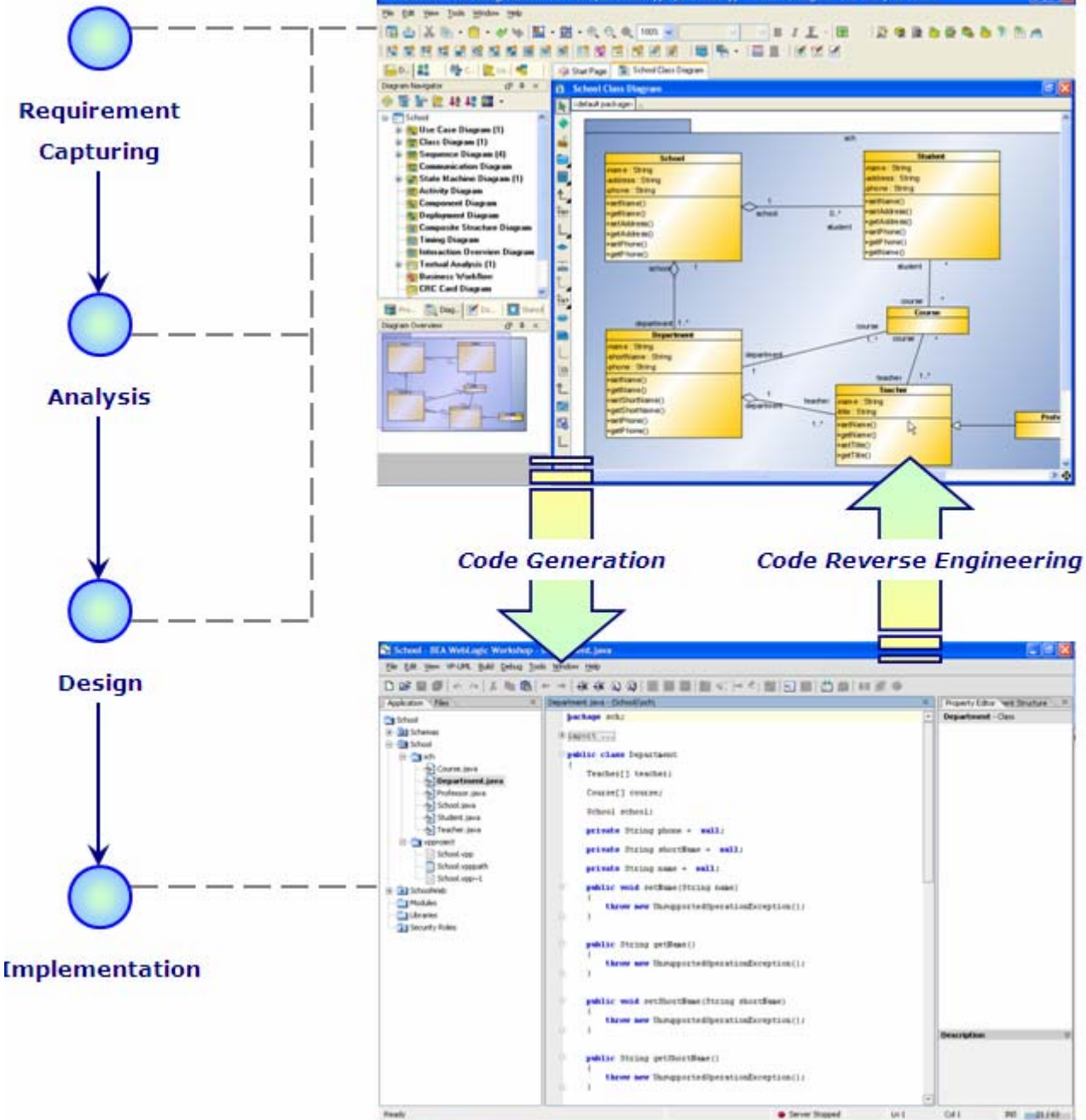


6

Integration with WebLogic Workshop™

Chapter 6 – Integration with WebLogic Workshop™

Overview



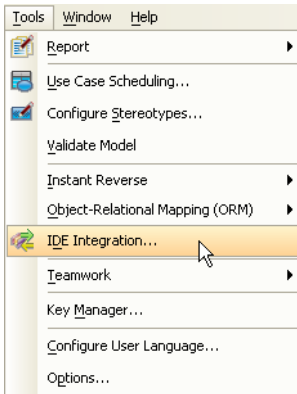
VP-UML Professional and Enterprise Edition allows you to integrate the VP-UML module with WebLogic Workshop™, providing full software development life cycle support. By designing your software system in VP-UML, you can generate programming source code from class diagram to a WebLogic Workshop™ project. Also, you can reverse engineer your source code into class models in VP-UML.

