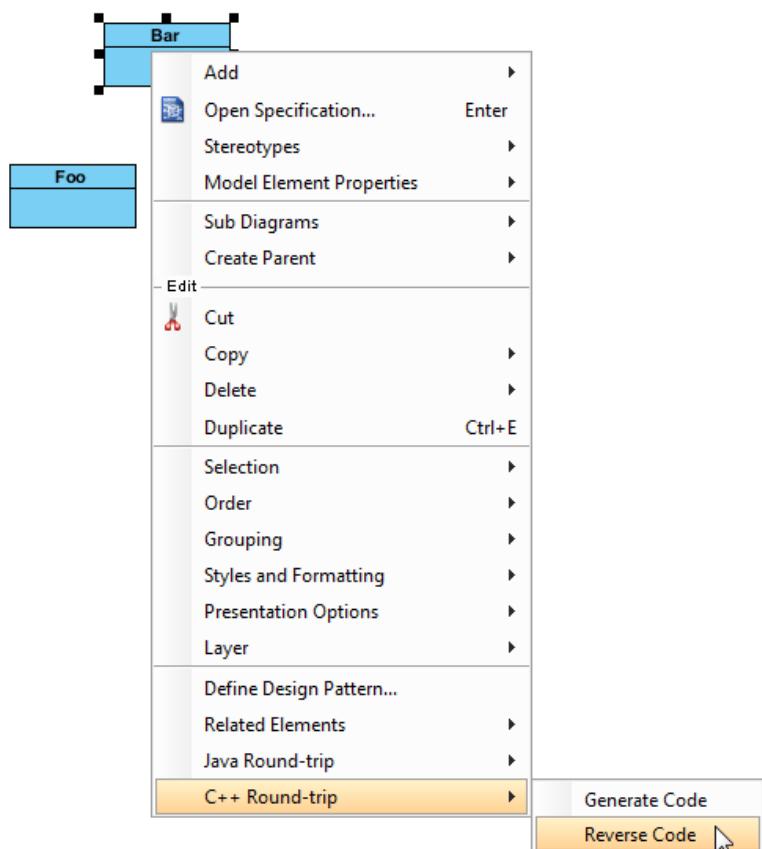


To update UML classes in a diagram from code

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.

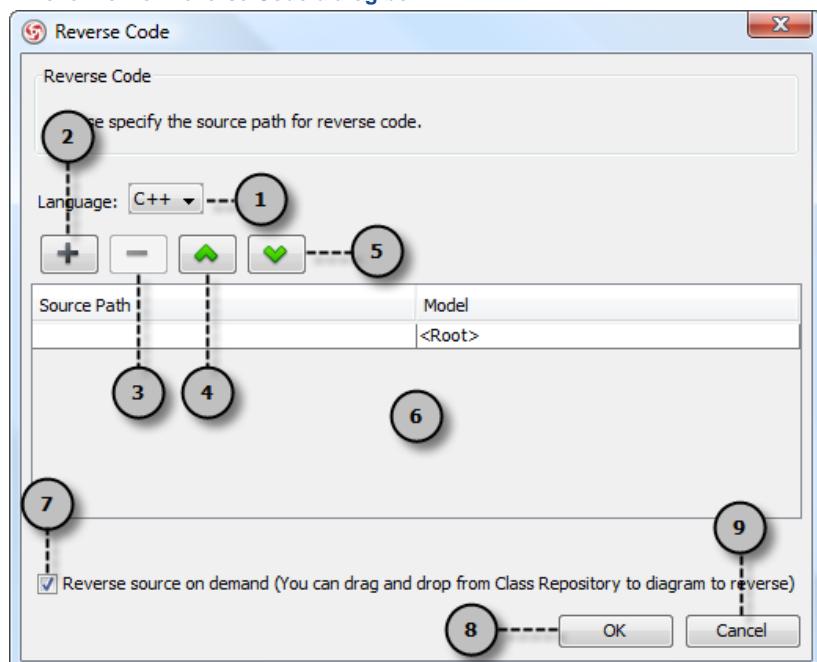
Updating specific UML classes from code

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 popup menu.



To update specific UML class from code

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.

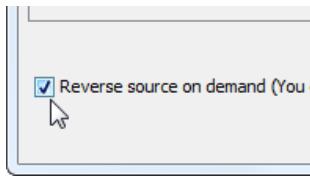
3 Remove source-path-to-model mapping	Click to remove chosen source-path-to-model mapping.
4 Move source-path-to-model mapping up	Click to move chosen source-path-to-model mapping one item upward.
5 Move source-path-to-model mapping down	Click to move chosen source-path-to-model mapping one item downward.
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

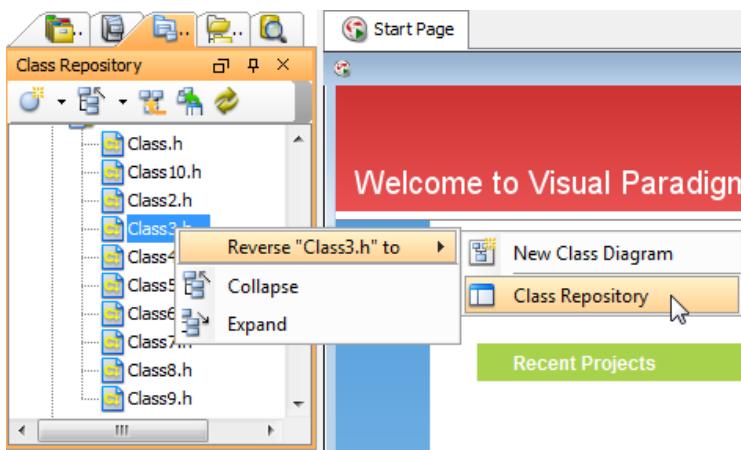
Consider if you have a project that contains million of C++ source file, and now you want to re-develop just a few classes in it. If you try to reverse the whole project it will take you a long time to complete the reverse due to the amount of classes (and relationships) are just too many. With on-demand 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.



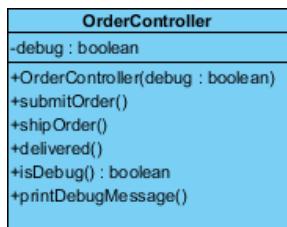
Reversing a C++ source file from index tree

Modeling guidelines

A state machine involve 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

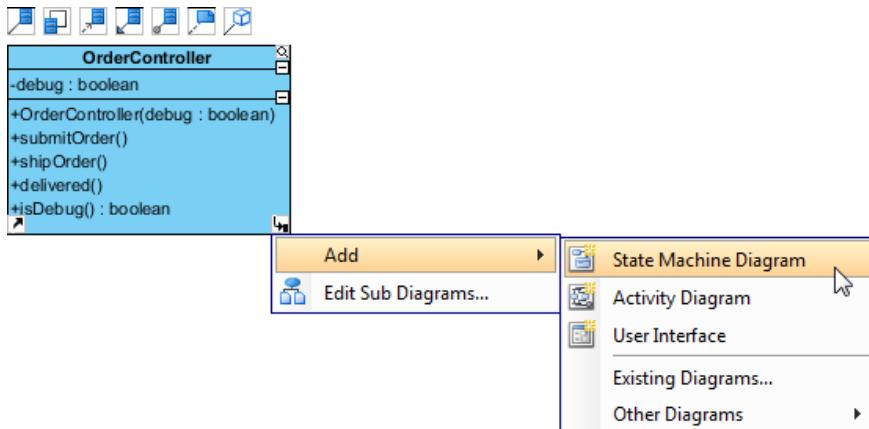
You can create a controller class by selecting **Class** from diagram toolbar and click on the diagram. Name the class properly to represent the nature of controller class. Very often people name controller class as *SomethingController* where the *Something* refers to a use case, or the model that the 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 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.

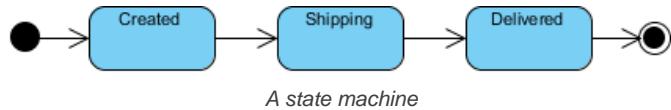
Step 2 - Modeling state machine

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 **Add > State Machine Diagram** from the popup menu.



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 advised 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

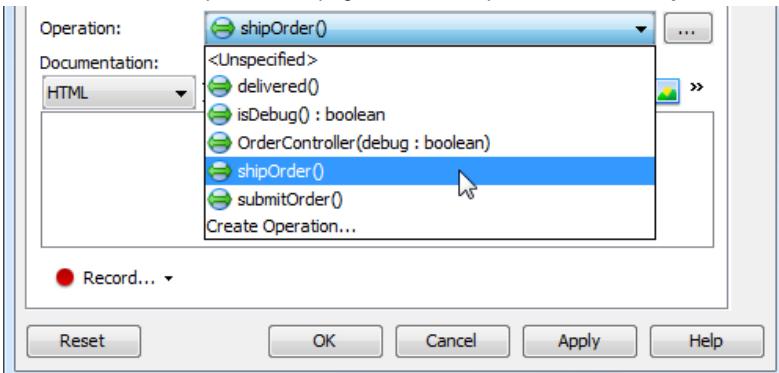
A transition is a relationship between two states, representing the update of states. Previously you have defined operations in controller class. Now, you need to assign those operations to the transitions to indicate the cause of state change. To assign an operation to transition:

1. Right click on a transition and select **Open Specification...** from the popup menu.



Open specification of transition

2. At the bottom of the specification page, select the operation from the **Operation** drop down menu.



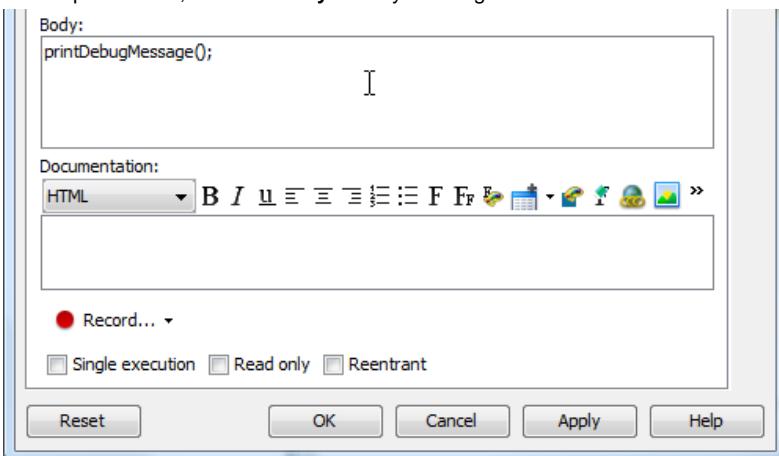
Select operation for transition

Repeat the steps to assign operations to all transitions.

Step 4 - Specifying method body for the entry/exit of state

You can specify the invocation of method call when entering and exiting a state by updating the Entry and Exit properties of state. To do this:

1. Right click on the state and select **Open Specification...** from the popup menu.
2. Click on **Edit...** next to the **Entry** field.
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 **Effect** property of a transition. To do this:

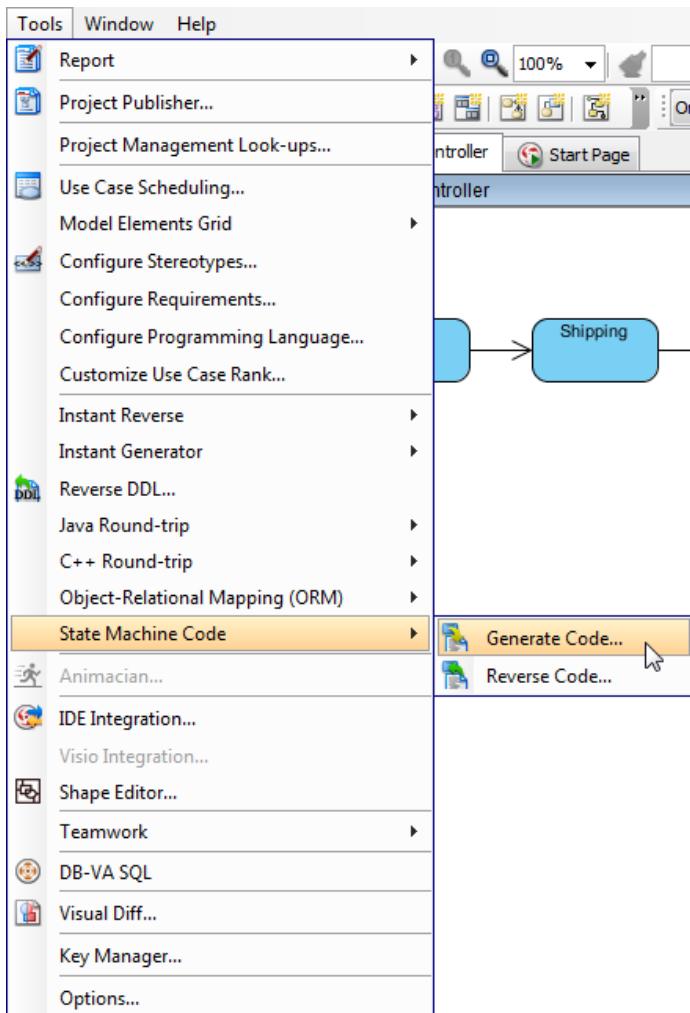
1. Right click on the transition where operation was assigned.
2. In the **General** page, click on **Edit...** next to the **Effect** field.
3. Fill in the **Body**. Click **OK** to confirm.
4. Click **OK** to confirm and go back to diagram.

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 incorporate with the state machine code.

To generate state machine code:

1. Select **Tools > State Machine Code > Generate Code...** from the main menu.

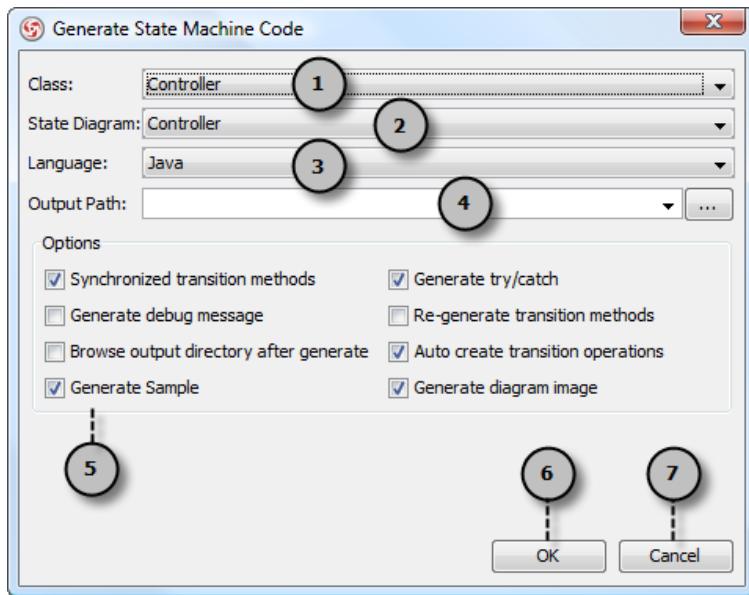


To generate state machine code

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: <ul style="list-style-type: none"> Synchronized transition methods - By checking, it causes the generated code to: <ul style="list-style-type: none"> 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.

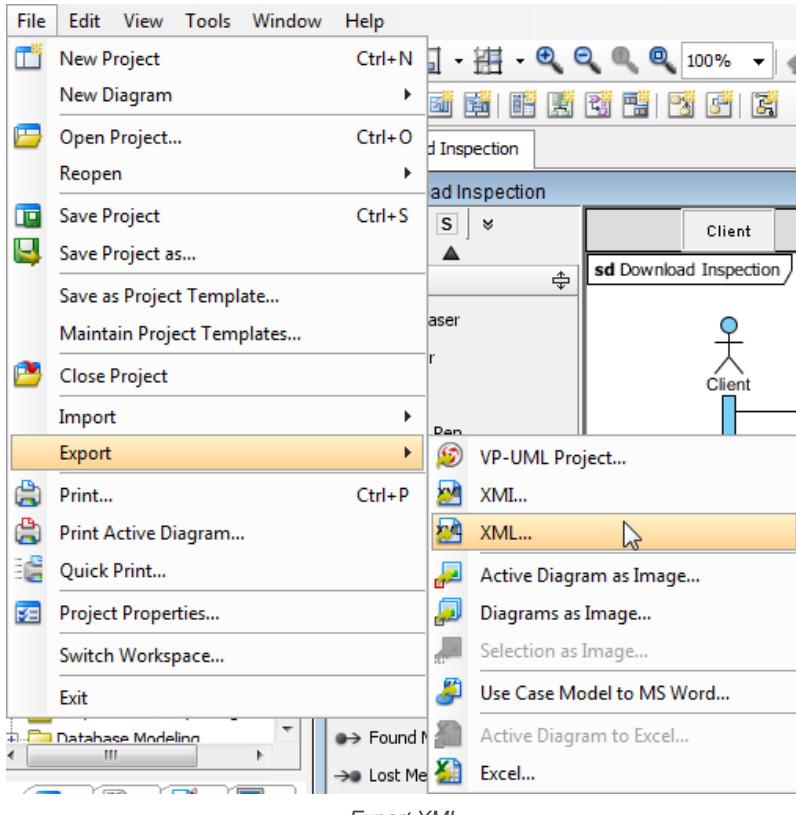
A description of **Generate State Machine Code** dialog box

Exporting XML

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. VP-UML supports interoperability with XML file. You can export project data to an XML, manipulate it externally, and feed the changes back to VP-UML. 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 **File > Export > XML...** from the main menu.

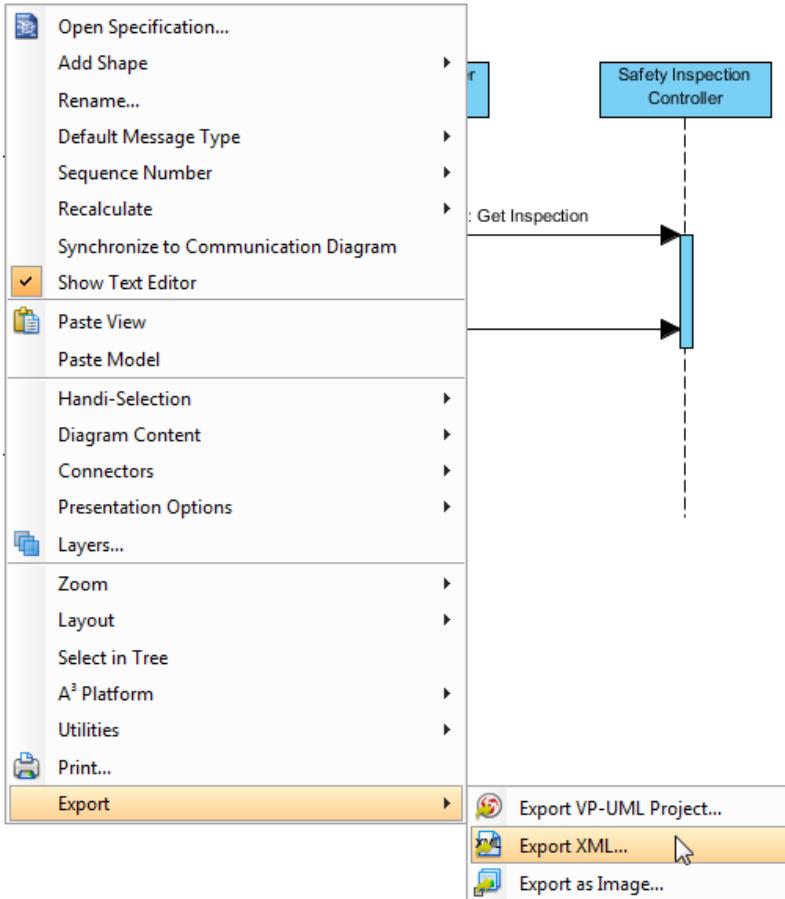


Export XML

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

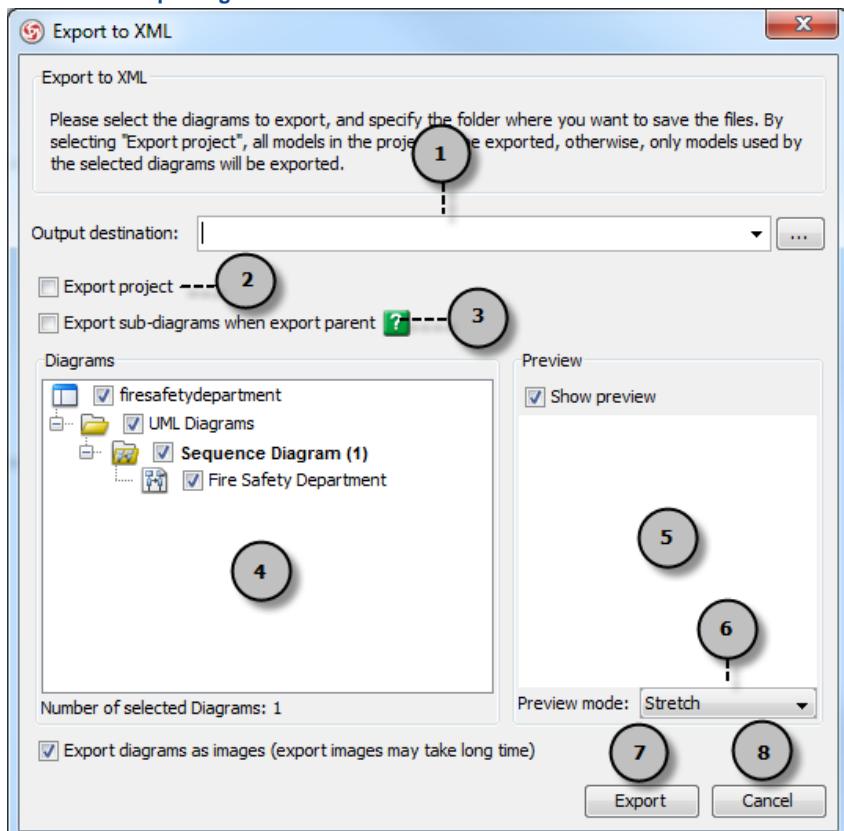
- Right click on diagram and select **Export > Export XML...** from the popup menu.



Export XML from diagram popup

- In the **Export to XML** dialog box, specify the output destination, which is a folder for containing the exported XML files and optionally the image files of diagrams.
- Click **Export**.

Overview of exporting XML

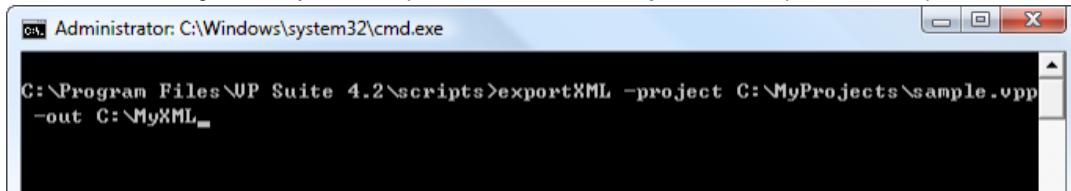


No.	Name	Description
1	Output destination	The location where you want to save the file.
2	Export project	By checking Export project , all models in the project will be exported.
3	Export sub-diagrams when export parent	<p>By selecting Export sub-diagrams when export parent, the sub-diagram(s) will also be exported when the parent diagram(s) is exported.</p> <p>For example, package <i>MyWorks</i> 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 <i>MyWorks</i>. By turning off the option, B will be ignored when export.</p>
4	Diagram list	All diagrams in your project will be shown in here.
5	Preview window	By checking the selected diagram and Show preview , it will be shown in preview window.
6	Preview mode	<p>You can choose either Stretch or Real size to preview your diagram.</p> <p>Stretch: The ratio of your diagram will be fit in the size of preview window.</p> <p>Real size: The ratio of your diagram will be shown on the preview window as its real size.</p>
7	Export	Click Export to proceed with exporting to XML.
8	Cancel	Click Cancel to discard exporting to XML.

Description of Export XML dialog box

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.



```
C:\Program Files\UP Suite 4.2\scripts>exportXML -project C:\MyProjects\sample.vpp
-out C:\MyXML-
```

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

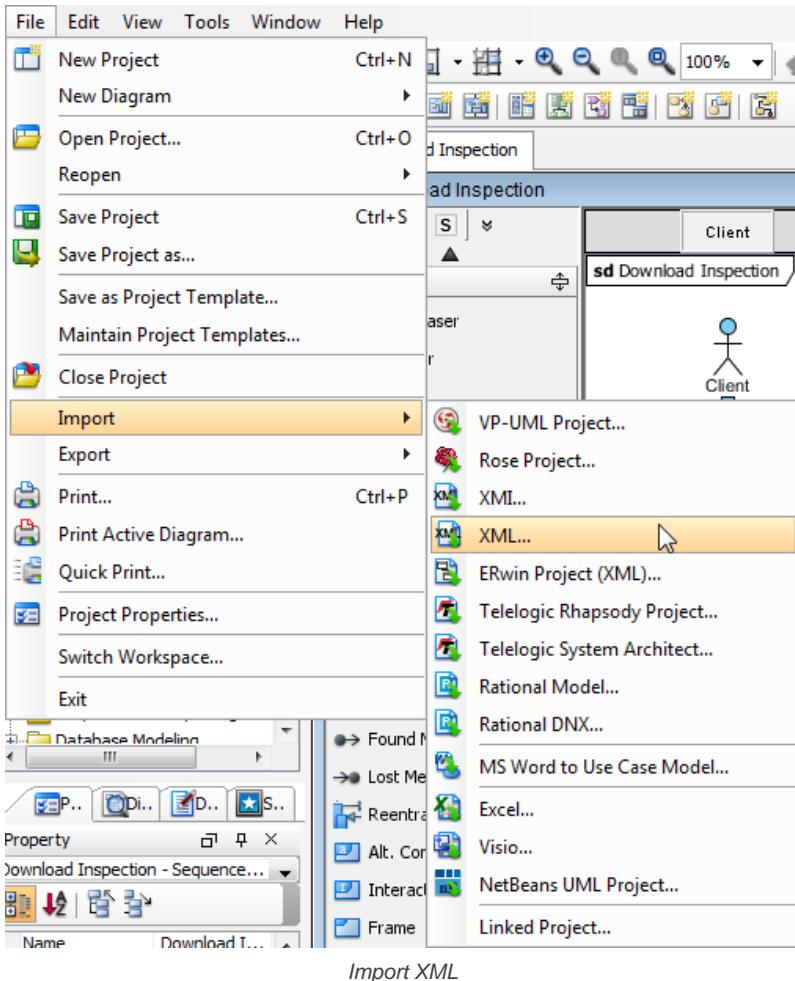
Upon finishing, you can visit the output destination specified to obtain the XML files.

Importing XML

You can import changes made externally in XML back to VP-UML, 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 VP-UML. Only XML exported from VP-UML can be imported.

Importing XML to current project

1. Select **File > Import > XML...** from the main menu.



2. Specify the file path of the XML to import.

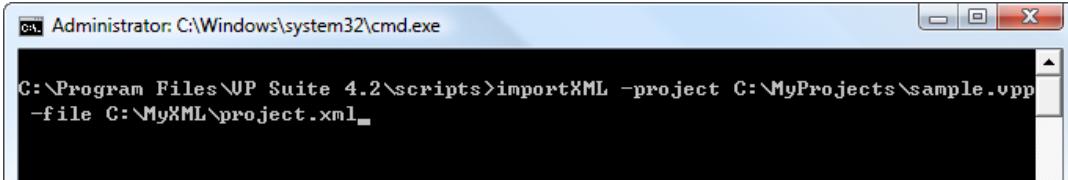
3. Click **OK**.

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.



```
Administrator: C:\Windows\system32\cmd.exe
C:\Program Files\WP Suite 4.2\scripts>importXML -project C:\MyProjects\sample.vpp
-file C:\MyXML\project.xml
```

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

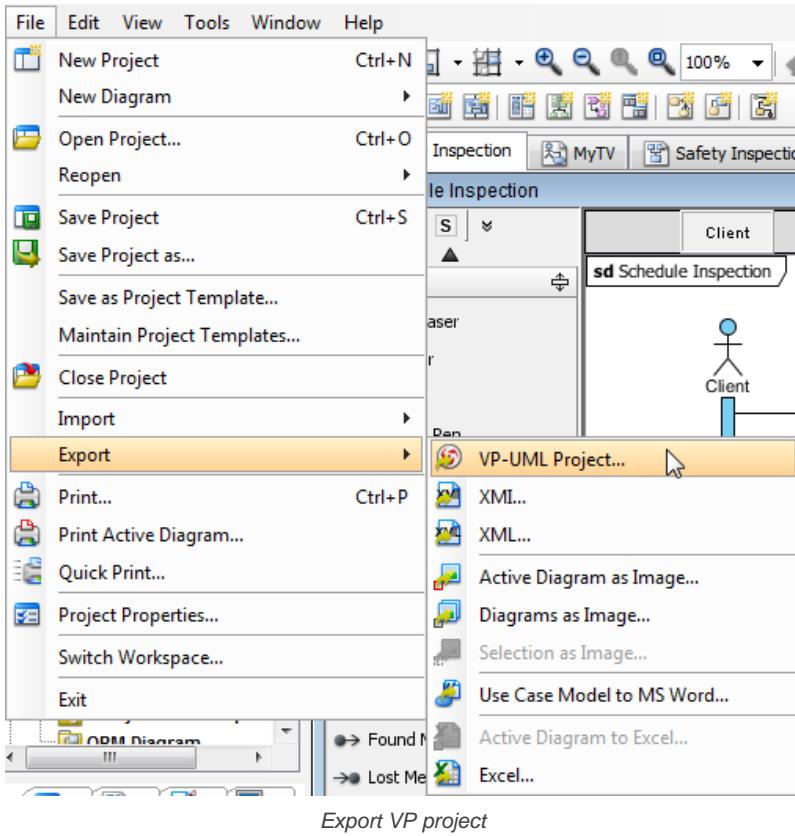
Parameters for ImportXML

Upon finishing, the project file will be updated with the data presented in the XML file.

Exporting VP project

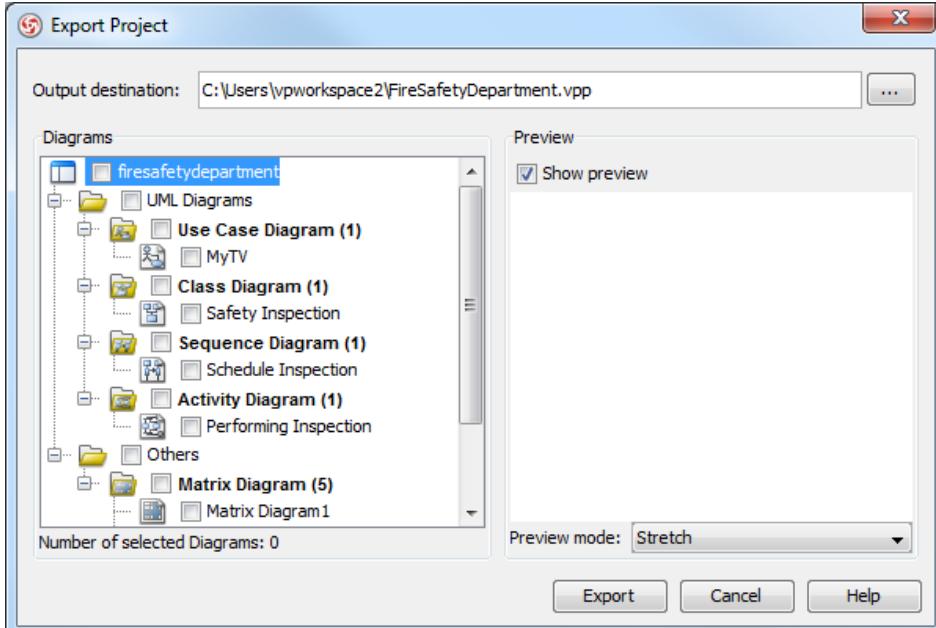
Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. VP-UML supports interoperability with native .vpp project file. You can export some of the diagrams to a project file, send to your team member for editing, and feed the changes back to VP-UML. In this chapter, you will see how to export project file for diagrams in project. To export VP project:

1. Select **File > Export > VP-UML Project...** from the main menu.



Export VP project

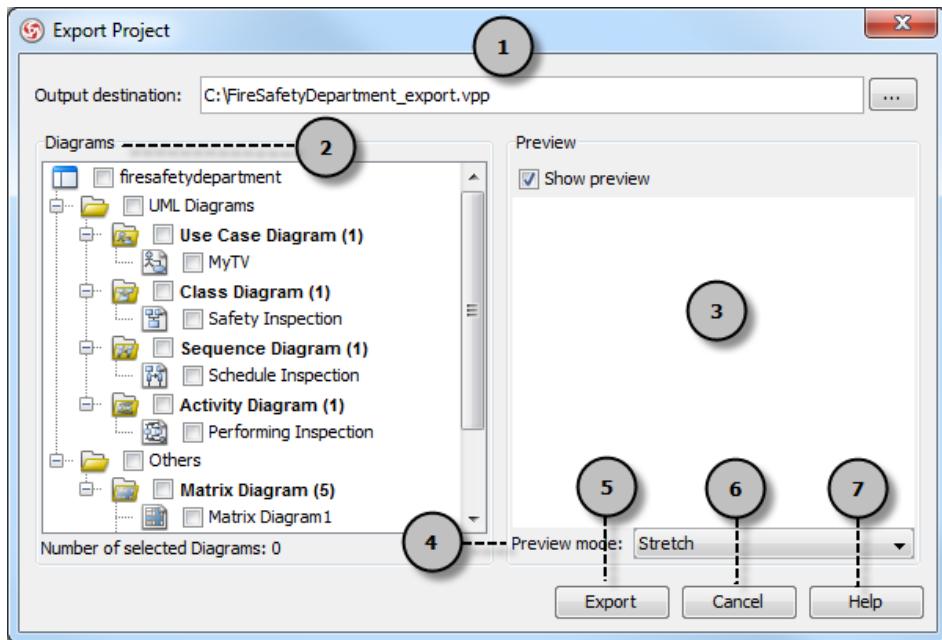
2. Specify the output destination.



Inputting 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**. 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

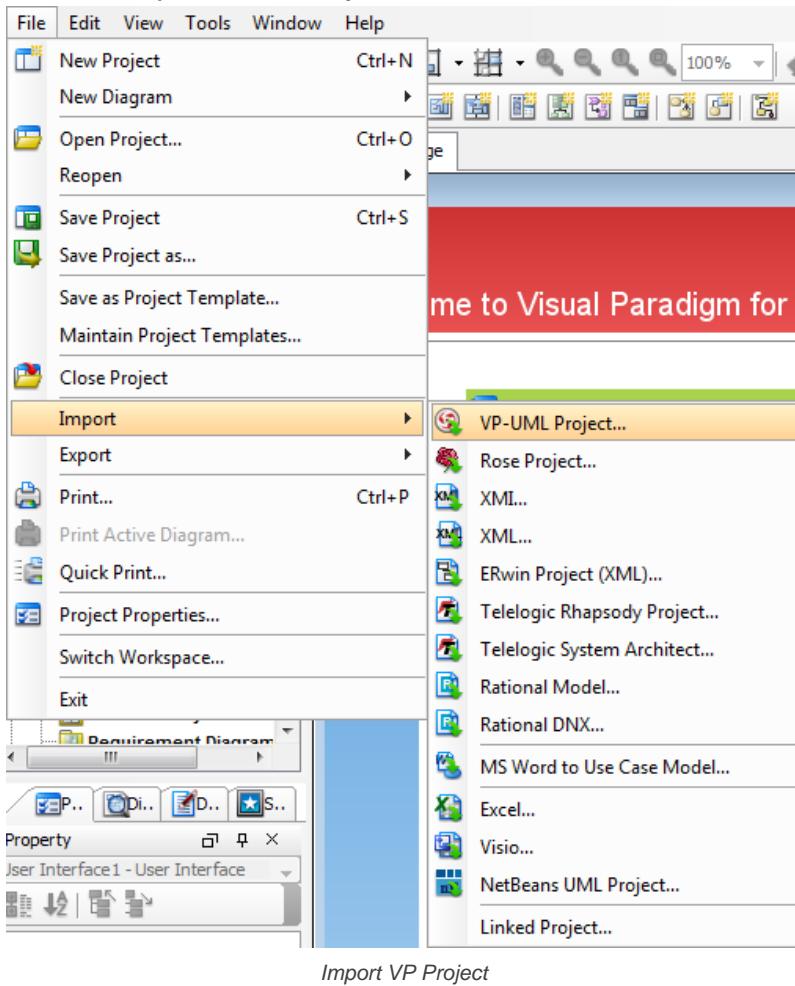
No.	Name	Description
1	Output destination	The location where you want to save the file.
2	Diagrams	All diagrams in your project will be shown in here.
3	Preview window	By checking the selected diagram and Show preview, it will be shown in preview window.
4	Preview mode	You can choose either Stretch or 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

Importing VP project

You can import a VP project to VP-UML 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 **File > Import > VP-UML Project...** from the main menu.



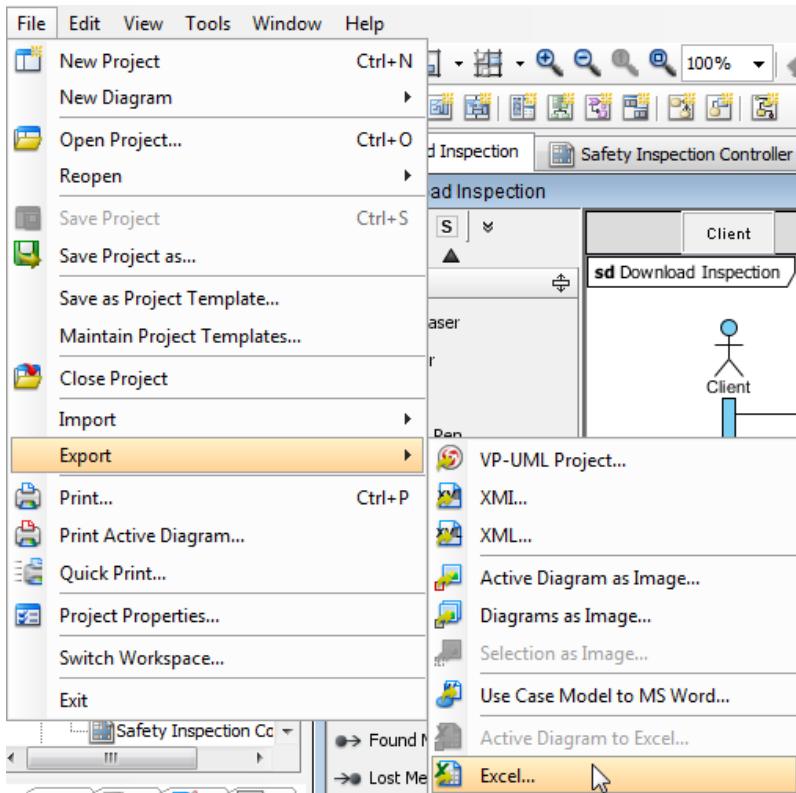
2. Select the file path of the .vpp file to import in file chooser.
3. Click **Open**.

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.

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. VP-UML 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 VP-UML. In this chapter, you will see how to export an Excel file from a project. To export Excel:

1. Select **File > Export > Excel...** from the main menu.



To export Excel

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.

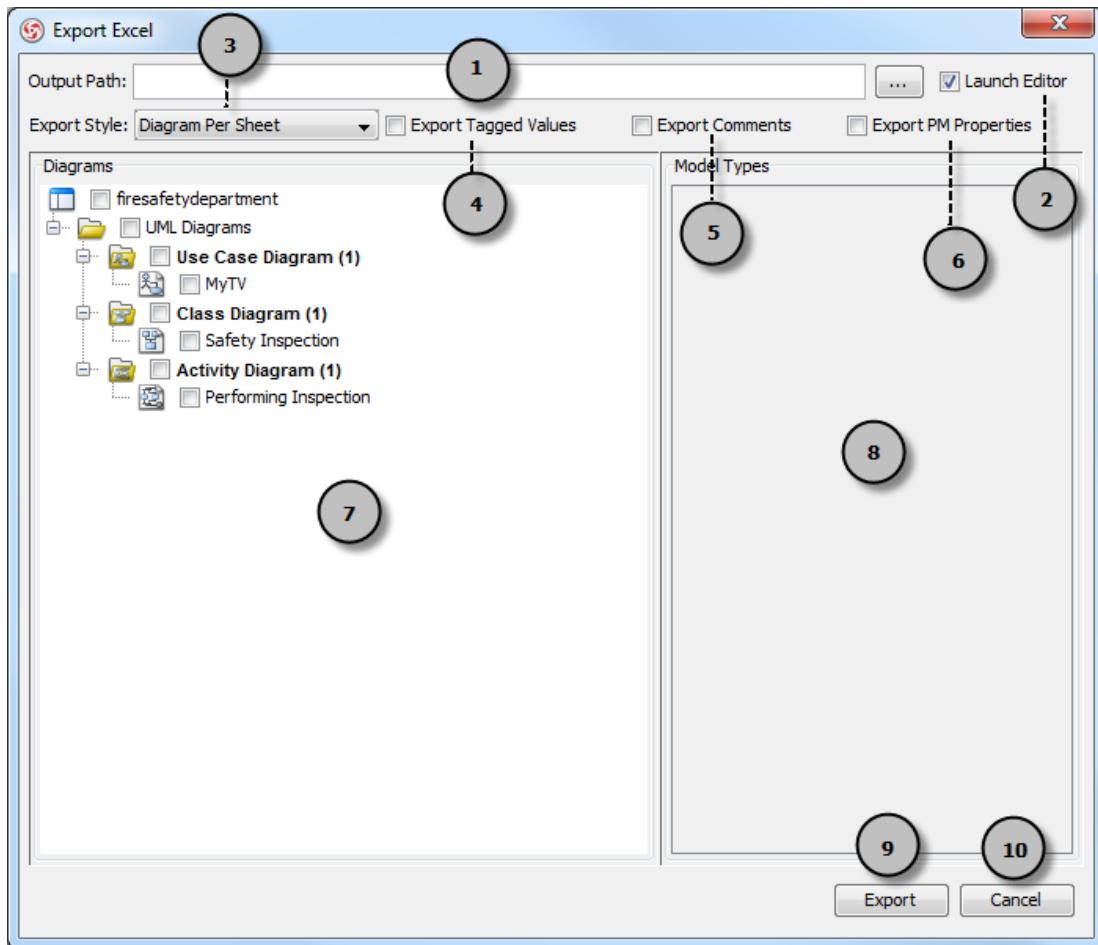
5. Click **Export**. Upon finishing, you can visit the output destination specified to obtain the Excel file.

The screenshot shows a Microsoft Excel window titled "export.xls [Compatibility Mode] - Microsoft Excel". The ribbon tabs are Home, Insert, Page Layout, Formulas, Data, Review, and View. The Home tab is selected. The toolbar includes Paste, Font, Alignment, Number, Styles, Cells, and Editing. The main area displays a table representing a Class Diagram. The columns include ID, Name, Type, Documentation, and Delete ? for Diagrams; Class, ID, Name, Stereotypes, Visibility, and Documentation for Classes; Attribute, ID, Name, Stereotypes, Initial value, and Documentation for Attributes; Association, ID, Name, Stereotypes, From, From Role, and To for Associations; and Class, ID, Name, Stereotypes, Visibility, and Documentation for Item entities. The table has 22 rows, numbered 1 through 22. Row 1 is a header for the Diagram section. Rows 2-3 show a Safety Inspection ClassDiagram. Rows 4-9 show the attributes of the SafetyInspection class. Rows 10-15 show the attributes of the Inspector class. Rows 16-18 show the associations of the Safety Inspection class. Row 19 is a header for the Item entity. Rows 20-22 show the attributes of the Item entity. The status bar at the bottom shows "Ready" and "100%".

A	B	C	D	E	F
1	Diagram	ID	Name	Type	Documentation
2	1	Safety Inspection	ClassDiagram		No
3					
4	Class	ID	Name	Stereotypes	Visibility
5	2	SafetyInspection	<<entity>> <<ORM Persistable>>	public	No
6	Attribute	ID	Name	Stereotypes	Initial value
7		3	inspectionDate		Unspec
8		4	ID		Unspec
9					
10	Class	ID	Name	Stereotypes	Visibility
11	5	Inspector	<<ORM Persistable>>	public	No
12	Attribute	ID	Name	Stereotypes	Initial value
13		6	name		Unspec
14		7	ID		Unspec
15					
16	Association	ID	Name	Stereotypes	From
17	8			2	1
18					
19	Class	ID	Name	Stereotypes	Visibility
20	9	Item	<<ORM Persistable>>	public	No
21	Attribute	ID	Name	Stereotypes	Initial value
22		10	description		Unspec

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.
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	Diagrams	All diagrams in the opening project are listed here. Select the diagrams to export to Excel.
8	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.
9	Export	Click Export to proceed with exporting Excel.
10	Cancel	Click Cancel to diocard exporting to Excel.

Overview of the **Export Excel** dialog box

Importing Microsoft Excel file

You can import changes made externally in Excel file back to VP-UML, 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 VP-UML. Only Excel exported from VP-UML can be imported.

Guidelines in modifying Excel

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.

The screenshot shows a Microsoft Excel spreadsheet titled "FireSafetyInspection". The data is organized into several tables:

	A	B	C	D	E	F	
1	Diagram	ID	Name	Type	Documentation	Delete ?	
2	1	Safety Inspection	ClassDiagram		No		
3							
4	Class	ID	Name	Stereotypes	Visibility	Documentation	
5	2	FireSafetyInspection	<<entity>> <<ORM Persistable>>	public	No		
6	Attribute	ID	Name	Stereotypes	Initial value		
7	3	inspectionDate			Unspec		
8	4	ID			Unspec		
9	Class	ID	Name	Stereotypes	Visibility	Documentation	
10	5	Inspector	<<ORM Persistable>>	public	No		
11	Attribute	ID	Name	Stereotypes	Initial value		
12	6	name			Unspec		
13	7	ID			Unspec		
14	Association	ID	Name	Stereotypes	From	From Role	To
15	8				2		1
16	Class	ID	Name	Stereotypes	Visibility	Documentation	
17	9	Item	<<ORM Persistable>>	public	No		
18	Attribute	ID	Name	Stereotypes	Initial value		
19	10	description					Unspec

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.

The screenshot shows a Microsoft Excel spreadsheet titled "export.xls [Compatibility Mode] - Microsoft Excel". The table has several rows and columns. The first two rows (6 and 7) have headers: "Visible", "Setter", "Getter", "Abstract", and "Delete ?". The third row (8) has values: Yes, No, No, No, Yes. The fourth row (12) has headers: "Documentation", "Abstract", "Leaf", "Derived", "ORM Collection Type", "Transit From", "Transit To", and "Delete ?". The fifth row (17) has values: No, No, No, Unspecified, and No. The sixth row (21) has headers: "Visible", "Setter", "Getter", "Abstract", and "Delete ?". The seventh row (22) has values: Yes, No, No, No, and No. The status bar at the bottom shows "Safety Inspection" and "System Data".

	Visible	Setter	Getter	Abstract	Delete ?			
7	Yes	No	No	No	No			
8	Yes	No	No	No	Yes			
	Visible	Setter	Getter	Abstract	Delete ?			
13	Yes	No	No	No	No			
14	Yes	No	No	No	No			
	Documentation	Abstract	Leaf	Derived	ORM Collection Type	Transit From	Transit To	Delete ?
17		No	No	No	Unspecified			No
	Visible	Setter	Getter	Abstract	Delete ?			
22	Yes	No	No	No	No			

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 leave blank.

The screenshot shows a Microsoft Excel window titled "export.xls [Compatibility Mode] - Microsoft Excel". The spreadsheet contains several tables representing UML elements:

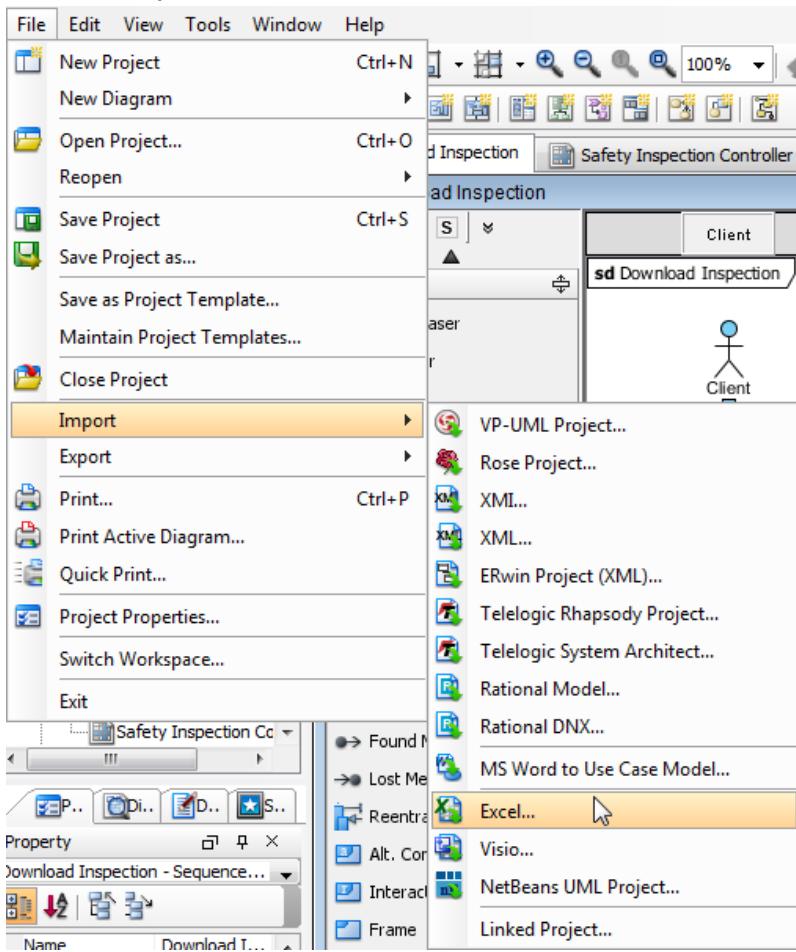
- Diagram**: A table with columns: ID, Name, Type, Documentation, Delete ?.
- Class**: A table with columns: ID, Name, Stereotypes, Visibility, Documentation.
- Attribute**: A table with columns: ID, Name, Stereotypes, Initial value.
- Class**: A table with columns: ID, Name, Stereotypes, Visibility, Documentation.
- Association**: A table with columns: ID, Name, Stereotypes, From, From Role, To, To Role.
- Class**: A table with columns: ID, Name, Stereotypes, Visibility, Documentation.
- Attribute**: A table with columns: ID, Name, Stereotypes, Initial value.

The "Safety Inspection" sheet is selected. The status bar at the bottom shows "Count: 14" and "100%".

Adding attribute in Excel

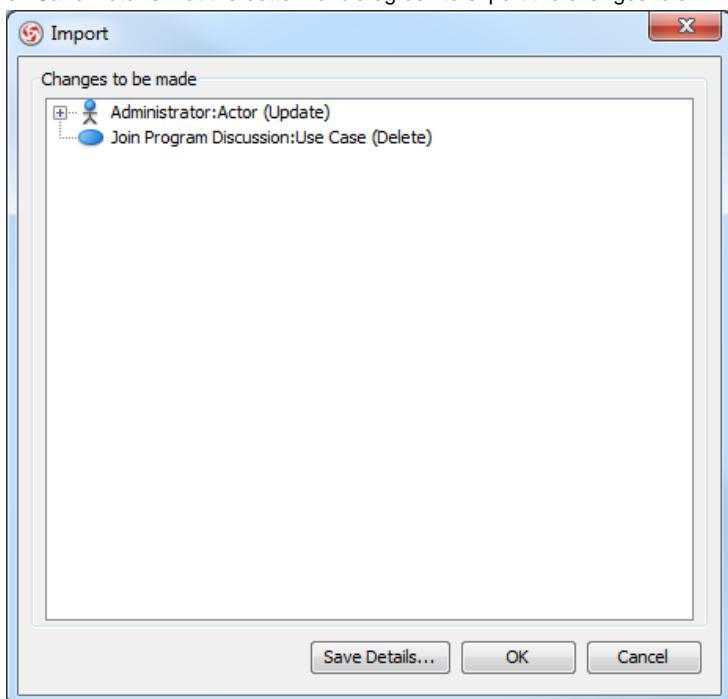
Importing an Excel file

1. Select **File > Import > Excel...** from the main menu.



To import Excel

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 dialog box

4. Click **OK** to proceed.

Excel modification guidelines

When you are going to modify Excel file exported from VP-UML, make sure the changes you make are all valid. Making invalid changes will cause the file cannot be imported back to VP-UML. 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.

The screenshot shows a Microsoft Excel spreadsheet titled "Safety Inspection". The data is organized into several tables:

- Diagram**: A table with columns for ID, Name, and Type. One row shows ID 1, Name "Safety Inspection", and Type "ClassDiagram".
- Class**: A table with columns for Class, ID, Name, Stereotypes, Visibility, and Documentation. One row shows Class 2, ID 2, Name "FireSafetyInspection<<entity>> <<ORM Persistable>>", Stereotype "<<ORM Persistable>>", Visibility "public", and Documentation "No".
- Attribute**: A table with columns for Class, ID, Attribute, ID, Name, Stereotypes, and Initial value. Rows include inspectionDate (Name) and ID (Name).
- Inspector**: A table with columns for Class, ID, Name, Stereotypes, Visibility, and Documentation. One row shows Class 5, ID 5, Name "Inspector", Stereotype "<<ORM Persistable>>", Visibility "public", and Documentation "No".
- Item**: A table with columns for Association, ID, Class, ID, Name, Stereotypes, From, From Role, To, To Role, and Documentation. One row shows Association 8, Class 9, ID 2, Name "Item", Stereotype "<<ORM Persistable>>", From "2", To "1", and Documentation "No".
- System Data**: A table with columns for Class, ID, Name, Stereotypes, Visibility, and Documentation. One row shows Class 9, ID 9, Name "Item", Stereotype "<<ORM Persistable>>", Visibility "public", and Documentation "No".

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.

The screenshot shows a Microsoft Excel spreadsheet titled "export.xls [Compatibility Mode] - Microsoft Excel". The table has several rows and columns of data. The first two rows (6 and 7) define headers for attributes like Visible, Setter, Getter, Abstract, and Delete?. Rows 8 and 13 also have these headers. Row 16 defines headers for Documentation, Abstract, Leaf, Derived, ORM Collection Type, Transit From, Transit To, and Delete?. Row 22 contains a single "Yes" value under the Delete? column. The "Delete?" column is highlighted in blue.

	Visible	Setter	Getter	Abstract	Delete ?			
7	Yes	No	No	No	No			
8	Yes	No	No	No	Yes			
13	Yes	No	No	No	No			
14	Yes	No	No	No	No			
16	Documentation	Abstract	Leaf	Derived	ORM Collection Type	Transit From	Transit To	Delete ?
17		No	No	No	Unspecified			No
21	Visible	Setter	Getter	Abstract	Delete ?			
22	Yes	No	No	No	No			

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 leave blank.

	A	B	C	D	E	F
1	Diagram	ID	Name	Type	Documentation	Delete ?
2	1	Safety Inspection	ClassDiagram		No	
4	Class	ID	Name	Stereotypes	Visibility	Documentation
5	2	FireSafetyInspector	<<entity>> <<ORM Persistable>>	public	No	
6	Attribute	ID	Name	Stereotypes	Initial value	
7	3	inspectionDate			Unspec	
8	4	ID			Unspec	
9		added			Unspec	
11	Class	ID	Name	Stereotypes	Visibility	Documentation
12	5	Inspector	<<ORM Persistable>>	public	No	
13	Attribute	ID	Name	Stereotypes	Initial value	
14	6	name			Unspec	
15	7	ID			Unspec	
17	Association	ID	Name	Stereotypes	From	From Role
18	8				2	1
20	Class	ID	Name	Stereotypes	Visibility	Documentation
21	9	Item	<<ORM Persistable>>	public	No	
22	Attribute	ID	Name	Stereotypes	Initial value	

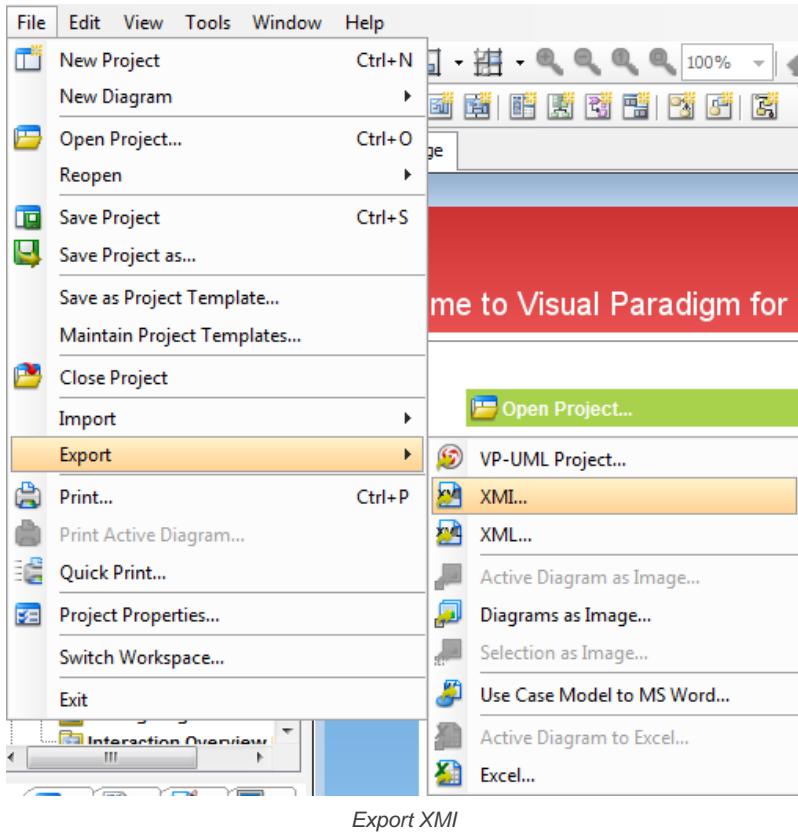
Adding attribute in Excel

Exporting XMI

Interoperability is the ability to exchange information between two systems and to use the information that has been exchanged. VP-UML 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 **File > Export > XMI...** from the main menu.

