Pathfinder Solutions

PathMATE™ for No Magic *MagicDraw UML* version 15.5

Quick Start Guide

Version 0.1

November 20, 2008

PathMATE™ Series

Pathfinder Solutions LLC
33 Commercial Street, Suite 2
Foxboro, MA 02035 USA

www.pathfindermda.com
+1 508-568-0068

Table of Contents

Overview

This document is provided as a first step in learning how to use PathMATE with the No Magic *MagicDraw UML* environment to build platform independent models and transform them to an executable system. The instructions in this guide are detailed with screen shots and other aids to keep you moving forward quickly.

The Platform Independent MDD (PI-MDD) approach for building systems and the PathMATE environment for automating this approach are specifically intended to address the key challenges faced when building complex, high-performance systems. Unfortunately we cannot have you quickly build a highly complex system that shows off all of PathMATE's capabilities – at least not as your first step. Instead we will start you with SimpleOven. This is a very simple example system that covers some of the key elements of a PathMATE system. The goal of this Quick Start Guide is to expose you to some of the core aspects of modeling, code generation, system construction and execution as quickly as possible. After completing this Guide please explore the more sophisticated example systems provided with PathMATE, such as ExperimentControl, to gain a more complete understanding of how real-world systems are constructed.

How to Use this Guide

If you are not yet familiar with the PathMATE toolset, please continue to read this Overview section.

If you have not installed the PathMATE toolset on your computer, follow the Download PathMATE section to download the software from the PathTECH area of www.pathfindermda.com.

Continue to learn how PathMATE works by completing the walk-though tutorial beginning with the "Model a Simple Oven" section.

Whether you have created hundreds of models and systems or none at all, you can follow this tutorial. Learn quickly how to use PathMATE to develop an executable system.

Audience

The *Quick Start Guide* is for software modelers who want to learn how to design high performance and embedded systems with PathMATE. It's helpful but not essential to have some familiarity with the No Magic *MagicDraw UML* toolset.

Additional Resources

The Pathfinder Solutions website at www.pathfindermda.com offers information on products, services, and model-driven approaches and technology. After completing this walkthrough tutorial, you'll want to explore the extensive collection of whitepapers available as PDF downloads. Direct email support is available from support@pathfindermda.com.

Eclipse

Eclipse is supported by an international open source effort and is freely downloaded by thousands of professionals. PathMATE provides feature plugins to Eclipse which extend its functionality using the framework as a basis for interoperability. For further information on Eclipse, go to **www.eclipse.org**. This tutorial will often refer to common framework elements reflecting this heritage; e.g., "Eclipse Menu" or "Eclipse Toolbar". The tutorial assumes that you are familiar with basic Eclipse terminology including the specialized terms, "View", "Editor", "Perspective", and "Navigator".

Conventions

The *Quick Start Guide* uses these conventions:

- **Bold** is for clickable buttons and menu selections.
- *Italics* is for screen text, path and file names, and other text that needs special emphasis.
- `Courier` denotes code, or text in a log or a batch file.
- A **Note** contains important information, or a timesaving tip.
- The scissors icon marks text that you copy from this document and paste elsewhere.

What You'll Need

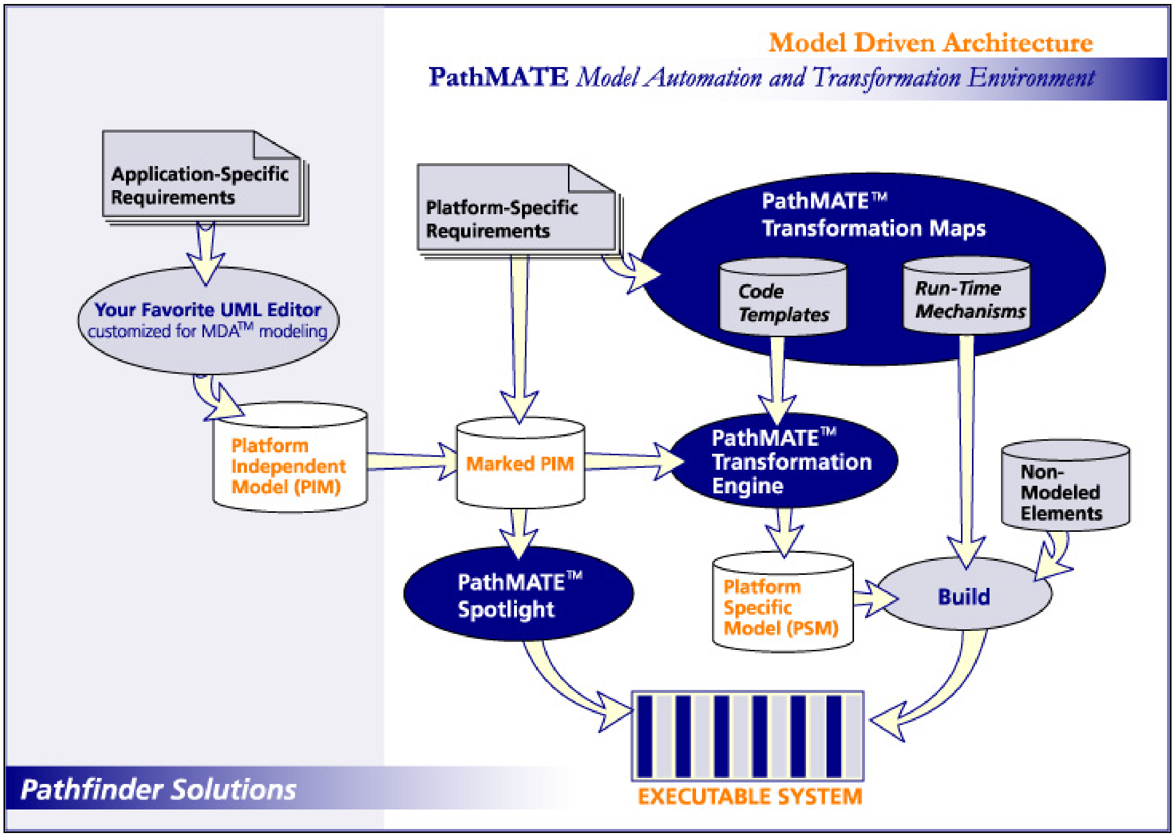
To complete the steps in this guide, you'll need the following software on your computer:

- Microsoft Windows 2000 or Windows XP Professional Edition
- PathMATE for MagicDraw software development toolkit
- Microsoft Visual C++ version 6, 7 or 8

PathMATE PI-MDD Toolset

Through its *Architectural Focus* and advanced automation, PathMATE is the leading MDD environment for embedded, real-time systems development. This powerful technology automates the verification, execution and deployment of Platform-Independent MDD models and automatically transforms them into high performance C, C++, and Java systems that are specifically optimized for resource-constrained embedded environments. Over years of rigorous refinement in several industries, PathMATE tools have proven their value in rapid and effective embedded systems development.

The PathMATE Model Automation and Transformation Environment includes all the tools required to transform your models into high-performance embedded systems:


Model Driven Architecture
PathMATE Model Automation and Transformation Environment
Application-Specific Requirements
Your Favorite UML Editor customized for MDA™ modeling
Platform Independent Model (PIM)
Platform-Specific Requirements
Marked PIM
PathMATE™ Transformation Maps
Code Templates
Run-Time Mechanisms
PathMATE™ Transformation Engine
Non-Modeled Elements
PathMATE™ Spotlight
Platform Specific Model (PSM)
Build
EXECUTABLE SYSTEM
Pathfinder Solutions

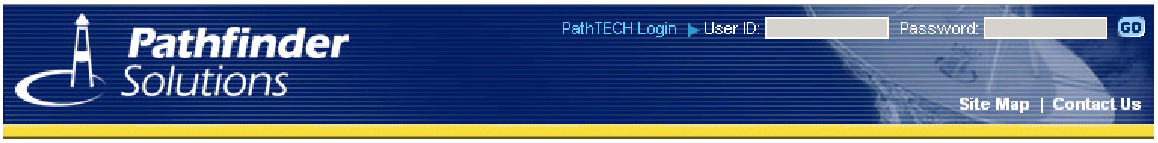
The PathMATE PI-MDD Toolset is comprised of three parts which work together to turn your models into executable embedded systems:

- *Transformation Engine* – generates embedded software applications from your application-specific platform-independent models.
- *Transformation Maps* – guide the conversion process from model to platform-specific executable code. You can choose from standard off-the-shelf C, C++, and Java maps. To meet the demands for other languages or specific platform needs, our consultants can work with you to develop customized high-performance maps.
- *Spotlight* – provides the most advanced model testing environment available to verify and debug your application logic.

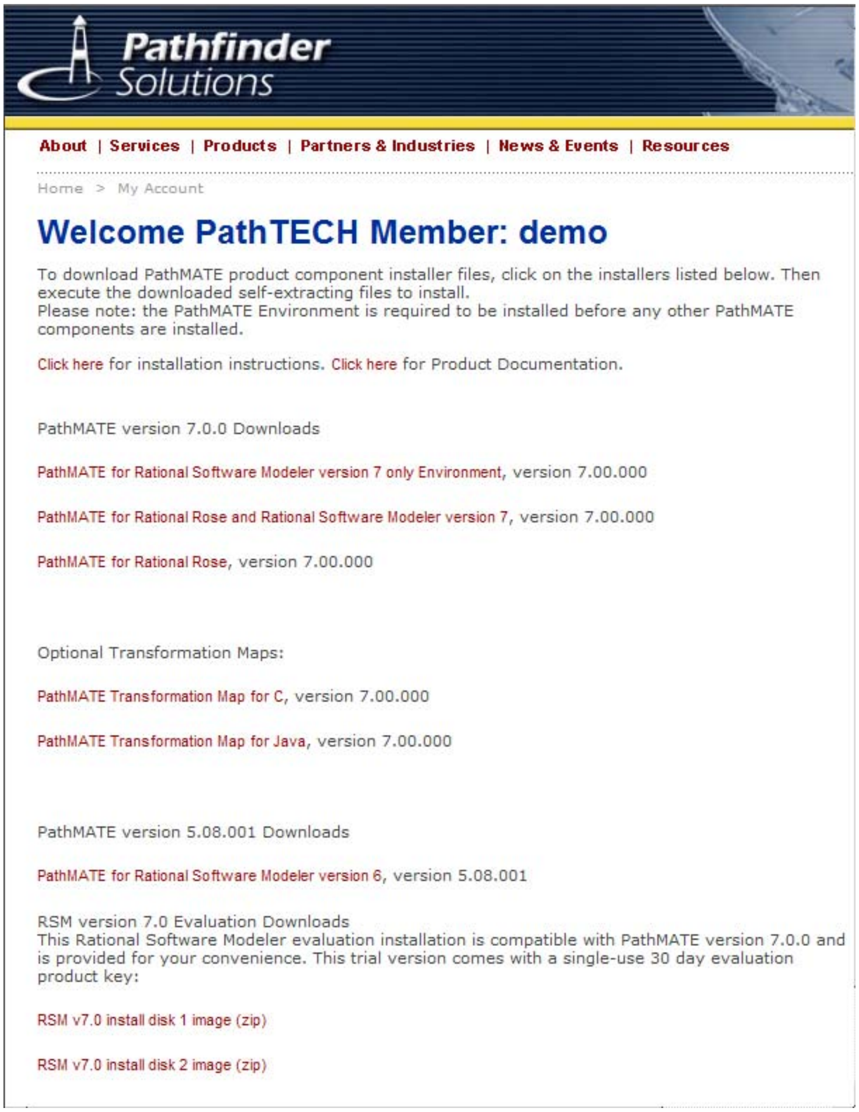
No other MDD transformation environment offers a more open or configurable set of development tools, designed to meet the requirements of embedded systems engineers.

Download and Install PathMATE

1. Navigate to http://www.pathfindermda.com. The Pathfinder Solutions home page opens.
2. Log into PathTECH at the top of the main web page with user ID *demo*. Please contact your account manager to obtain your password.



On the PathTECH download page you'll be able to select the appropriate PathMATE product downloads:


Pathfinder Solutions
About | Services | Products | Partners & Industries | News & Events | Resources
Home > My Account

Welcome PathTECH Member: demo

To download PathMATE product component installer files, click on the installers listed below. Then execute the downloaded self-extracting files to install.
Please note: the PathMATE Environment is required to be installed before any other PathMATE components are installed.

Click here for installation instructions. Click here for Product Documentation.

PathMATE version 7.0.0 Downloads

PathMATE for Rational Software Modeler version 7 only Environment, version 7.00.000

PathMATE for Rational Rose and Rational Software Modeler version 7, version 7.00.000

PathMATE for Rational Rose, version 7.00.000

Optional Transformation Maps:

PathMATE Transformation Map for C, version 7.00.000

PathMATE Transformation Map for Java, version 7.00.000

PathMATE version 5.08.001 Downloads

PathMATE for Rational Software Modeler version 6, version 5.08.001

RSM version 7.0 Evaluation Downloads
This Rational Software Modeler evaluation installation is compatible with PathMATE version 7.0.0 and is provided for your convenience. This trial version comes with a single-use 30 day evaluation product key:

RSM v7.0 install disk 1 image (zip)

RSM v7.0 install disk 2 image (zip)

3. Download the following files:
    - *PathMATE, version 7.40.000.*
    - *PathMATE Transformation Map for Java, version 7.40.000 (if you will be generating Java implementation code.)*
    - *Installation Instructions* – Follow these instructions to install the product and secure a key file.

After installation, see the file C:/pathmate/doc/PathMATE_with_Visual_Studio_Express.pdf for specific instructions on how use Microsoft Visual Studio 2005 Express Edition if you plan to use this environment.

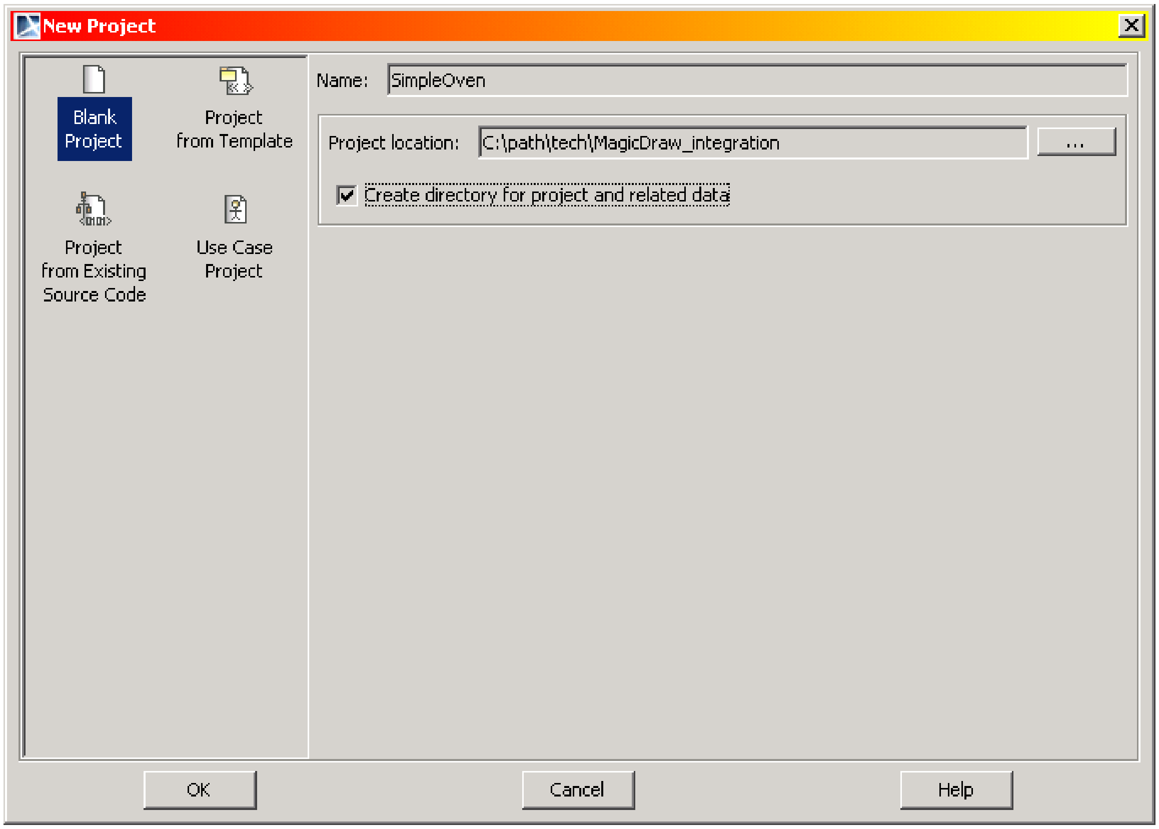
Model a Simple Oven

Task 1: Create and Set Up a UML Project

In MagicDraw, all model data is contained within a project. We will create a new blank project and set it up for use with PathMATE.

PROCEDURE: Use MagicDraw File → New to setup project

1. Launch MagicDraw and select **File → New**.
2. Specify the project name as *SimpleOven*.
3. Specify a location to store the project files.
4. Specify to Create directory for project and related data.
5. Select **OK**.

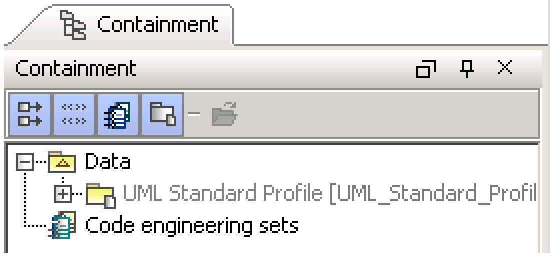


Name: SimpleOven

Project location: C:\path\tech\MagicDraw_integration

☑ Create directory for project and related data

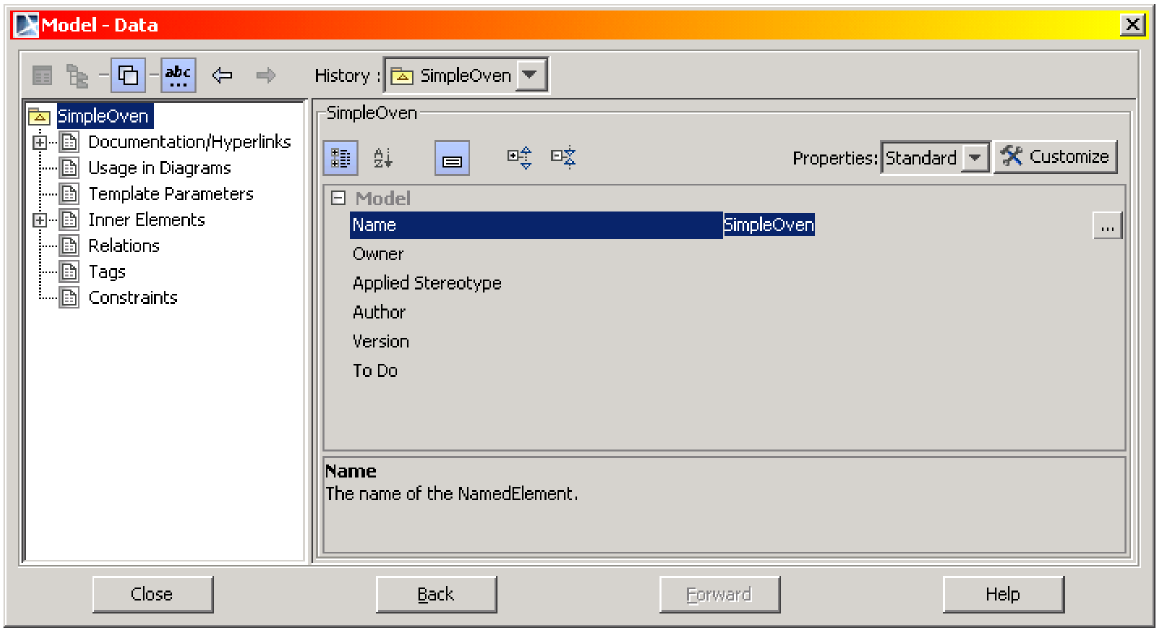
The Containment browser (usually in the upper left of your MagicDraw environment) shows your new project called *Data*:



6. Select your new model Data, right click and select **Specification**.

The Specification dialog is available for all elements in your MagicDraw model, including all containers (like models and packages) diagrams and other model elements. You can open the Specification dialog on any element using the context menu (as in Step 6 above), or by selecting the item and hitting the Enter key, or by double clicking on the element.

7. Using the Name field rename your model *SimpleOven:*



Name: SimpleOven

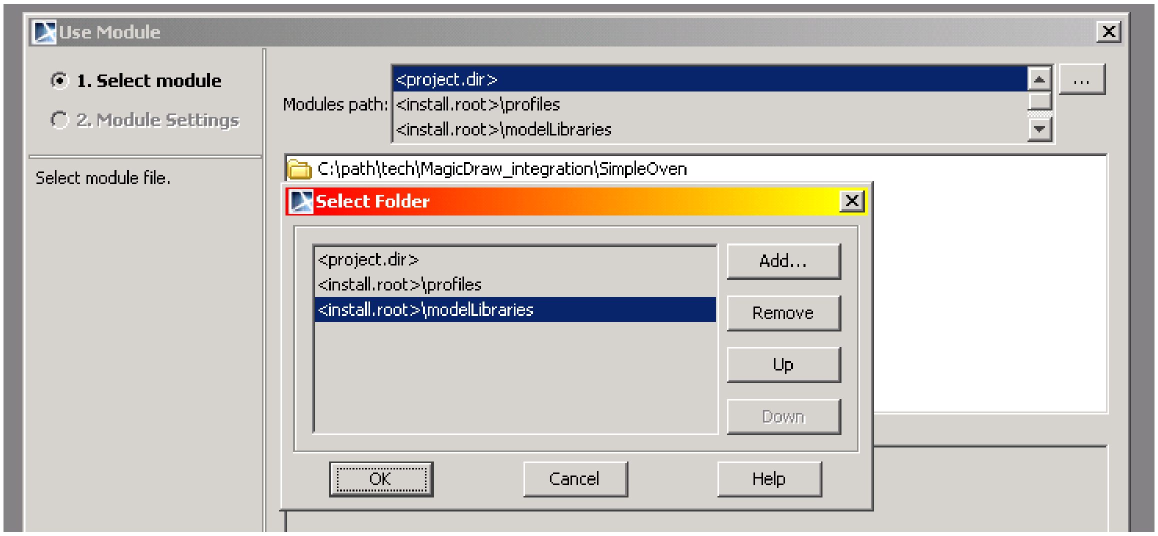
Name
The name of the NamedElement.