Installation

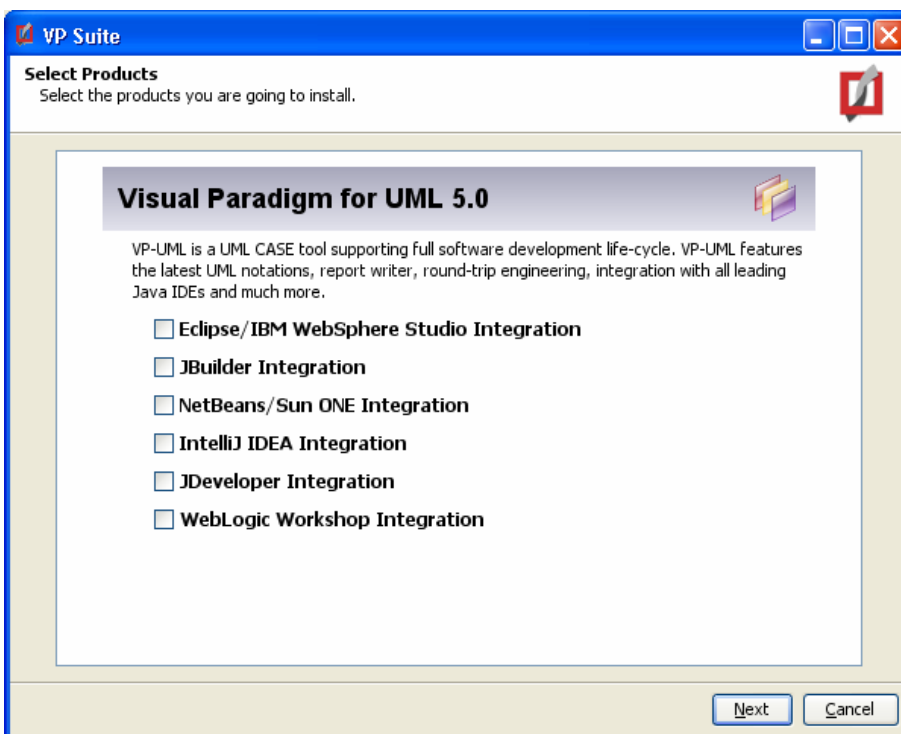
First of all, please make sure your machine has WebLogic Workshop™ 8.1 or above properly installed.

To install WebLogic Workshop™ Integration from VP-UML:

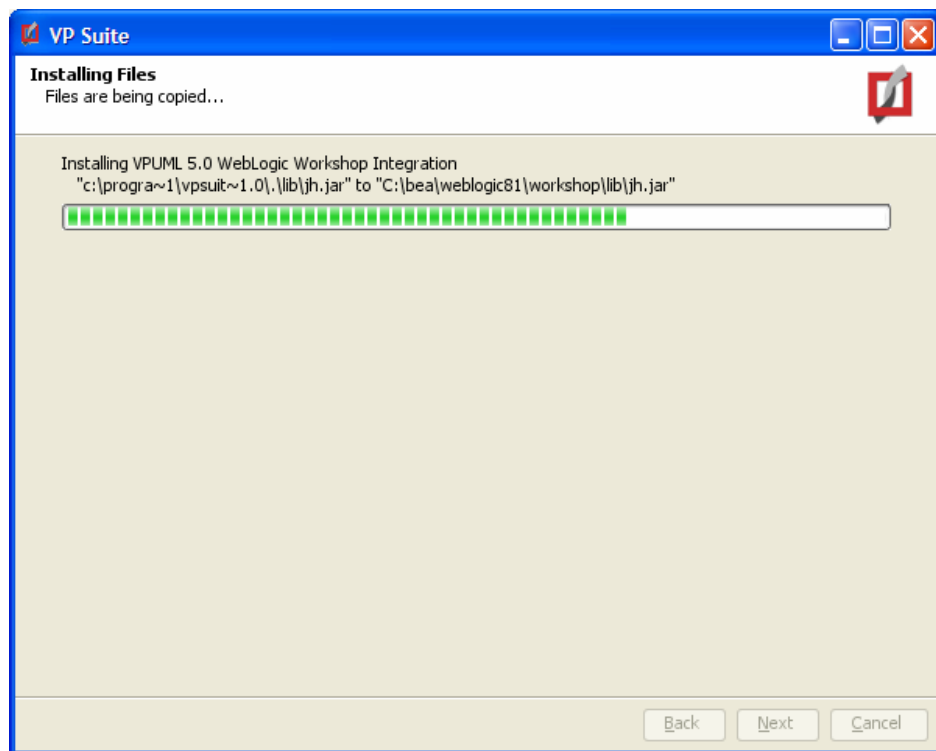
1. Start VP-UML
2. Select **Tools > IDE Integration...** from the main menu of VP-UML.



3. This displays the **VP-Suite** dialog box.



4. Check **WebLogic Workshop Integration**.
5. Click **Next**. This displays the **Product Configuration** page.
6. Locate the WebLogic Workshop™ home directory in the **Directory** field. You can enter the path directly in the text field, or press **...** to select the directory from the **Open** dialog box.
7. Click **Next**. This displays the **Installing Files** page and starts the installation process.



8. Click **Finish** to close the dialog box when the installation is completed.



You can install WebLogic Workshop™ Integration only on a “clean” copy of WebLogic Workshop™. A clean copy of WebLogic Workshop™ is one which has no other kinds of integration such as SDE for WebLogic Workshop™ and DB-VA for WebLogic Workshop™ installed.

