



---

## **PathMATE Graphical User Interface**

Version 1.11  
July 26, 2005

---

### *PathMATE Technical Notes*

Pathfinder Solutions LLC  
33 Commercial Drive, Suite 2  
Foxboro, MA 02035 USA  
[www.PathfinderMDA.com](http://www.PathfinderMDA.com)  
888-662-7284

# Table Of Contents

<b>1. History .....</b>	<b>1</b>
<b>2. Introduction .....</b>	<b>1</b>
Built on the Eclipse Framework .....	1
<b>3. Product Overview .....</b>	<b>3</b>
Terminology .....	3
Phase 1 – Replace Transformation Batch Scripts .....	4
Phase 2 – Markings, Instances, and Batch Build Support .....	5
Phase 3 – Deployments .....	6
Getting Started .....	7
<b>4. Features of the PathMATE Editor .....</b>	<b>25</b>
<b>5. Using Default Deployments .....</b>	<b>26</b>
<b>6. Creating Deployments .....</b>	<b>26</b>
Starting from an Existing Deployment .....	26
Creating a New Deployment .....	27
Editing a Deployment .....	27
<b>7. Creating Transformations .....</b>	<b>28</b>
Starting from an Existing Transformation .....	28
Creating a New Transformation .....	28
Editing a Transformation .....	28
<b>8. Transforming PIMs into Implementation Code .....</b>	<b>31</b>
<b>9. Validating PathMATE Projects and Platform Models .....</b>	<b>32</b>
Selecting the Scope of Validation .....	32
Starting Validation .....	32
Determining the Result of Validation .....	32
<b>10. Selecting Directories .....</b>	<b>34</b>
<b>11. Creating Custom Platform Models .....</b>	<b>37</b>
Starting From an Existing Transformation Map .....	37
Creating a New Transformation Map .....	37
Editing a Transformation Map .....	37

<b>12. Deploying Custom Platform Models .....</b>	<b>39</b>
Creating an Extension Point Project .....	39
Defining a Platform Model Extension Point .....	40
Building the Plug-in .....	41
Installing the Plug-in .....	42
<b>13. Editing Preferences .....</b>	<b>42</b>
Showing/Hiding Resolved Paths .....	43
Defining New Path Variables .....	43
Removing Path Variables .....	43
<b>14. Automated Build .....</b>	<b>44</b>
<b>15. Phase 2 .....</b>	<b>45</b>
Transformation Map .....	45
Transformation .....	47
Incremental Domain Generation .....	51
<b>16. Phase 3 .....</b>	<b>52</b>
Platforms .....	52
Transformation Map .....	52
Transformation .....	53
Deployment .....	54
<b>A. Validation Error Messages .....</b>	<b>60</b>
Deployments .....	60
Transformations .....	61
Transformation Maps .....	62
Platform Independent Model .....	63

## 1. History

Version	Changes
1.5	Added more detail to Ant integration
1.6	Moved Ant integration from Phase 2 to Phase 1
1.7	Added validation Changed name of parse button on Transformation Map details page to Verify Changed transform procedure
1.8	Added JRE configuration instructions for Ant
1.9	Change plugin names in the Deploying Custom Maps section Allow directories with .. in them in Selecting Directories section Describe how to use alt_working_dir in Deploying Custom Maps section Update Rose Integration section – Describe transformation options and connectivity option dialogs
1.10	Added details to Phase 2 Incremental Transformation Section Added error message to Transformation Map required for single domain transformation
1.11	Describe workspace browser on Rational Software Modeler transform configuration tab

## 2. Introduction

The PathMATE Graphical User Interface (GUI) provides one central user interface to the PathMATE suite. It collects the inputs to and results from of the Transformation Engine and Maps in one convenient location. The GUI will also provide advanced capabilities that support:

- Transformation, build, and test of multiple deployments of the same Platform Independent Model
- Coordinated development of product line elements
- Report generation for multiple deployments
- Acceleration of transformation and build cycle
- Domain reuse

In addition to adding powerful new capabilities, the GUI will also make PathMATE easier to use. It replaces the batch files currently used to capture code generation options as well as the text file based marking approach (properties.txt) with an easy to use GUI interface and a repository that automatically keeps configurations and deployments up to date with the model data. Import capabilities will be provided to upgrade existing properties.txt files.

### Built on the Eclipse Framework

The PathMATE GUI is built on the Eclipse Framework. Eclipse is an open source Integrated Development Environment which is rapidly gaining acceptance across the software development industry. Eclipse provides a very flexible plug-in architecture, a rich set of development

tools from multiple vendors as well as open source projects, and powerful integrations with a variety of tools including IBM Rational Software Architect and embedded compiler and OS vendors such as Wind River and Accelerated Technology.

## 3. Product Overview

### Terminology

A **Platform Independent Model** (PIM) is a complete specification of a problem solution in terms of the problem space itself. The PIM is a UML model containing a PathMATE system and all of its domains. Analyzed domain implementations are modeled. The PIM captures a platform independent interface for realized domains which can be implemented in one or more target programming languages.

A **Transformation** specifies how to transform a PIM or subset of a PIM into implementation code for a specific Platform. This includes the transformation map used to generate code and reports, which domain implementations to use, the population of static class instances available at startup, PAL preprocessor and include paths, and platform dependent markings to apply to PIM elements.

A **Deployment** defines one possible implementation of a PIM on a target platform and target system topology. **System Topologies** define the processing elements in the system such as processing nodes, processes, and tasks and the paths of communication between the processing nodes. Deployments allocate Transformations to Processes. Deployments determine what executables are built, what makefile or project files are required to build the executables, as well as which tasks to start and how to route service invocations, incidents and incident handles.

A **PathMATE Project** contains a reference to a PIM and its associated Deployments and Transformations. A PathMATE Project may reside in an Eclipse project. PathMATE projects are stored in files with the .pathmate extension.

A **Platform Model** allows the user to specify Transformation Maps and Platform definitions. **Transformation Maps** specify the location of the code templates, the name of the top level templates, and the output directory. **Platform definitions** describe the markings available on the processing elements of a system topology. For example, operating system, build environment, and communication protocol specific settings. Platform Models are not specific to a PIM so they may be referenced by many PathMATE Projects. Platform Models will be supplied for the Maps included in PathMATE. Alternatively, the user can create Platform Models for their own custom Maps. The user can create plug-ins that extend PathMATE. The extensions specify which platform models to load and default deployments and transformations to create at when a new PathMATE Project is created.

Platform Models are stored in files with the .pathplat extension.

## Phase 1 – Replace Transformation Batch Scripts

Phase 1 of the PathMATE GUI replaces the current code generation batch scripts. Markings and instances will be read from the properties.txt and comma separated value files.

PathMATE currently supports four deployments – c, c++, java, and reports. PathMATE Projects may contain any number of deployments and will allow the user to locate the generated code in any directory. The PathMATE GUI supports path variables that allow PathMATE Projects to be moved easily from one machine to another.

Integration will be provided for IBM Rational Software Architect / Modeler first. IBM Rational Rose integration will be delivered in a follow-on release.

Finally, Phase 1 will support a batch build using the Ant capabilities integrated with Eclipse.

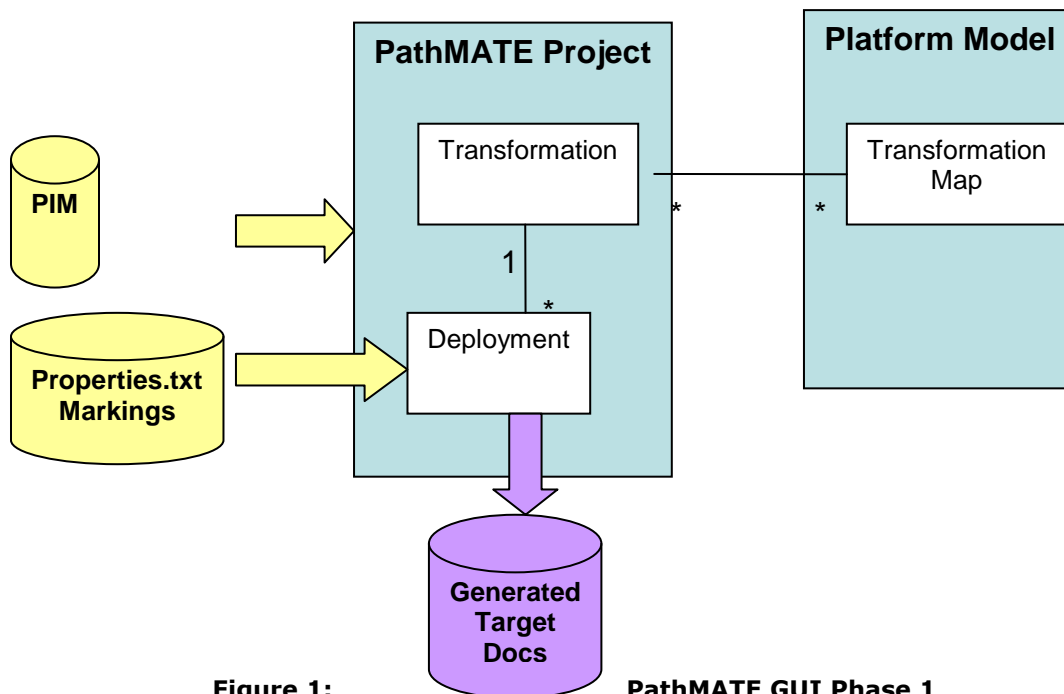


Figure 1:

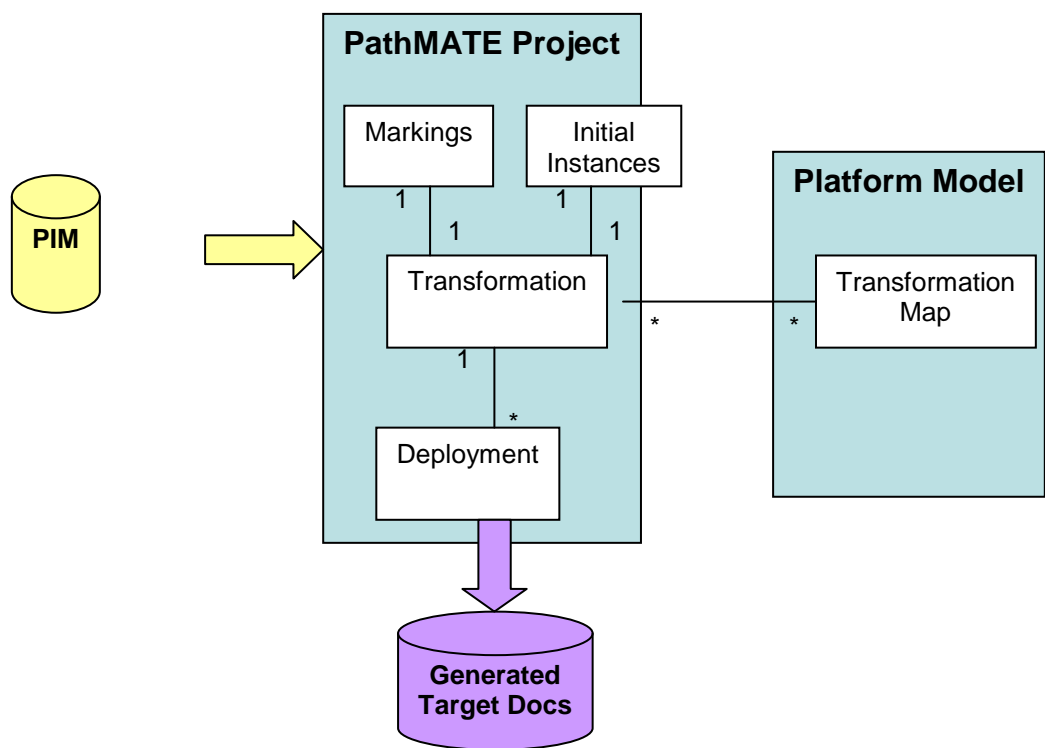
PathMATE GUI Phase 1

## Phase 2 – Markings, Instances, and Batch Build Support

Phase 2 allows the user to specify markings and initial instance populations. Phase 2 will support importing properties.txt and comma separated value files to provide a migration path for existing projects.

PathMATE GUI will reference model element universally unique identifiers (UUIDs) rather than their names eliminating the need to change markings and instances after renaming.

Phase 2 will provide support for incremental domain transformation. The user will be able to transform a selected domain in addition to the whole system.



**Figure 2: PathMATE GUI Phase 2**



### Phase 3 – Deployments

Phase 3 allows the user to specify system topologies for deployments. The PathMATE GUI will replace the markings currently used to specify deployments. In addition the deployment editor will allow the user to visualize their deployments.

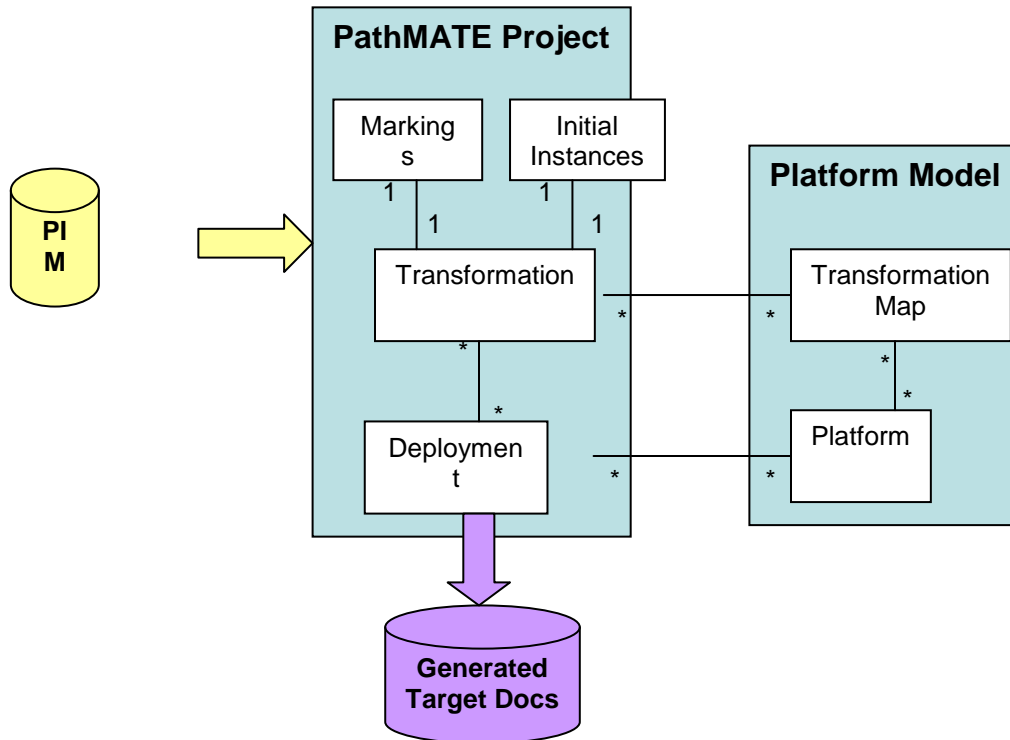
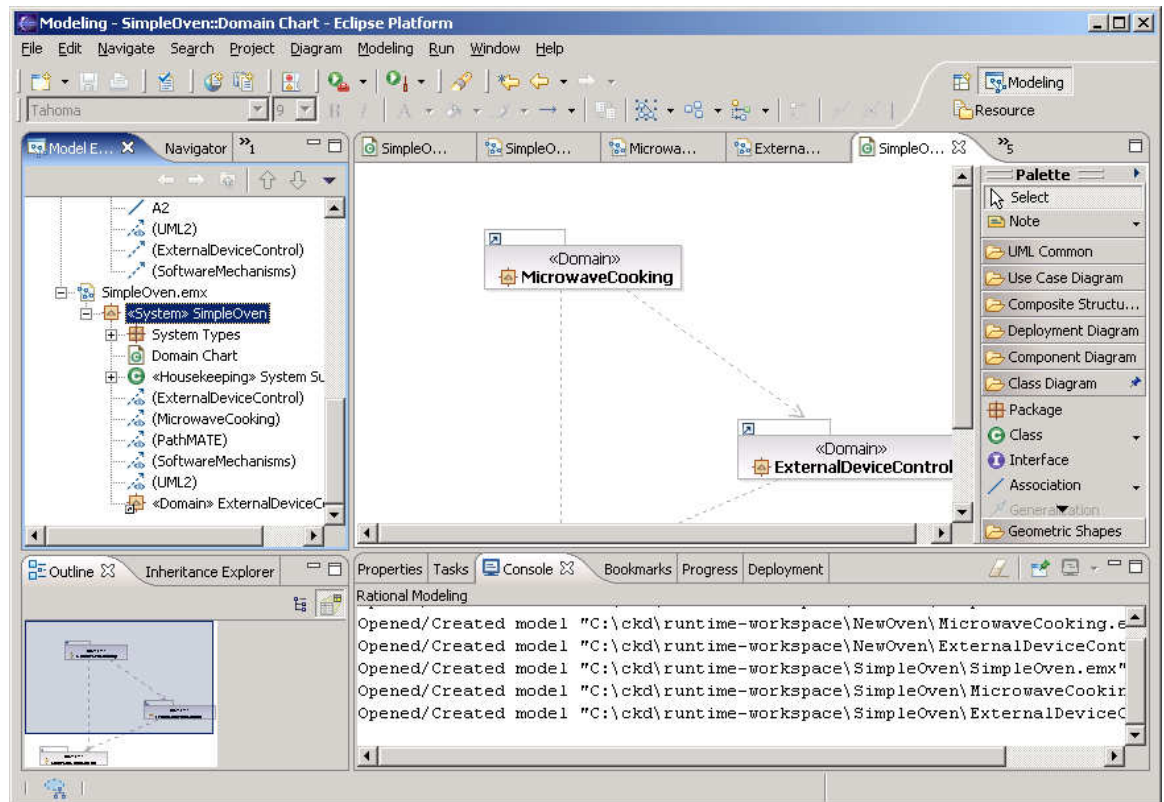


Figure 3: PathMATE GUI Phase 3

## Getting Started

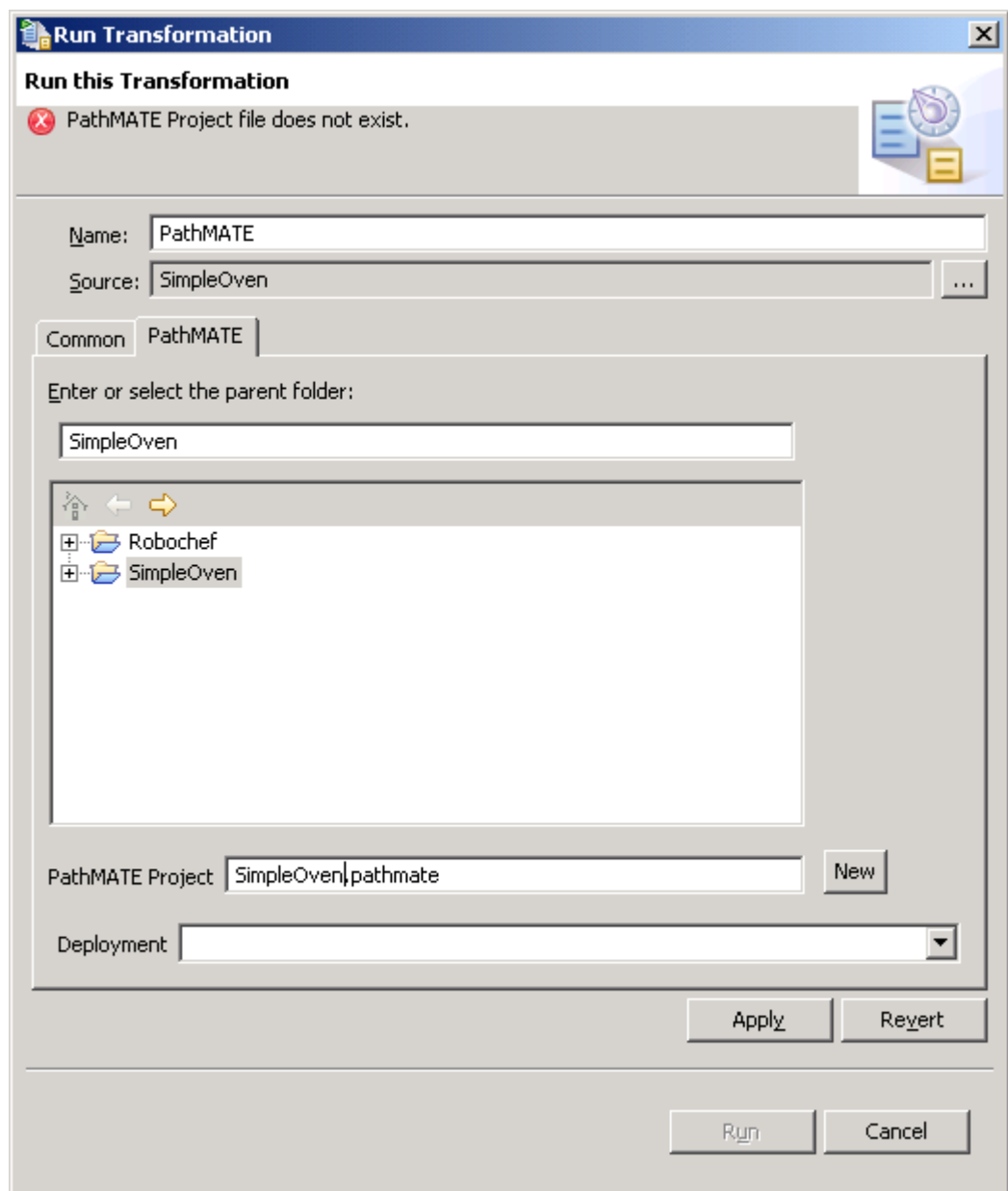
### From Rational Software Modeler

Open the Modeling perspective in Software Modeler. Select a PathMATE system model or domain from the Model Explorer view. See Figure 4.