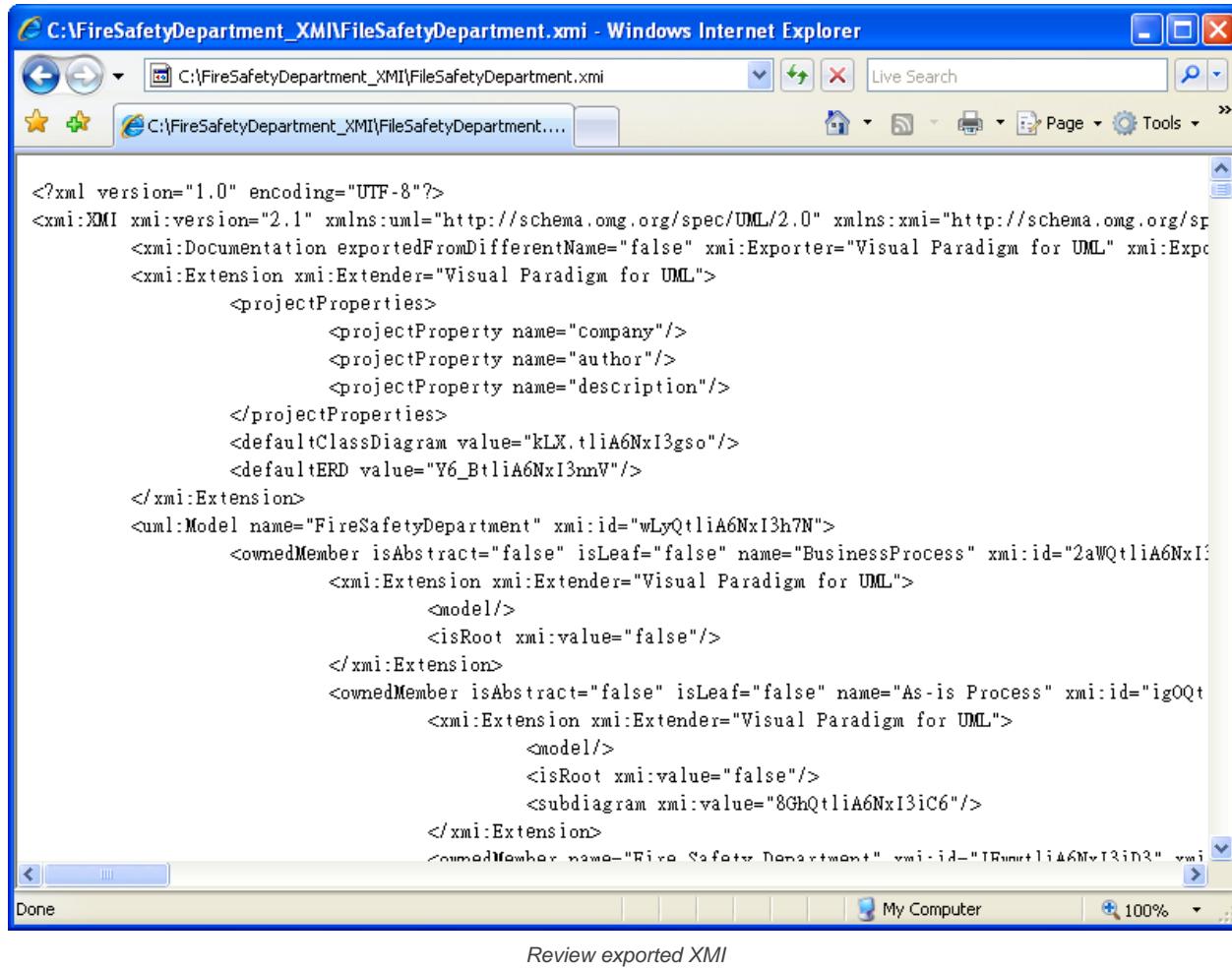


Export XMI

This displays the **Export XMI** dialog box.

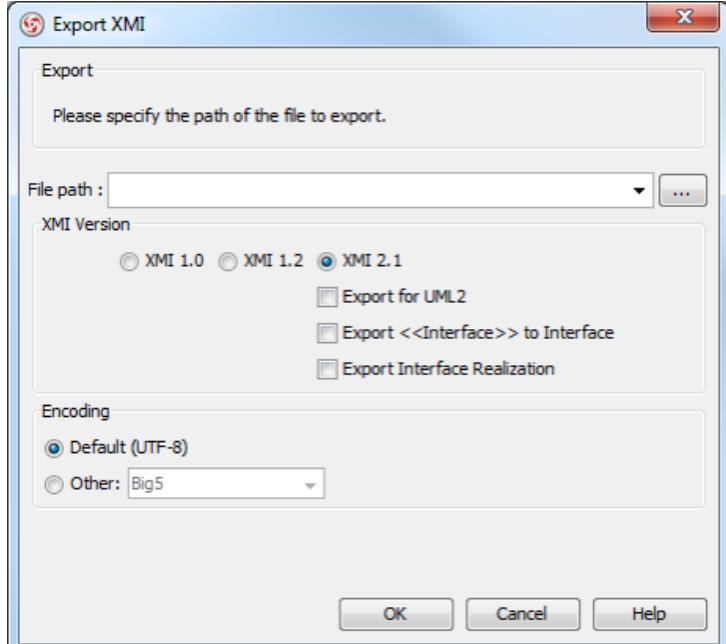
2. Specify the file path of the XMI file.
3. Configure the exporting such as setting the XMI version and changing the encoding of XMI.

4. Click **OK** to start exporting. Upon finishing, you can visit the output destination specified to obtain the XMI.



The screenshot shows a Windows Internet Explorer window with the title "C:\FireSafetyDepartment_XMI\FileSafetyDepartment.xmi - Windows Internet Explorer". The address bar displays "C:\FireSafetyDepartment_XMI\FileSafetyDepartment.xmi". The page content is the XML code for the "FileSafetyDepartment" model, which includes project properties, default class diagram, default ERD, and two owned members: "BusinessProcess" and "As-is Process". The XML uses namespaces for UML and XMI, and specific attributes like "xmi:id" and "xmi:version". The code is scrollable, and the status bar at the bottom shows "Review exported XMI".

An overview of Export XMI dialog box



An overview of the **Export XMI** dialog box

No.	Name	Description
1	File path	The file path for the XMI file to export.
2	XMI version	There are three versions of XMI to suit different interoperability needs, including 1.0, 1.2 and 2.1. If the latest version 2.1 is selected, the following options are then available for selection:

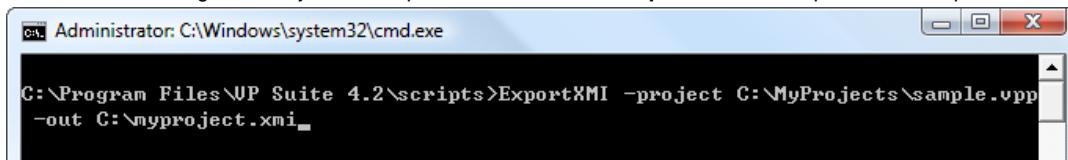
- **Export for UML 2** - By checking this option, the exported XMI will conform to the tool UML2's standard. The following options will appear in further.
 - **Export Data Type to** - Determine whether to export data type to UML or Ecore primitive type
 - **Export Java Annotation to EAnnotation** - Determine 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 dialog box

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.



```
C:\Program Files\WP Suite 4.2\scripts>ExportXMI -project C:\MyProjects\sample.vpp
-out C:\myproject.xmi
```

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: <ul style="list-style-type: none"> • 1.0 • 1.2 • 2.1 • 2.1UML2 	2.1
-encoding [optional]	Encoding of XMI file	UTF-8

Parameters for ExportXMI

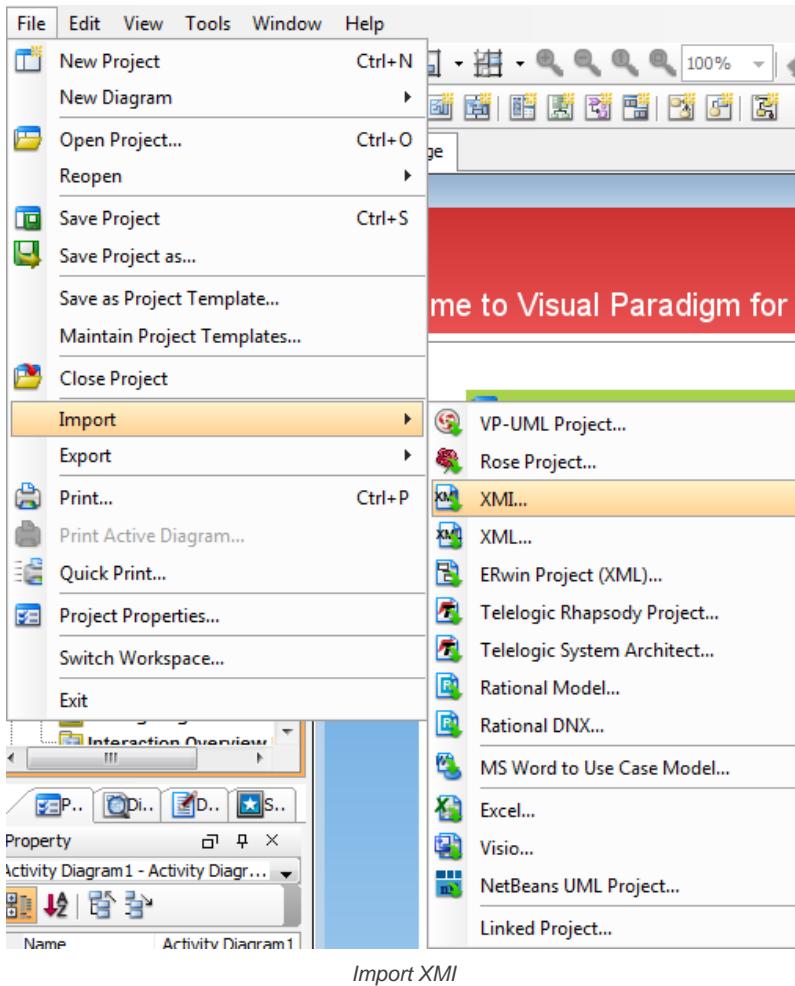
Upon finishing, you can visit the output destination specified to obtain the XMI.

Importing XMI

You can migrate your works done in another software to VP-UML 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 **File > Import > XMI...** from the main menu.

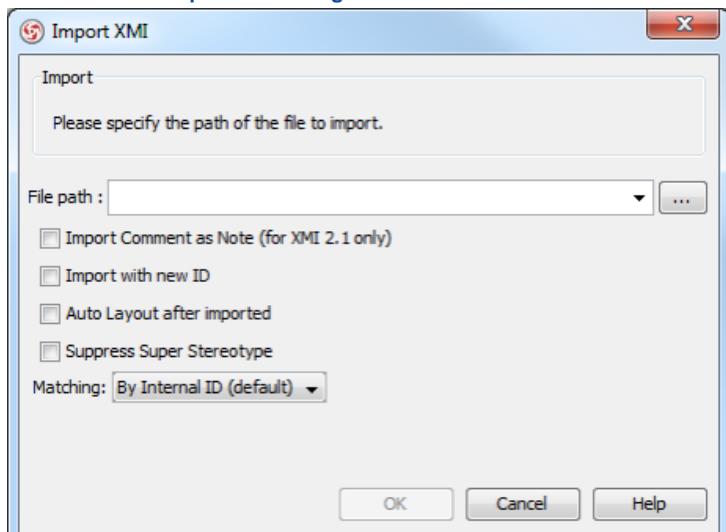


This displays the **Import XMI** dialog box.

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 dialog box

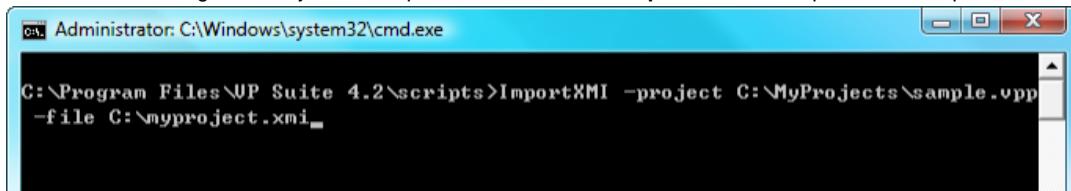


No.	Name	Description
1	File path	The file path of the XMI file to import.
2	Import Comment as Note (for XMI 2.1 only)	Determines whether to import comment as documentation, or as note of model.
3	Auto Layout after imported	Determines whether to run a layout on diagram after import. Note that running layout may takes time for a massive amount of diagram data.
4	Suppress Super Stereotype	Determines whether to ignore super stereotype when importing.
5	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.

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.



```
Administrator: C:\Windows\system32\cmd.exe
C:\Program Files\UP Suite 4.2\scripts>ImportXMI -project C:\MyProjects\sample.vpp
-file C:\myproject.xmi
```

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

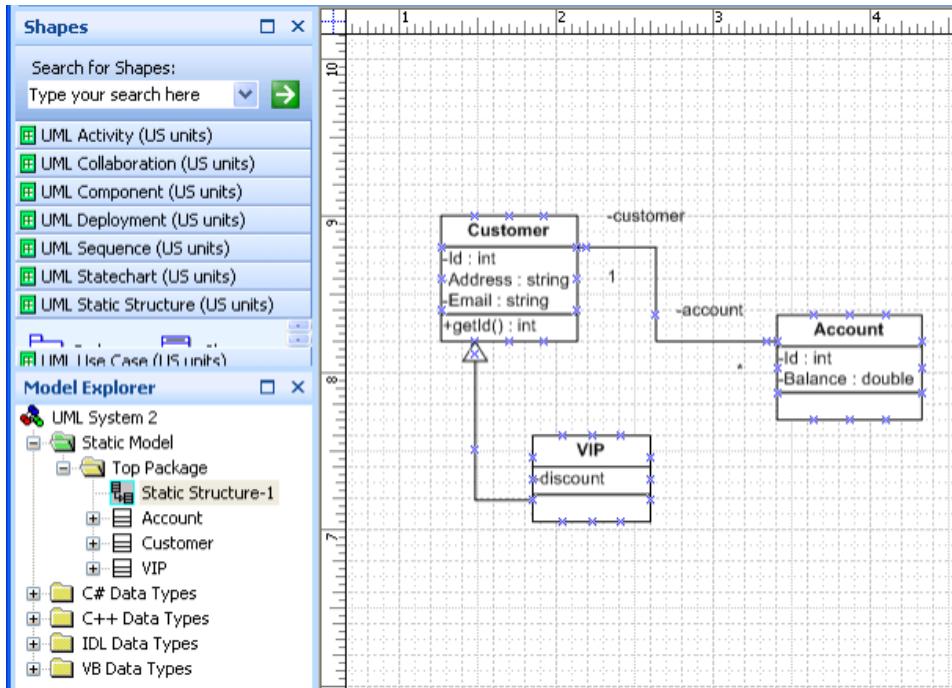
Table 4-4 Parameters for ImportXMI

Upon finishing, the project file will be updated with the data presented in the XMI file.

Importing Visio drawing into VP-UML

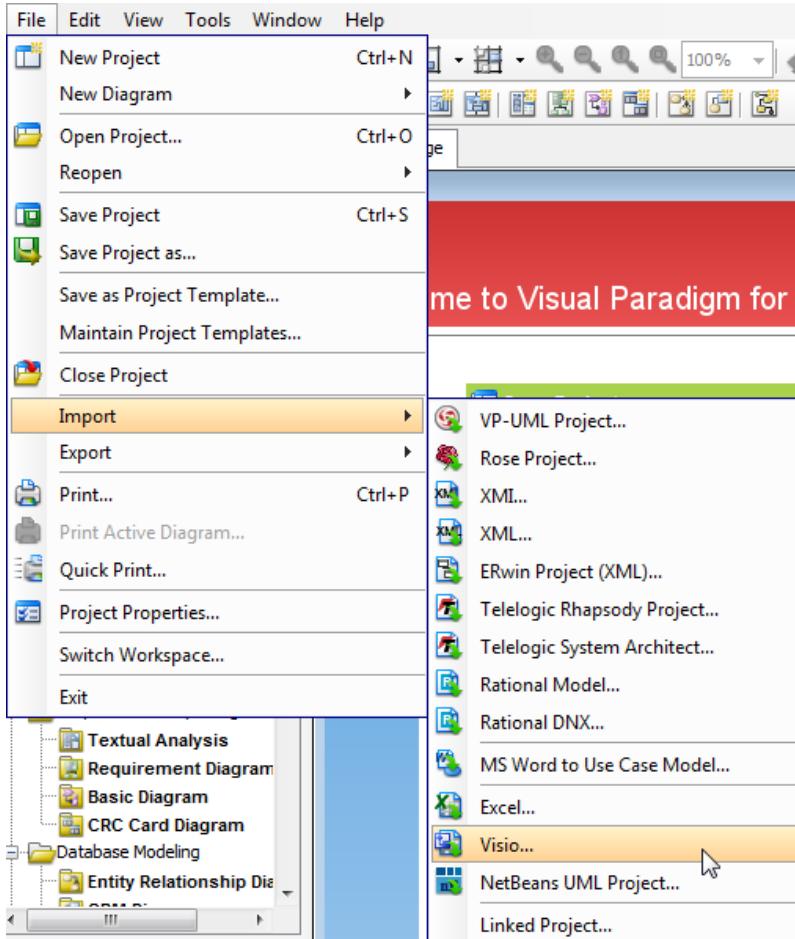
You may have drew diagrams in Visio. You can now import your previous work through the import feature.

1. Save your Visio drawing as a .vdx file.



A Visio drawing

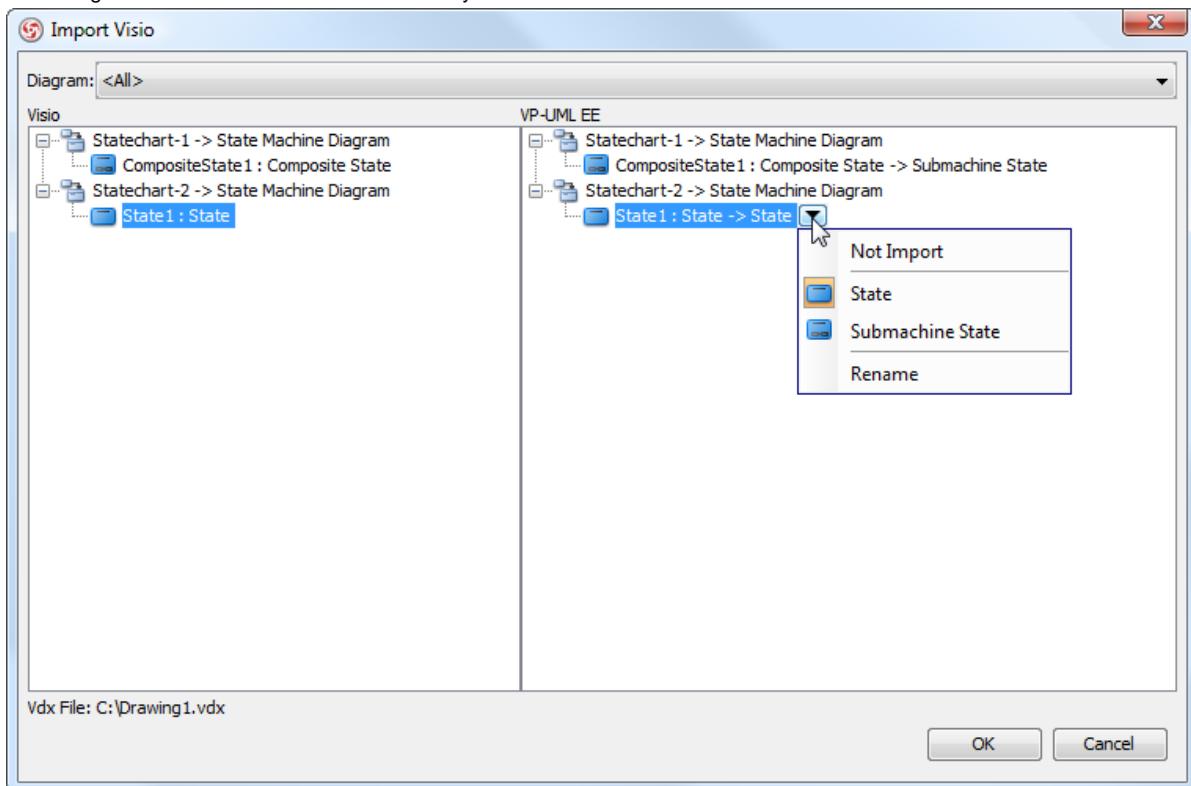
2. To import a Visio drawing into VP-UML, select **File > Import > Visio...** in the main menu of VP-UML.



3. Specify the file path of the Visio drawing.

NOTE: Only valid XML Drawing (*.vdx) can be imported. If your Visio drawing does not have a .vdx as extension, open it in Visio and save it as a .vdx file.

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 VP-UML.



The Import Visio dialog box

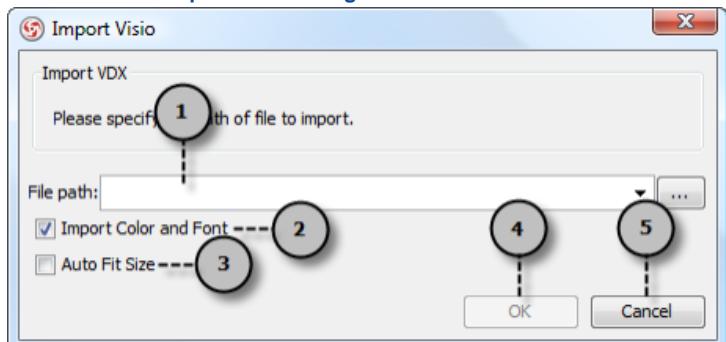
The left hand side of the dialog box represents the structure of Visio drawing, while the right hand side represents the expected outcome in VP-UML through importing. Users can perform the following actions in this dialog box.

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

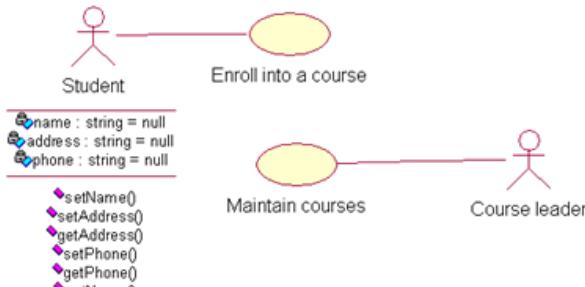
1 File path	The path of Visio .vdx file to be imported.
2 Import Color and Font	By selecting this option, colors and fonts of the shapes to be imported will remain unchanged. Otherwise, Visual Paradigm's default settings will be applied.
3 Auto Fit Size	By selecting this option, shapes' size will be 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

Importing Rational Rose model into VP-UML

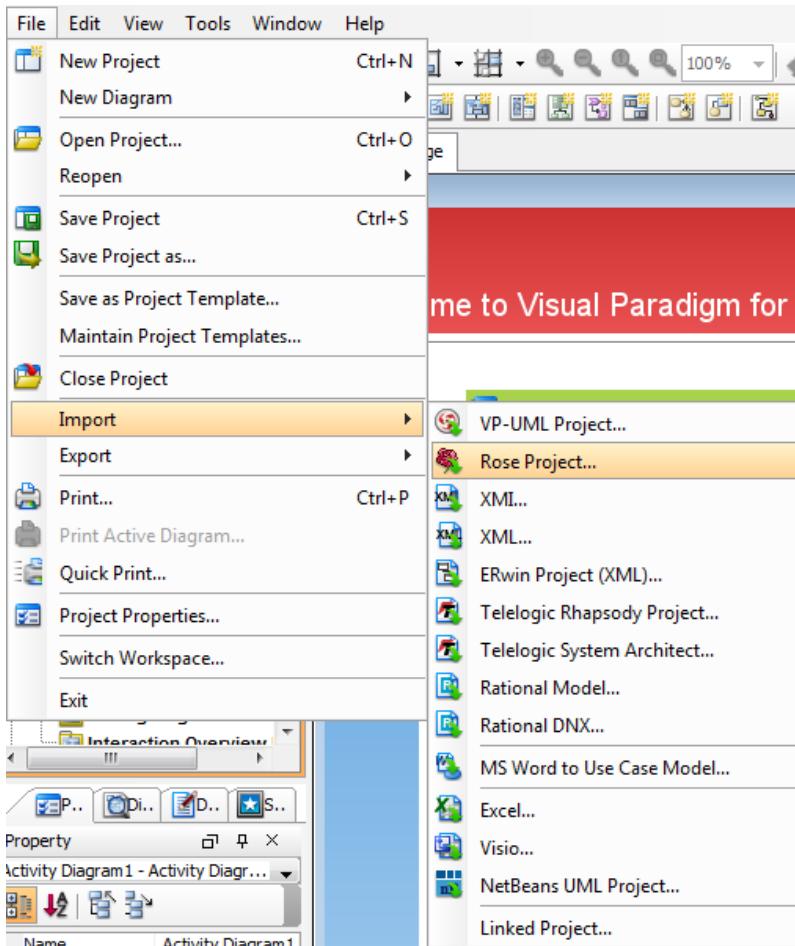
Rational Rose is one of the most widely used UML CASE Tool in the 90's. Visual Paradigm supports the importing of Rational Rose model. With this, users can import legacy design made in Rose into VP-UML, with all the model data as well as formatting retained.

1. Save your work in Rose.

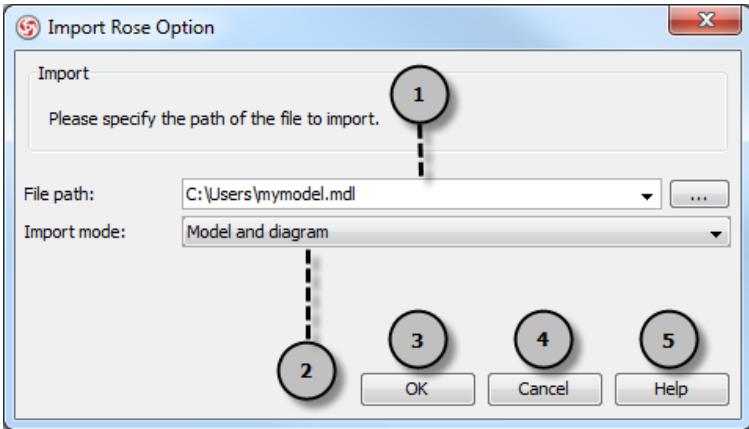


A Rose drawing

2. To import a Rose model into VP-UML, select **File > Import > Rose Project...** in the main menu of VP-UML.



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	OK	Click to import.
4	Cancel	Click to cancel importing.
5	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, click **Close** to close the progress dialog box. If **Model and diagram** was set for **Import mode**, the drawings can then be accessible in the **Diagram Navigator**.

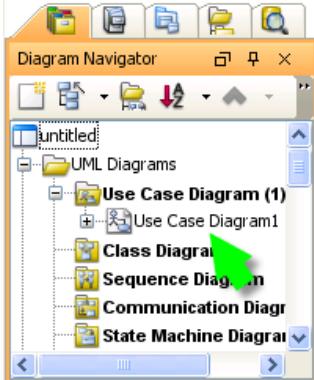
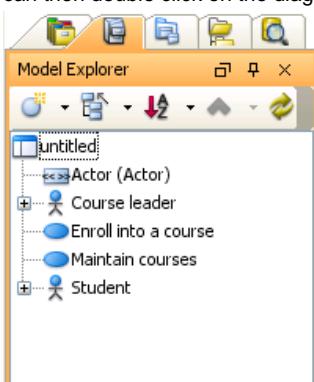


Diagram Navigator listing the imported diagram(s)

Model element can be accessed in the **Model Explorer**. User can form diagrams with them by dragging and dropping them onto diagrams. You can then double click on the diagram node to open the diagram.

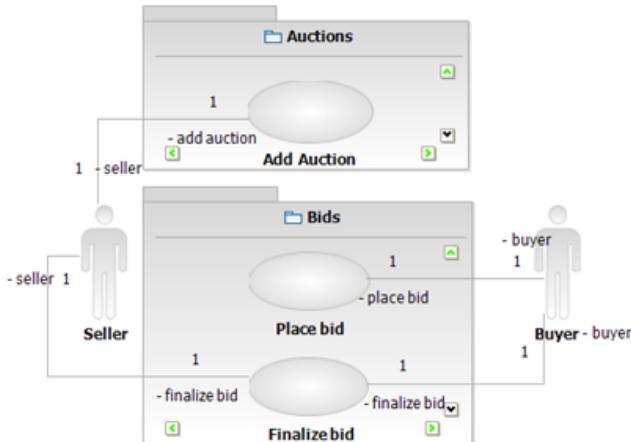


Model Explorer listing the imported model element(s)

Importing Rational Software Architect EMX into VP-UML

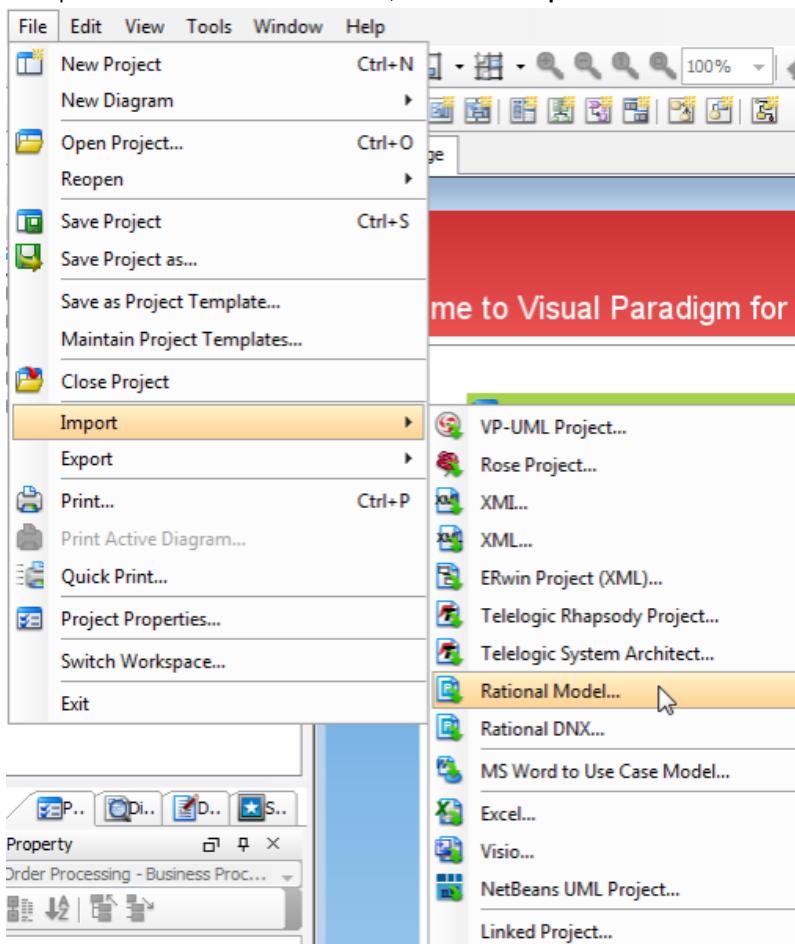
Rational Software Architect (RSA) is a modeling and development environment, which leverages UML for architectural design for C++ and Java 2 Enterprise Edition (Java2EE) applications and web services. Import of the RSA file, i.e. the .emx file, is supported in VP-UML so that users can simply migrate the work from RSA to VP, also perform further modeling on the imported models in the VP tool.

1. Save your work in Rational.



A Rational drawing

2. To import a Rational model into VP-UML, select **File > Import > Rational Model...** in the main menu of VP-UML.



The import Rational model menu

3. In the **Import Rational Software Architect UML Model** dialog box, specify the file path of the .emx file and click **OK**.

4. The drawings can then be accessible in the **Diagram Navigator**. You can then double click on the diagram node to open the diagram.

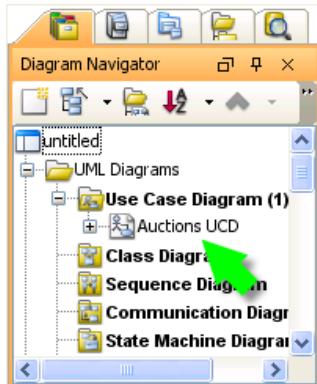
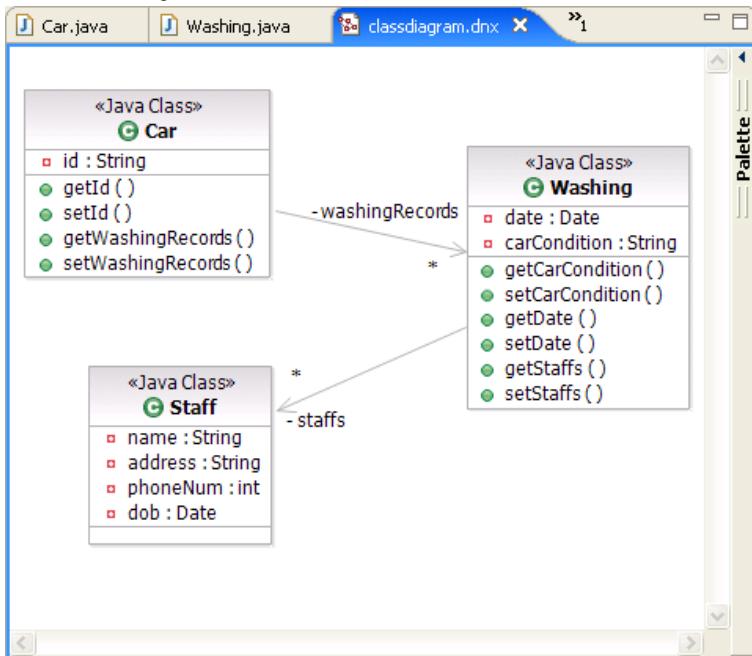


Diagram Navigator listing the imported diagram(s)

Importing Rational Software Architect DNX into VP-UML

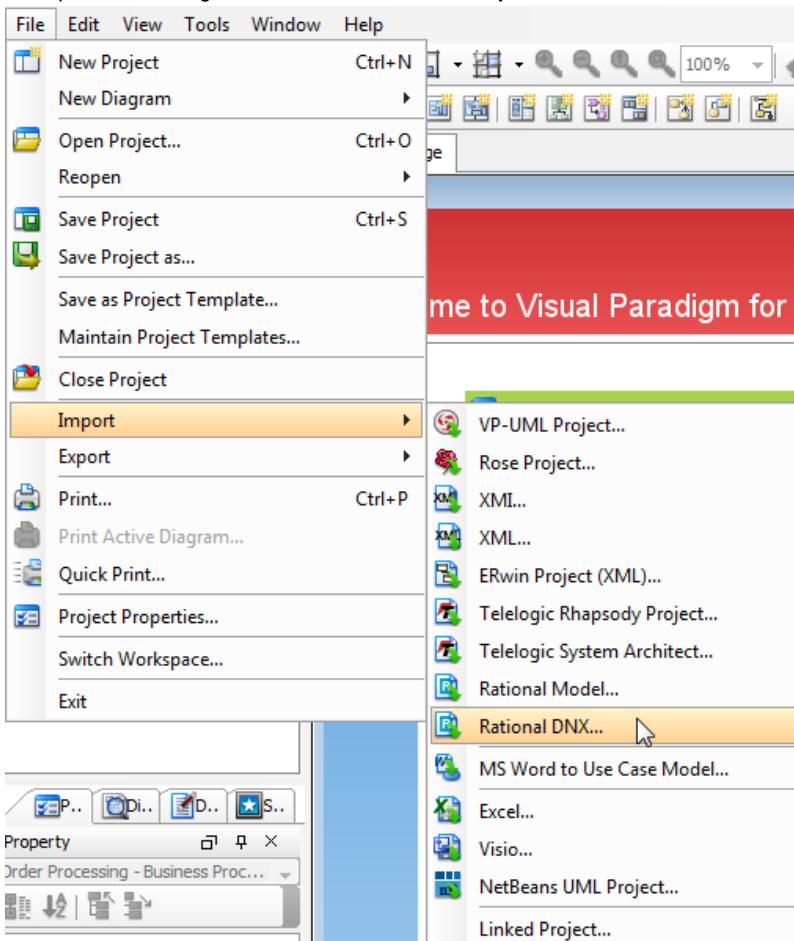
VP-UML supports importing drawing drew in Rational Software Architect with a .dnx extension. By importing a drawing, all diagrams, shapes and model information will be imported.

1. Save the drawing in Rational Software Architect.



A Rational drawing

2. To import the drawing into VP-UML, select **File > Import > Rational DNX...** in the main menu of VP-UML.



The import Rational DNX menu

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, you can then double click on the diagram node to open the diagram.

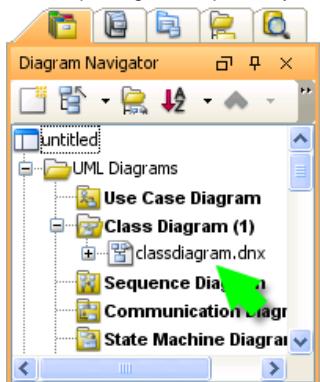
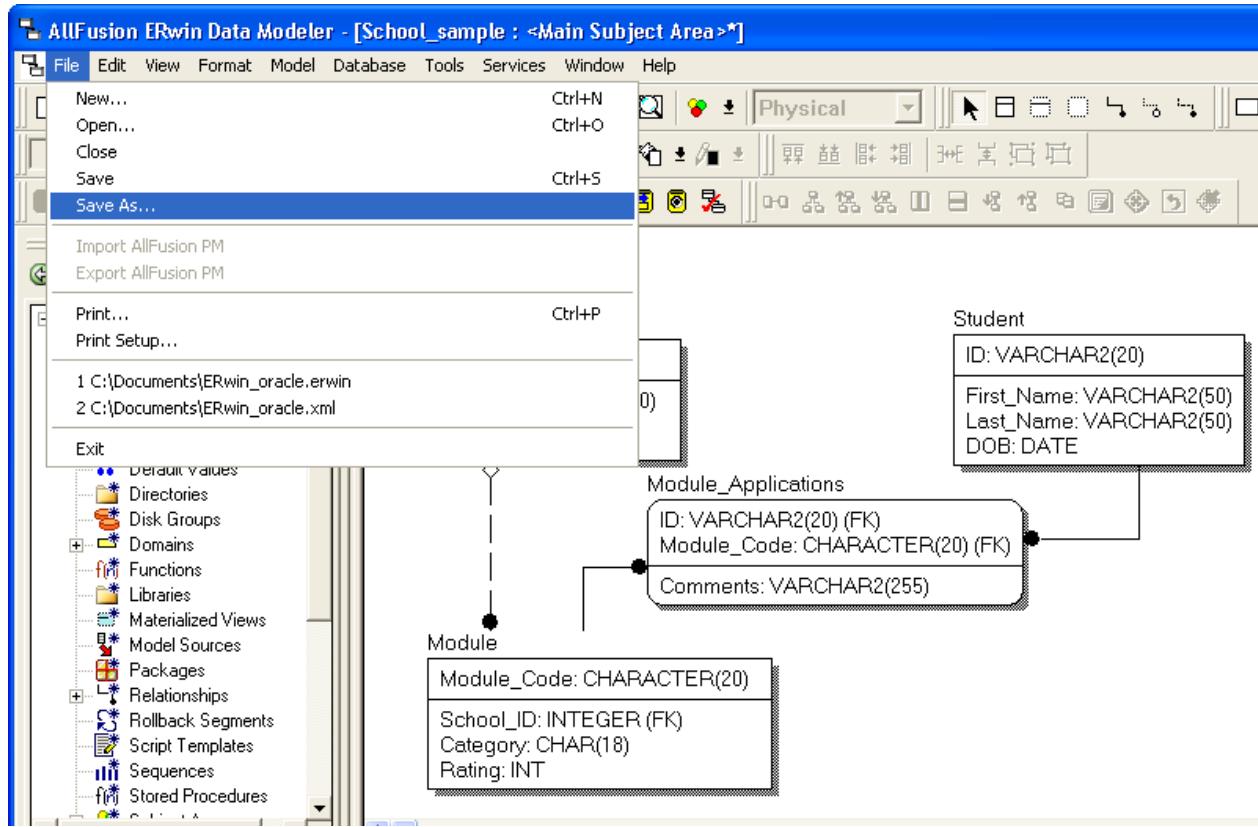


Diagram Navigator listing the imported diagram(s)

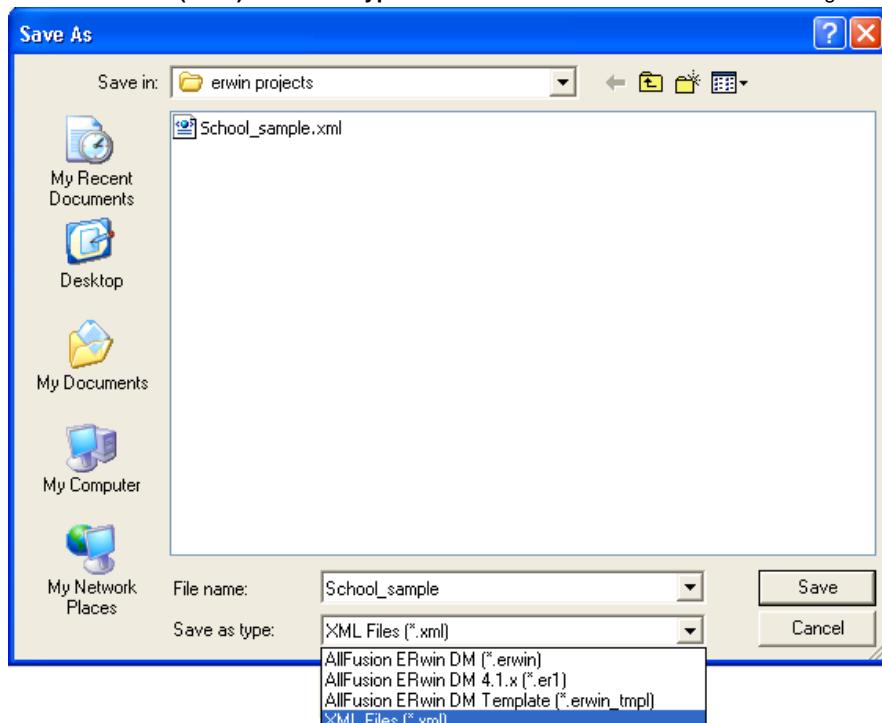
Importing ERwin data modeler project into VP-UML

AllFusion ERwin Data Modeler is a popular tool for data modeling. You can import ERwin diagrams and entity models into VP-UML with all properties preserved.

1. Here is a ERwin Data Modeler Project. In order to let VP-UML import it, you need to save it as an XML file. Select **File > Save As...** from the menu.



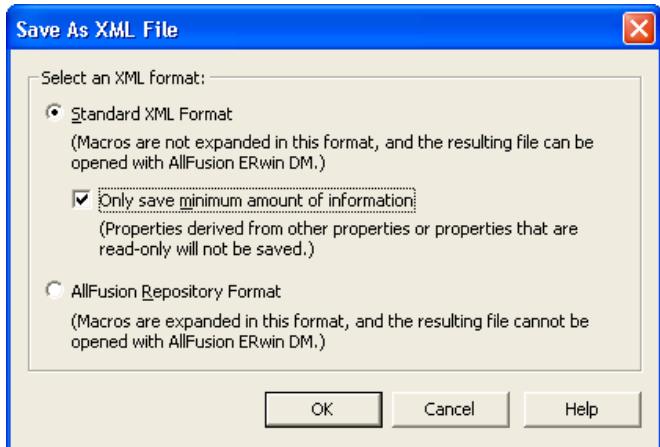
2. Select **XML Files (*.xml)** in **Save as type** and enter the file name in the **Save As** dialog box.



Save the ERwin Data Modeler project as XML

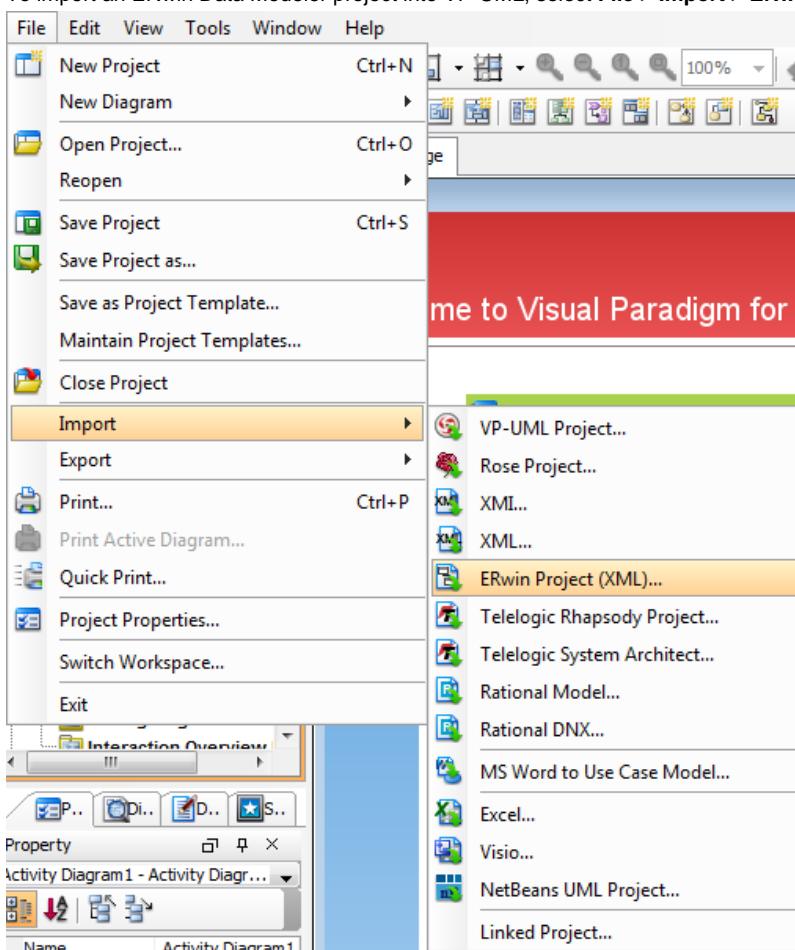
3. Click **Save**. This popups the **Save as XML File** dialog box.

4. Keep using the default settings **Standard XML Format** and **Only save minimum amount of information**.



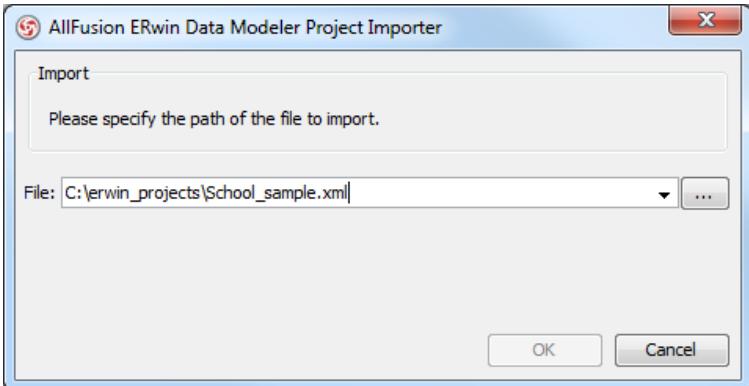
Exporting the XML in standard XML format

5. Click **OK** to confirm. This saves an XML file that can be used for importing into VP-UML.
 6. To import an ERwin Data Modeler project into VP-UML, select **File > Import > ERwin Project(XML)...** in the main menu of VP-UML.



The import ERwin Data Modeler Project menu

7. Specify the file path of the XML file.

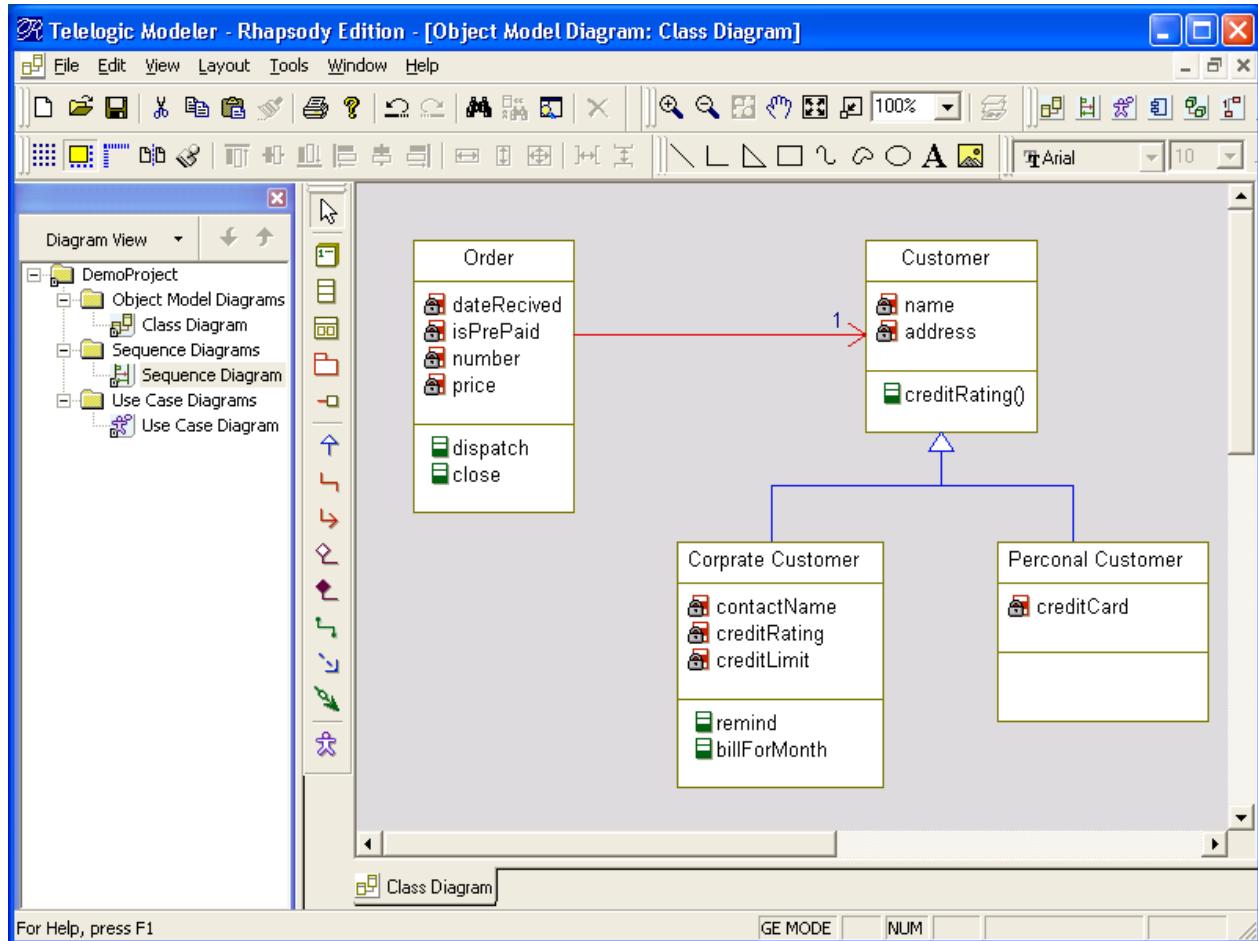


Specifying ERwin Data Modeler project file path

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.

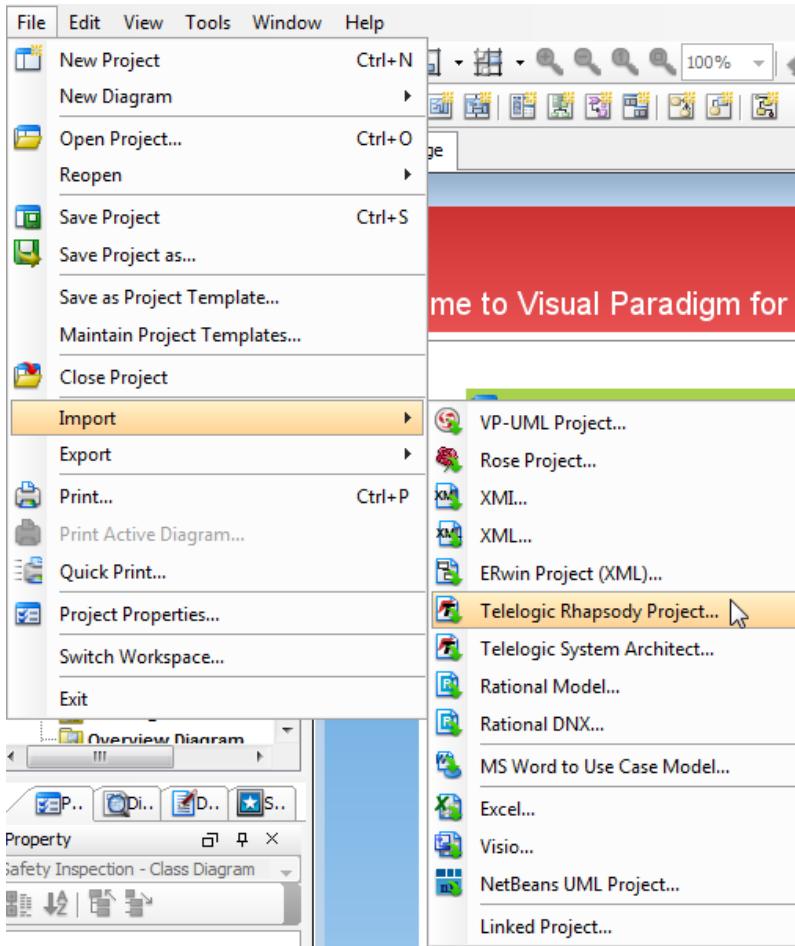
Importing Telelogic Rhapsody project into VP-UML

1. Save your Telelogic Rhapsody Project.



A drawing in Telelogic Modeler

2. To import a Telelogic Rhapsody project into VP-UML, select **File > Import > Telelogic Rhapsody Project ...** in the main menu of VP-UML.



The import Telelogic Rhapsody Project menu

3. Specify the file path of the Telelogic Rhapsody Project in the pop-up **Import Rhapsody** dialog box.
 4. 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.

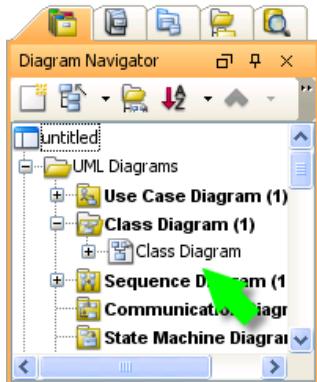
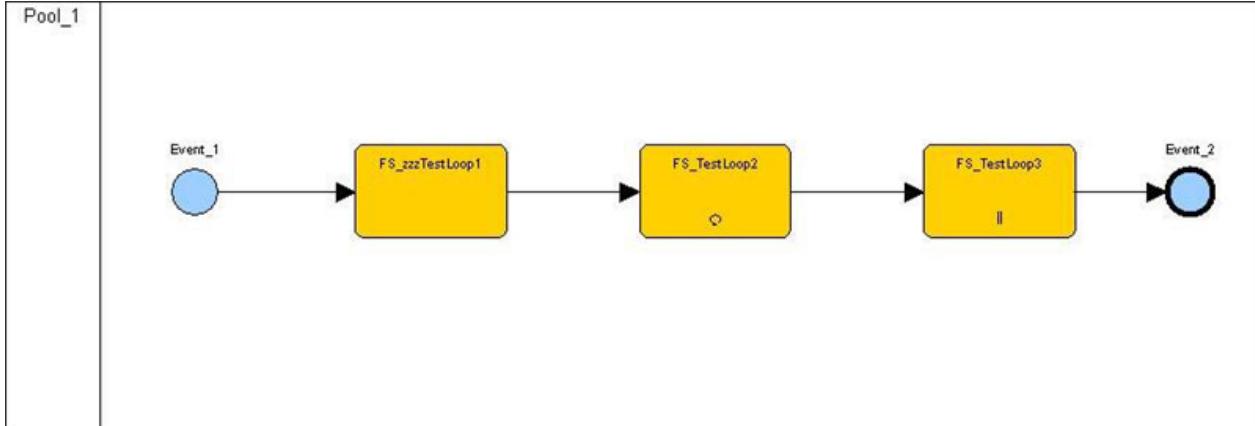


Diagram Navigator listing the imported diagram(s)

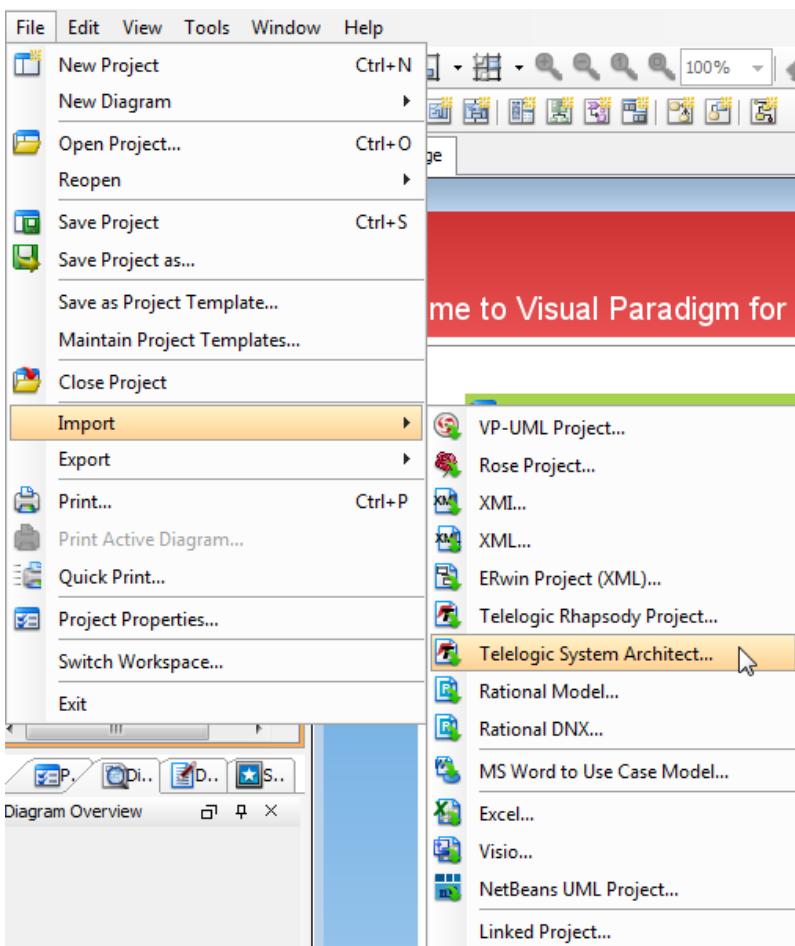
Importing Telelogic System Architect into VP-UML

1. Save your Telelogic System Architect drawing:



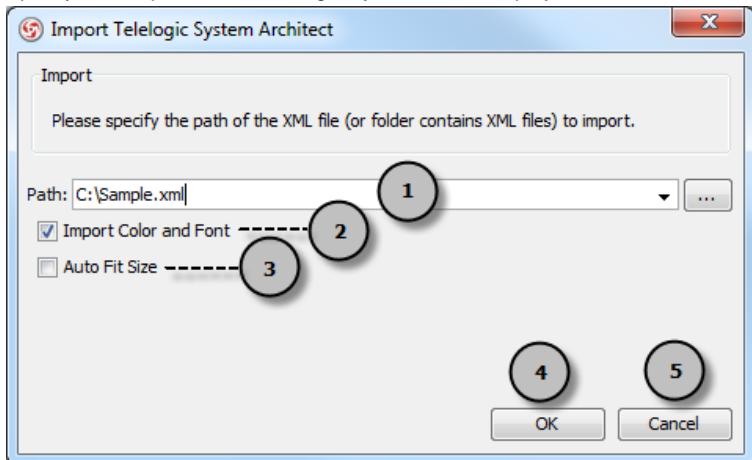
A Telelogic System Architect drawing

2. To import a Telelogic System Architect project into VP-UML, select **File > Import > Telelogic System Architect ...** in the main menu of VP-UML.