8. Select **Close**, then **File → Save Project** to save your model.

PROCEDURE: Import PathMATE Profile

In order to properly construct a Platform Independent model suitable for use with PathMATE we will to import the PathMATE Profile, which contains PathMATE specific data types, stereotypes, and the SoftwareMechanisms domain. Once this procedure is complete, you will be able to browse the services offered by the SoftwareMechanisms domain on your own.

1. Select **File → Use Module**
2. In the Use Module dialog, click the **browse** button to the right of the Modules path list
3. In the Select Folder dialog click Add.

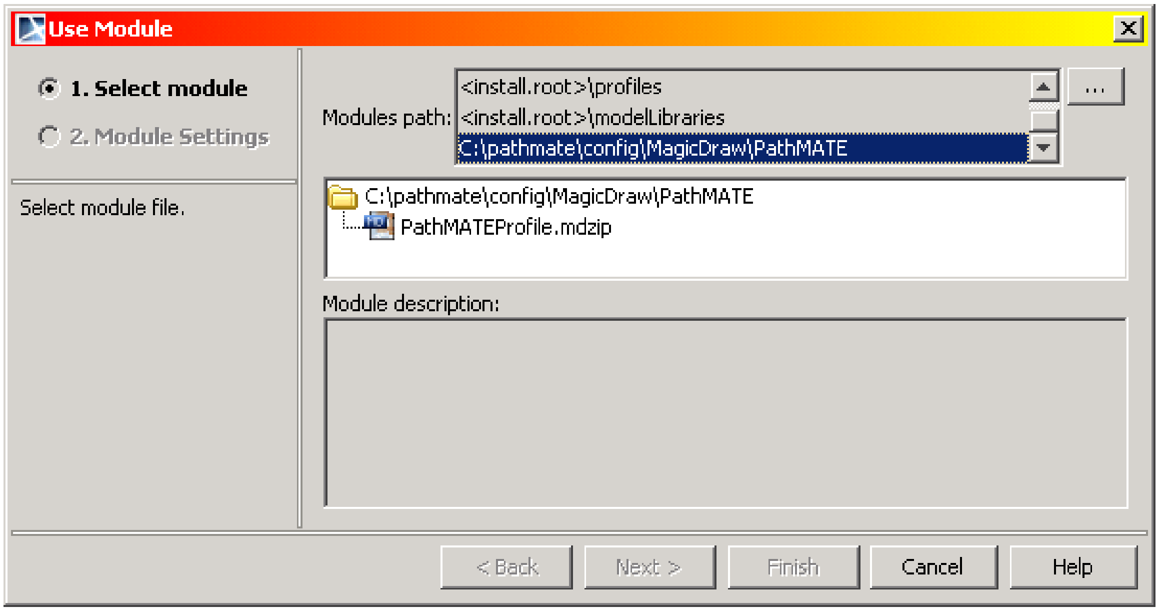


Modules path:
- <project.dir>
- <install.root>\profiles
- <install.root>\modelLibraries

C:\path\tech\MagicDraw_integration\SimpleOven

Select Folder:
- <project.dir>
- <install.root>\profiles
- <install.root>\modelLibraries

4. Browse to PathMATE MagicDraw configuration folder, by default in *C:\pathmate\config\MagicDraw\PathMATE* (you can copy this filename directly from here and paste it directly into the File name field). Click **Open** and **OK**.
5. Scroll to the bottom of the Modules path list and click on the PathMATE MagicDraw configuration folder



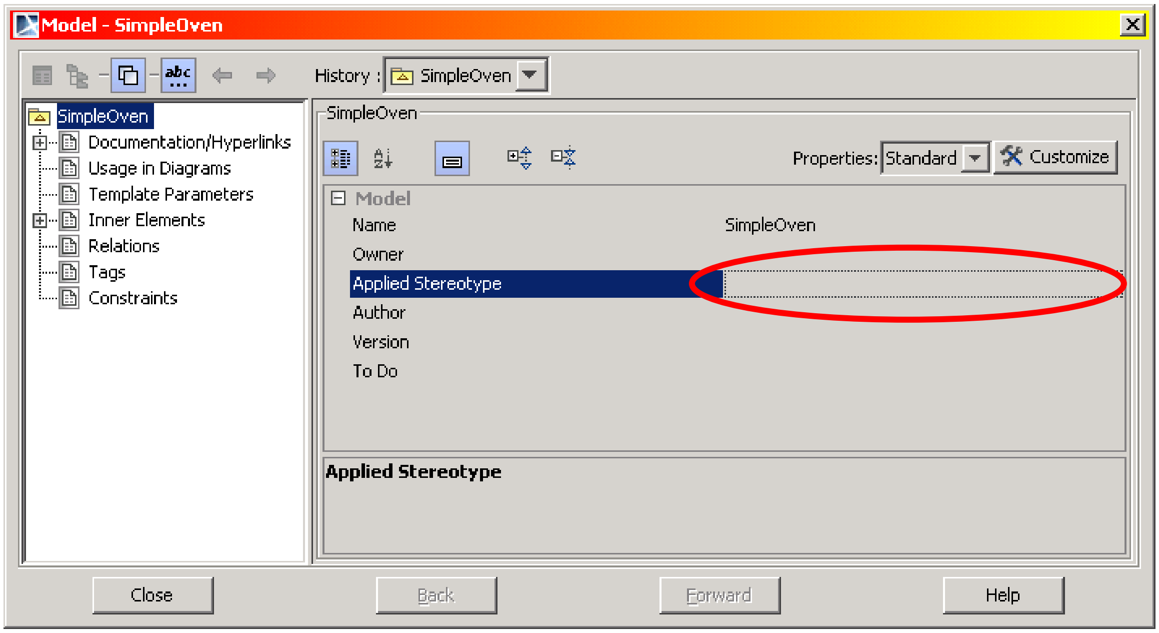
Modules path:
- <install.root>\profiles
- <install.root>\modelLibraries
- C:\pathmate\config\MagicDraw\PathMATE

C:\pathmate\config\MagicDraw\PathMATE
- PathMATEProfile.mdzip

6. Select the *PathMATEProfile.mdzip* file. Click **Finish.**

PROCEDURE: Indentify our Model as a PathMATE System

1. Open the Specification for the SimpleOven model.
2. Click to the right of the Applied Stereotype field:



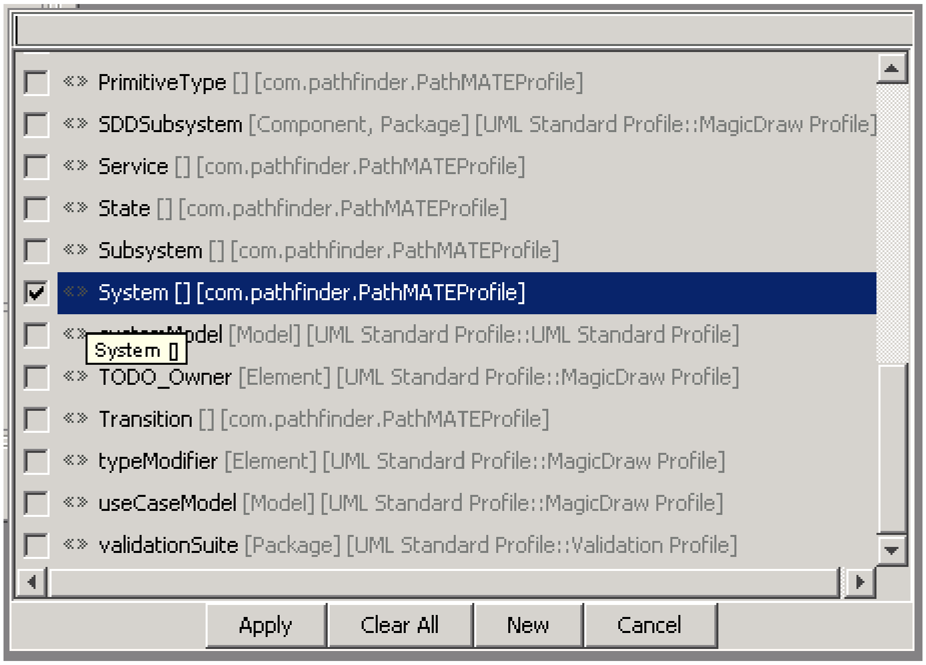
SimpleOven

Model

Name	SimpleOven
Owner	
Applied Stereotype	
Author	
Version	
To Do	

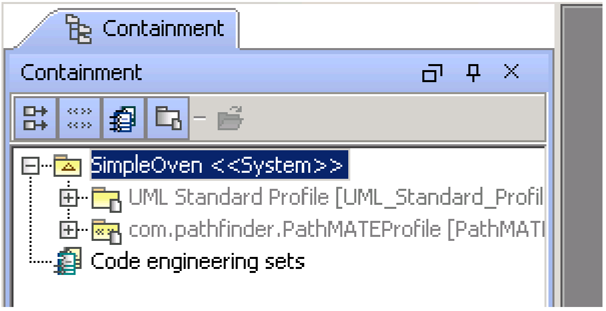
Applied Stereotype

3. On the right edge of this field click on the **edit button** (it contains "…").
4. Scroll down in the Stereotype pick list and select **System**


- □ «» PrimitiveType [] [com.pathfinder.PathMATEProfile]
- □ «» SDDSubsystem [Component, Package] [UML Standard Profile::MagicDraw Profile]
- □ «» Service [] [com.pathfinder.PathMATEProfile]
- □ «» State [] [com.pathfinder.PathMATEProfile]
- □ «» Subsystem [] [com.pathfinder.PathMATEProfile]
- ☑ «» System [] [com.pathfinder.PathMATEProfile]
- □ «» systemModel [Model] [UML Standard Profile::UML Standard Profile]
- □ «» TODO_Owner [Element] [UML Standard Profile::MagicDraw Profile]
- □ «» Transition [] [com.pathfinder.PathMATEProfile]
- □ «» typeModifier [Element] [UML Standard Profile::MagicDraw Profile]
- □ «» useCaseModel [Model] [UML Standard Profile::MagicDraw Profile]
- □ «» validationSuite [Package] [UML Standard Profile::Validation Profile]

5. Click **Apply**, and then **Close** in the Specification dialog.

6. Your Containment browser should look like:



SimpleOven <<System>>
UML Standard Profile [UML_Standard_Profil
com.pathfinder.PathMATEProfile [PathMATI
Code engineering sets

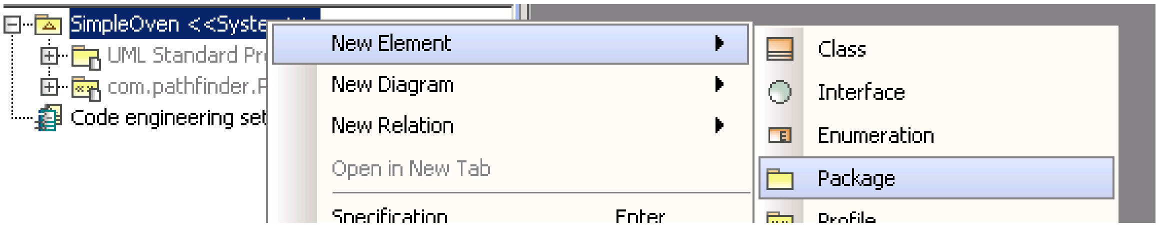
Task 2: Add a Modeled Domain

Domains are logical components – the primary top-level building block throughout the PathMATE modeling process. A domain is a separate conceptual "world," inhabited by a distinct set of elements that rely on each other, but do not rely on any elements outside of their own domain.

A "modeled" or "analyzed" domain is one that we have created a platform independent model for. The model is complete and executable, in the sense that all behavior will be included in the model.

PROCEDURE: Add a new Package and Apply the Domain Stereotype

1. In the Containment browser select the *SimpleOven* model, right-click and select **New Element → Package**



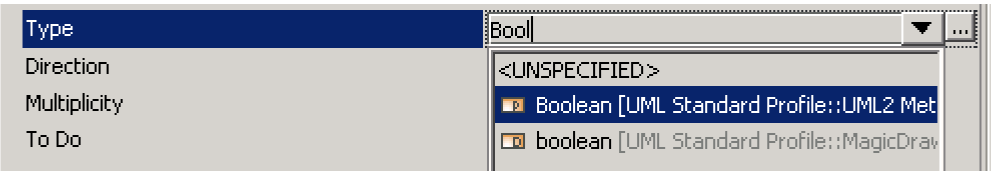
2. Type *MicrowaveCooking* as the name of the package and hit **Enter**.
3. Open the Specification dialog for the new package. Using the Applied Stereotype field apply the stereotype **Domain**. Close the Specification dialog.

PROCEDURE: Create the Domain Interface for the MicrowaveCooking Domain and add a Service Operation

Each domain publishes a set of services contained in a stereotyped interface class.

1. Right click on *MicrowaveCooking* in the project explorer and select **New Element → Interface**, and name it *MC services.*
2. Use the Specification dialog for the new interface to apply the stereotype **DomainInterface**.
3. Right click on *MC services,* select **New Element → Operation**, and name it *ReportDoorStatus*.
4. Open the Specification dialog for the new operation.
5. In the Type field start to enter *Boolean.*

6. The Boolean type from the PathMATE profile should be selected in the pick list:



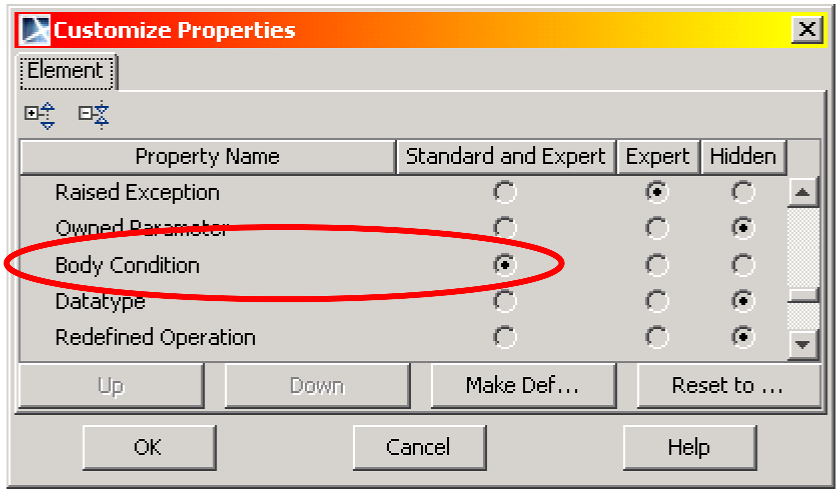
Type	Bool
Direction	<UNSPECIFIED>
Multiplicity	Boolean [UML Standard Profile::UML2 Met
To Do	boolean [UML Standard Profile::MagicDrav

7. Hit **Enter** to pick this type.
8. Hit **Close** on the Specification dialog.

PROCEDURE: Add Action Language for ReportDoorStatus Domain Service

PathMATE uses a Platform-independent Action Language (PAL), to specify the behavior of model elements. PAL directly accesses and manipulates model-level constructs for simplicity and convenience, and is transformed to self-optimized implementation code for optimal runtime efficiency and performance. For more information about PAL specifics, see AL_oview.pdf in the PathMATE documentation, c:\pathmate\doc.

1. Open the Specification dialog for the *ReportDoorStatus* operation.
2. Click on the **Customize** button – this in the upper right region of the dialog.
3. For the *Body Condition* property, click the button in the **Standard and Expert** column.



Property Name	Standard and Expert	Expert	Hidden
Raised Exception	○	●	○
Owned Parameter	○	○	●
Body Condition	●	○	○
Datatype	○	○	●
Redefined Operation	○	○	●

4. Click **OK**.
5. Click in the **Body Condition** field – in the area to the right of the "Body Condition" label. The create condition button ("...") will appear. Click the button. Select **Constraint** from the pick list.

6. The Constraint Specification dialog will appear. Name the constraint *action*.
7. Click in the **Specification** field.
8. Copy the text below into the Specification field.

```
Ref<Door> door = FIND CLASS Door;
IF (door != NULL)
{
    IF (is_open)
    {
        GENERATE Door:IsOpen() TO (door);
    }
    ELSE
    {
        GENERATE Door:IsClosed() TO (door);
    }
}
```

TIP: If you are reading this using Adobe Acrobat, click the Select Text tool in your Acrobat toolbar. Then select the text, copy it, and paste using Control-V into the MagicDraw field.

9. Hit **Close** on the Specification dialog.

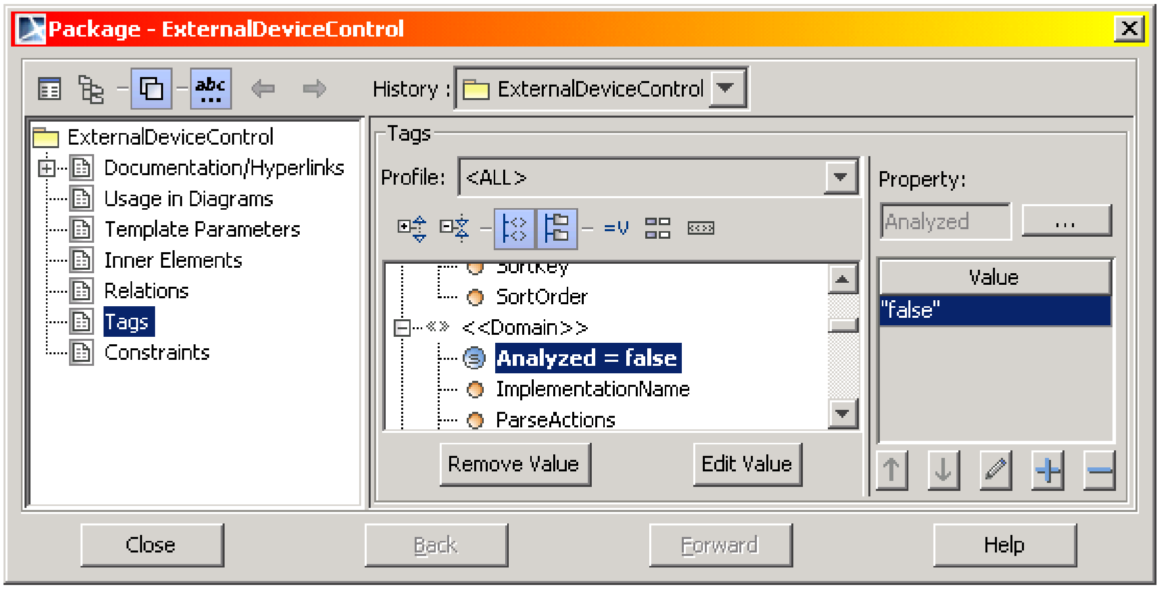
Task 3: Create the Realized Domain ExternalDeviceControl

ExternalDeviceControl is a domain that will have no model elemented inside it. It only provides wrapper for external implementation code that directly provides it's required domain. These procedures explain how to abstract the interface to the *ExternalDeviceControl* services at the Platform Independent Model (PIM) level for use by modeled domains.

PROCEDURE: Create the ExternalDeviceControl Domain and Set the UseAsExternal Property

1. Just like you did with *MicrowaveCooking* add a new package to the SimpleOven model and name it *ExternalDeviceControl*.
2. Open the Specification dialog for the new domain.
3. Apply the **Domain** stereotype to it.
4. On the right pane of the Specification dialog select **Tags**.
5. Scroll down through the list of tags to the <<*Domain*>> tags, and select **Analyzed**, and click **Create Value.**
6. In the Value pane overwrite the default *true* value with *false*.

False will appear in the center Tag value table:



Package - ExternalDeviceControl

History : ExternalDeviceControl

Profile: <ALL>

Property: Analyzed

Value: "false"

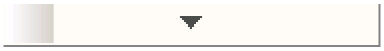
<<Domain>>

Analyzed = false

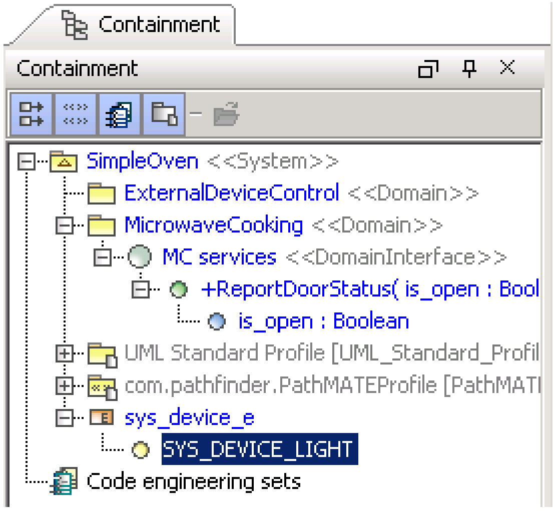
7. **Close** the Specification dialog.

PROCEDURE: Add a UML Enumeration Type to the System's Types

1. Right click on the *SimpleOven* and select **New Element**.
2. At the bottom of the element type list there is a show more indicator – click this:



3. Select **Enumeration**n and name the new type *sys_device_e*.
4. Right click on ***sys_device_e***, select **New Element → Enumeration Literal,** and name it *SYS_DEVICE_LIGHT*.



```
Containment
Containment
SimpleOven <<System>>
    ExternalDeviceControl <<Domain>>
    MicrowaveCooking <<Domain>>
        MC services <<DomainInterface>>
            +ReportDoorStatus( is_open : Bool
                is_open : Boolean
    UML Standard Profile [UML_Standard_Profil
    com.pathfinder.PathMATEProfile [PathMATI
    sys_device_e
        SYS_DEVICE_LIGHT
Code engineering sets
```

PROCEDURE: Add Service Operations to the Domain Interface

1. Right click on *ExternalDeviceControl* in the project explorer and select **New Element → Interface**, and name it *EDC services.*
2. Use the Specification dialog for the new interface to apply the stereotype **DomainInterface**.
3. Right click on *MC services,* select **New Element → Operation**, and name it *ActivateDevice*.
4. Call the new argument *device_id*.
5. Open the Specification for *device*_id, and use the Type field to select the type **sys_device_e**.
6. **Close** the Specification.
7. Repeat this procedure to add another service called *DeactivateDevice* with one parameter, called *device_id*, of type sys_device_e.
8. Hit **Control-S** to save your project.

Task 4: Complete the System Domain Chart

The Domain Chart shows the Logical Architecture of the system in terms of domains arranged in a hierarchy where the most abstract (pertaining most directly to the overall mission of the system) are at the top, and least abstract (closest to implementation) are at the bottom. Dependency arrows show the flow of requirements downward through the system in a form of delegation.