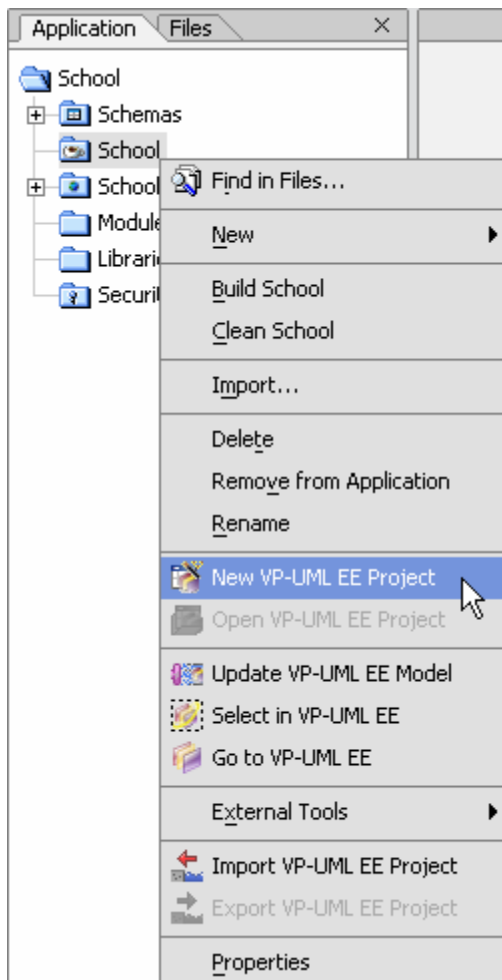


WebLogic Workshop™ Integration can only be installed on ONE WebLogic Workshop™ directory only. The next time you start the VP-Suite dialog from VP-UML you will see the option WebLogic Workshop™ Integration is selected.

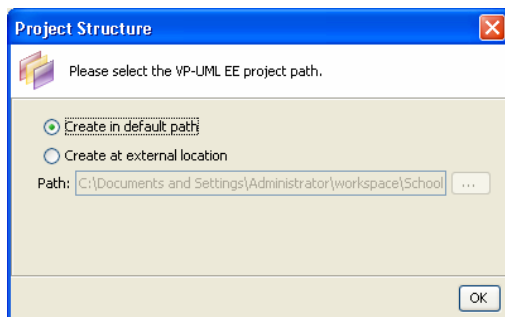
Creating a VP-UML Project in WebLogic Workshop™

To create a VP-UML Project in WebLogic Workshop™:

1. Start WebLogic Workshop™.
2. Select the WebLogic Workshop™ project for which you want to create a VP-UML project for it.
3. Right-click on the selected project and choose **New VP-UML %EDITION% Project** from popup menu.

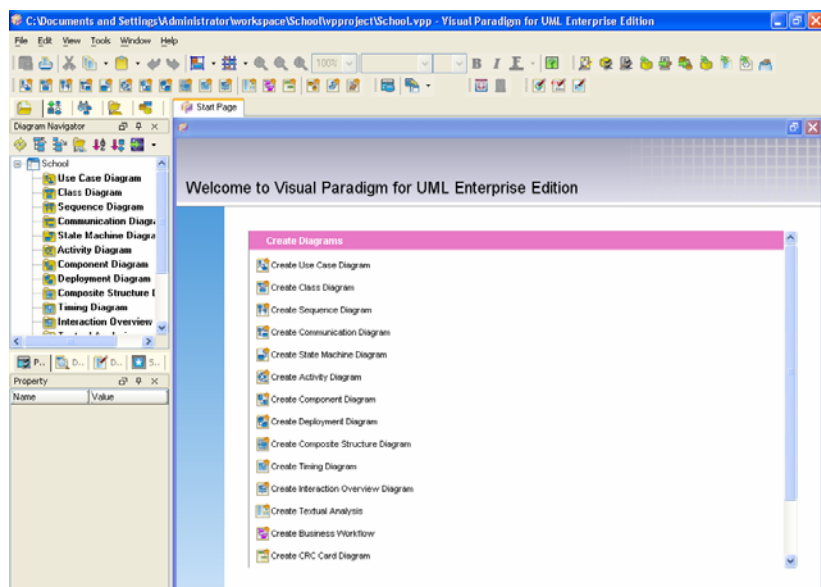


This displays the **Project Structure** dialog box.



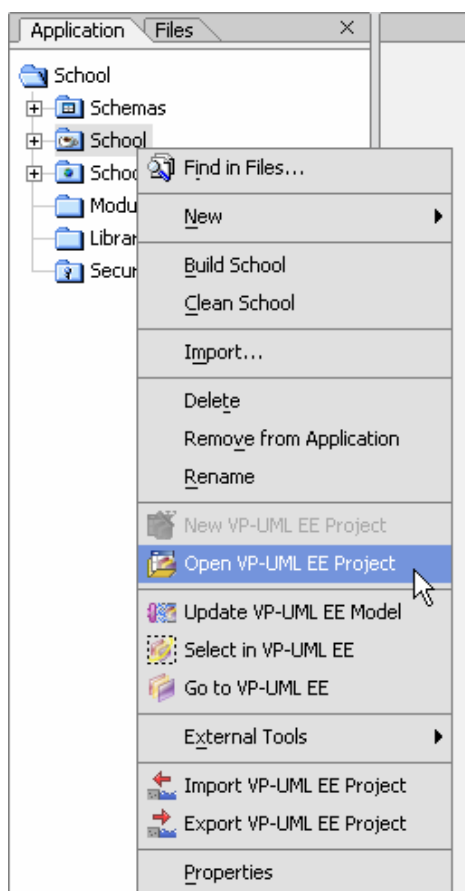
4. Select from the **Project Structure** dialog box the location of the VP-UML project is to be saved. The VP-UML project, with .vpp extension, is the UML project file that is going to be associated with the selected WebLogic Workshop™ project file. Select **Create in default path** result in saving the VP-UML project to %WebLogic_Workshop_Project_Directory%/vpproject while selecting **Create at external location** require you to specify the project path.
5. Click **OK**.