**Figure 4: Rational Software Modeler with System Model Selected**

Right click and select Transform -> Run Transformation -> PathMATE. The Run Transformation dialog will appear. Click the PathMATE tab. See Figure 5.

**Figure 5: Run Transformation Dialog**

By default the project containing the model will be selected from the tree. The default PathMATE Project file name will be set as follows:

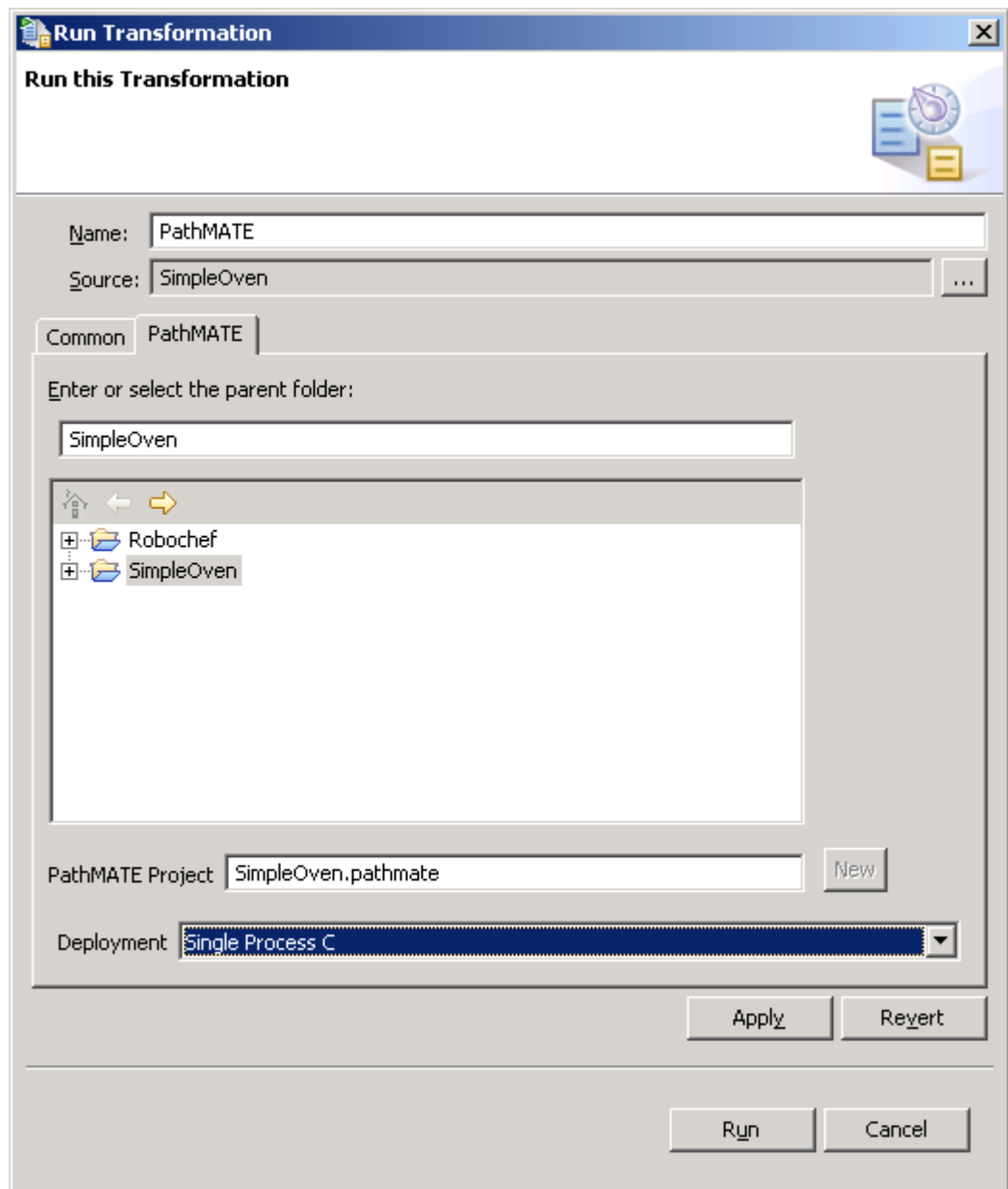
Source Type	Default Pathmate Project Name
System Model	Same name as system model or the first file contained in the project or its subdirectories with the extension .pathmate.

Domain Model	The first file contained in project or its subdirectories with the extension .pathmate. If none, no default name specified.
Domain Contained in System Model	Default for system model containing the domain.

If you want to select a different file name, select the file from the tree or enter a new folder or file name. New folders will be automatically created as long as they are contained in an existing project.

If the PathMATE Project does not exist, the New button will be selectable. To create a new PathMATE Project, press the New button. The project will be created and the New button will be disabled. The Deployment combo will be populated with the names of the deployments from the PathMATE Project. See Figure 6.

NOTE: The New button is only selectable when a system model source is selected.



**Figure 6: Run Transformation Dialog with Deployments**

If the PathMATE Project already exists, the deployment combo box will contain the available deployments with the active deployment selected.

Select a deployment from the combo box. Press the Run button.

The transformation runs. When the transformation is complete, any problems will appear in the Console view.

When a transformation is run, PathMATE associates a PathMATE project to a model. To repeat the transformation, select the system

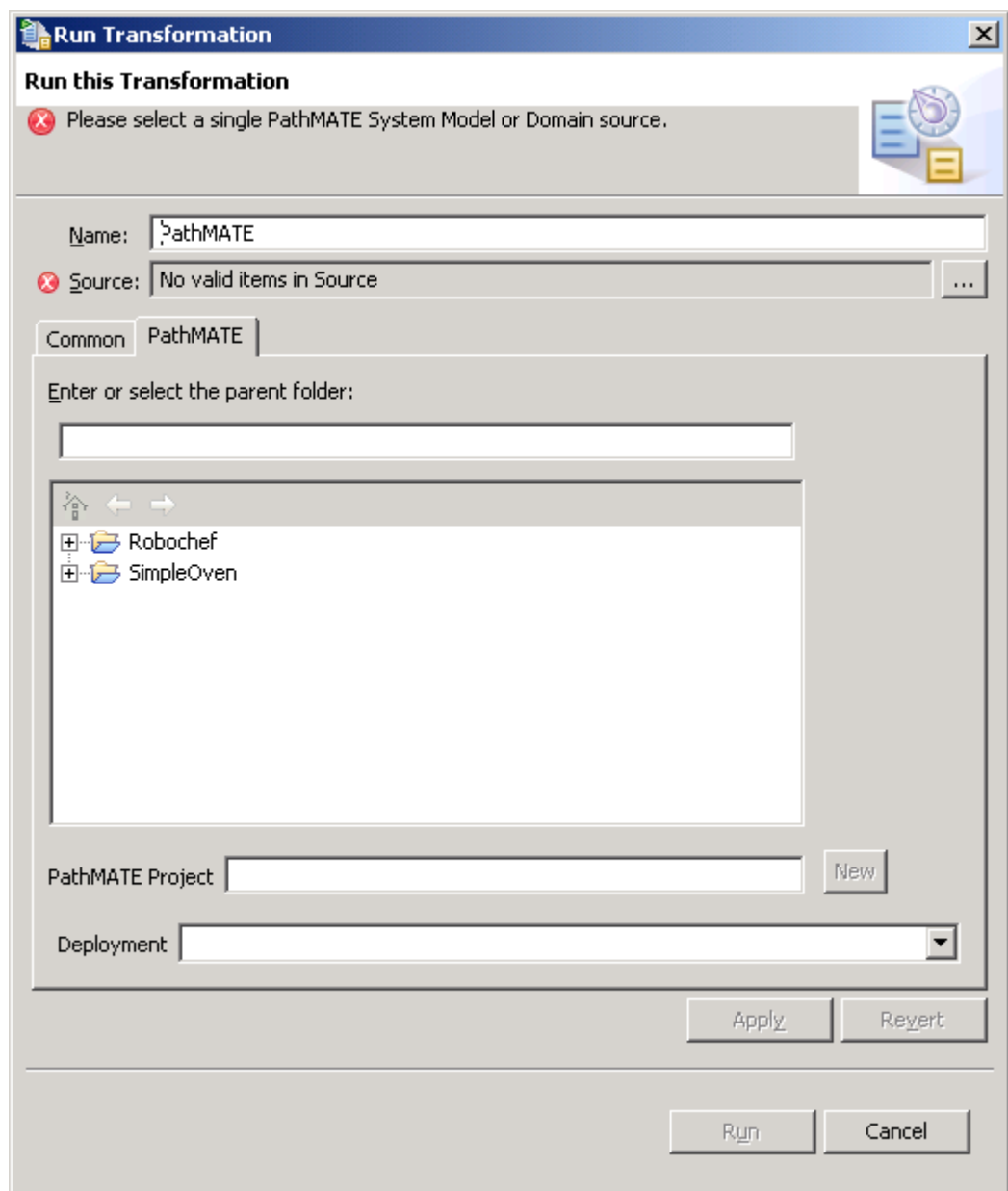
model and select Transform -> PathMATE. The transformation will use the active deployment of the PathMATE project used during the previous transformation.

To change the settings for a transformation, select a system or domain. Right click and select Transform > Run Transformation > PathMATE. The Run Transformation dialog will appear. Select a new PathMATE project and a deployment. Press Run to run the transformation.

### **Error Handling**

If you select a model element that is not a PathMATE System Model or domain, an error message will be displayed at the top of the Run Transformation dialog. See Figure 7. Press the ... button to select a PathMATE System Model or domain as the source.

TIP: Select the System Model or Domain from the Model Explorer rather than the .emx file.



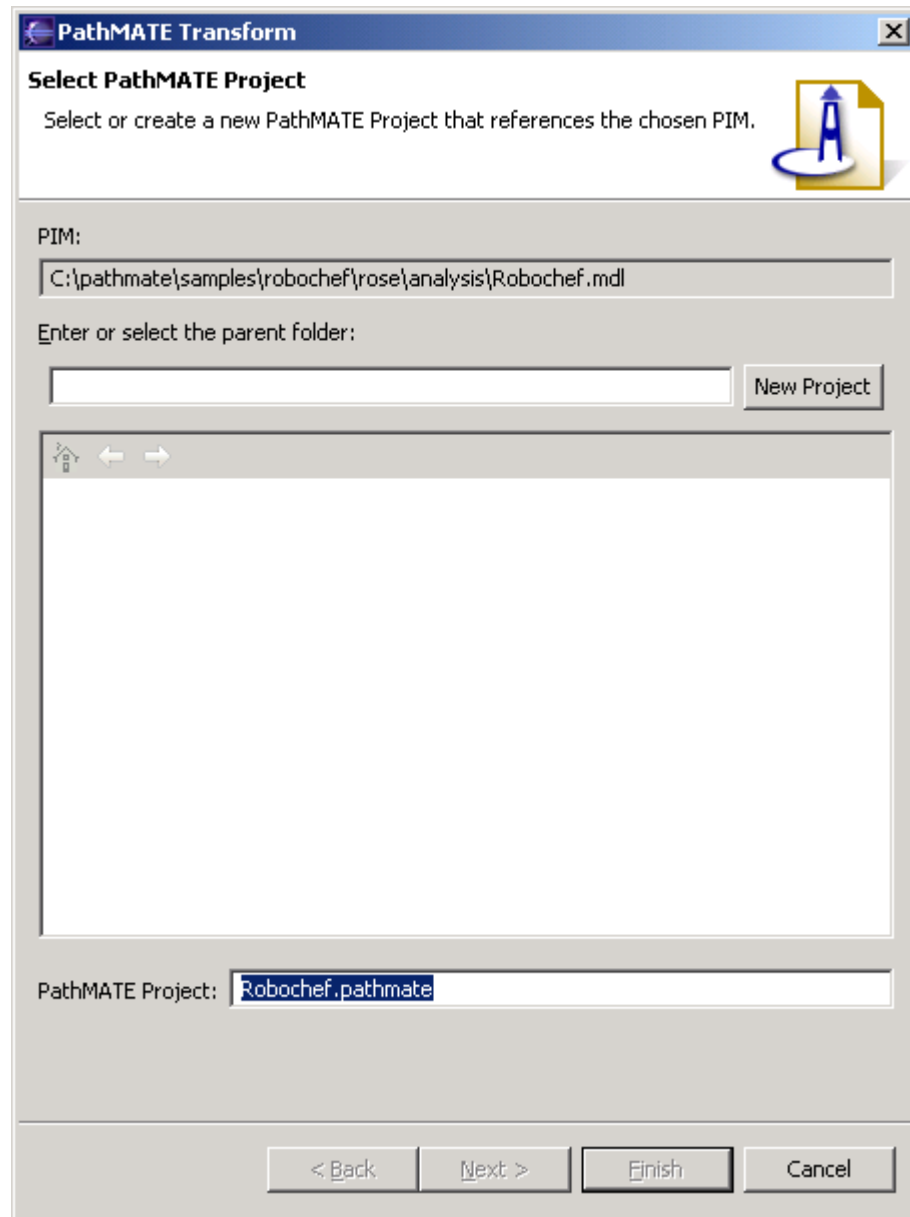
**Figure 7: Run Transformation Dialog with Error Message**

### **From Rose**

#### **Starting a Transformation**

Select Tools->PathMATE->Transform. The Extract progress dialog will appear in Rose as the XMI for the Rose model is extracted. Eclipse starts if it is not running already. If the Workspace Launcher dialog appears select the desired Eclipse workspace and press the OK button.

If this is the first time that this Rose model has been transformed in this workspace or if the PathMATE project or deployment used previously does not exist, the PathMATE Transform Wizard appears. By default, the PathMATE Project field will be populated with the name of the model followed by the .pathmate extension.



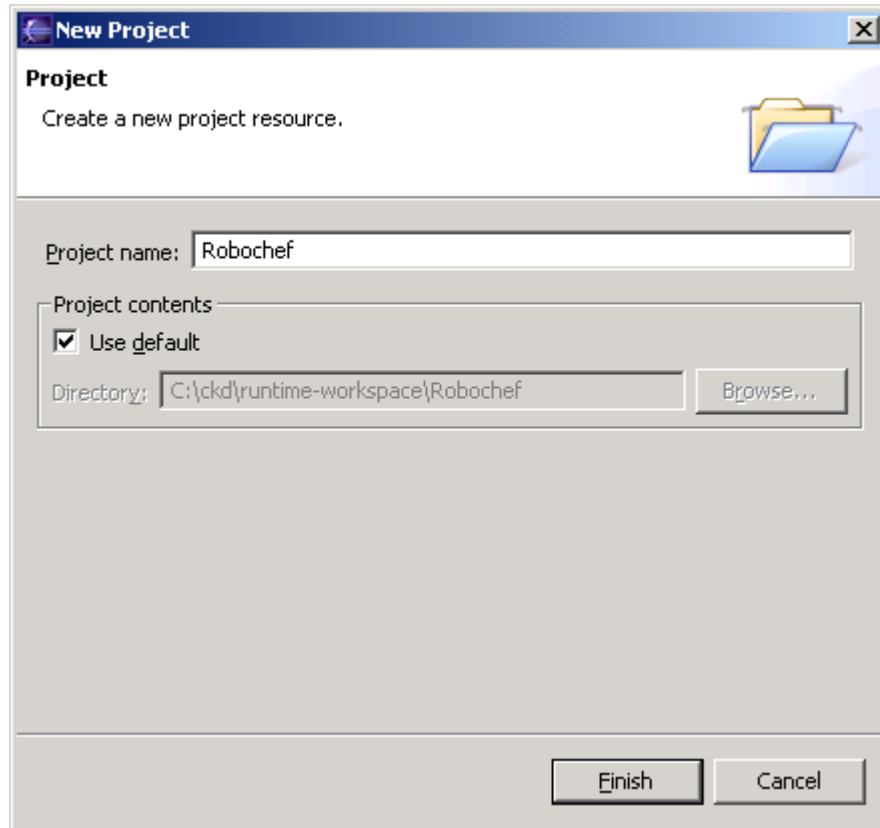
**Figure 8: PathMATE Transform Wizard Before Creating Project**

To use an existing Eclipse project, select the project from the tree control.

To create a new Eclipse project, press the New Project button. The New Simple Project Wizard appears. See Figure 9. Enter the name of the project and press the Finish button.

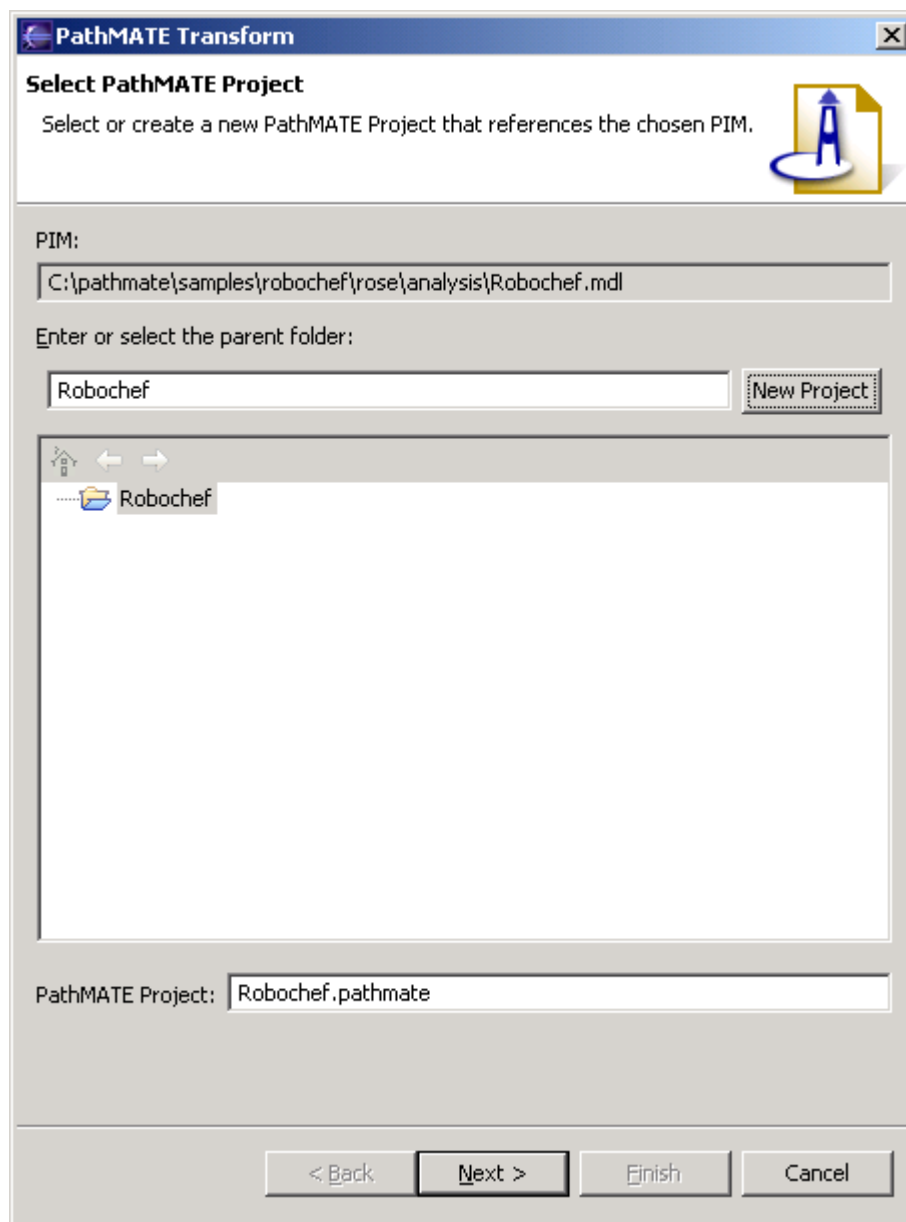


NOTE: If you want to create a different type of project, Cancel the PathMATE Transform wizard. Create the project. Then select Tools->PathMATE Transform from Rose again.