The import Telelogic System Architect menu

3. Specify the file path of the Telelogic System Architect project.



Specifying Telelogic System Architect path

No.	Name	Description
1	File path	The path of Telelogic 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 Telelogic System Architect.
5	Cancel	Click Cancel to discard importing XML.

Description of Import Telelogic 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. The drawings can then be accessible in the **Diagram Navigator**. You can then double click on the diagram node to open the diagram.

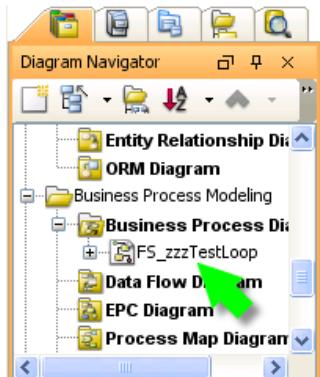


Diagram Navigator listing the imported diagram(s)

1. Here is a NetBeans UML Diagram:

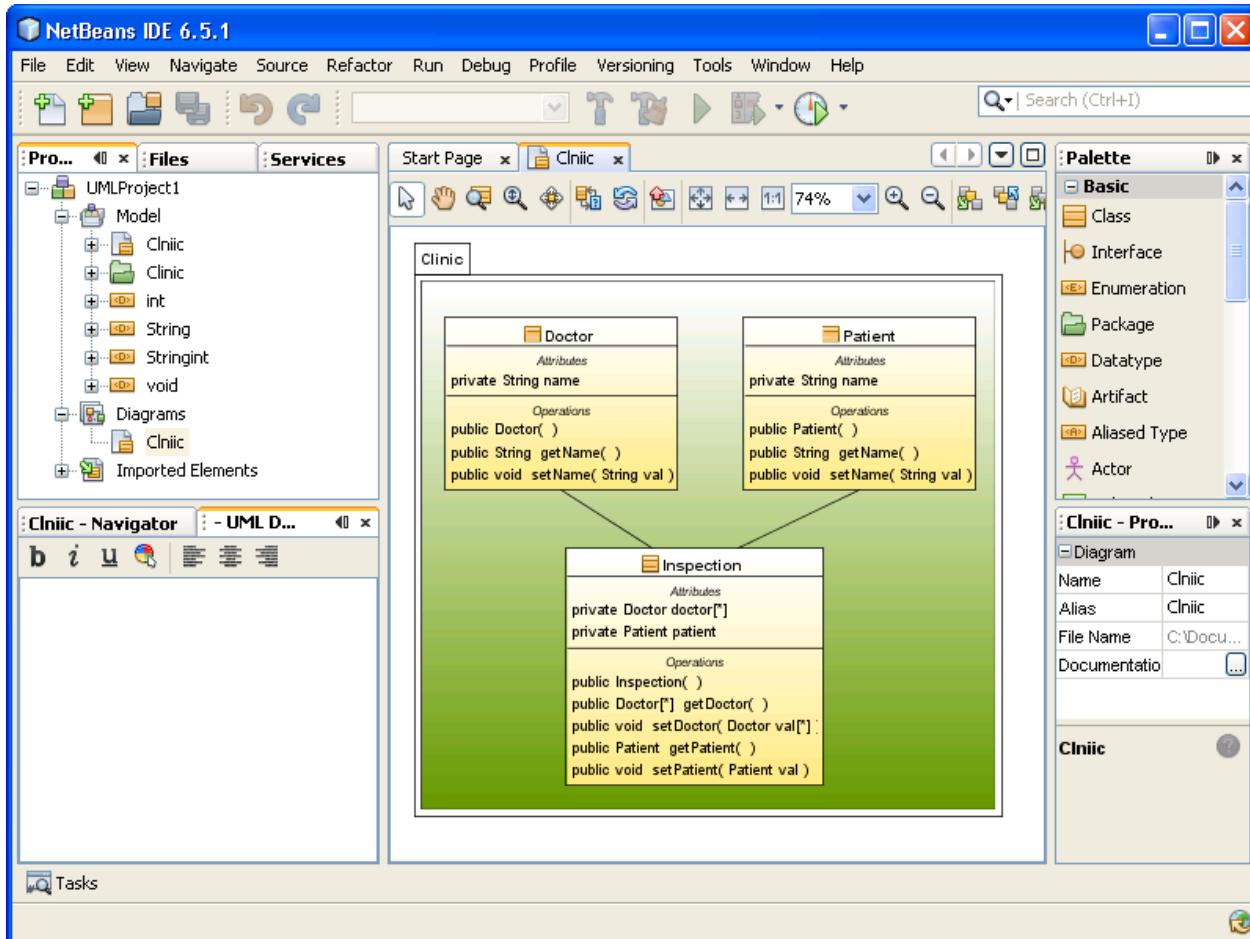


Figure 10-1 A Class Diagram drew in NetBeans

To import a NetBeans UML project into VP-UML, select **File > Import > NetBeans UML Project...** in the main menu of VP-UML.

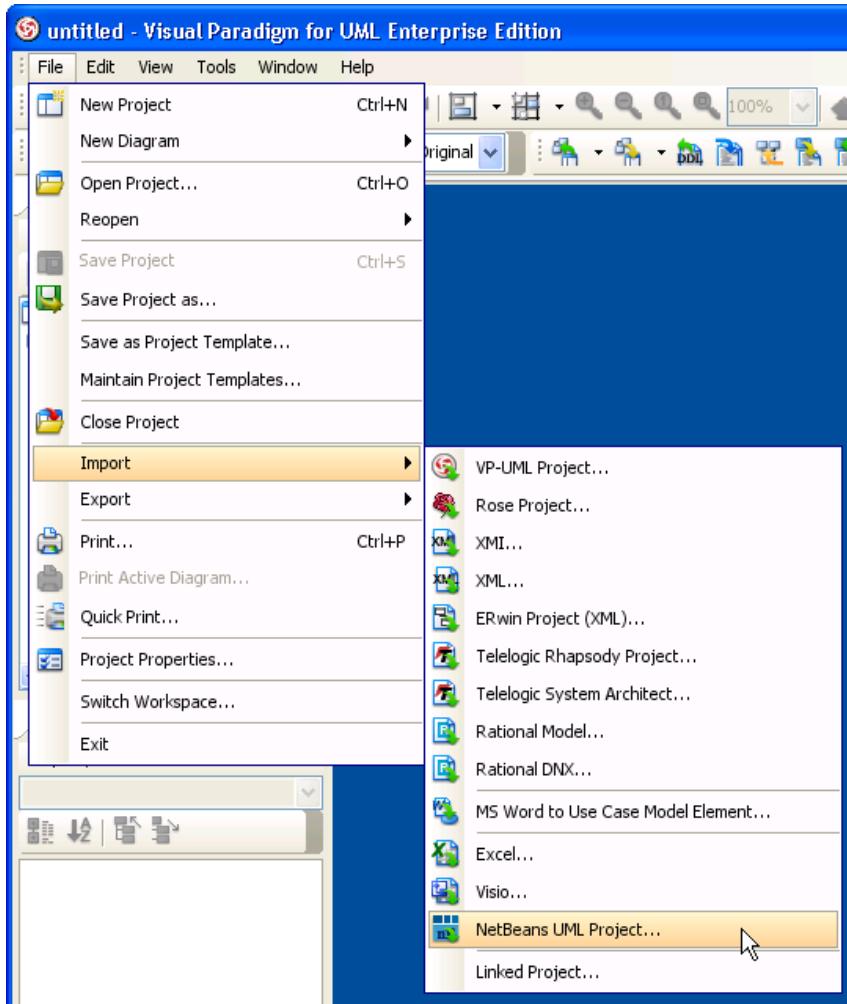


Figure 10-2 The import NetBeans UML Project menu

2. Specify the file path of the NetBeans project.

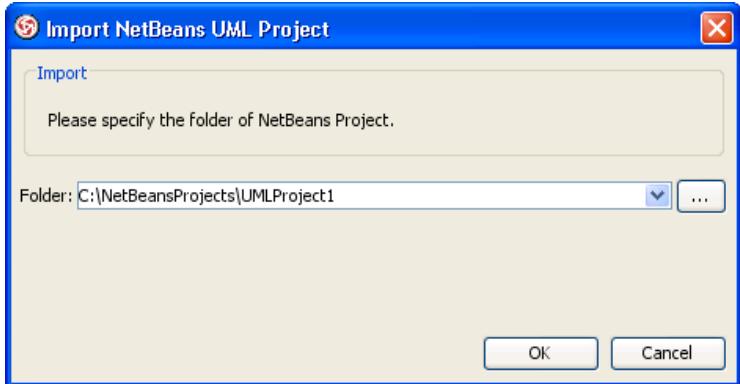


Figure 10 -3 Specifying NetBeans UML Project path

3. Click **OK** to start importing. When import is completed, the message pane will popup, with a notification appear in it.

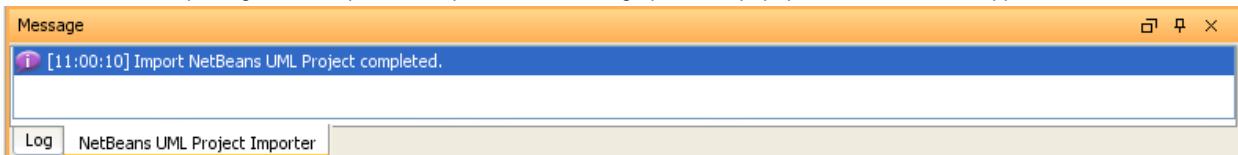


Figure 10 -4 The dialog box indicating the progress of importing

The drawings can then be accessible in the **Diagram Navigator**.

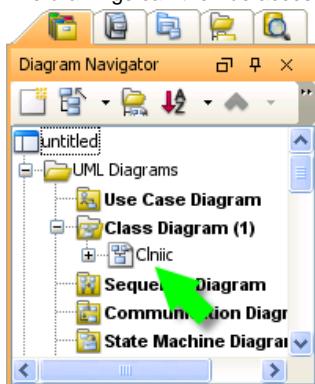


Figure 10 -5 Diagram Navigator listing the imported diagram(s)

You can then double click on the diagram node to open the diagram.

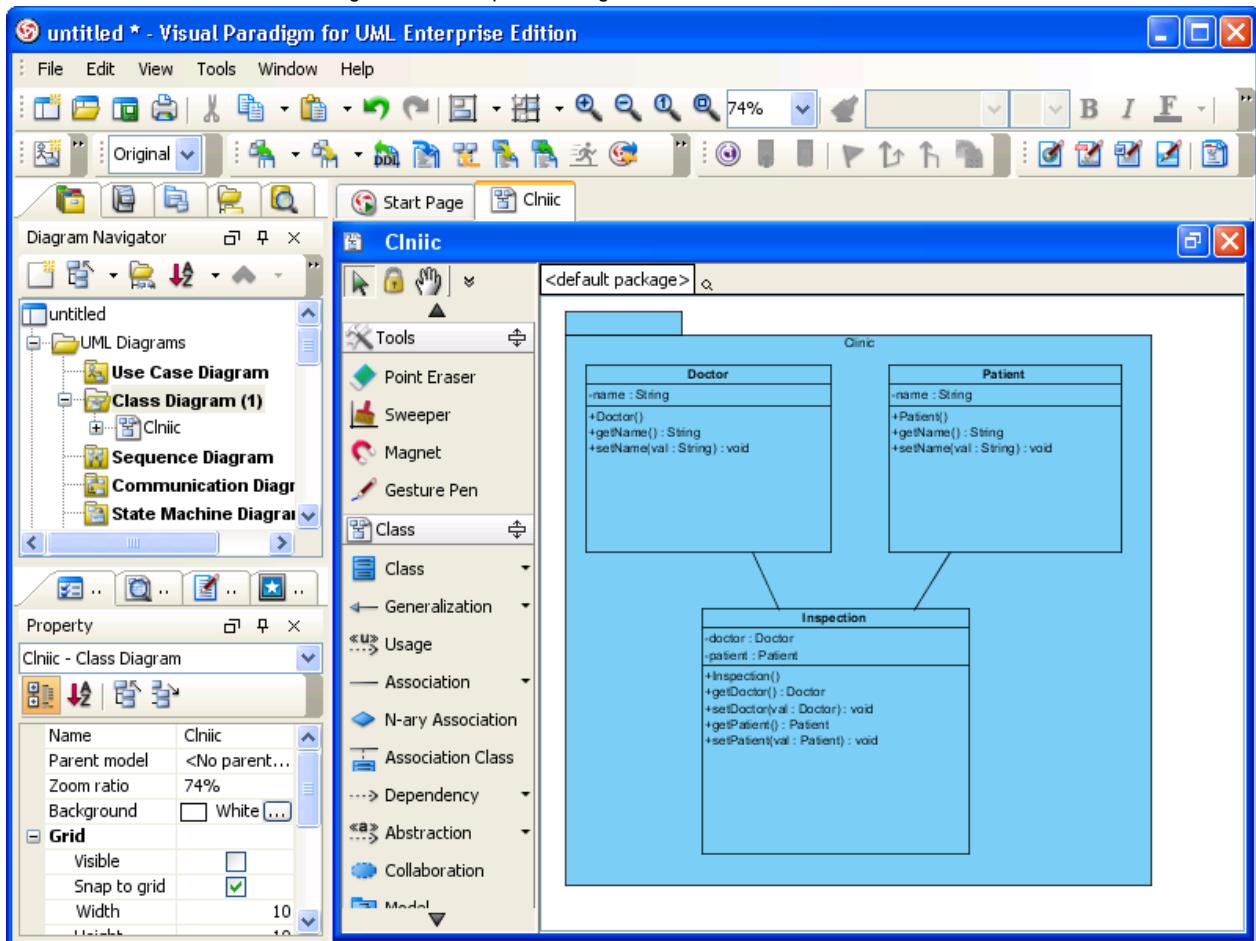


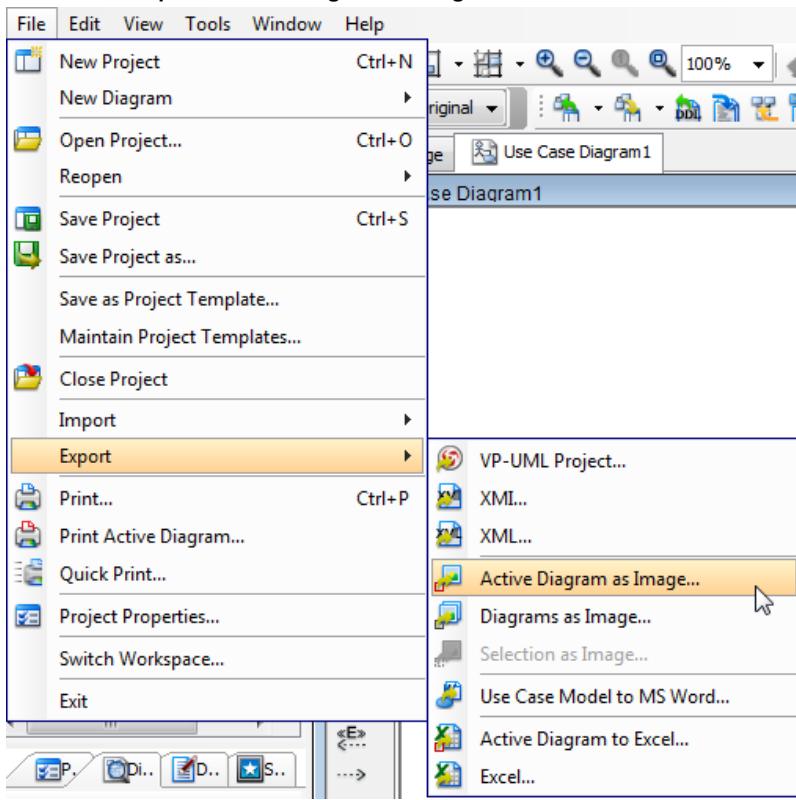
Figure 10 -6 A Class Diagram imported from NetBeans UML Project

NOTE: Due to different ways in presenting diagrams in VP-UML and NetBeans, the imported shapes may contain unnecessary spaces in them. 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 on a diagram, right click on the diagram background and select **Diagram Content > Auto Fit Shapes Size** in the popup menu.

Exporting active diagram as image

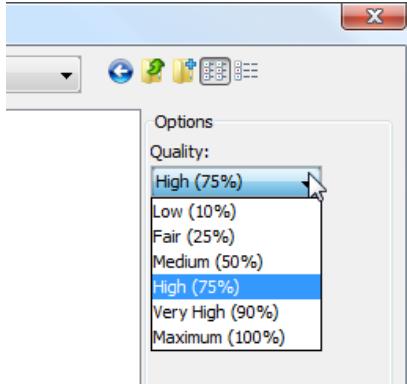
You can export the opening diagram to image file. To export the active diagram as an image file:

1. Select **File > Export > Active Diagram as Image...** from main menu.



Export active diagram as image

2. 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

3. 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 will export diagram's background color. Without diagram will ignore the background color by exporting transparent background.

NOTE: You can export VP-UML 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.

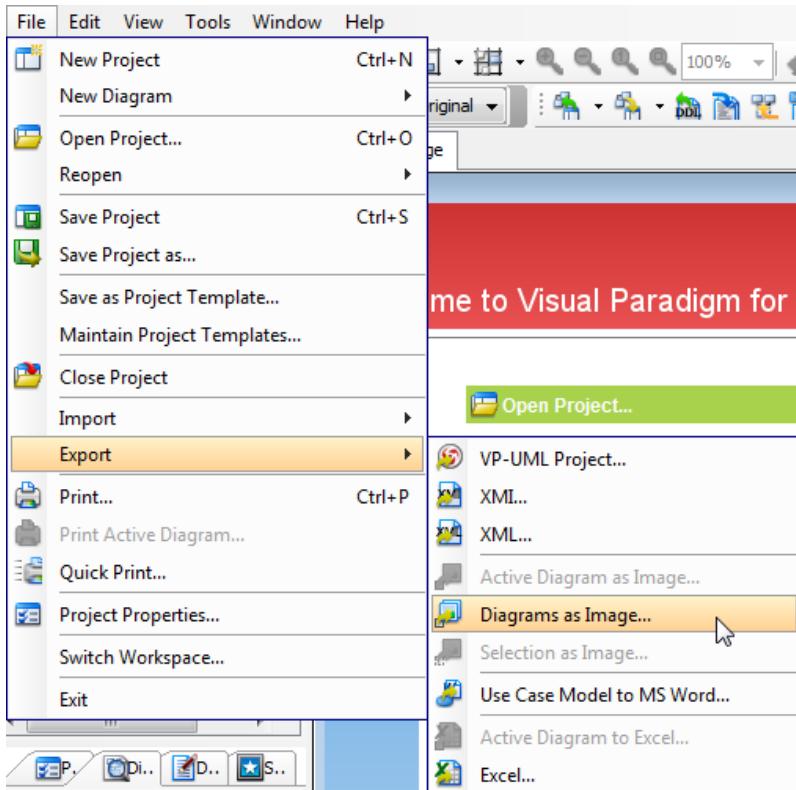
4. Specify the filename.
5. Click **Save** to export.

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

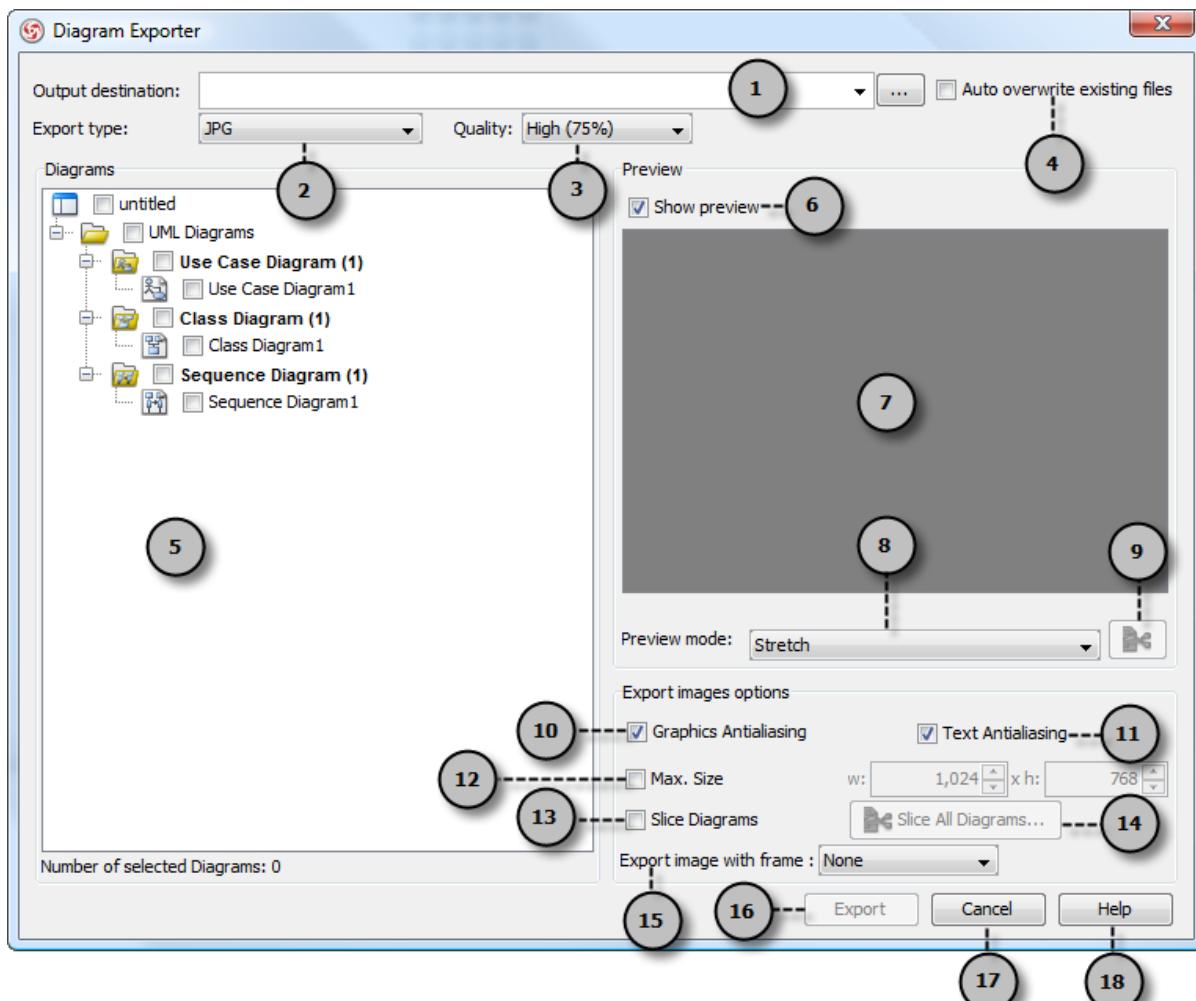
1. Select **File > Export > Diagrams as Image...** from main menu.



To export diagrams as images

2. In the **Diagram Exporter** dialog box, select the diagram(s) to export.
3. Specify the output destination for storing the image files.
4. Click **Export** to export the diagrams.

An overview of Diagram Exporter



An overview of **Diagram Exporter** dialog box

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 VP-UML 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 more clear, 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 max. 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 print 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.

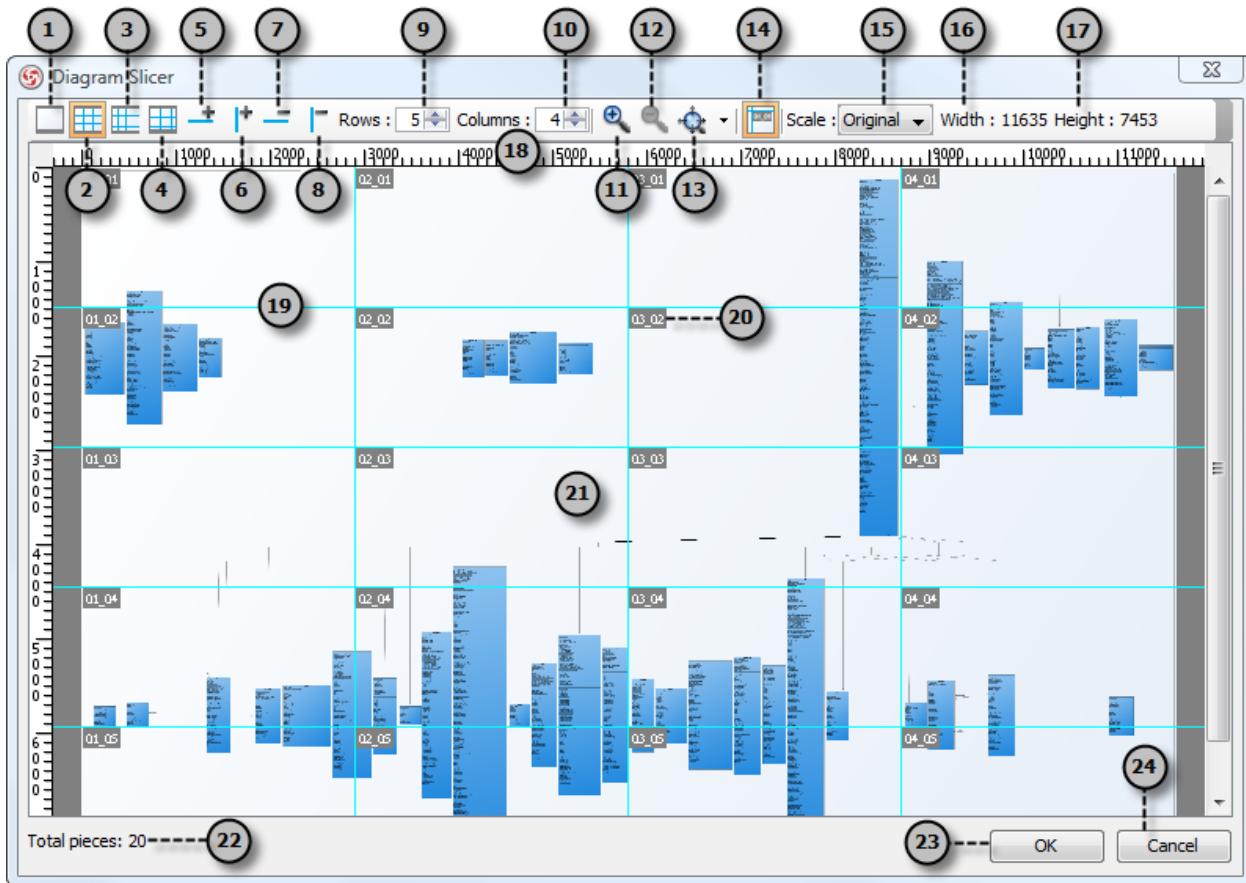
*Description of **Diagram Exporter** dialog box*

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 dialog box

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** dialog box. 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 dialog box

Below is the description of different parts of the dialog box.

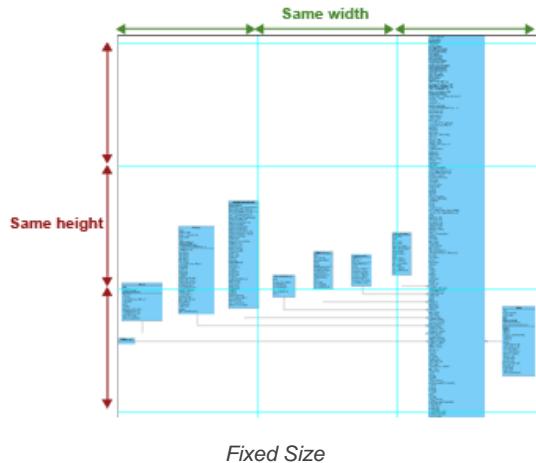
No.	Name	Description
1	No slicing	Do not slice diagram.
2	Fixed size	A simple strategy which slice exported diagram into pieces that have the same size.
3	Free slicing	User can customize how to slice the exported diagram by specifying the position of vertical slices and horizontal slices.
4	Fixed ratio	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	Demagnify the diagram content.
13	Zoom fit to screen	Zoom 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. They 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.

Slicing strategies

There are three slicing strategies - **Fixed Size**, **Free Slicing** and **Fixed Ratio**. Each gives a distinct way of slicing images.

Fixed size

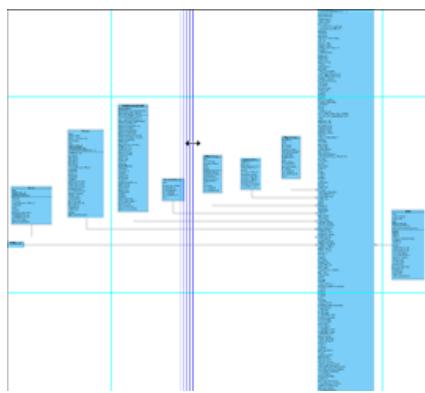
Fixed Size is a simple strategy which slice exported diagram into pieces that have the same size. User specifies the number of columns and rows to slice and then the exported diagram will be sliced into specific pieces.



Fixed Size

Free slicing

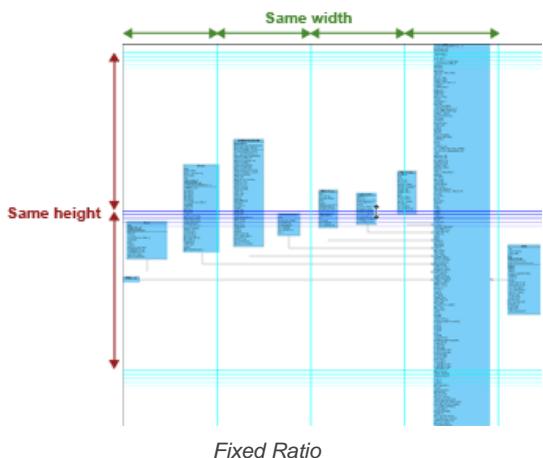
User can customize how to slice the exported diagram by specifying the position of vertical slices and horizontal slices. It is particularly useful to prevent a shape from being sliced into pieces.



Free Slicing

Fixed ratio

Fixed Ratio strategy gain 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.

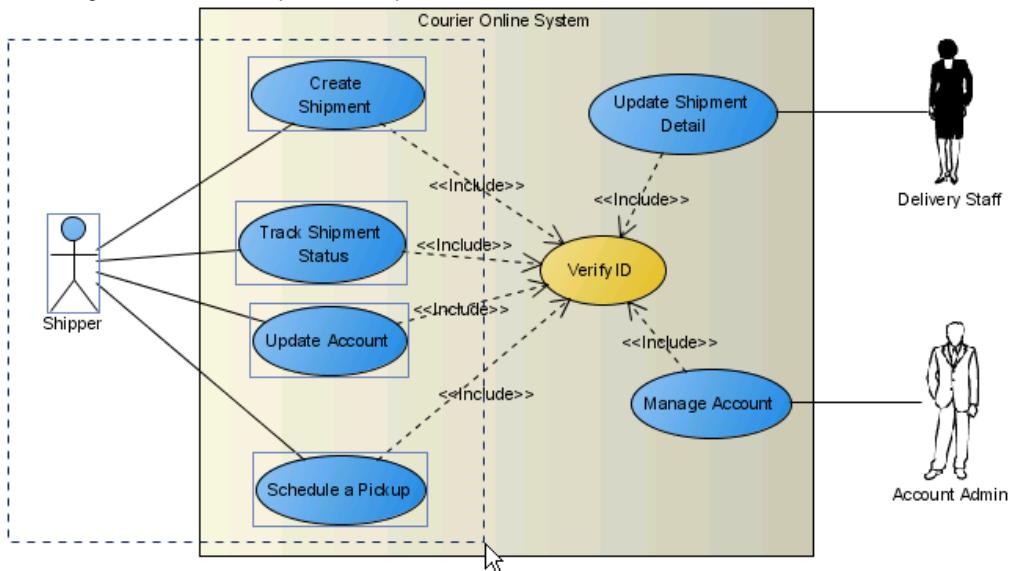
Control by ratio

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.

Exporting portion of diagram as image

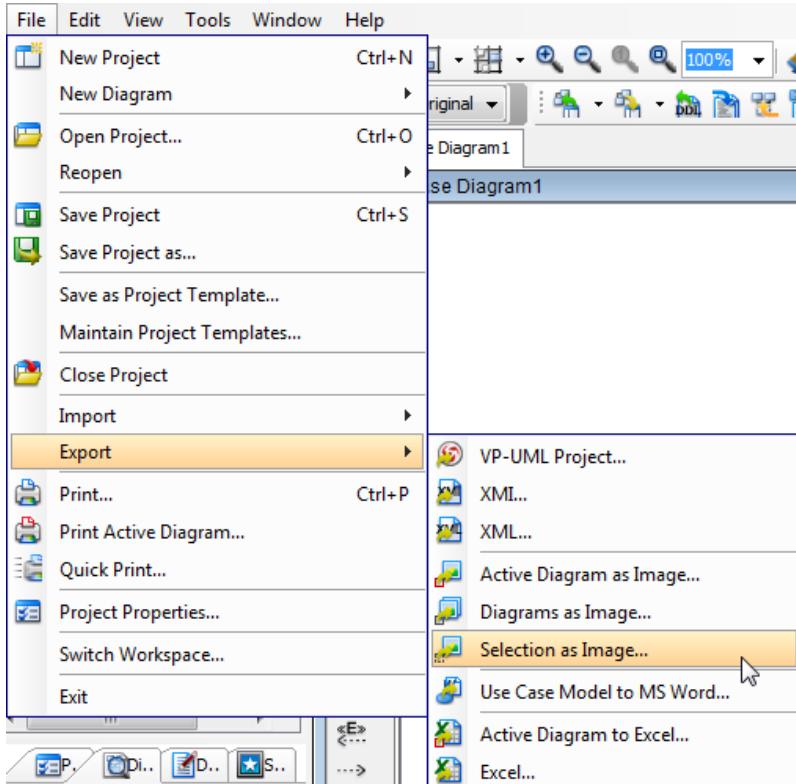
You can export some shapes in a diagram as an image file by selecting the shapes you want to export then perform export. To export selected shapes to image:

1. On a diagram, select the shapes to be exported.



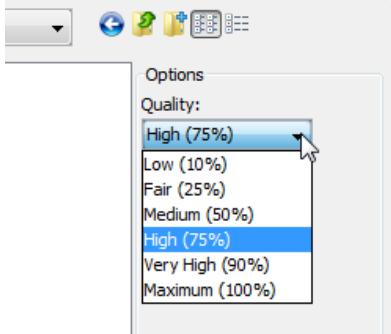
Selecting shapes to export as image

2. Select **File > Export > Selection as Image...** from main menu.



Export selected shapes as image

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 will export diagram's background color. Without diagram will ignore the background color by exporting transparent background.

NOTE: You can export VP-UML 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.

5. Specify the filename of the image file.

6. Click **Save** to export.

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 diagrams which model how to achieve the user goals. 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 avoid unexpected consequences resulted by updating your design blueprint. Contrary to this, it lets you find out everything you need to update along when a change is to be made.

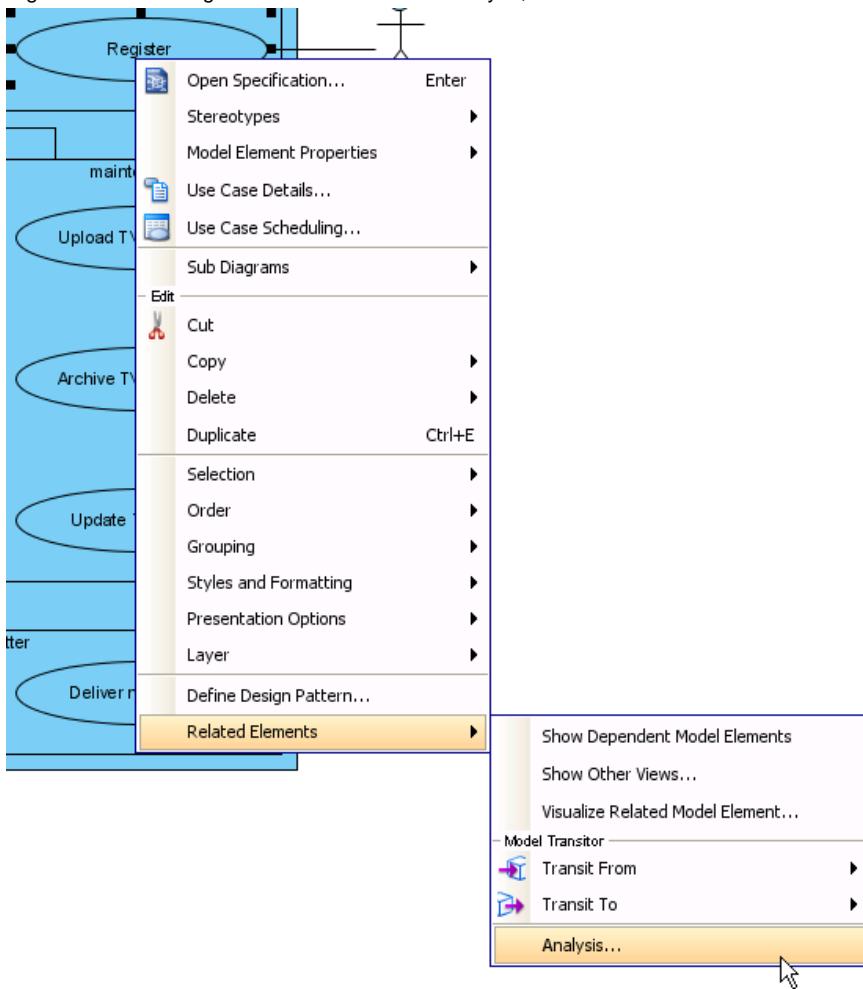
Analyzing a model element

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:

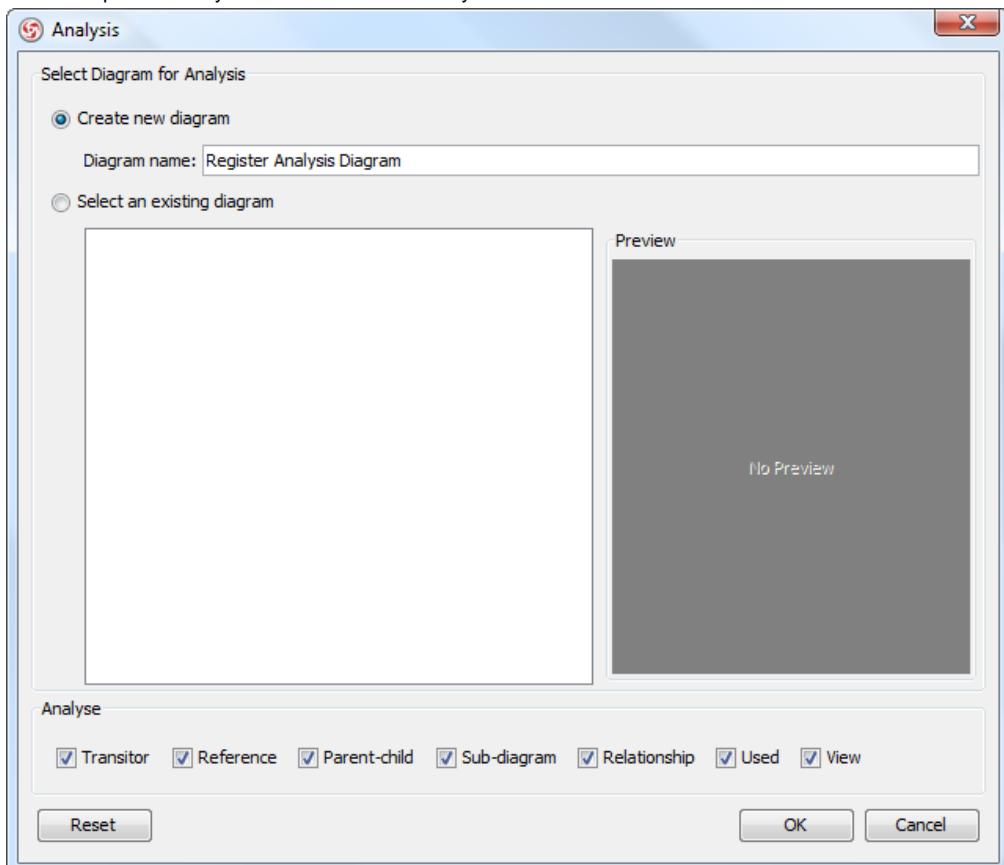
1. Right click on the diagram element we want to analyze, and select **Related Elements > Analysis...** from the popup menu.



To analyze a diagram element

NOTE: You can analyze a model element in Model Explorer by right clicking on the desired element node in Model Explorer, and selecting **Analysis...** from the popup menu.

2. The result of analysis will be presented in analysis diagram. In the **Analysis** dialog box, 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.



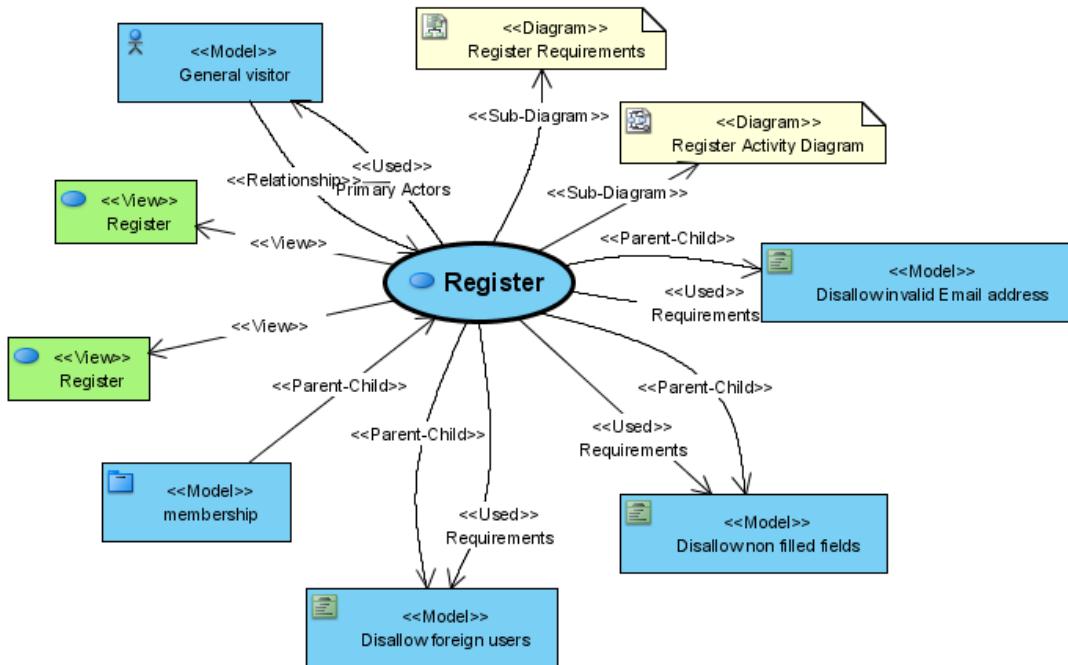
Create a diagram, or select an existing analysis diagram to present the result

Type	Description
Transistor	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.



The oval node at the center of 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.

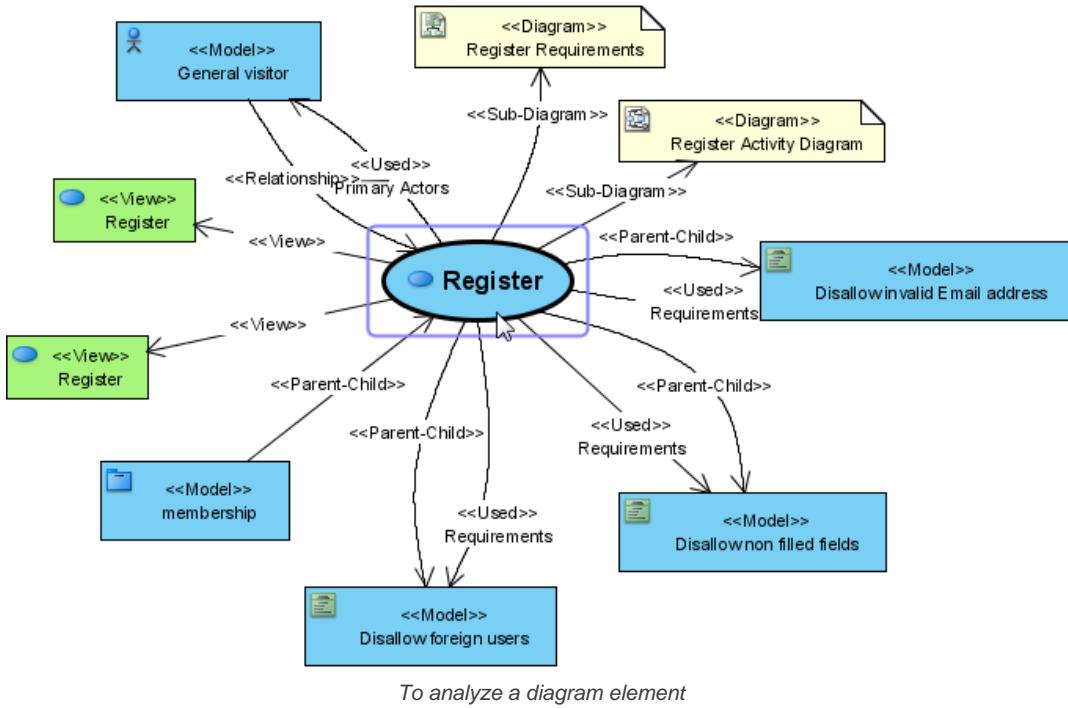
Inside a node of a related element, you can see a tag (e.g. <<View>>), which represents the type of that related element. At the bottom part of a node box is the name of the related element.

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.

Updating analyzed result

Once a model is refined, the previous analysis result may no longer reflecting 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.

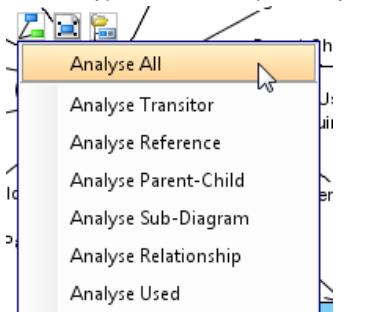


NOTE: Besides the central node, you may update/show the relationships of other Model nodes on diagram, too.

2. Click on the **Analyze** resource icon.



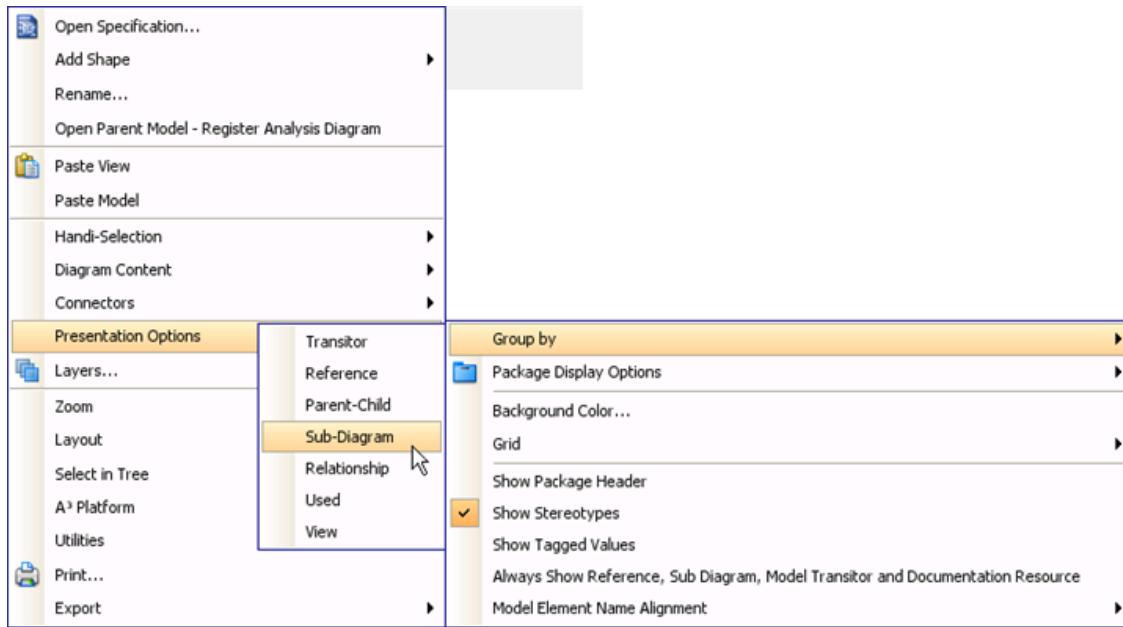
3. Select a type of relationship to analyze.



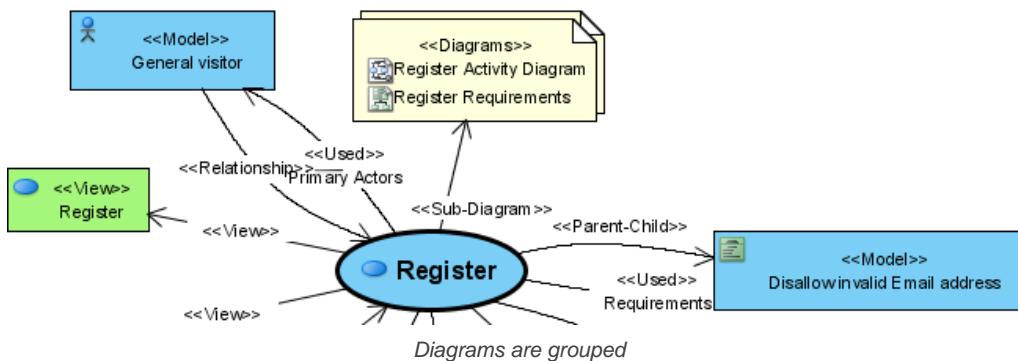
To analyze all kinds of relationships

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.



Nodes of same type are grouped.



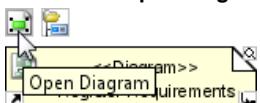
To ungroup, simply deselect the node type by walking through the same popup menu path.

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 nodes.

Opening a diagram of diagram node

1. Move the mouse pointer over a **Diagram** node.
2. Click on the **Open Diagram** resource icon.

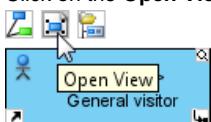


To open diagram of Diagram node

This opens the diagram.

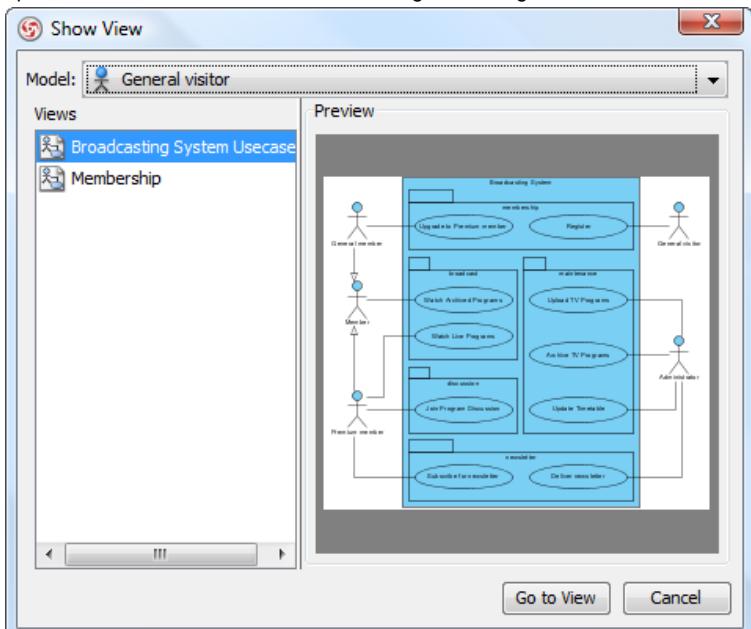
Opening a view of model node

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 are multiple views, the **Show View** dialog box is presented. Select a view to open, and click **Go to View** at the bottom right of dialog box.

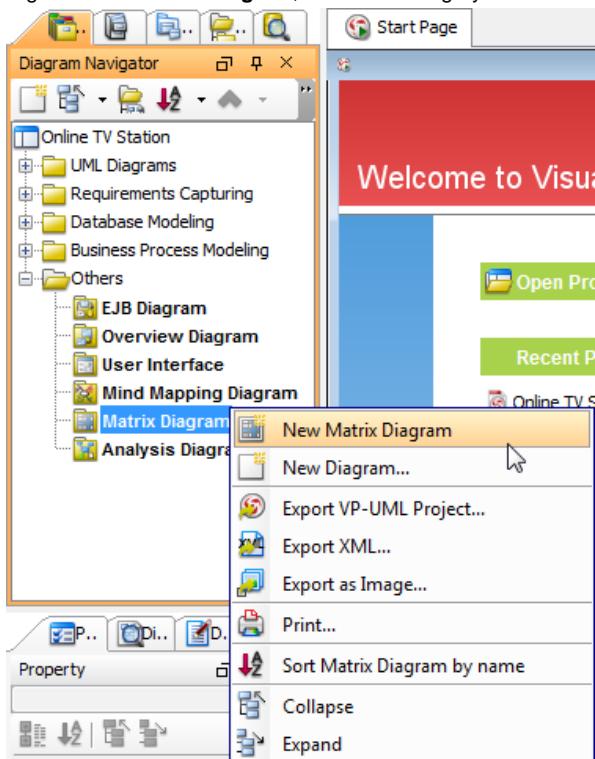


The **Show View** dialog box

Creating a matrix

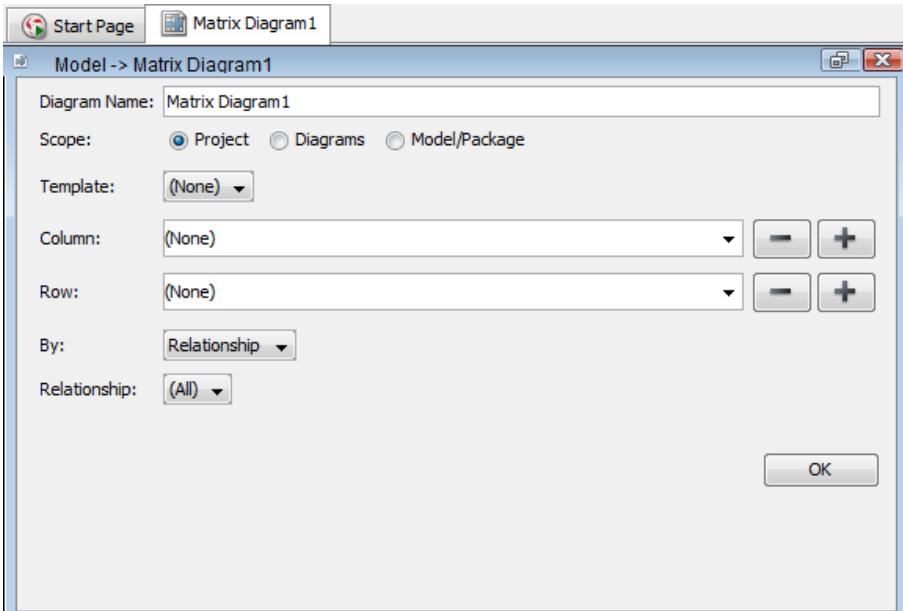
A matrix is a table, which shows the existence of relationships among model elements of particular types. By reading a matrix, you can tell easily whether two model elements are related or not, and what kind of relationship do they have.

1. Right click on **Matrix Diagram**, under the category **Others** in **Diagram Navigator**, and select **New Matrix Diagram** from the popup menu.