This starts a new instance of VP-UML on a separate window. The project opened from VP-UML is associated with the WebLogic Workshop™ project.



Opening a VP-UML Project from WebLogic Workshop™

1. Start WebLogic Workshop™.
2. Select the WebLogic Workshop™ project for which you want to open the VP-UML project from it.
3. Right-click on the selected project and choose **Open VP-UML %EDITION% Project** from popup menu.



This starts a new instance of VP-UML on a separate window. The project opened from VP-UML is associated with the WebLogic Workshop™ project.

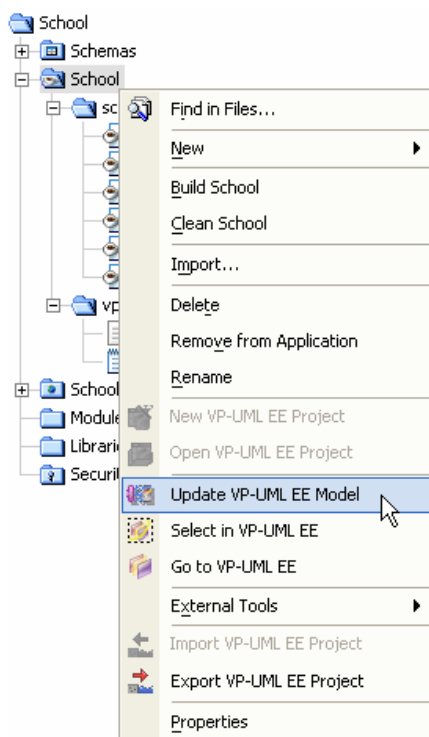
Reverse Engineering from Code to Model

Code reverse engineering updates UML class models from source code in WebLogic Workshop™. You can update the whole project, package (s) and class (es) from WebLogic Workshop™ to VP-UML. Before reverse engineering, you must open the UML model of the desired project. More information about how to open a VP-UML from WebLogic Workshop™ can be found from the section [Opening a VP-UML Project from WebLogic Workshop™](#) in this Chapter.

Project Based Reverse Engineering

You can update models in VP-UML from a WebLogic Workshop™ project. Models of the selected project, child packages and classes will be updated or created (if the models are not already exists).

To reverse engineer code from a WebLogic Workshop™ project to UML model, right-click on the project node in WebLogic Workshop™ and select **Update VP-UML %EDITION% Model** from popup menu.

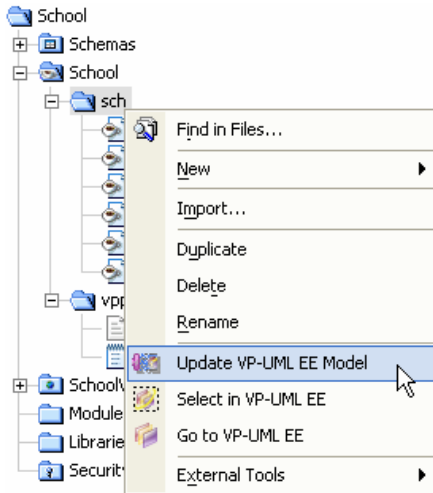


This updates/creates the corresponding UML model in VP-UML.

Package Based Reverse Engineering

You can update models in VP-UML from a WebLogic Workshop™ project. Models of the selected package, child packages and classes will be updated or created (if the models are not already exists).

To reverse engineer code from a package in an WebLogic Workshop™ project to UML model, right-click on the package in WebLogic Workshop™ and select **Update VP-UML %EDITION% Model** from popup menu.



This updates/creates the corresponding UML model in VP-UML.

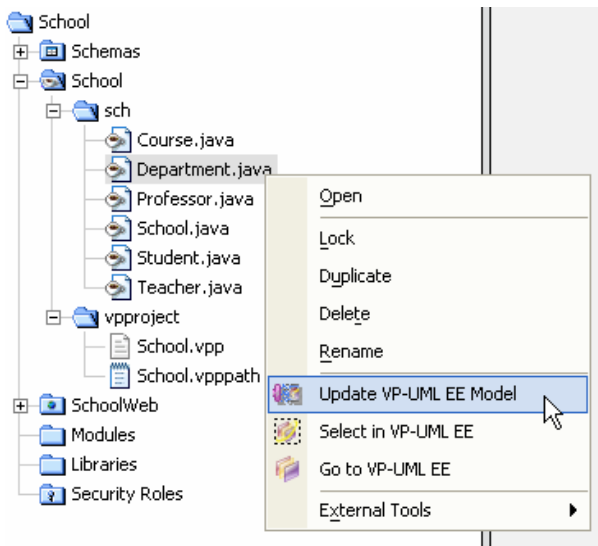


The model will be updated according to the package hierarchy.

Class Based Reverse Engineering

You can update models in VP-UML from a WebLogic Workshop™ project. Models of the selected class and child classes (inner class) will be updated or created (if the models are not already exists).

To reverse engineer code from a class in a WebLogic Workshop™ project to UML model, right-click on the class file in WebLogic Workshop™ and select **Update VP-UML %EDITION% Model** from popup menu.