**Figure 9: New Simple Project Wizard**

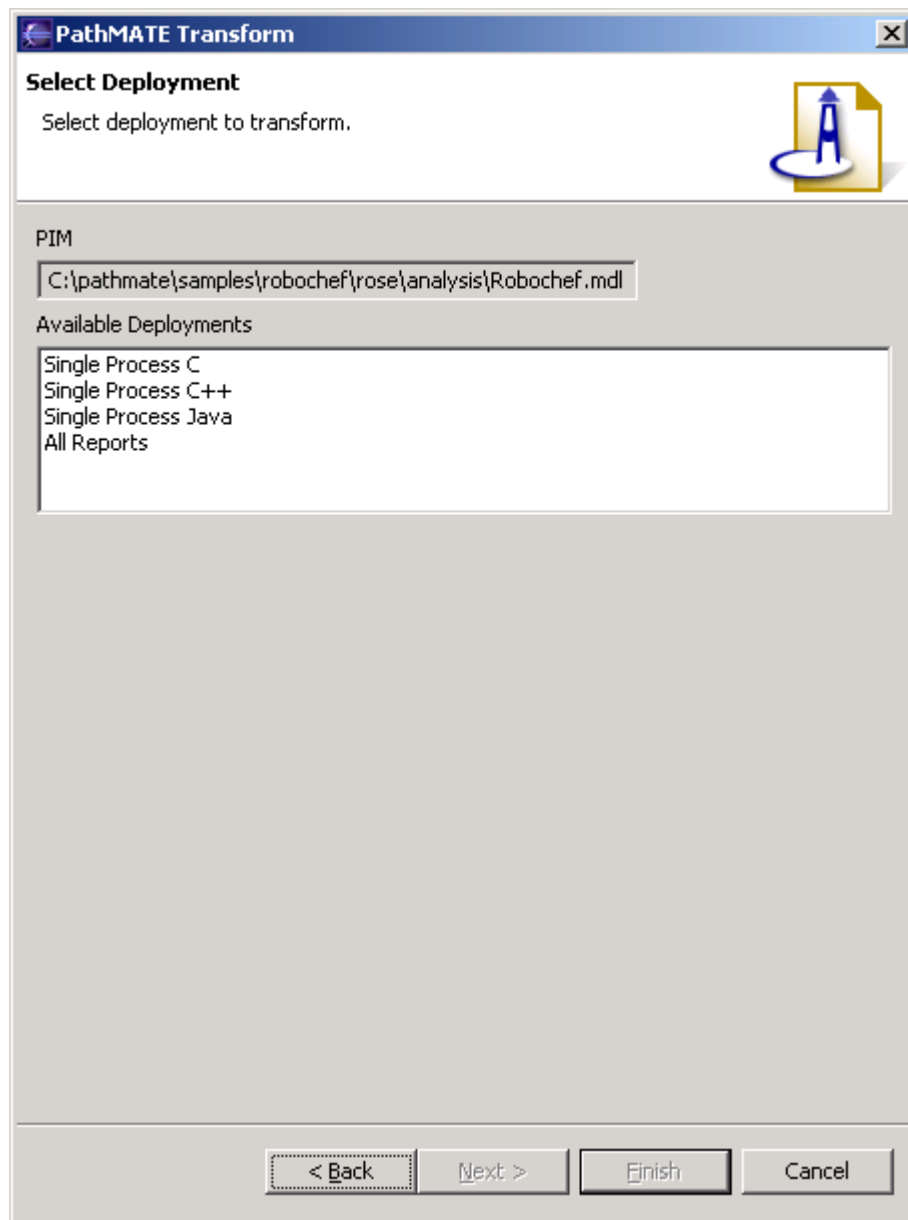
Any newly created project appears selected in the PathMATE Transform wizard. See Figure 10.



**Figure 10: PathMATE Transform Wizard After Adding Project**

Enter the name of a new PathMATE project or select an existing PathMATE Project from the tree control.

Press the Next button. The Select Deployment page shown in Figure 11 will appear:



**Figure 11: Select Deployment Page**

Select a Deployment and press the Finish button. If the PathMATE Project does not exist, it will be created. The PathMATE Project file opens in Eclipse and the Transformation begins. You will see the Progress dialog as the Transformation proceeds.

#### **Changing the Transformation Options**

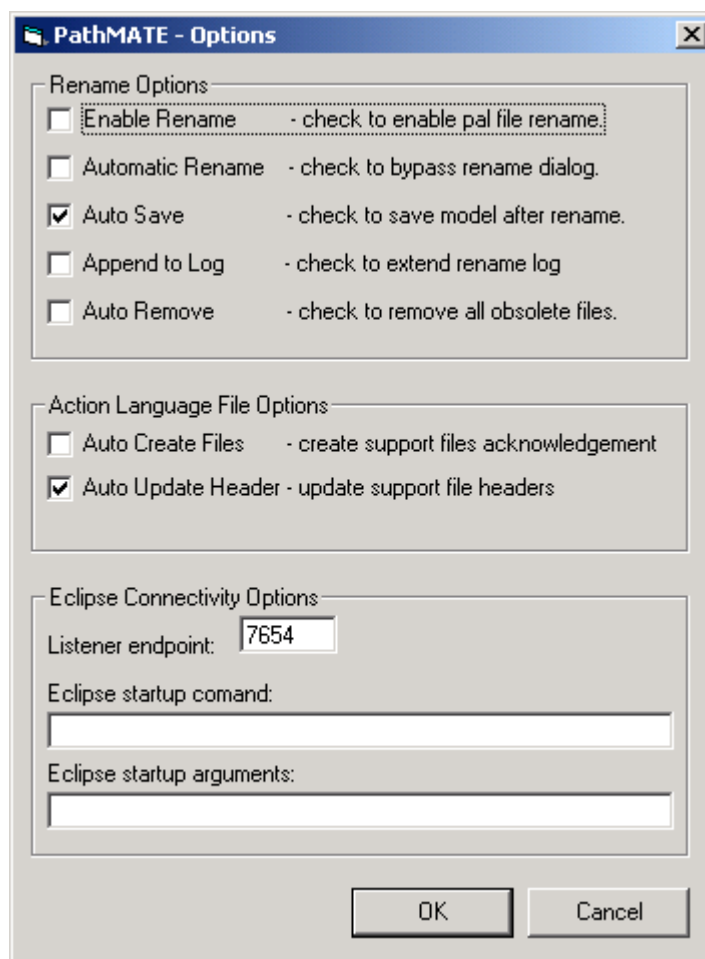
PathMATE remembers your selections so the next time you invoke a transformation from Rose, PathMATE will use the same PathMATE Project file.

To switch to a different PathMATE Project, select the Tools -> PathMATE Transformation Options menu. The PathMATE Transform Wizard will appear in Eclipse. Select the PathMATE Project file as a Deployment. The PathMATE Project will open in Eclipse. You may now make changes to the PathMATE Project file. Press the Transform button in the PathMATE Project editor to invoke the transformation.

### Setting the Connectivity Options

By default a PathMATE installation uses sockets to connect to an Eclipse platform redistributed in the PathMATE installation file on a default port. To connect to another Eclipse platform or to change ports, use the connectivity settings provided in Rose and Eclipse.

From Rose, select Tools > PathMATE Options. The PathMATE Options dialog shown in Figure 12 below will appear.



**Figure 12: PathMATE Options dialog**

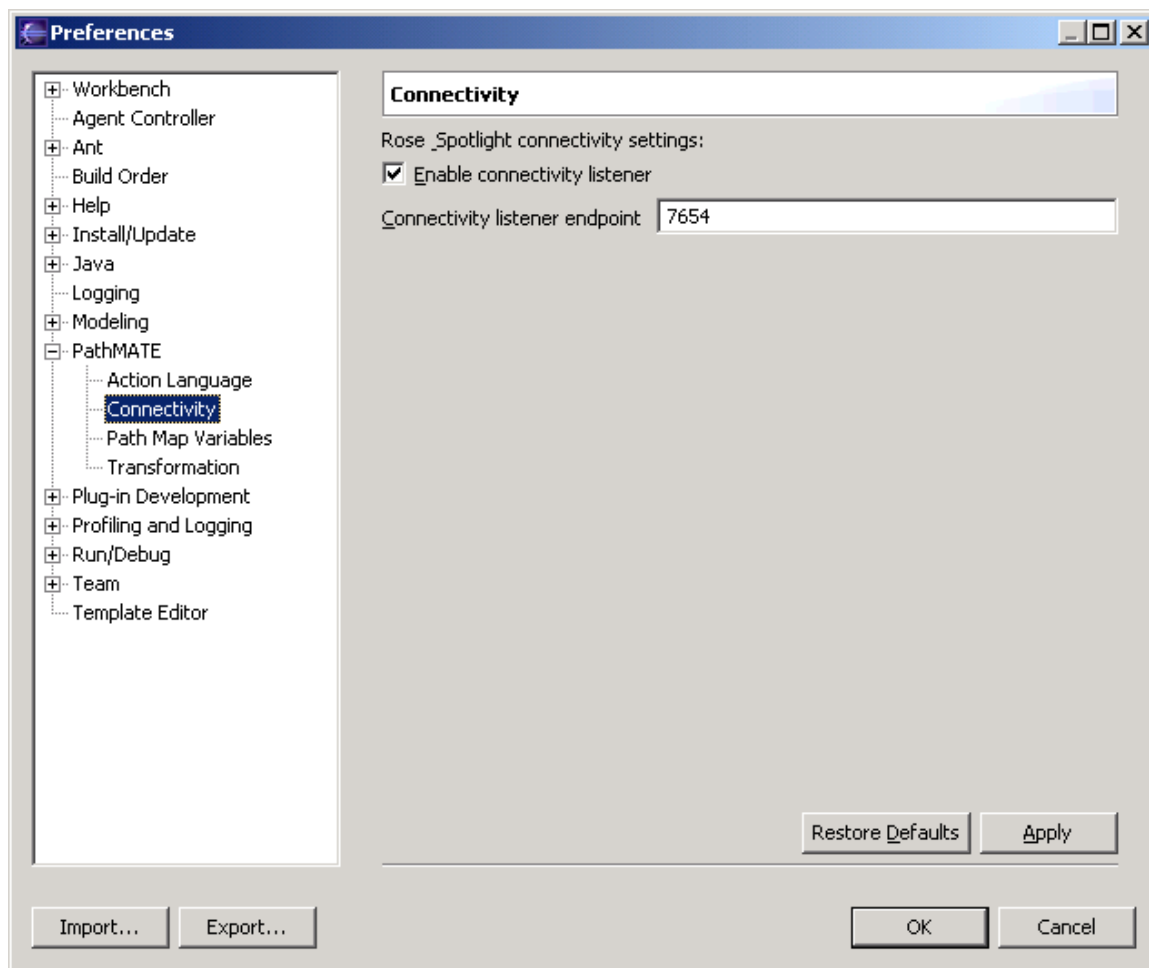
**Listener endpoint** – Specify the number of the port that the PathMATE Rose integration uses when connecting and sending messages to Eclipse. This port must match the port specified in

Connectivity listener endpoint in the Eclipse preferences shown in Figure 13.

**Eclipse startup command** – The fully qualified path to the program that starts Eclipse. If no command is specified, the Rose integration will not start Eclipse if it is not already running. In this case, the user must start eclipse manually.

**Eclipse startup arguments** – The arguments to pass to the Eclipse command when it starts. For example, this field may be used to specify the sizes of the memory used by Eclipse.

In Eclipse, select Window > Preferences. Expand the PathMATE entry in the tree on the left hand side of the dialog. Select Connectivity. See Figure 13 below.



**Figure 13: Eclipse PathMATE Connectivity Preferences**

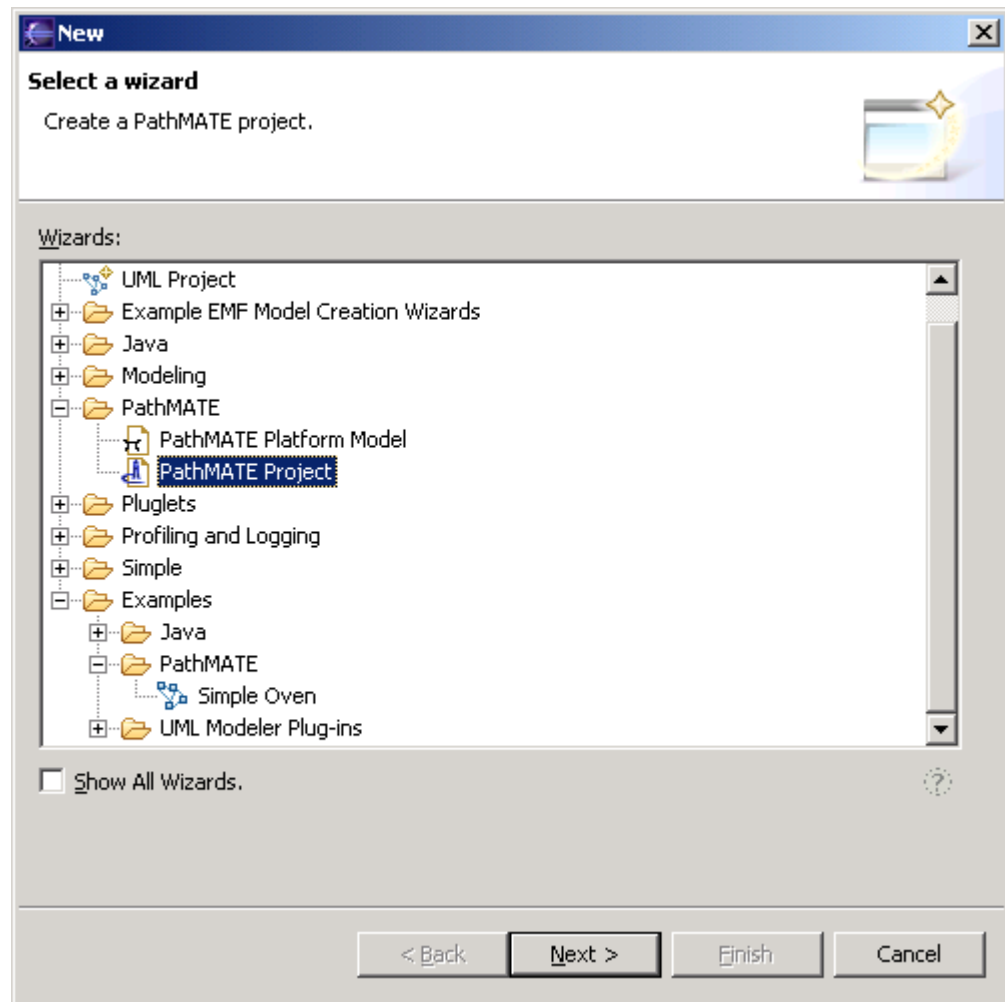
**Enable connectivity listener** – If this preference is checked, automatically listen for Rose connections upon Eclipse startup. Remove the check from this preference if you want to run more than Eclipse session. For example, if you want to run Rational Software

Modeler and the Eclipse redistribution provided with PathMATE simultaneously.

**Connectivity listener endpoint** – Specify the number of the port that the PathMATE Eclipse integration uses when listening for commands from Rose or Spotlight. This port must match the port specified in Listener endpoint in the Rose preferences shown in Figure 12 .

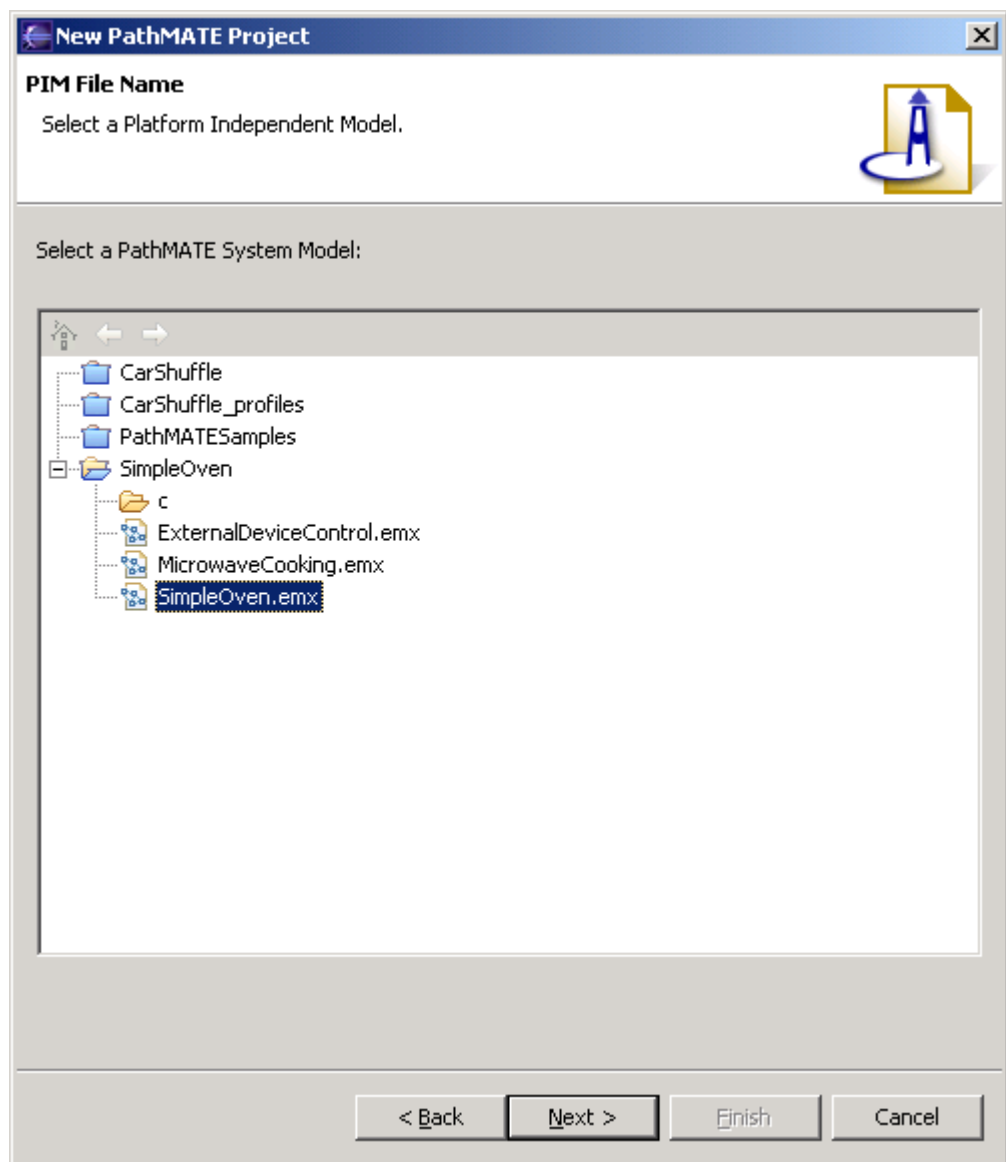
### **Creating a PathMATE Project From Eclipse**

To create a new PathMATE Project, select File -> New -> Other... The New dialog shown in Figure 14 will open.