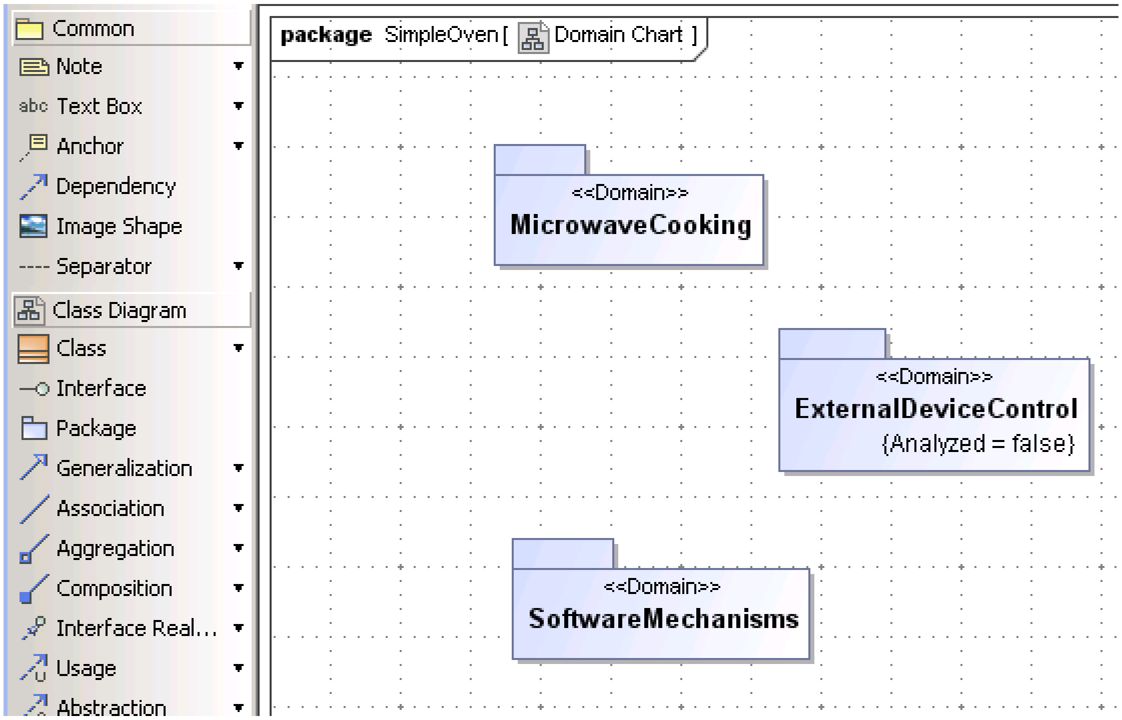
For PathMATE systems, the SoftwareMechanisms domain is always at the bottom.

PROCEDURE: Build the Domain Chart

1. Right click on the *SimpleOven* and select **New Diagram → Class Diagram**. Name the new diagram *Domain Chart*.

The Domain Chart opens in the MagicDraw main editing pane.

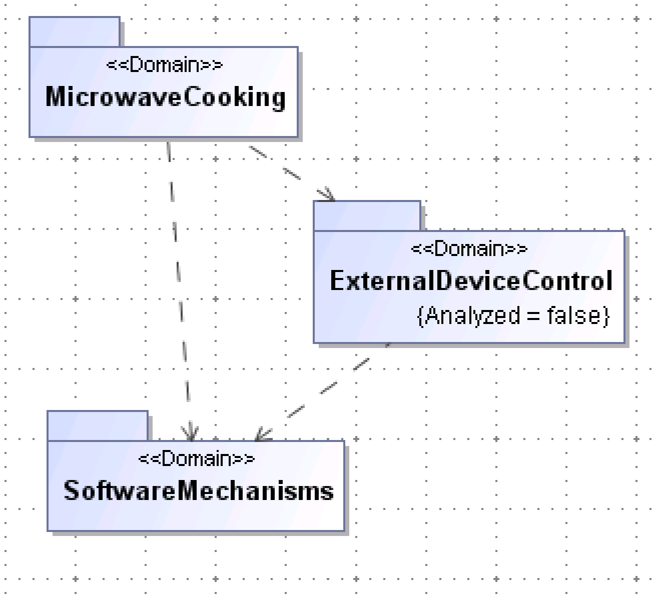
2. From the Containment browser drag each of the three domains into the drawing pane: *MicrowaveCooking, ExternalDeviceControl,* and *SoftwareMechanisms* (from within the PathMATE profile package).



3. From the toolbar on the left edge of the diagram select the **Dependency Tool** ↗.

4. Click on the interior of the *MicrowaveCooking* domain box. Then click on the interior of the *ExternalDeviceControl*. This draws a dependency arrow between the domains.

5. Draw another dependency from *MicrowaveCooking* to *SoftwareMechanisms*, and another from *ExternalDeviceControl* to *SoftwareMechanisms*.



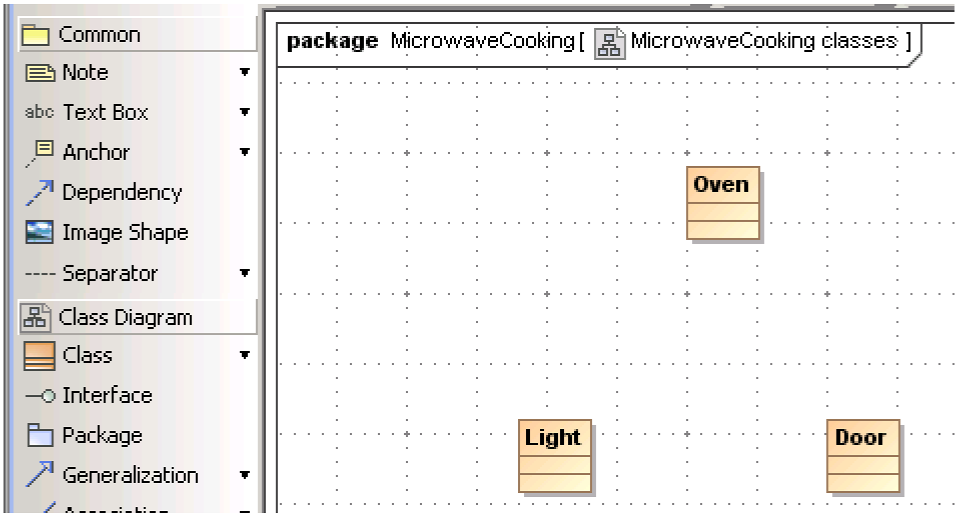
9. Hit **Control-S** to save your project.

Task 5: Complete the Class Diagram for the MicrowaveCooking Domain

The MicrowaveCooking Domain is the heart of the application, delivering a galaxy of sophisticated capability and complexity (a door triggers a light). In this task, you will build the MicrowaveCooking's Domain Class Diagram with Classes, Attributes, and Associations.

PROCEDURE: Create Classes in the Microwave Cooking Domain

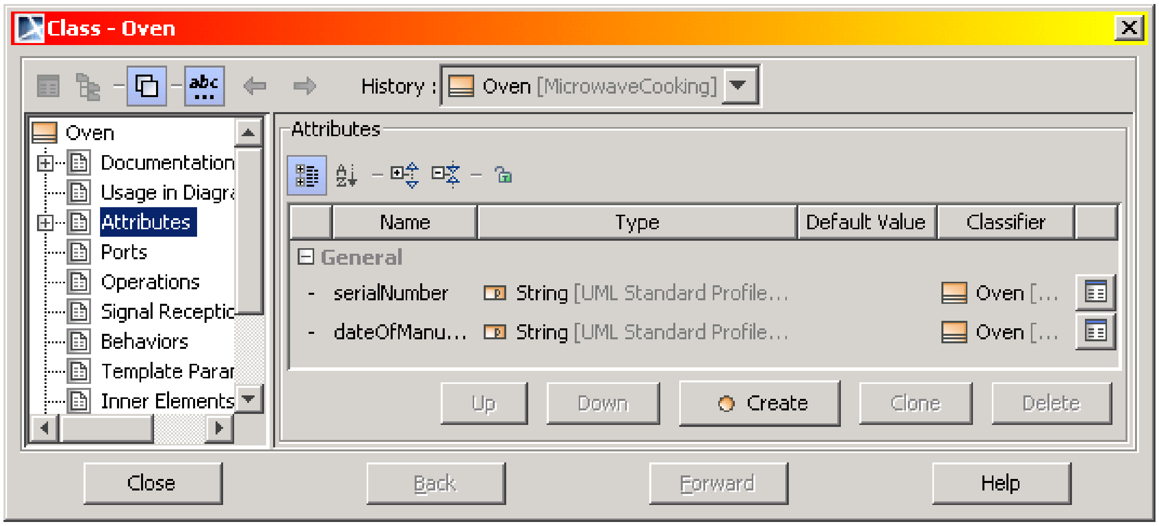
1. Right click on the *MicrowaveCooking* domain in the Containment browser and select **New Diagram → Class Diagram**. Name the diagram *MicrowaveCooking classes*.
2. From the toolbar on the left edge of the diagram select the **Class Tool** ☰.
3. Click in the drawing area to create a new class.
4. Title it *Oven*.
5. Create two more classes in the same manner called *Light* and *Door*.



PROCEDURE: Add Attributes to the Oven and Light Classes

1. Select the *Oven* class in the drawing area.
2. Click the **Insert New Attribute** button ○ on the right edge of the class symbol. Name it *serialNumber*.
3. Hit **Enter**. This creates a second attribute. Name it *dateOfManufacture*.
4. Double-click the **Oven** on the diagram to bring up its Specification.
5. On the right pane of the Specification dialog select **Attributes.**

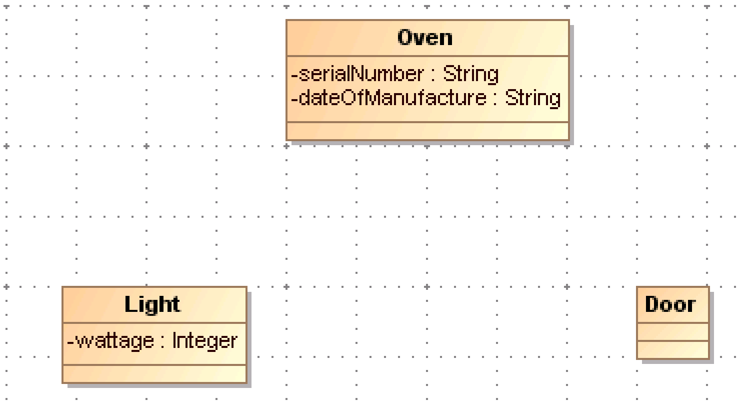
6. In the Type field for both attributes enter *String*. Note how the type chooser narrows the type selections available as you type.



	Name	Type	Default Value	Classifier
⊟ General				
-	serialNumber	String [UML Standard Profile...		Oven [...
-	dateOfManu...	String [UML Standard Profile...		Oven [...

7. **Close** the Specification dialog.

8. Follow the same procedure to add an attribute called *wattage* of type *Integer* to the *Light* class.



PROCEDURE: Add Associations between Classes

Classes within a domain work together. Associations are used to capture how classes work together.

1. From the toolbar select the **Association** tool .

2. Use the **Association** tool to draw an association from the *Oven* class to the *Light* class. Remember to click on the interior of the class boxes.

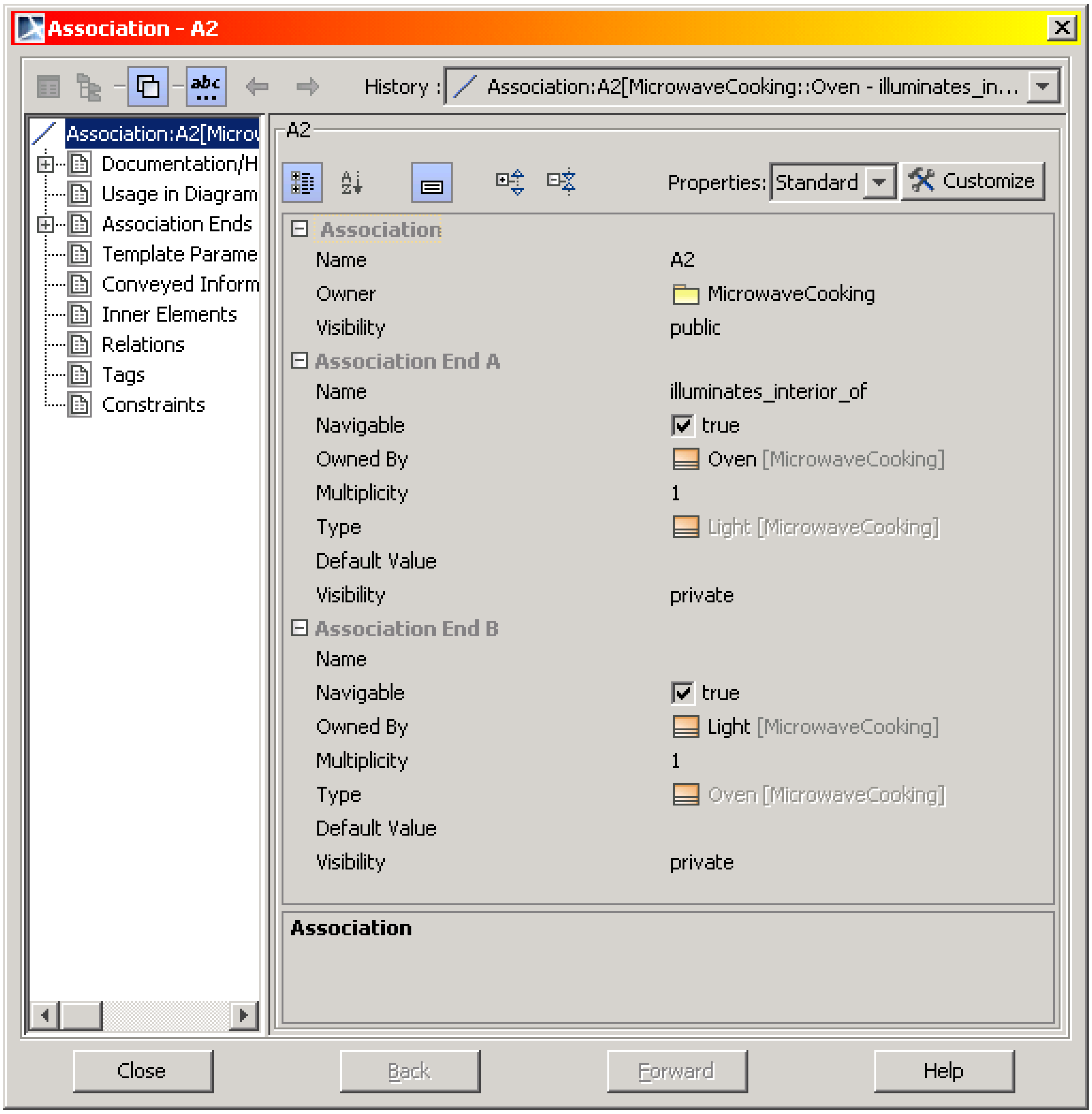
3. Double click on the association to bring up the Specification dialog.

4. In the Name field, enter *A2.*

NOTE: This name is used to quickly refer to an association, including from action language. You construct association names for all associations using the A<number> form. All associations within a domain must be named (numbered) uniquely.

5. Under the Association End A fields enter *illuminates_interior_of* in the Name field, and select **1** from the Multiplicity pick list.
6. Under the Association End B fields select **1** from the Multiplicity pick list.

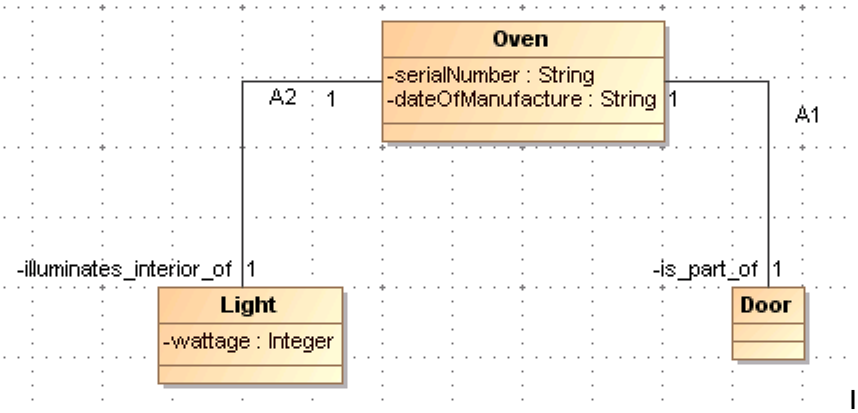
NOTE: End A and End B may be reversed if you started drawing the association from the Light class. That will work just fine, if you keep the *illuminates_interior_of* Name with the association end that has a Type of *Light*.



Association	
Name	A2
Owner	MicrowaveCooking
Visibility	public
Association End A	
Name	illuminates_interior_of
Navigable	☑ true
Owned By	Oven [MicrowaveCooking]
Multiplicity	1
Type	Light [MicrowaveCooking]
Default Value	
Visibility	private
Association End B	
Name	
Navigable	☑ true
Owned By	Light [MicrowaveCooking]
Multiplicity	1
Type	Oven [MicrowaveCooking]
Default Value	
Visibility	private

Association

7. **Close** the Specification dialog.
8. Use the **Association** tool to create association *A1* between the *Oven* and *Door* classes as shown here:

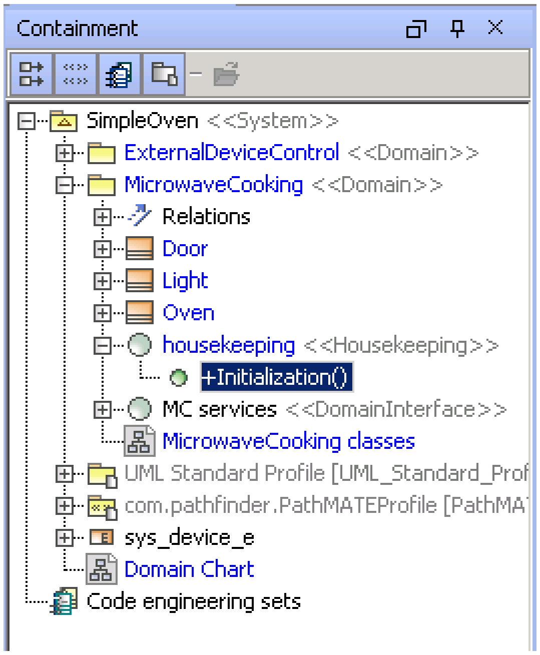


10. Hit **Control-S** to save your project.

Task 6: Complete Domain Housekeeping Details

PROCEDURE: Add MicrowaveCooking Initialization Action

1. Right-click on the *MicrowaveCooking* domain in the Containment browser and select **New Element → Interface**. Name it *housekeeping.*
2. Use the *housekeeping* Specification dialog to apply the stereotype **Housekeeping** to it.
3. Right-click on the *housekeeping* interface and select **New Element → Operation**. Name it *Initialization.* Now your Containment browser looks like:



Containment

- SimpleOven <<System>>
  - ExternalDeviceControl <<Domain>>
  - MicrowaveCooking <<Domain>>
    - Relations
    - Door
    - Light
    - Oven
    - housekeeping <<Housekeeping>>
      - +Initialization()
    - MC services <<DomainInterface>>
    - MicrowaveCooking classes
  - UML Standard Profile [UML_Standard_Prof
  - com.pathfinder.PathMATEProfile [PathMA
  - sys_device_e
  - Domain Chart
- Code engineering sets

4. Open the *Initialization* Specification dialog
5. Click in the **Body Condition** field and add a **Constraint** from the pick list button on the right. Name the constraint *action*.
6. Click in the **Specification** field.
7. Copy the text below into the Specification text field:

```
// Set up instances for MicrowaveCooking
Ref<Oven> mw_oven = CREATE Oven (serialNumber = "G023-4ZZ-8811", dateOfManufacture = "2004/02/22;
10:41");
Ref<Light> interior_light = CREATE Light(wattage = 25);
Ref<Door> door = CREATE Door();
LINK mw_oven A1 door;
LINK mw_oven A2 interior_light;
// Now just to start things off, open the door
GENERATE Door:IsOpen() TO (door);
```

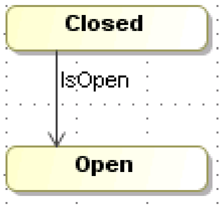
8. Hit **Control-S** to save your project.

Task 7: Create the Door State Machine

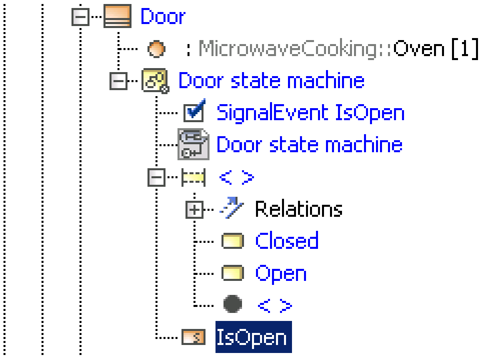
State Machines are used to describe the lifecycle of a class using an event-response pattern.

PROCEDURE: Create the Door State Machine Diagram with States and Transitions

1. In the Containment browser right click on the Door class and select **New Diagram → State Machine Diagram**. Name the new diagram *Door state machine*.
2. Use the **State tool** ▭ to create two states: *Closed* and *Open*.
3. Use the **Initial tool** ● to create an initial pseudostate.
4. Select the **Transition** tool ↗ click in the *Closed* state box, click in the *Open* state box and type the trigger signal name *IsOpen.*



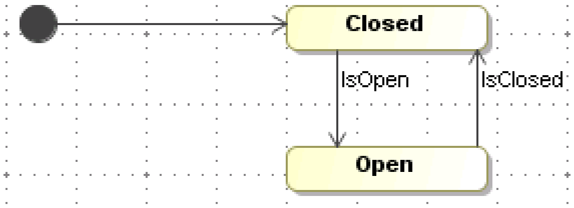
NOTE: This action created the Signal *IsOpen* within the Door state machine:



```
Door
    : MicrowaveCooking::Oven [1]
    Door state machine
        SignalEvent IsOpen
        Door state machine
        < >
            Relations
            Closed
            Open
            < >
        IsOpen
```

5. Use the **Transition** tool ↗ to create a transition from *Open* to *Closed* with the trigger signal *IsClosed.*
6. Use the **Transition** tool ↗ to create a transition from the initial pseudostate to *Closed* with no trigger signal *(this is untriggered).*

The *Door* state machine looks like:



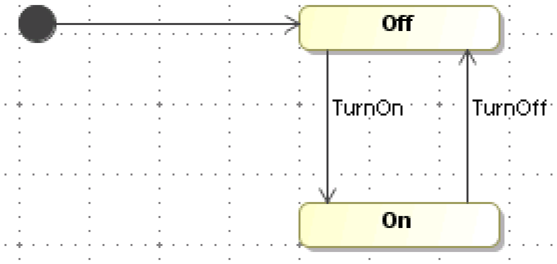
7. Hit **Control-S** to save your project.