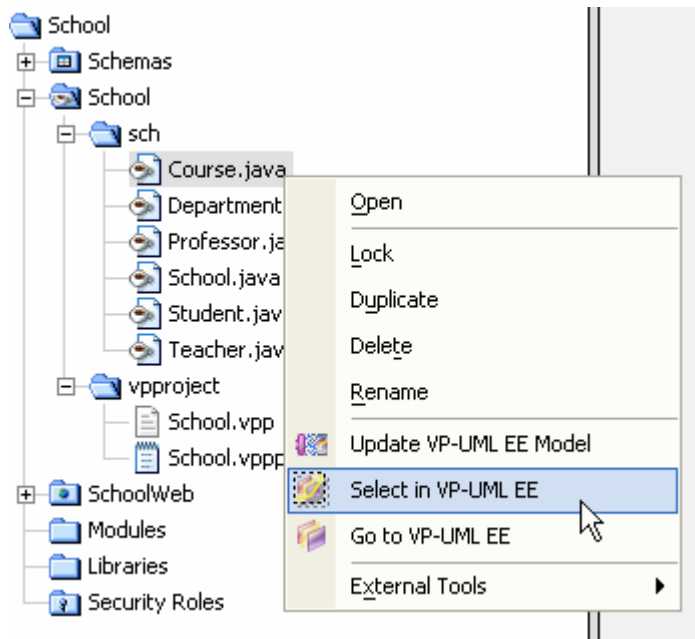


This updates/creates the corresponding UML model in VP-UML.

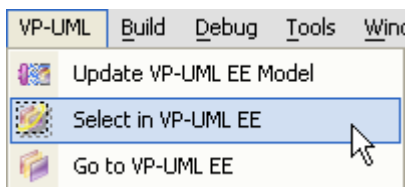
Selecting Corresponding Elements in VP-UML from WebLogic Workshop™

VP-UML helps selects class models in VP-UML corresponding to the source code in WebLogic Workshop™. To select corresponding models in VP-UML from WebLogic Workshop™, perform one of the following actions:

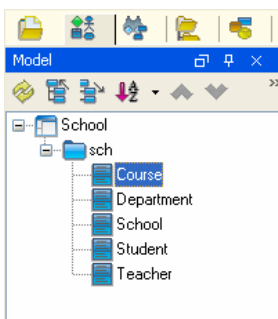
- Select on the desired piece of source elements from the WebLogic Workshop™. Right-click on the selection and choose **Select in VP-UML %EDITION%** from popup menu.



- When the code editor for the desired piece of source is active, choose **VP-UML > Select in VP-UML %EDITION%** from WebLogic Workshop™ main menu.



In both cases, the corresponding model in VP-UML is selected.



Code Generation from Model to Code

Code generation creates/updates source code in a WebLogic Workshop™ project from UML models. You can select to update the whole project, package (s) and class (es) from VP-UML to WebLogic Workshop™. Before updating/generating code, you must open the UML model of the desired project. More information about how to open a VP-UML from WebLogic Workshop™ can be found from the section [Opening a VP-UML Project from WebLogic Workshop™](#) in this Chapter.

Furthermore, there are 2 ways in synchronizing (updating) model to code. They are **Sync to Code** and **Force Sync to Code**.

Sync to Code

Removed sources will not be recovered. Only the existing sources will be synchronized with the UML models.

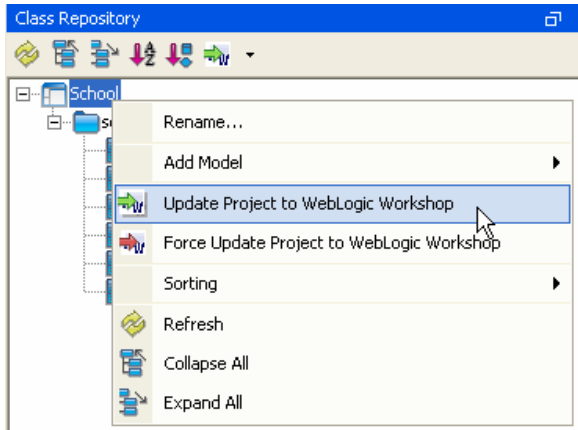
Force Sync to Code

Removed sources will be recovered. Existing sources will be synchronized with the UML models, while removed sources will be reconstructed.

Project Based Code Generation

To generate all classes and/or packages within a UML project:

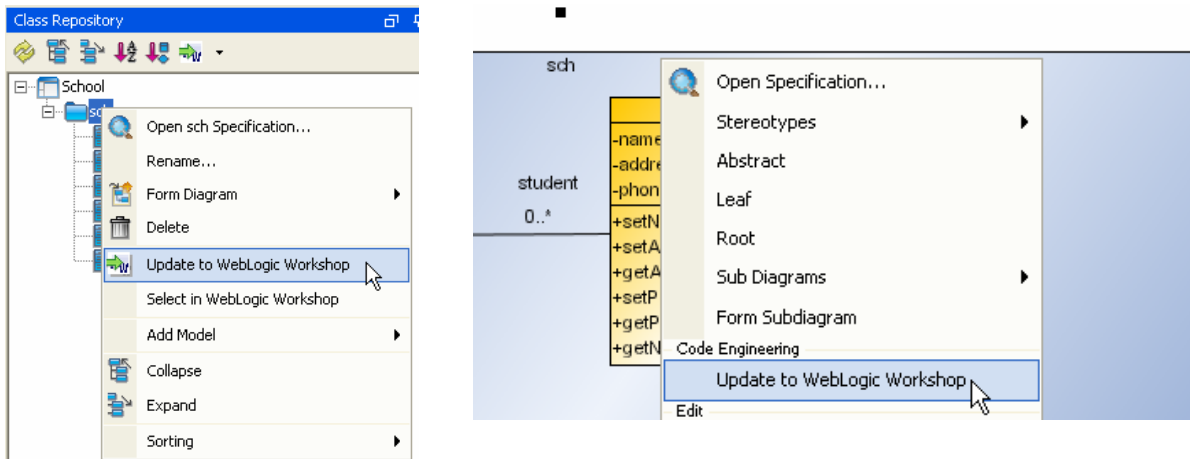
1. Right-click the project (the root) in the **Diagram Navigator**, in the **Model Tree** or in the **Class Repository Tree**.
2. Select **Update Project to WebLogic Workshop** from popup menu.