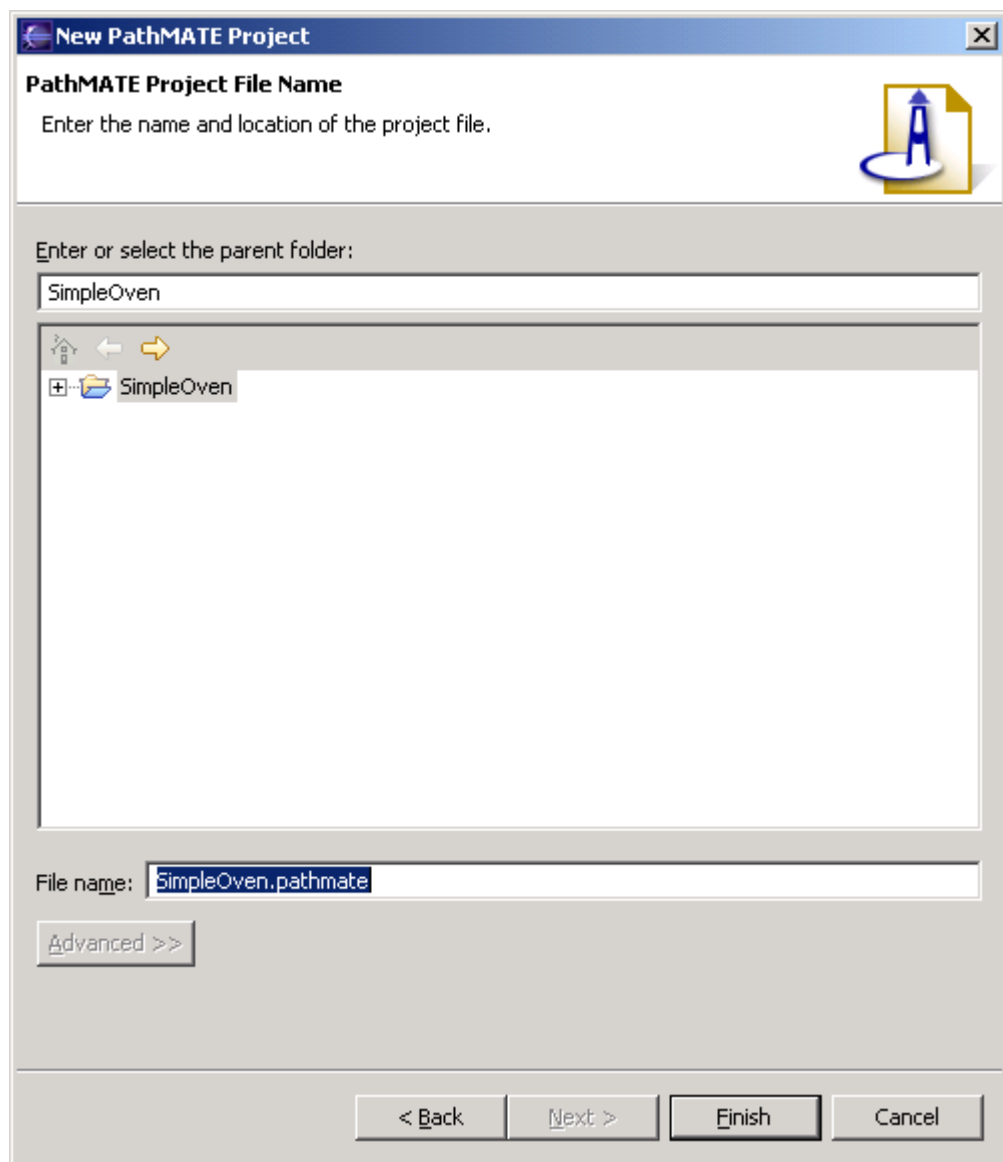
**Figure 14: New Model Wizard**

Select PathMATE -> PathMATE Project from the list. Press the Next button. The New PathMATE Project wizard will be displayed. See Figure 15.



**Figure 15: Selecting the PIM on the New PathMATE Project Wizard**

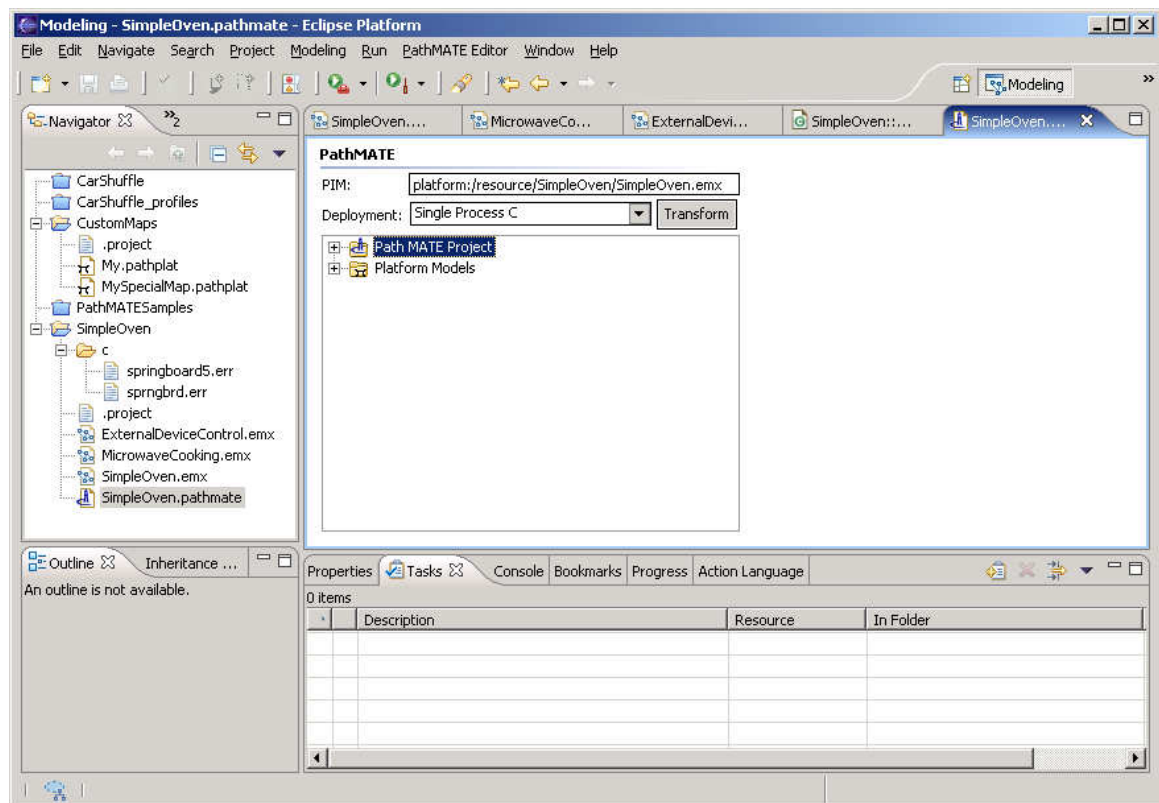
Select the Platform Independent Model file containing the PathMATE System model from the tree. Press the Next button. See Figure 15.



**Figure 16: Specifying the PathMATE Project Filename on the New PathMATE Project Wizard**

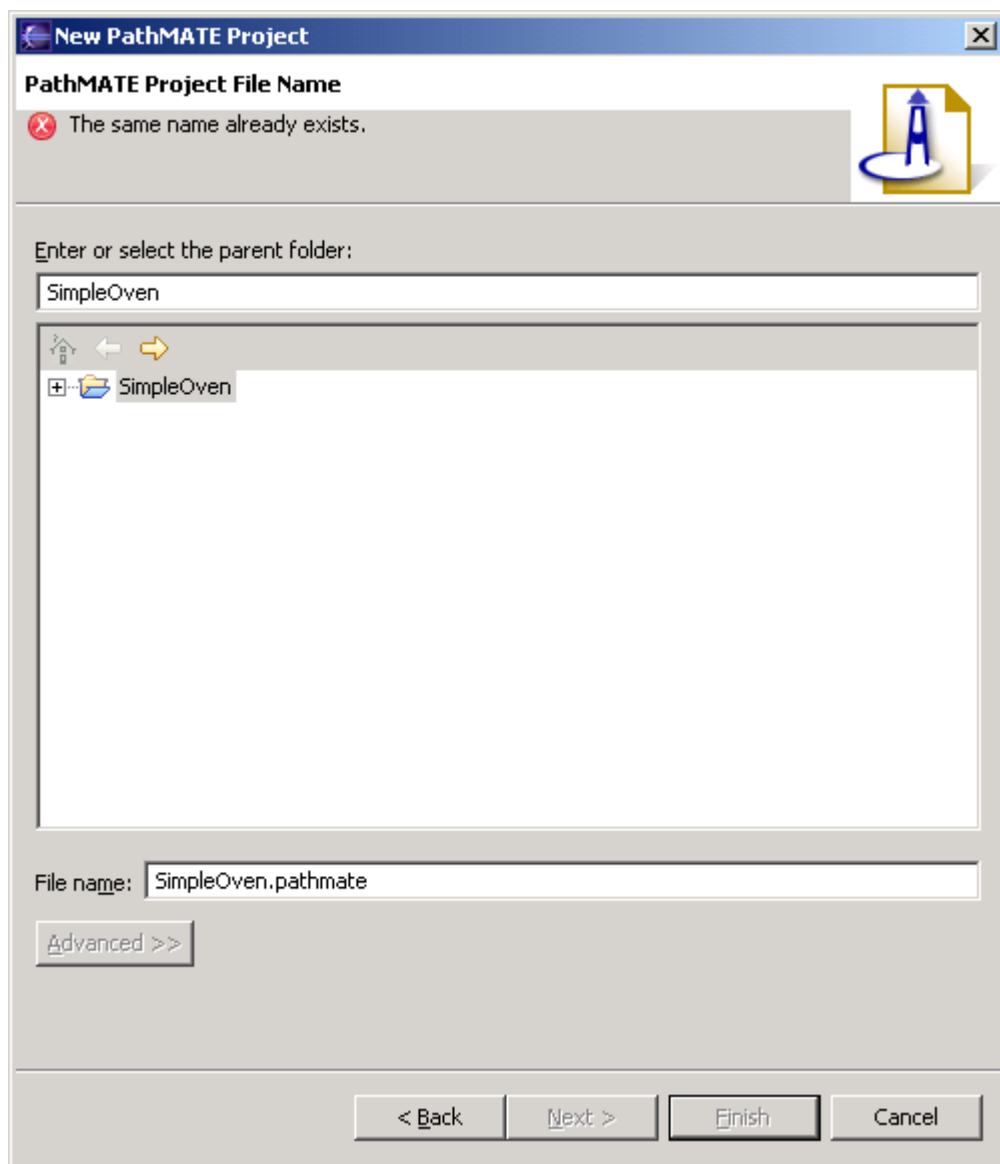
Enter the name of the PathMATE Project file in the File name field and press Finish to create the project. By default, the project file will have the same name as the selected model. Press the Finish button. The newly created PathMATE Project will open. See Figure 17.





**Figure 17: Newly Created PathMATE Project**

If the entered filename is incorrect, the top of the dialog will show an error message. The Finish button will be disabled. See Figure 18. The Finish button will be enabled when the filename error is corrected.

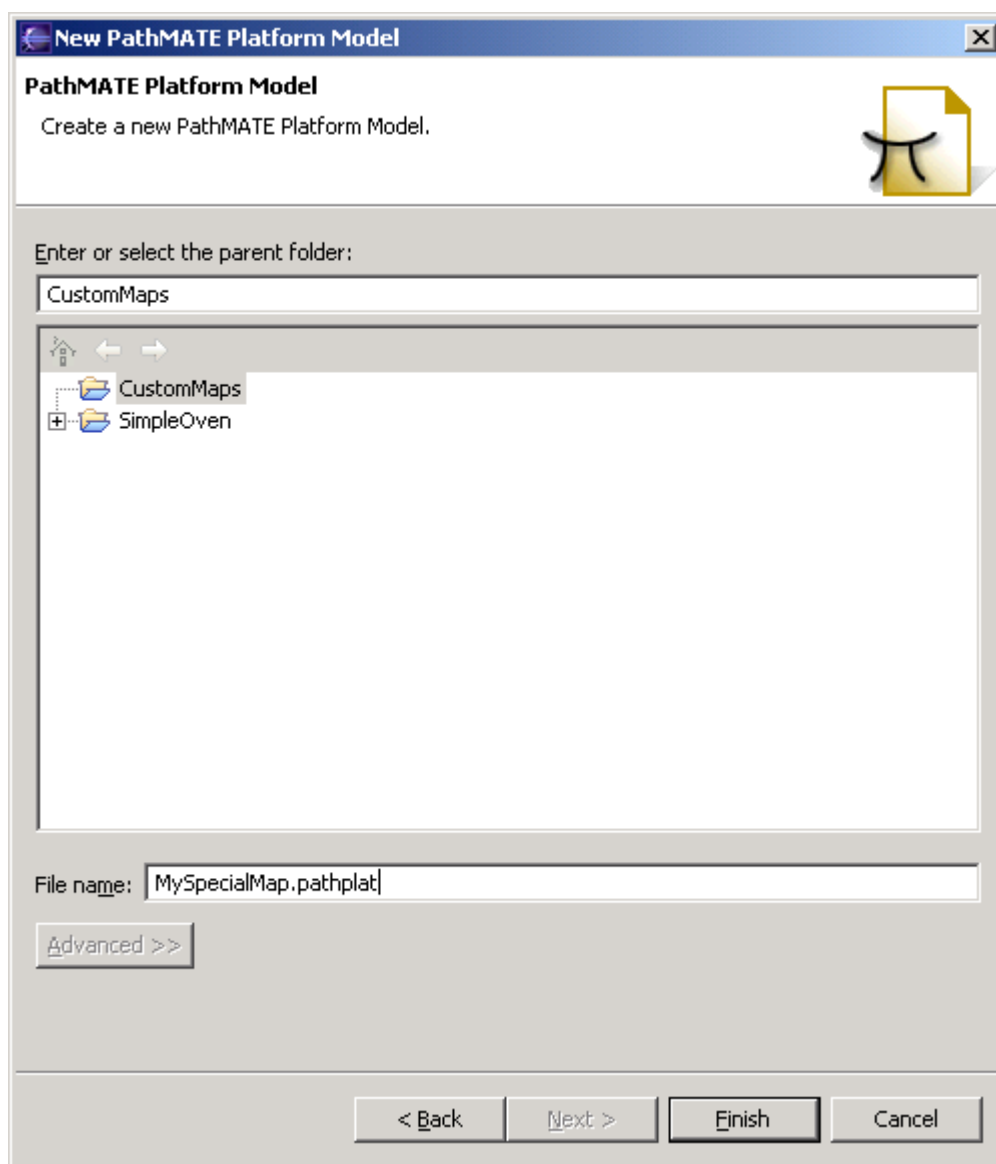


**Figure 18: Filename error in New PathMATE Project Wizard**

### ***Creating a Platform Model From Eclipse***

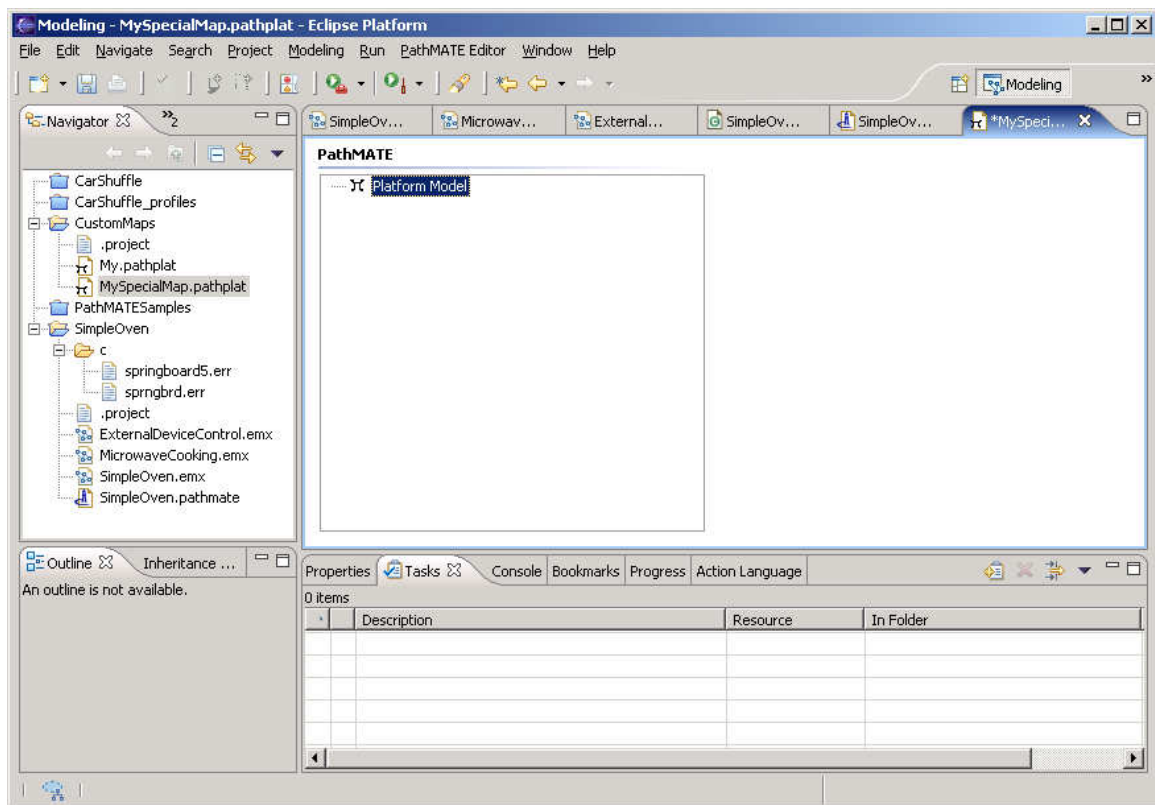
To facilitate sharing of custom Platform Models, .pathplat files should be created in a separate Eclipse project. See *Creating Custom Platform Models* and *Deploying Custom Platform Models*.

To create a new Platform Model, select File -> New -> Other... The New dialog shown in Figure 14 will open. Select PathMATE -> Platform Model from the tree. Press the Next button. The New PathMATE Platform Model wizard shown in Figure 19 will appear.



**Figure 19: New Platform Model Dialog**

Select the folder for the Platform Model from the tree. Enter a name in the File name field. Press the Finish button.



**Figure 20: Newly Created Platform Model**

### **Opening Existing PathMATE Projects and Platform Models**

Switch to the Resource perspective by selecting Window -> Open Perspective -> Other. Select Resource from the list box. Press Ok.

Go to the Navigator view and expand the project containing the PathMATE file. Double click on the file.