Task 9: Create the Light State Machine

PROCEDURE: Create the Light State Machine Diagram with States and Transitions

1. In the Containment browser right click on the *Light* class and select **New Diagram → State Machine Diagram**. Name the new diagram *Light state machine*.
2. Use the **State tool** ▭ to create two states: *Off* and *On*.
3. Use the **Initial tool** ● to create an initial pseudostate.
4. Use the **Transition** tool ↗ to create a transition from *Off* to *On* with the trigger signal *TurnOn.*
5. Use the **Transition** tool ↗ to create a transition from *On* to *Off* with the trigger signal *TurnOff.*
6. Use the **Transition** tool ↗ to create an untriggered transition from the initial pseudostate to *Off.*

The *Light* state machine looks like:



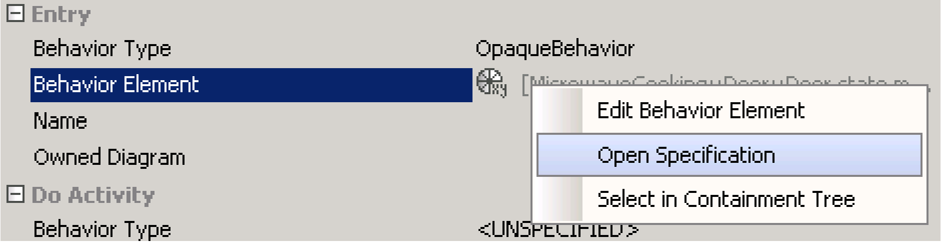
7. Hit **Control-S** to save your project.

Task 11: Add State Actions

When a lifecycle receives a signal, various transitions may be taken and actions may be executed: State Exit, Transition and/or State Entry. We will use State Entry Actions to specify behavior, including the generation of signals between the Door and the Light state machines. When the oven door is closed, the interior light will be turned off. When the oven door is open, the interior light will be turned on.

PROCEDURE: Add State Actions on the Door and Light States

1. Use the tabs at the top of the MagicDraw editing pane to select the *Door state machine*.
2. Open the Specification for the *Closed* State.
3. In the Entry section for Behavior Type pick **OpaqueBehavior**.
4. An OpaqueBehavior is added to the state.
5. Select the OpaqueBehavior, right click and select **Open Specification:**



⊟ Entry
Behavior Type OpaqueBehavior
Behavior Element [MicrowaveCooking::Door::Door state m
Name
Owned Diagram
⊟ Do Activity
Behavior Type <UNSPECIFIED>

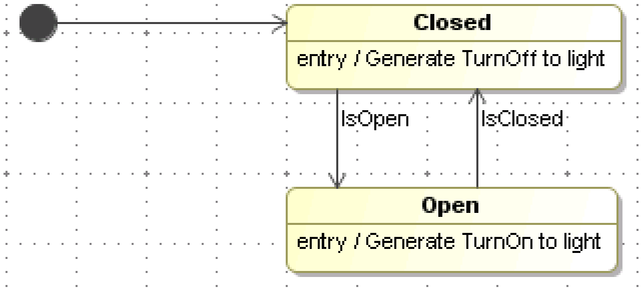
6. The Opaque Behavior Specification dialog opens. In the Name field enter *Generate TurnOff to light*.
7. Click to the right of the Body field and paste the following PAL into the Body field:

```
Ref<Light> interior_light = FIND this->A1->A2;
GENERATE Light:TurnOff() TO (interior_light);
```

8. **Close** the Specification.
9. Follow the same procedure on the *Open* state to add an entry OpaqueBehavior named *Generate TurnOn to light with* the following PAL:

```
Ref<Light> interior_light = FIND this->A1->A2;
GENERATE Light:TurnOn() TO (interior_light);
```

The *Door state machine* is now complete:


Closed
entry / Generate TurnOff to light
IsOpen
IsClosed
Open
entry / Generate TurnOn to light

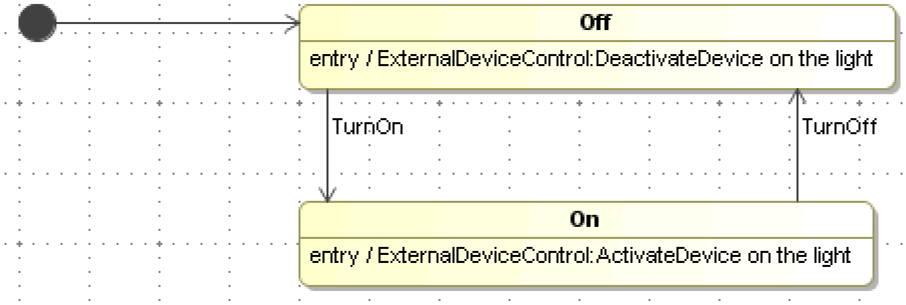
10. Use the tabs at the top of the MagicDraw editing pane to select the *Light state machine*.
11. Select the Off state and add an entry OpaqueBehavior named *ExternalDeviceControl:DeactivateDevice on the light* with the following PAL:

```
ExternalDeviceControl:DeactivateDevice(SYS_DEVICE_LIGHT);
```

12. Select the On state and add an entry OpaqueBehavior named *ExternalDeviceControl:ActivateDevice on the light* with the following PAL:

```
ExternalDeviceControl:ActivateDevice(SYS_DEVICE_LIGHT);
```

The *Light state machine* is now complete:


Off
entry / ExternalDeviceControl:DeactivateDevice on the light
TurnOn
TurnOff
On
entry / ExternalDeviceControl:ActivateDevice on the light

8. Hit **Control-S** to save your project.

Congratulations! Your QuickStart project has a Real Model.

Prepare for Transformation

You now have a complete, executable platform independent model and your project is just a few steps from being ready for transform into an executable system. You'll reuse some items from the SimpleOven Example project which comes with PathMATE. This section takes you through the following steps:

- Navigate to the complete sample SimpleOven project and use it as a reference, copy C++ and Java transformation properties and realized code from it.
- Add a PathMATE Project to the QuickStart project which will control transformation and deployment.
- "Smoke Test" your model by generating documentation from it.

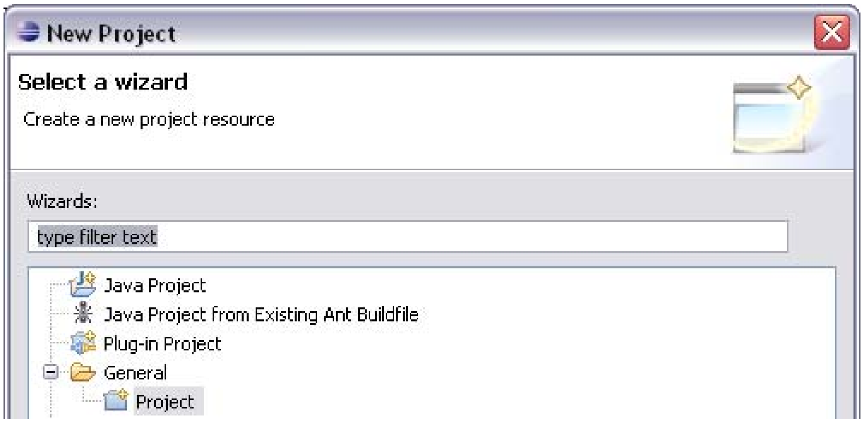
Task 1: Reuse elements from Example Project

It is now time to start working in the Eclipse environment. Open the version of Eclipse supplied with PathMATE (usually Start → All Programs → Pathfinder Solutions → Eclipse). Designate a workspace to store all the files for the sample system. Typically you designate your workspace once, just after you install Eclipse. We refer to this directory as your Workspace Directory. Use the same workspace whenever you work on this tutorial.

SimpleOven exists as a sample model in the PathMATE examples library. We will use SimpleOven as a reference project to copy the realized ExternalDeviceControl implementation and properties that guide the automatic generation of a C++ and Java artifacts for this system.

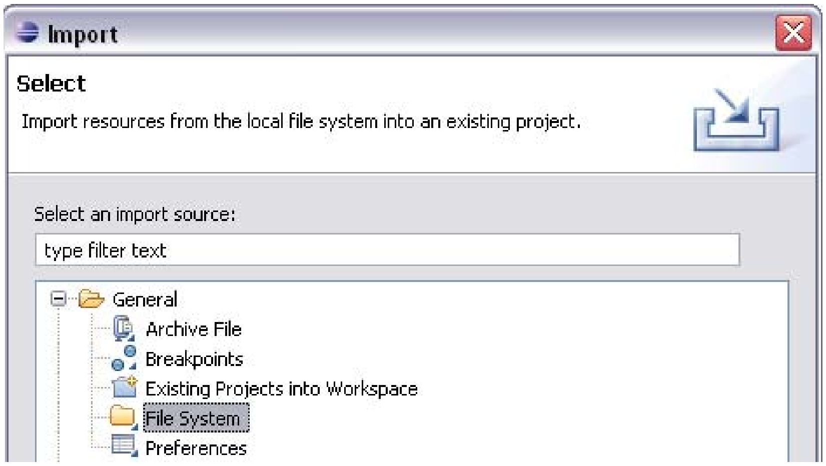
PROCEDURE: Use Eclipse to Instantiate a Reference Project for SimpleOven

4. Select **File → New → Project**.
5. Select **General →Project**.



6. Select **Next**.

7. Name the project *QuickStart*.
8. Repeat the same process to create a project called *ReferenceSimpleOven*.
9. Select **Finish**.
10. Select **File → Import** in the top menu bar.
11. Under the General folder select **File System**.



Import

Select

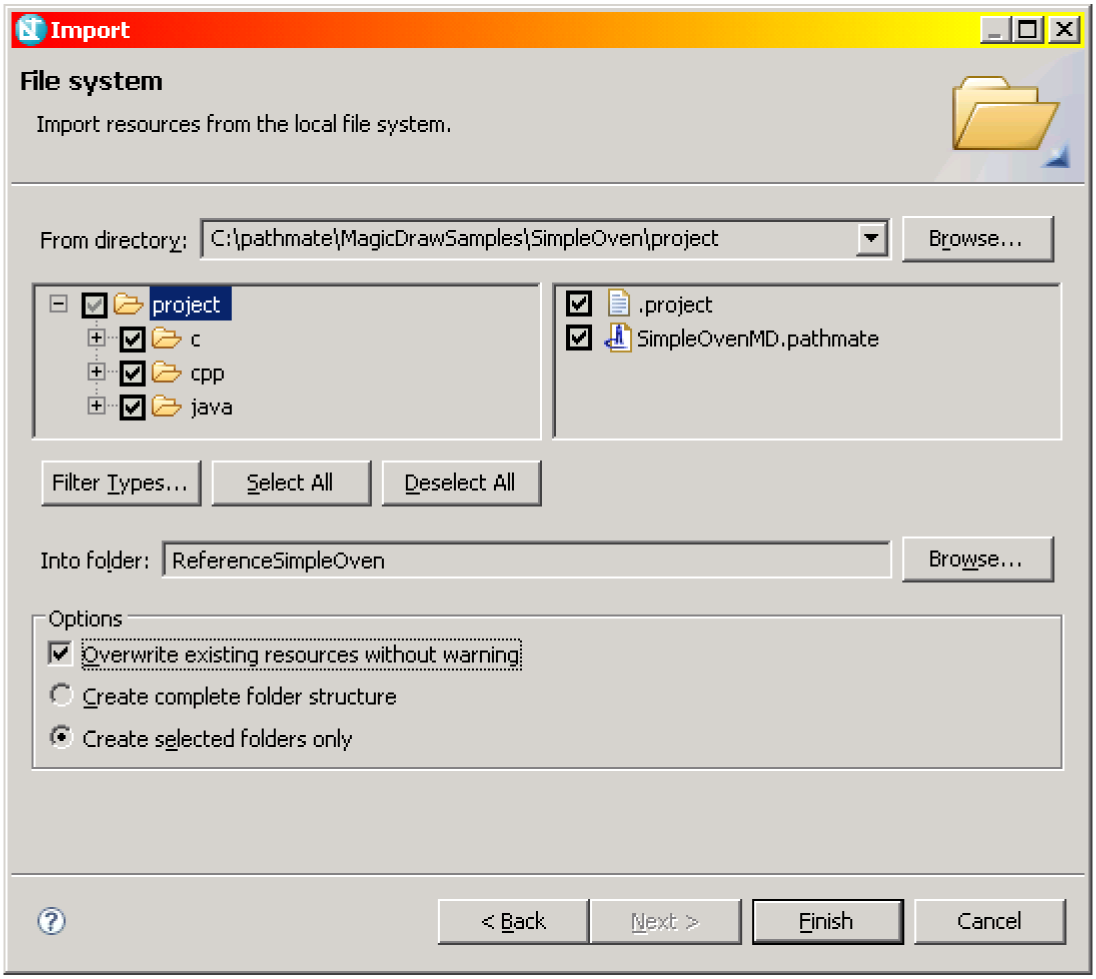
Import resources from the local file system into an existing project.

Select an import source:

type filter text

12. Press **Next**.
13. Click the **Browse**… button to choose the *From Directory*.
14. Browse to *C:\pathmate\MagicDrawSamples\SimpleOven\project*, and select **OK**.
15. Expand the *project* folder and select all folders under the *project* directory (but not the main *project* directory itself). Also select the **.project** and **SimpleOvenMD.pathmate** files.
16. Select **ReferenceSimpleOven** as the *Into folder*.
17. In the *options* section select **Create selected folders only**.

18. Select **Overwrite existing resources without warning**.



Import

File system

Import resources from the local file system.

From directory: C:\pathmate\MagicDrawSamples\SimpleOven\project

☑ project
- ☑ c
- ☑ cpp
- ☑ java

☑ .project
☑ SimpleOvenMD.pathmate

Into folder: ReferenceSimpleOven

Options
- ☑ Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

19. Press **Finish**.

PROCEDURE: Use Eclipse Resource Perspective to Copy Files Needed for C++ and Java Systems

1. Ensure you are in the Eclipse Resource perspective. Typically this is done from the upper right of your Eclipse desktop, but on your first time in you may need to use the Eclipse menu *Window->Open Perspective->Other...* and select **Resource**.
2. In the *ReferenceSimpleOven* project, select the **cpp** subdirectory, right click and select **Copy**.
3. Select the **QuickStart project**, right click and select **Paste**.
4. Repeat this process for the **java** subdirectory.

5. Use the same procedure to copy the *SimpleOvenRhp.pathmate* file into the root of the QuickStart project.
6. Right click **SimpleOvenMD.pathmate** in the QuickStart project. Select **Rename**. Change the name to *Quick Start.pathmate*.

Task 2: Connect PathMATE Project to your new PIM

PROCEDURE: Use Eclipse to Create a New PathMATE Project

1. Double click on **Quick Start.pathmate** to open it.
2. Next to the PIM field, press **Change…**
3. Select **Browse File System…**
4. Browse to and select your MagicDraw Model.
5. If the .mdzip file does not appear, type in *.* and hit enter in the Open field.
6. Press **Open** and then **OK**.
7. In the Confirm PIM Path Change dialog press **Yes**.

Task 3: Check Your Model

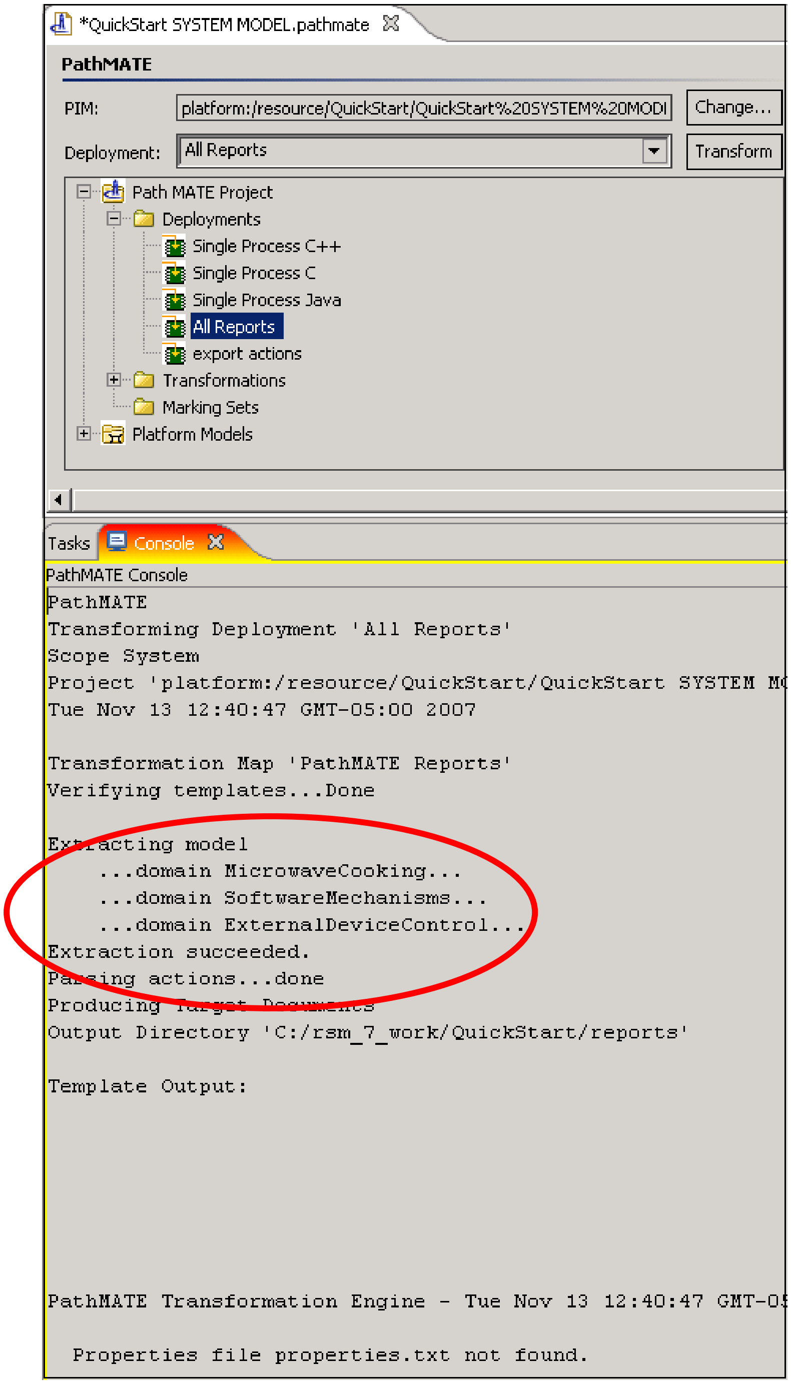
Before going on, this is a good point to check for any errors that may have occurred in entering your model.

We will use the documentation transformation All Reports, and transform to let the PathMATE Transformation Engine automatically validate our model. The transformation maps applied in this procedure can optionally be controlled by markings specified in a properties.txt file, but we will not use one in this case.

PROCEDURE: Use the PathMATE Project Editor to test Your Project

1. Open the Quick Start.pathmate in the editor pane.
2. Select the deployment, **All Reports,** from the Deployment pick list.
3. Click **Transform**. A progress dialog will appear. When it finishes, your Console window will show your results.

4. Verify no model extraction errors, or other errors are listed.

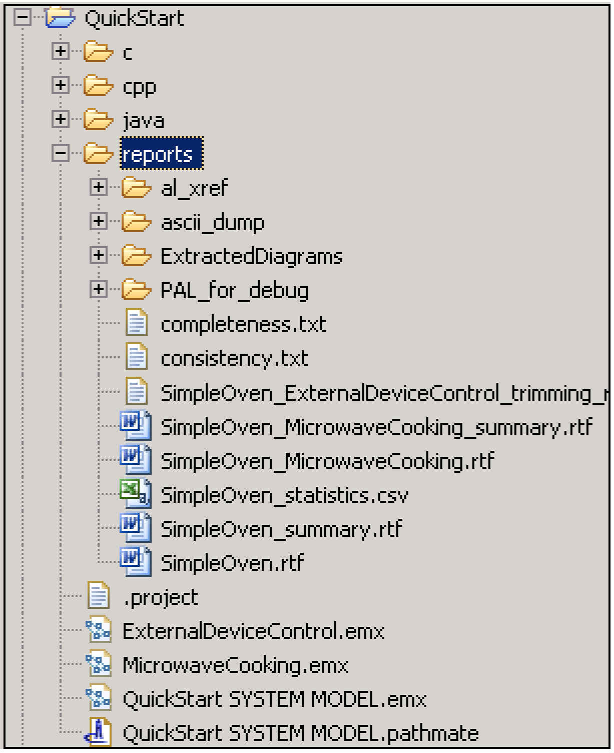


PIM: platform:/resource/QuickStart/QuickStart%20SYSTEM%20MODI

Deployment: All Reports

```
PathMATE
Transforming Deployment 'All Reports'
Scope System
Project 'platform:/resource/QuickStart/QuickStart SYSTEM M
Tue Nov 13 12:40:47 GMT-05:00 2007

Transformation Map 'PathMATE Reports'
Verifying templates...Done

Extracting model
    ...domain MicrowaveCooking...
    ...domain SoftwareMechanisms...
    ...domain ExternalDeviceControl...
Extraction succeeded.
Parsing actions...done
Producing Target Documents
Output Directory 'C:/rsm_7_work/QuickStart/reports'

Template Output:




PathMATE Transformation Engine - Tue Nov 13 12:40:47 GMT-0

  Properties file properties.txt not found.
```

5. Check in the Project Explorer that you have the generated documentation files. If not, you may have a problem with your model.



```
QuickStart
    c
    cpp
    java
    reports
        al_xref
        ascii_dump
        ExtractedDiagrams
        PAL_for_debug
        completeness.txt
        consistency.txt
        SimpleOven_ExternalDeviceControl_trimming_r
        SimpleOven_MicrowaveCooking_summary.rtf
        SimpleOven_MicrowaveCooking.rtf
        SimpleOven_statistics.csv
        SimpleOven_summary.rtf
        SimpleOven.rtf
    .project
    ExternalDeviceControl.emx
    MicrowaveCooking.emx
    QuickStart SYSTEM MODEL.emx
    QuickStart SYSTEM MODEL.pathmate
```

6. Some problems that might turn up are:
   - Extraneous elements that were created in the process of experimentation are not fully specified.
   - Spaces in Names; e.g. “External Device Control” should be “ExternalDeviceControl”.
   - Type not specified for a parameter.
   - Issues with directory structure.
7. Correct any problems identified in the messages, then return to QuickStart System Model.pathmate in the editor pane and click **Transform** again.

Congratulations! You are Ready to Generate Real Code, Real Fast.

Generate C++ Code and Visual Studio Project Files

You are now ready to generate C++ implementation code and Visual Studio project files.