Package Based Code Generation

To generate a package together with class/classes inside:

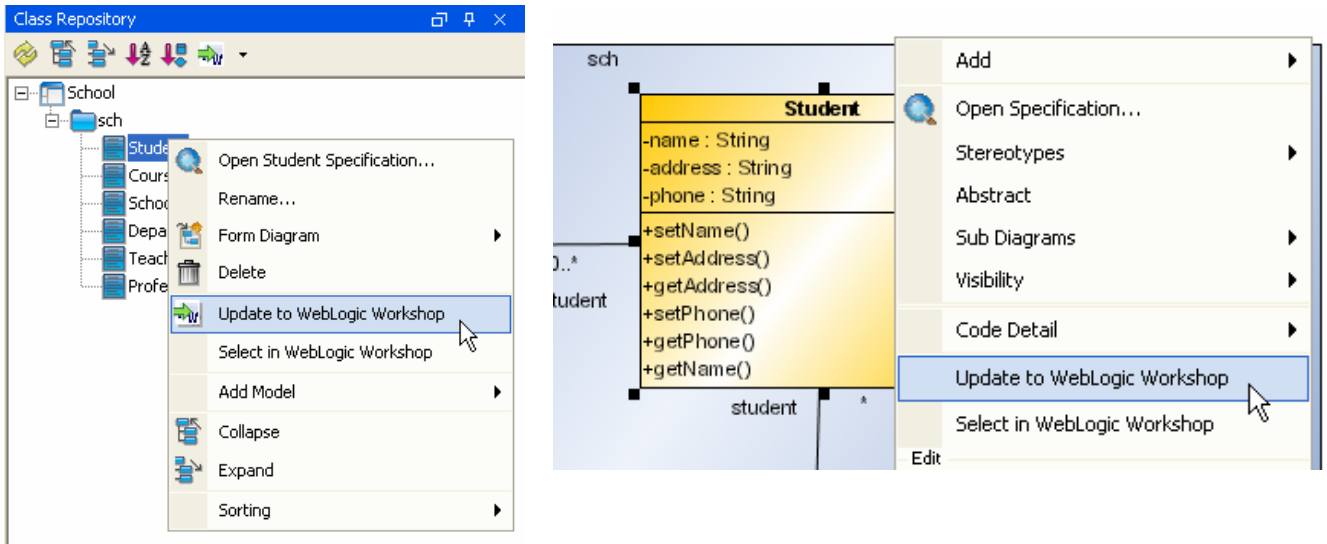
1. Select the desired package for generating code on a class diagram, in the **Diagram Navigator**, in the **Model Tree** or in the **Class Repository Tree**.
2. Right-click on the selection and choose **Update to WebLogic Workshop** from popup menu.



Class Based Code Generation

To generate class/classes:

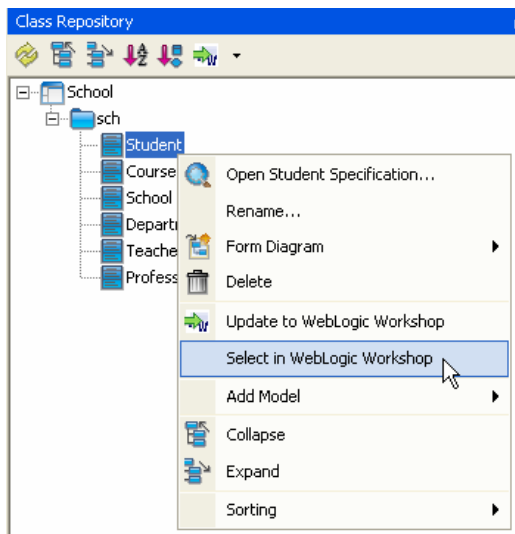
1. Select class/classes on a class diagram, in the **Diagram Navigator**, in the **Model Tree** or in the **Class Repository Tree** for generating code.
2. Right-click on the selection and select **Update to WebLogic Workshop** from popup menu.