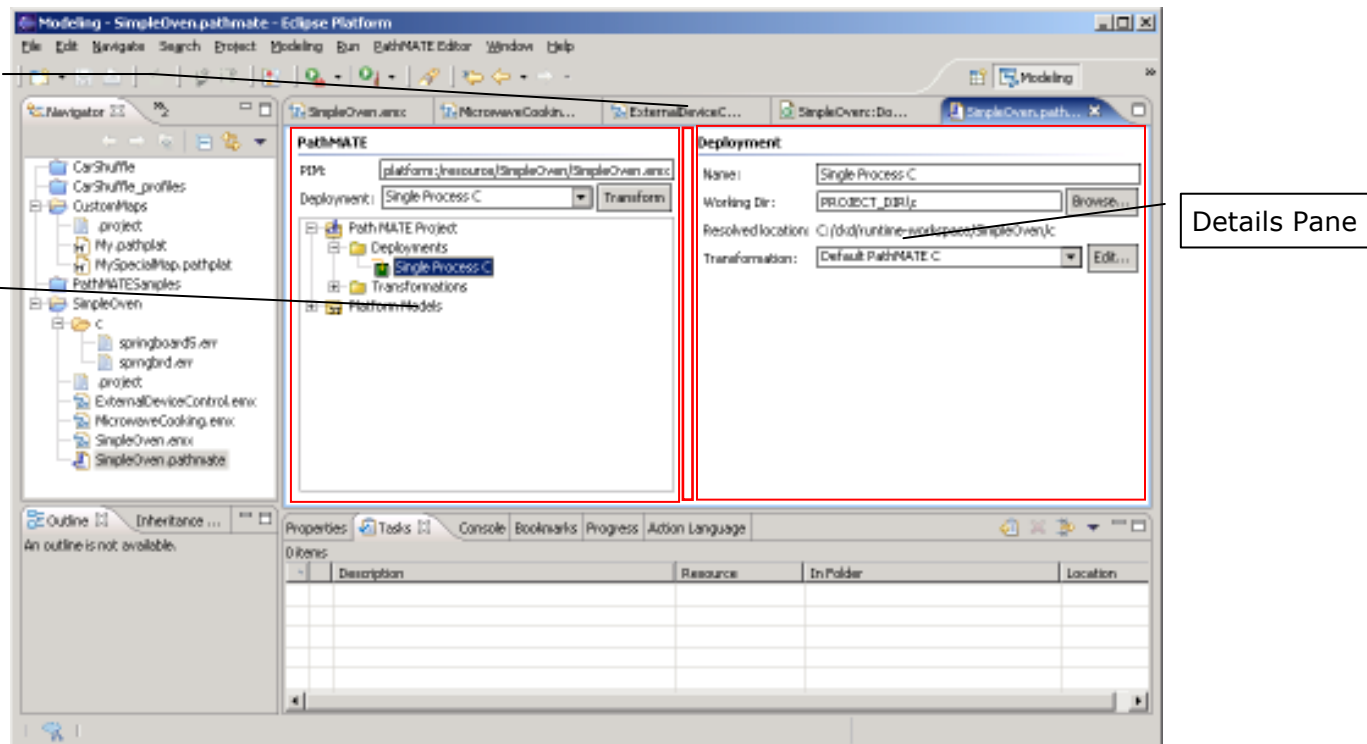
If you want to do this frequently, add the Navigator view to your Modeling perspective. Select Window->Show View->Other. Select Basic -> Navigator from the tree. Press Ok. The Navigator will open. The Navigator view appears at the bottom of the window. If you want it to appear on side of the window, press the left mouse button down on the tab containing the Navigator view name. Drag and drop the view to the desired location. Eclipse remembers custom perspectives and restores them when the restarting.

## **4. Features of the PathMATE Editor**

The PathMATE window consists of two parts divided by a splitter. See Figure 21. The left side of the editor contains the browser. The browser shows the contents of the PathMATE Project and its associated Platform Models in a tree.

The details pane on the right side of the window changes depending on what is selected in the browser.

The window splitter changes the proportions of the browser and the details pane. To change the proportions, rest the cursor over the splitter area. When the cursor is over the splitter, it will change shape to a  $\leftrightarrow$ . Press down on the left mouse button and drag the cursor to make the browser wider or narrower.



**Figure 21: PathMATE Editor Features**

## 5. Using Default Deployments

The PathMATE Transformation Maps will contribute a set of default Deployments, Transformations, and Transformation Maps when the project is created.

To use a default deployment, create a new PathMATE Project. Select the Deployment from the deployment drop down and press the Transform button.

## 6. Creating Deployments

### Starting from an Existing Deployment

To create a new deployment by copying an existing deployment, select the deployment to be copied from the browser. Right click and select

Copy. Select the Deployments folder and press the Paste button. The copy will be selected and will appear in the details pane.

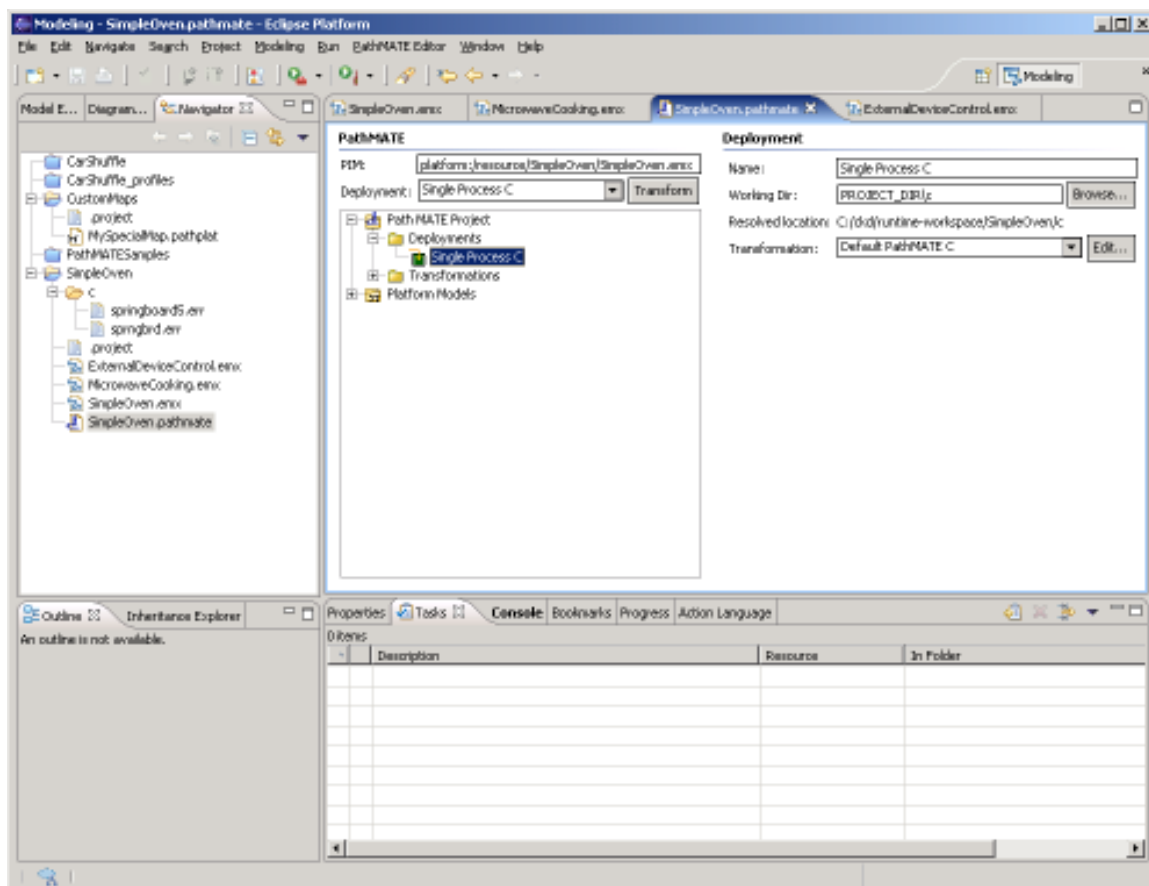
Change the name of the new deployment copy in the details pane.

## Creating a New Deployment

Select the Deployments folder from the browser. Right click and select New Child -> Deployment. The new deployment will be selected in the browser and the details will show in the detail pane.

## Editing a Deployment

Select a deployment from the browser. The details pane will show the properties of the deployment. See Figure 22.



**Figure 22: Deployment Details Pane**

## Changing the Deployment Name

Specify the deployment name in the Name field. Deployment names must be at least one character long and must be unique amongst all the deployments in the PathMATE Project.

### ***Specifying the Working Directory***

The working directory and its child directories holds the documents produced during the transformation, the properties.txt file, and initial instance population comma separated value file.

Enter the Working Directory in the Working Dir field.

Alternatively, press the Browse button to display the Select Directory dialog. The Select Directory Dialog allows the user to specify an absolute path or a path based on variables. See section Selecting Directories for more information.

### ***Selecting a Transformation***

Select the transformation used to produce target documents from the drop down list. See Creating Transformations for instructions on how to create transformations.

## **7. Creating Transformations**

### **Starting from an Existing Transformation**

To create a new transformation by copying an existing transformation, select the transformation to be copied from the browser. Right click and select Copy. Select the Transformations folder and press the Paste button. The copy will be selected and will appear in the details pane.

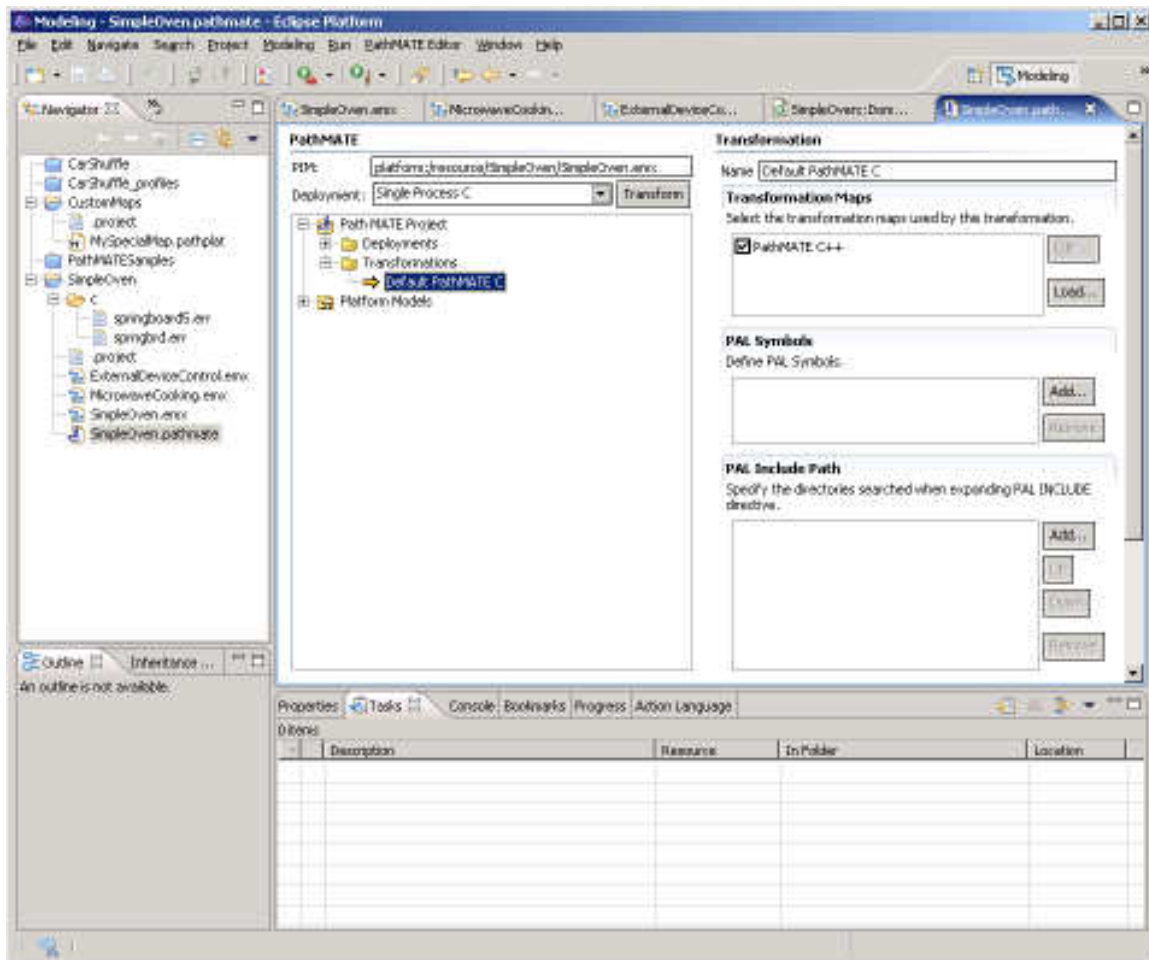
Change the name of the new transformation copy in the details pane.

### **Creating a New Transformation**

Select the Transformations folder from the browser. Right click and select New Child -> Transformation. The new transformation will be selected in the browser and the details will show in the detail pane.

### **Editing a Transformation**

Select a transformation from the browser. The details pane will show the properties of the transformation.



**Figure 23: Transformation Details Pane**

### **Changing the Transformation Name**

Specify the Transformation name in the Name field. Transformation names must be at least one character long and must be unique amongst all the Transformations the PathMATE Project.

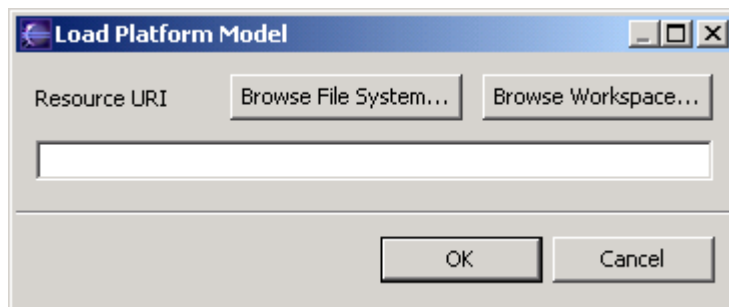
### **Selecting the Transformation Maps**

When a Transformation is run, it produces documents using one or more Transformation Maps. Check all the Transformation Maps used by this transformation.

A Transformation Map name may appear in the list of Transformation Maps more than once if two different platform models loaded into the editor contain Transformation Maps with the same name. To distinguish between the two maps, select the map name from the list and press Edit. The detail pane will show the selected transformation. The browser will show the owning Platform Model.

To add a new Transformation Map to the check list, press the Load... button. The Load Platform Model dialog shown in Figure 24 will be displayed.

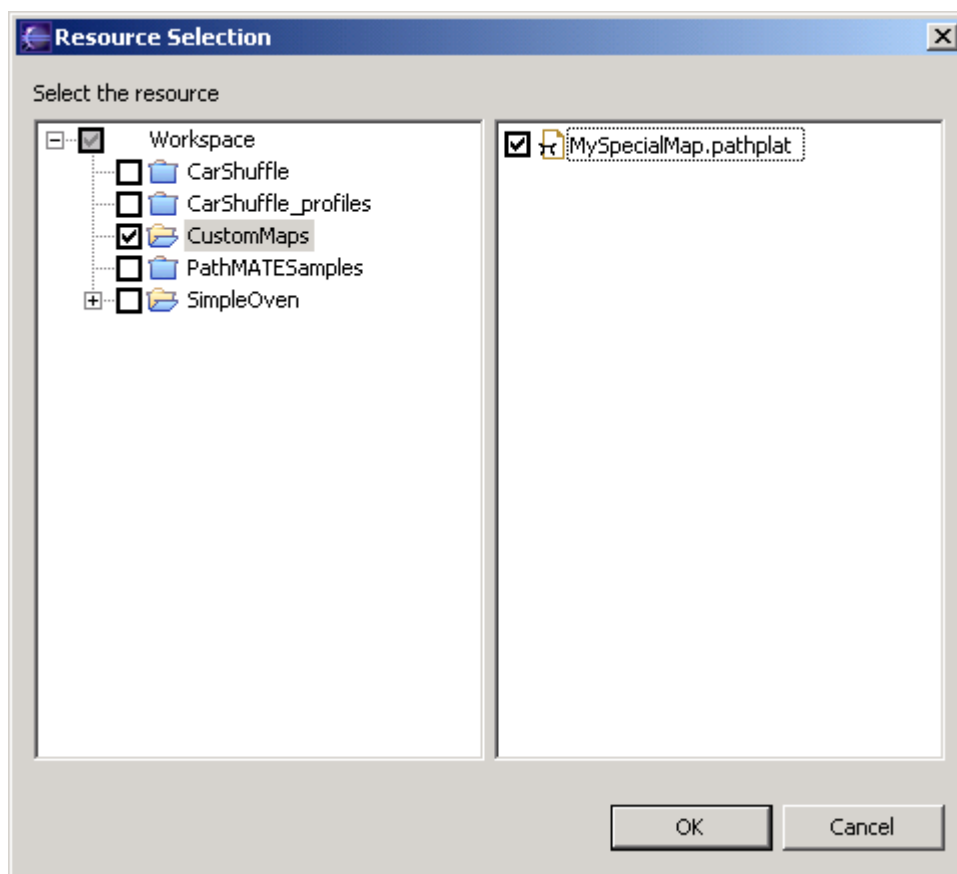




**Figure 24: Load Platform Model Dialog**

To load a Platform Model located anywhere on the file system, press the Browse File System button. The File Dialog will appear. Select a .pathplat file. Press the Ok button. The URI of the Platform Model will appear in the Load Platform Model dialog.

To load a Platform Model stored in the Eclipse workspace, select the Browse Workspace button. The Resource Selection dialog shown in Figure 25 will open.



**Figure 25: Selecting a Platform Model in the Eclipse Workspace**

Check one or more Platform Models to load. Press the Ok button. The URI of the platform files will appear in the Load Platform Model Dialog.

After selecting the Platform Models to load, press the Ok button on the Load Platform Model Dialog.

The Platform Models will be loaded into the browser under the Platform Models folder. Any Transformation Maps contained in the Platform Models will appear in the check list under the Transformation detail pane.

### ***Navigating to a Transformation Map***

To view or edit a Transformation Map while in the Transformation details pane, select a Transformation Map from the list. Press the Edit... button. The Transformation Map will open in the details pane.

### ***Defining PAL Symbols***

If you don't use the #IFDEF PAL directive in your PIM, skip this section.

To define a PAL Preprocessor symbol, press the Add... button in the PAL Symbols section of the details pane. A new define with a default name will be added to the PAL Symbol definitions.

To change the name of the define, click on the define. The define gains focus and the whole name of the define is highlighted. Click one more time and the insertion cursor will appear. Change the name of the define using the keyboard and mouse.

To remove a define, select the define and press the Remove button.

### ***Defining PAL Include Path***

If you don't use the #INCLUDE PAL directive in your PIM, skip this section.

To add a new directory to the search path that the Transformation Engine will use when expanding PAL #INCLUDE directives, press the Add... button in the PAL Include Path section. The Select Directory Dialog will appear. See Selecting Directories for more information.

To remove a path from the remove PAL include path, select the directory and press the remove button.

To change the order of the directories in the path, select the directory and press the up or down button. The order of the directories in the path defines the precedence. When locating an include file the directories are searched starting at the top until the file is found.

To view the paths with path variables resolved, check Show resolved path names, on the PathMATE -> Path Map Variables preference page. See Editing Preferences for more information.

## **8. Transforming PIMs into Implementation Code**

To transform a PIM into a PSM, select a Deployment from the Deployments combo above the browser. Press the Transform button.

The Progress dialog will appear. The following phases of transformation will occur:

1. Validate the selected Deployment, the Transformation that the Deployment references, and the Transformation Maps used by that Transformation. Transformation will proceed if there are problems in elements that are not referenced by the selected Deployment.
2. Verify templates if they haven't been already or if they were modified since the last transformation or if the settings in the Transformation Map changed.
3. Load PIM.
4. Interpret templates to produce target documents.

Errors detected during transformation will appear in the Console view. Validation errors will also appear in the Problems view.

## 9. Validating PathMATE Projects and Platform Models

PathMATE will verify PathMATE Projects and Platform Models for correctness. A full set explanation of error messages and their causes may be found in Appendix A on page 60.

### Selecting the Scope of Validation

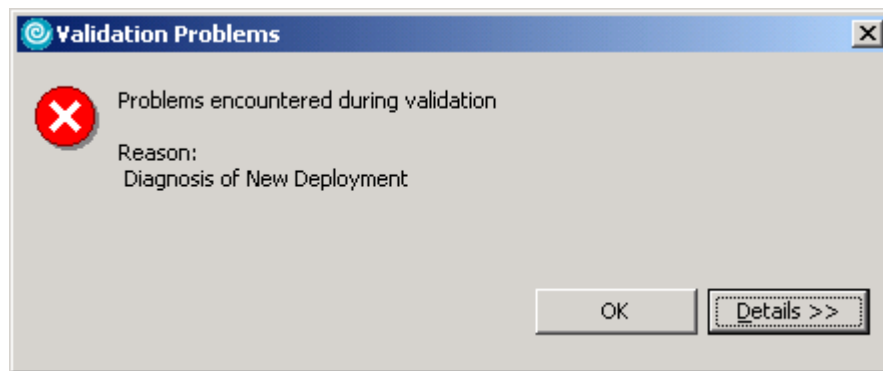
Select the element to validate from the browser. The element and all of its children will be validated. For example to validate a single Deployment or Transformation, select the Deployment or Transformation from the browser. To validate all Deployments and Transformations in a PathMATE Project, select the PathMATE Project from the browser. To validate all the Transformation Maps in a Platform Model, select the Platform Model from the browser.