To create a new matrix diagram

2. Configure the matrix.



Configuring 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, either in Project (all model elements), Diagram (only model elements in specific diagrams, to be chosen by user) or under Model/Package.
Template	Template offers a default setup to Column, Row and By. It is available according to the project content. For example, template "Use Case <-> Requirement" appear for selection when a project have use case and requirement.
Column	The type of model element to list at the column side of matrix. In order to list multiple types of model element, click on the + button at the left hand side to popup another entry of column.
Row	The type of model element to list at the row side of matrix. In order to list multiple types of model element, click on the + button at the left hand side to popup another entry of row.
By	<p>The way how matrix will match against rows and columns.</p> <p>Sub Diagram - The column/row model element is placed in a sub diagram of the matching model element</p> <p>Child - The column/row model element is a child of the matching model element</p> <p>Relationship - The column/row model element is related with the matching model element</p> <p>Reference - The column/row model element is referencing the matching model element</p> <p>As Classifier - The column/row model element takes the matching model element as classifier</p> <p>Dependent - The column/row model element has properties that depend on the matching model element</p>

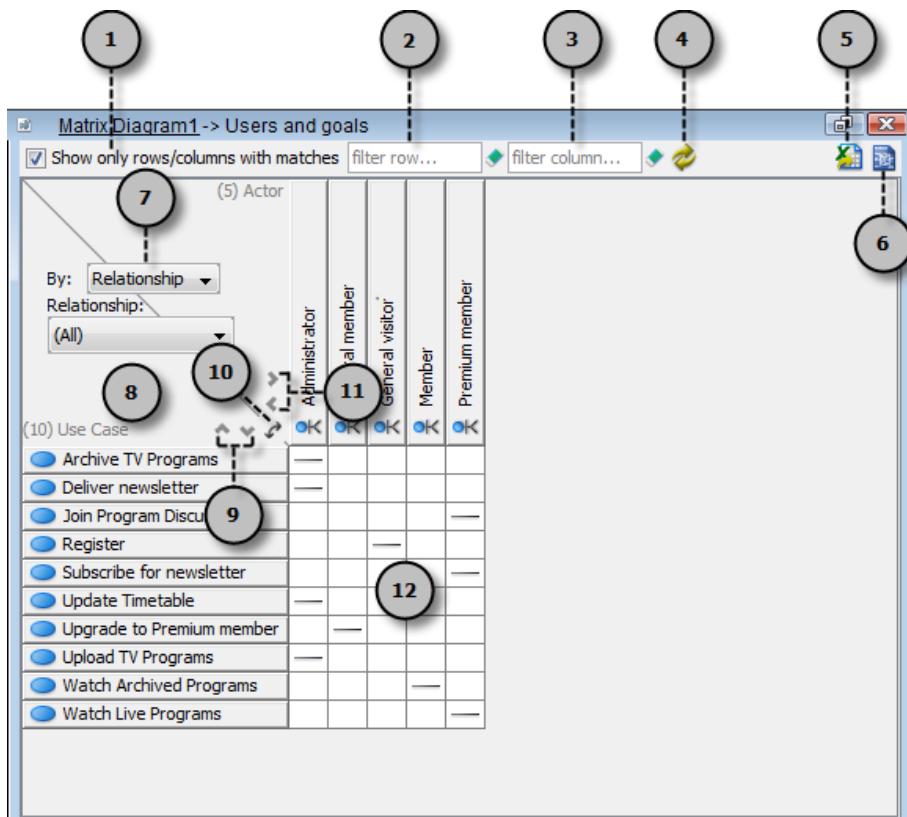
Fields for matrix configuration

3. Click **OK** to form the matrix.

Reading a matrix

Knowing how to read a matrix helps you 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 representing 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	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.
2	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.
3	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.
4	Refresh	You can update the content of matrix by clicking this button manually, for reflecting the changes you have made in models.
5	Export to Excel	Click this button to export the opening matrix to Excel.
6	Configure	Click this button to configure the content to display in matrix.
7	By	It shows how matrix will match against rows and columns.
8	Rows	The model elements have been chosen will be displayed in rows.
9	Move up and move down	Rows and columns are ordered alphabetically by default. They help to re-order rows and columns in moving vertically.
10	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.
11	Move left and move right	Rows and columns are ordered alphabetically by default. They help to re-order rows and columns in moving horizontally.
12	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.



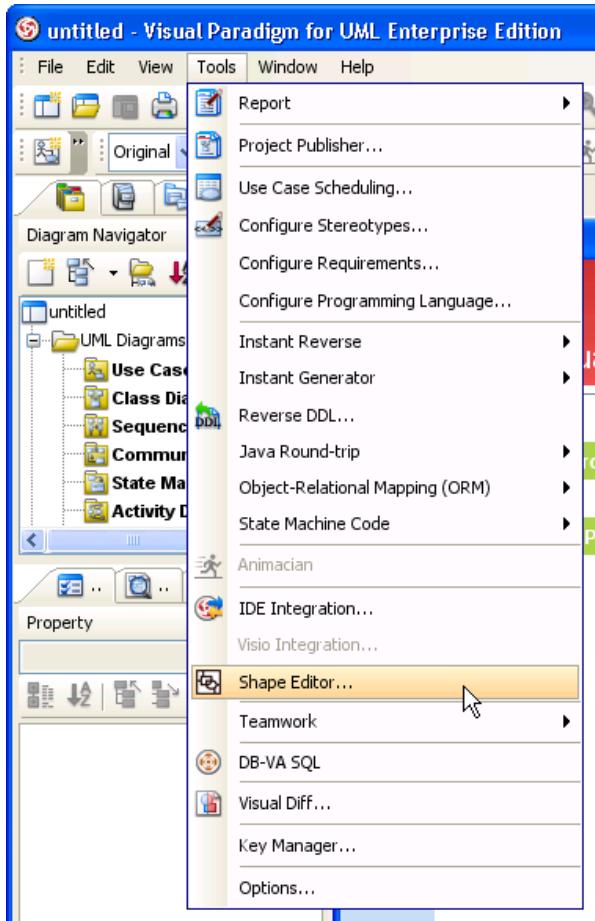
To Click Export to Excel

2. In the **Export Excel** dialog box, specify the output destination and click **Save**.

Creating shape in shape editor

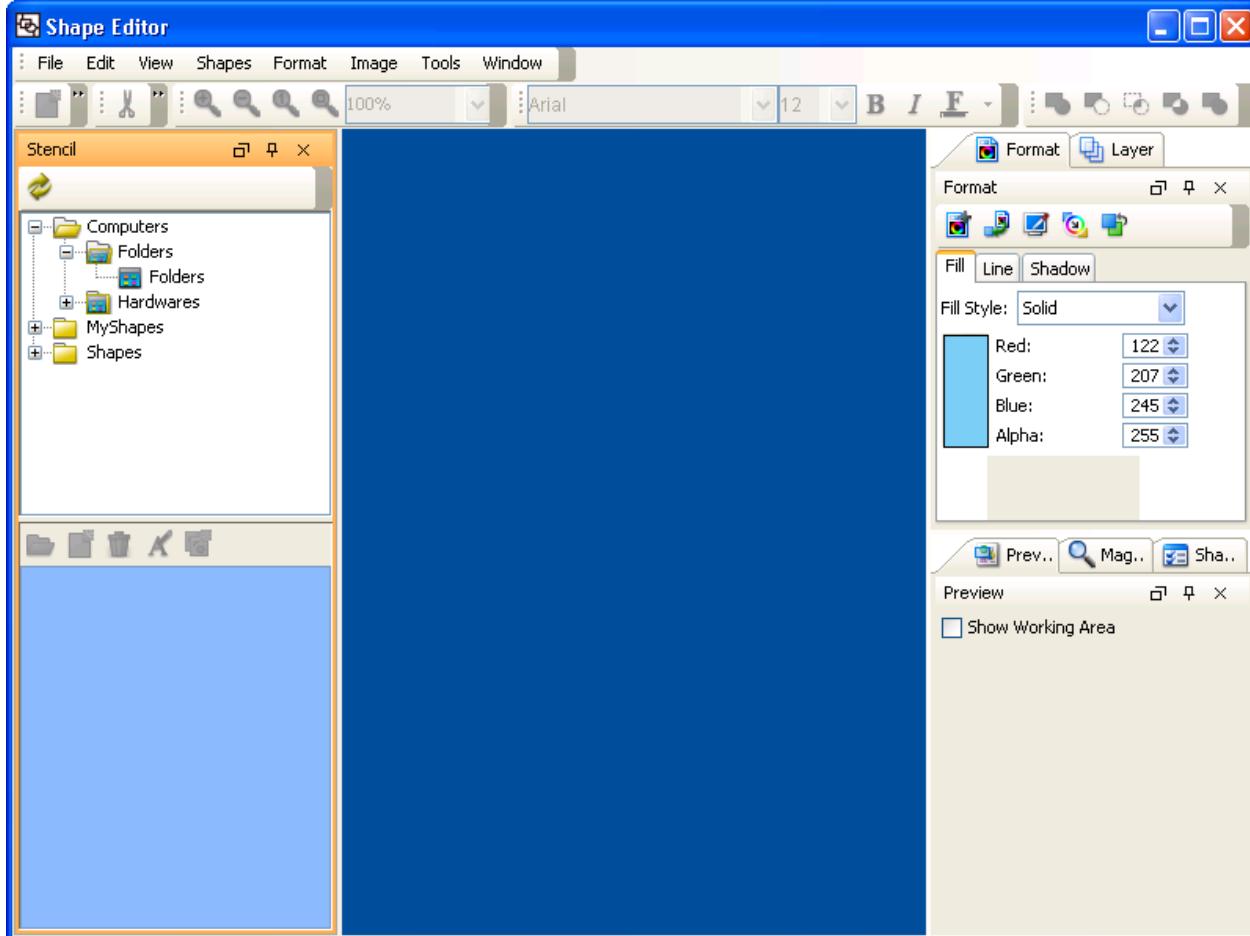
Although UML and BPMN are well-established notations, sometimes they still not rich enough to express domain specific idea. Shape Editor is a diagramming tool for you to design your own notation (stencil). Notations created can be incorporated into diagrams in VP-UML. To use shape Editor:

1. To launch Shape Editor, select **Tools > Shape Editor...** from the main menu.



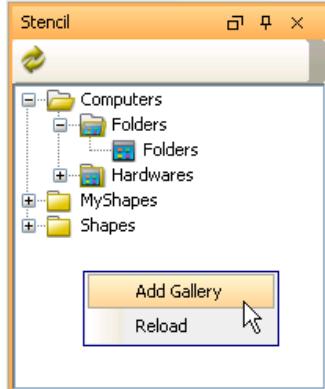
Opening Shape Editor through the main menu

This starts Shape Editor.



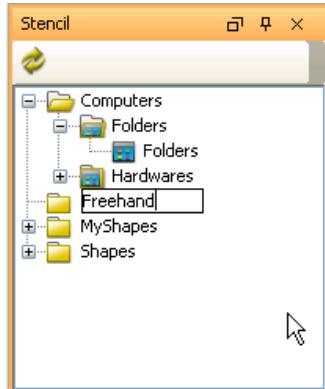
Shape Editor

2. 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 popup menu.



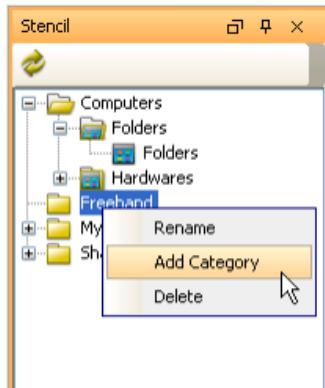
To add a gallery

3. Enter the gallery name and press the **Enter** key to confirm editing.



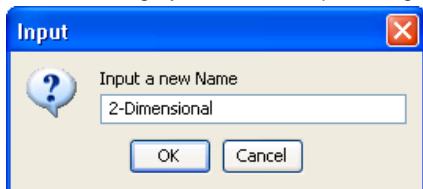
Naming a gallery

4. To create a category, right click on a gallery and select **Add Category** in the popup menu.



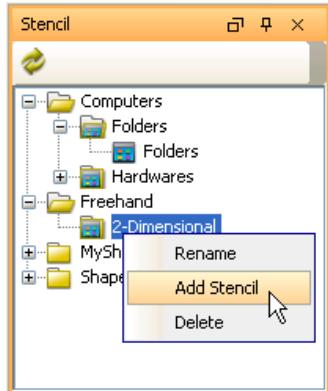
To add a category

5. Enter the category name in the Input dialog box and click OK to confirm.



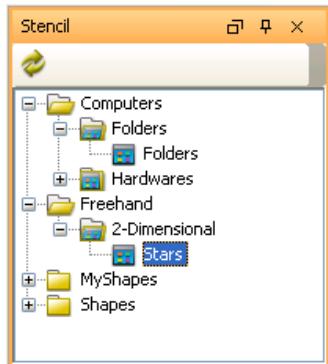
Naming a category

6. To create a stencil, right-click on a category and select **Add Stencil** in the popup menu.



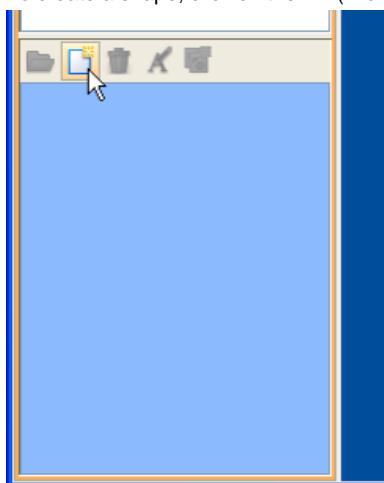
To add a stencil

7. Enter the stencil name and press the **Enter** key to confirm editing.



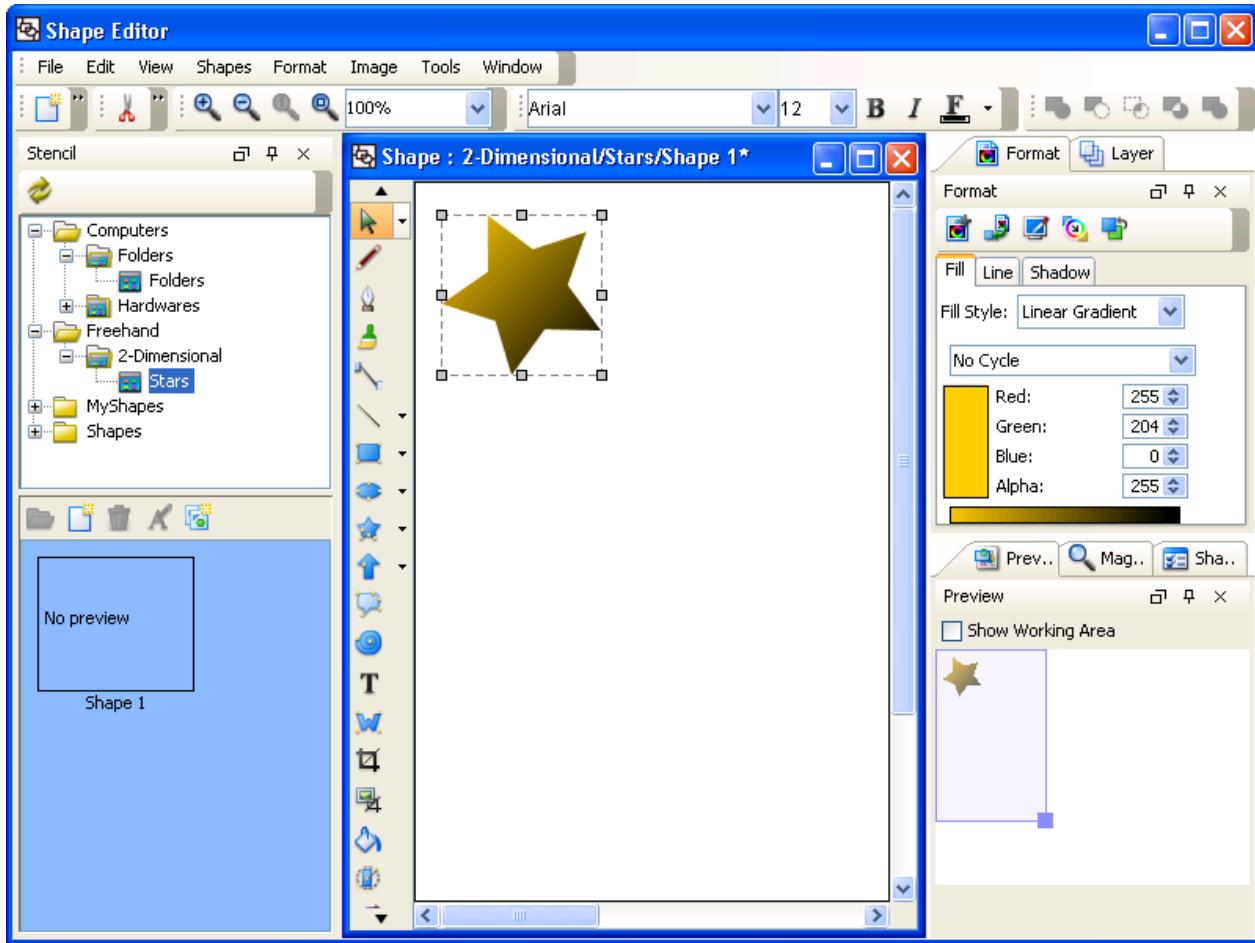
Naming a stencil

8. 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.



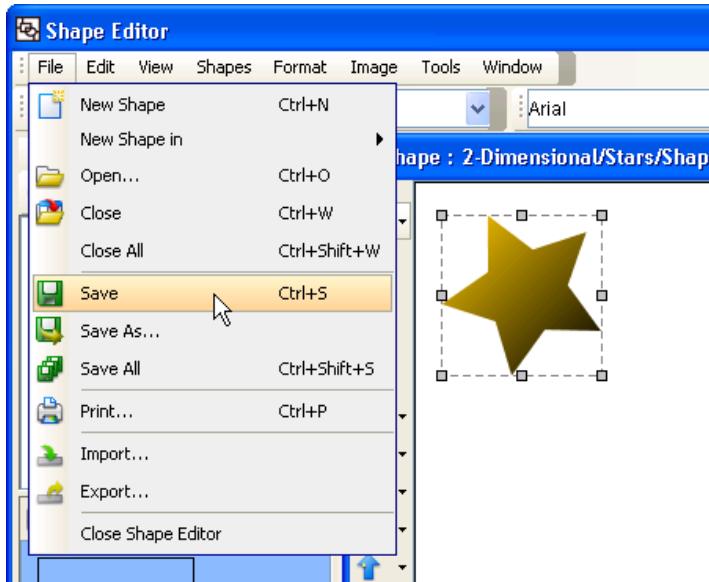
To add a shape

9. Draw the shape in the drawing pane.



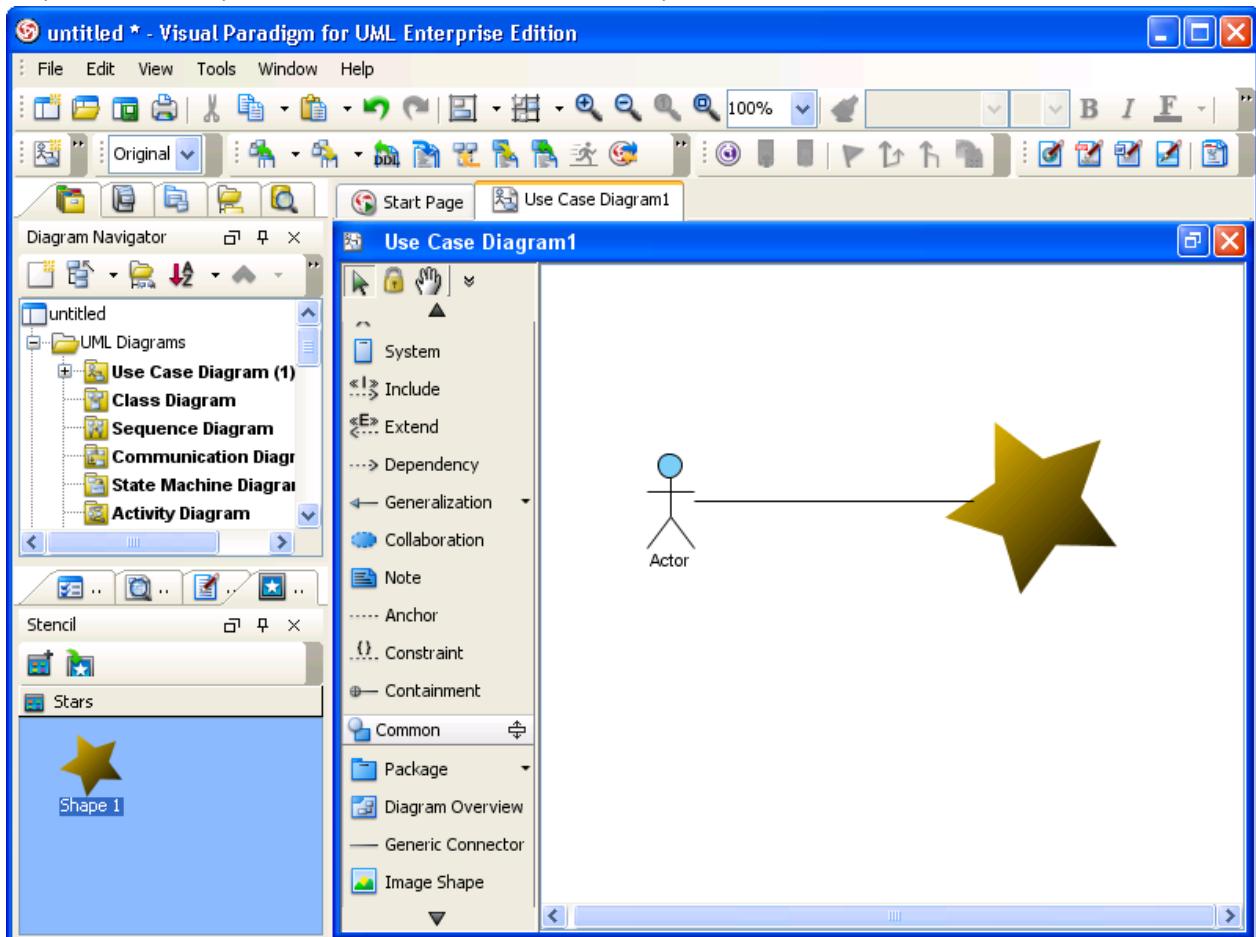
Drawing a shape in drawing pane

10. Save the drawing by selecting **File > Save** in the main menu.



Saving a drawing

Shapes created in Shape Editor can be used in VP-UML. For details, please refer to the next section.

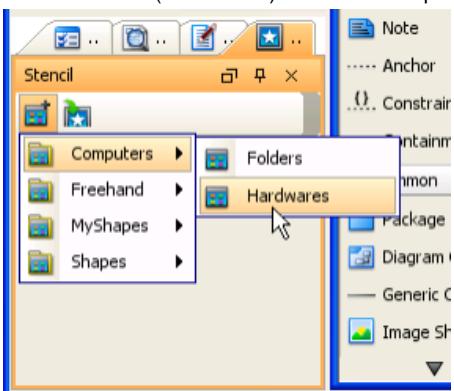


Apply a stencil shape in VP-UML

Creating shapes from stencil pane

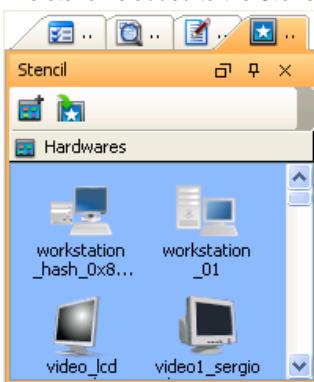
The Stencil pane is where user-created stencil shapes are stored. User can create a stencil shape on diagram by first displaying a stencil, dragging and dropping a shape from **Stencil** pane to diagram. Below are the steps in detail.

1. Open the **Stencil** pane. It is by default resided at the panes at the bottom left of VP-UML. If it does not appear, select **View > Panes > Stencil** from the main menu.
2. Click on the  (Add Stencil) button in the top of **Stencil** pane. This popup a list of gallery. Select the stencil to add.



To show a stencil

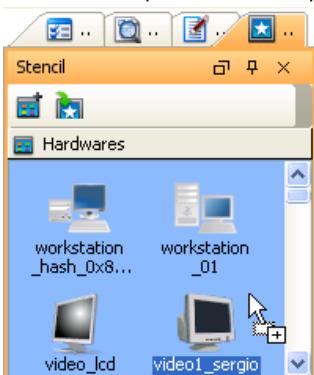
The stencil is added to the **Stencil** pane.



Stencil with shapes appearing in Stencil pane

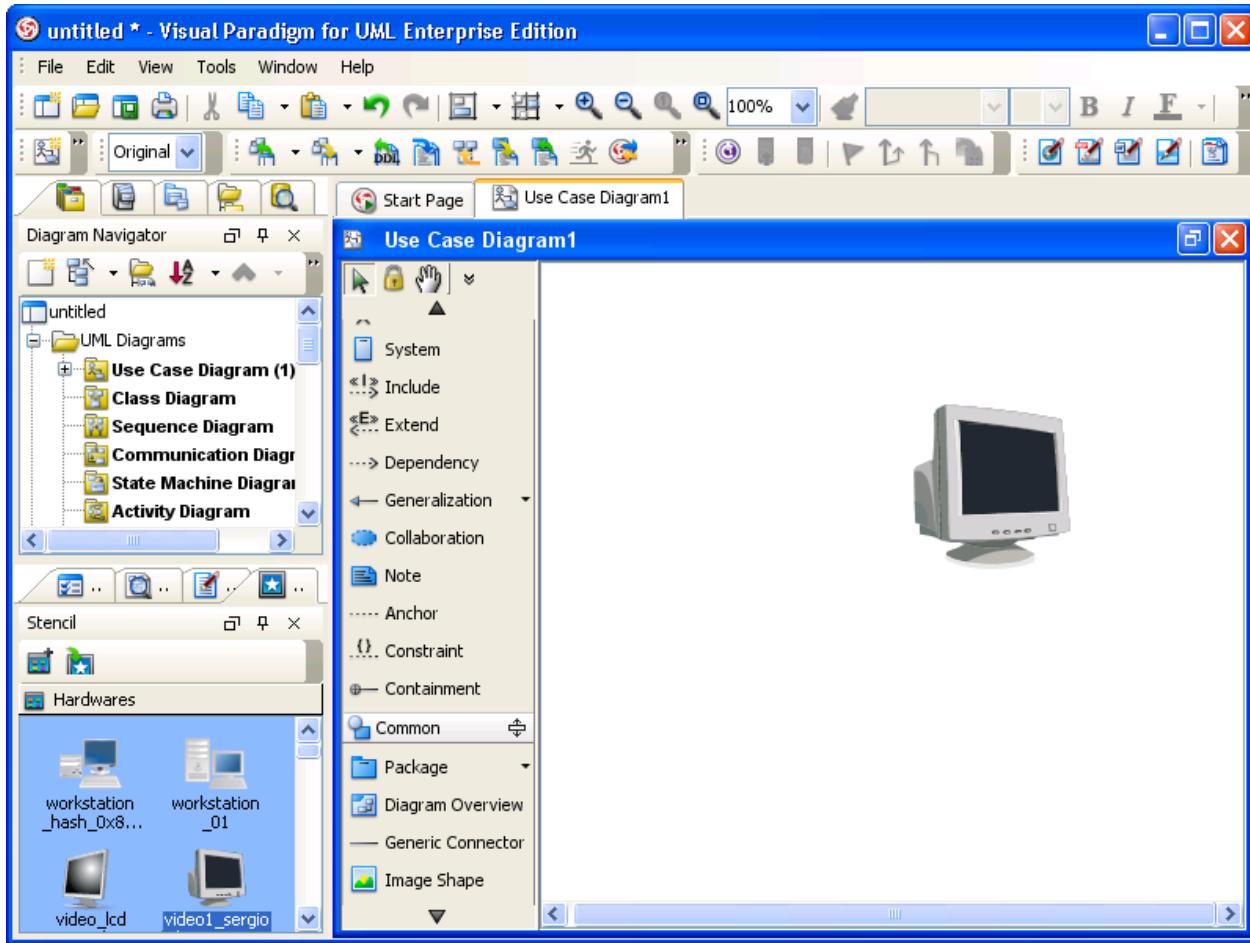
NOTE: You can add multiple stencil by repeating this step.

3. Press on a shape in the **Stencil** pane and drag it out of the **Stencil** pane.



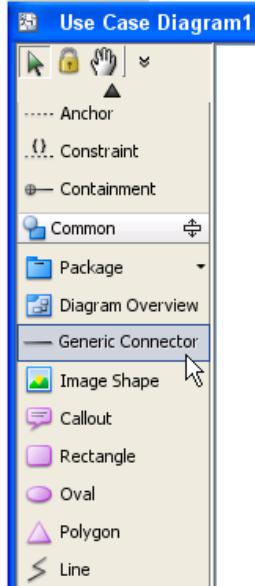
Dragging shape out of Stencil pane

4. Drop it onto the diagram to create the shape.



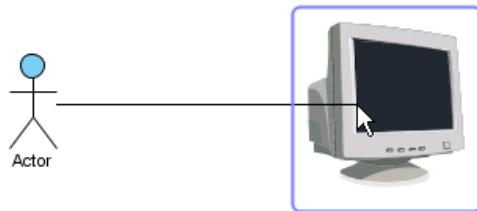
Stencil shape is created on diagram

5. 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, under the **Common** category.



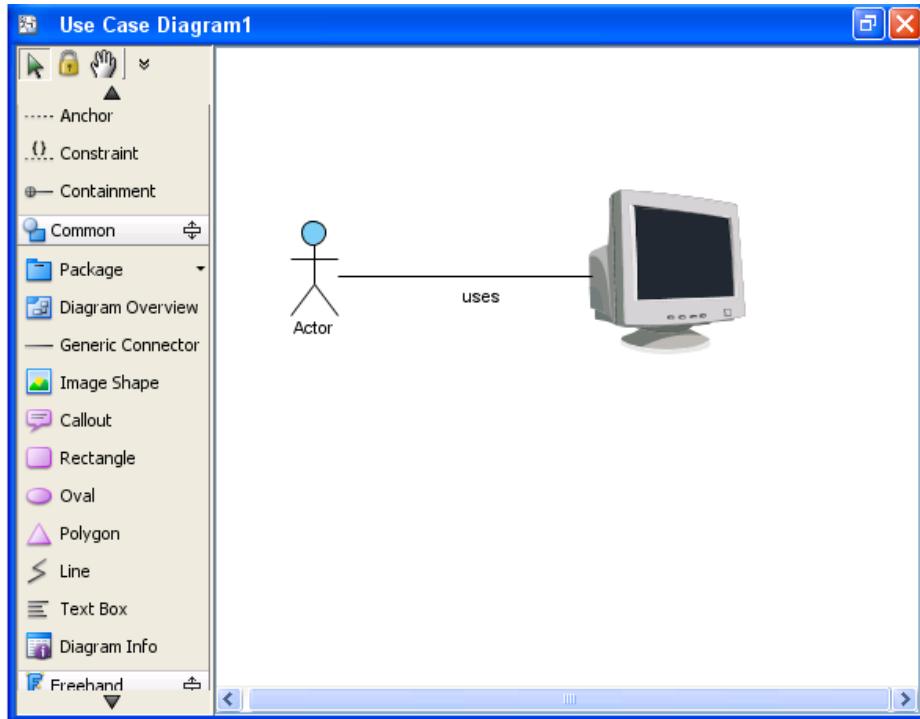
Selecting Generic Connector from the diagram toolbar

6. Press on the source shape, hold the mouse button, move the mouse cursor to the target shape and release the mouse button.



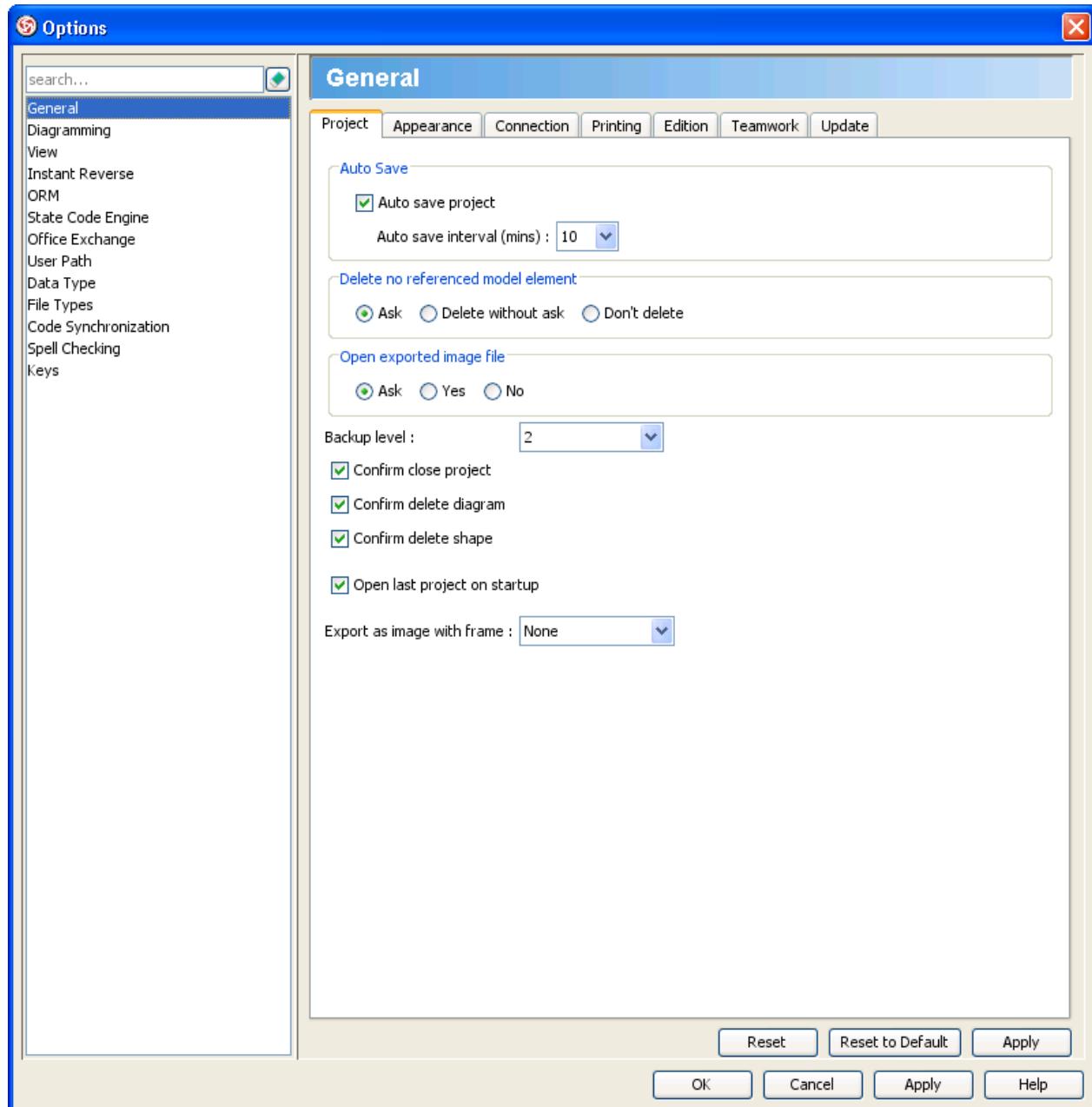
Connecting an Actor with a stencil shape

Connector is created.



Connector is created

General options



General Options in Option List

Project

General

Project Appearance Connection Printing Edition Teamwork Update

Auto Save

Auto save project

Auto save interval (mins) : 10

Delete no referenced model element

Ask Delete without ask Don't delete

Open exported image file

Ask Yes No

Backup level :

2

Confirm close project

Confirm delete diagram

Confirm delete shape

Open last project on startup

Export as image with frame :

None

Project Options in General

Option Name	Description
Auto save project	Save the opening project as backup periodically
Auto save interval (mins)	Time needed to wait until the next auto saving
Delete no referenced model element	Action that happen when all views of a model element are being deleted: <ul style="list-style-type: none">• Ask - (default) Prompt if you want to delete the model element as well• Delete without ask - 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 <ul style="list-style-type: none">• 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 delete diagram	(default true) Prompt if you really want to close diagram
Confirm delete shape	(default true) Prompt if you really want to delete a shape on diagram
Open last project on startup	(default true) If checked, it will open the last opened project immediately when starting VP-UML. If unchecked, it will open new project.
Export as image with frame	<ul style="list-style-type: none">• 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

Project Options details

Appearance

General

Project Appearance Connection Printing Edition Teamwork Update

Look and Feel

Look and feel : Office 2003 LookAndFeel (Windows)
Theme (for Office 2003 L&F only) : Default

User Language

Add or remove user language.
The checked user language will be used.

Installed User Language

Use	Language Name	Add...
<input checked="" type="checkbox"/>	English	<input type="button" value="^"/>
<input type="checkbox"/>	Simplified Chinese	<input type="button" value="▼"/>
<input type="checkbox"/>	Traditional Chinese	<input type="button" value="≡"/>
<input type="checkbox"/>	French	<input type="button" value=" "/>
<input type="checkbox"/>	German	<input type="button" value="v"/>

Date Time Format

Date Format : MMM d, yyyy
Date Sample : May 6, 2009
Time Format : h:mm:ss a
Time Sample : 1:46:57 PM

Measurement Unit

inch cm

Appearance Options in General

Option Name	Description
Look and feel	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.
Date Format	Format of date values that appear in the application
Time Format	Format of time values that appear in the application
Measurement Unit	<ul style="list-style-type: none">inch - (default) Set the measurement unit to be inchcm - Set the measurement unit to be centimeters

Appearance Options details

Connection

General

Project Appearance Connection Printing Edition Teamwork Update

E-mail :

Proxy Setting

Use proxy

Host : Port :
Login name : Password :

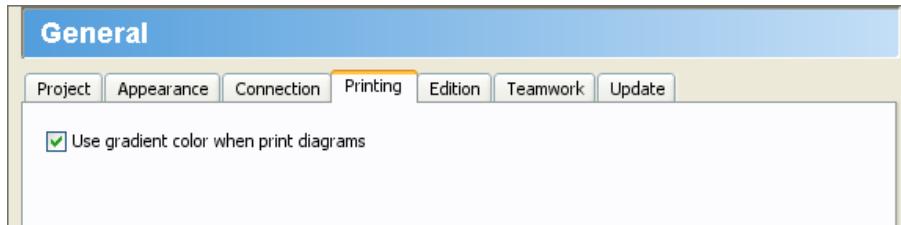
Connection Options in General

Option	Description
Name	

E-mail	Enter the Email field to specify your email address
Use proxy	(default false) To enable/disable the need of using 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).

Connection Options details

Printing

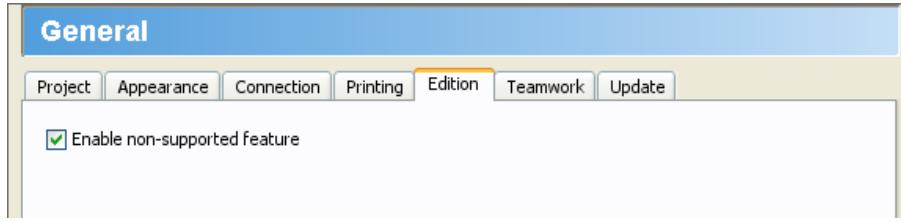


Connection Options in General

Option Name	Description
Use gradient color when print diagrams	Control whether to use gradient or solid color for shapes and diagrams when printing.

Printing Options details

Edition

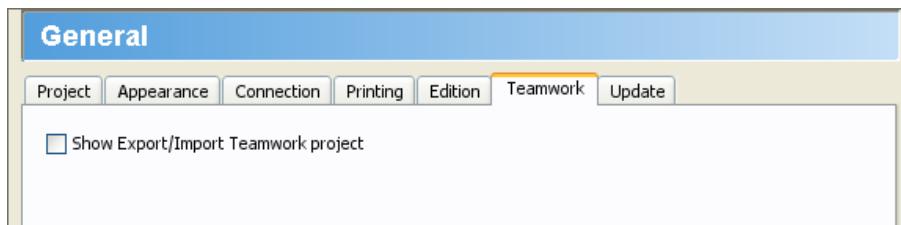


Edition Options in General

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.

Printing Options details

Teamwork



Teamwork Options in General

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

Update

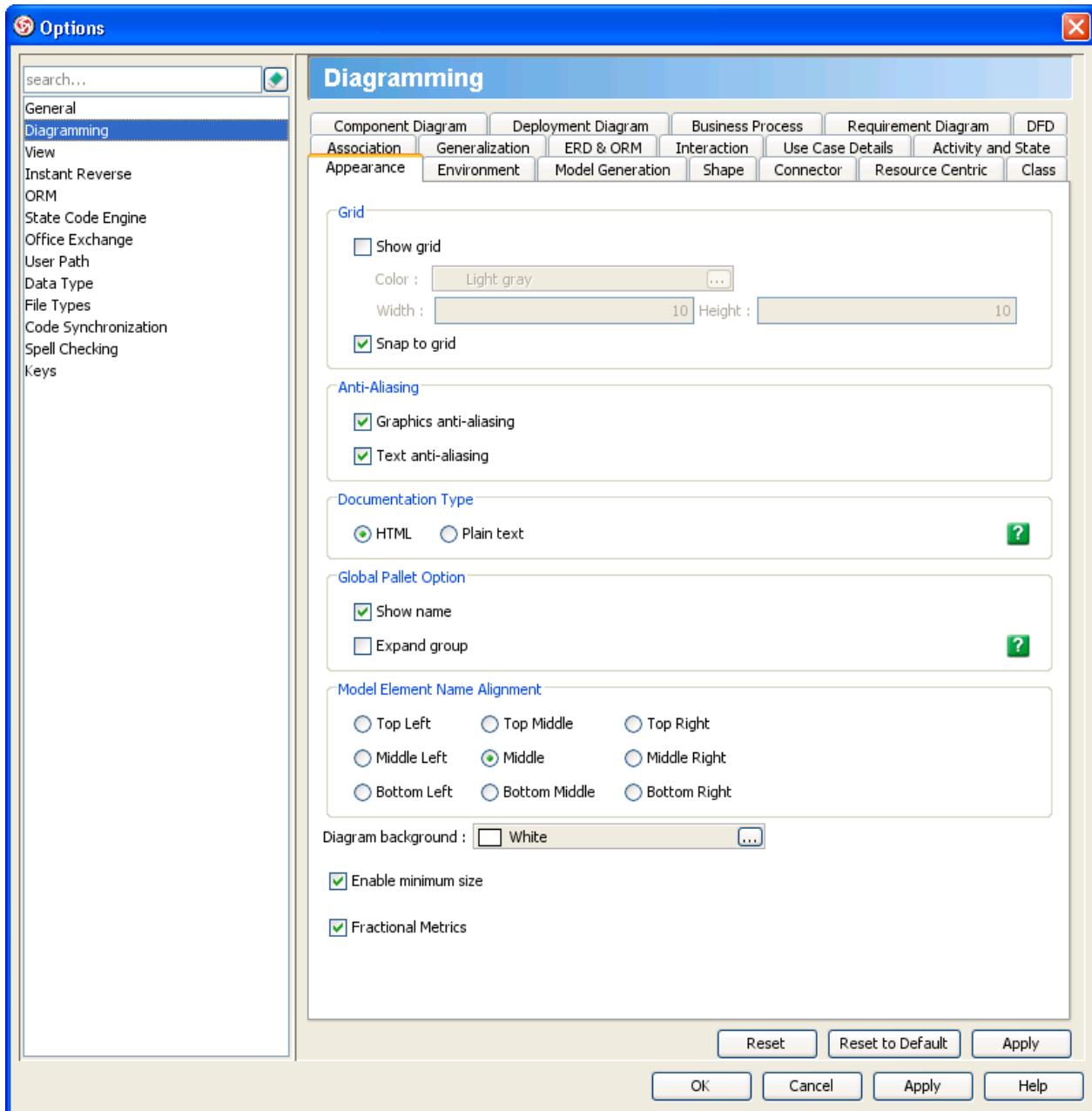


Update Options in General

Option Name	Description
Auto update	<p>Auto Update lets VP-UML to check for available updates with respect to the running build/version, and notify you to perform update whenever possible.</p> <ul style="list-style-type: none">• Never - Do not inform product update• On every start - Check for updates everytime when starting VP-UML• Daily - Check for updates daily• Weekly - (default) Check for updates weekly• Monthly - Check for updates monthly

Update Options details

Diagramming options



Diagramming Options in Options List

Appearance

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector

Grid

Show grid

Color :

Width : Height :

Snap to grid

Anti-Aliasing

Graphics anti-aliasing

Text anti-aliasing

Documentation Type

HTML Plain text



Global Pallet Option

Show name

Expand group



Model Element Name Alignment

- Top Left Top Middle Top Right
- Middle Left Middle Middle Right
- Bottom Left Bottom Middle Bottom Right

Diagram background :

Enable minimum size

Fractional Metrics

Appearance Options in Diagramming

Option Name	Description
Show grid	(default false) Show grid lines on diagram
Color	Color of grid lines
Width	Determines the horizontal spaces between grid lines
Height	Determines the vertical spaces between grid lines
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
Graphics anti-aliasing	(default true) Smoothen the graphics
Text anti-aliasing	(default true) Smoothen the text
Documentation Type	Default type of documentation <ul style="list-style-type: none"> • HTML - (default) HTML text that consists of formatting such as bold, italic, underline, table • Plain text - Text without formatting
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
Model Element Name Alignment	<ul style="list-style-type: none"> • 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

Apearance Options details

Environment

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector

Textual Analysis Highlight Option

Case insensitive Case sensitive

Alignment Guide

Show diagram alignment guide

Show edges Show center



Shape Selection Detection

Inside selection area Overlapped with selection area

Copy as XML with RTF style

Yes No Prompt

Delay of show Quick Preview in Diagram Tree (second) :

Show Copy to Clipboard as OLE

Default Copy Action :

Within VP-UML EE



Unspecified

Show shape content when dragging

Default HTML Documentation Font

Font : Size : Bold Italic

Color :

Stereotype support HTML tagged value

Prompt when Apply Design Pattern

Show Use Case Extension Points

Environment Options in Diagramming

Option Name	Description
Textual Analysis Highlight Option	<ul style="list-style-type: none"> 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.
Alignment Guide - Show diagram alignment guide	(default true) Show alignment guide which appear when moving a shape on a diagram
Alignment Guide	<ul style="list-style-type: none"> Show edges - (default) Show guides at edges of the closest shape Show center - Show a guide that lies on the center of the closest shape
Shape Selection Detection	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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 of show Quick Preview in Diagram Tree (second)	<ul style="list-style-type: none"> 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
Show Copy to Clipboard as OLE	(default false) Determines whether the Copy to Clipboard as OLE menu is available or not
Default Copy Action	<ul style="list-style-type: none"> Within VP-UML - (default) When triggering the hotkey for Copy (Ctrl-C, by default), it will perform copying within VP-UML 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
Copy as image with frame	<ul style="list-style-type: none"> 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
Default HTML Documentation Font	The default font face, size, color, bold and italic status for HTML content in documentation pane
Stereotype support HTML tagged value	Enables you to define tagged value in HTML format for stereotype
Prompt when Apply 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
Show Use Case Extension Points	(default true) Show Use Case Extension Points within Use Case shapes

Environment Options details

Model generation

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector

Generate Sequence Diagram from Use Case Description Overwrite Existing Diagram

Yes Prompt

Generate Diagram from Scenario Overwrite Existing Diagram

Yes Prompt

Overwrite Flow of Events when Synchronize from Sequence Diagram

Ask Yes

Default generate diagram type from scenario :

Model Generation Options in Diagramming

Option Name	Description
Generate Sequence Diagram from Use Case Description Overwrite Existing Diagram	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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
Default generate diagram type from scenario	<ul style="list-style-type: none"> • 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

Model Generation Options details

Shape

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector

Font

Font : Dialog Size : 11 Bold Italic
Color : Black

Initial Shape Size When Set to Display Image

Fit shape to image Fit image to shape

Add Covered Shapes as Children After Resize

Yes No Prompt

Auto Expand Parent When Moving Child

Enable auto expand parent

Expand parent delay (second): 0.2

Shape line format : Black

Shape fill format : (122, 207, 245)

Auto fit size (diagram-based)

Create new line key : Confirm becomes <Enter>

Show unlimited level in preview shape (May slower the diagram display)

Show rich text content in preview shape

Shape Options in Diagramming

Option Name	Description
Font	Default font settings for shape content
Initial Shape Size When Set to Display Image	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Yes - Automatic 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
Auto Expand Parent When Moving Child - Enable auto expand parent	(default true) Expand parent's width or height when a child shape is being moved around the edge of parent
Auto Expand Parent When Moving Child -Expand parent delay (second)	The time needed to response to child's movement around parent's edge
Shape line format	The default line format for shapes
Shape fill format	The default fill format for shapes
Auto fit size (diagram-based)	(default false) Determines whether shapes in diagrams will fit in size automatically
Create new line key	<ul style="list-style-type: none"> <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

Shape Generation Options details

Connector

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector

Font

Font : Dialog Size : 11 Bold Italic
Color :

Connector Style

Rectilinear Round Rectilinear Oblique Round Oblique Curve

Connection Point Style

Round the shape Follow center

Line Jumps

Off Arc Gap Square

Caption orientation :

Foreground :

Background :

Default pin from connection point

Default pin to connection point

Show relationship connectors for dropped models

Auto relocate connector when overlapped with other shapes

Scroll connector delay (second) :

Connector Options in Diagramming

Option Name	Description
Font	Default font settings for connector caption
Connector Style	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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
Caption orientation	<ul style="list-style-type: none"> 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 at the same horizontal level as the connector Follow Connector Angle and Keep Text Upright - Enforce connector caption to appear at the same horizontal level as the connector, but keep the text upright
Foreground	Foreground color of connector
Background	Background color of connector
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)	<ul style="list-style-type: none"> No delay - Immediately scroll to the other side of connector 1 - 9 - Time provided for scrolling to the other side of connector

Connector Generation Options details

Resource centric

Diagramming

Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD
Association	Generalization	ERD & ORM	Interaction	Use Case Details
Appearance	Environment	Model Generation	Shape	Connector
				Resource Centric
				Class

Show resources
 Show group resources
 Show extra resources
 Show generic resources only
 Show resources delay (second) :
 Auto hide resources delay (second) :
 Always show reference and sub-diagram resource

Resource Centric Options in Diagramming

Option Name	Description
Show resources	(default true) Show resource icons around shapes
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
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
Always show reference and sub-diagram resource	(default true) Always show the reference and subdiagram resource icon at the bottom of shape no matter whether the shape has reference and sub-diagram added.

Resource Centric Generation Options details

Class

Diagramming

Component Diagram Deployment Diagram Business Process Requirement Diagram DFD

Association Generalization ERD & ORM Interaction Use Case Details Activity and State

Appearance Environment Model Generation Shape Connector Resource Centric Class

Behavior Presentation Auto Attribute Type

Name completion
 Auto-synchronize role name
 Auto-generate role name
 Support multiple-line attribute
 Support multiple-line class name

Update constructor after class renamed :

Tooltips of Class and Package

Show parent style :
 Show fully qualified class and package name in tooltip

Show row grid line within compartment of Classes in Class Diagram (diagram type-based)

Default parameter direction :

Default Visibility

Class :
Attribute : Operation :

Class Options in Diagramming

Option Name	Description
Name completion	(defaultGeneralizaion true) When creating a class, you can select an existing class from a list so that a view of the selected class can be created. You can enable or disable the popup of such list.
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 between created
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	<ul style="list-style-type: none"> • 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)
Show parent style	<ul style="list-style-type: none"> • UML - (default) Show tooltip of child class in UML style like <i>Package::Class</i> • Java/C# - Show tooltip of child class in Java/C# style like <i>Package.Class</i>
Show fully qualified class and package name in tooltip	(default true) Enable to show fully qualified class and package name in tooltip like <i>Package::Class/Package.Class</i> (depending on the setting of Show parent style). Disable to show only the hovering class or package name and type like <i>Class : Class</i> .
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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

Behavior	Presentation	Auto Attribute Type
Show attribute option :	Show all	Show type option : Name only
Show operation option :	Show all	
Visibility style :	UML	
<input checked="" type="checkbox"/> Show attribute initial value <input type="checkbox"/> Show attribute multiplicity <input type="checkbox"/> Show attribute getter/setter <input checked="" type="checkbox"/> Show operation signature <input checked="" type="checkbox"/> Show class member stereotype <input type="checkbox"/> Wrap class member		
<input type="checkbox"/> Show owner of class/package <input checked="" type="checkbox"/> Show template parameter <input checked="" type="checkbox"/> Display as Robustness Analysis icon <input type="checkbox"/> Display as stereotype icon <input checked="" type="checkbox"/> Show operation parameter name <input type="checkbox"/> Show empty compartments		

Presentation of Class Options in Diagramming

Option Name	Description
Show attribute option	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

Behavior Presentation Auto Attribute Type

Default attribute type :

Auto set attribute type by name

Name	Type	Default Value

Add Edit Delete

Auto Attribute Type of Class Options in Diagramming

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

Association

Diagramming

Appearance Environment Model Generation Shape Connector Resource Centric Class
Component Diagram Deployment Diagram Business Process Requirement Diagram DFD
Association Generalization ERD & ORM Interaction Use Case Details Activity and State

Presentation Options

- Show association name
- Show association stereotype
- Show from role name
- Show to role name
- Show from role visibility
- Show to role visibility
- Show from multiplicity
- Show to multiplicity
- Show multiplicity constraints
- Show direction
- Show association role stereot...

Preview

```

graph LR
    Class[Class] ---> Class2[Class2]
    Class -- "1..*" --> Class2
    Class -- "+_from" --> Class2
    Class -- "+_to" --> Class2
    Class -- "association name" --> Class2
  
```

Default Association End Navigable :

Default Association End Visibility :

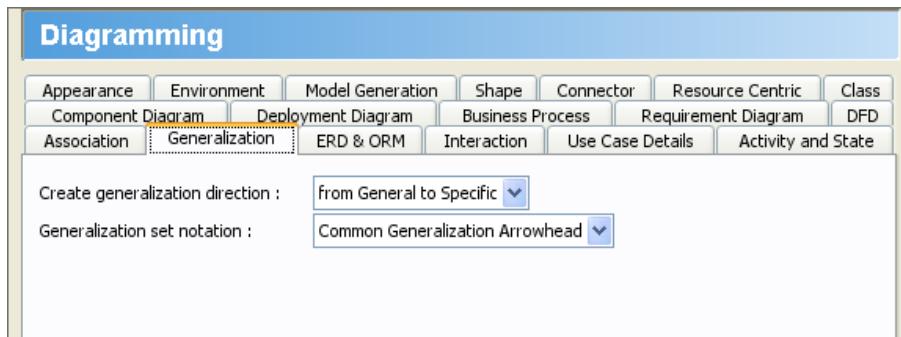
Suppress implied "1" multiplicity for attribute and association end

Association Options in Diagramming

Option Name	Description
Show association name	(default true) Show the name of association
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
Default Association End Navigable	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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

Association Options details

Generalization



Generalization Options in Diagramming

Option Name	Description
Create generalization direction	<ul style="list-style-type: none"> from General to Specific - (default) When creating a generalization, the arrow head will appear at the mouse release side from Specific to General - When creating a generalization, the arrow head will appear at the firstly selected shape
Generalization set notation	<ul style="list-style-type: none"> 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

Generalization Options details

ERD & ORM

Diagramming

Appearance Environment Model Generation Shape Connector Resource Centric Class
Component Diagram Deployment Diagram Business Process Requirement Diagram DFD
Association Generalization ERD & ORM Interaction Use Case Details Activity and State

Behavior & Presentation Auto Column Type

Show column type ?
 Show Foreign key name
 Foreign key connector end points to associated column
 Align column properties
 Show extra column properties
 Show row grid line within compartment of Entities/Views in ERD (diagram type-based)
 Show row grid line within compartment of Entities/Views/Classes in ORM Diagram (diagram ...)
 Warning on create ORM-Persistable Class in default package
 Show schema name in ERD: \${name}.\${tableName}

Primary Key Pattern

Format : Add Variables...

Example :

Primary Key Constraint Pattern

Format : Add Variables...

Example :

Foreign Key Pattern

Format : Add Variables...

Example :

Foreign Key Relationship Pattern

Format : Add Variables...