- Configure Development Environment Options
- Transform Your Model to Code and Project Files

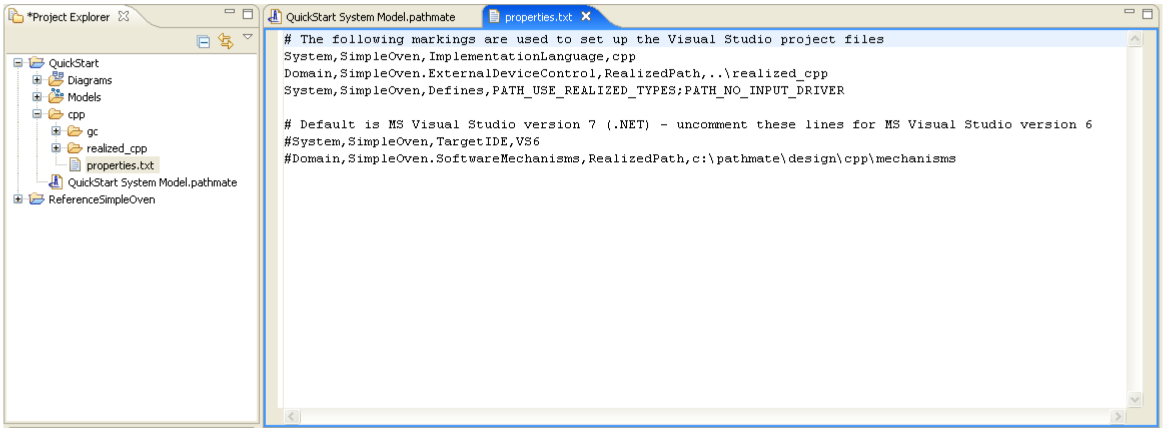
NOTE – if you prefer to generate Java implementation code, please skip ahead to the Generate Java Code section.

Task 1: Configure Development Environment Options

The *markings* file properties.txt controls many aspects of transformation, inclusion of “Execution Instrumentation” in runtime code to support the Spotlight debugger and IDE environment when generating IDE project files. The markings file is located in the project folder for the implementation language you are generating. So for C++ generation, it is in the QuickStart/cpp folder.

PROCEDURE: Use Eclipse Text Editor to Enable Spotlight Instrumentation by Changing “Markings” in properties.txt

1. Double click **properties.txt** to open it.



```
# The following markings are used to set up the Visual Studio project files
System,SimpleOven,ImplementationLanguage,cpp
Domain,SimpleOven.ExternalDeviceControl,RealizedPath,..\realized_cpp
System,SimpleOven,Defines,PATH_USE_REALIZED_TYPES;PATH_NO_INPUT_DRIVER

# Default is MS Visual Studio version 7 (.NET) - uncomment these lines for MS Visual Studio version 6
#System,SimpleOven,TargetIDE,VS6
#Domain,SimpleOven.SoftwareMechanisms,RealizedPath,c:\pathmate\design\cpp\mechanisms
```

2. Add the following as the top line:

```
Domain,SimpleOven.*,SpotlightEnabled,T
```

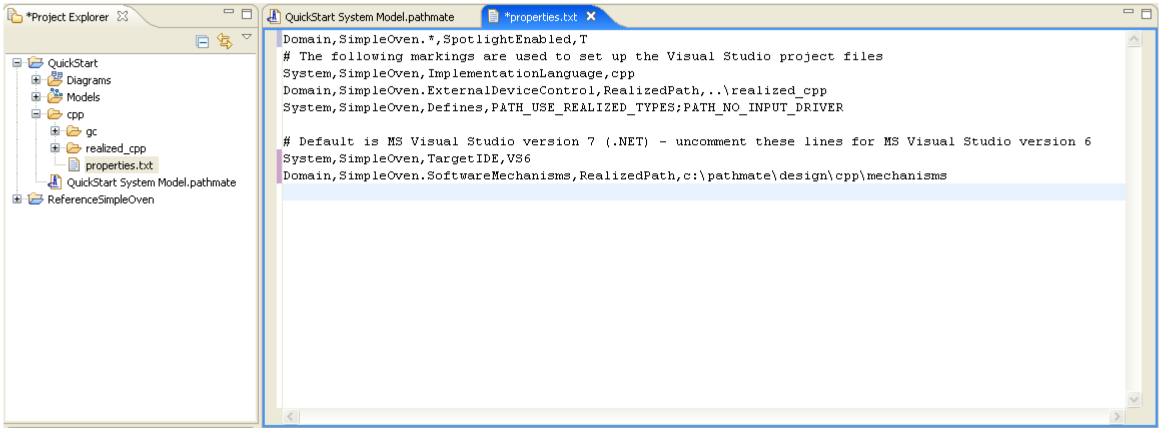
3. Select **File > Save** to save the changes.

PROCEDURE: Change Markings for Visual Studio 6 ONLY

The default target when generating IDE project files is to support Visual Studio version 7 (SimpleOven.vcproj). To generate Visual Studio version 6 project files (SimpleOven.dsp and SimpleOven.dsw) change properties.txt to specify Visual Studio 6 is the Target IDE.

If you are using Visual Studio version 7, skip this procedure.

1. Uncomment the last 2 lines in the file.



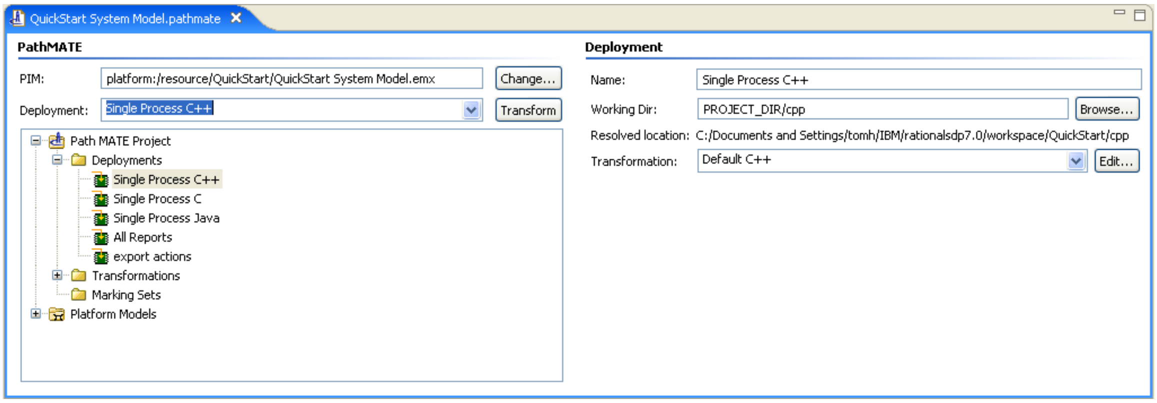
```
Domain,SimpleOven.*,SpotlightEnabled,T
# The following markings are used to set up the Visual Studio project files
System,SimpleOven,ImplementationLanguage,cpp
Domain,SimpleOven.ExternalDeviceControl,RealizedPath,..\realized_cpp
System,SimpleOven,Defines,PATH_USE_REALIZED_TYPES;PATH_NO_INPUT_DRIVER

# Default is MS Visual Studio version 7 (.NET) - uncomment these lines for MS Visual Studio version 6
System,SimpleOven,TargetIDE,VS6
Domain,SimpleOven.SoftwareMechanisms,RealizedPath,c:\pathmate\design\cpp\mechanisms
```

2. Select **File > Save** to save the changes.

Task 2: Transform Your Model to C++ Code and Visual Studio Project Files

PROCEDURE: Generate C++ Implementation Code and a Visual Studio Project File

1. Open *QuickStart.pathmate* in the editor pane.
2. Verify that the deployment *Single Process C++* is selected in the Deployment pick list.



PIM: platform:/resource/QuickStart/QuickStart System Model.emx

Deployment: Single Process C++

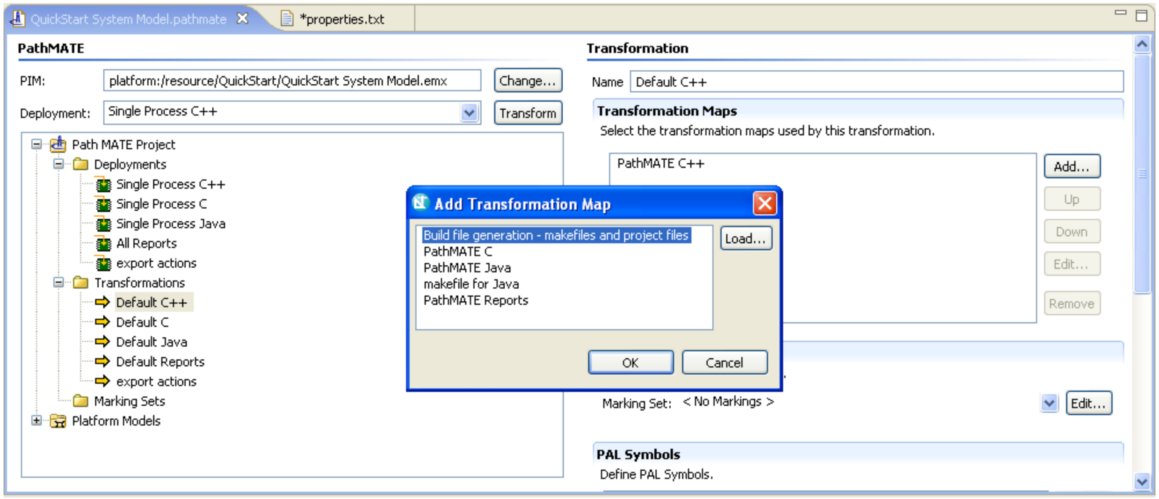
Name: Single Process C++

Working Dir: PROJECT_DIR/cpp

Resolved location: C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp

Transformation: Default C++

3. Click the **Edit…** button to the right of the Transformation drop down. In the Transformation Maps window ensure the *PathMATE C++* and *Build file generation* Maps appear in order. If not, use the Add and/or Up/Down buttons to get both maps to appear in the proper order.



PIM: platform:/resource/QuickStart/QuickStart System Model.emx

Deployment: Single Process C++

Name Default C++

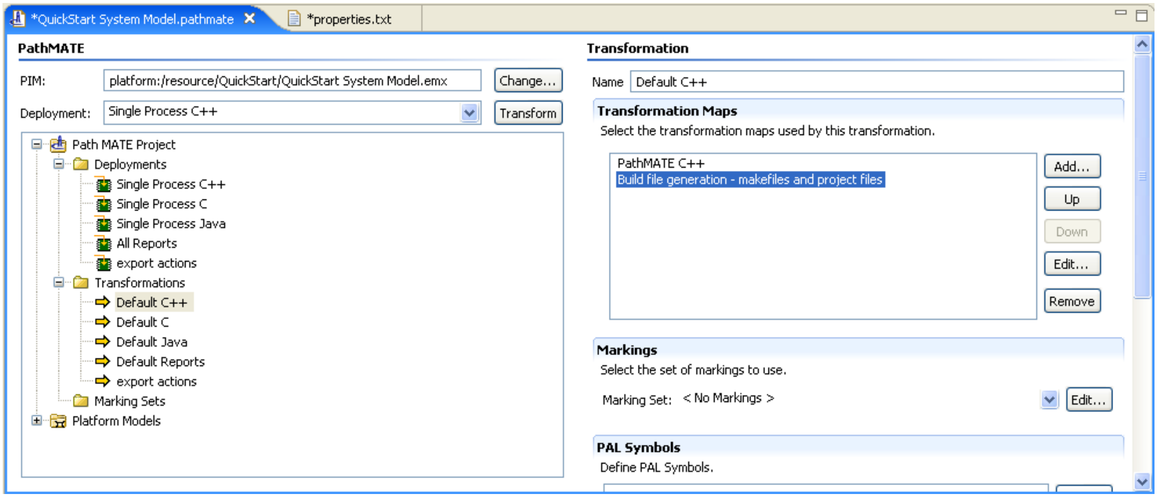
Select the transformation maps used by this transformation.

PathMATE C++

Build file generation - makefiles and project files
PathMATE C
PathMATE Java
makefile for Java
PathMATE Reports

Marking Set: < No Markings >

Define PAL Symbols.



PIM: platform:/resource/QuickStart/QuickStart System Model.emx

Deployment: Single Process C++

Name Default C++

Select the transformation maps used by this transformation.

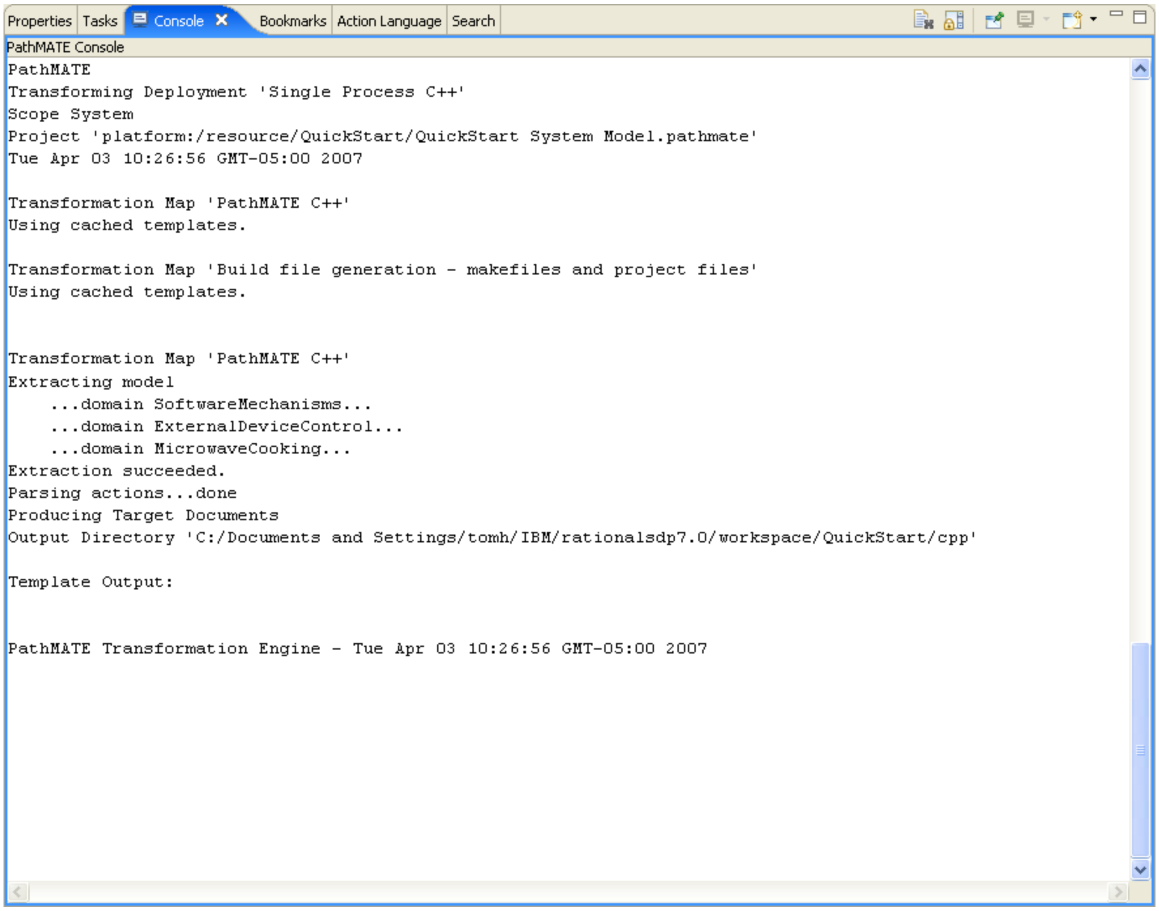
PathMATE C++
Build file generation - makefiles and project files

Select the set of markings to use.

Marking Set: < No Markings >

Define PAL Symbols.

4. Click **Transform**. A progress bar appears.
5. Verify in the Console tab that both transformations were successful. (If any problems arise at this point they are like due to *properties.txt* typos.)



```
PathMATE
Transforming Deployment 'Single Process C++'
Scope System
Project 'platform:/resource/QuickStart/QuickStart System Model.pathmate'
Tue Apr 03 10:26:56 GMT-05:00 2007

Transformation Map 'PathMATE C++'
Using cached templates.

Transformation Map 'Build file generation - makefiles and project files'
Using cached templates.


Transformation Map 'PathMATE C++'
Extracting model
    ...domain SoftwareMechanisms...
    ...domain ExternalDeviceControl...
    ...domain MicrowaveCooking...
Extraction succeeded.
Parsing actions...done
Producing Target Documents
Output Directory 'C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp'

Template Output:


PathMATE Transformation Engine - Tue Apr 03 10:26:56 GMT-05:00 2007
```

Task 3: Build an Executable System

In the resource perspective, browse to QuickStart\cpp\MAIN. To launch the Visual C++ environment and build SimpleOven.exe:

PROCEDURE: Build SimpleOven.exe - Visual Studio Version 7

1. Right-click **SimpleOven.vcproj** under QuickStart, cpp, MAIN in the Project Explorer and select **Open With > System Editor**.
2. Build the SimpleOven system in Visual Studio 7.

PROCEDURE: Build SimpleOven.exe - Visual Studio Version 6

1. Right-click **SimpleOven.dsw** under QuickStart, cpp, MAIN in the Project Explorer and select **Open With > System Editor.**
2. Build the SimpleOven system in Visual Studio 6.

Congratulations! You now have a complete C++ implementation and project file for your system!

Please skip the **Generate Java and Build with Eclipse JDT** section and move ahead to the **Run SimpleOven with Spotlight** section.

Generate Java and Build with Eclipse JDT

The Eclipse Java Development Toolkit (JDT) is included with the Eclipse installation that comes with PathMATE. This section takes you through the following steps:

- Instantiate a Reference Project
- Create a JDT Project to build the generated code
- Create a PathMATE Project
- Transform Your Model
- Build an Executable System

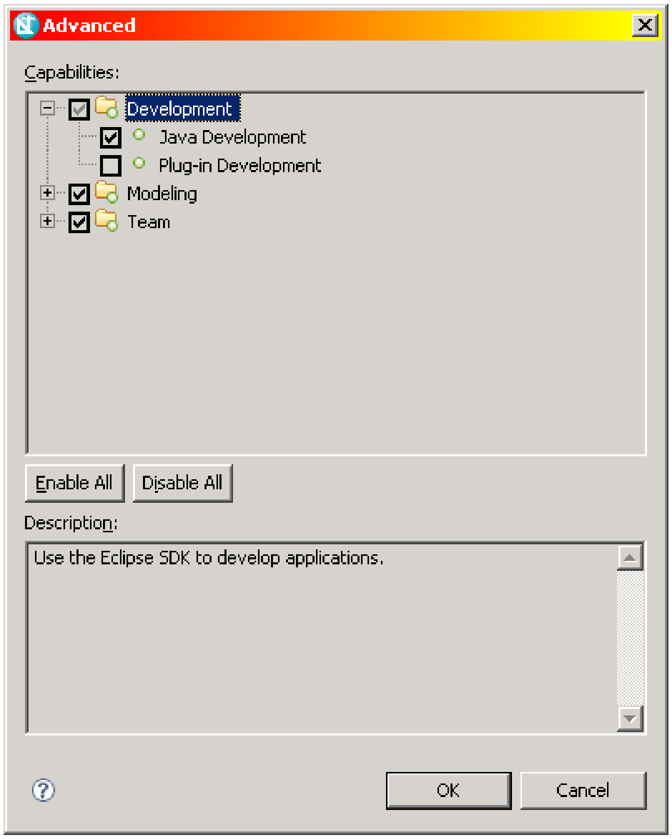
Task 1: Set Up Java and Generate Java Code

Depending on the state of your Eclipse workbench, you may need to activate the JDT capability.

PROCEDURE: Enable Java Development in you Eclipse Workbench

1. Enable the Java Development capability. Select **Window > Preferences…** from the main menu bar. Select **General / Capabilities** from the tree on the left side of the dialog. Click the **Advanced…** button.

2. Expand **Development** and ensure **Java Development** is checked.



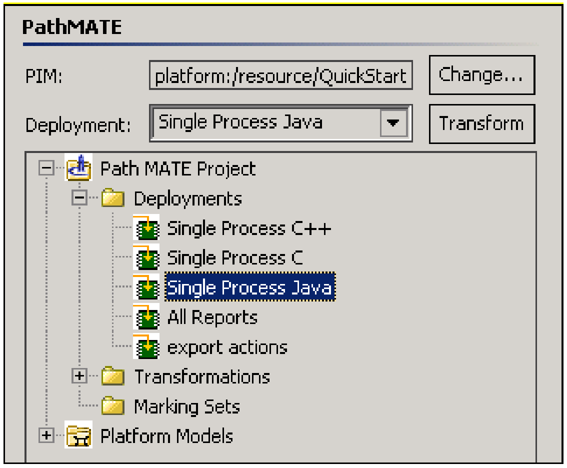
☑ Development
☑ Java Development
□ Plug-in Development
☑ Modeling
☑ Team

Use the Eclipse SDK to develop applications.

3. Click **OK** on both dialogs.

PROCEDURE: Generate Java Code

1. Open your PathMATE project file in your QuickStart project. Select **Single Process Java** under the Deployments. Click **Transform**.

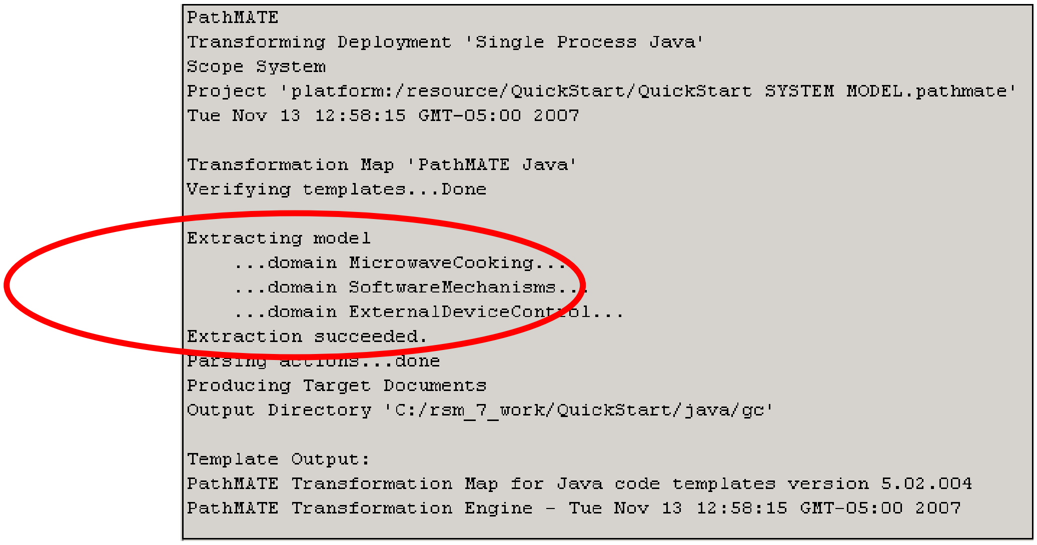


PIM: platform:/resource/QuickStart

Deployment: Single Process Java

Path MATE Project
Deployments
Single Process C++
Single Process C
Single Process Java
All Reports
export actions
Transformations
Marking Sets
Platform Models

2. Open your PathMATE project file in your QuickStart project. Select **Single Process Java** under the Deployments. Click **Transform**.
3. Verify no model extraction errors, or other errors are listed.