### Starting Validation

After selecting the element to validate from the browser, right click and select Validate. Alternatively select PathMATE Editor > Validate from the menu bar. The progress dialog will appear briefly during the validation.

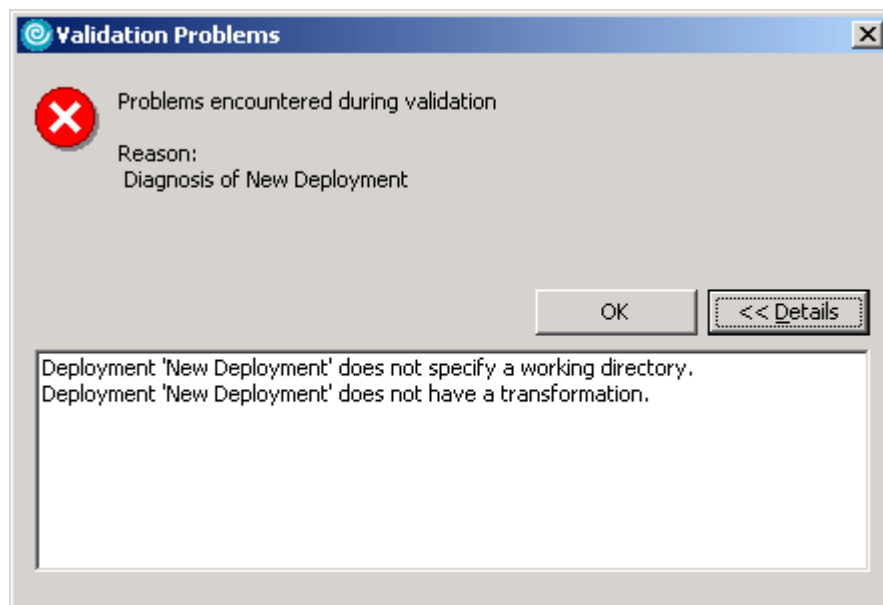
### Determining the Result of Validation

If no validation errors were found, the progress dialog is dismissed. If validation errors were found, the Validation Problems dialog will appear. See Figure 26 below.



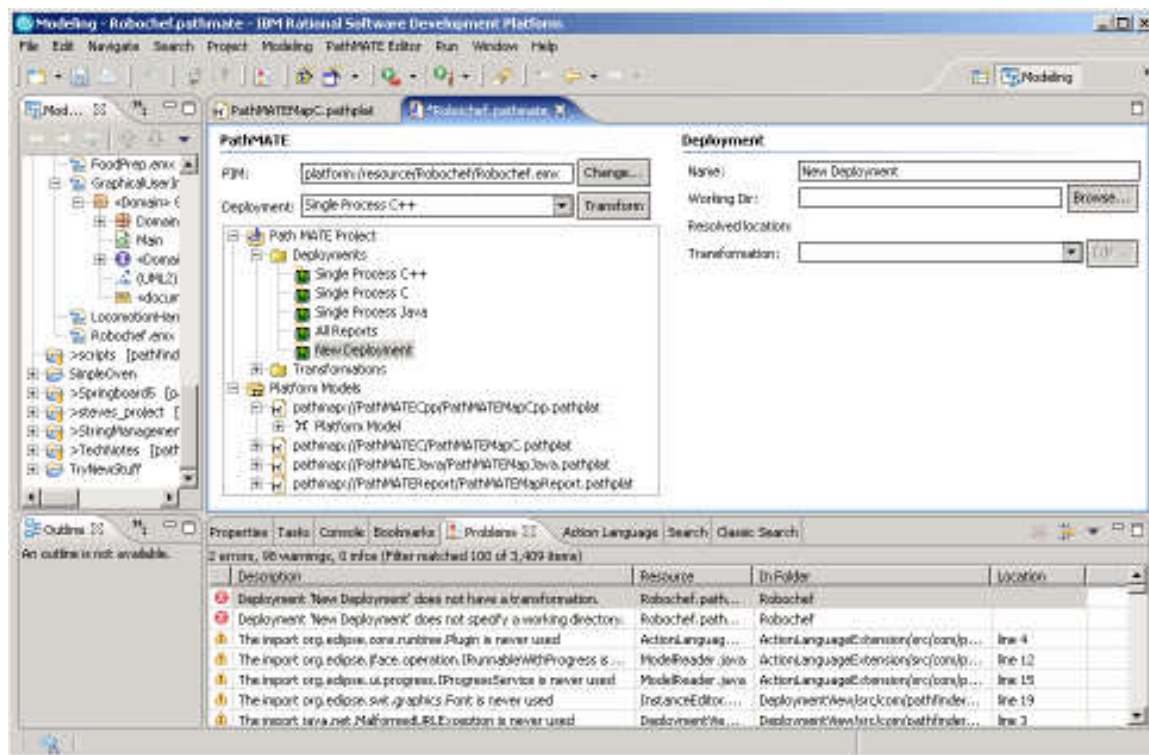
**Figure 26: Validation Problems Dialog with Details Hidden**

Press the Details button to see the error messages. See Figure 27.



**Figure 27: Validation Problems Dialog with Details Expanded**

Press the OK button to dismiss the Validation Problems dialog. The error messages appear in the Problems View. Double click on the marker in the Problems view to open the PathMATE editor to the correct element. See Figure 28. Correct the problem and then validate again.

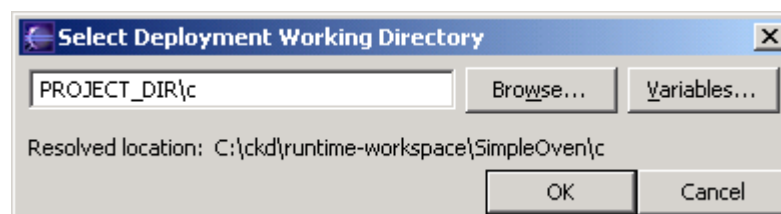


**Figure 28: PathMATE Editor with Problems View**

## 10. Selecting Directories

The Select Directory Dialog shown in Figure 29 allows the user to select a directory. The user may select an absolute path or a path based on path variables. Using path variables allows the PathMATE project to be easily moved from one location to another.

Tip: Using Variables is very similar to creating Eclipse links.

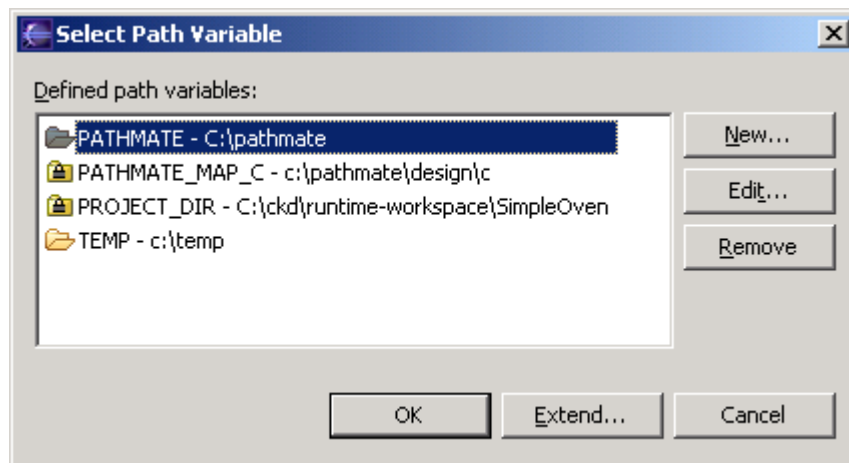


**Figure 29: Select Deployment Working Directory Dialog**

Enter a directory in the Text field or press the Browse or Variables button to use a dialog.

Press the browse button to select the directory from the Browse Folder dialog.

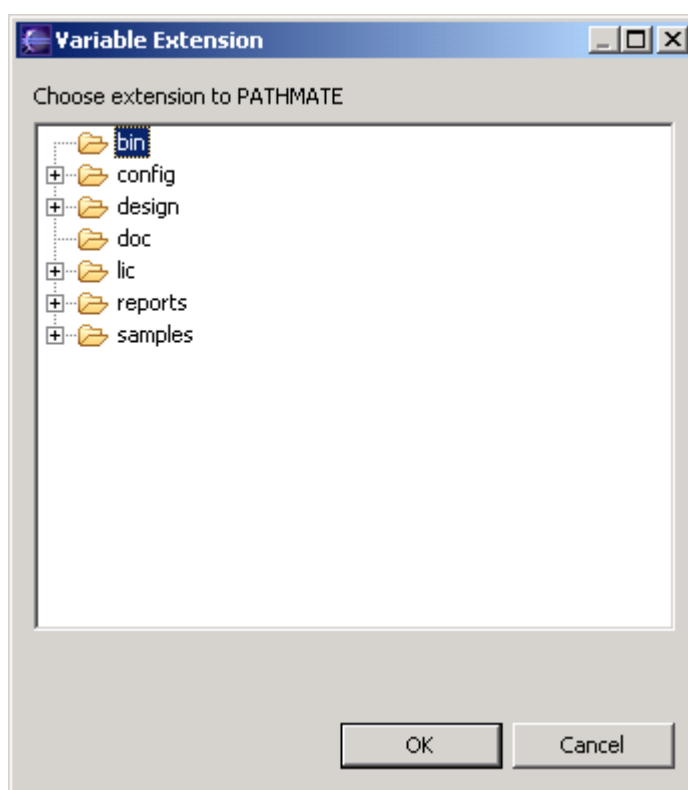
Press the Variables button to show the Select Path Variable Dialog shown in Figure 30.



**Figure 30: Select Path Variable Dialog**

The list shows the available path variables. The variables shown with open folder icons are user defined variables. Variables shown with padlocked folder icons are restricted. Restricted variables are either contributed by Platform Model plugins or are builtin variables such as the PROJECT\_DIR variable. The PROJECT\_DIR variable resolves to the folder containing the PathMATE Project.

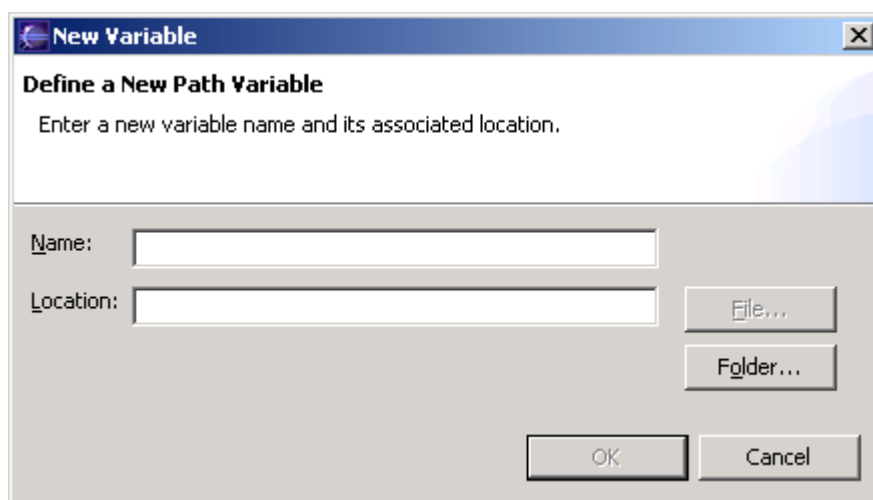
To create a directory based on an existing variable, select the variable and press the Extend... button. The Variable Extension dialog shown in Figure 31 is displayed.



**Figure 31: Extending a Variable**

The tree will contain all subdirectories of the current value of the variable. Select the appropriate subdirectory and press the Ok button. The Select Deployment Working Directory dialog will update with the new location. The resolved location shows the current location of the directory. Press the Ok button to save the location of the directory.

To create a new variable press the New button on the Select Path Variable dialog. The New Variable dialog shown in Figure 32 is displayed.



**Figure 32: New Variable Dialog**

Enter the name of the variable in the Name field. Press the Folder button and select a directory from the Folder Selection dialog.

To edit a user defined variable, select the variable from the list and press the Edit button. If a restricted variable is selected, the Edit... button is disabled. The Edit Variable dialog similar to the New Variable dialog shown in Figure 32 will appear. Select a new folder location for the variable.

To remove a variable, select the variable and press the Remove button. The variable will be removed from the list.

Tip: Variables are specific to a user rather than a project. Do not delete variables that are referenced in this project or other projects. If you make a mistake and remove a variable that is referenced by another project, you will get an error when the project is validated or transformed. When other users import your project into their workspace, they will need to define the variables your project references.

Tip: Relative paths using .. can be created by entering the path directly into the text editor field. They can't be created using the Variables or Browse buttons. For example if you wanted to specify a working directory for a deployment that was relative to the parent of the directory containing the PIM, enter the following in the text field:

```
MODEL_DIR/../../project/cpp
```

## 11. Creating Custom Platform Models

### Starting From an Existing Transformation Map

To create a new Transformation Map by copying an existing Transformation Map, select the Transformation Map to be copied from the browser. Right click and select Copy. Select the Platform Model from the browser. Right click and select Paste. The copy will be selected and will appear in the details pane.

Change the name of the new Transformation Map copy in the details pane.

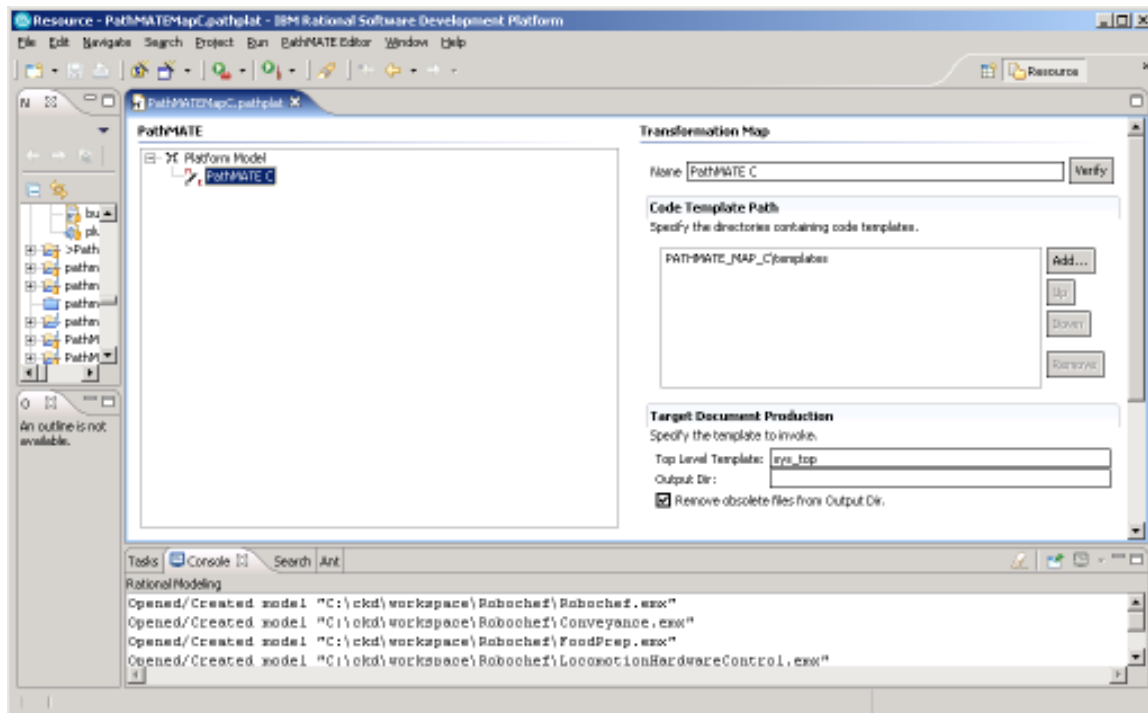
### Creating a New Transformation Map

Select the Platform Model to which the new Transformation Map will belong from the browser. Right click and press the New Child -> Transformation Map. A new Transformation Map will be created in the selected Platform Model. The new map will be open in the details pane.

### Editing a Transformation Map

Select the Transformation Map in the browser. The Transformation Map will open in the details pane. See Figure 33.





**Figure 33: Transformation Map Details Pane**

### ***Changing the Transformation Map Name***

Specify the Transformation Map name in the Name field. Transformation Map names must be at least one character long and must be unique amongst all the Transformation Maps in the Platform Model.

### ***Defining the Code Template Path***

To add a directory to the code template path, press the Add... button. The Select Directory dialog will appear. The Select Directory Dialog allows the user to specify an absolute path or a path based on variables. See section Selecting Directories for more information.

When locating a template, the engine starts searching in the directory at the top of the list. The engine continues down until it finds the template. To change the order of the code template path, select a directory and press up button to move the directory up in the path or down to move it down in the path.

To remove a directory from the path, select the directory and press the Remove button.

To view the paths with path variables resolved, check Show resolved path names, on the PathMATE -> Path Map Variables preference page. See Editing Preferences for more information.

### ***Defining the Target Document Production***

Enter the name of the top level template in the Top Level Template field. Omit the .arc or .rtf extension. The top level template file must exist in one of the directories in the code template path.

Define the path of the target documents, the output of the templates. The path will be relative to the working directory of the deployment using the Transformation Map. If you don't specify an output directory, the output will go directly into the deployment working directory.

For generated code, check the Remove obsolete files from Output Dir to keep the output directory structure in sync with the generated source tree. Checking this box ensures that stale code is not included in generated projects and makefiles.

### ***Verifying the Code Templates***

To check the syntax and semantics of the code templates, press the Verify button next to the Name field. Parse validates the Transformation Map. Any errors in the Transformation Map are reported to the Eclipse Problems view. The Console will contain a message directing the user to the Problems view. For more information about validation see If the Transformation Map passes validation, PathMATE checks the syntax and semantics of the code templates and reports any errors to the Eclipse Console view.

NOTE: PathMATE will always reparse the templates even if a cached version of the templates exists.

### ***Using the Transformation Map***

Save the Platform Model by selecting File->Save. Load the Platform Model from the Transformation details page. See Creating Transformations for more information. Alternatively you may create a plug-in to automatically load the Platform Model when a PathMATE Project is opened.

## **12. Deploying Custom Platform Models**

Custom Platform Models may be loaded automatically at startup by defining a plug-in that uses the pathmate.platformModel extension point. Create a plug-in when you have a Platform Model that will be used over and over by many users.

### **Creating an Extension Point Project**

Create a new extension point by selecting New->Other. Select the Plug-in Development -> Plugin-in Project wizard from the tree. Press Next.

Enter the Project name and press Next. By convention, plugins are named com.<company name>.<plugin name> to avoid name conflicts.

Fill in the fields on the Plug-in Content Page. Press Finish.

The plugin.xml file for the new plug-in project will open.

## Defining a Platform Model Extension Point

In plugin.xml, select the Dependencies tab. Press the Add... button and select "com.pathfinder.pathmate" from the list of plugins.

Create a new Platform Model and add it to the plug-in project. For more information see the Section Creating Custom Platform Models.

Select the Extensions tab. The New Extension dialog will appear. Select "com.pathfinder.pathmate.platformModel" from the Extension Points tab. Press the Finish button.

The new extension will appear in the list of extensions. Select pathmate.platformModel from the list of extensions. Right click and select New platform\_model.

In the platform\_model, enter the path of the .pathplat file relative to the plug-in directory.

The default\_transformation\_name, default\_deployment\_name, and default\_working\_dir fields allow the user to specify contents of PathMATE project files created by the New Wizard. If the default\_transformation\_name field is set, a Transformation with this name will be created that uses the first Transformation Map in the referenced .pathplat file. If a default\_transformation\_name is specified, a deployment with the default\_deployment\_name will be created. The working directory for the default deployment is specified in the default\_working\_dir. If no working directory is specified or no default deployment name is specified, the default deployment will not be created.

To specify an alternate working directory based on the PIM file extension, add a alt\_working\_dir child to the platform\_model. Select the Extensions tab of the plugin.xml file. Right click on (platform\_model) in the All Extensions tree control. Select New -> alt\_working\_dir from the menu. In the Extension Elements Details selection, enter the file extension of the PIM and the default working directory for the pim. For example, if Rose PIMs require a different default working directory than Rational Software Modeler PIMs, specify an alternate working directory that overrides the default working directory. If no alternate working directory is defined for the PIM extension, the default is used.