Example :

ERD & ORM Options in Diagramming

Option Name	Description
Show column type	(default true) Show data type of column
Show foreign key name	(default false) Show
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
Primary Key Pattern Format	Pattern of primary keys that will be applied when synchronizing Class Diagram to Entity Relationship Diagram, which may create primary key
Primary Key Constraint Pattern Format	Pattern of primary key constraint that will be applied when creating Entity
Foreign Key Pattern Format	Pattern of foreign key that will be applied when creating a primary key on an entity which result in creating foreign key in connected entities
Foreign Key Relationship Pattern Format	Pattern of foreign key relationship that will be applied when creating a primary key on an entity which result in creating foreign key in connected entities

ERD & ORM Options details

Diagramming

Appearance	Environment	Model Generation	Shape	Connector	Resource Centric	Class
Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD		
Association	Generalization	ERD & ORM	Interaction	Use Case Details	Activity and State	

Behavior & Presentation Auto Column Type

Default column type : `integer(10)`

Auto set column type by name

Name	Type	Default Value	Add
			<input type="button" value="Add"/>

Auto Column Type of ERD & ORM Options in Diagramming

Option Name	Description
Default column type	Define column type that will be applied to newly created columns
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

Interaction

Diagramming

Appearance	Environment	Model Generation	Shape	Connector	Resource Centric	Class
Component Diagram	Deployment Diagram	Business Process	Requirement Diagram	DFD		
Association	Generalization	ERD & ORM	Interaction	Use Case Details	Activity and State	

Auto fit message completion size

Show message completion documentation

Mark target lifeline stopped when attached by destroy message

Yes No Prompt

Presentation Options

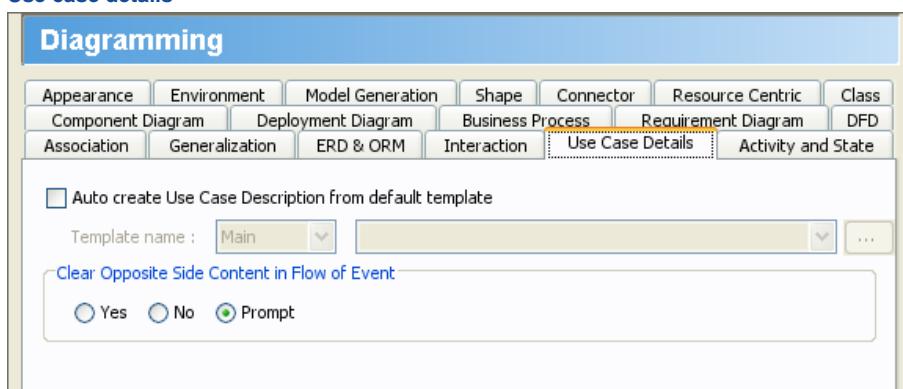
- Show sequence number in Communication Diagram
- Show sequence number in Sequence Diagram
- Show messages operation signature in Sequence Diagram and Communication Diagram (diagram...)
- Show stereotype of message in Sequence Diagram and Communication Diagram (diagram-based)
- Show activations in Sequence Diagram (diagram-based)
- Display as Robustness Analysis icon in Sequence Diagram
- Show associated diagram name of Interaction

Interaction Options in Diagramming

Option Name	Description
Auto fit message completion size	(default true) Fit message completion box's size everytime you activate it. If disabled, size adjusted manually won't be remembered
Show message completion documentation	(default true) When selecting an Operation in the message completion box, the documentation of operation will appear next to the completion box
Mark target lifeline stopped when attached by destroy message	The effect when attaching a Destroy Message to a Lifeline, whether the Lifeline will be marked stopped or not <ul style="list-style-type: none"> • 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

Use case details

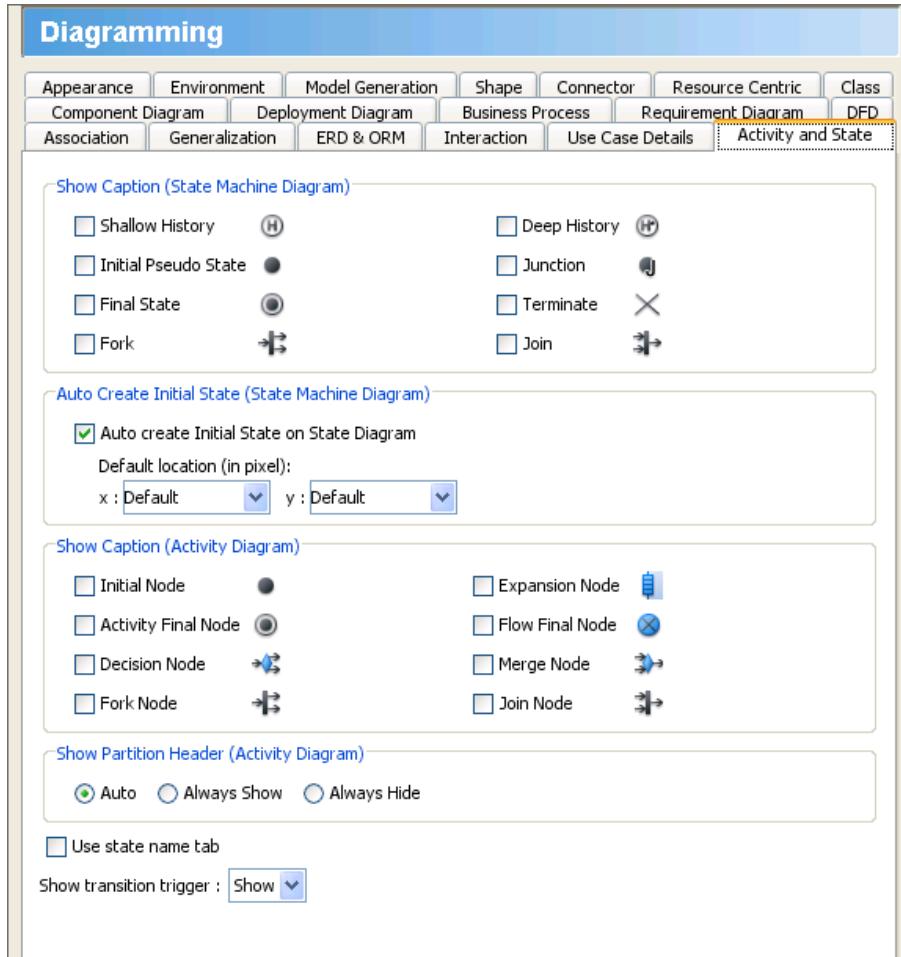


Use Case Details Options in Diagramming

Option Name	Description
Auto create Use Case Description from default template	(default false) When opening Use Case Description for a Use Case, a Use Case Description will be created automatically using the chosen template. You can select a template from the drop down menu under the checkbox, or select Other in the drop down menu and import the template from file.
Template name	<ul style="list-style-type: none"> Main - (default) Main template will be used as Use Case Description default template Alternative - Alternative template will be used as Use Case Description default template Basic - Basic template will be used as Use Case Description default template Full - Full template will be used as Use Case Description default template Scenario - Scenario template will be used as Use Case Description default template Other - Select Other to enable the drop down menu on the right, which is made for selecting a template file to be used for default template
Clear Opposite Side Content in Flow of Event	For every row of a flow of events table, either the Actor or the System side should be filled, but not both. This option is for controlling the effect when you try to enter content for both Actor and System cell. <ul style="list-style-type: none"> Yes - Automatically clear the other side's cell content No - Keep the other side's content Prompt - (default) Ask for whether to clear or keep the other side's content

Use Case Details Options details

Activity and state

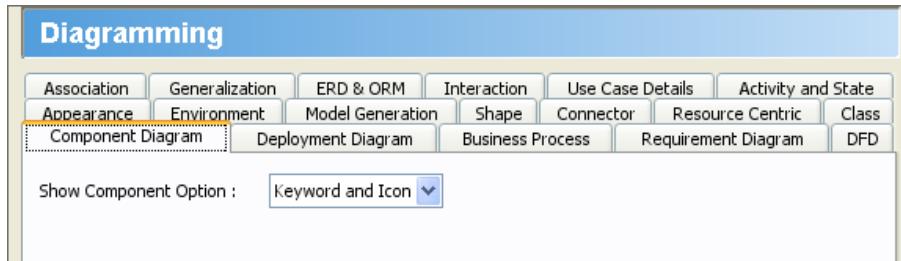


Activity and State Options in Diagramming

Option Name	Description
Show Caption (State Machine Diagram)	<ul style="list-style-type: none"> 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
Show Caption (Activity Diagram)	<ul style="list-style-type: none"> 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 NodeArtifact Fork Node - (default false) Show caption for Fork Node Join Node - (default false) Show caption for Join Node
Show Partition Header (Activity Diagram)	<ul style="list-style-type: none"> 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
Use state name tab	(default false) Name tab is 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. <ul style="list-style-type: none"> Show - (default) Show Triggers information on a Transition connector Hide - Do not show Triggers information on a Transition connector

Activity and State Options details

Component diagram



Component Diagram Options in Diagramming

Option Name	Description
Show Component Option	<ul style="list-style-type: none"> 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
<i>Component Diagram Options details</i>	

Deployment diagram

Diagramming

Association	Generalization	ERD & ORM	Interaction	Use Case Details	Activity and State
Appearance	Environment	Model Generation	Shape	Connector	Resource Centric
Component Diagram	Deployment Diagram	Business Process		Requirement Diagram	DFD

Show Artifact Option :

Messag

Flo

Component Diagram Options in Diagramming

Option Name	Description
Show Artifact Option	<ul style="list-style-type: none"> 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

Business process

Diagramming

Association	Generalization	ERD & ORM	Interaction	Use Case Details	Activity and State
Appearance	Environment	Model Generation	Shape	Connector	Resource Centric
Component Diagram	Deployment Diagram	Business Process		Requirement Diagram	DFD

Invalid Connection Handling

Ignore all Cancel move Prompt

Show Lane Handle

Auto Always Show Always Hide

Connection Point Style

Round the shape Follow center

Connector Style

Rectilinear Round Rectilinear Oblique Round Oblique Curve



Connect Gateway with Flow Object in Different Pool

Prompt Connect with Message Flow Connect with Sequence Flow

ID Generator Format

Prefix : Num of digits : Suffix :

Show ID option :

Task

Show Activities type icon (diagram-based)

Show convert Sub-Process/Task warning

Auto stretch pools

Business Process Options in Diagramming

Option Name	Description
Invalid Connection Handling	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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
Connection Point Style	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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
Connect Gateway with Flow Object in Different Pool	<ul style="list-style-type: none"> 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
ID Generator Format Prefix	Prefix of ID that will be automatically generated when creating BPMN shapes
ID Generator Format Num of digits	The number of digits of ID that will be automatically generated when creating BPMN shapes
ID Generator Format Suffix	Suffix of ID that will be automatically generated when creating BPMN shapes
ID Generator Format Show ID option	<ul style="list-style-type: none"> 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 Activities type icon (diagram-based)	(default true) Show icons that represent the type of Task and Sub-Process
Show convert Sub-Process/Task warning	(default true) Show warning when trying to convert between Sub-Process and Task
Auto stretch pools	(default true) Stretch Pools automatically to the reach diagram bound

Business Process Options details

Requirement diagram

The screenshot shows the 'Diagramming' tab selected in a toolbar with various sub-options like Association, Generalization, ERD & ORM, Interaction, Use Case Details, Activity and State, Appearance, Environment, Model Generation, Shape, Connector, Resource Centric, Class, Component Diagram, Deployment Diagram, Business Process, Requirement Diagram (which is highlighted), and DFD.

Below the toolbar, there's a section titled 'ID Generator Format' with fields for 'Prefix', 'Num of digits', and 'Suffix'. There are dropdown menus for each.

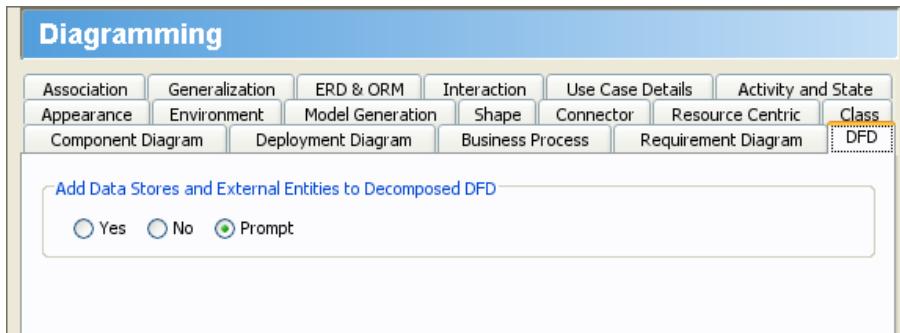
At the bottom, there are checkboxes for 'Show Attributes' (set to 'Show All Attributes'), 'Wrap member', and 'Support HTML Attribute'.

Requirement Diagram Options in Diagramming

Option Name	Description
ID Generator Format Prefix	Prefix of Requirement ID that will be automatically generated when creating Requirement
ID Generator Format Num of digits	The number of digits of Requirement ID that will be automatically generated when creating Requirement
ID Generator Format Suffix	Suffix of Requirement ID that will be automatically generated when creating Requirement
Show Attributes	<ul style="list-style-type: none"> • 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

DFD

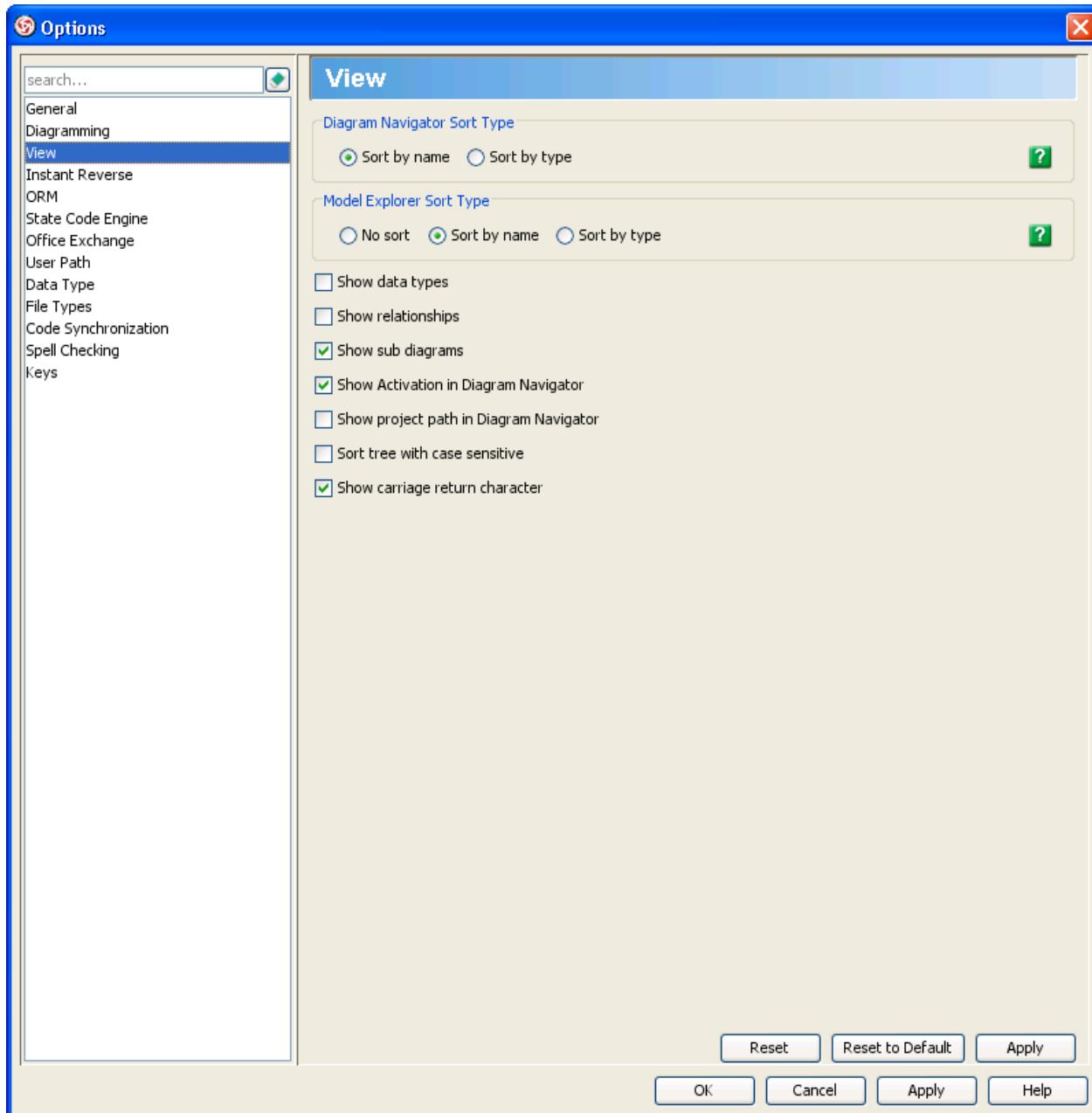


DFD Options in Diagramming

Option Name	Description
Add Data Stores and External Entities to Decomposed DFD	<ul style="list-style-type: none"> • 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

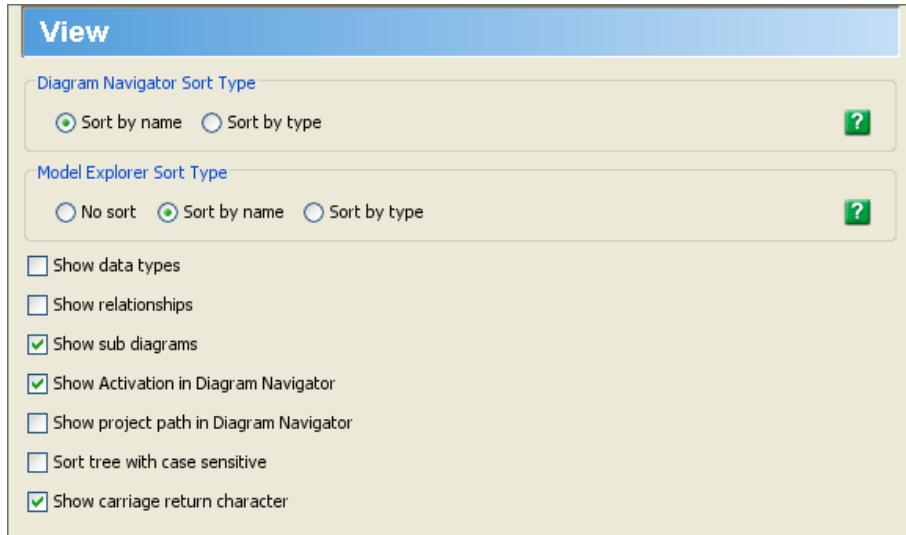
DFD Options details

View options



View Options in Option List

Options

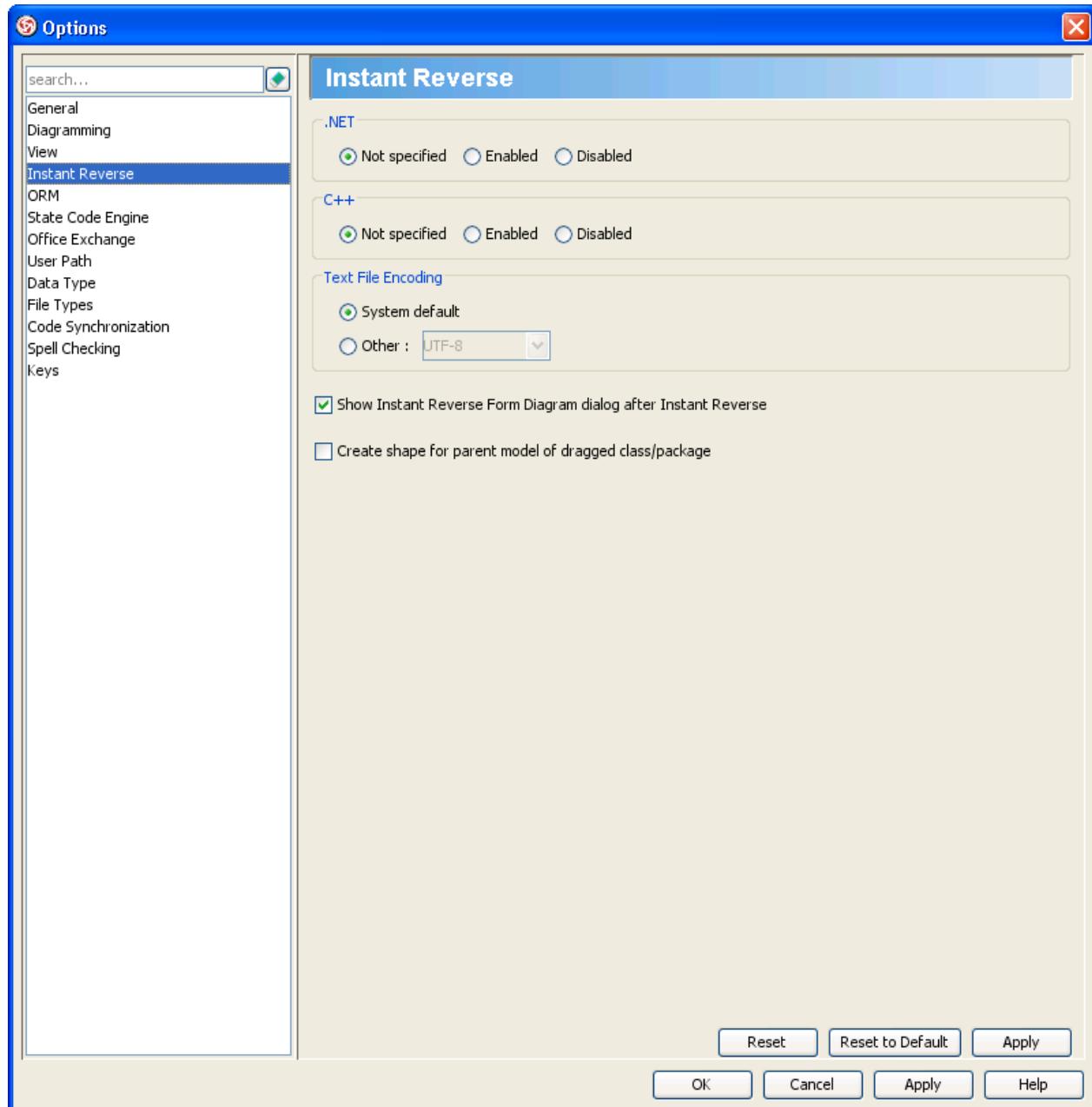


[View Options](#)

Option Name	Description
Diagram Navigator Sort Type	<ul style="list-style-type: none">Sort by name - (default) Sort tree nodes in Diagram Navigator by their namesSort by type - Sort tree nodes in Diagram Navigator by their types
Model Explorer Sort Type	<ul style="list-style-type: none">No sort - Do not sort tree nodes in Model ExplorerSort by name - (default) Sort tree nodes in Model Explorer by their namesSort 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 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

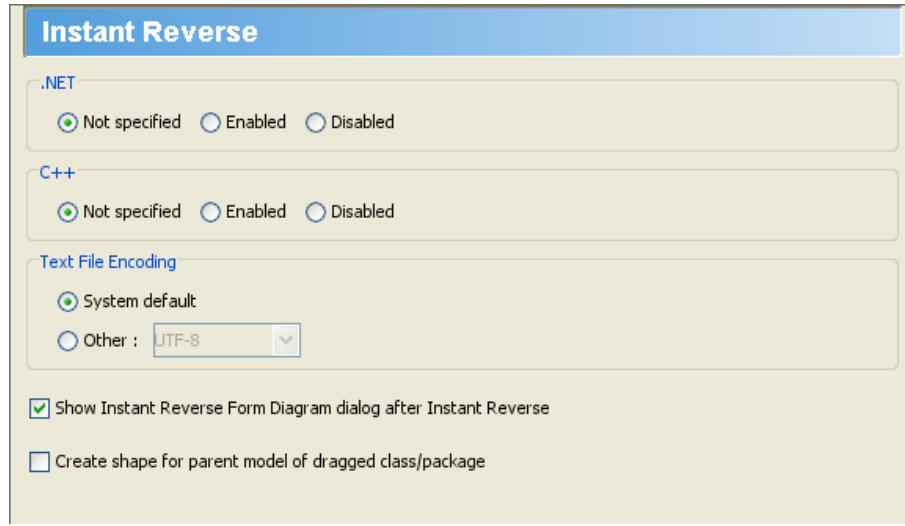
[View Options details](#)

Instant reverse options



Instant Reverse Options in Option List

Options

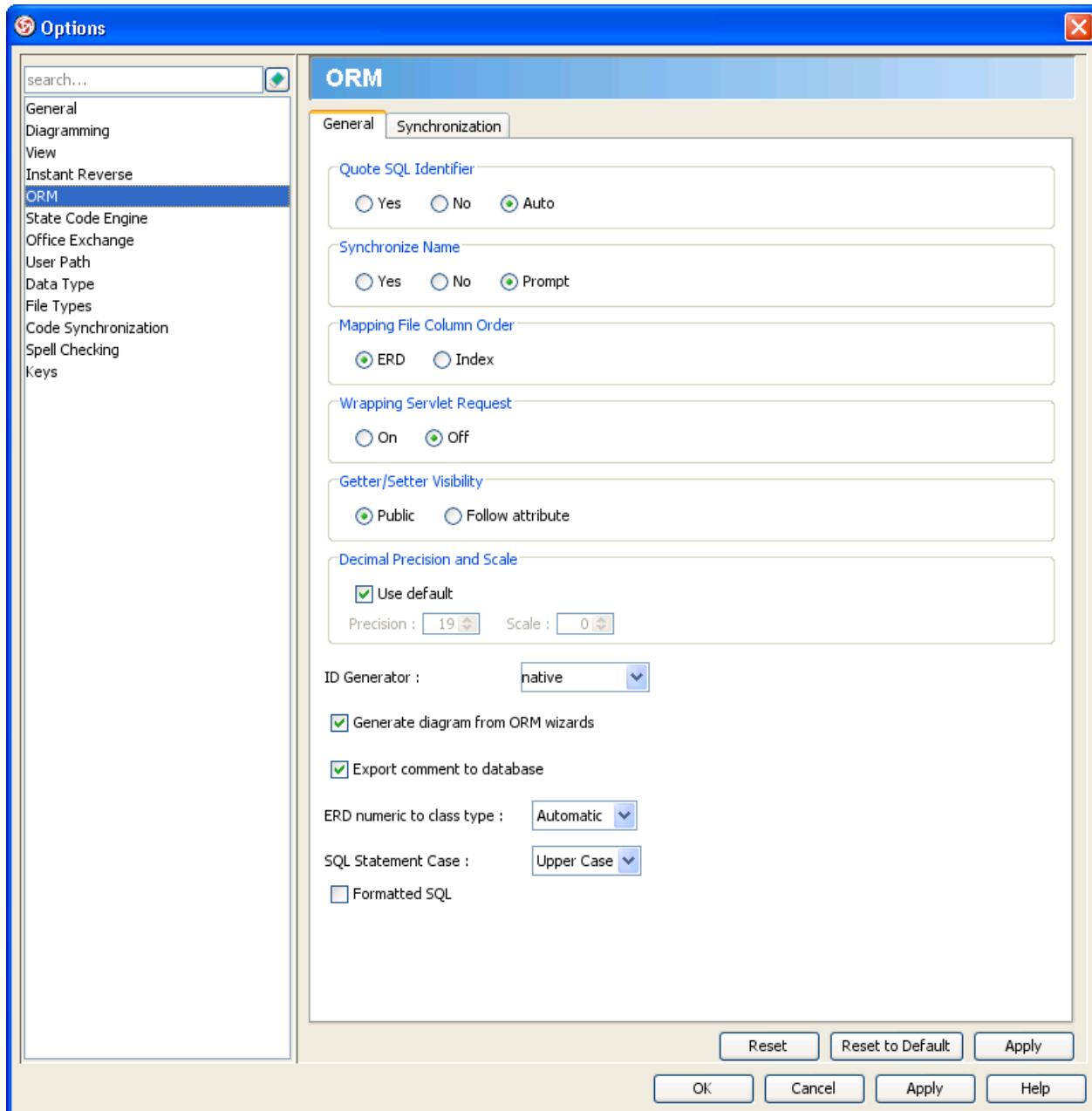


Instant Reverse Options

Option Name	Description
.NET	<ul style="list-style-type: none"> 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++	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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 VP-UML
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

Instant Reverse Options details

ORM options



ORM Options in Option List

General

ORM

General **Synchronization**

Quote SQL Identifier
 Yes No Auto

Synchronize Name
 Yes No Prompt

Mapping File Column Order
 ERD Index

Wrapping Servlet Request
 On Off

Getter/Setter Visibility
 Public Follow attribute

Decimal Precision and Scale
 Use default
Precision : Scale :

ID Generator :

Generate diagram from ORM wizards

Export comment to database

ERD numeric to class type :

SQL Statement Case :

Formatted SQL

General Options in ORM

Option Name	Description
Quote SQL Identifier	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> ERD - (default) Generate columns in mapping file in same order as ERD Index - Generate index columns first in mapping file
Wrapping Servlet Request	<ul style="list-style-type: none"> On - Automatic lock persistable object when get by HttpSession.getAttribute() Off - (default) Do not lock object automatically
Getter/Setter Visibility	<ul style="list-style-type: none"> Public - (default) Generate public getter/setter Follow attribute - Getter/setter visibility follow attribute's visibility
Decimal Precision and Scale - Use default	(default true) Automatic 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	<ul style="list-style-type: none"> assigned - lets the application to assign an identifier to the object before 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 , or , 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 , or . increment - generates identifiers of type , or that are unique only when no other process is inserting data into the same table. <i>Do not use in a cluster.</i> native - (default) picks , or depending upon the capabilities of the underlying database. seqhilo - uses a hi/lo algorithm to efficiently generate identifiers of type , or , given a named database sequence. sequence - uses a sequence in DB2, PostgreSQL, Oracle. The returned identifier is of type , or .
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	<ul style="list-style-type: none"> Automatic - (default) Automatic 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	<ul style="list-style-type: none"> 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

Synchronization

ORM

General Synchronization

Entity => Class Name

Prefix :	Class name :	Suffix :
<input type="text"/>	Capitalize <input type="button" value="▼"/>	<input type="text"/>
e.g. username => Username		

Column => Attribute Name

Prefix :	Attribute Name :	Suffix :
<input type="text"/>	Decapitalize <input type="button" value="▼"/>	<input type="text"/>
e.g. Username => username		

Class => Entity Name

Prefix :	Table name :	Suffix :
<input type="text"/>	Don't change <input type="button" value="▼"/>	<input type="text"/>
no change		

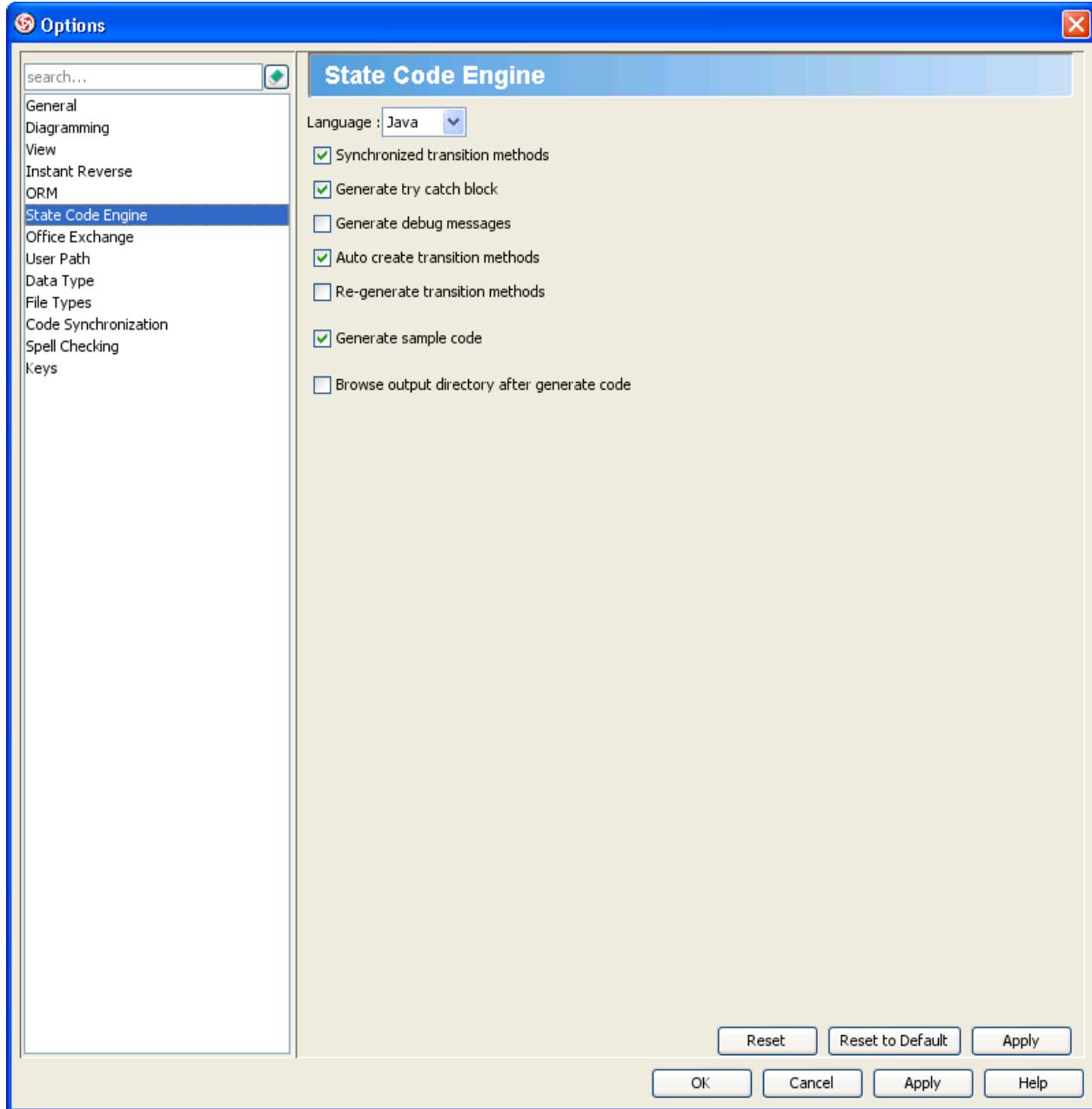
Attribute => Column Name

Prefix :	Column name :	Suffix :
<input type="text"/>	Capitalize <input type="button" value="▼"/>	<input type="text"/>
e.g. username => Username		

Synchronization Options in ORM

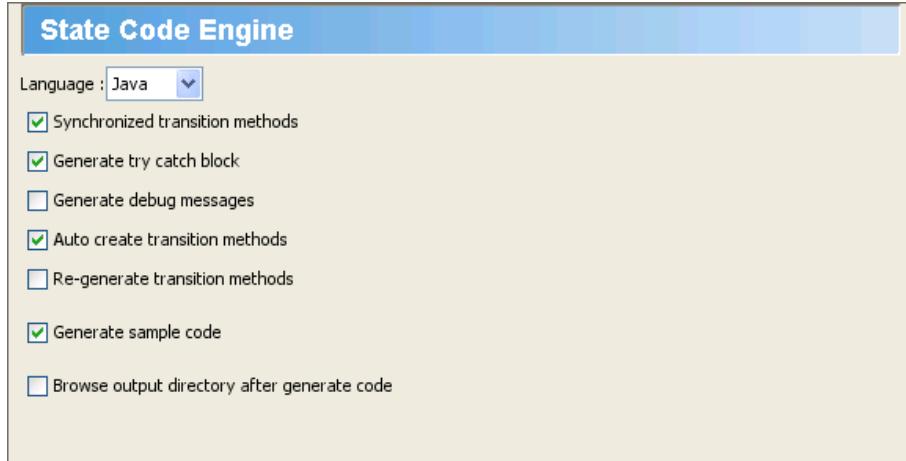
Option Name	Description
Entity => Class Name Prefix	Append characters/words in front of name
Entity => Class Name Class name	<ul style="list-style-type: none"> Capitalize - (default) The first character of each word become uppercase Decapitalize - The first character of each word become lowercase Upper 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
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	<ul style="list-style-type: none"> Capitalize - The first character of each word become 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	<ul style="list-style-type: none"> Capitalize - The first character of each word become uppercase Decapitalize - 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 - (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	<ul style="list-style-type: none"> Capitalize - (default) The first character of each word become uppercase Decapitalize - 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
Attribute => Column Name Suffix	Append characters/words after name

State code engine options



State Code Engine Options in Option List

Options

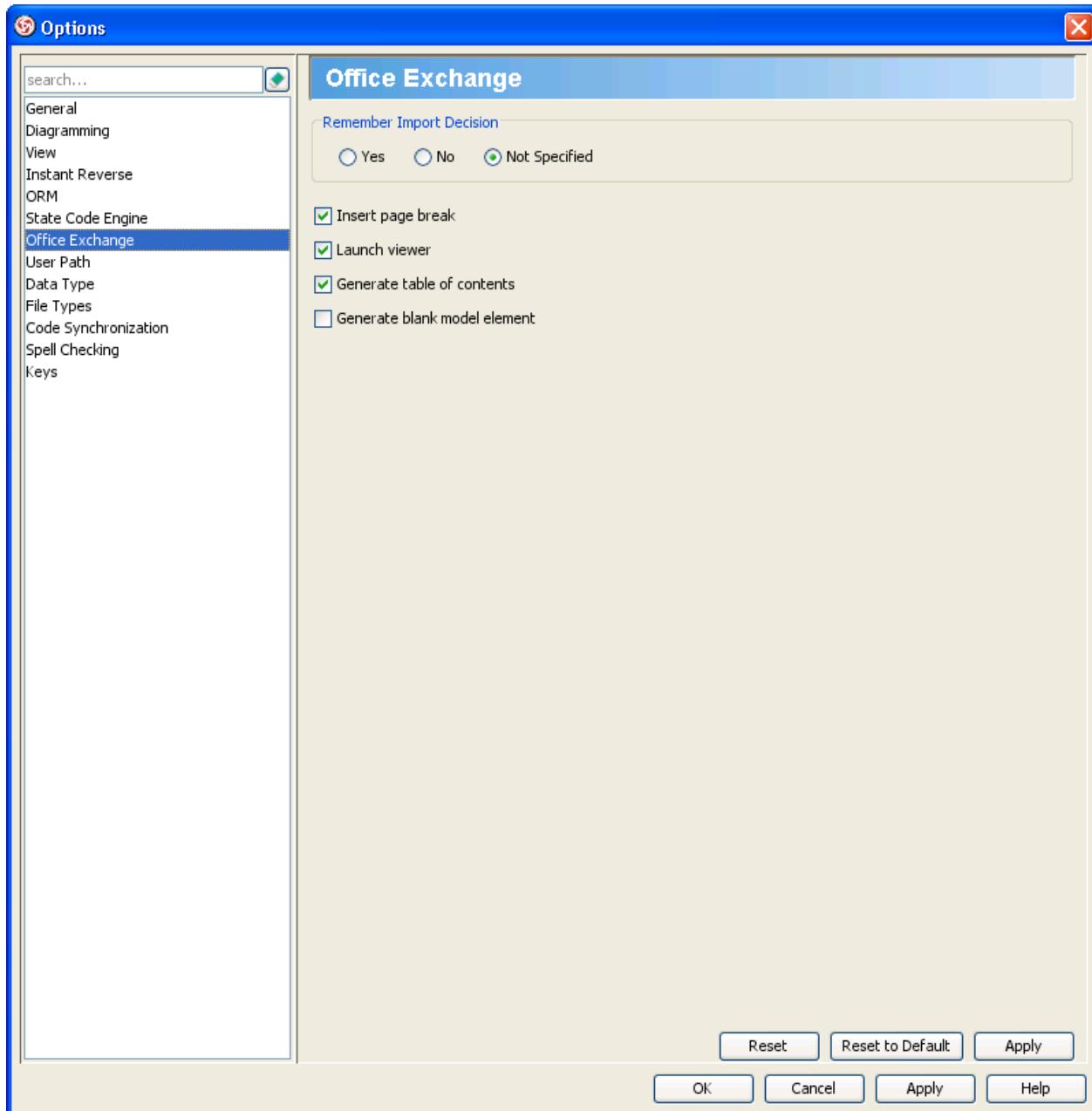


State Code Engine Options

Option Name	Description
Language	<ul style="list-style-type: none"> 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 understand how to work with generated code
Browse output directory after generate code	(default false) Open the directory of generated state code

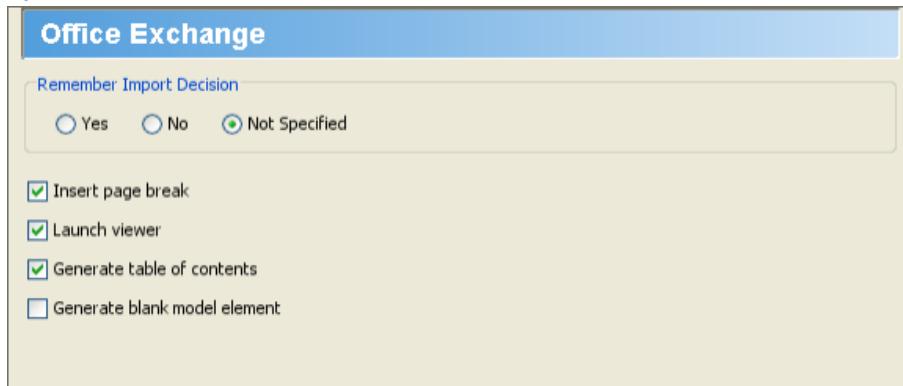
State Code Engine Options details

Office exchange options



Office Exchange Options in Option List

Options



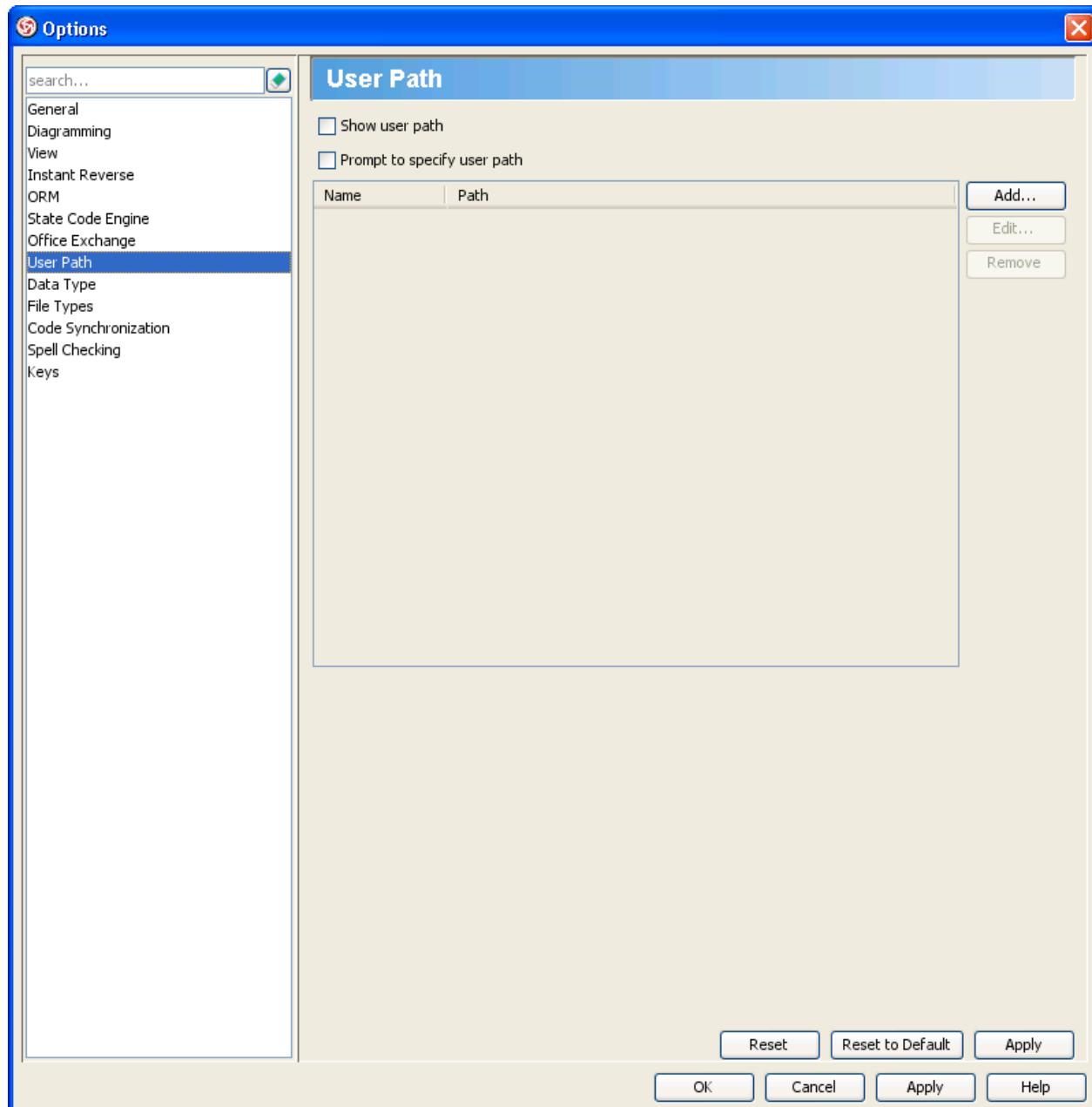
Office Exchange Options

Option Name	Description
Remember Import Decision	VP-UML detects changes made in exported document, and will suggest you to import changes back to VP-UML. This option determines whether the import will be on or off. <ul style="list-style-type: none"> • 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 VP-UML whenever changes are detected.
Insert page break	(default true) Insert a page break for each element
Launch viewer	(default true) Open the document after export
Generate table of contents	(default true) Generate Table of Contents in document
Generate blank model element	(default false) Export an empty form for user to create new model

Office Exchange Options details

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.



User Path Options in Option List

Options

User Path

- Show user path
- Prompt to specify user path

Name	Path	Add...	Edit...	Remove

User Path Options

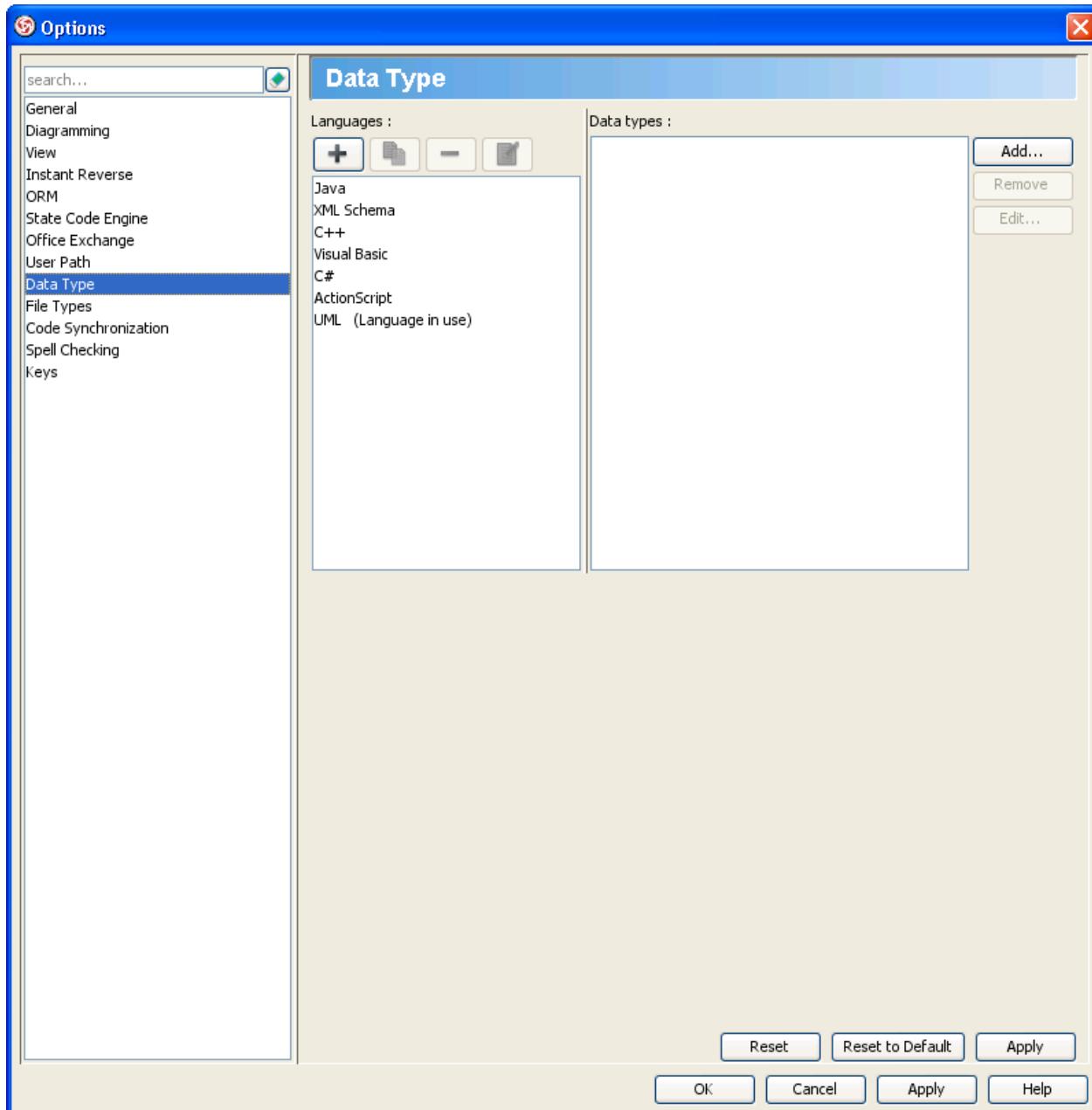
Option Name	Description
Show user path	(default false) Select to show user paths in references, instead of displaying resolved absolute paths. A user path is displayed with its name enclosed by \${ }.
Prompt to specify user path	(default false) When adding a reference comprises a path that is not defined as a user path, you will be prompted to add path as user path

User Path Options details

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.



Data Type Options in Option List

Configure programming language

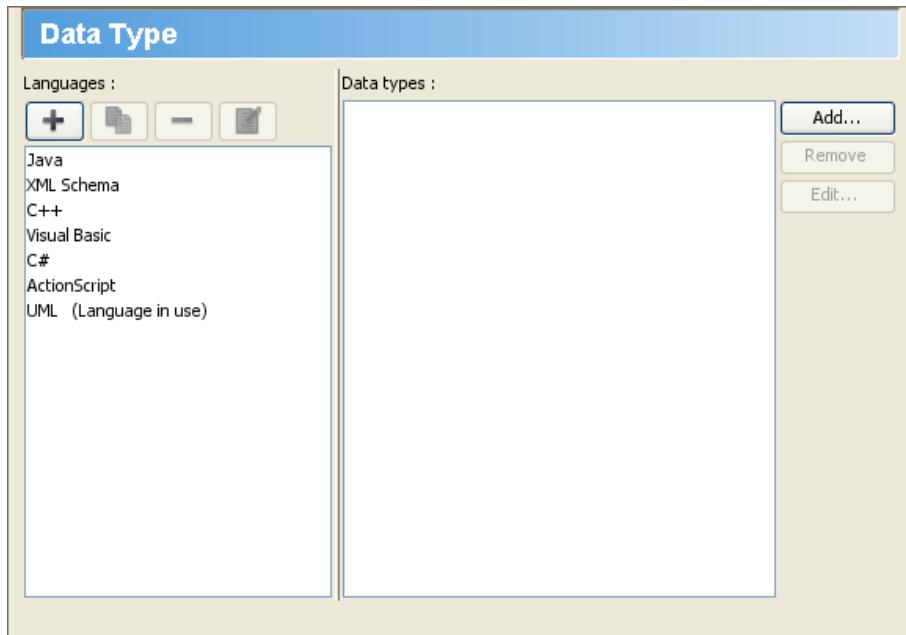
1. Right-click on the project root node under **Diagram Navigator / Model Pane / Class Repository**
2. Select **Configure Programming Language...** from the pop-up menu. This shows the **Programming Language** dialog box.
3. Select the language to switch to.
4. The way how data-type will be mapped from 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. Press on the **plus** sign to add a language.

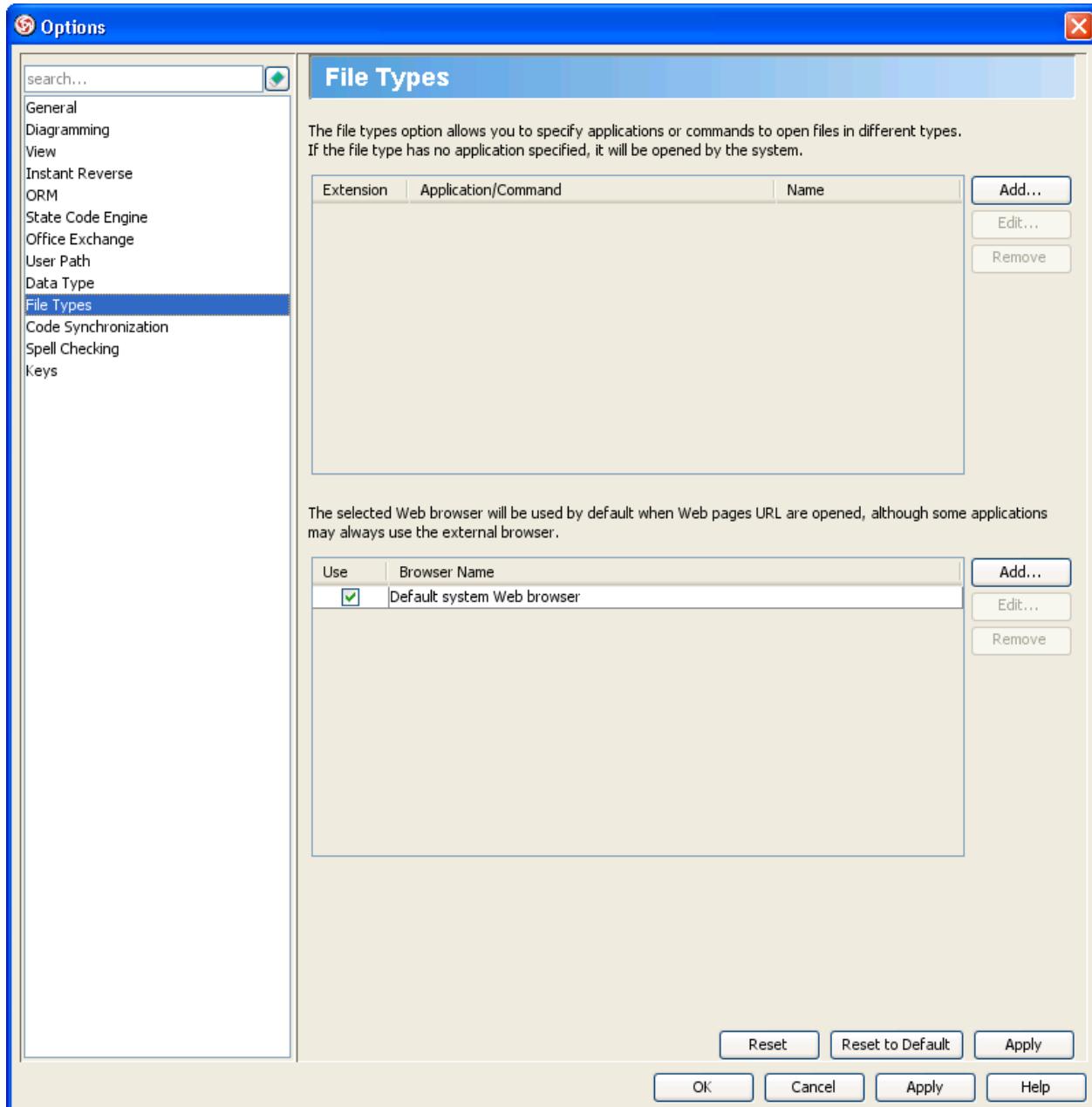
2. Enter its name, and press **OK** to confirm.
3. Press **Add...** to add a data-type to the chosen language.
4. Enter its name, and press **OK** to confirm. From now on, once you have set your own language as the language for your project, you can pickup the associated data-types as attribute type, operation return type and parameter type.



Data Type Options

File type options

Model Reference lets you add reference(s) to external file or URL into diagram element. You can open the referenced file or 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.



File Type Options in Option List

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 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.

File Types

The file types option allows you to specify applications or commands to open files in different types.
If the file type has no application specified, it will be opened by the system.

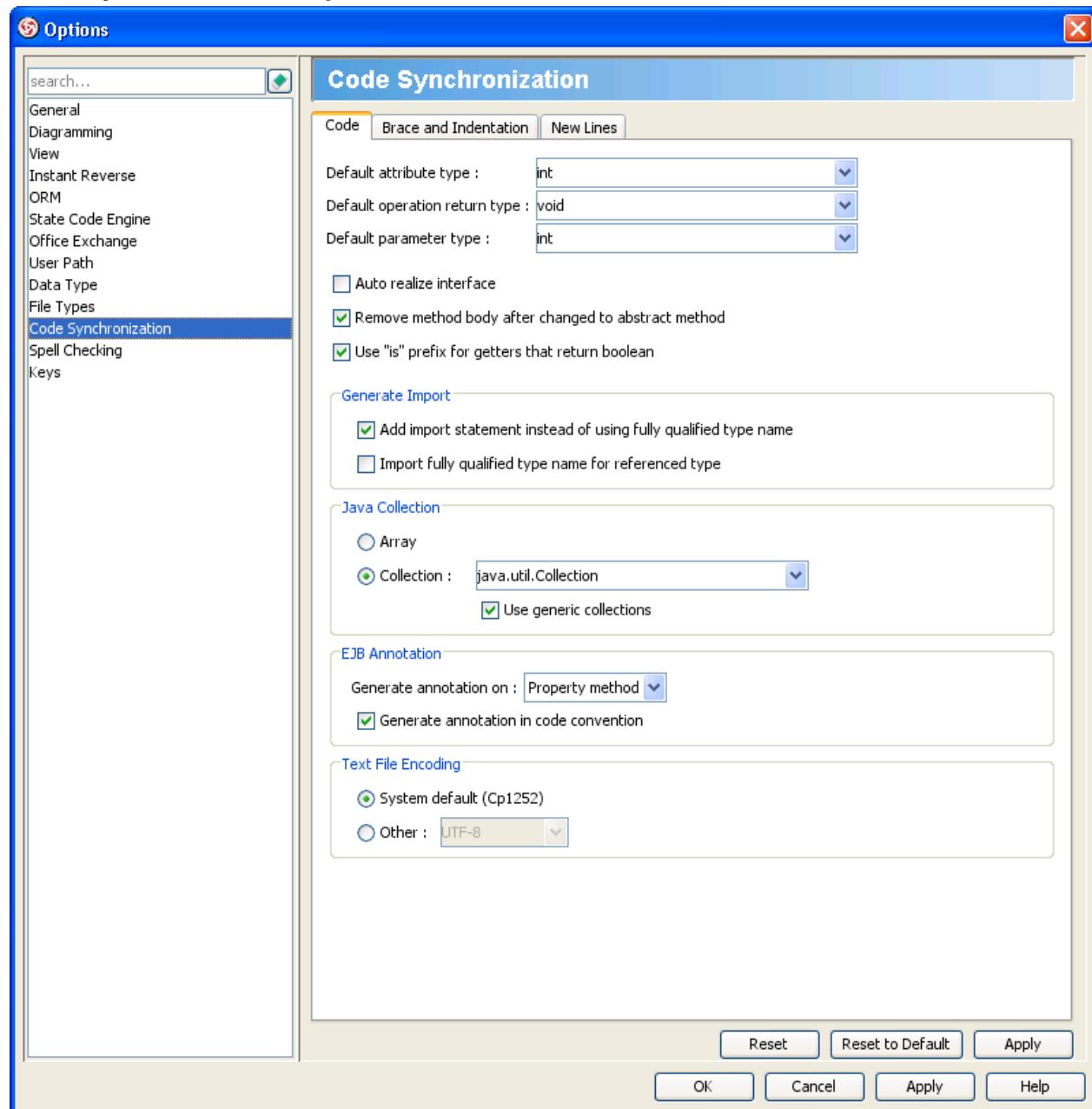
Extension	Application/Command	Name	Add...
			Edit... Remove

The selected Web browser will be used by default when Web pages URL are opened, although some applications may always use the external browser.