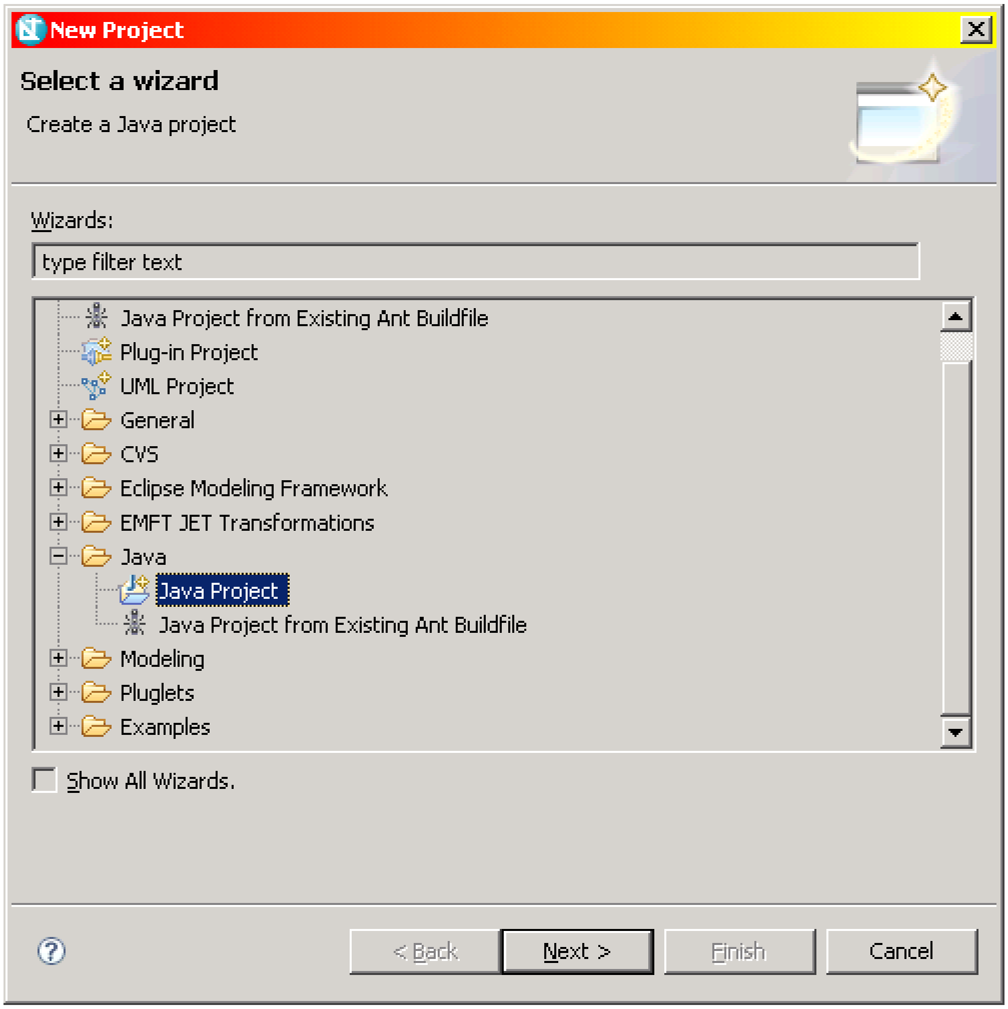
```
PathMATE
Transforming Deployment 'Single Process Java'
Scope System
Project 'platform:/resource/QuickStart/QuickStart SYSTEM MODEL.pathmate'
Tue Nov 13 12:58:15 GMT-05:00 2007

Transformation Map 'PathMATE Java'
Verifying templates...Done

Extracting model
    ...domain MicrowaveCooking...
    ...domain SoftwareMechanisms...
    ...domain ExternalDeviceControl...
Extraction succeeded.
Parsing actions...done
Producing Target Documents
Output Directory 'C:/rsm_7_work/QuickStart/java/gc'

Template Output:
PathMATE Transformation Map for Java code templates version 5.02.004
PathMATE Transformation Engine - Tue Nov 13 12:58:15 GMT-05:00 2007
```

Task 2: Create a Java Project Ready the System for Debug

PROCEDURE: Create a Java Project and Build

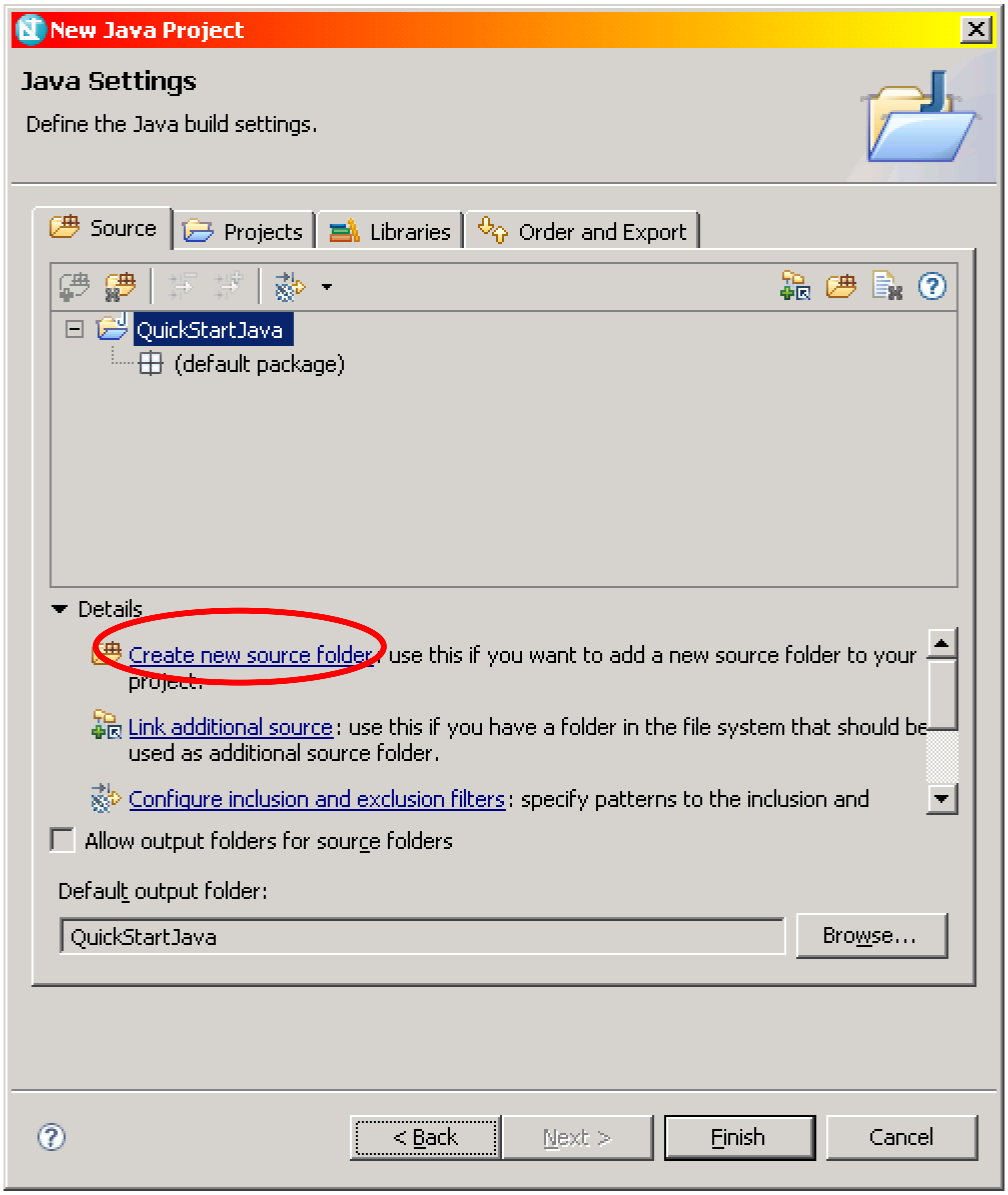
1. From the main Eclipse menu select **File > New > Project**.
2. The New Project Wizard Appears.

3. Select **Java / Java Project** from Wizards.



4. Click **Next**. The Create a Java Project page appears.
5. Enter the name *QuickStartJava*. Click **Next**.

6. Click **Create new source folder**.


New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
QuickStartJava
(default package)
Details
Create new source folder: use this if you want to add a new source folder to your project.
Link additional source: use this if you have a folder in the file system that should be used as additional source folder.
Configure inclusion and exclusion filters: specify patterns to the inclusion and
Allow output folders for source folders
Default output folder:
QuickStartJava
Browse...
< Back
Next >
Finish
Cancel

7. Enter the Folder name *src*. Click **Finish** on both dialogs.

8. In the Package Explorer browser in your *QuickStartJava* project select the *src* source folder, right click and select **New > Package**.

9. Enter *realized_java* for the package name, and click **Finish**.

10. In the Package Explorer browser, navigate in the *QuickStart* project *java/realized_java* folder, select ExternalDeviceControlRealized.java, and hit control-C to copy it.

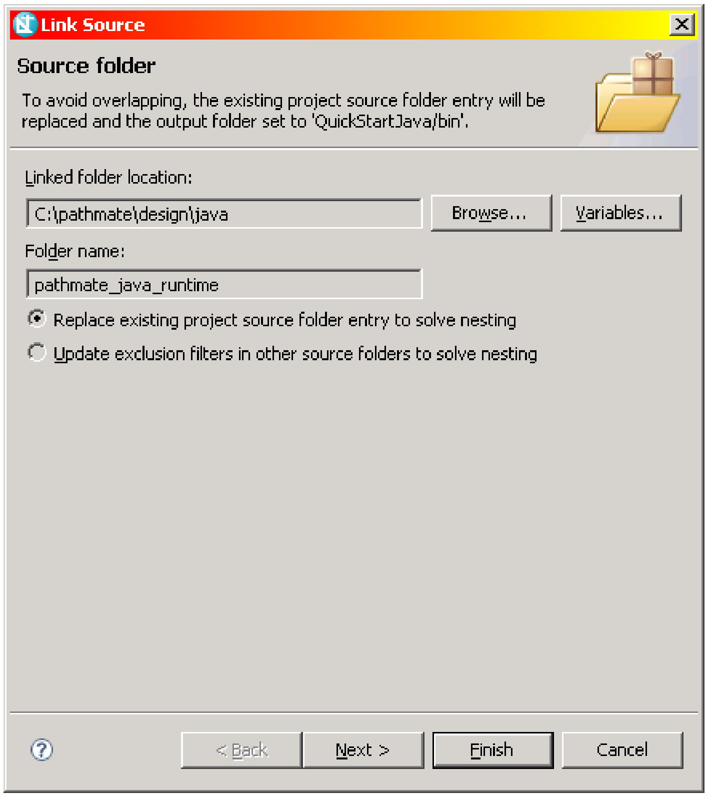
11. Back in your *QuickStartJava* project select **src/realized_java**, and hit control-V to paste in *ExternalDeviceControlRealized.java*.

12. In the Package Explorer browser, select your *QuickStartJava* project, right click, and select **Build Path > Link Source...**

13. Click **Browse...**, navigate to c:\pathmate\design and select the **java** folder.

14. Click **OK**.

15. Enter *pathmate_java_runtime* in Folder name.



Source folder

To avoid overlapping, the existing project source folder entry will be replaced and the output folder set to 'QuickStartJava/bin'.

Linked folder location: C:\pathmate\design\java

Folder name: pathmate_java_runtime

● Replace existing project source folder entry to solve nesting

○ Update exclusion filters in other source folders to solve nesting

4. Click **Finish**.

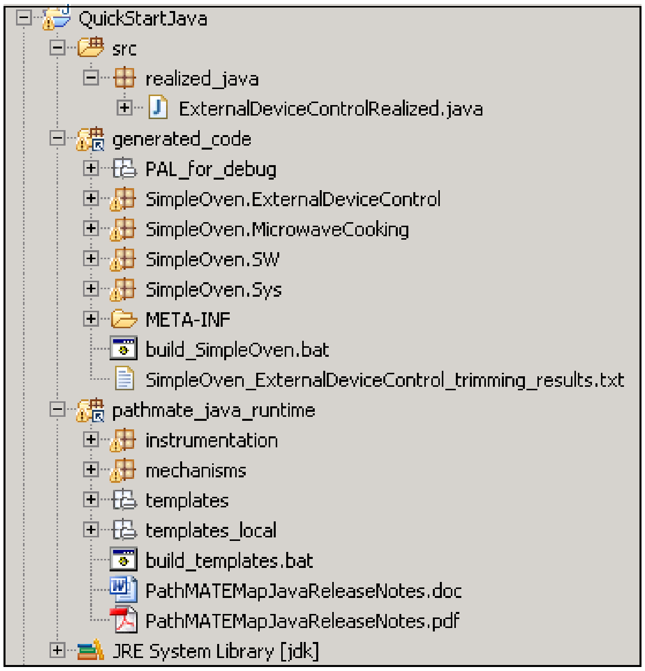
5. Go back to your *QuickStartJava* project, right click, and select **Build Path > Link Source** again.

6. Select **File > Switch Workspace > Other**. Hit control-C to copy the pathname of your current workspace from **Workspace** field. Click **Cancel**.

7. Paste your workspace pathname into the **Linked folder location** field, click **Browse...** and navigate to the generated code in your *QuickStart* project: <your Eclipse workspace>/QuickStart/ java/gc. Name this folder *generated_code*.

8. Click **Finish**.

9. Verify that your *QuickStartJava* project has these contents:



```
QuickStartJava
  src
    realized_java
      ExternalDeviceControlRealized.java
  generated_code
    PAL_for_debug
    SimpleOven.ExternalDeviceControl
    SimpleOven.MicrowaveCooking
    SimpleOven.SW
    SimpleOven.Sys
    META-INF
    build_SimpleOven.bat
    SimpleOven_ExternalDeviceControl_trimming_results.txt
  pathmate_java_runtime
    instrumentation
    mechanisms
    templates
    templates_local
    build_templates.bat
    PathMATEMapJavaReleaseNotes.doc
    PathMATEMapJavaReleaseNotes.pdf
  JRE System Library [jdk]
```

10. At this point your system should have automatically been built. Verify your Problems window reports no Errors. (Some Warnings are expected.)

PROCEDURE: Change Markings and Defines to Enable Spotlight Instrumentation

1. Open **QuickStart/java/properties.txt**. The *properties.txt* file appears in the Editor pane.
2. Add the following to the top of *properties.txt* :

```
Domain,SimpleOven.*,SpotlightEnabled,T
```

3. Select **File > Save** from the main menu to save the changes.

The *markings* file *properties.txt* controls many aspects of transformation, including Spotlight debugger configuration.

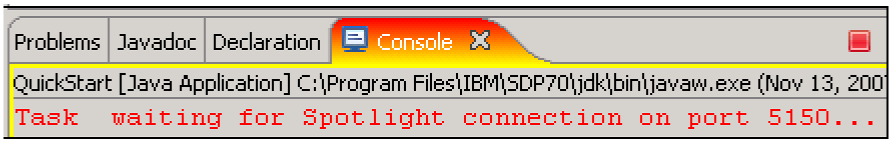
1. In addition to generating instrumentation code (controlled by the SpotlightEnabled marking) you will need to enable Spotlight instrumentation in the runtime layer. In the Package Explorer view, expand **QuickStartJava > pathmate_java_runtime > mechanisms**.
2. Double click to open **PfdDefine.java**. The file opens in the Editor pane. Set the *NO_PATH_IE* to *false* as shown below:

```
public static final boolean NO_PATH_IE = false;
```

3. Select **File > Save** from the main menu to save this change.

PROCEDURE: Create a Debug Configuration and Smoke Test it.

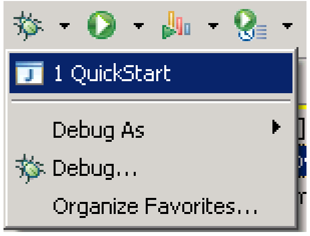
1. Go to **Window / Open Perspective** to switch to the **Debug** perspective.
2. Select **Debug…** from the debug toolbar menu .
3. The Debug dialog appears.
4. Select **Java Application** from the Configurations tree.
5. Right click and select **New**.
6. Enter *QuickStart* in the Name field.
7. Select *QuickStartJava* for the Project if it is not selected already.
8. In the Main Class section, press the **Search…** button. The Select Main Type dialog appears.
9. Select **SimpleOvenApp** from the Matching Types and click **OK**.
10. Click **Debug**. This starts the system. In the Console window you will see the program has started and is trying to connect to Spotlight:



```
QuickStart [Java Application] C:\Program Files\IBM\SDP70\jdk\bin\javaw.exe (Nov 13, 200
Task  waiting for Spotlight connection on port 5150...
```

11. Press the Terminate button . (We will connect to Spotlight in the next section).

To debug this application at a later time, go to the Debug button and select *1 QuickStart*:



Congratulations! You now have a complete Java executable for your system!

Run SimpleOven with Spotlight

Task 1: Start the Simple Oven executable with Spotlight

Run the new program with the Spotlight to visualize SimpleOven system execution at the model level.

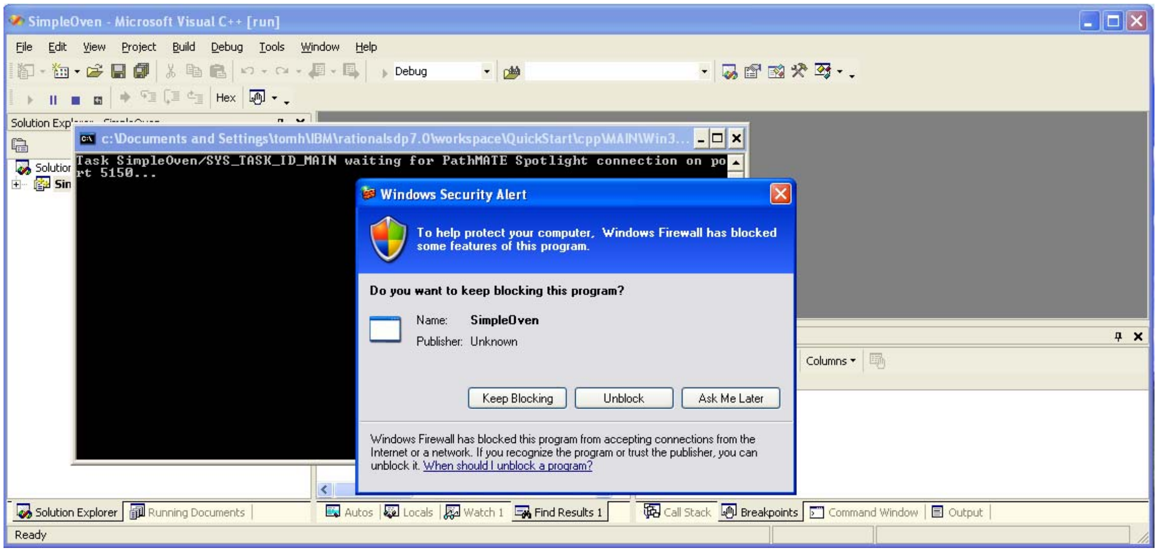
A command window opens to say that the application is running and waiting for a connection to Spotlight.

Please choose the appropriate Procedure below to start the SimpleOven executable, depending on whether you built it in C++ or Java.

PROCEDURE: Run C++ SimpleOven from Visual Studio

If you built Java, please skip this procedure.

1. Launch the application from within Visual Studio (usually **Debug > Start Debugging**, or the **F5** key). A command window opens to say that the application is running and waiting for a connection to Spotlight.



c:\Documents and Settings\tomh\IBM\rationalsdp7.0\workspace\QuickStart\cpp\MAIN\Win3...

```
Task SimpleOven/SYS_TASK_ID_MAIN waiting for PathMATE Spotlight connection on po
rt 5150...
```

Windows Security Alert

To help protect your computer, Windows Firewall has blocked some features of this program.

Do you want to keep blocking this program?

Name: SimpleOven

Publisher: Unknown

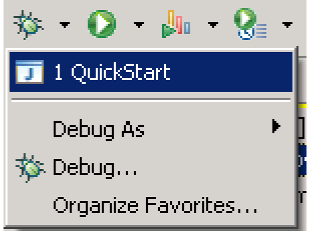
Windows Firewall has blocked this program from accepting connections from the Internet or a network. If you recognize the program or trust the publisher, you can unblock it. When should I unblock a program?

2. If a Windows Security Alert appears, click **Unblock**.
3. Skip ahead to PROCEDURE: Use Eclipse to Launch the Spotlight Debugger.