The following example specifies an alternate working directory. If the PIM extension is .mdl, the working directory is "MODEL\_DIR/../project/cpp". For all other PIM extensions, the working directory is "PROJECT\_DIR/cpp".

```
<extension
  point="com.pathfinder.pathmate.platformModel">
  <platform_model
```

```

filename="PathMATEMapCpp.pathplat"
default_transformation_name="Default C++"
default_working_dir="PROJECT_DIR/cpp"
default_deployment_name="Single Process C++">
<alt_working_dir
    pim_extension="mdl"
    working_dir="MODEL_DIR/../project/cpp"/>
</platform_model>
<pathmap
    name="PATHMATE_MAP_CPP"
    value="c:\pathmate\design\cpp">
</pathmap>
</extension>

```

To facilitate relocating the templates, define path variables for each directory used in the code template path. Select pathmate.PlatformModel from the list of extensions. Right click and select New->pathmap. Enter the name of the path variable in the name field and the directory that it maps to in the value field. The variable will appear as a restricted variable in the PathMATE preferences dialog when the plug-in is deployed. See Editing Preferences.

To make it easier to rename your plug-in, define a URI mapping. The URI mapping will use a pathmap scheme when saving references to your map. If no uri mapping is defined, the file will be referenced with a URI that looks like this:

platform:/plugin/<plugin name>

If the plug-in name providing the map changes, all references to this plug-in will be broken.

To define a URI mapping, go to the plugin.xml tab in the plugin.xml file. Add the following text above the last line of the file containing </plugin>.

```

<extension
    point="org.eclipse.emf.ecore.uri_mapping">
    <mapping source="pathmap://<your variable name>/"
target="platform:/plugin/<your plugin name>/" />
</extension>

```

## Building the Plug-in

Select the Build tab of plugin.xml. In the Binary Build section, check the Platform Model file.

To export the plug-in select File->Export. Select Deployable plug-ins and fragments. Press the Next button.

Check your plug-in in the Available Plug-ins and Fragements list. Select a directory from Export Options. Enter a directory in the Destination section. Press the Finish button.

The progress dialog will appear while exporting. If there are errors an error dialog will appear. Otherwise, your plug-in exported properly.

## Installing the Plug-in

1. If Rational Software Modeler/Architect is running, close it.
2. Open any text editor and create a file named:

```
C:\Program Files\ibm\Rational\SDP\6.0\  
eclipse\links\  
<your plugin id>.link
```

Add the following text to the file:

```
path=<plugin contents directory>
```

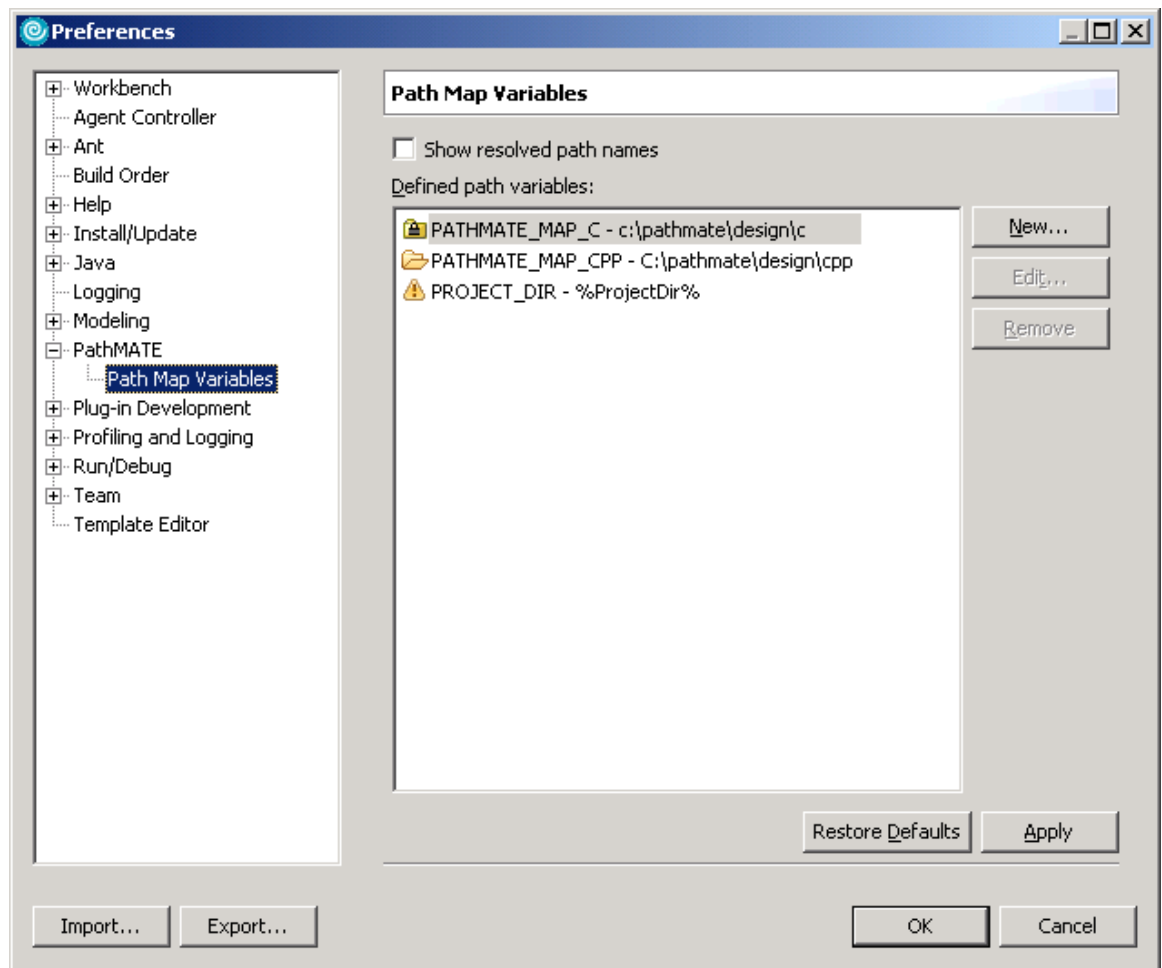
NOTE: <plugin contents directory> refers to the directory to which the plug-in was exported. Do not include the plugin directory in the path.

3. Cleanup any old versions of your plugins. Remove any .link files and their associated directories of old versions of your plugins.
4. Open a command prompt and start rsa cleanly. You only need to do this once to remove old versions of your plugins and read new versions:

```
cd C:\Program Files\ibm\Rational\SDP\6.0  
rationalsdp.exe -clean
```

## 13. Editing Preferences

To edit PathMATE preferences, select the Windows->Preferences menu. The Preferences dialog is displayed. Select PathMATE -> Path Map Variables from the tree on the left side of the dialog. See Figure 34.



**Figure 34: Preferences Dialog**

## Showing/Hiding Resolved Paths

To show resolved paths for the code template and PAL include paths, check Show resolved path names. To show path names with variables, remove the check from Show resolved path names.

## Defining New Path Variables

To define a new variable press the New... button. Enter the name and folder location in the New Variable dialog. For more information about how to use variables see 10.

## Removing Path Variables

To remove a variable, select the variable and press Remove.

## 14. Automated Build

PathMATE will provide extensions of Ant tasks to transform a deployment. Ant (<http://ant.apache.org>) is an open source build tool integrated with Eclipse. Ant scripts are XML files that describe targets and the tasks necessary build the targets. Eclipse includes a context sensitive editor for Ant scripts as well as an Ant View capable of running an Ant script.

Ant may be extended to include new tasks. PathMATE will implement an Ant task to perform a transformation. The pathmate.transform task will accept the following two attributes:

project – the path of the PathMATE Project file

deployment – the name of the deployment

For example a simple Ant script to perform a transformation follows:

```
<?xml version="1.0"?>

<project name="DoATransform" default="main" basedir=".">
  <target name="main">
    <pathmate.transform project="TryNewStuff/SimpleOven.pathmate"
deployment="Single Process C++"/>
  </target>
</project>
```

The exec task calls scripts or other command line programs. The following example Ant script executes a prepare batch script before transformation and a cleaning script after transformation:

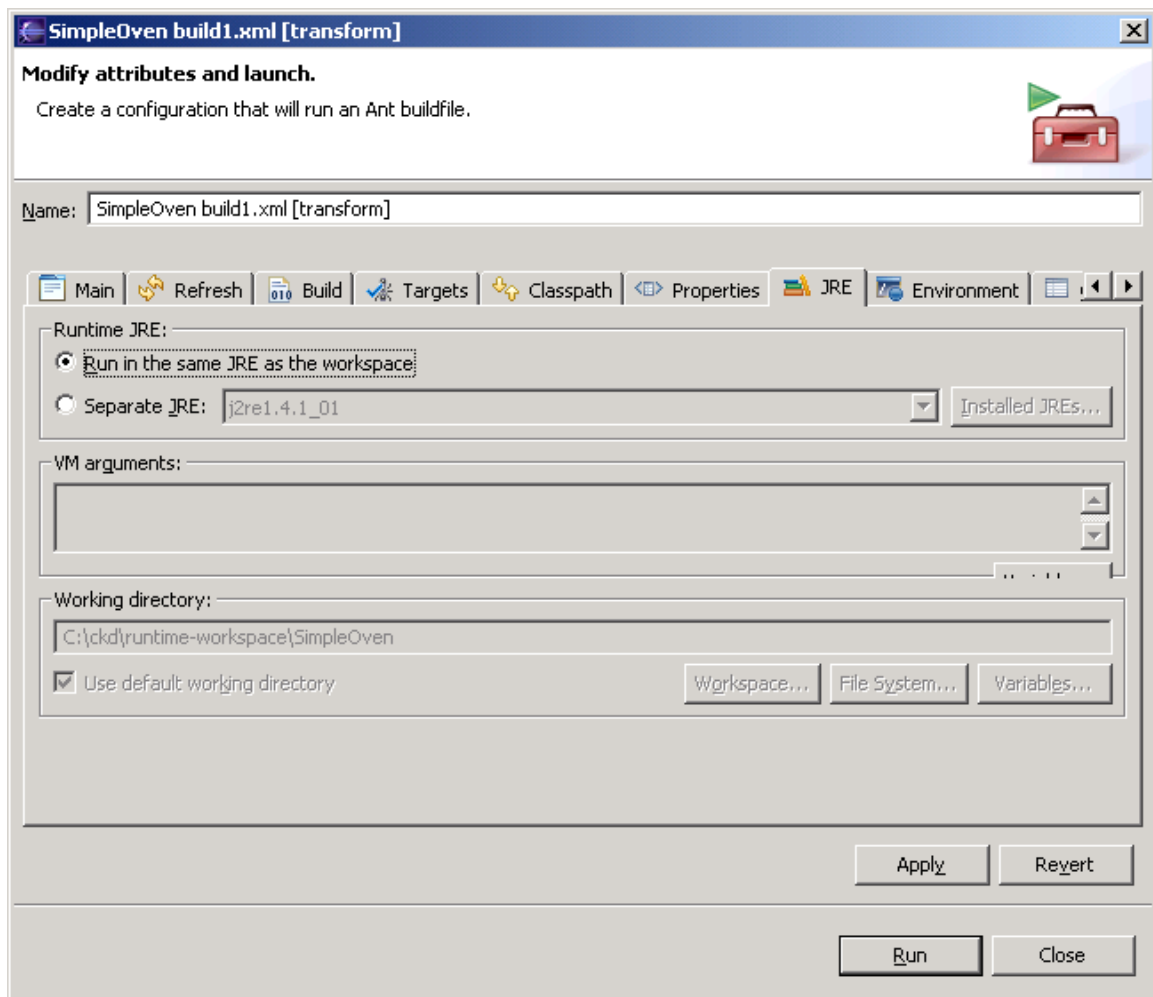
```
<?xml version="1.0"?>

<project name="DoATransform" default="main" basedir=".">
  <property name="prepare"
location="C:/ckd/workspace/TryNewStuff/prepare.bat"/>
  <property name="cleaning"
location="C:/ckd/workspace/TryNewStuff/cleaning.bat"/>

  <target name="main">
    <exec executable="{prepare}"/>
    <pathmate.transform project="TryNewStuff/SimpleOven.pathmate"
deployment="Single Process C++"/>
    <exec executable="{cleaning}"/>
  </target>
</project>
```

To run an Ant script, save the Ant script to a file called build.xml in the Eclipse workspace. Drag and drop the build.xml file onto the Ant view.

From the Ant view select a target, right click, and select Run Ant... The options dialog for the ant configuration will appear.



**Figure 35: Ant Options dialog**

Select the JRE tab. Click the Run in the same JRE as the workspace radio button. Press the Run button. The result of the script will appear in the Eclipse Console view.

To run the script again, double click on the target in the Ant view.

NOTE: Because of constraints with Rational Software Modeler, ant scripts containing the pathmate.transform task will not run in headless mode.

## 15. Phase 2

The following interfaces will be added to support the definitions of profiles, markings, and initial instance populations.

### Transformation Map



## Mechanism Path

The Transformation Map details pane will include a group for defining the code template path and mechanisms source code path. Select Code Template from the Category drop down to edit the code template path. Select Mechanisms Source Code Path from the Category drop down to edit the mechanisms path. See Figure 36.

**Transformation Map**

Name  Parse

**Paths**

Category:

Item 00
Item 10
Item 20
Item 30

Add...  
Up  
Down  
Parse  
Remove

**Target Document Production**

Specify the template to invoke.

System Top Template:

Domain Top Template:

Output Dir:

☐ Remove obsolete files from Output Dir.

**Profile**

PIM Type:

Name	Type	Default	Constraint
Item 00			
Item 10			
Item 20			
Item 30			

Add...  
Edit...  
Remove

**Figure 36: Phase 2 Transformation Map Detail Pane**

## Profile

The Transformation Map details pane will include a group for defining the profile, a set of definitions of markings used by the templates of a Transformation Map. Figure 36 shows a prototype of the interface.

To specify a mark definition, select the PIM Type (Domain, Class, Attribute, Operation, etc.) to which the mark applies from the combo box. The table specifies the name of the marking, the type of value allowed for the marking, the default value of the marking, and any

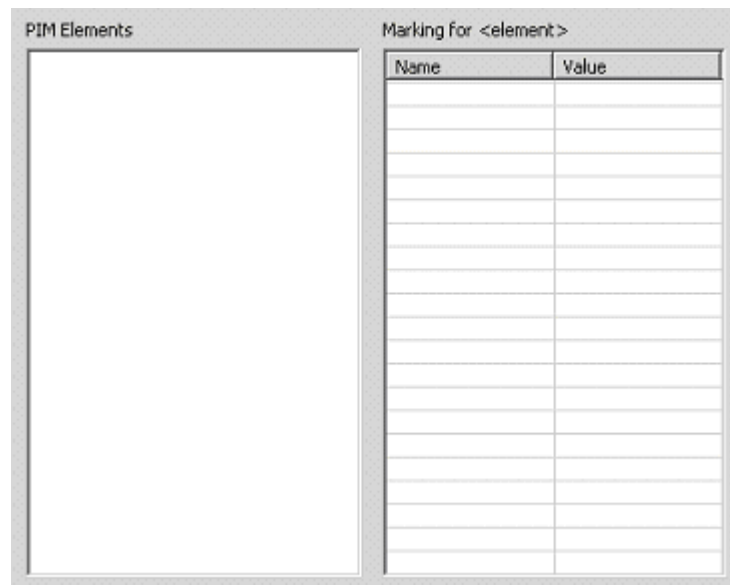
constraints. Constraints include a set of values for String markings or maximum and minimum values for numeric markings.

## Transformation

### Markings

The Transformation contains platform specific markings on elements of the PIM that augment the platform independent markings specified in the PIM. A marking may be global to all Transformations or local to a particular Transformation. Global markings may be overridden in a particular Transformation. The Markings interface shown in Figure 37 contains two panes divided by splitters so their relative size may be adjusted.

To change the markings for a Transformation, select the Transformation from the browser. Otherwise, to change the global markings, select the Transformations folder from the browser.



**Figure 37: Markings Tab**

### Locating a PIM Element

To locate a PIM element, expand the tree control in the PIM elements pane. When a PIM element is selected the Markings that apply to the type of element are shown in the Markings pane and the fully qualified name of the selected PIM element is shown in the <Element > field. The set of markings that apply to each PIM element type can be edited by editing the Map.

If a marking has never been set before, the default value appears in the Value column in gray font. If the marking has been overridden, the overridden value appears in regular font.

### Changing the Value of a Marking

Locate the Marking that you want to change. Select the cell in the value column. Either Select a new value from the dropdown list or enter a value. The new value will appear in regular font.

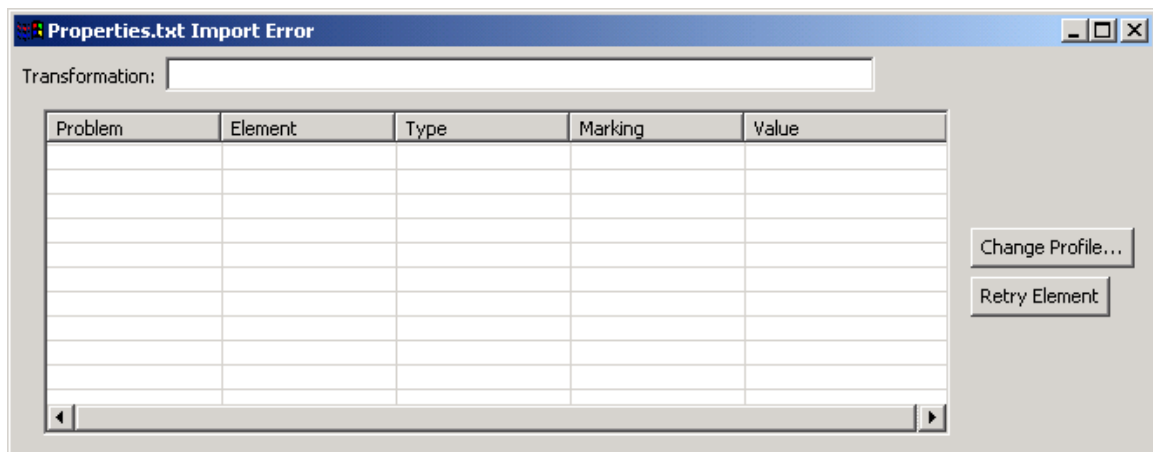
### Importing Markings from properties.txt

To import the markings specified in properties.txt, select the Transformation from the browser. Right click and select Import Markings... Select the properties.txt file from the file browser. The importer will set values of markings where the PIM element is found and the marking is defined in the profile for one of the maps used by the Transformation. After import is complete, an error dialog will show any errors encountered during import. See Figure 38.

The Transformation field will show the Transformation accepting the imported markings. The problem column indicates if the element is undefined and or if the marking is not legal for any profile for any of the maps. The Element, Type, Marking, and Value fields show the entries from the properties.txt file.

To add the markings required by the properties.txt file to the profile for one of the maps referenced by this Transformation, press Change Profile... The Change Profile dialog will appear. Select the Map to add the markings to. Select the markings to add. Press the Add button. Keep adding markings to maps until all the marks have been added to profiles. Press the Close button when complete. Any problems resolved will be removed from the table.

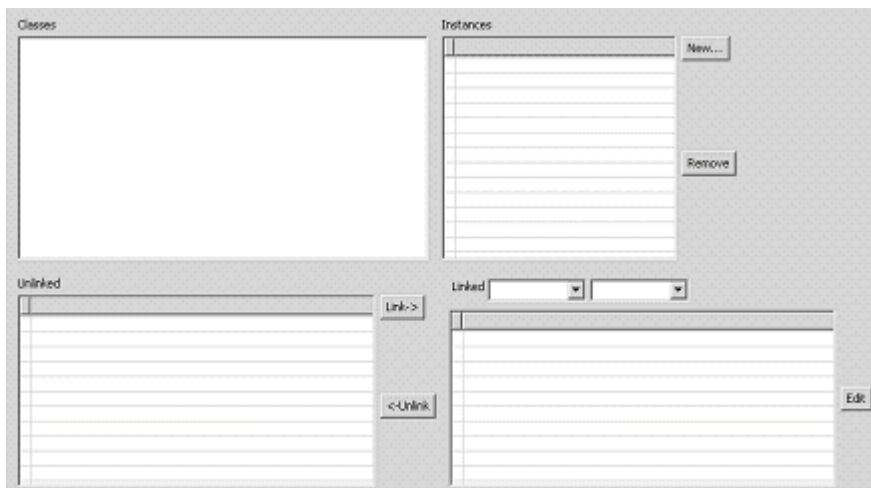
To correct an undefined PIM element, change the name of the element and press the Retry Element button. Any problems resolved will be removed from the table.