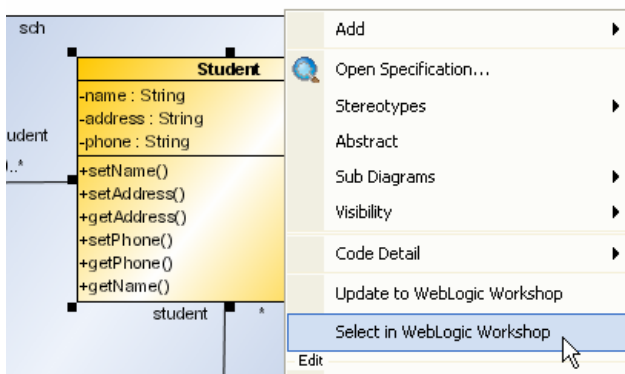
Selecting Corresponding Elements in WebLogic Workshop™ from VP-UML

VP-UML helps select element in VP-UML corresponding to the source code in WebLogic Workshop™. To select corresponding piece of source code in WebLogic Workshop™ from VP-UML, perform one of the following actions:

- Select on the desired UML model in the **Diagram Navigator**, in the **Model Tree** or in the **Class Repository Tree**. Right-click on the selection and choose **Select in WebLogic Workshop** from popup menu.



- Select the desired model for generating code on a class diagram. Right-click on the selection and choose **Select in WebLogic Workshop** from popup menu.



In both cases, the corresponding source files will be opened and activated in WebLogic Workshop™.

```
package sch;
import ...
public class Student
{
    Course[] course;
    School school;
    private String phone = null;
    private String address = null;
    private String name = null;
    public void setName(String name)
    {
        throw new UnsupportedOperationException();
    }
    public void setAddress(String address)
    {
        throw new UnsupportedOperationException();
    }
}
```

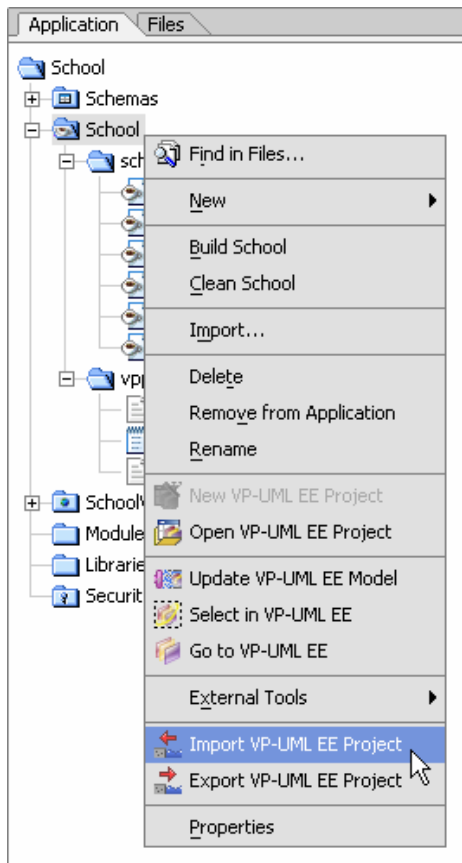
Importing a VP-UML Project into WebLogic Workshop™

You can import another VP-UML project into a WebLogic Workshop™ project. If there is a VP-UML associating with the WebLogic Workshop™ project, importing a VP-UML project results in replacing the original one with the one going to be imported.

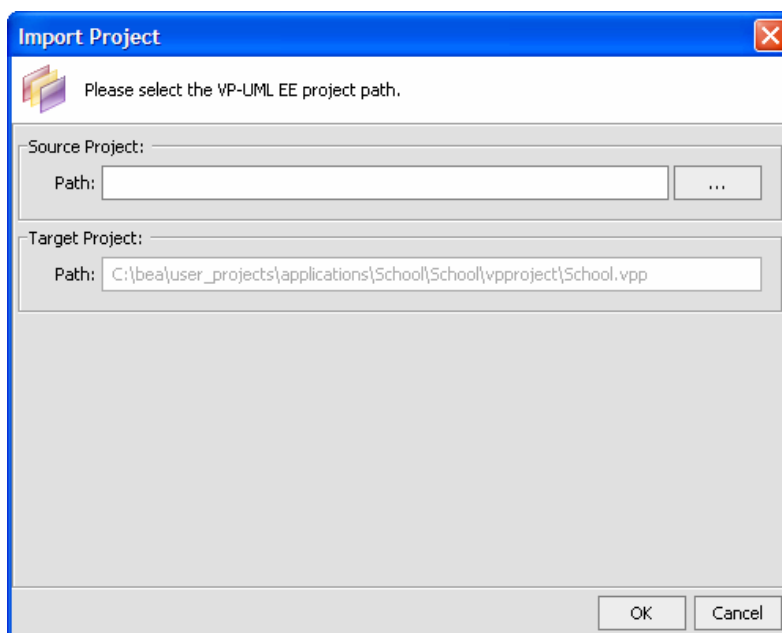
Before importing a VP-UML project, you must open the UML model of the desired project. More information about how to open a VP-UML from WebLogic Workshop™ can be found from the section [Opening a VP-UML Project from WebLogic Workshop™](#) in this Chapter.

To import a VP-UML project:

1. Select the WebLogic Workshop™ project for which you want to import a VP-UML project into it.
2. Right-click on the selected project and choose **Import VP-UML %EDITION% Project** from popup menu.