Use	Browser Name	Add...
<input checked="" type="checkbox"/>	Default system Web browser	Edit... Remove

File Type Options

Code synchronization options



Code Synchronization Options in Option List

Code

Code Synchronization

Code Brace and Indentation New Lines

Default attribute type :

Default operation return type :

Default parameter type :

Auto realize interface

Remove method body after changed to abstract method

Use "is" prefix for getters that return boolean

Generate Import

Add import statement instead of using fully qualified type name

Import fully qualified type name for referenced type

Java Collection

Array

Collection :

Use generic collections

EJB Annotation

Generate annotation on :

Generate annotation in code convention

Text File Encoding

System default (Cp1252)

Other :

Code Options in Code Synchronization

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
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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

Brace and indentation

Code Synchronization

Code Brace and Indentation New Lines

Brace Positions

Class declaration :	Same line
Constructor declaration :	Same line
Method declaration :	Same line
Enum declaration :	Same line
Annotation type declaration :	Same line

Indentation

Indentation policy :	Tabs
Indentation size :	1

Preview

```
/** 
 * Indentation and Braces
 */
interface ILoader {
    void changeType(int t);
}

enum LoaderType {
    FAST, SAFE, COMPLETE
}

@interface LoaderAnnotationType {
}

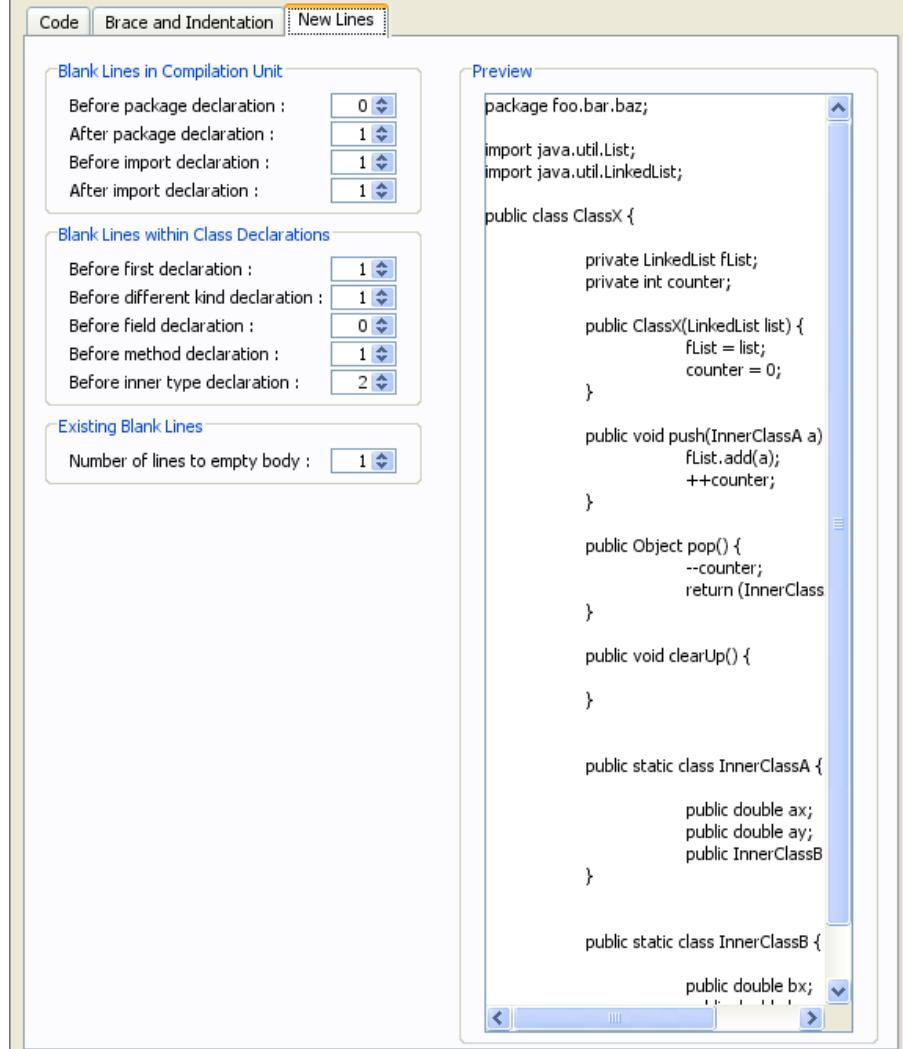
public class ELoader implements ILoader {
    SomeType fType = new SomeType();
    ELoader() {
    }

    void changeType(int t) {
        fType.setType(t);
    }
}
```

Option Name	Description
Class declaration	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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

Code Synchronization



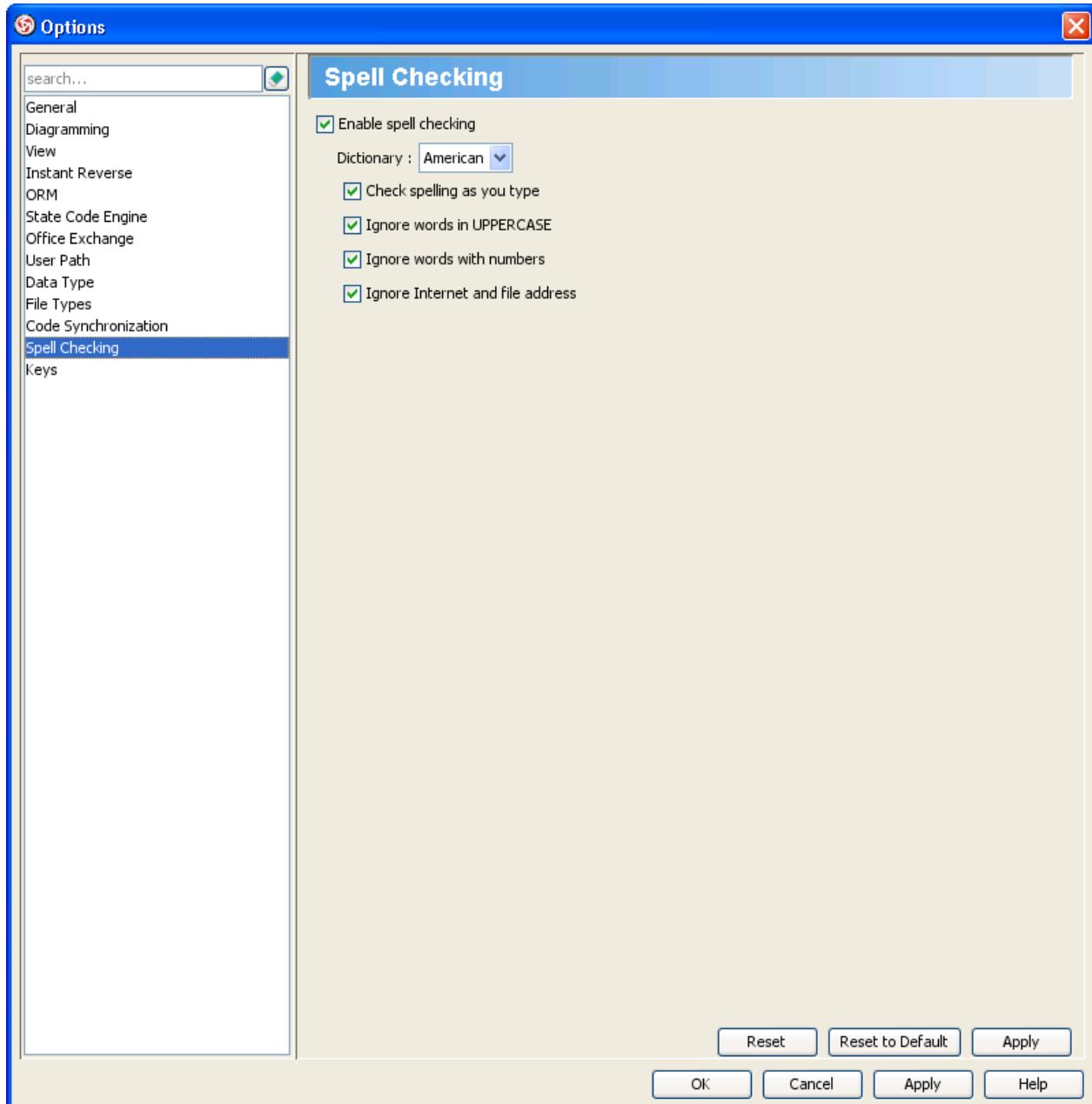
New Lines Options in Code Synchronization

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

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.



Spell Checking Options in Option List

Options



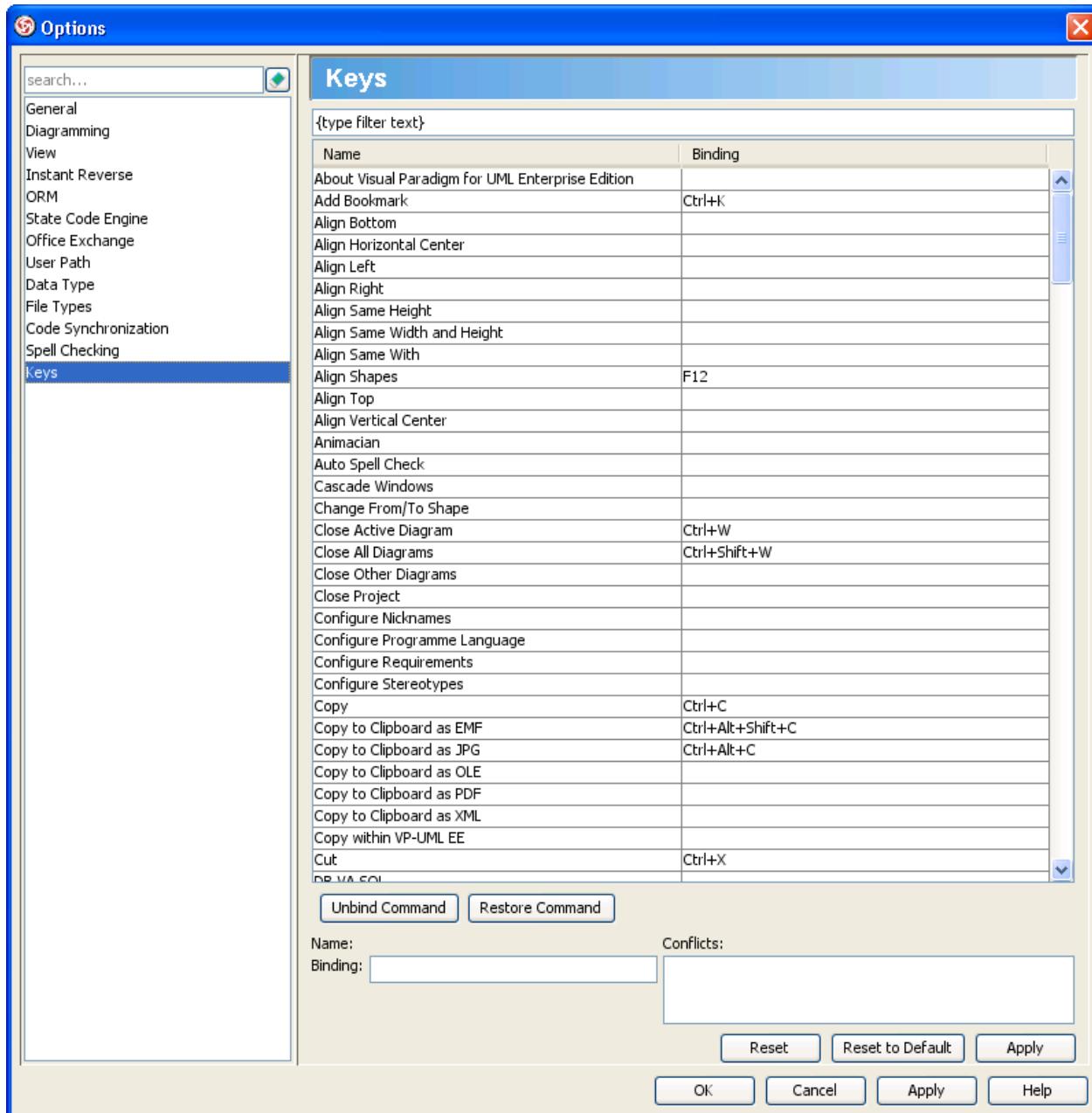
Spell Checking Options

Option Name	Description
-------------	-------------

Enable spell checking	(default true) Enable spell checking.
Dictionary	<p>The choose of dictionary affects the judgment of correctness of words.</p> <ul style="list-style-type: none">• 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

Keys options



Keys Options in Option List

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 VP-UML.

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** 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.

Keys

{type filter text}

Name	Binding
About Visual Paradigm for UML Enterprise Edition	
Add Bookmark	Ctrl+K
Align Bottom	Alt+Shift+4
Align Horizontal Center	
Align Left	Alt+Shift+1
Align Right	Alt+Shift+3
Align Same Height	
Align Same Width and Height	
Align Same With	
Align Shapes	F12
Align Top	Alt+Shift+2
Align Vertical Center	
Animacian	
Auto Spell Check	
Cascade Windows	
Change From/To Shape	
Close Active Diagram	Ctrl+W
Close All Diagrams	Ctrl+Shift+W
Close Other Diagrams	
Close Project	
Configure Nicknames	
Configure Programme Language	
Configure Requirements	
Configure Stereotypes	
Copy	Ctrl+C
Copy to Clipboard as EMF	Ctrl+Alt+Shift+C
Copy to Clipboard as JPG	Ctrl+Alt+C
Copy to Clipboard as OLE	
Copy to Clipboard as PDF	
Copy to Clipboard as XML	
Copy within VP-UML EE	
Cut	Ctrl+X

[Unbind Command](#)

[Restore Command](#)

Name:

Binding:

Conflicts:

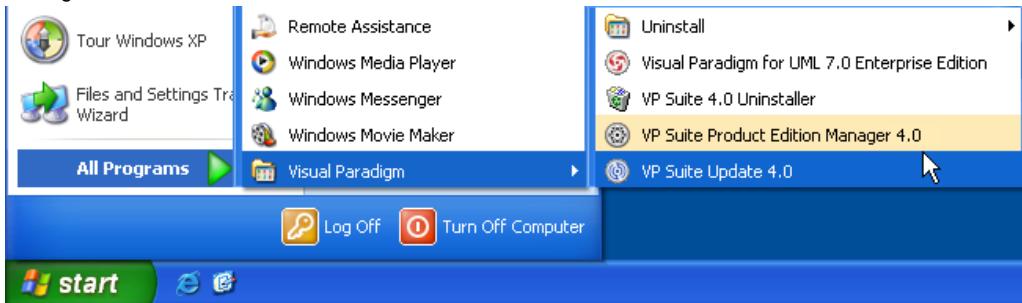
Keys Options

Updating VP-UML with VP Suite Update

Updating of VP-UML can be done by updating Visual Paradigm Suite. To update Visual Paradigm Suite:

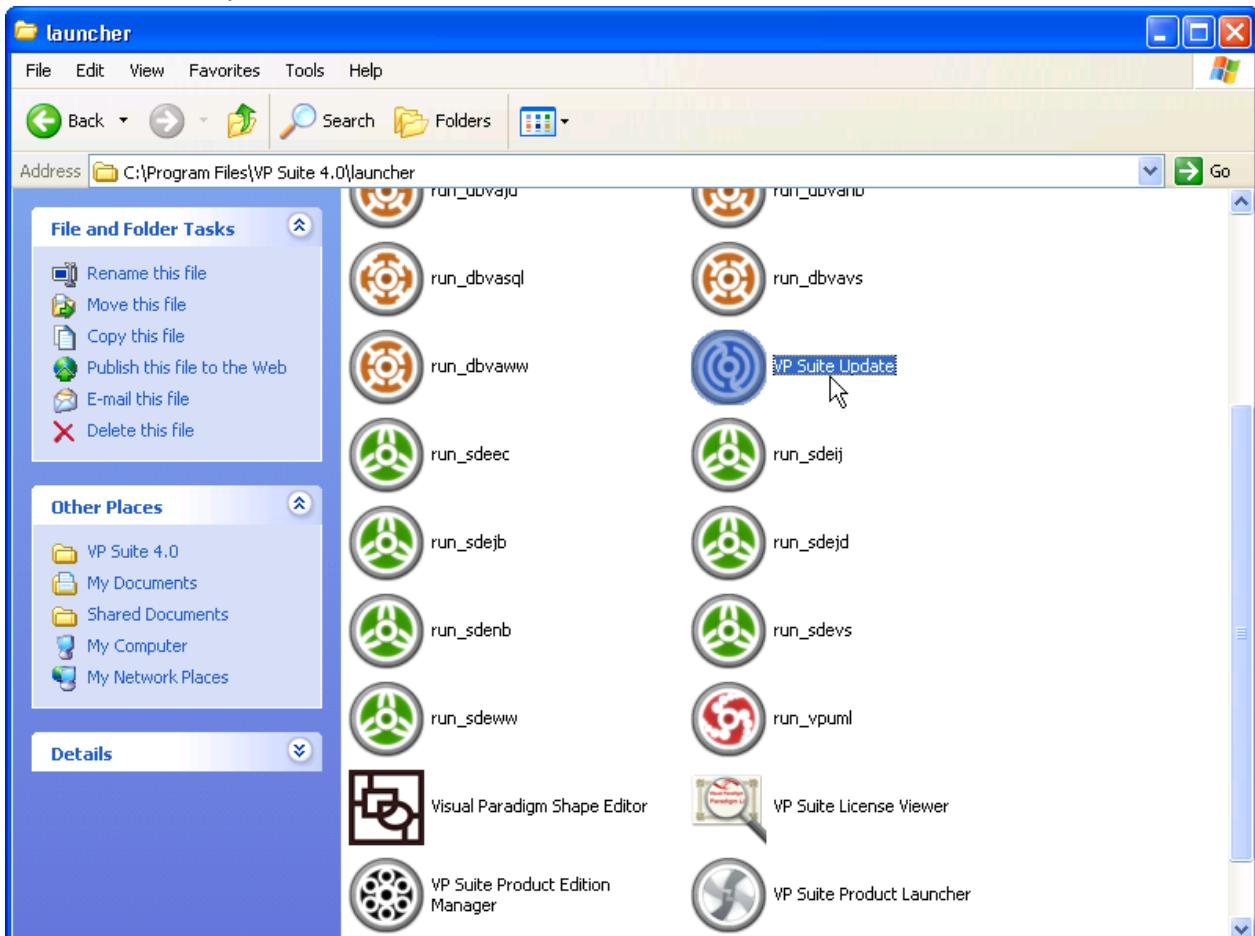
1. Launch VP Suite Update by any of the ways below:

- Through the Start Menu.



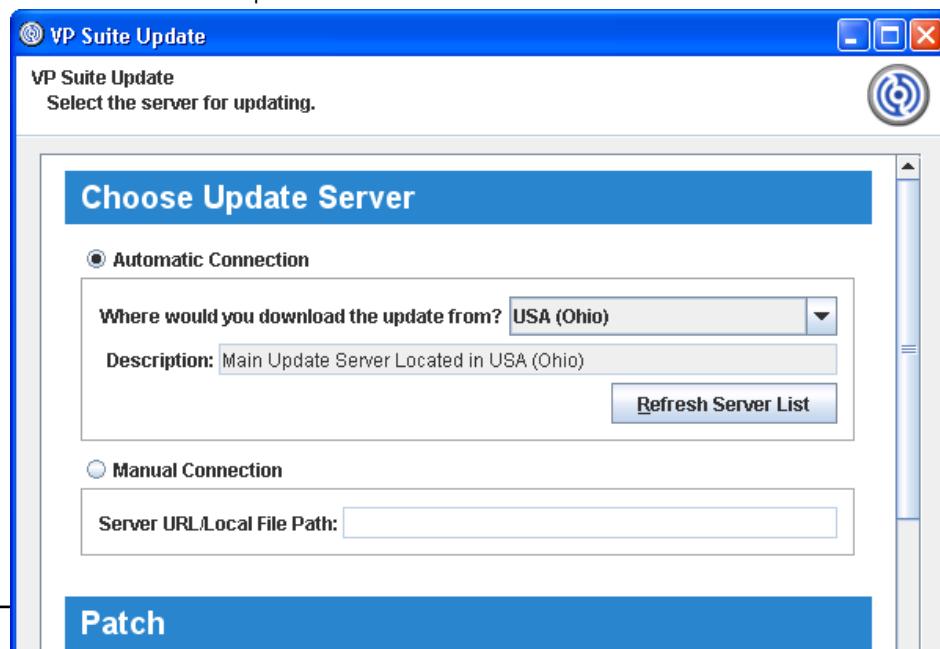
Starting VP Suite Update via Start Menu

- By executing the launcher **VP Suite Update** in the launcher folder of VP Suite installation directory. Users in all operating system can start VP-UML in this way.

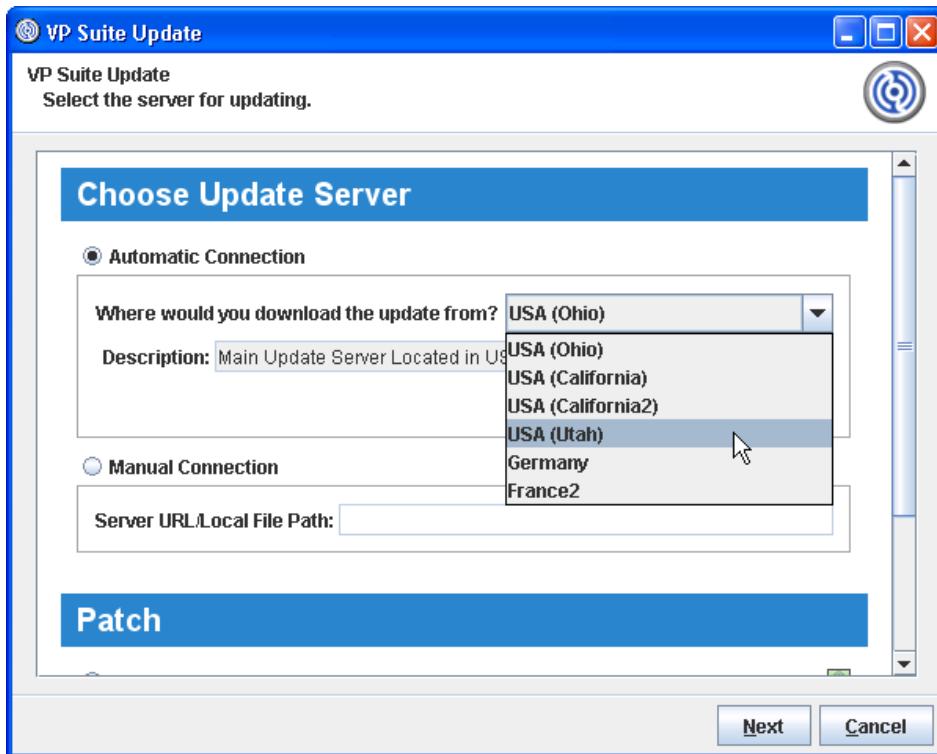


Starting VP Suite Update with launcher

This starts the VP Suite Update.



2. Select a server from the drop-down menu for checking and downloading update files. The list of servers can be refreshed by using the **Refresh Server List** button.



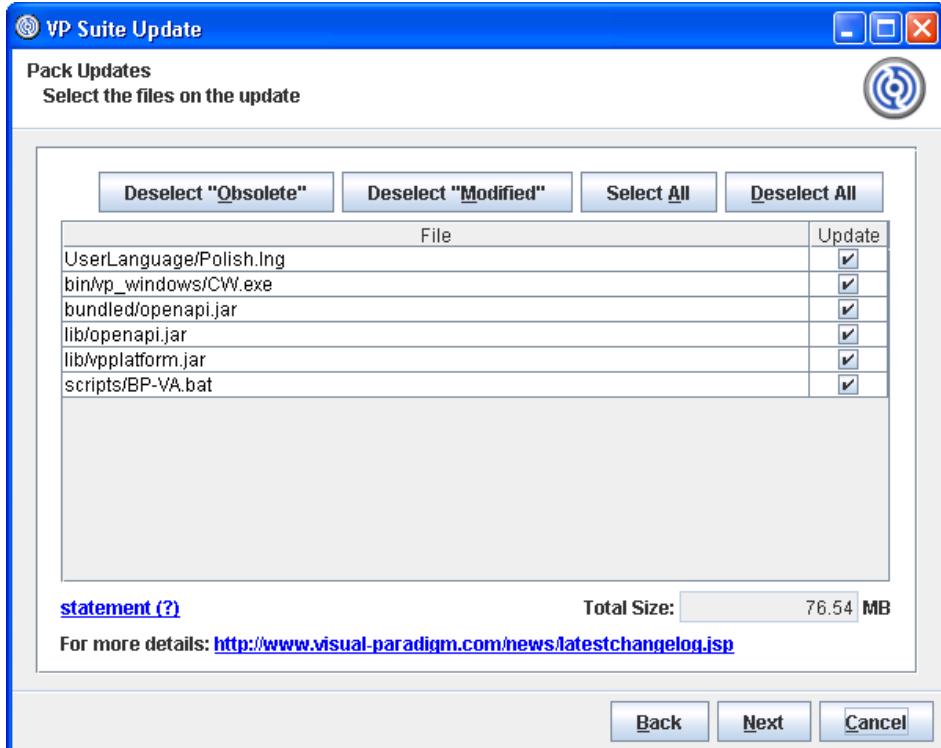
Selecting a location to check and download updates

NOTE: If there is a connection problem when downloading the update files from the specified server under **Automatic Connection**, VP Suite Update will connect to another server randomly instead of the specified connection to continue the update automatically.

NOTE: By using the Update Synchronizer to keep the files in the update repository up-to-date, network client can specify the or file path of the update repository using the **Manual Connection**.

NOTE: Patch is special build for specific users. You should choose the **Update to Latest Patch** option only when you are recommended by Visual Paradigm Support Team.

3. Click **Next** to continue. After checking is completed, the updated files are listed in the **Pack Updates** page.



The Pack Updates page for selecting files to be updated

4. Review the files to be updated. You may select or deselect files to be updated by checking and unchecking individual file listing in the page, under the **Update** column. Usually, you can keep the selection unchanged. But if you see the filenames appear in blue or red, you better confirm the selection carefully. Below is a description of red and blue colored filename:

Color	Description
Blue	Modified. This refers to files which are modified by the users and will be replaced with the original files
Red	Obsolete. This refers to files which are no longer useful in the new version of the product and will be deleted.

Description of blue and red colored filename in Pack Updates page

If you want to keep the modified and obsolete files, press the **Deselect "Obsolete"** and **Deselect "Modified"** buttons so that these files will not be updated.

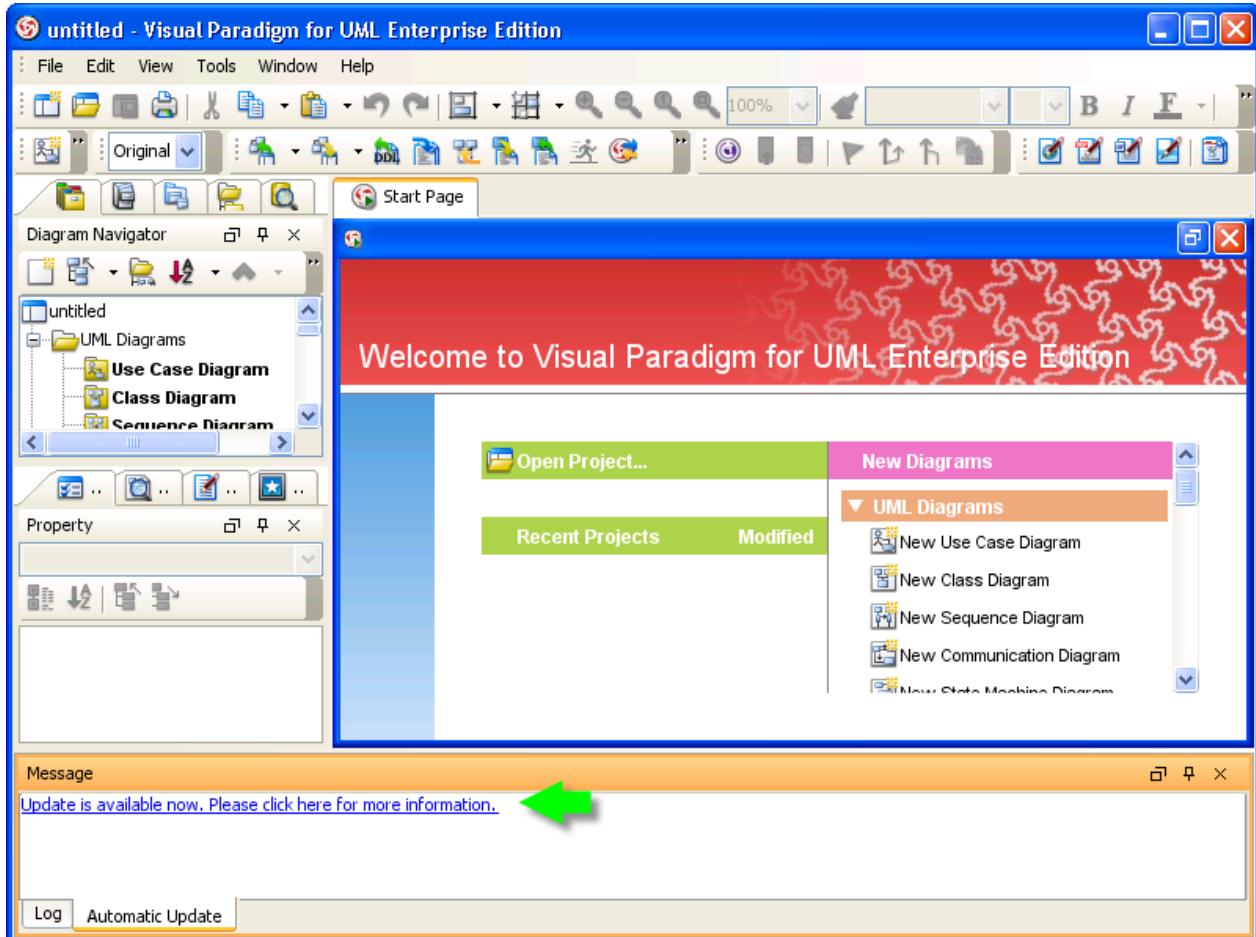
5. Click **Next** to proceed with updating the files.
6. Click **Finish** to confirm.

Automatic update notification

When you run VP-UML, checking of possible updates can be 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.

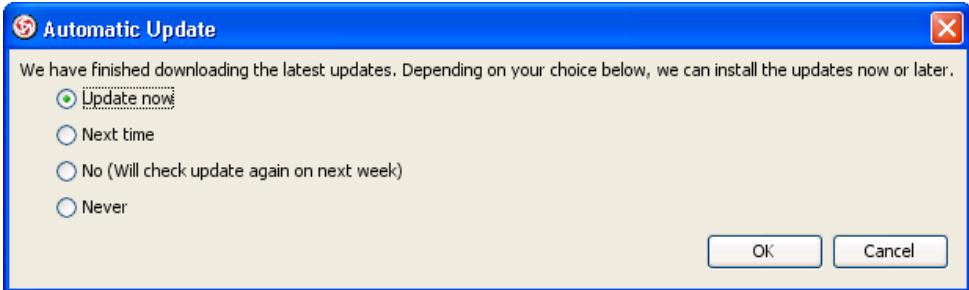
Updating when running VP-UML

1. When running VP-UML, a message " **Update is available now. Please click here for more information**" 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. Click on the message. This popup the **Automatic Update** dialog box.



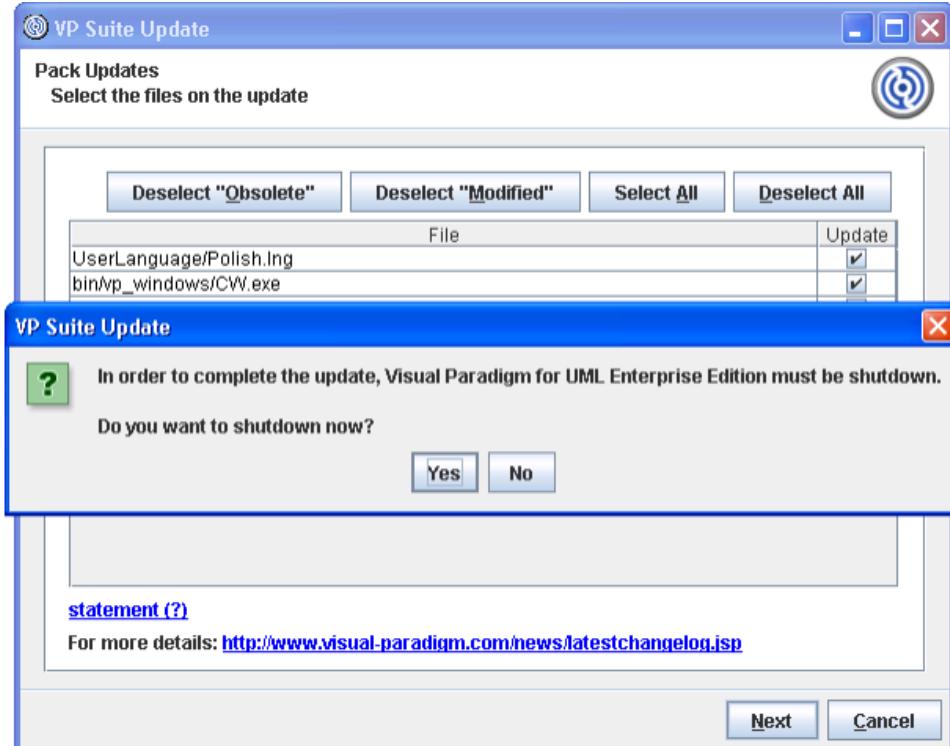
Automatic update options

Here are the available options in the dialog box.

Option	Description
Update now	Run the product update now
Next time	Check for product updates next time when starting VP-UML
No	Check for product updates after a certain period of time (The interval can be defined in the Options dialog box. Refer to the section below for details)
Never	Do not check for product updates anymore

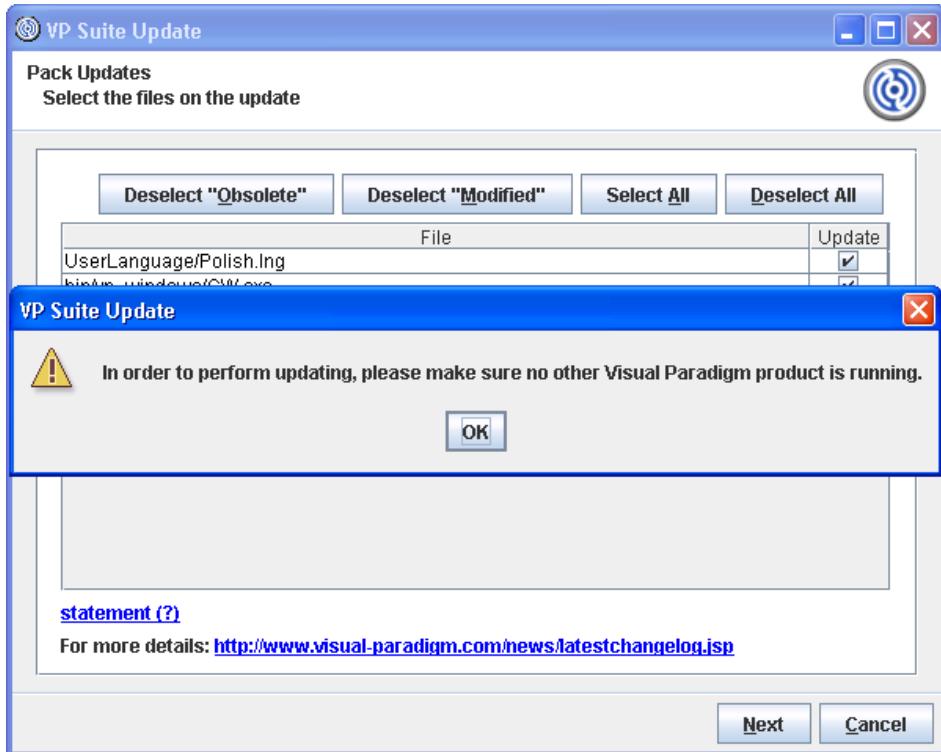
Available options for automatic update

3. Select **Update now** and click **OK** to proceed. This popup the **VP Suite Update**.



To shutdown VP-UML

4. In order to update, you need to close all the running VP application. Click **Yes** first, and then click **OK** in the dialog shown below to let VP Suite Update close the applications for you.



Asked to close Visual Paradigm products

5. Review the files to be updated. You may select or deselect files to be updated by checking and unchecking individual file listing in the page, under the **Update** column. Usually, you can keep the selection unchanged. But if you see the filenames appear in blue or red, you better confirm the selection carefully. Below is a description of red and blue colored filename:

Color	Description
Blue	Modified. This refers to files which are modified by the users and will be replaced with the original files
Red	Obsolete. This refers to files which are no longer useful in the new version of the product and will be deleted.

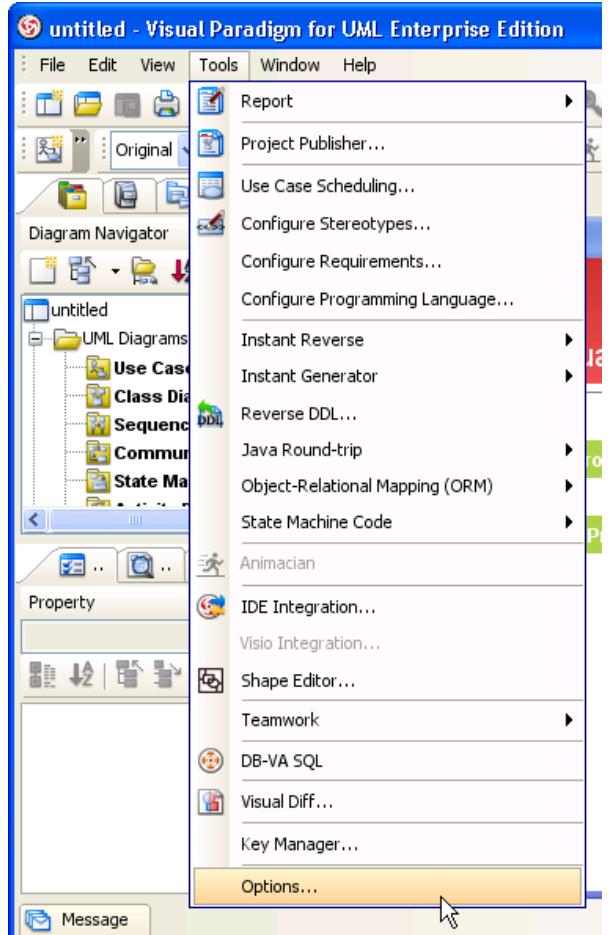
Description of blue and red colored filename in Pack Updates page

6. Click **Next** to proceed with updating the files.
 7. When update is complete, click **Finish** to confirm.

Setting the interval of checking updates

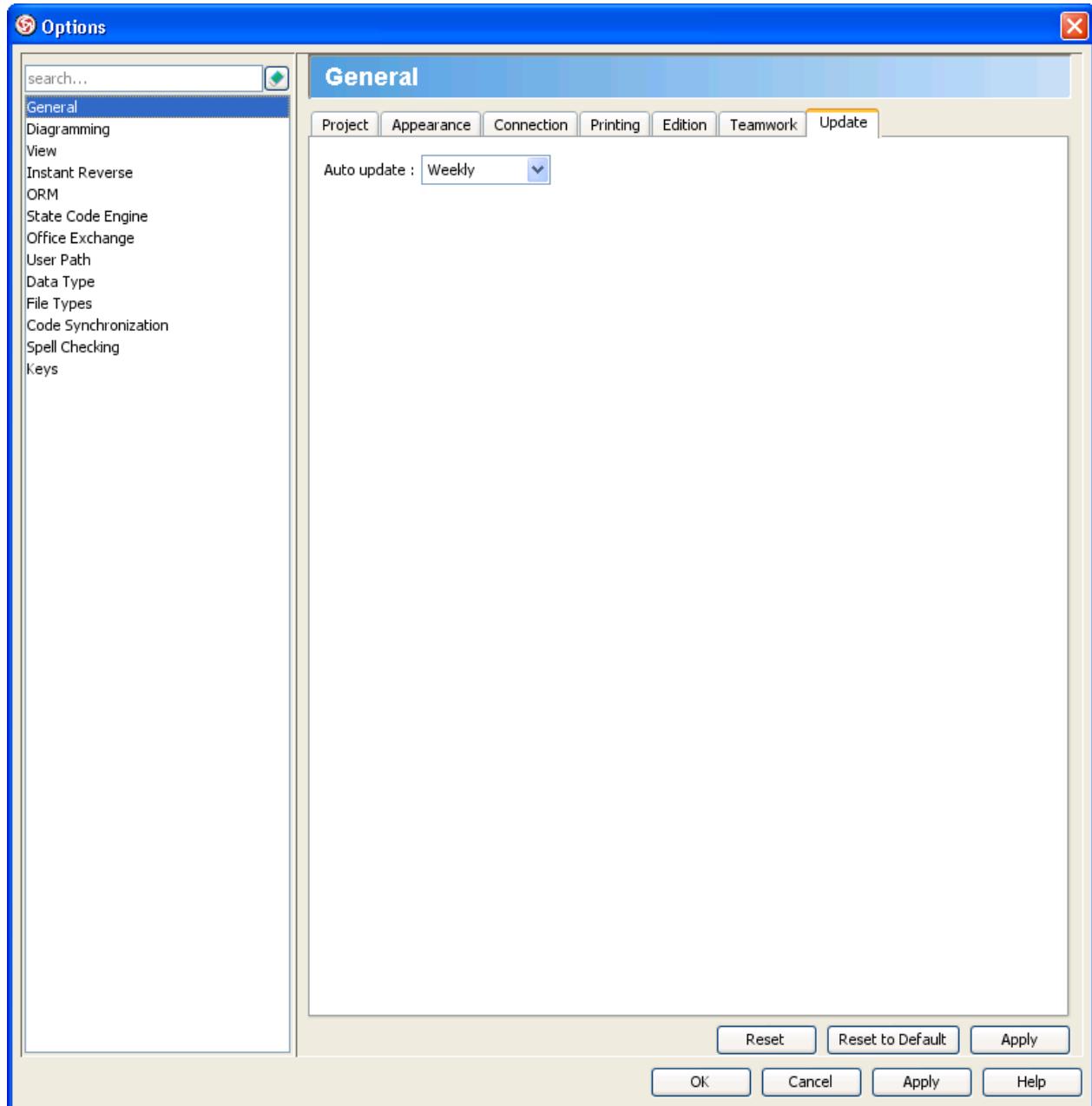
By default, update is checked weekly when starting VP-UML. You can change the interval of checking updates through the Options dialog box. To change:

1. Open the **Options** dialog box by selecting **Tools > Options...** from the main menu.



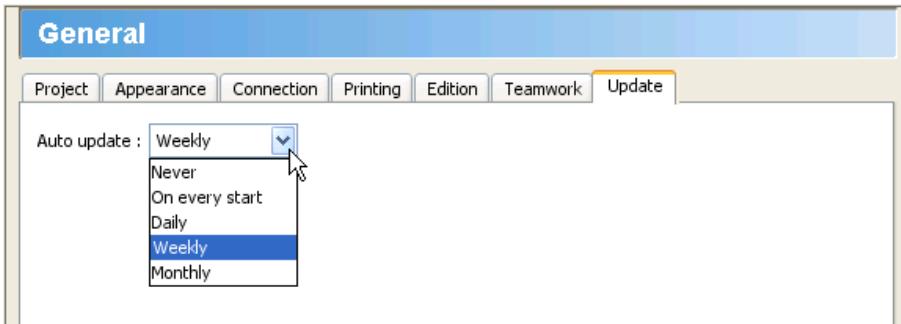
Opening the Options dialog box

2. In the **Options** dialog box, select **General** from the list at the left hand side, then open the **Update** tab.



Update page in Options dialog box

3. Select the interval of performing auto update from the **Auto update** drop down menu.



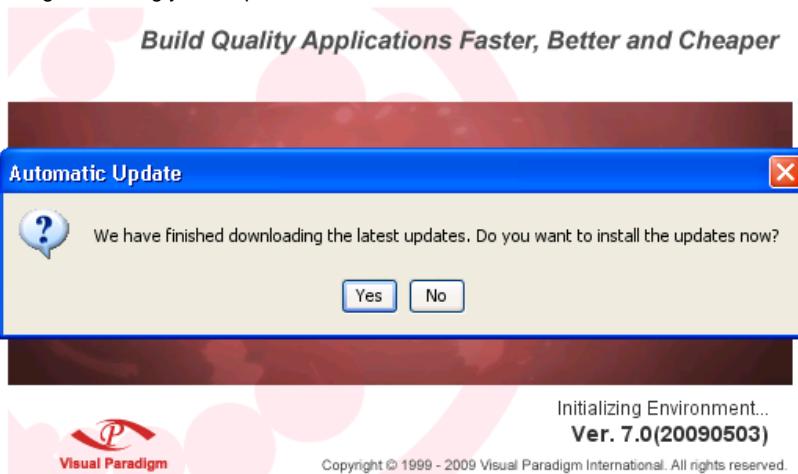
Selecting update interval

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 VP-UML
Daily	Check for product updates everyday, when starting VP-UML
Weekly	Check for product updates every week, when starting VP-UML
Monthly	Check for product updates every month, when starting VP-UML

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.



Prompting for update when starting VP-UML

Introduction to plugin support

Plugin Support provides an interface for developers to integrate with VP-UML. Developers can develop their plugins for what they want. In this section, we will introduce the structure of a plugin.

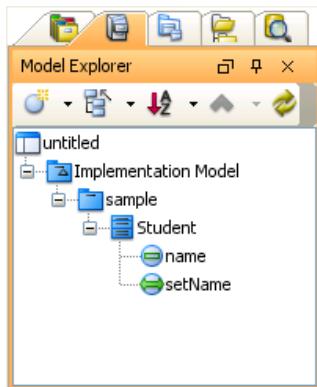
Plugin.xml

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 VP-UML 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).

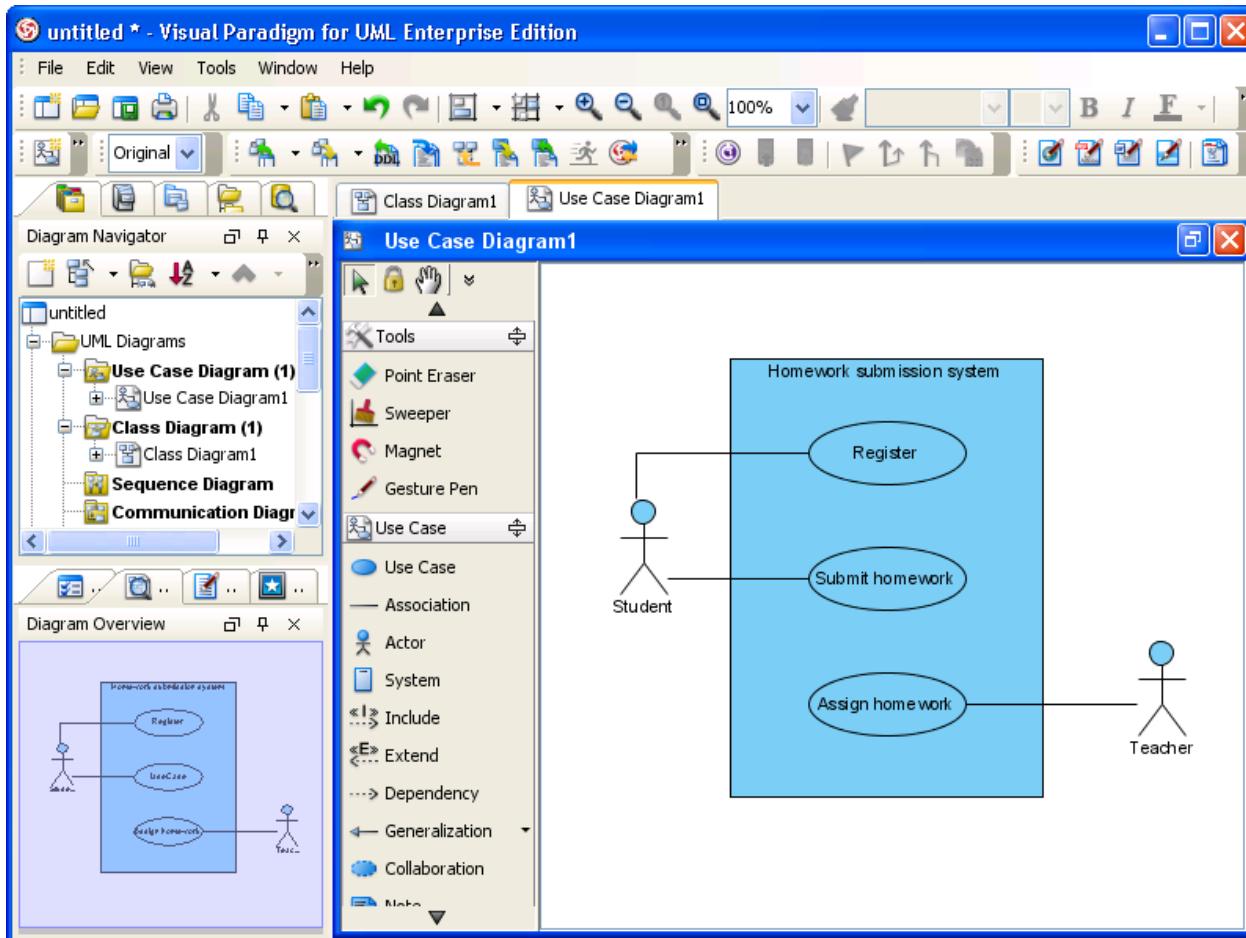


Model Elements under Model Explorer

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)

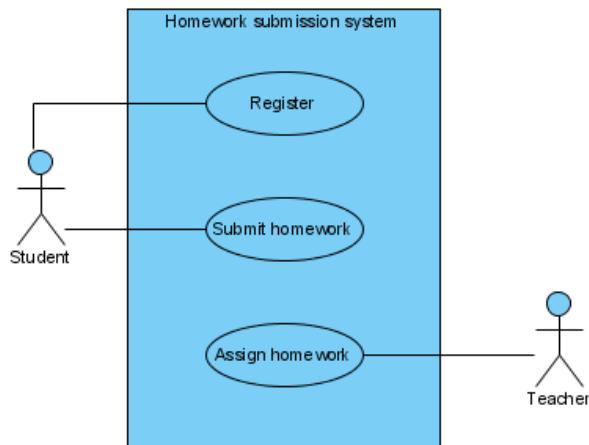


An opening Use Case Diagram

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).



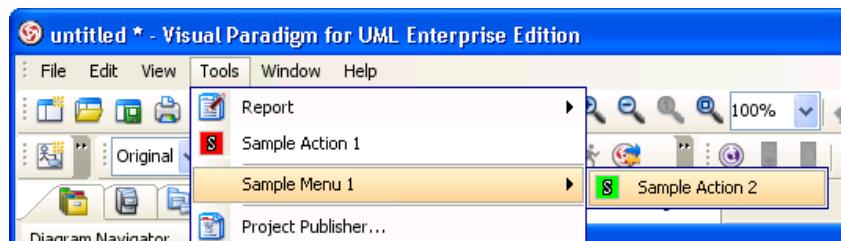
Shapes and connectors are both diagram elements

Action/Action controller

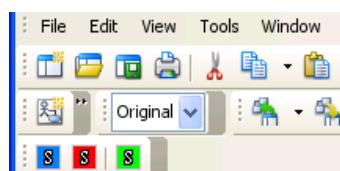
Action represents buttons and menus (menu, toolbar and popup menu), which contains the information on outlook (such as label, icon, mnemonic, etc...) and responses to trigger the function call.

Action is used to represent the button on 3 regions: menu/toolbar, popup menu and diagram toolbar

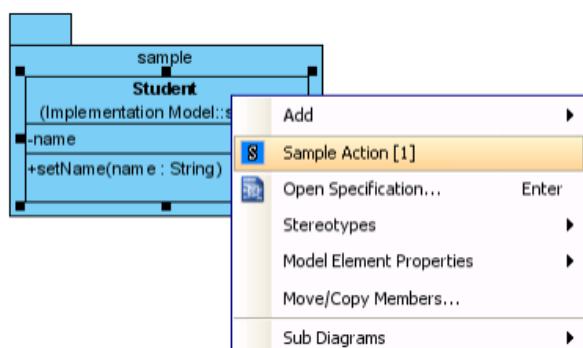
Action Controller is the control (function call) of actions. Developer needs to implement different Action Controller on different region's actions.



Menu with user-defined menus



Toolbar with user-defined buttons



Popup menu with user-defined menus

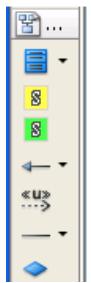


Diagram toolbar with user-defined buttons

Implementing plugin

Configuring development environment

Plugin Support API is placed on %VP_SUITE%/lib/openapi.jar. In order to working with VP-UML, 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, should be start from writing the plugin.xml. The basic directory structure is "VP_SUITE_HOME/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 assignment 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..*)		Specifies 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)	Specifies 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: %VP_SUITE%/plugins)of VP-UML from the VPPluginInfo.
    }
    public void unloaded() {
        // called when the plugin is unloaded (when the VP-UML 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

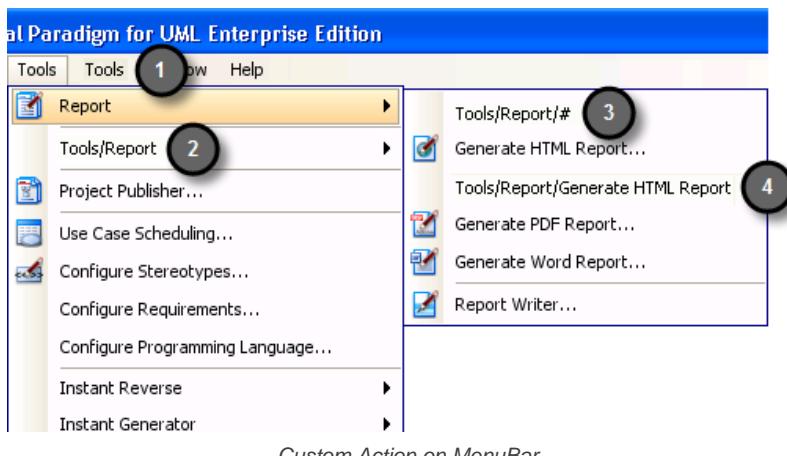
There are differences on xml definition and code implementation of the 3 kinds of Actions (menu/toolbar, popup menu, diagram toolbar).

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 are 4 examples will be given:



Custom Action onMenuBar

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/Report	Tools/Report		Under the Tools menu, after the Report menu
3 Tools/Report/#	Tools/Report/#		Under the Tools menu, and under the Report menu, place on the front
4 Tools/Report/Generate HTML Report	Tools/Report/Generate HTML Report		Under the Tools menu, and under the Report menu, after the Generate HTML Report 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/Report"/>
    <action
        id= "sample.plugin.actions.Action1"
        actionType= "generalAction"
        label= "Sample Action 1"
        tooltip= "Sample Action 1"
        icon= "icons/red.png"
        style= "normal"
        menuPath= "Tools/Report"
        toolbarPath= "sample.plugin.actions.Toolbar1/#">
        <actionController class= "sample.plugin.actions.ActionController"/>
    </action>
</actionSet>
```

```
< separator  
    id= "sample.plugin.actions.Separator1"  
    menuPath= "Tools/sample.plugin.actions.Action1"  
    toolbarPath= "sample.plugin.actions.Toolbar1/sample.plugin.action.Action1"/>  
</ actionSet>
```

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.
toolbar (0..*)		Specifies a toolbar, contains the location information of the toolbar.
toolbar (0..*)	orientation [north east south west]	Specifies which side will be the toolbar placed on.
toolbar (0..*)	index [(number) last new]	Based on the orientation, where will be the toolbar placed. e.g. the orientation is "north" and there is 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..*)		Specifies a menu or pull down button on menu bar or toolbar. It contains the outlook information of the menu.
action (0..*)		Specifies 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		Specifies 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.VPActionC
separator (0..*)		Specified 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 updated(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
contextSensitiveActionSet (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		Specifies 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 updated(
        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 trigger. 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 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 of 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"
        actionPerformed= "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"
        actionPerformed= "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" 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.

Description on Code:

VPShapeController (com.vp.plugin.diagram.VPShapeController)

It response to handle the outlook of the shape on the diagram.

Sample:

```
package sample.plugin.actions;
// import the necessities
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 VP-UML. 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 refers 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 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 VP-UML. 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 assolter = classModel.fromRelationshipEndIterator();
while (assolter.hasNext()) {
    IRelationshipEnd relationshipEnd = (IRelationshipEnd) assolter.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 it sub-class is necessary. For example, developer get the **IModelElement** from the popup menu's context. Developer check whether the model is a **IClass**, then developer cast 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 VP-UML. 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 contains by the diagrams.

Creating diagrams/Diagram elements

Developer can create the diagram or diagram element by **com.vp.plugin.DiagramManager**. DiagramManager can be access 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 VP-UML. 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 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 VP-UML

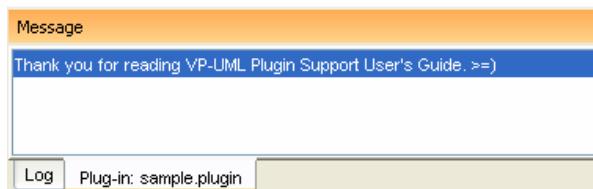
Since VP-UML may be integrated with different platforms which may not support Swing (e.g. Eclipse, Visual Studio). That may make to hang on the process if using the Swing dialog technology (e.g. JOptionPane and JDialog). So, there is necessary to use a special method to show the dialog with Swing technology.