PROCEDURE: Run Java SimpleOven from the Eclipse JDT

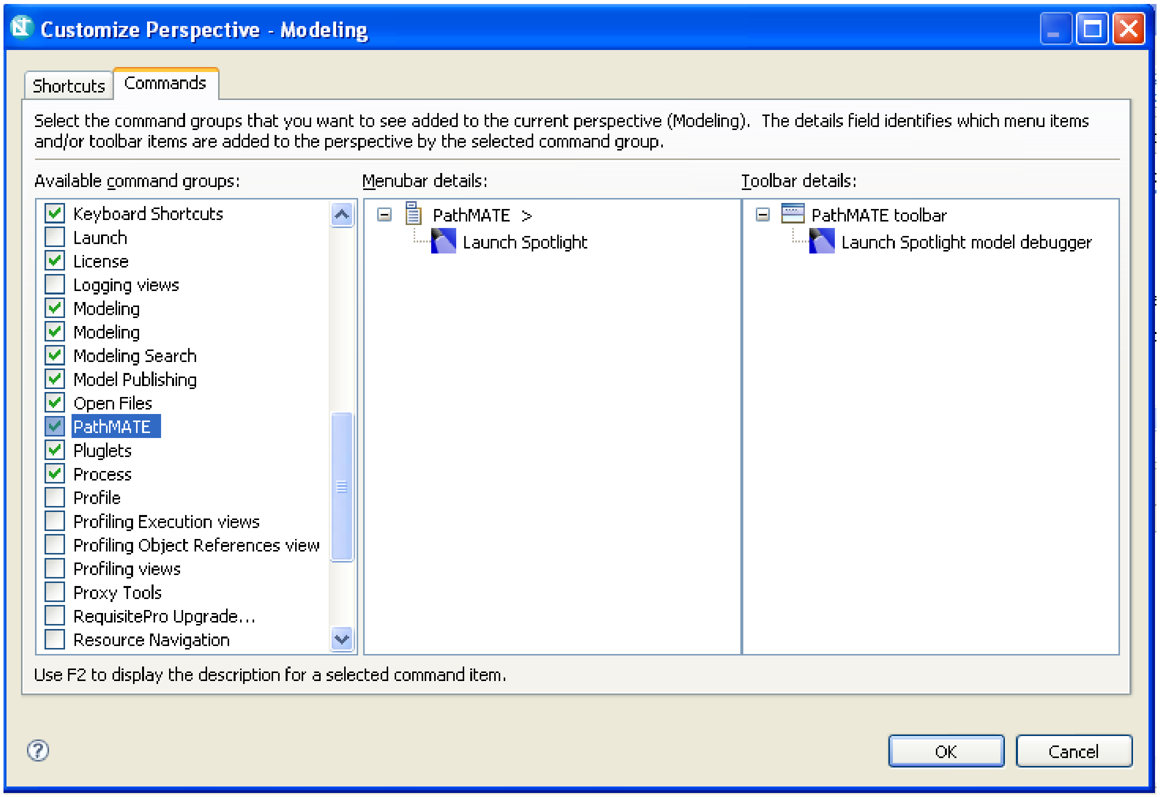
If you built C++, please skip this procedure.

1. Go to the Debug button and select **1 QuickStart.**



PROCEDURE: Use Eclipse to Launch the Spotlight Debugger

1. If a PathMATE menu does not appear on the top menu bar, open the **Customize Perspective…** option in the Window menu. Select the **Commands** tab and check the **PathMATE** command group on the Commands tab:



Select the command groups that you want to see added to the current perspective (Modeling). The details field identifies which menu items and/or toolbar items are added to the perspective by the selected command group.

Available command groups:
- ☑ Keyboard Shortcuts
- ☐ Launch
- ☑ License
- ☐ Logging views
- ☑ Modeling
- ☑ Modeling
- ☑ Modeling Search
- ☑ Model Publishing
- ☑ Open Files
- ☑ PathMATE
- ☑ Pluglets
- ☑ Process
- ☐ Profile
- ☐ Profiling Execution views
- ☐ Profiling Object References view
- ☐ Profiling views
- ☐ Proxy Tools
- ☐ RequisitePro Upgrade...
- ☐ Resource Navigation

Menubar details:
- PathMATE >
  - Launch Spotlight

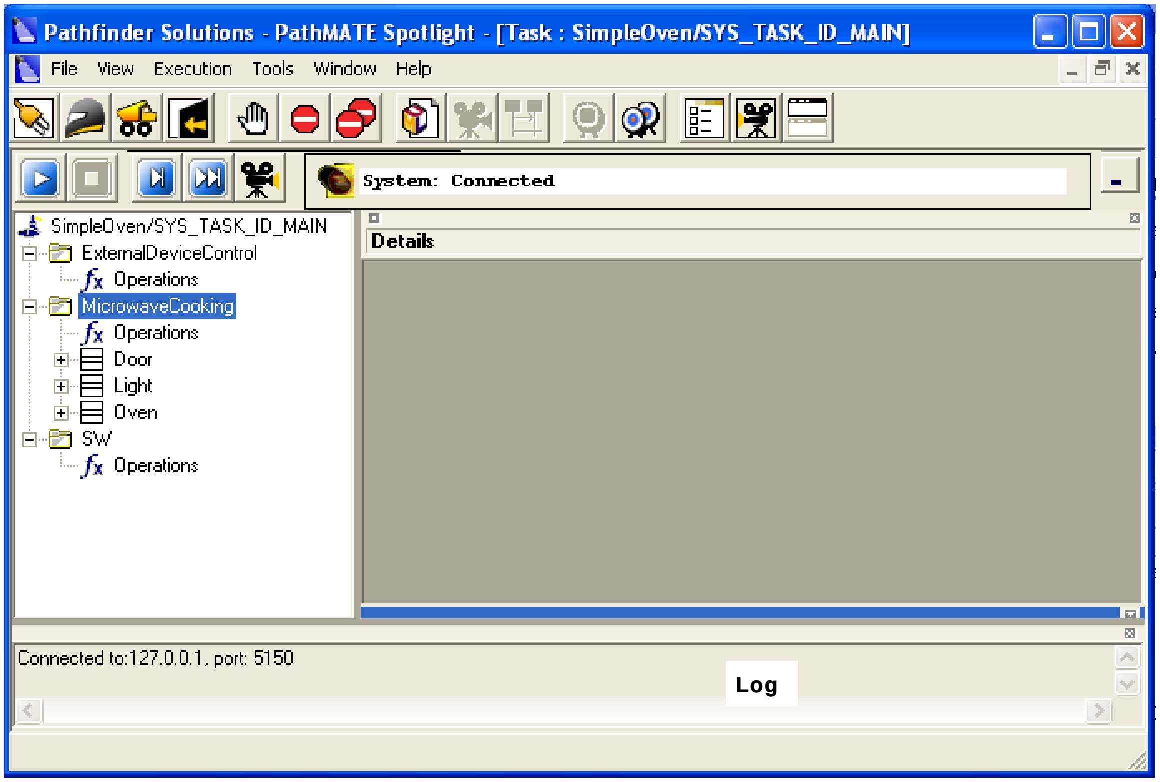
Toolbar details:
- PathMATE toolbar
  - Launch Spotlight model debugger

Use F2 to display the description for a selected command item.

2. Click **OK.**
3. To launch Spotlight, pick **PathMATE > Launch Spotlight** or choose the Eclipse toolbar **Launch Spotlight** button ( ).

PROCEDURE: Connect Spotlight to the Running SimpleOven Program

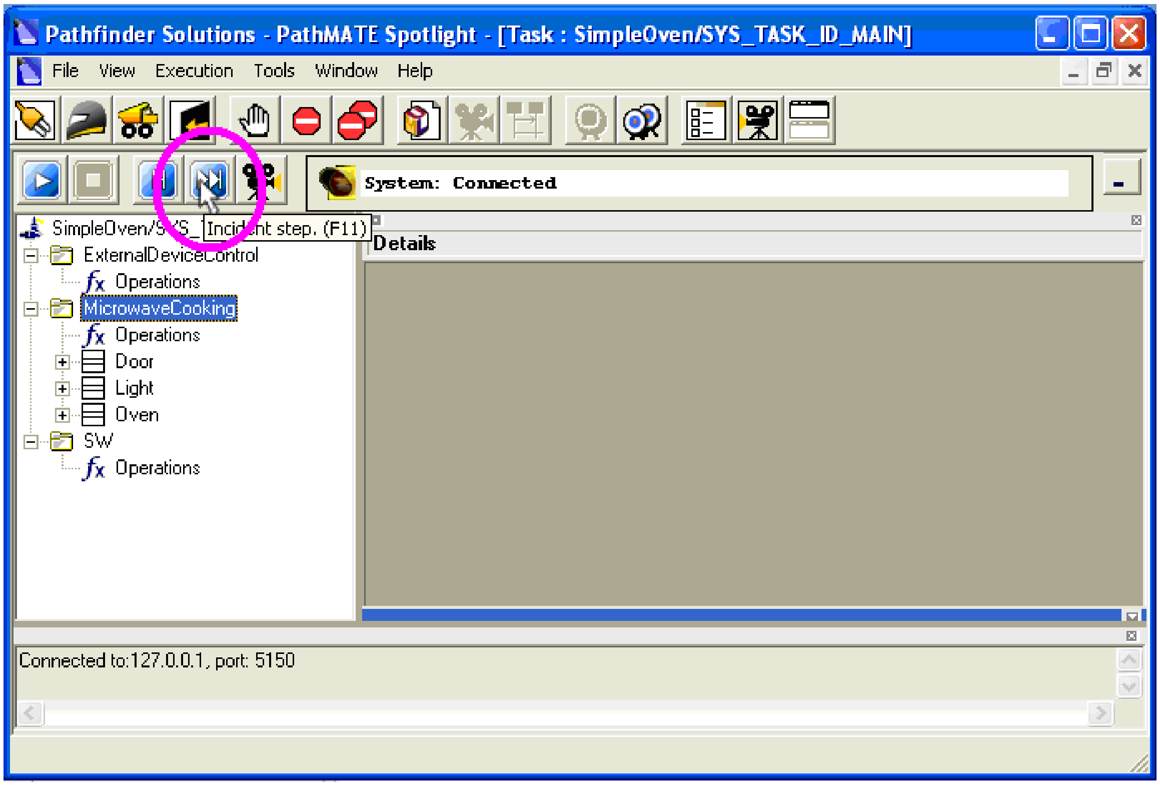
1. Once Spotlight starts, click the **Connect** button at the left end of the Spotlight toolbar to connect Spotlight to the target application.
2. Check to see that Spotlight is successfully connected. The three domains in *SimpleOven* appear in the browser on the left, and the status bar indicates *System: Connected*.
3. Expand the domains:



```
Connected to:127.0.0.1, port: 5150
```

PROCEDURE: Use Spotlight to Initialize the SimpleOven System

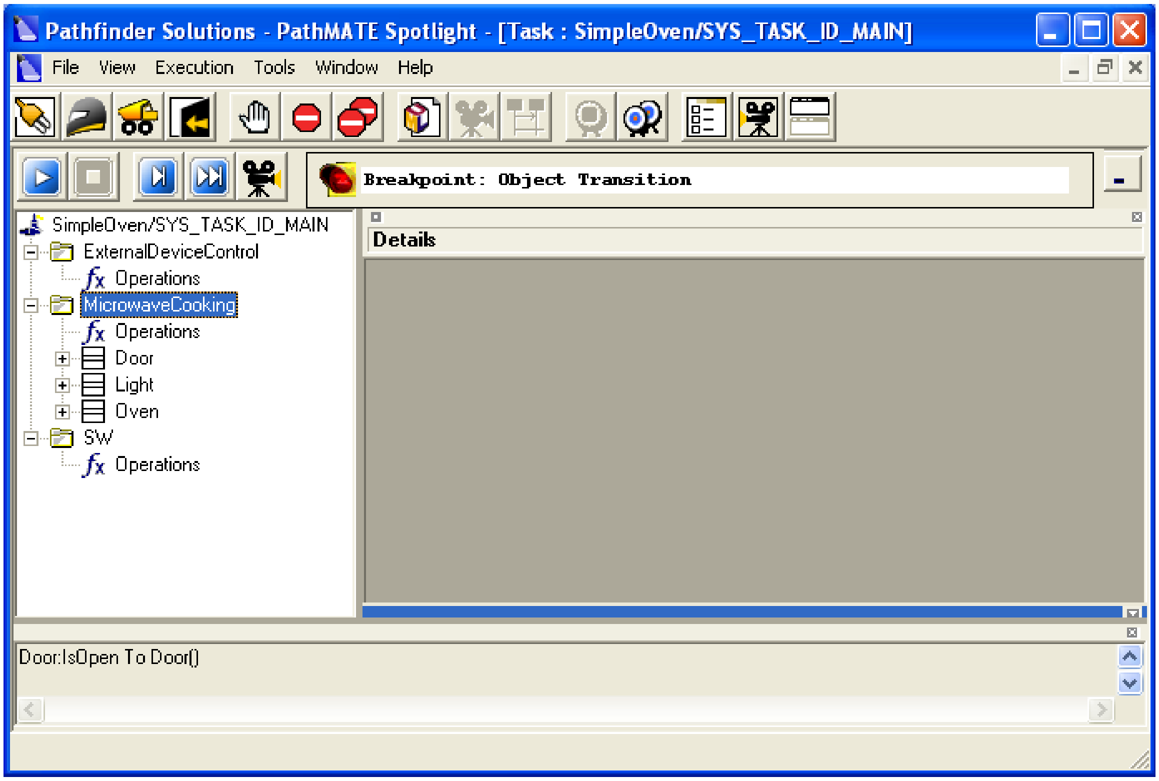
1. Locate the Incident Step button .



System: Connected

Connected to:127.0.0.1, port: 5150

2. Click the **Incident Step button** ⏭. The status indicator changes from *Connected* to *Object Transition*.



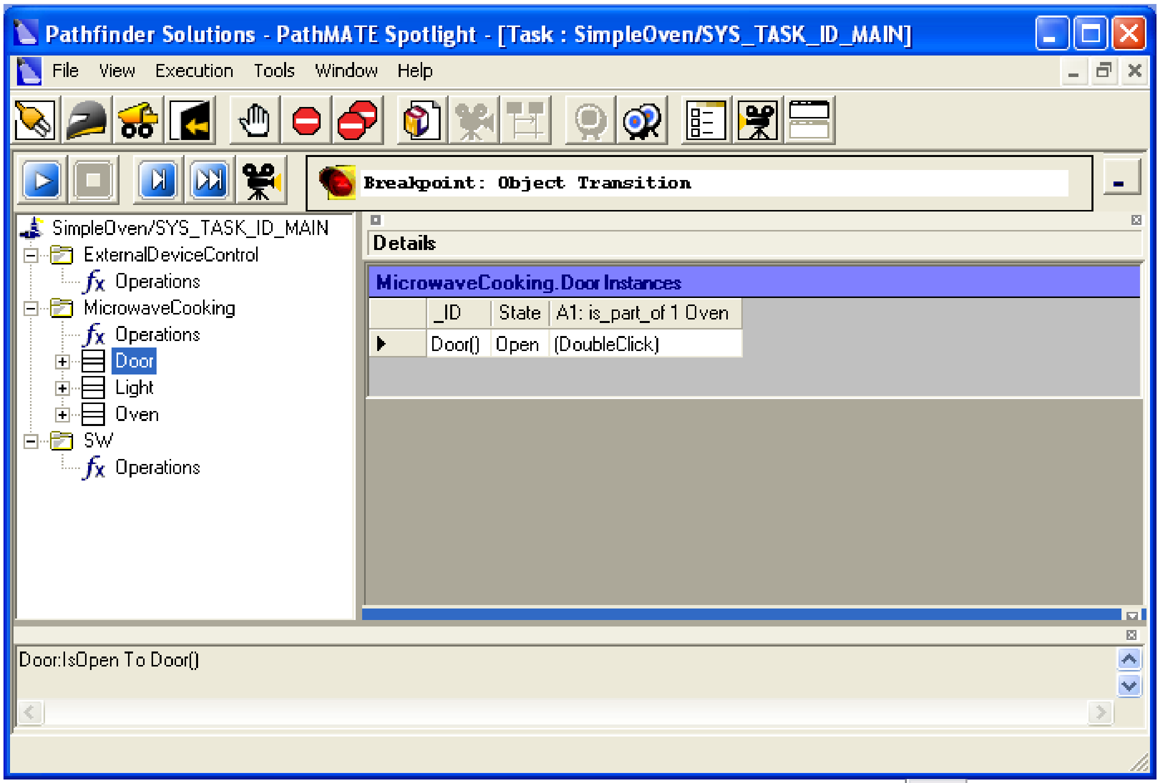
Breakpoint: Object Transition

Door:IsOpen To Door()

PROCEDURE: Use Spotlight to Browse the Door Class

1. Select the **Door** class in the browser.

The details pane shows one instance of the Door class. The door object is in the Open state, and it is part of one oven (association *A1*).



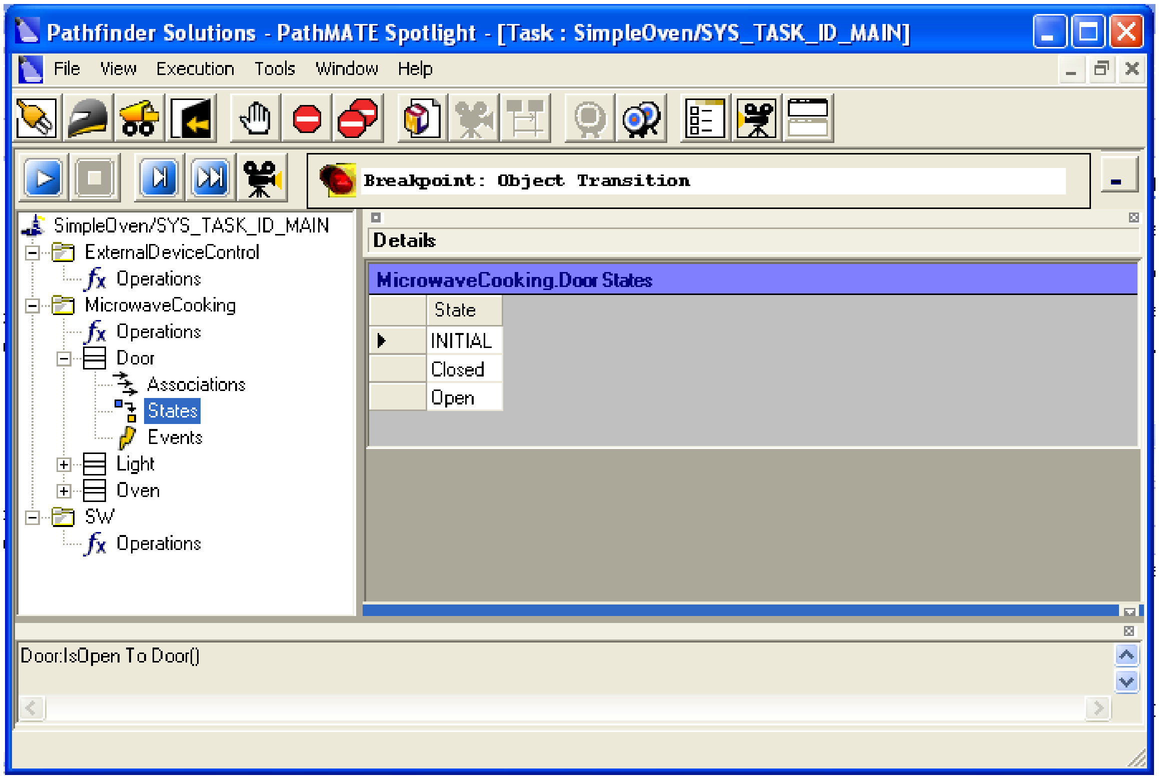
Breakpoint: Object Transition

MicrowaveCooking.Door Instances

	_ID	State	A1: is_part_of 1 Oven
▶	Door()	Open	(DoubleClick)

Door:IsOpen To Door()

2. Click the **Incident Step button** ⏭ to cause the Domain Initialization action to be executed and first Event to be processed.
3. Expand the **Door** class in the browser and review the run-time state of this class. Note the Door is in the Open State. Recall that the only transition out of the initial state is to the Closed state. To see why is the Door in the Open State, review the domain initialization action in the model under *<<Housekeeping>> Domain Support* for *<<Domain>>MicrowaveCooking* in the Project Explorer.



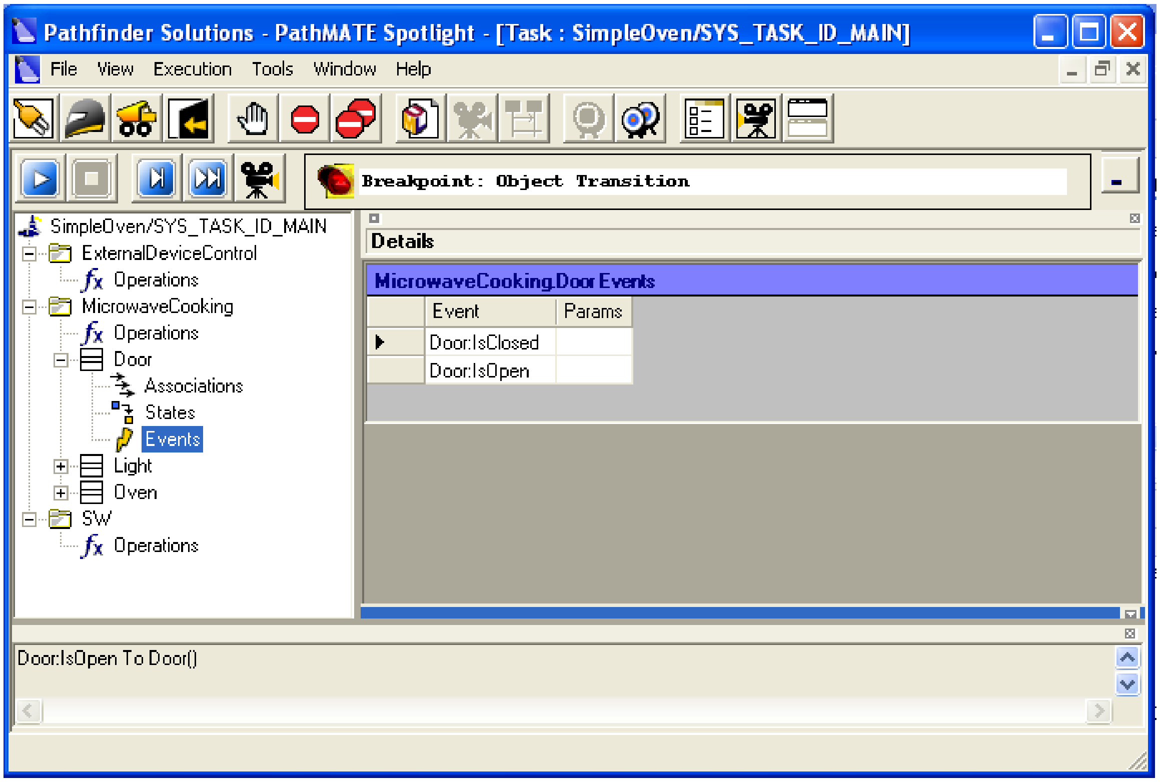
Breakpoint: Object Transition

MicrowaveCooking.Door States

	State
▶	INITIAL
	Closed
	Open

```
Door:IsOpen To Door()
```

4. Select **Events** in the browser. The details pane shows the events that you can send to the *Door* class.

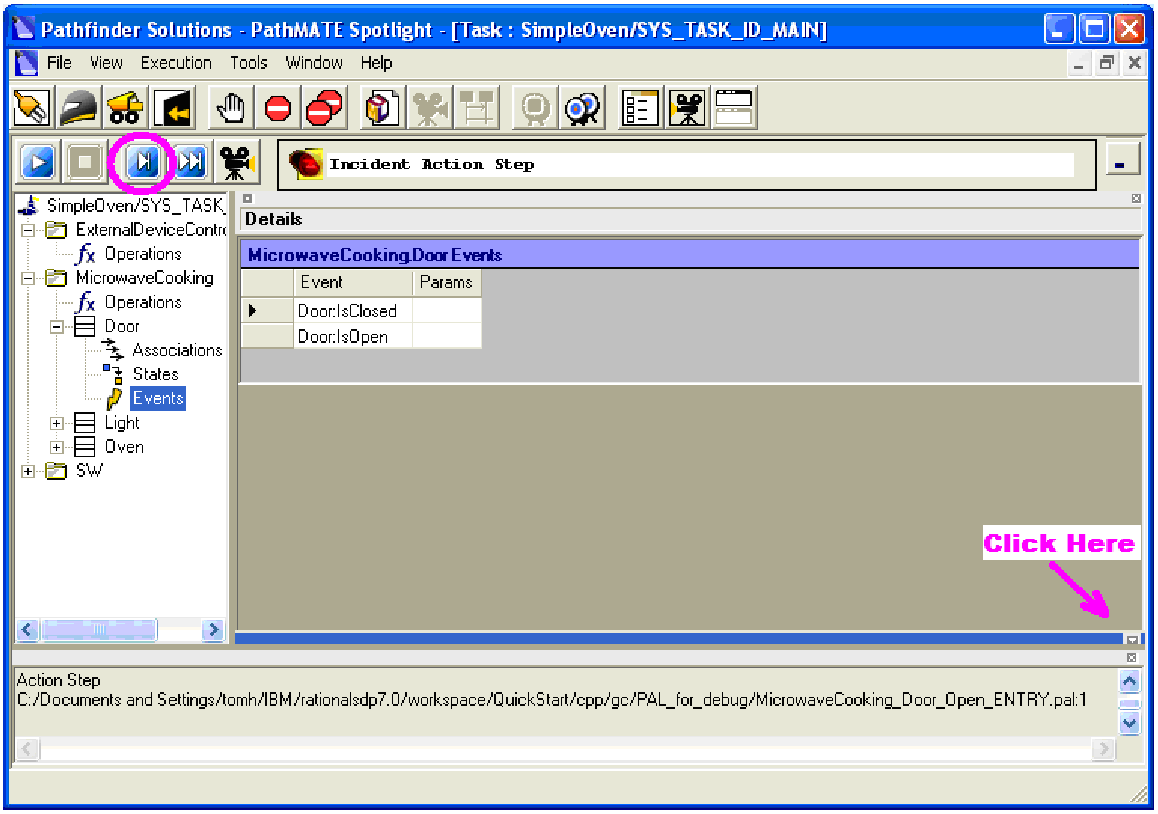

Pathfinder Solutions - PathMATE Spotlight - [Task : SimpleOven/SYS_TASK_ID_MAIN]
File View Execution Tools Window Help
Breakpoint: Object Transition
SimpleOven/SYS_TASK_ID_MAIN
ExternalDeviceControl
Operations
MicrowaveCooking
Operations
Door
Associations
States
Events
Light
Oven
SW
Operations
Details
MicrowaveCooking.Door Events

Event	Params
Door:IsClosed	
Door:IsOpen	

Door:IsOpen To Door()

PROCEDURE: Use Spotlight to step through SimpleOven's Action Language

1. Click the **Action Step button** ⏭. *MicrowaveCooking_Door_Open.pal* appears in the state window.
2. Open the **State Window**. The figure below shows where to click in the lower right to open and close the action/event window.