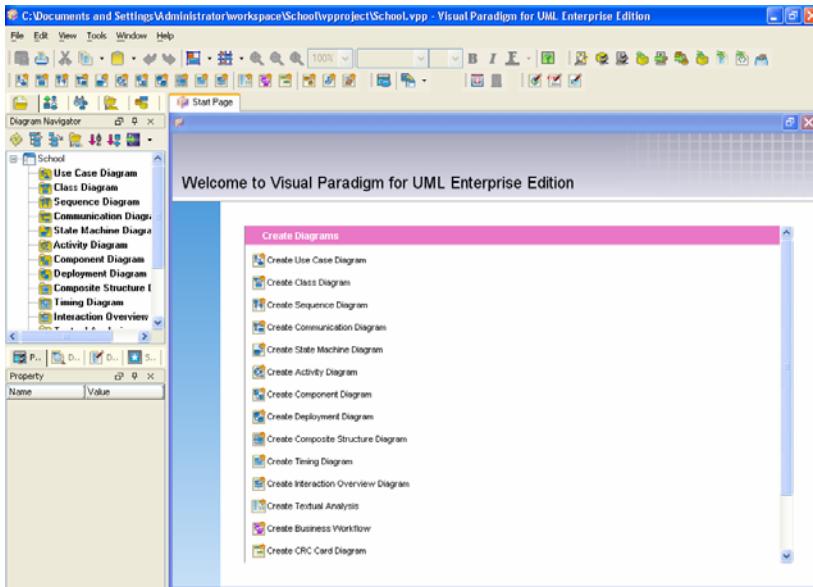


- This displays the **Import Project** dialog box.



- Locate the file path of the source VP-UML project. The source project is the one that is going to be imported into the selected WebLogic Workshop™ project. If there is an existing VP-UML project associated with the selected WebLogic Workshop™ project, the target project path is the path of the existing VP-UML project. If there is no existing VP-UML project associated with the selected WebLogic Workshop™ project, the path is the default one, which is %WebLogic_Workshop_Project_Directory%/vpproject.
- Click **OK**.

This starts a new instance of VP-UML on a separate window. The project opened from VP-UML is the imported one which is now associated with the WebLogic Workshop™ project.



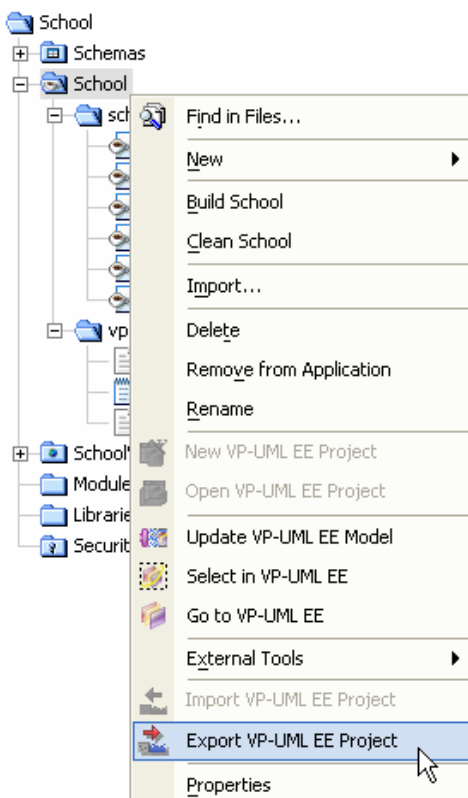
Exporting a VP-UML Project from WebLogic Workshop™

You can export the VP-UML project that is associated with a WebLogic Workshop™ project. The exported VP-UML project has no association with both the WebLogic Workshop™ project and the VP-UML project it exported from.

Before exporting a VP-UML project, you must open the UML model of the desired project. More information about how to open a VP-UML from WebLogic Workshop™ can be found from the section [Opening a VP-UML Project from WebLogic Workshop™](#) in this Chapter.

To export a VP-UML project:

1. Select the WebLogic Workshop™ project for which you want to export the VP-UML associated with it.
2. Right-click on the selected project and choose **Export VP-UML %EDITION% Project** from popup menu.







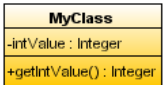
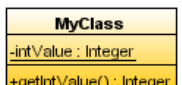
This displays the **Save As** dialog box.

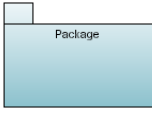
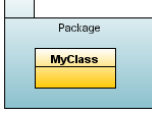
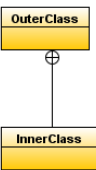
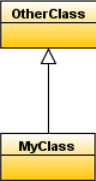
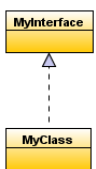
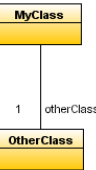
3. Locate the project file path for the output project.
4. Click **Save**.

The VP-UML project is exported to the specified location.

Model Representation of Code

The following table shows some of the model representation of code.

	Model	Code
Class		<pre>public class MyClass { }</pre>
Abstract Class		<pre>public abstract class MyClass { }</pre>
Attribute		
Instance Scope		<pre>public class MyClass { private Integer intValue; }</pre>
classifier Scope		<pre>public class MyClass { private static Integer intValue; }</pre>
Operation		
Instance Scope		<pre>public class MyClass { private Integer intValue; public integer getIntValue() { } }</pre>
Classifier Scope		<pre>public class MyClass { private static Integer intValue; }</pre>

		<pre> public static Integer getIntValue() { } </pre>
Package		<pre> package Package; </pre>
Containment		<pre> package Package; public class MyClass { } </pre>
Inner Class		<pre> public class OuterClass { class InnerClass { } } </pre>
Generalization		<pre> public class MyClass extends OtherClass { } </pre>
Realization		<pre> public class MyClass implements MyInterface { } </pre>
Association		<pre> public class MyClass { Otherclass otherClass; } </pre>