**Figure 38: Properties.txt Import Error Dialog**

## Instances

A PathMATE Project has a global population of initial instances that are available at startup. The global instance population may be expanded or overridden by a Transformation. To edit the global instance population, select the Transformations folder from the browser. To edit the instances for a particular transformation, select the transformation. The details pane will include a group for defining instances and links between instances. Figure 39 shows a prototype of the interface.



**Figure 39: Initial Instance Population Editor**

### Viewing the Instances of a Class

The Instances group contains four panes divided by splitters so their relative sizes can be adjusted.

Locate and expand the domain containing the class you want to view in the Classes tree control. Locate and expand the class whose instances you want to view in the Classes tree control. The existing instances of the class will appear in the Instances pane to right. The first column contains the identifier of the instance prefixed by an icon that indicates if the instance is a global, local or overridden instance. If Transformations is selected in the Browser, only global instances will be shown. If a Transformation is selected, global, local, and overridden instances will appear.

### Creating a Class Instance

To create a new class instance, view the instances of the class to be instantiated.

Press the New button. An empty row appears at the bottom of the instance list. Set the attribute values by typing the values into the table. Alternatively select the new instance and double click or press the Edit button. A dialog will appear that allows you to specify the values of each attribute.

### **Editing a Class Instance**

To change the attribute values of a class instance, view the instances of the class that you want to change. Select the instance that you want to edit from the Instances list. Edit the values of the attributes in the table view or alternatively press the Edit button to bring up the instance edit dialog.

If Transformations is selected in the browser, the edit will be applied to all transformations. If a transformation is selected and a global instance is edited, the overridden values appear in the table in bold font and the icon in the first column changes to overridden. The attribute value overrides will only apply to this Transformation.

### **Linking and Unlinking a Class Instance**

To link or unlink a class instance, view the instances of either of the classes participating in the association to be linked.

Select the instance that will be one of the participants in the link. Choose the association to be linked by selecting the association number from the bottom pane. The name of the other participating class will be shown in the class drop down. The instances that are linked by this association will be shown in the Linked pane. All remaining instances of that class will be shown in the Unlinked pane.

Alternatively select the name of the participant class from the class drop down list. The association list will show only the associations between the class selected in the Classes browser and the class selected in the class drop down list. If there is only one association relating the two classes the Linked and Unlinked panes are populated. If there is more than one association, select the association number to populate the Linked and Unlinked panes.

To link, select a class instance from the Unlinked pane. If the association has multiplicity 1, the Link-> button becomes selectable if the association is not linked. If the association has multiplicity many association and there are instances that are not linked, the Link-> button will be selectable. Press the Link button. The instance will be removed from the Unlinked pane and added to the Linked pane. If editing all transformations, the link will be available to all Transformations. If any other Transformation is selected, the link will be local to the Transformation.

To unlink, select a class instance from the Linked pane. Press the <-Unlink button. The instance will be added to the Unlinked pane and removed from the Linked pane.

### **Removing a Class Instance**

To remove a class instance, select the instance from the Instances pane. Then press the Remove button. To remove a global instance, select Transformations from the browser. If another Transformation is selected and a global or overridden instance is selected, the Remove button will be unselectable.

### **Importing Instance Populations**

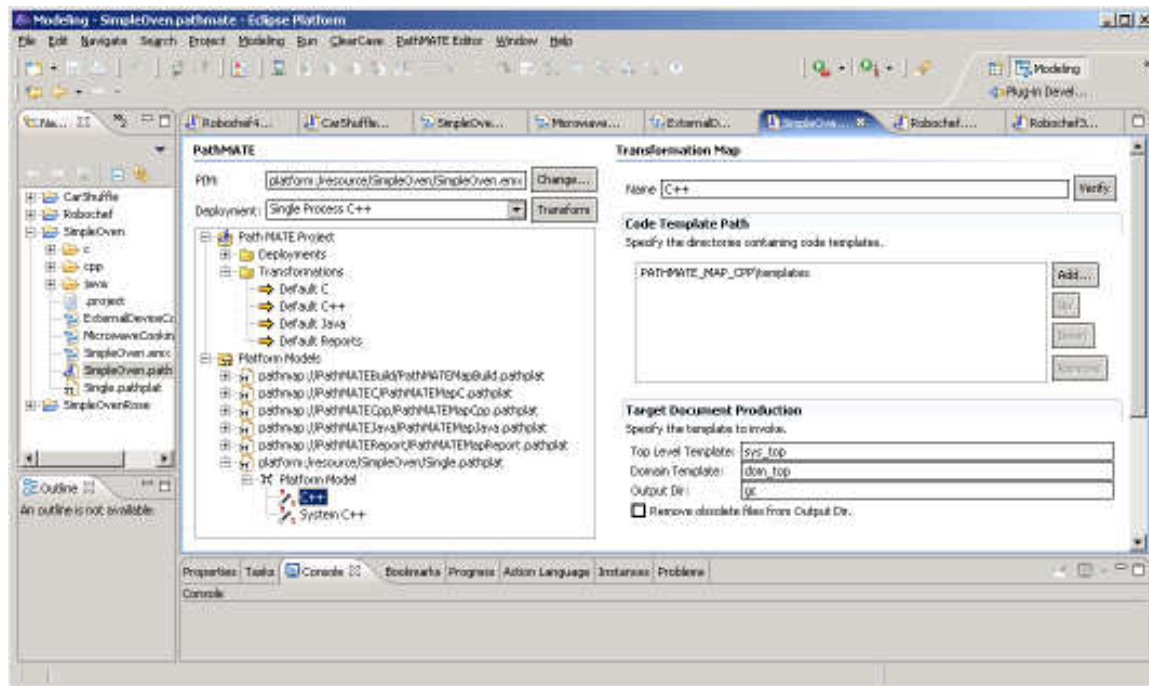
Select a Transformation from the Browser. Right click and select Import -> Instances. Select the comma separated values file containing the instances to import. Class instances and links will be created. An error dialog will show any classes, attributes or links that cannot be resolved.

## **Incremental Domain Generation**

To perform a single domain transformation in Rose, select the domain package. Right click and select PathMATE Transform. If no PathMATE Project is associated with the model containing the domain, the PathMATE Transformation Options dialog will be shown. See Figure 10 on Page 15. Select a PathMATE Project and Deployment. The single domain transformation will start and the results will appear in the Eclipse Console window.

To perform a single domain transformation in Software Modeler, select the domain package. Right click and select Transform -> PathMATE. Enter a PathMATE Project in the PathMATE Transformation configuration tab. Select a Deployment. Press the Run button. The single domain transformation will start and the results will appear in the Eclipse Console window.

In order to perform a Single Domain transformation, all maps referenced by the selected deployment must specify a value in the Domain Template field. The Domain Template is the top level template expanded when performing a single domain transformation. See Figure 40 below. If a map does not specify a value for this field it can only be used when transforming the entire system.



**Figure 40: Transformation Map with Domain Template field**

## 16. Phase 3

The Transformation Map, Transformation, and Deployment interfaces will be extended to allow the specification of deployments in the PathMATE GUI. A new Platform element is added to support defining profiles on platforms.

### Platforms

The Platform Model will support the definition of profiles on Operating Systems, Target Hardware, Build Environments, and Communication Protocols.

To create a new platform, select Platform Model from the browser. Select New Child -> Operating System, Processor, Build Environment or Communication Protocol. The new platform will be selected in the browser.

The Platform will have a Profile section similar to the Transformation Map shown in Figure 36 on page 46 will be shown. The markings defined in the Platform profile will control which markings are available on the processing elements of the deployment.

### Transformation Map

#### *Supported Platforms*

Specify which platforms are supported by a Transformation Map by selecting the map. The Supported Platforms Interface shown in Figure

41 will be shown on the Transformation Map details pane. Select the type of platform (Operating System, Target Hardware, Build Environment, or Communication Protocol) from the Platform drop down. Check the platforms that are supported. The check box lists all of the platforms defined in all the loaded platform models.

The supported platforms restrict which platforms can be selected for the processing elements of the deployment.

### Transformation Map

Name

**Paths**

Category:

Item 00  
Item 10  
Item 20  
Item 30

**Target Document Production**
Specify the template to invoke.  
System Top Template:   
Domain Top Template:   
Output Dir:   
☐ Remove obsolete files from Output Dir.

**Supported Platforms**
Platform: 

☐ Item 00  
☒ Item 10  
☐ Item 20  
☒ Item 30

**Profile**

PIM Type:

Name	Type	Default	Constraint
Item 00			
Item 10			
Item 20			
Item 30			

**Figure 41: Phase 3 Transformation Map Details Pane**

## Transformation

### PIM

The user can select the implementation of each domain included in the transformation using the PIM interface on the Transformation details pane shown in Figure 42.





To add tasks to processes, select the process, right click and select New Child -> Task.

The processing element may be edited by selecting the element in the browser. The details will show the processing element interface.

### Processing Node Detail

Figure 43 shows the detail pane when a Processing Node is selected. A Processing Node represents a machine on the network.

**Processing Node**

Name:

Working Dir:

Address:

**Operating System**

Target OS:

Mark	Value
Item 00	
Item 10	
Item 20	
Item 30	

**Target Hardware**

Target Hardware:

Mark	Value
Item 00	
Item 10	
Item 20	
Item 30	

**Allocations**

Domain	Process	Task
Item 00		
Item 10		
Item 20		
Item 30		

**Figure 43: Processing Node Details Pane**

Enter a name in the name field. The name must be unique amongst all the processing nodes in the deployment.

Enter a working directory where the code for the processing node will be generated. The working directory should be relative to the working directory for the deployment.

Enter a network address for the processing node in the Address field.

Select a target operating system from the Target OS drop down. Set any target specific markings in the table below. The markings available are defined by the profile for the selected operating system defined in the Platform Model.

Select the target hardware for the processing node from the Target Hardware drop down. Set any target specific markings in the table below. The markings available are defined by the profile for the selected target hardware defined in the Platform Model.

The Allocations section shows the which domains are allocated to the processes running on this processing node. Press Add to add another allocation. The Allocate Domain dialog will appear allowing the user to select a Domain and the Process and Task to which it is allocated. Press Edit to change the allocation. Press Remove to remove the allocation.

### Process Detail

Figure 44 shows the Process details pane. A Process represents a process running on a processing node.

**Process**

Name:

Priority:

SubAddress:

**Build Environment**

IDE:

Mark	Value
Item 00	
Item 10	
Item 20	
Item 30	

**Allocations**

Transformation:

Domain	Task
Item 00	
Item 10	
Item 20	
Item 30	

**Figure 44: Process Details Pane**

Enter a unique process name in the name field. The process name must be unique amongst all the processes on a processing node.

Select a priority for the process. The priority reflects importance of the process relative to other processes in the system. If more than one process is ready to execute, the operating system schedules the process with the highest priority.

Select the build environment used to compile and link the process from the IDE combo. Set any build environment specific markings in the table below. The markings available are defined by the profile for the selected build environment defined in the Platform Model.

Select a Transformation that will be used to generate the implementation for the process. Only the Transformations with Maps

that support the Build Environment of the process and the Target OS and Target Hardware of the parent processing node will be selectable. See Figure 41 for more information.

Once a transformation is selected, domains from the transformation may be allocated to tasks. Press Add to add another allocation. The Allocate Domain dialog will appear allowing the user to select a Domain and Task to which it is allocated. Press Edit to change the allocation. Press Remove to remove the allocation.

### Task Detail

#### Task

Name:

Priority:

#### Communication

From:

Operation	Resolution	Rule
Item 00		
Item 10		
Item 20		
Item 30		

#### Allocations

Domain	
<input type="checkbox"/> Item 00	
<input checked="" type="checkbox"/> Item 10	
<input type="checkbox"/> Item 20	
<input checked="" type="checkbox"/> Item 30	

Figure 45 shows the Task details pane. A Task represents a thread running in a process. Tasks in the same process share the same memory space.

**Task**

Name:

Priority:

**Communication**

From:

Operation	Resolution	Rule
Item 00		
Item 10		
Item 20		
Item 30		

**Allocations**

Domain	
<input type="checkbox"/> Item 00	
<input checked="" type="checkbox"/> Item 10	
<input type="checkbox"/> Item 20	
<input checked="" type="checkbox"/> Item 30	

**Figure 45: Task Details Pane**

Enter a unique name for the task in the Name field. A task name must be unique amongst all the tasks in the process.

Select a priority for the task. The priority reflects importance of the task relative to other processes in the system. If more than one task is ready to execute, the operating system schedules the task with the highest priority.

The Communication section shows a table with the resolution for all inter domain operation calls from this task. Select a domain allocated to this task from the From combo box. The table will fill up with a list of the operations called from this domain. The Resolution column shows whether the call is local, intertask, interprocess or interprocessor. If the call is statically determined at translation time, the Resolution column will show the destination task. The Rule column shows if the call resolution is static, implicitly resolved at runtime or explicitly resolved at runtime.

The Allocation section specifies which domains are allocated to this task. All the domains included in the transformation allocated to the parent process are included in the table. Check the domains allocated to this Task.

**Communication Path Detail**

Figure 46 shows the Communication Path details pane. A communication path is a network connection between two processing nodes.

**Communication Path**

To Node:

Preference:

Timeout:

☐ Force network byte ordering conversion

**Protocol**

Protocol:

Marking	Value
Item 00	
Item 10	
Item 20	
Item 30	

**Figure 46: Communication Path Details Pane**

Select the destination processing node from the To Node combo box. The combo box will only contain destination nodes where there is no existing communication path.

Enter a number ranking the preference of this communication path relative to other communication paths when using run-time resolution.

Enter the amount of time to wait for a response from this communication path in the Timeout field.

Check Force network byte ordering conversion to convert byte ordering to a network standard even when the source and destination processors use the same byte ordering.

Select the protocol used by the communication path from the Protocol dropdown. Each protocol type has a profile, a set of markings on the communication path that are recognized by the map. The markings defined by the profile will show in the table below. The default values will be shown. To change the value of a marking, select the Value column and enter the new value.

**Creating Deployments from Markings**

To create a deployment from the markings associated with a transformation, select the Transformation. Right click and select Create Deployment. PathMATE will create a system topology using the markings used in Phase 1 and 2 to specify distributed deployments.

**Engine Integration**

Engine templates will be able to access instances, the processing elements of the deployments and platforms. The Engine PROPERTY element will only read the markings for the appropriate Transformation context.

## A. Validation Error Messages

### Deployments

***Deployment '<deployment\_name>' does not specify a working directory.***

Nothing is entered in the Working Dir field for the Deployment. Press the Browse... button and select a Working Directory where the transformation runs.

***Could not create deployment working directory "<unresolved working dir>". Resolved path = "<resolved working dir>"***

The working directory specified for the deployment does not exist and can't be created. This problem could be caused by an undefined pathmap variable or because of permission problems. Define any undefined variables. Change the permissions for the working directory or select a new working directory.

***Deployment working directory '<unresolved working dir>' is not a directory. Resolved path = '<resolved working dir>'***

The working directory is a file rather than a directory. Select a new working directory or delete the file.

***Deployment working directory '<unresolved working dir>' is not readable. Resolved path = '<resolved working dir>'***

The user running the transformation does not have read permission on the working directory. Select a new working directory or change the permissions on the working directory.

***Deployment working directory '<unresolved working dir>' is not writable. Resolved path = '<resolved working dir>'***

The user running the transformation does not have write permission on the working directory. Select a new working directory or change the permissions on the working directory.

***Deployment does not have a name.***

The Name field of the deployment is empty. Enter a legal and unique Deployment name.

***Deployment name '<deployment\_name>' is not unique.***

There is more than one deployment in the same PathMATE project with the same name. Change the names of the deployments so they are distinct. Names are case sensitive.

***Deployment '<deployment\_name>' does not have a transformation.***

Every Deployment must have exactly one Transformation that determines the target documents created. Select a Transformation from the Transformation combo.

***Could not create output directory '<output dir>' when used in Deployment '<deployment name>'.***

The directory resulting from the combination of the working directory for the Deployment and the relative directory for the Transformation Map could not be created. A pathmap variable may be undefined or there may be a problem with permissions. Define any undefined variables or fix the permission problem. Alternatively, select a new relative output directory for the Transformation Map or a new working directory for the Deployment.

***Output directory '<output dir>' is not a directory when used in Deployment '<deployment name>'.***

The directory resulting from the combination of the working directory for the Deployment and the relative directory for the Transformation Map is a file rather than a directory. Either delete the file or select a new relative output directory for the Transformation Map or select a new working directory for the Deployment.

***Output directory '<output dir>' is not readable when used in Deployment '<deployment name>'.***

The user does not have permission to read the directory resulting from the combination of the working directory for the Deployment and the relative directory for the Transformation Map. Either change the permissions of the directory or select a new relative output directory for the Transformation Map or select a new working directory for the Deployment.

***Output directory '<output dir>' is not writable when used in Deployment '<deployment name>'.***

The user does not have permission to write the directory resulting from the combination of the working directory for the Deployment and the relative directory for the Transformation Map. Either change the permissions of the directory or select a new relative output directory for the Transformation Map or select a new working directory for the Deployment.

## **Transformations**

***Pal include directory '<unresolved include dir>' does not exist. Resolved path = '<resolved include dir>'***

The Pal include directory specified by the Transformation does not exist. Either create the include directory or remove the include directory from the Pal Include Directory path.

***Pal include directory '<unresolved include dir>' is not a directory. Resolved path = '<resolved include dir>'***

The Pal include directory specified by the Transformation is a file rather than a directory. Either remove the file and create a directory



in its place or remove the include directory from the Pal Include Directory path.

***Pal include directory '<unresolved include dir>' is not readable.  
Resolved path = '<resolved include dir>'***

The user does not have read permission for the pal include directory specified by the Transformation. Either modify the permissions on the include directory or remove the include directory from the Pal Include Directory path.

***Transformation '<name>' must use at least one Transformation Map***

The Transformation does not specify at least one Transformation Map that determines which code to generate. Check at least one Transformation Map on the Transformation details page.

***Transformation does not have a name.***

The name field for the Transformation is empty. Enter a unique name into the name field.

***Transformation name '<name>' is not unique.***

Two or more transformations have the same name. Modify the names of the transformations so they are all different.

***Unable to load Transformation Map '<map file and uuid>'.***

The Transformation Map selected does not exist. Check to make sure the file exists. If the file exists, the referenced Transformation Map may have been deleted. Remove the check from the unresolved map and add a check to another map.

***Pal Symbol '<symbol name>' is defined multiple times in Transformation '<name>'.***

The PAL Symbols section contains two or more definitions of the same symbol name. Remove the duplicate symbol names or change the name of the symbol.

## Transformation Maps

***Template path '<unresolved template dir>' does not exist.  
Resolved path = '<resolved template dir>'***

A directory in the template path does not exist or can't be resolved. If a variable at the beginning of the path is undefined, define the variable. If the path is properly resolved but the directory does not exist, create the directory or remove the directory for the template path.

***Template path '<unresolved template dir>' is not a directory.  
Resolved path = '<resolved template dir>'***

An entry in the template path is a file rather than a directory. Change the file to a directory or remove it from the path.

***Template path '<unresolved template dir>' is not readable.  
Resolved path = '<resolved template dir>'***

The user does not have read permission to a directory in the template path. Modify the permissions on the directory or remove the directory from the path.

***No domain top level template specified for Transformation Map  
'<map name>'***

When performing a single domain transformation, the selected deployment references a Transformation Map that doesn't have a value entered in the Domain Template field. Open the Transformation Map and enter the name of a domain level template. Alternatively use a different Transformation that doesn't reference this map or select a system scope for transformation.

## **Platform Independent Model**

***PIM '<pim path>' does not exist.***

The Platform Independent Model referenced by the PathMATE Project does not exist because it has been renamed or deleted. Press the Change button to change the path or rename the model file so it matches the model referenced by the PathMATE Project.