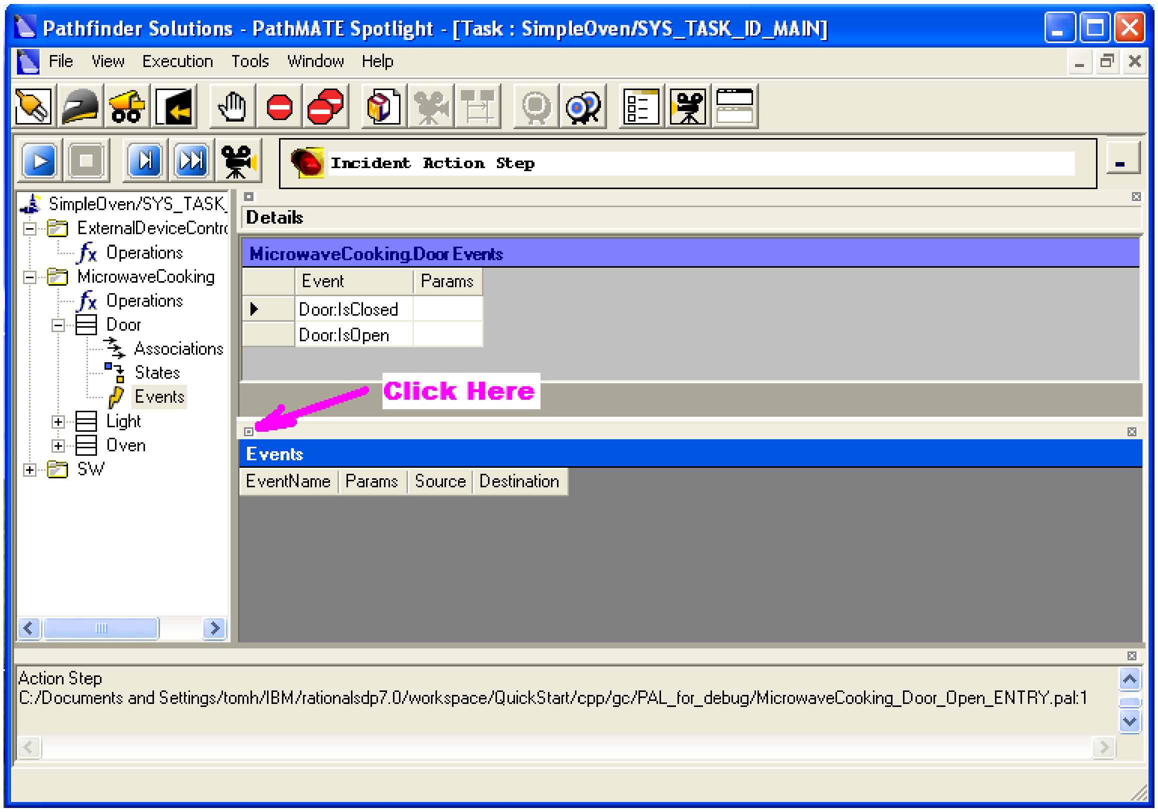


MicrowaveCooking.Door Events

	Event	Params
▶	Door:IsClosed	
	Door:IsOpen	

```
Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Door_Open_ENTRY.pal:1
```

3. Select the **Action Language view**. The figure below shows how to toggle between action language and the event queue.



MicrowaveCooking.Door Events

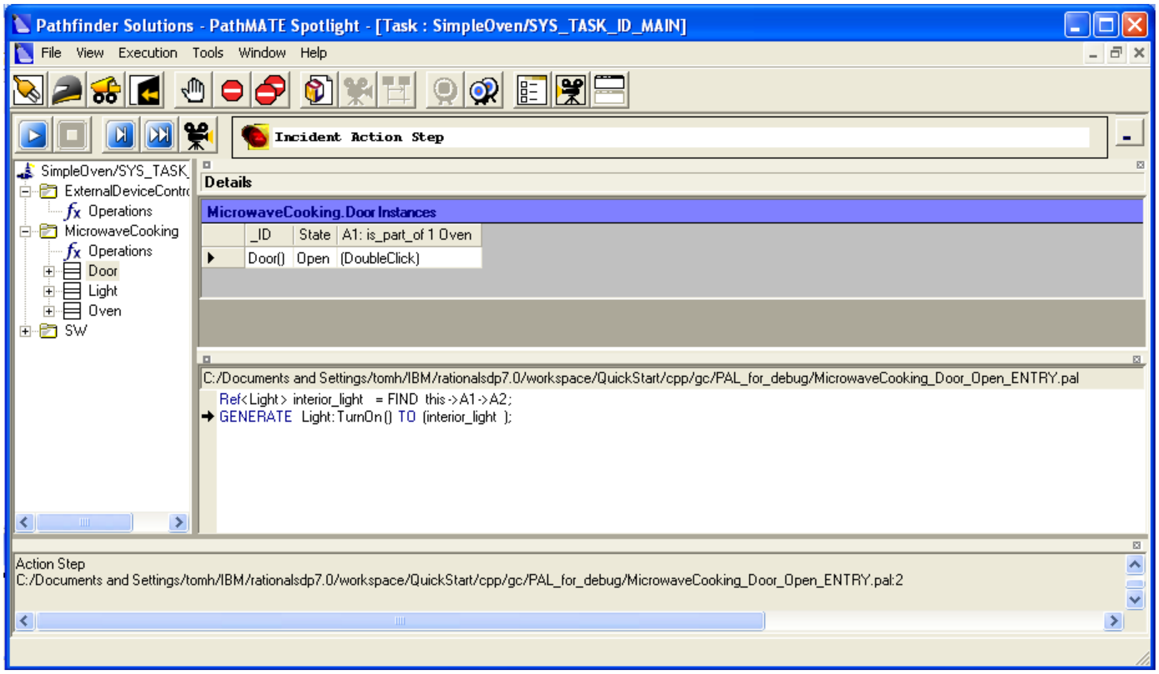
	Event	Params
▶	Door:IsClosed	
	Door:IsOpen	

Click Here

EventName	Params	Source	Destination

Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Door_Open_ENTRY.pal:1

4. Click **Action Step** again . The active step arrow advances to the next PAL statement.


Pathfinder Solutions - PathMATE Spotlight - [Task : SimpleOven/SYS_TASK_ID_MAIN]
File View Execution Tools Window Help
Incident Action Step
SimpleOven/SYS_TASK_
ExternalDeviceContr
Operations
MicrowaveCooking
Operations
Door
Light
Oven
SW
Details

MicrowaveCooking.Door Instances

	_ID	State	A1: is_part_of 1 Oven
▶	Door()	Open	(DoubleClick)

C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Door_Open_ENTRY.pal

```
  Ref<Light> interior_light  = FIND this->A1->A2;
➜ GENERATE Light:TurnOn() TO (interior_light );
```

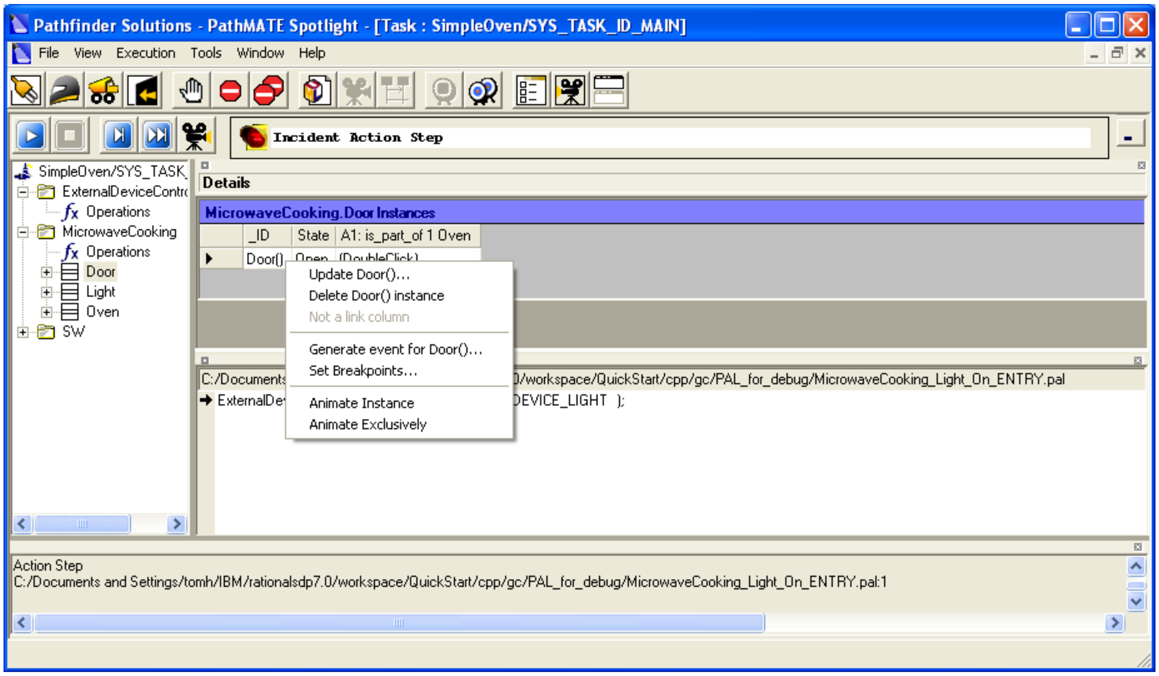
Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Door_Open_ENTRY.pal:2

5. Continue to click the **Action Step button** until *MicrowaveCooking_Light_On_entry.pal* appears in the PAL viewer.

(DO NOT CLICK AGAIN. If you do, you will queue up an action. Incident Action Step will change to Action Step Pending and the Red stoplight will change to Green. If this happens, all is not lost, but the Event Processing in the next procedure will proceed immediately on the event hitting the queue.)

PROCEDURE: Use Spotlight to Send Event (Signal) to SimpleOven

1. Select **Door** in the browser. Right-click the **Door***()* field in the details pane to bring up a context-sensitive menu:



Incident Action Step

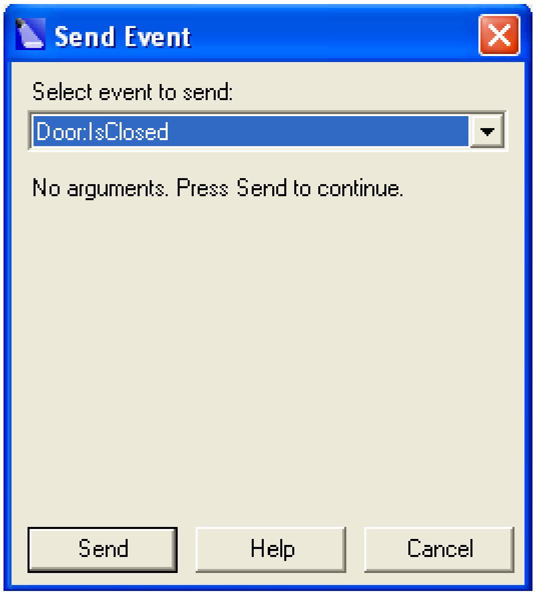
MicrowaveCooking.Door Instances

	_ID	State	A1: is_part_of 1 Oven
▶	Door()	Open	(DoubleClick)

```
C:/Documents                    0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Light_On_ENTRY.pal
➔ ExternalDe                    DEVICE_LIGHT  );
```

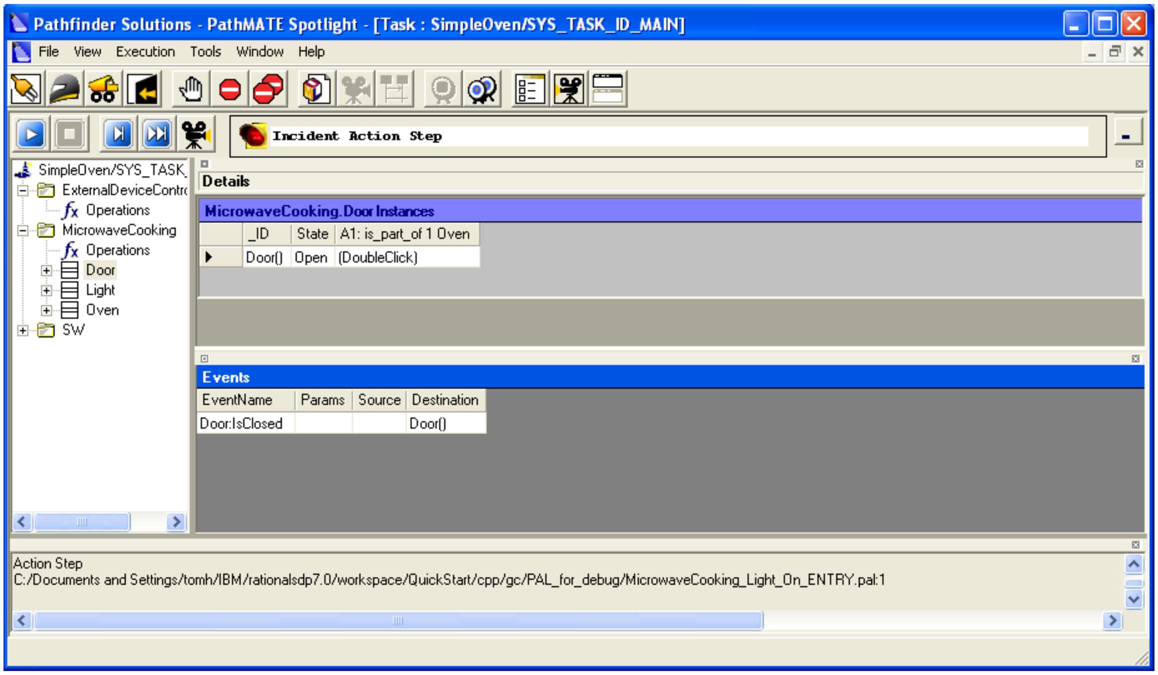
```
Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Light_On_ENTRY.pal:1
```

2. Select **Generate event for Door()** in the pop-up menu. The Send Event dialog opens. Select **Door:IsClosed** in the drop-down list.



3. Click **Send**.

4. Toggle from the Action Language viewer to the event queue in the lower pane. The event *Door:IsClosed* re queued to be sent to Door(). The event appears in the queue under Events in the lower pane.



MicrowaveCooking.Door Instances

	_ID	State	A1: is_part_of 1 Oven
▶	Door()	Open	(DoubleClick)

Events

EventName	Params	Source	Destination
Door:IsClosed			Door()

Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Light_On_ENTRY.pal:1

5. Press the Spotlight **Go button** ▶ to resume execution, and allow *SimpleOven* to process the new event.

6. You may choose to continue execution by resending *Door:IsOpen* and *Door:IsClosed* events.

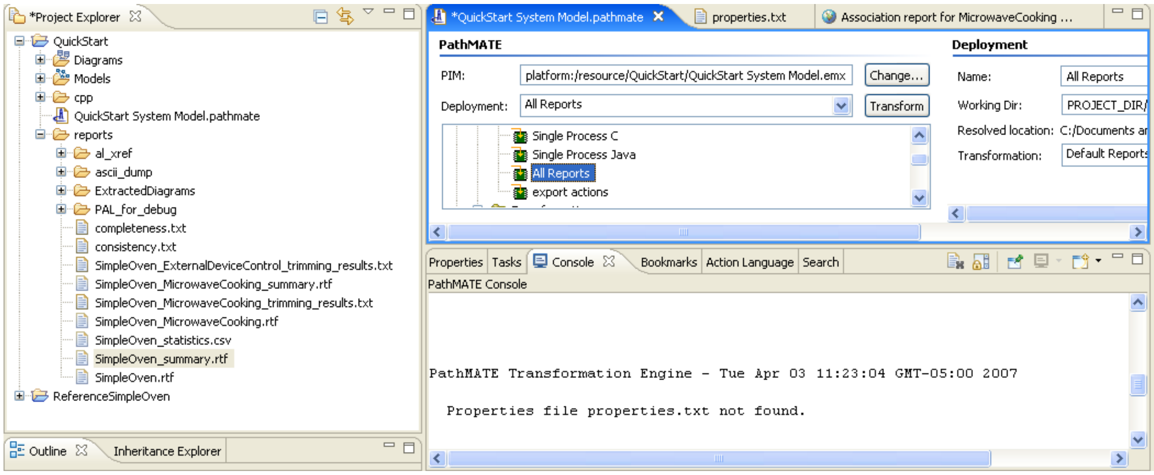
Generate System Documentation

PROCEDURE: Generate Documentation

Return to PathMATE and transform:

1. Select the **All Reports** deployment:
2. Select **Transform**.

From the resource perspective explore the generated reports in the reports folder. (Note: a benign warning "Properties file properties.txt not found." will be written to the Eclipse Console View.)



PathMATE

PIM: platform:/resource/QuickStart/QuickStart System Model.emx

Deployment: All Reports

Deployment

Name: All Reports

Working Dir: PROJECT_DIR/

Resolved location: C:/Documents ar

Transformation: Default Reports

PathMATE Console

```
PathMATE Transformation Engine - Tue Apr 03 11:23:04 GMT-05:00 2007

 Properties file properties.txt not found.
```

Generated documentation files:

Report Title	Filename
Domain Report for SimpleOven	SimpleOven_summary.rtf
Full Analysis Report for SimpleOven	SimpleOven.rtf
Class Modeling Report for SimpleOven.MicrowaveCooking	SimpleOven_MicrowaveCooking_summary.rtf
State Modeling report for SimpleOven.MicrowaveCooking	SimpleOven_MicrowaveCooking.rtf

3. To view any of the generated files, right-click on the file in the Project Explorer and select **Open With > System Editor**.

Congratulations!

You have documentation and reports for your QuickStart system.

You have now completed the Quick Start.

Summary

PathMATE separates the feature logic of a system from the details of its implementation on a specific platform. That separation yields simplicity, facilitating the solution of each aspect of the system in relative isolation from the others. The simplicity of platform independent models yields substantial benefits all across the development cycle. With PathMATE you will:

- Improve productivity in your initial development effort.
- React quickly to changing software and hardware requirements.
- Test integration of all system components at the model level, much earlier in the development cycle.
- Substantially reduce the time required for debugging.
- Improve reliability and performance.
- Consistently meet your deadlines.

Additionally, the use of platform-independent action language gives PathMATE a complete semantic awareness of everything that your system will do, allowing it to do Self Optimization of your system as it generates your implementation code. This yields substantial performance gains over code generated from code-in-the-model approaches.

Pathfinder's tools help you transform your models into executable code, predictably and accurately. Deploy faster, highly reliable embedded systems much more quickly than you thought possible.

Next Steps

The white papers at www.pathfindermda.com contain additional information. Most importantly, try the PathMATE toolset with real code, as outlined in this *Guide*. We can help you get started.

Pathfinder Solutions

Headquartered in Foxboro, Massachusetts, Pathfinder Solutions provides embedded software engineers with the tools, methods and services needed to reduce development costs and improve quality. Pathfinder Solutions is an active member of the Object Management Group (OMG).

If you would like to learn more, please contact us at:

Pathfinder Solutions
33 Commercial Street, Suite 2
Foxboro, MA 02035 USA
Phone: 888-662-7284 (+1 508-568-0068)
Email: info@pathfindermda.com

Acronyms

Acronym	Definition
JDT	Java Developer Toolkit
MDA	Model Driven Architecture
MDD	Model Driven Development
OMG	Object Management Group
PAL	Platform-independent Action Language
PathMATE	Pathfinder Model Automation and Transformation Environment
PIM	Platform Independent Model
UML	Unified Modeling Language