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 VP-UML'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 VP-UML, 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 VP-UML 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 VP-UML 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.parseInt( this._inputField1.getText());
        int b = Integer.parseInt( this._inputField2.getText());
        int c = Integer.parseInt( this._answerField.getText());
        return (a+b == c);
    }
    catch (Exception ex) { return false; }
}
}

```

Deploying plugin

After prepared all the required files for a plugin (plugin.xml, plugin.properties, classes/libraries and other resources), developer can plug the plugin into VP-UML.

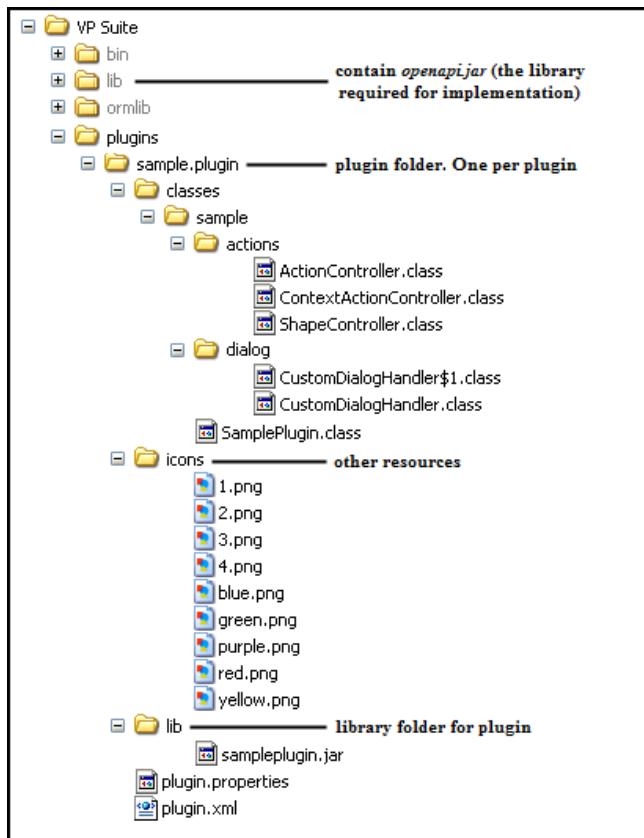
First, create a folder named **plugins** (notice the ' s') in the VP-Suite directory. Put the plugin files into "%VP-SUITE%\plugins\%PLUGIN_ID%". %PLUGIN_ID% is a directory named as the plugin id (use the id as the directory name to avoid duplicated directories defined in **plugins**)

The following structure should be obtained:

%VP_SUITE%

```
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 VP Suite installation folder with plugin created in the **plugins** folder.



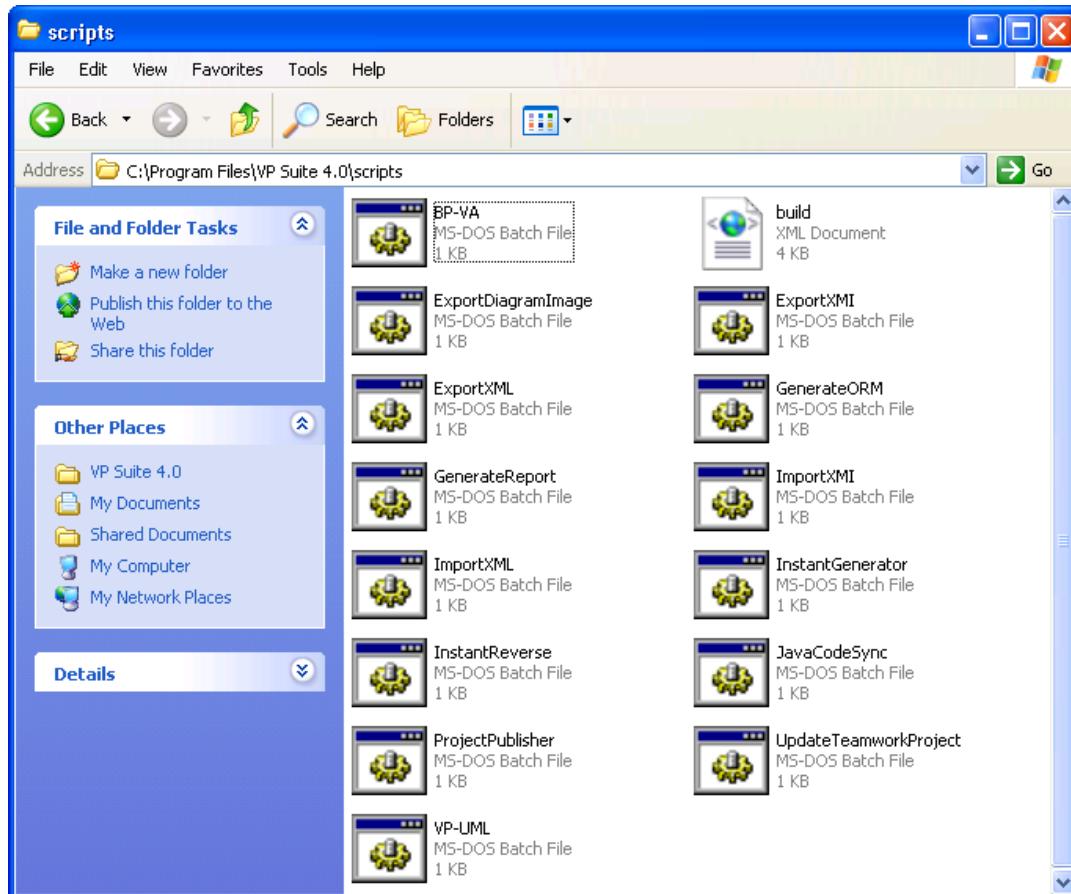
Plugin folder structure

After all, restart VP-UML will see the plugin available. If not, make sure the code was written correctly and can be compiled, and you have setup the above folder structure correctly.

Exporting diagram image

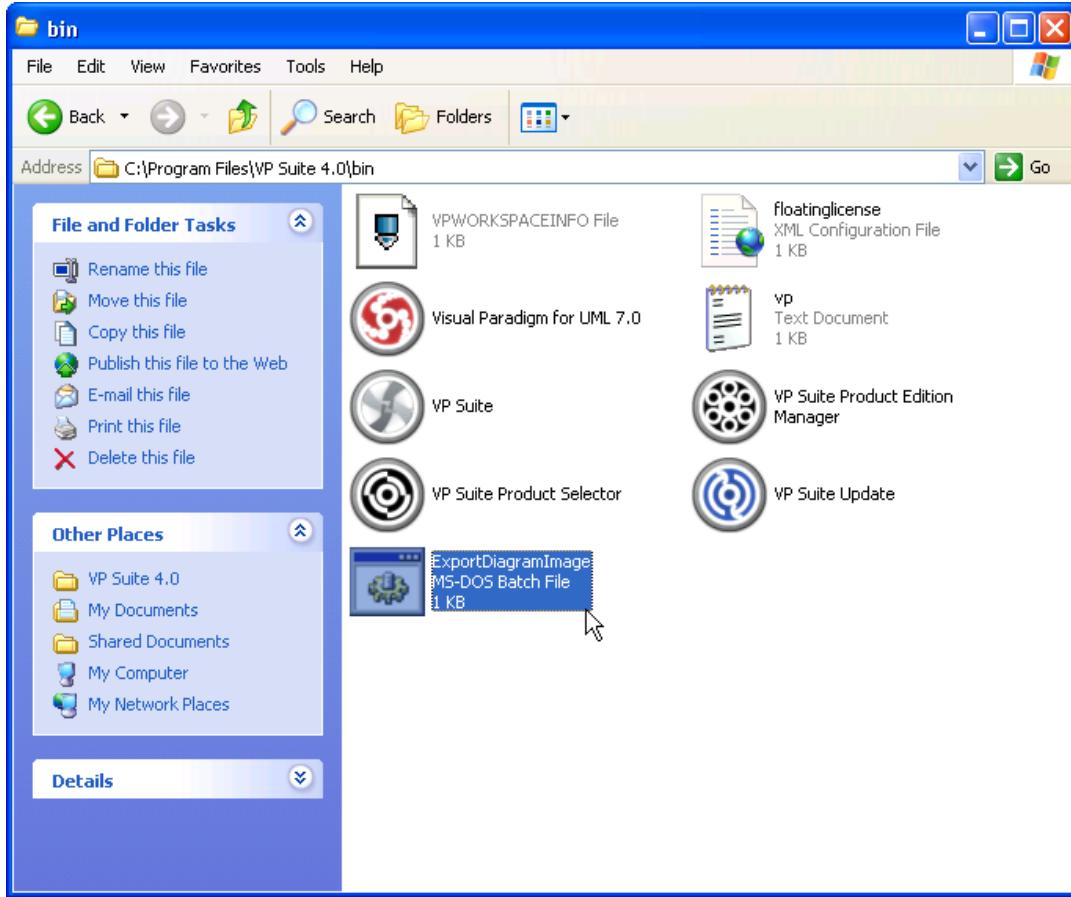
To export images from a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ExportDiagramImage** and paste to the bin folder of VP Suite installation directory.



Copy and paste ExportDiagramImage from scripts folder to bin folder

3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window shows the command line: 'C:\>cd C:\Program Files\VP Suite 4.0\bin' followed by a blank line indicating the command was executed successfully.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:

`ExportDiagramImage -project C:\Demo\Demo.vpp -out C:\Demo\Output -diagram "*" -type jpg`

```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>ExportDiagramImage -project C:\Demo\Demo.vpp
-out C:\Demo\Output -diagram "*" -type jpg
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
No JNI Library 8
No JNI Library 8
No JNI Library 8
C:\Program Files\UP Suite 4.0\bin>_
```

Executing ExportDiagramImage

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: <ul style="list-style-type: none"> • png • png_with_background • jpg • svg • pdf 	png

Parameters for ExportDiagramImage

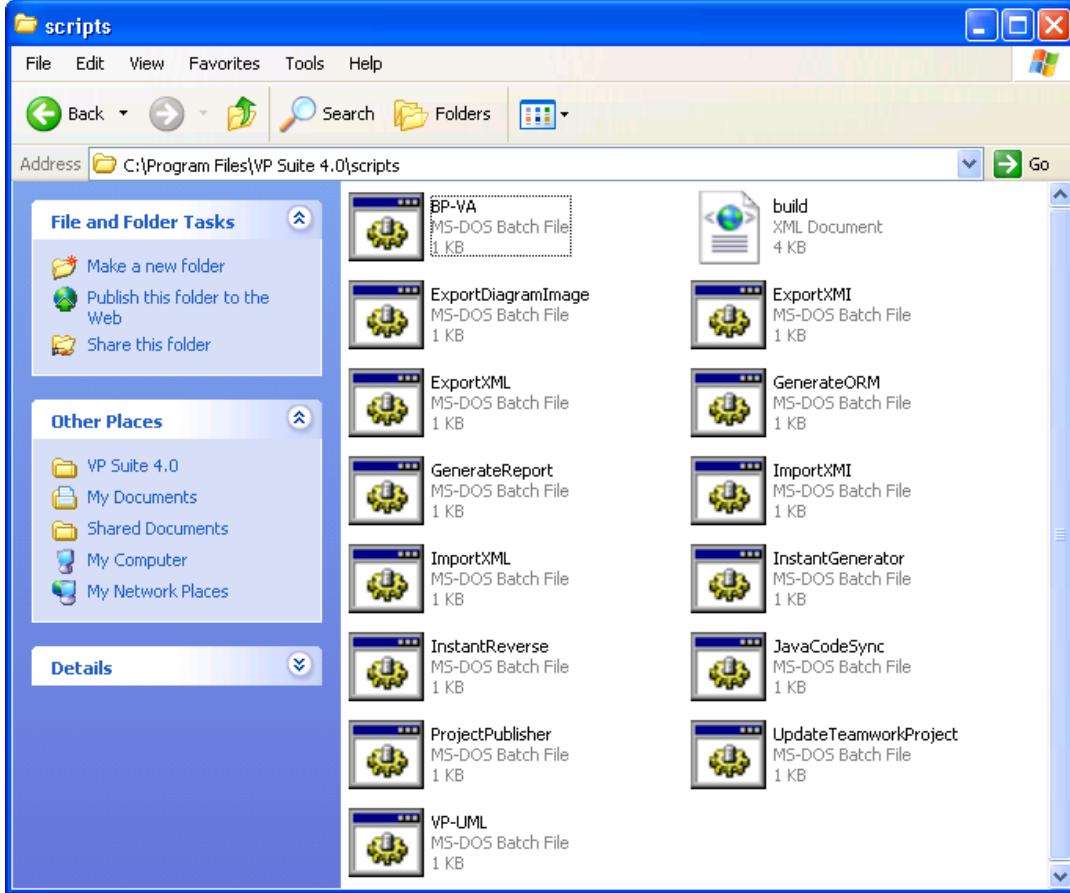
Exporting and import XMI

VP-UML 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

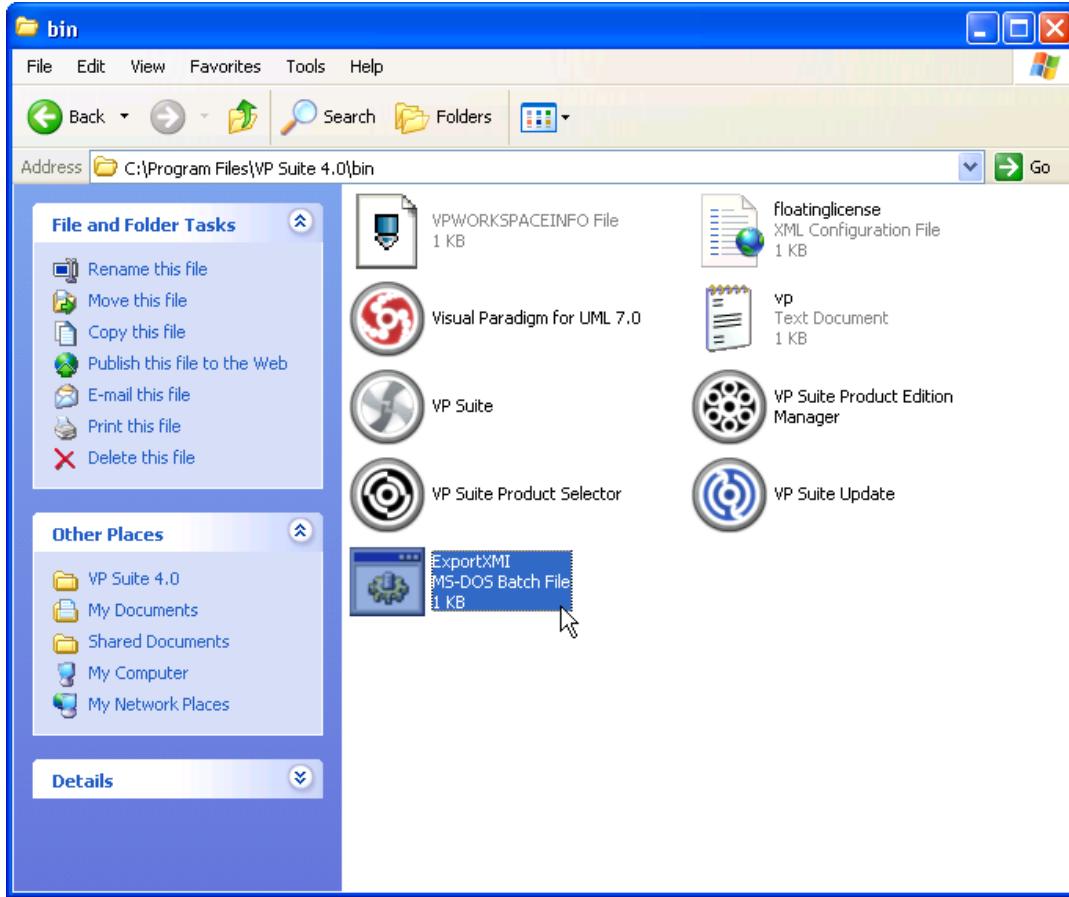
To export XMI from a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ExportXMI** and paste to the bin folder of VP Suite installation directory.



Copy and paste ExportXMI from scripts folder to bin folder

3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin  
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

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

```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>ExportXMI -project C:\Demo\Demo.vpp -out C:\Demo\Output\Sample.xmi -type 2.1
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
C:\Program Files\UP Suite 4.0\bin>_
```

Executing ExportXMI

Below is a description of parameters:

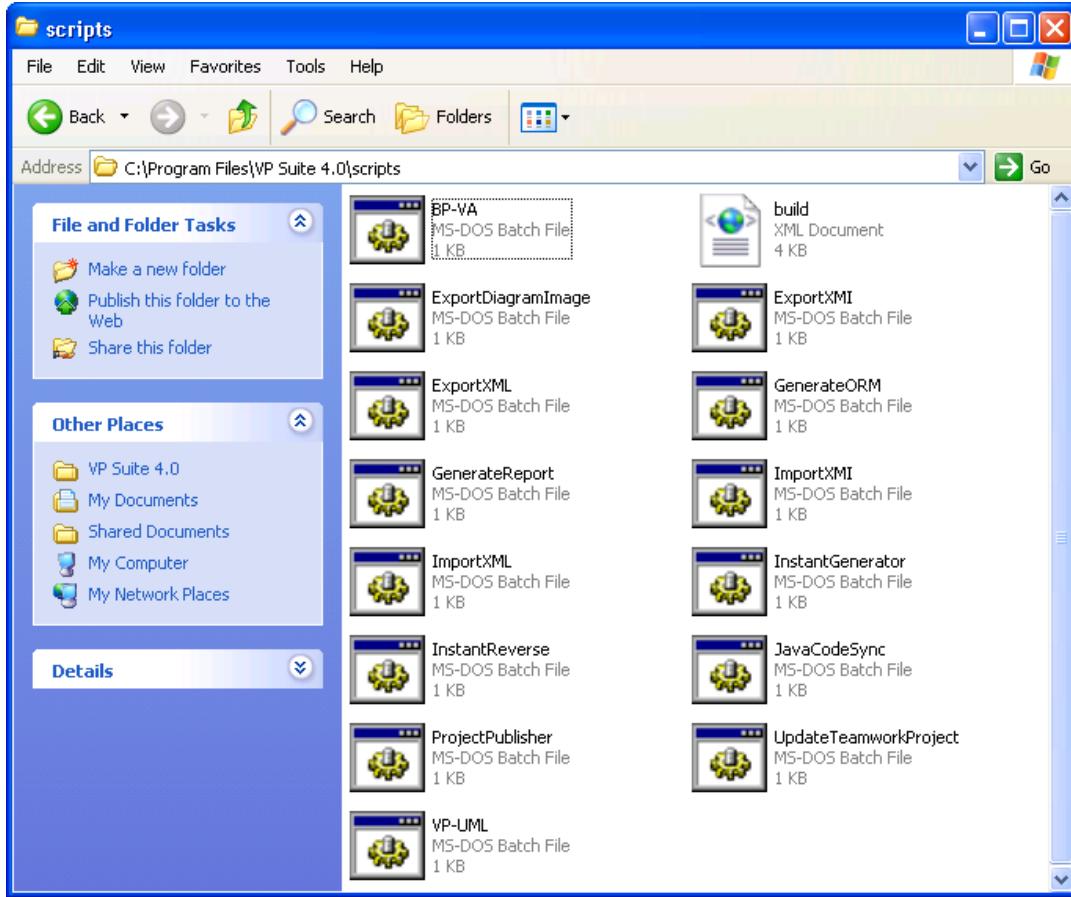
Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-out	The filepath of XMI file	C: \Demo \Output \sample.xmi
-type [optional]	Version of XMI. Unless specified, the lastly generated version will be selected. Here are the possible options: <ul style="list-style-type: none"> • 1.0 • 1.2 • 2.1 • 2.1UML2 	2.1
-encoding [optional]	Encoding of XMI file	

Parameters for ExportXMI

Importing XMI

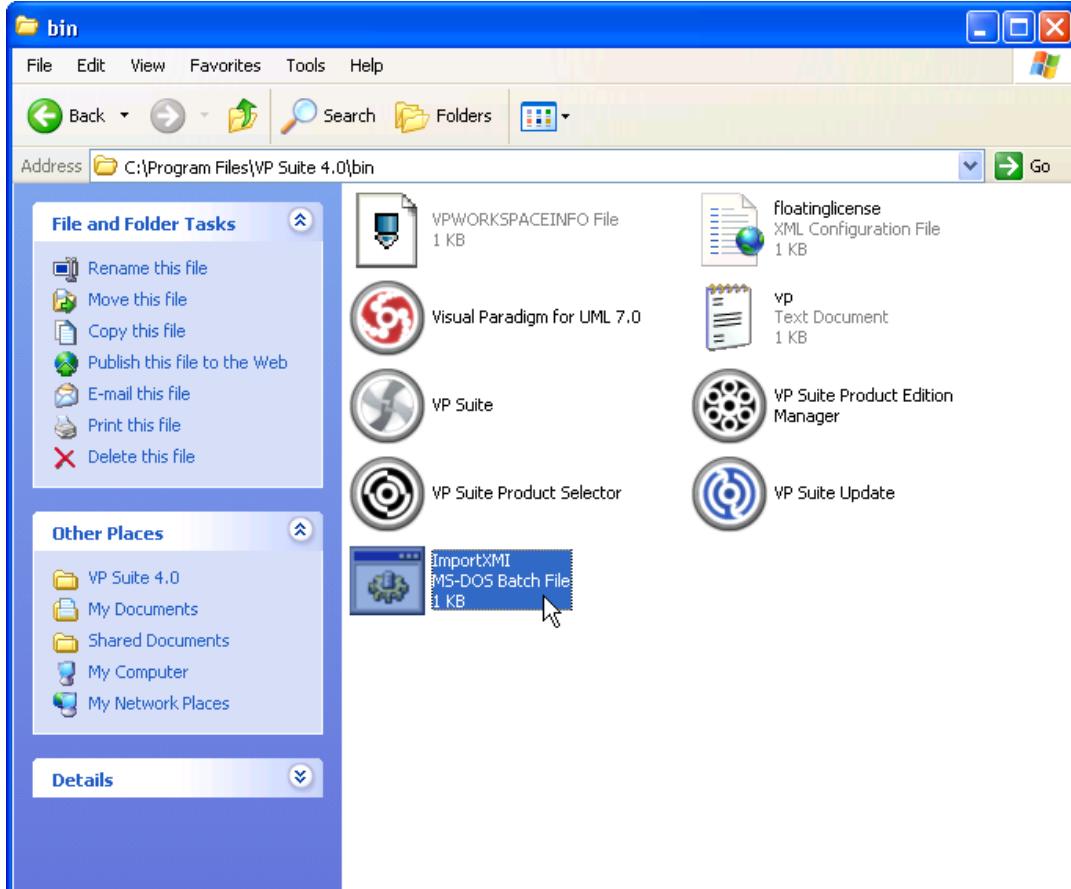
To import XMI to a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ImportXMI** and paste to the bin folder of VP Suite installation directory.



Copy and paste ImportXMI from scripts folder to bin folder

3. Start the command prompt.
4. Navigate to the bin folder of VP Suite installation directory.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>ImportXMI -project C:\Demo\Demo.vpp -file C:
\Demo\input\sample.xmi
unknown setting on VPPreference: v.fitg@5bf624.DefaultHTMLDocFont
C:\Program Files\VP Suite 4.0\bin>_
```

Executing ImportXMI

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

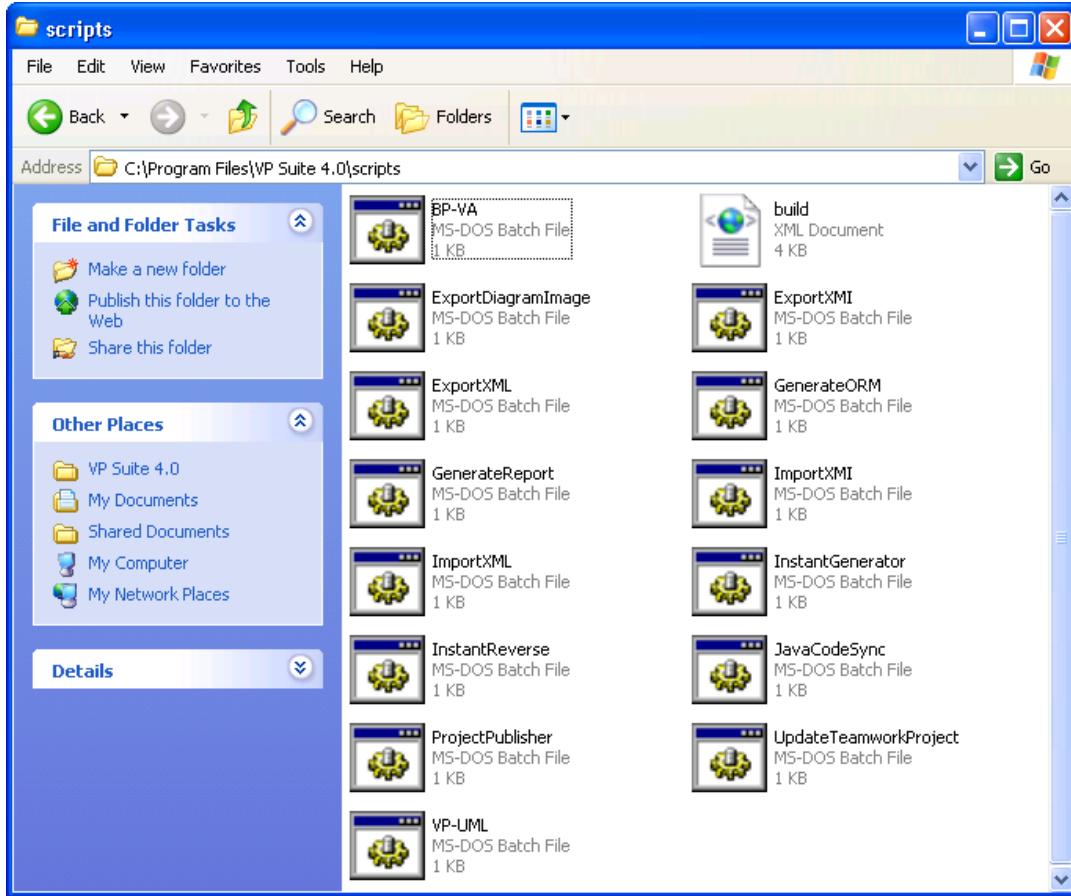
Exporting and import XML

You can export project data to an XML, manipulate it externally, and feed the changes back to VP-UML. In this chapter, you will see how to export XML file of whole project or specific diagram in project. In this chapter, you will see how to export and import XML through the command line interface.

Exporting XML

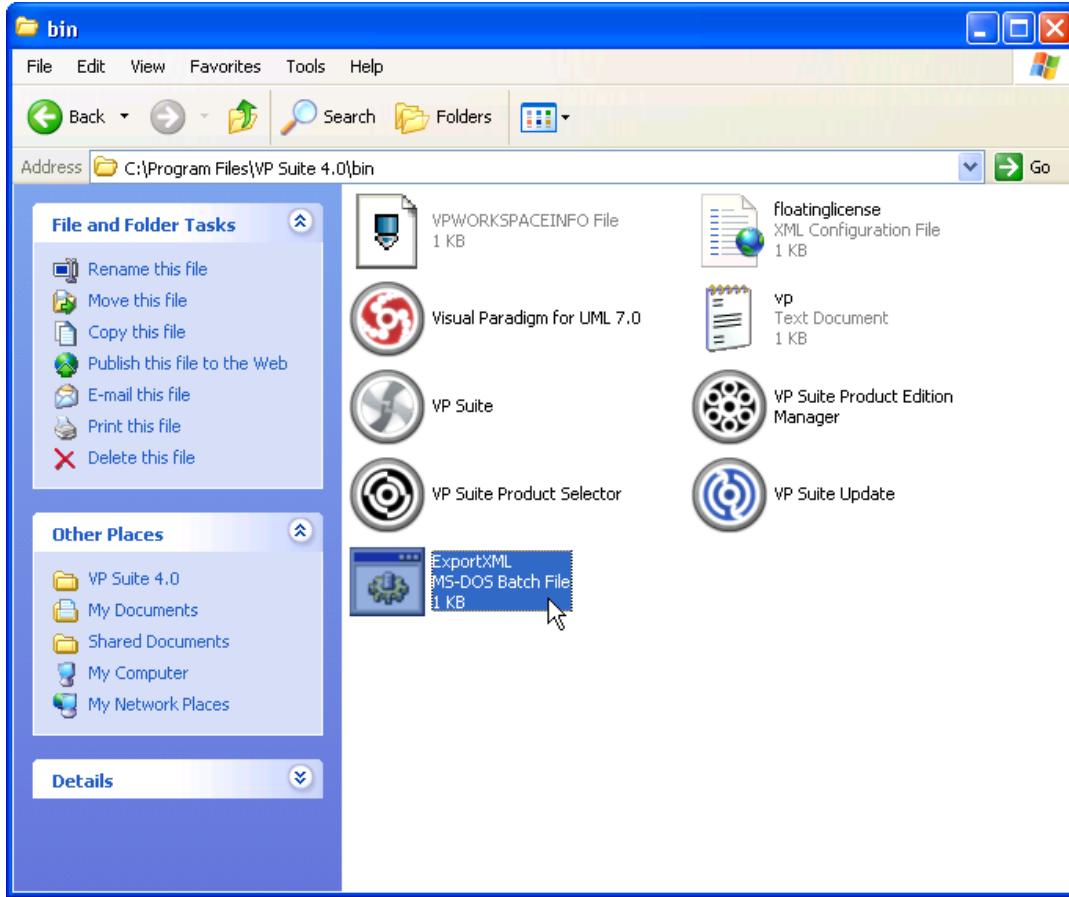
To export XML and images from a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ExportXML** and paste to the bin folder of VP Suite installation directory.



Copy and paste ExportXML from scripts folder to bin folder

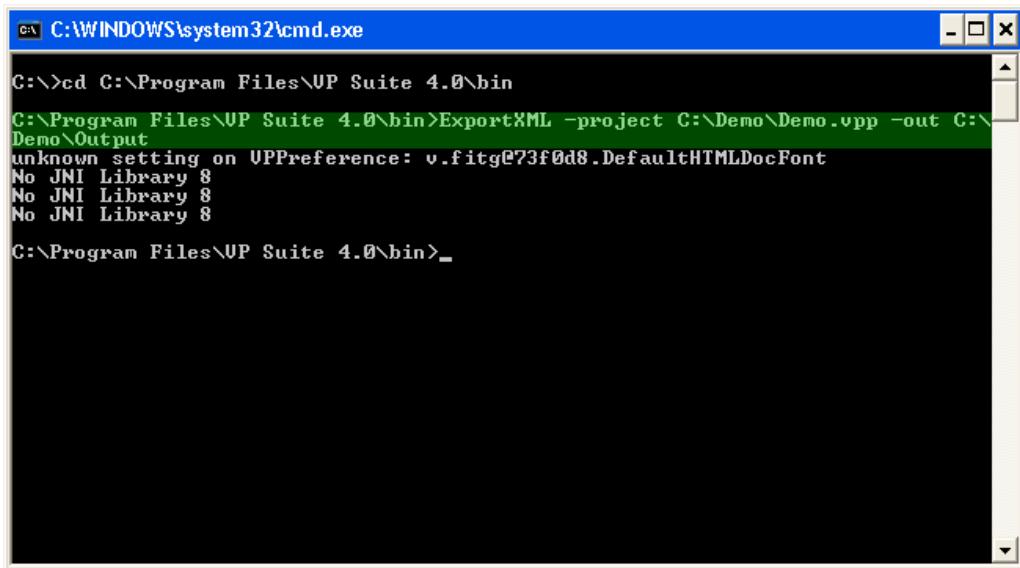
3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

```
C:\WINDOWS\system32\cmd.exe
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
ExportXML -project C:\Demo\Demo.vpp -out C:\Demo\Output



```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>ExportXML -project C:\Demo\Demo.vpp -out C:\Demo\Output
unknown setting on UPPreference: v.fitg@73f0d8.DefaultHTMLDocFont
No JNI Library 8
No JNI Library 8
No JNI Library 8
C:\Program Files\UP Suite 4.0\bin>_
```

Executing ExportXML

Below is a description of parameters:

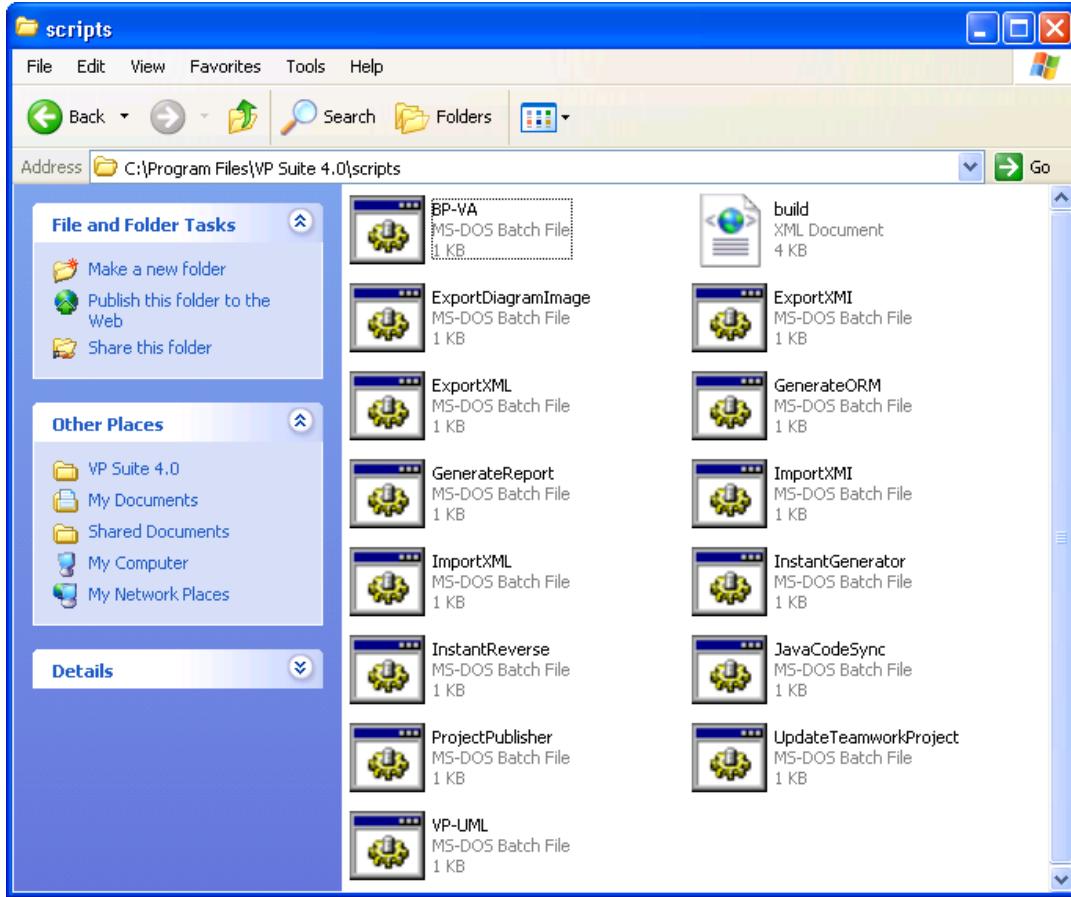
Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-out	Folder for storing the exported XML and images	C:\Demo\Output
-diagram	One or more diagrams to be exported	"Diagram A" "Diagram B"
-noimage	Do not export image files for diagrams	N/A

Parameters for ExportXML

Importing XML

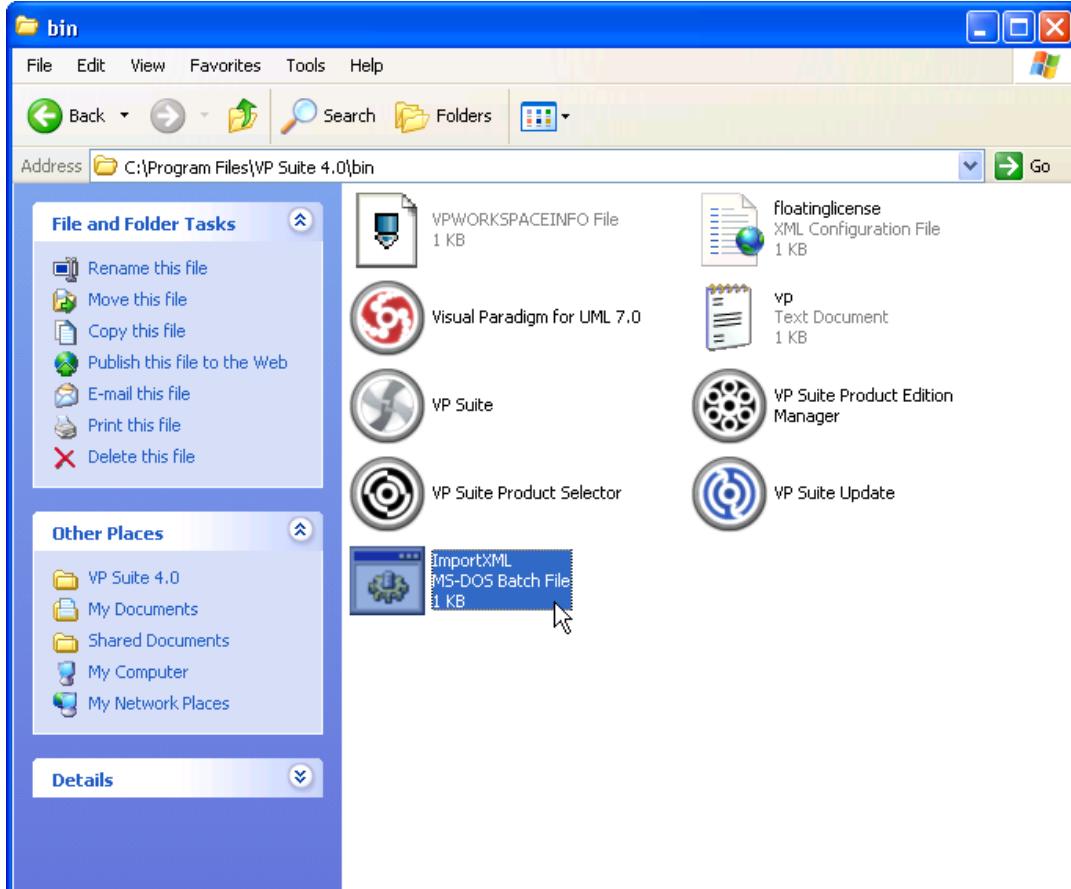
To import XML to a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ImportXML** and paste to the bin folder of VP Suite installation directory.



Copy and paste ImportXML from scripts folder to bin folder

3. Start the command prompt.
4. Navigate to the bin folder of VP Suite installation directory.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>ImportXML -project C:\Demo\Demo.vpp -file C:
\Demo\input\project.xml
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
>>> apply: LogicalView
C:\Program Files\VP Suite 4.0\bin>_
```

Executing ImportXML

Below is a description of parameters:

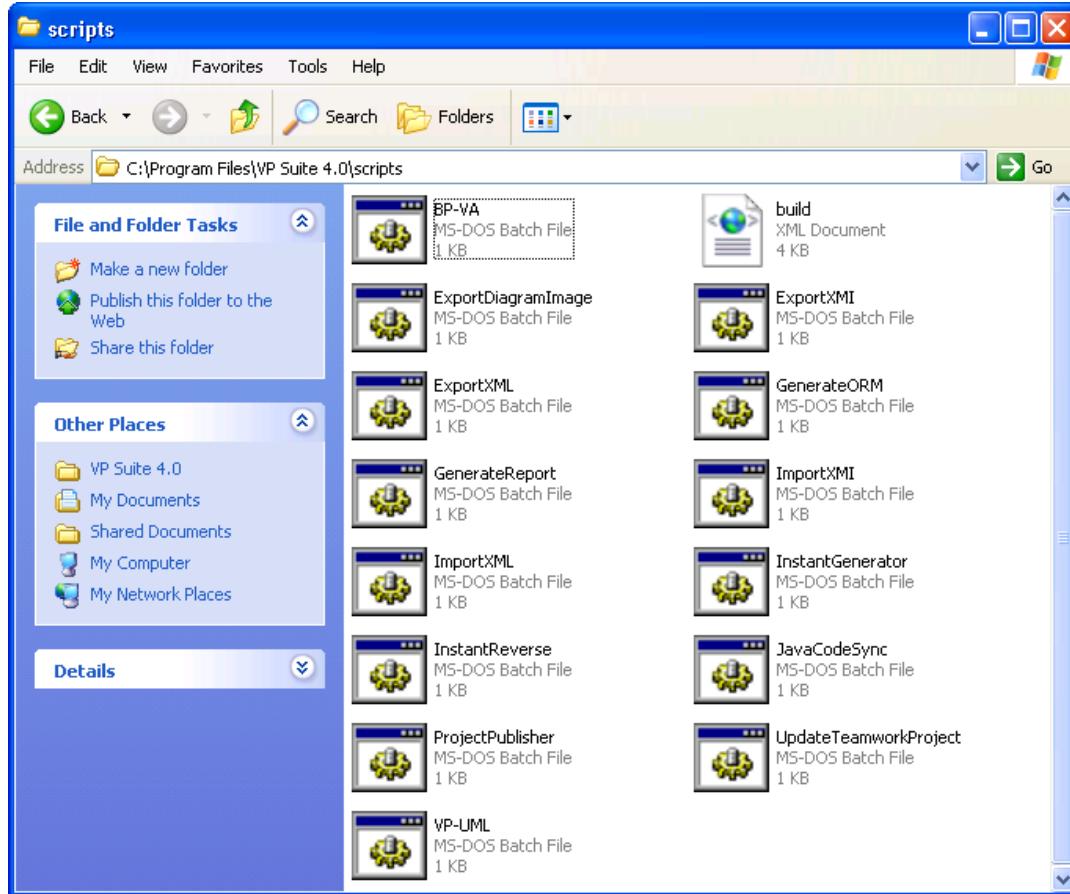
Parameter	Description	Example
-project	Project path	C:\Demo\Demo.vpp
-file	The filepath of the XML file to import	C:\Demo\input\sample.xml

Parameters for ImportXML

Generating ORM code and/or database

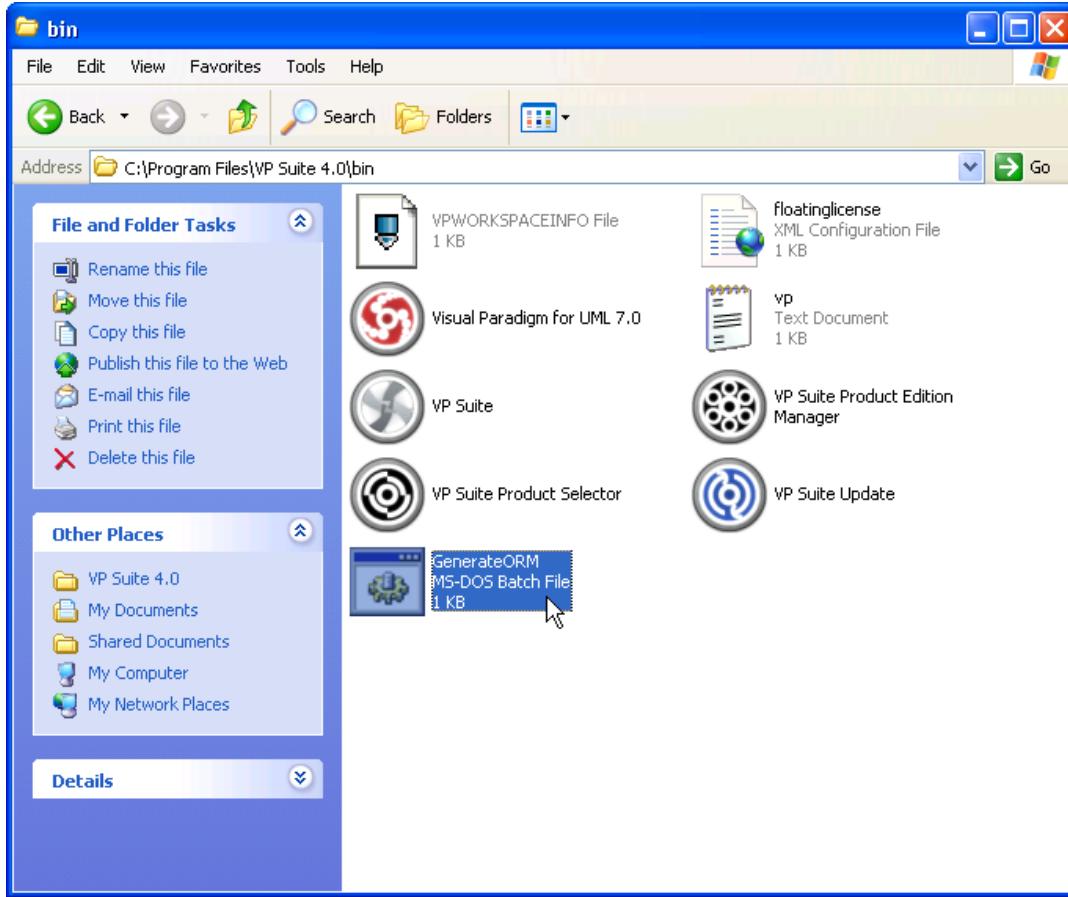
Generation of ORM code and database can be done through the command line interface. To do this:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **GenerateORM** and paste to the bin folder of VP Suite installation directory.



Copy and paste GenerateORM from scripts folder to bin folder

3. Start the command prompt.

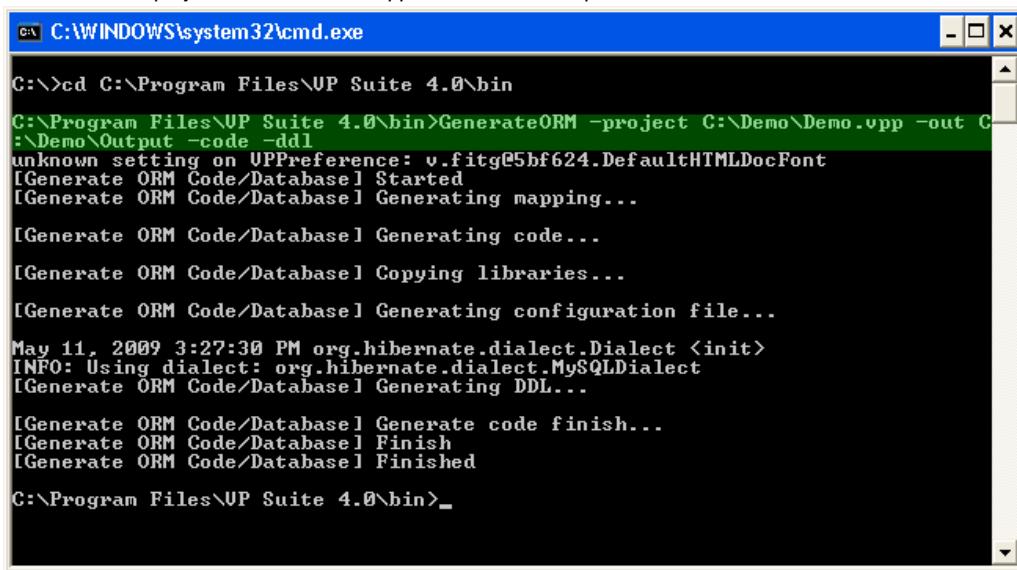
4. Navigate to the bin folder of VP Suite installation directory.

```
C:\WINDOWS\system32\cmd.exe
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

The screenshot shows a Windows Command Prompt window with the title bar 'cmd.exe' and the path 'C:\WINDOWS\system32\cmd.exe'. The command line shows the user navigating to the 'bin' directory of the 'VP Suite 4.0' installation: 'C:\>cd C:\Program Files\VP Suite 4.0\bin'. The command prompt then displays the directory path 'C:\Program Files\VP Suite 4.0\bin>' followed by a cursor.

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
GenerateORM -project C:\Demo\Demo.vpp -out C:\Demo\Output -code -ddl



```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>GenerateORM -project C:\Demo\Demo.vpp -out C:\Demo\Output -code -ddl
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
[Generate ORM Code/Database] Started
[Generate ORM Code/Database] Generating mapping...
[Generate ORM Code/Database] Generating code...
[Generate ORM Code/Database] Copying libraries...
[Generate ORM Code/Database] Generating configuration file...
May 11, 2009 3:27:30 PM org.hibernate.dialect.Dialect <init>
INFO: Using dialect: org.hibernate.dialect.MySQLDialect
[Generate ORM Code/Database] Generating DDL...
[Generate ORM Code/Database] Generate code finish...
[Generate ORM Code/Database] Finish
[Generate ORM Code/Database] Finished
C:\Program Files\UP Suite 4.0\bin>_
```

Executing GenerateORM

Below is a description of parameters:

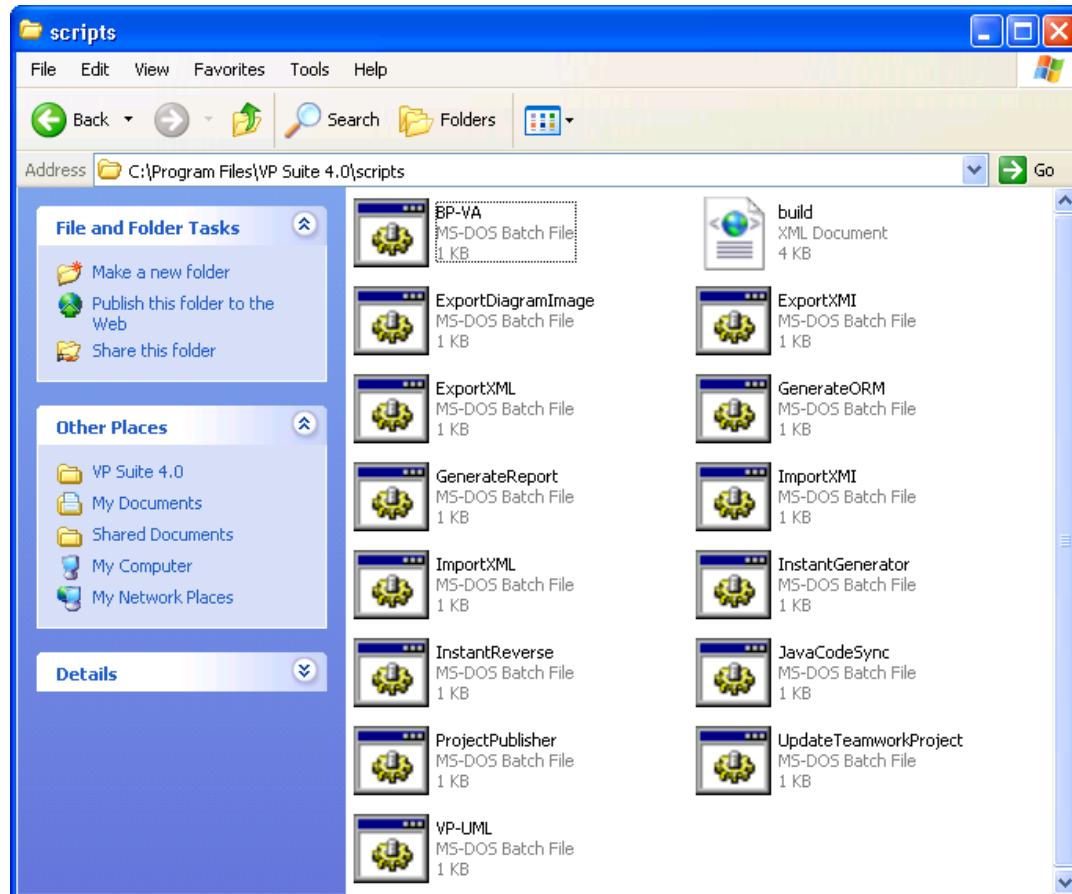
Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-out	Folder for storing the generated files including the source code, required libraries and mapping XML	C: \Demo \Output
-code [optional]	Include to generate code.	- code
-ddl [optional]	Include to export DDL	- ddl
-exportdb [optional]	Include to export database	- exportdb

Parameters for GenerateORM

Generating report

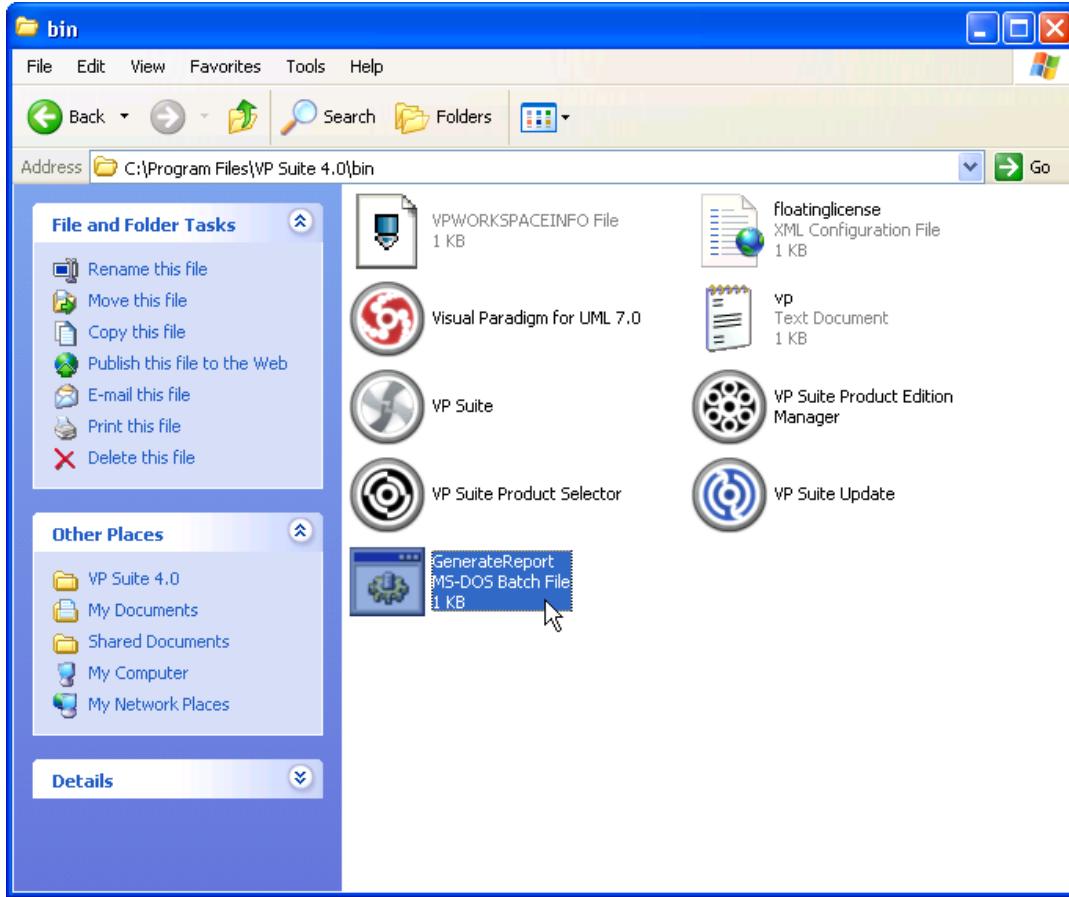
To generate HTML/PDF/Word report from a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **GenerateReport** and paste to the bin folder of VP Suite installation directory.



Copy and paste GenerateReport from scripts folder to bin folder

3. Start the command prompt.

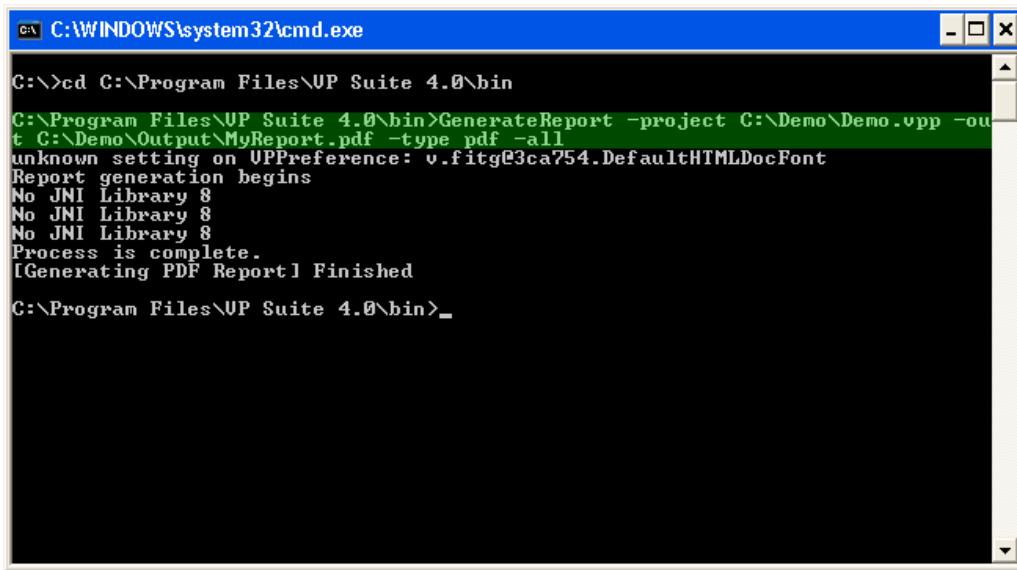
4. Navigate to the bin folder of VP Suite installation directory.

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window shows the command line: 'C:\>cd C:\Program Files\VP Suite 4.0\bin' followed by a blank line. The window has standard Windows-style scroll bars on the right side.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
GenerateReport -project C:\Demo\Demo.vpp -out C:\Demo\Output\MyReport.pdf -type pdf -all



```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>GenerateReport -project C:\Demo\Demo.vpp -out C:\Demo\Output\MyReport.pdf -type pdf -all
unknown setting on UPPreference: v.fitg@3ca754.DefaultHTMLDocFont
Report generation begins
No JNI Library 8
No JNI Library 8
No JNI Library 8
Process is complete.
[Generating PDF Report] Finished
C:\Program Files\UP Suite 4.0\bin>
```

Executing GenerateReport

Below is a description of parameters:

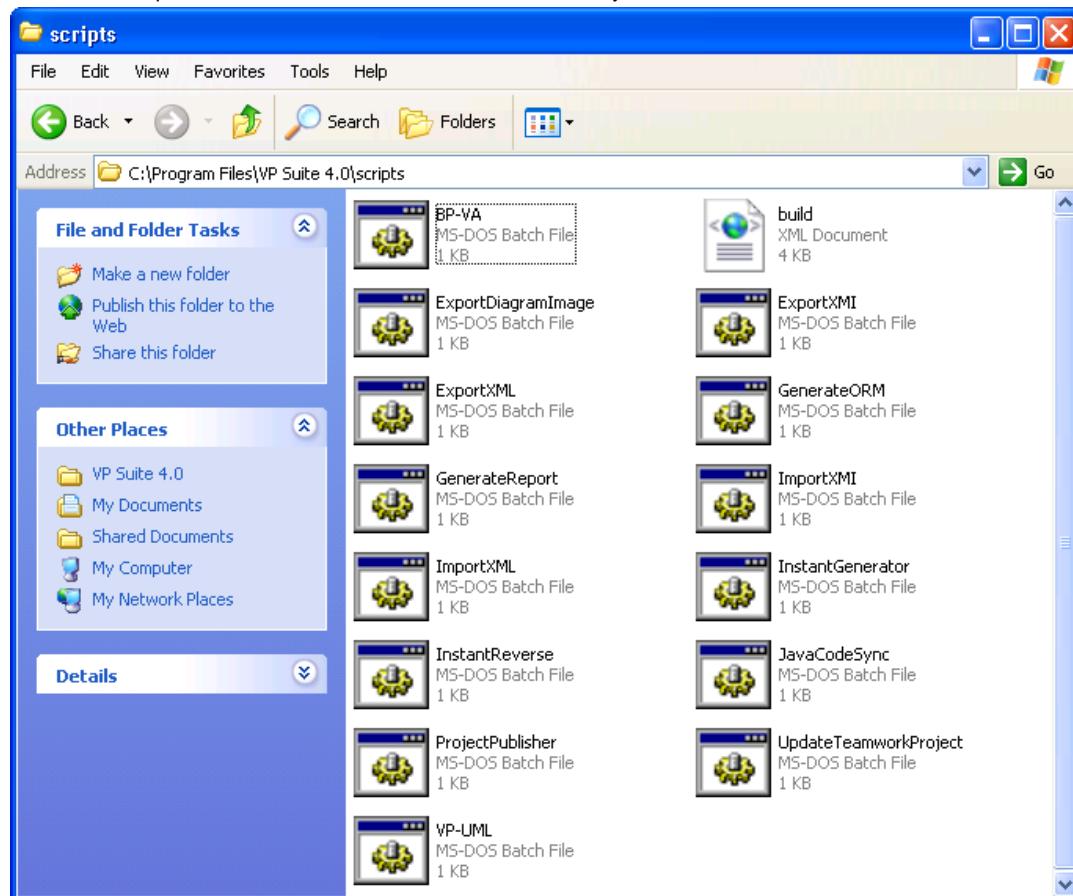
Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-out	The file or folder path of generated report file(s)	C: \Demo \Output \MyReport.p
-type	Type of report to generate. Here are the possible options: <ul style="list-style-type: none"> • html • pdf • word 	pdf
-all [optional]	By default, only the selected diagrams (saved in vpp) will be covered when generating report. By including -all, all diagrams will be covered.	- all

Parameters for GenerateReport

Instant generator

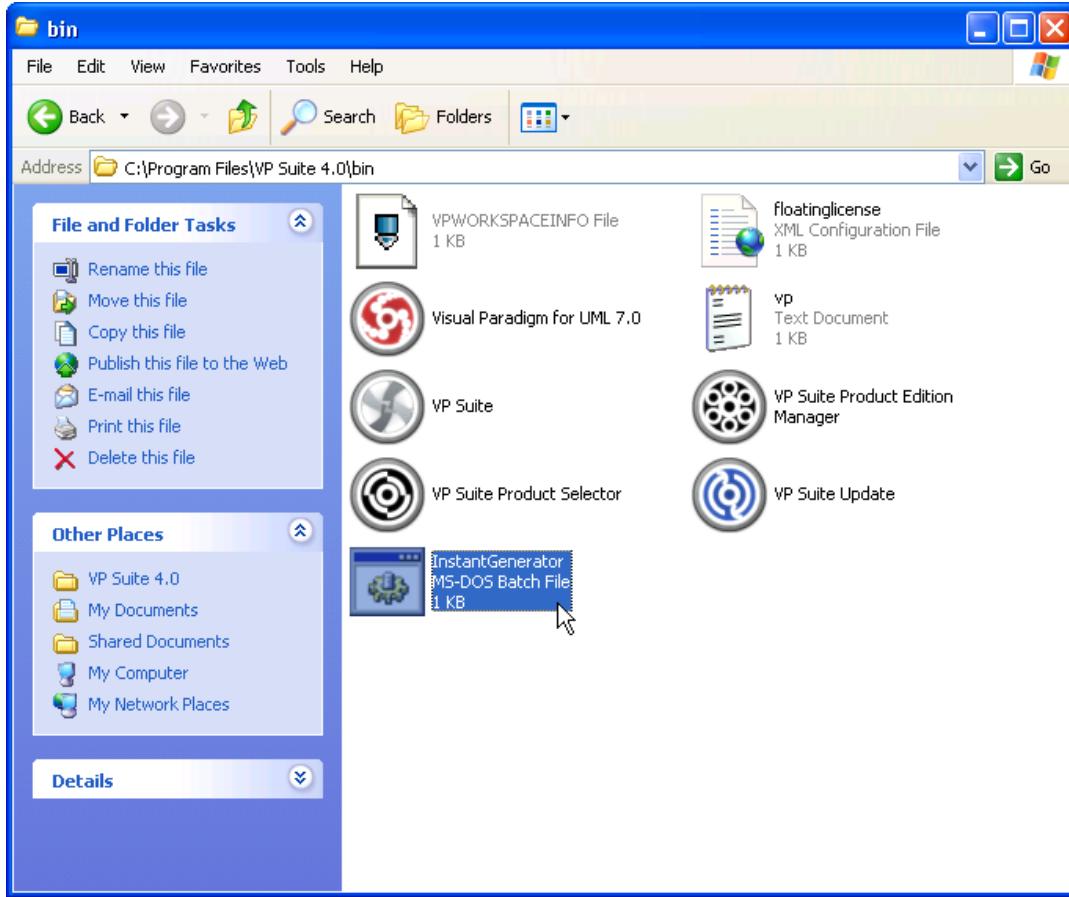
To generate code from a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **InstantGenerator** and paste to the bin folder of VP Suite installation directory.



Copy and paste InstantGenerator from scripts folder to bin folder

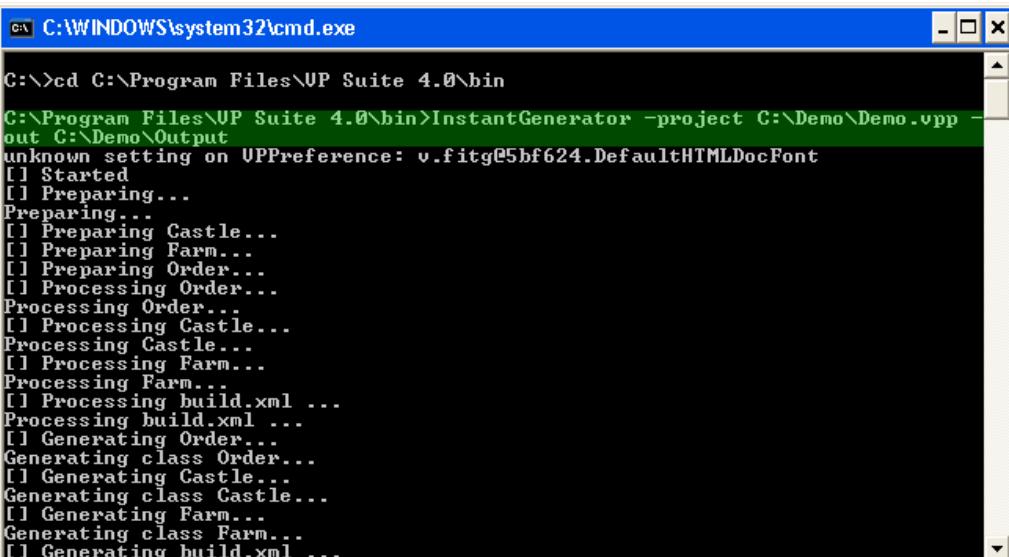
3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window shows the command line: 'C:\>cd C:\Program Files\VP Suite 4.0\bin' followed by a blank line. The title bar also displays the path 'C:\Program Files\VP Suite 4.0\bin>'.

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
InstantGenerator -project C:\Demo\Demo.vpp -out C:\Demo\Output



```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>InstantGenerator -project C:\Demo\Demo.vpp -
out C:\Demo\Output
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
[] Started
[] Preparing...
Preparing...
[] Preparing Castle...
[] Preparing Farm...
[] Preparing Order...
[] Processing Order...
Processing Order...
[] Processing Castle...
Processing Castle...
[] Processing Farm...
Processing Farm...
[] Processing build.xml ...
Processing build.xml ...
[] Generating Order...
Generating class Order...
[] Generating Castle...
Generating class Castle...
[] Generating Farm...
Generating class Farm...
[] Generating build.xml ...
```

Executing InstantGenerator

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:\MyTemplate
-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: <ul style="list-style-type: none"> • Java • C# • .NET • ODL • ActionScript • IDL • C++ • Delphi • Python • Objective-C • Ada95 • Ruby 	C + +

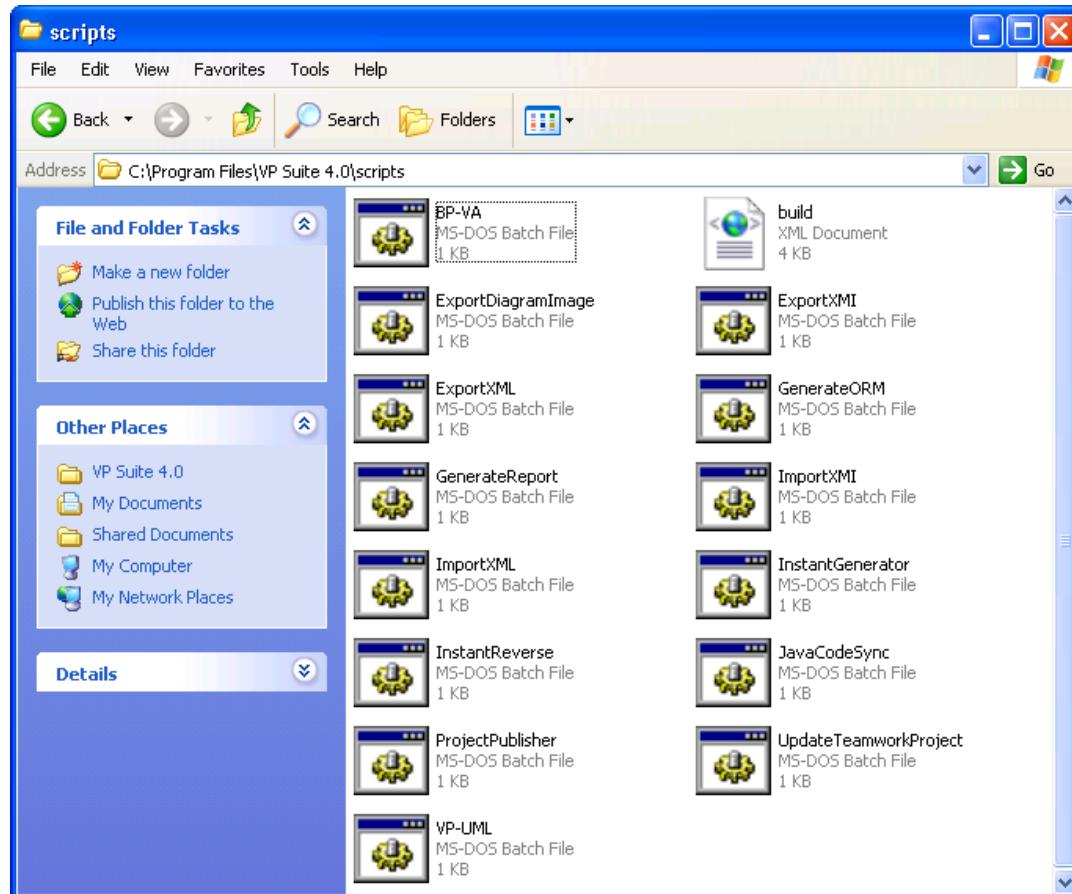
Parameters for InstantGenerator

NOTE: Code generation through command line generates only classes selected to generate when running VP-UML. In other words, you must at least generate once in VP-UML in order to make command line generation work.

Instant reverse

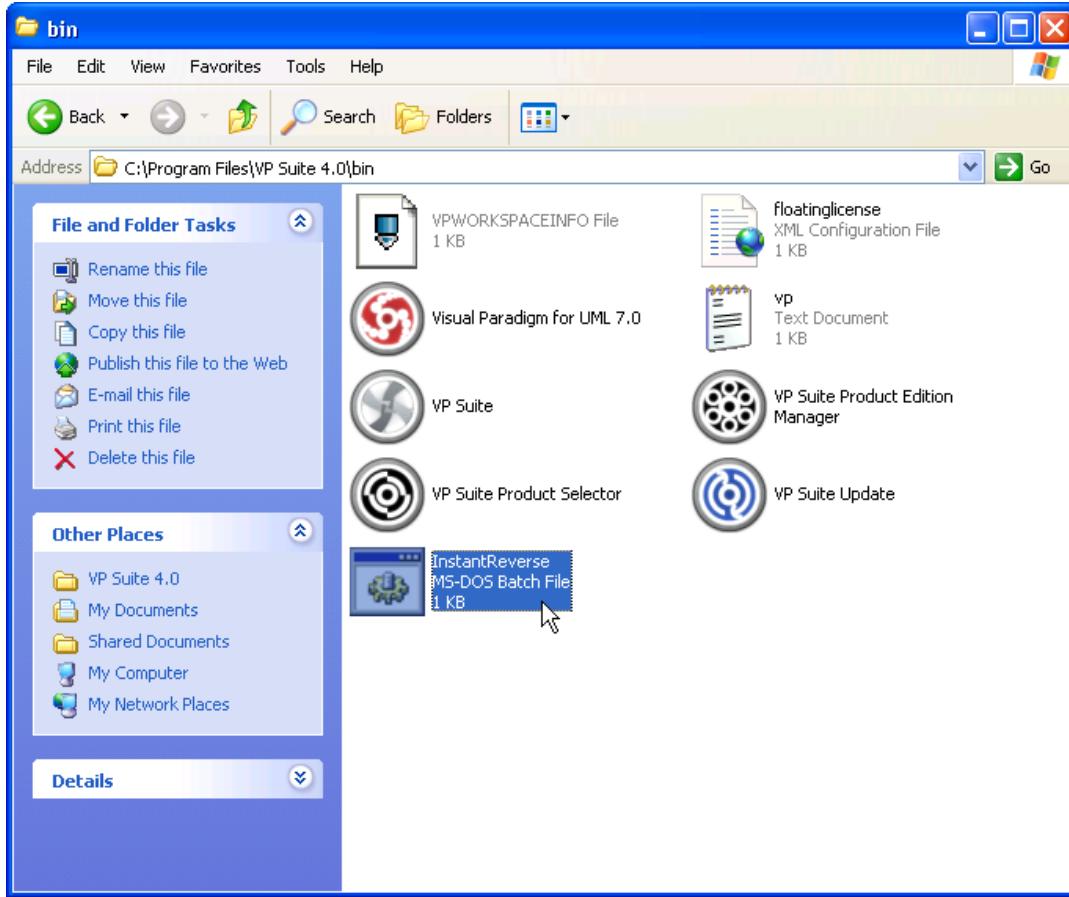
To reverse source code to a project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **InstantReverse** and paste to the bin folder of VP Suite installation directory.



Copy and paste *InstantReverse* from scripts folder to bin folder

3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window shows the command 'cd C:\Program Files\VP Suite 4.0\bin' being entered. The command prompt is located at the bottom of the window.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin  
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

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`

```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>InstantReverse -project C:\Demo\Demo.vpp -path C:\Demo\MyProject\src -lang Java -pathtype folder -sourcetype source
unknown setting on UPPreference: v.fitg@31fb31.DefaultHTMLDocFont
[Add Java Reverse] Reversing "src", please wait...
[Add Java Reverse] Finished
```

Executing InstantReverse

Below is a description of parameters:

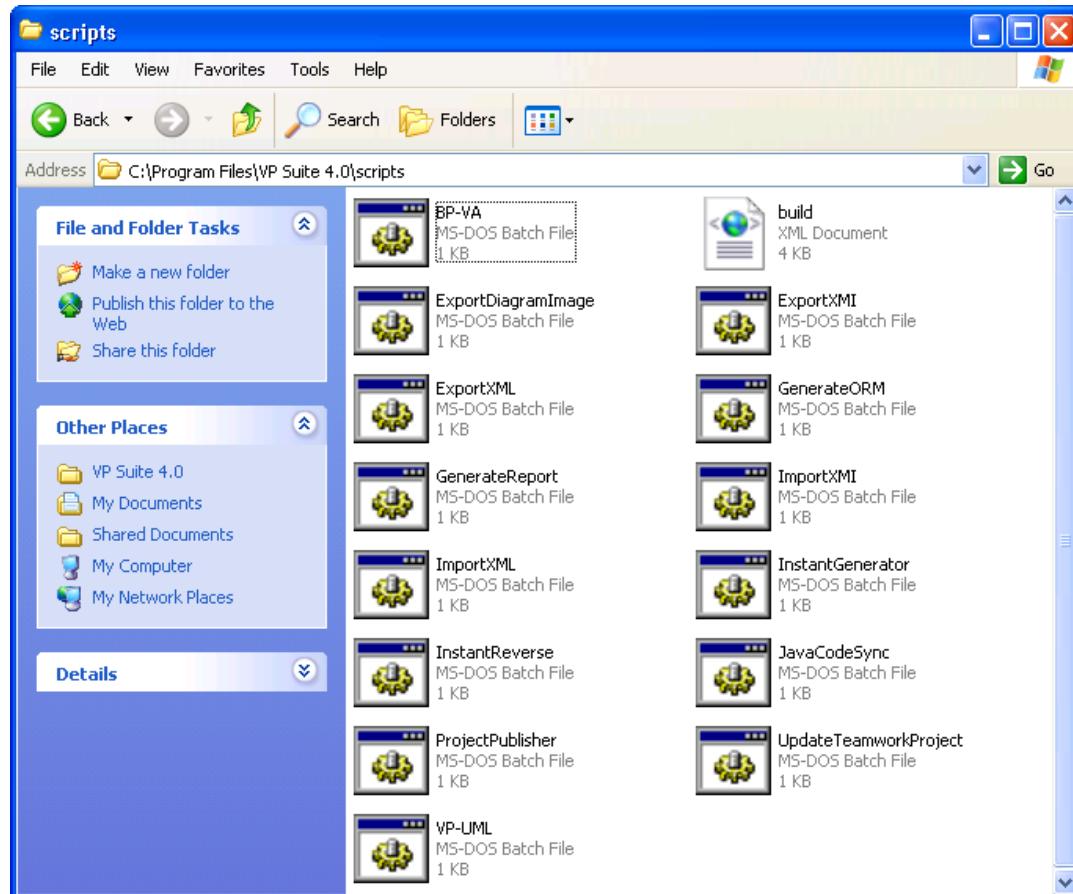
Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-path	The file or folder path of the source files to be reversed	C: \Demo \Output \src
-lang	Specify the language of the source code to reverse. Here are the possible options: <ul style="list-style-type: none"> • Java • "C++ Source" • ".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: <ul style="list-style-type: none"> • file • folder • zip 	file
sourcetype	Useful only for Java, sourcetype defines the type of source to reverse. Here are the possible options: <ul style="list-style-type: none"> • source • class 	source
-overwrite -update	Overwrite or update model from code	- overwrite

Parameters for InstantReverse

Java code synchronization

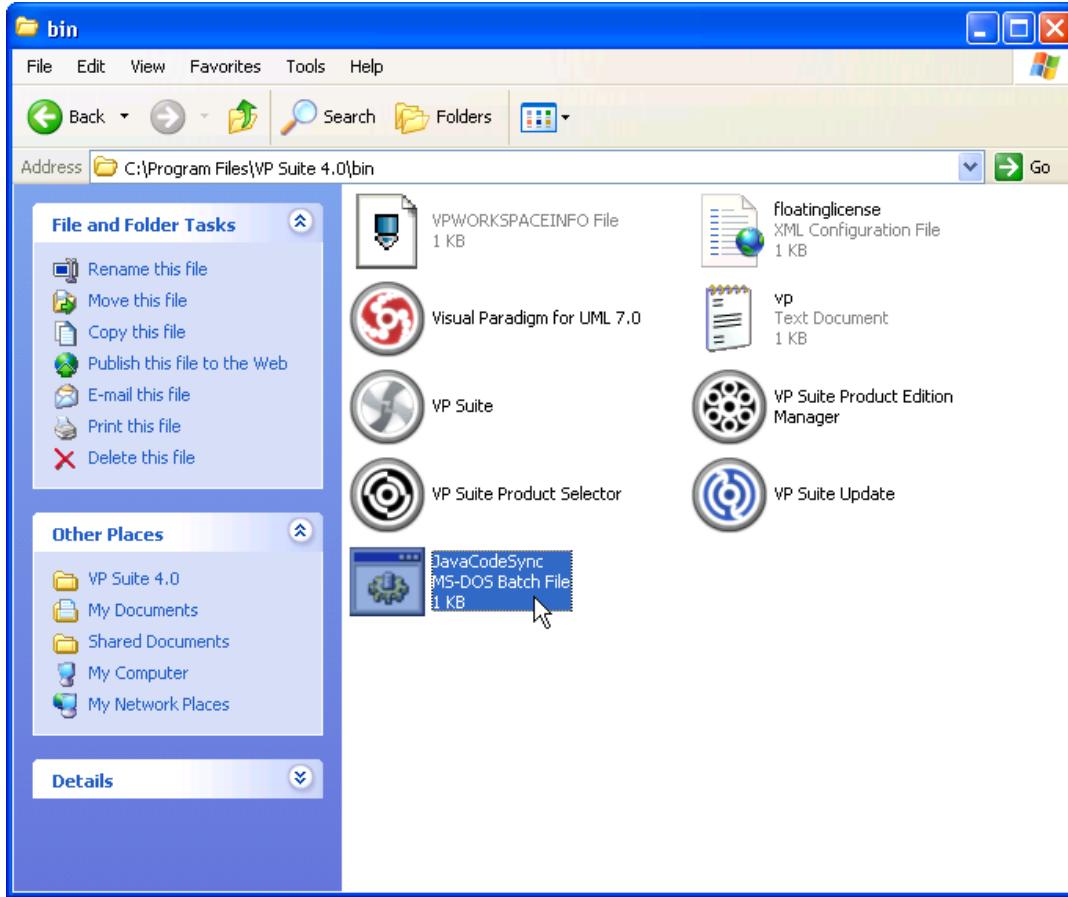
To perform synchronization between model and Java code through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

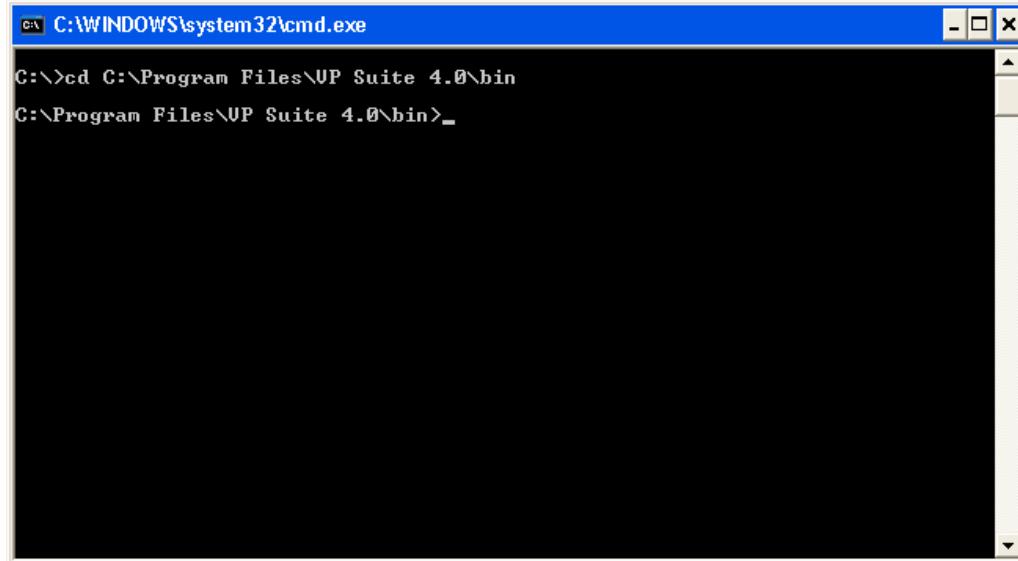
2. Copy the script file **JavaCodeSync** and paste to the bin folder of VP Suite installation directory.



Copy and paste JavaCodeSync from scripts folder to bin folder

3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.



Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
JavaCodeSync -project C:\Demo\Demo.vpp -src C:\Demo\MyProject\src -generate

```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>JavaCodeSync -project C:\Demo\Demo.vpp -src C:\Demo\MyProject\src -generate
unknown setting on UPPreference: v.fitg@5bf624.DefaultHTMLDocFont
[Java Code Generation Engine] Started
[Java Code Generation Engine] Validating Models...
[Java Code Generation Engine] Preparing Working Environment...
[Java Code Generation Engine] Updating: Order
[Java Code Generation Engine] Updating: Castle
[Java Code Generation Engine] Updating: Farm
[Java Code Generation Engine] Synchronize to code...
[Java Code Generation Engine] Finalize Models in Code Model Synchronization...
[Java Code Generation Engine] Finished
C:\Program Files\UP Suite 4.0\bin>
```

Executing JavaCodeSync

Below is a description of parameters:

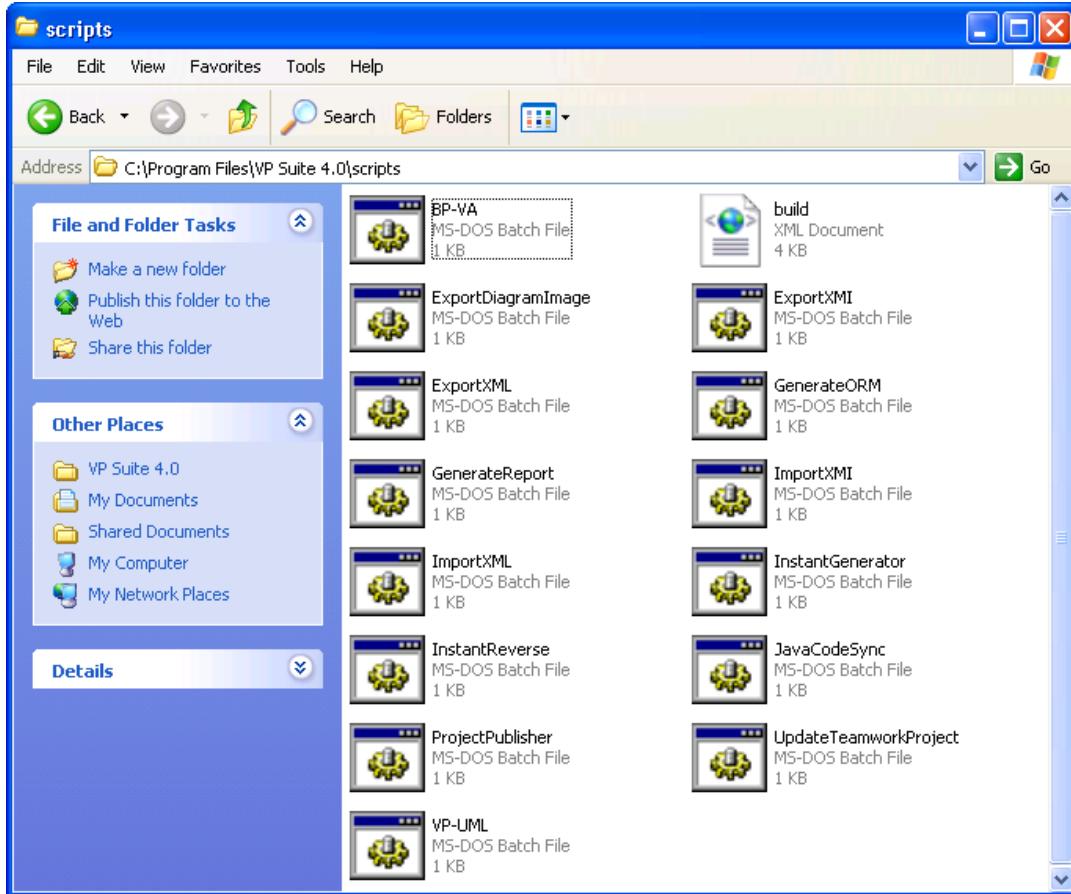
Parameter	Description	Example
-project	Project path	C: \Demo \Demo.vpp
-src	The folder path of the source file	C: \Demo \Output \src
-generate -reverse	Action to perform. Include -generate to indicate the update of code from model. Include -reverse to indicate the update of model from code.	- generate

Parameters for JavaCodeSync

Project publisher

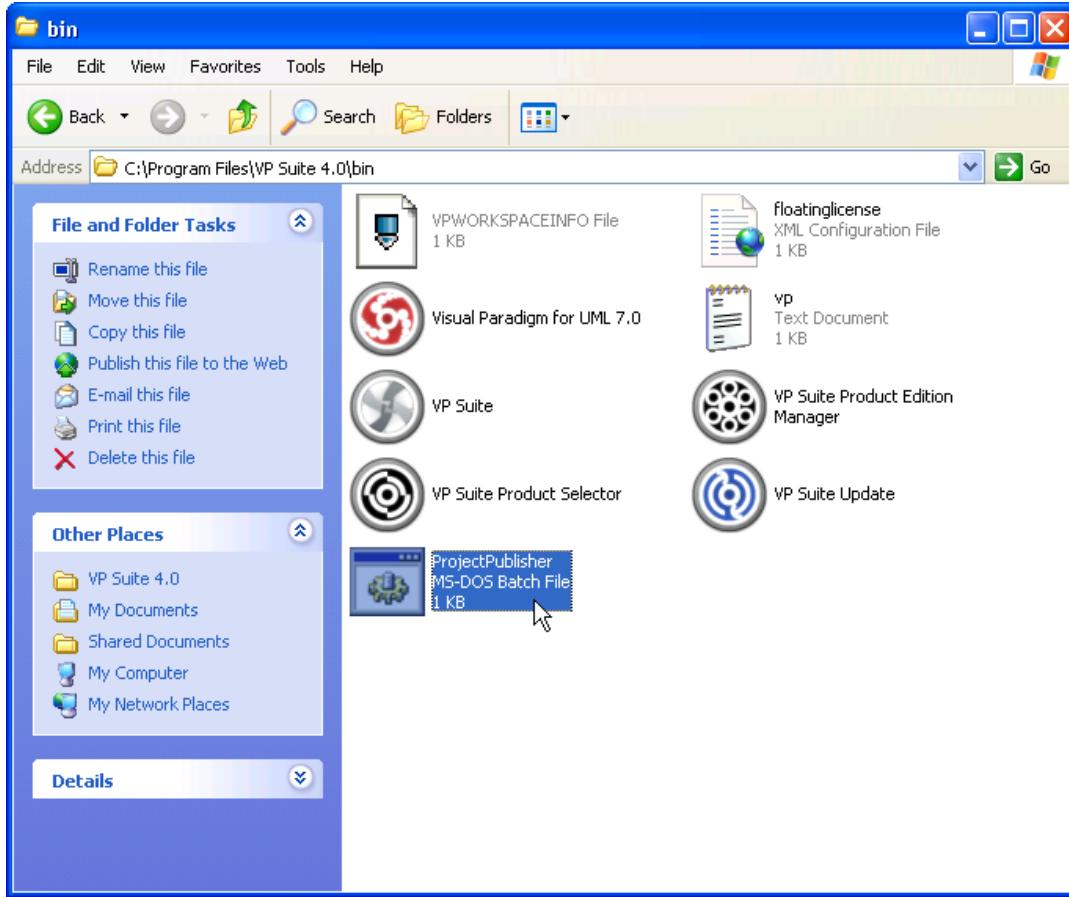
To publish project through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ProjectPublisher** and paste to the bin folder of VP Suite installation directory.



Copy and paste ProjectPublisher from scripts folder to bin folder

3. Start the command prompt.

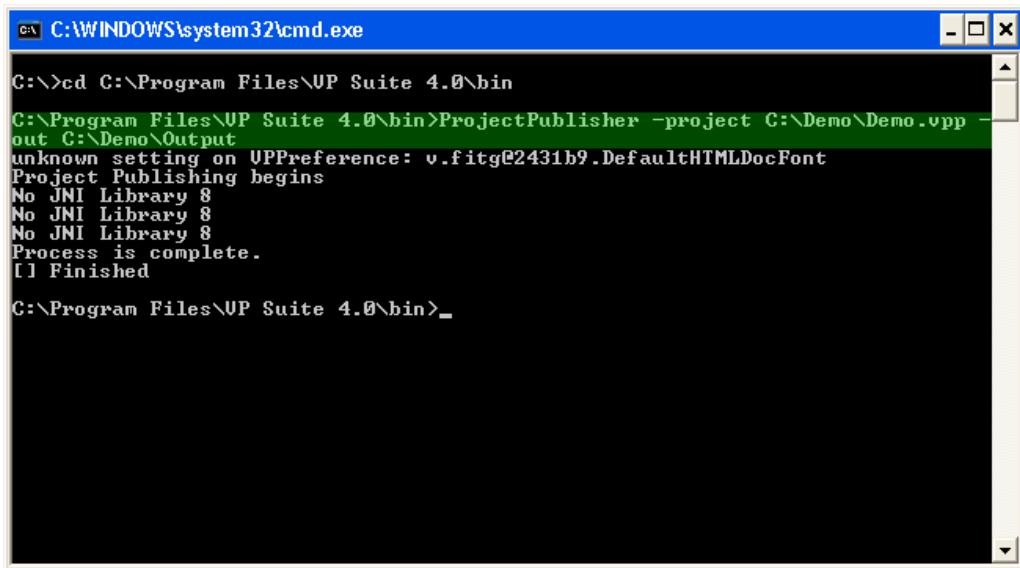
4. Navigate to the bin folder of VP Suite installation directory.

A screenshot of a Windows Command Prompt window titled 'cmd C:\WINDOWS\system32\cmd.exe'. The window shows the command 'cd C:\Program Files\VP Suite 4.0\bin' being entered and executed. The prompt then changes to 'C:\Program Files\VP Suite 4.0\bin>'. The rest of the window is a black blank space.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
`ProjectPublisher -project C:\Demo\Demo.vpp -out C:\Demo\Output`



A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The command entered is "ProjectPublisher -project C:\Demo\Demo.vpp -out C:\Demo\Output". The output shows the following log:

```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>ProjectPublisher -project C:\Demo\Demo.vpp -
out C:\Demo\Output
unknown setting on UPPreference: v.fitg@2431b9.DefaultHTMLDocFont
Project Publishing begins
No JNI Library 8
No JNI Library 8
No JNI Library 8
Process is complete.
[] Finished

C:\Program Files\UP Suite 4.0\bin>_
```

Executing ProjectPublisher

Below is a description of parameters:

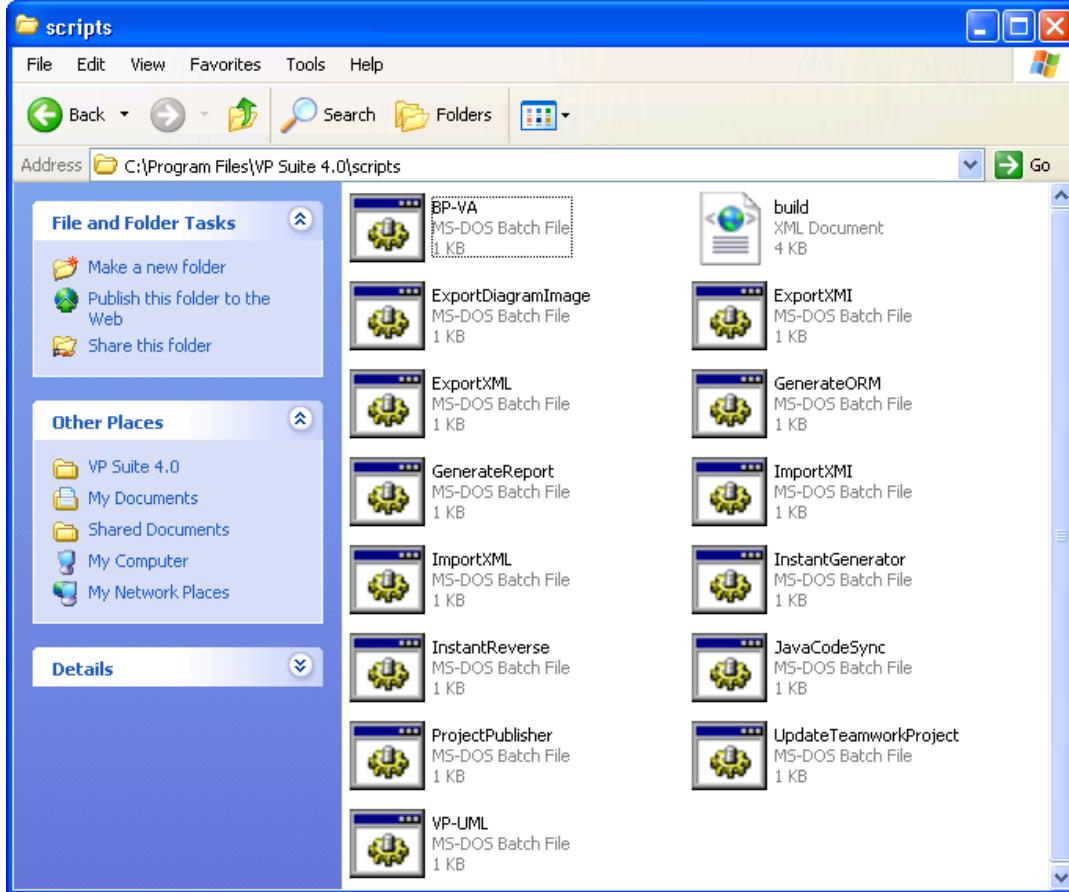
Parameter	Description	Example
-project	Project path	C:\Demo \\Demo.vpp
-out	The folder path of the files to be published	C:\Demo \\Output

Parameters for ProjectPublisher

Updating teamwork project from server

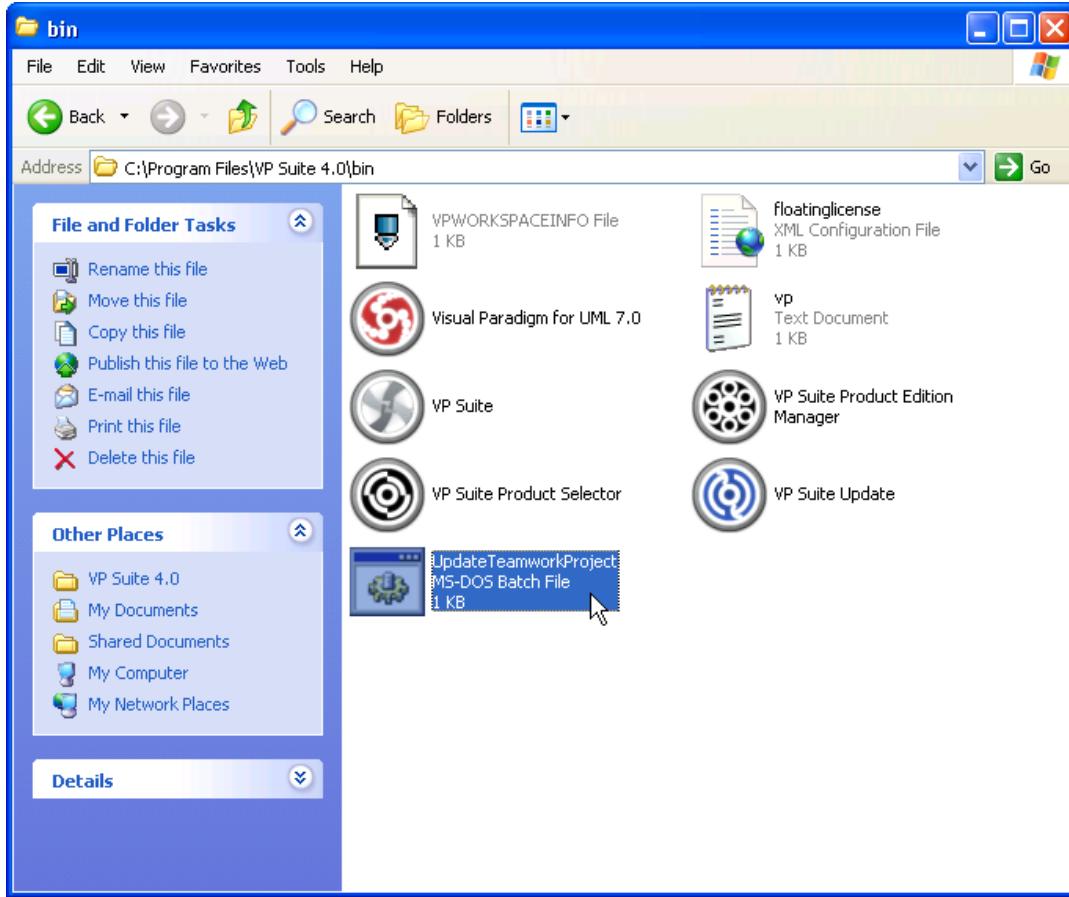
To update Teamwork project from server through command line:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **ProjectPublisher** and paste to the bin folder of VP Suite installation directory.



Copy and paste UpdateTeamworkProject from scripts folder to bin folder

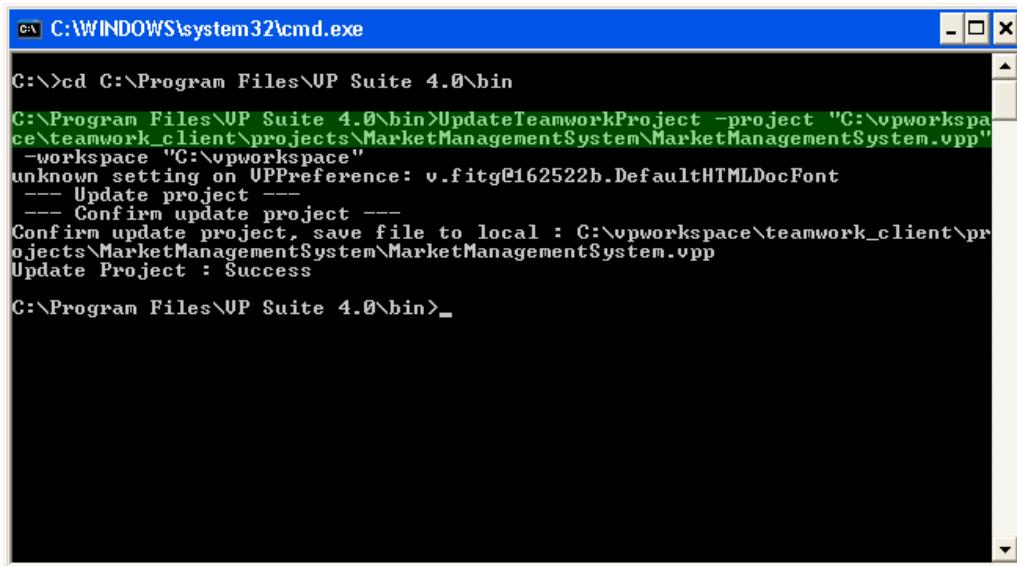
3. Start the command prompt.

4. Navigate to the bin folder of VP Suite installation directory.

```
C:\>cd C:\Program Files\VP Suite 4.0\bin
C:\Program Files\VP Suite 4.0\bin>_
```

Navigate to the bin folder of VP Suite in command prompt

5. Execute the script by supplying the required parameters. For example:
`UpdateTeamworkProject -project "C:\vpworkspace\teamwork_client\projects\MarketManagementSystem\MarketManagementSystem.vpp" -workspace "C:\vpworkspace"`



```
C:\>ed C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>UpdateTeamworkProject -project "C:\vpworkspace\teamwork_client\projects\MarketManagementSystem\MarketManagementSystem.vpp"
unknown setting on VPPreference: v.fitg@162522b.DefaultHTMLDocFont
--- Update project ---
--- Confirm update project ---
Confirm update project, save file to local : C:\vpworkspace\teamwork_client\projects\MarketManagementSystem\MarketManagementSystem.vpp
Update Project : Success
C:\Program Files\UP Suite 4.0\bin>_
```

Executing UpdateTeamworkProject

Below is a description of parameters:

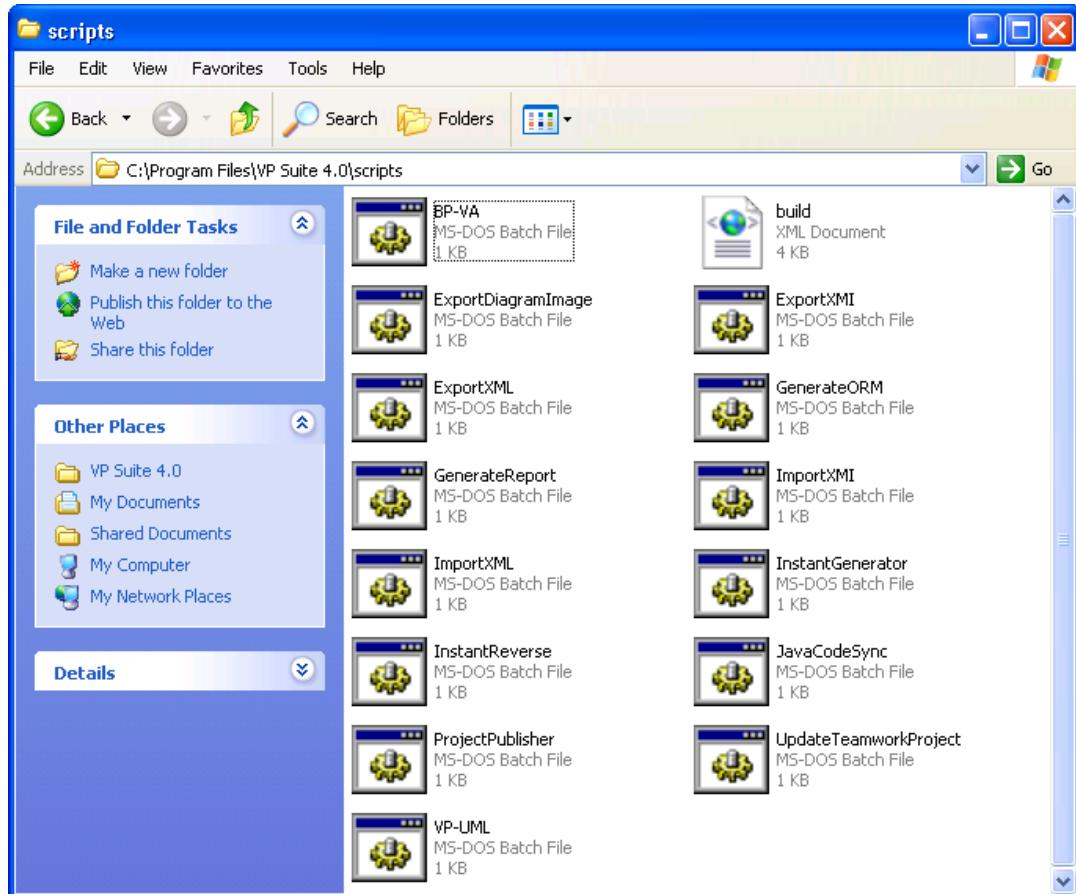
Parameter	Description	Example
-project	Project path	C:\Demo \Demo.vpp
-workspace	The path of workspace of the supplied project	C: \vpworkspace

Parameters for UpdateTeamworkProject

Executing operations with Apache Ant

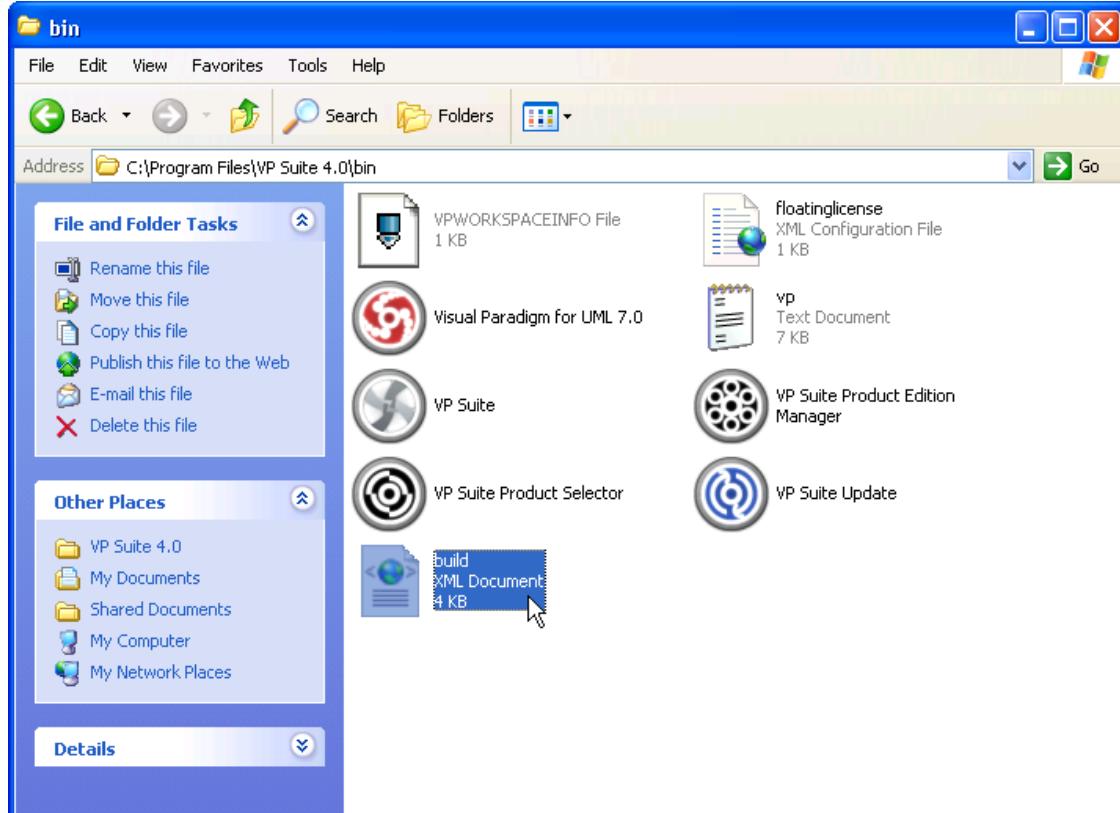
Apache Ant is a software tool for automating software build processes. It is written in the Java language and is primarily intended for use with Java. If you are not familiar with Ant, you can find more information about it at [Ant's webpage](#). To execute commands with Ant:

1. Browse the scripts folder under the VP Suite installation directory.



The scripts folder inside VP Suite installation directory

2. Copy the script file **build.xml** and paste to the bin folder of VP Suite installation directory.



Copy and paste build.xml from scripts folder to bin folder

3. Open the build file in any text editor. Modify the properties **vpsuiteInstallationPath**, **vpproduct**, **vpworkspace** and **headless** to suit your environment.

```

<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="default">
    <property name="vpsuiteInstallationPath" value="C:/Program Files/VP Suite 3.3" />
    <!-- VPPRODUCT: VP-UML, BP-VA, DB-VA, AG, SDE-EC, SDE-VS, SDE-IJ, SDE-NB, SDE-JB, S. -->
    <property name="vpproduct" value="VP-UML" />
    <property name="vpworkspace" value="${user.home}/vpworkspace" />
    <property name="headless" value="false" />

    <taskdef resource="com/vp/ant/taskdef.properties" classpath="${vpsuiteInstallationP:>

```

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.

```

<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="default">
    <property name="vpsuiteInstallationPath" value="C:/Program Files/VP Suite 3.3" />
    <!-- VPPRODUCT: VP-UML, BP-VA, DB-VA, AG, SDE-EC, SDE-VS, SDE-IJ, SDE-NB, SDE-JB, S. -->
    <property name="vpproduct" value="VP-UML" />
    <property name="vpworkspace" value="${user.home}/vpworkspace" />
    <property name="headless" value="false" />

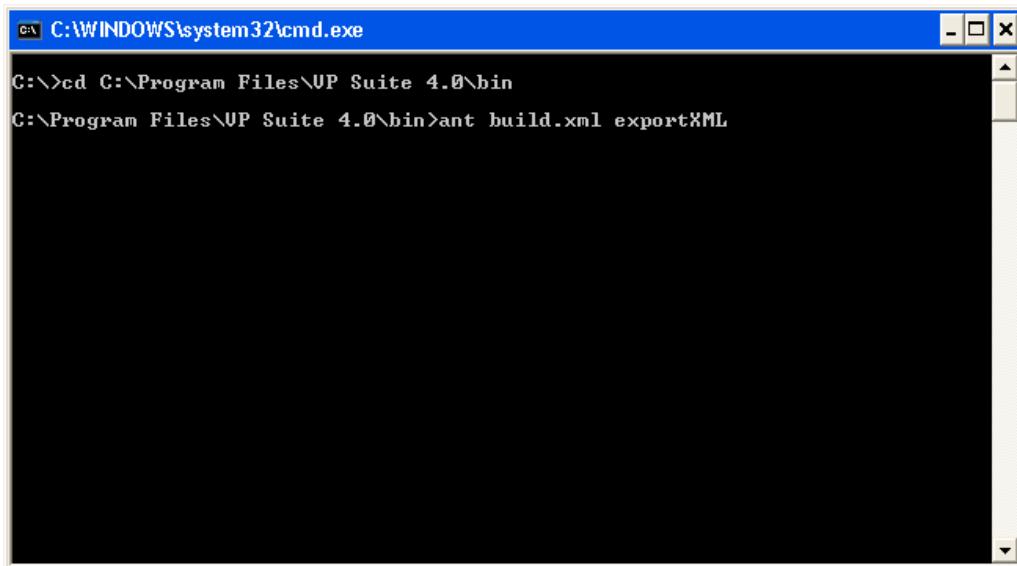
    <taskdef resource="com/vp/ant/taskdef.properties" classpath="${vpsuiteInstallationP:>

```

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 VP Suite installation directory.

7. Enter **ant build.xml**, and then the task name to execute specific task.



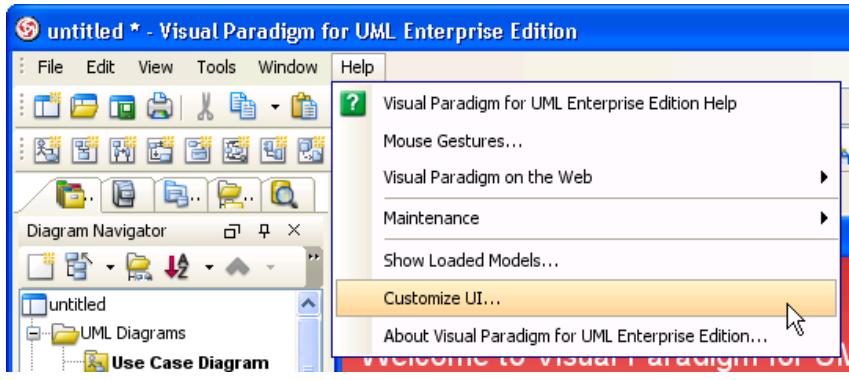
```
C:\>cd C:\Program Files\UP Suite 4.0\bin
C:\Program Files\UP Suite 4.0\bin>ant build.xml exportXML
```

Execute an ant script

Hiding user interface components

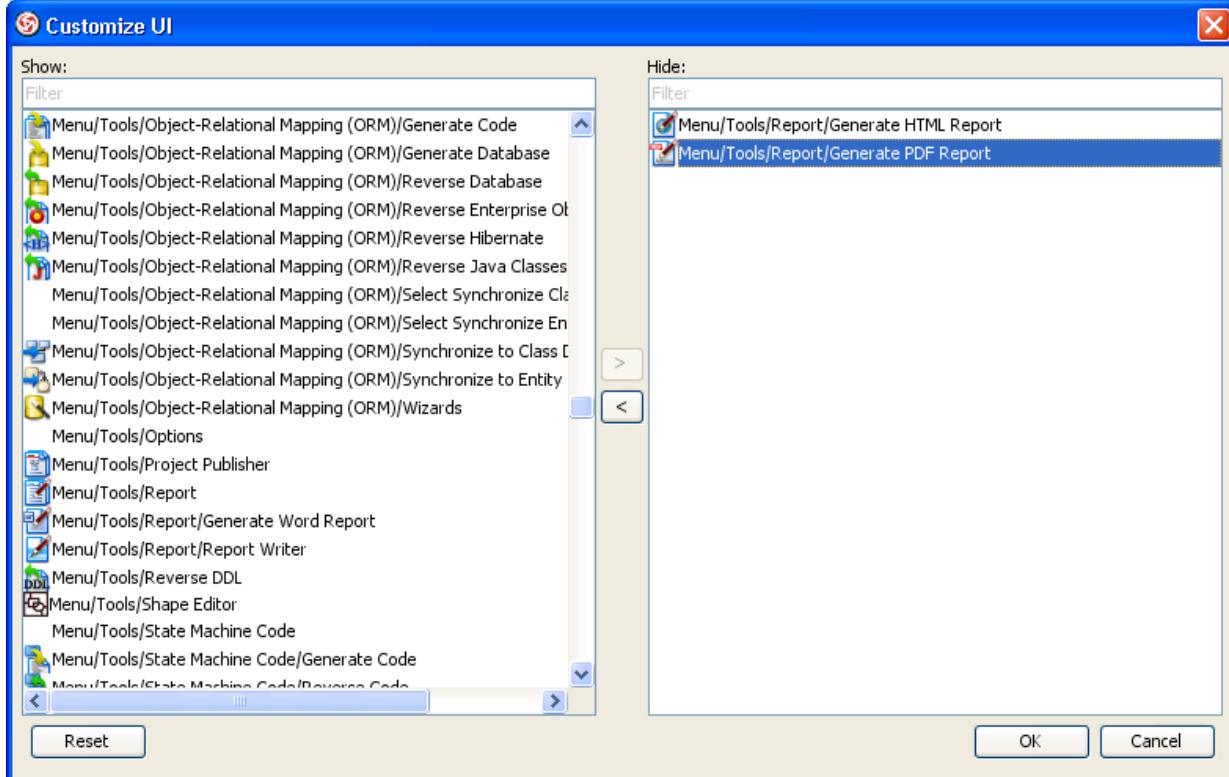
You may want to hide away some diagram types, menu items or toolbar items to avoid your team creating wrong types of model. This can be done by user interface (UI) customization.

1. Select **Help > Customize UI** from the main menu.



To open the customize UI dialog box

2. In the **Customize UI** dialog box, select the menu(s)/toolbar button(s)/popup menu(s) to hide, and click on the **>** button to move them to the 'Hide:' list.



Select the menus to hide

3. Click **OK** to confirm. By restarting the application, the chosen user interface components will be hidden.

