Pathfinder Solutions

PathMATE™ for IBM Rational Software Modeler Version 7

Quick Start Guide

Version 8.0.2

September 21, 2009

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 IBM Rational Software Modeler (RSM) 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 MDA 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 IBM Rational Software Modeler 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

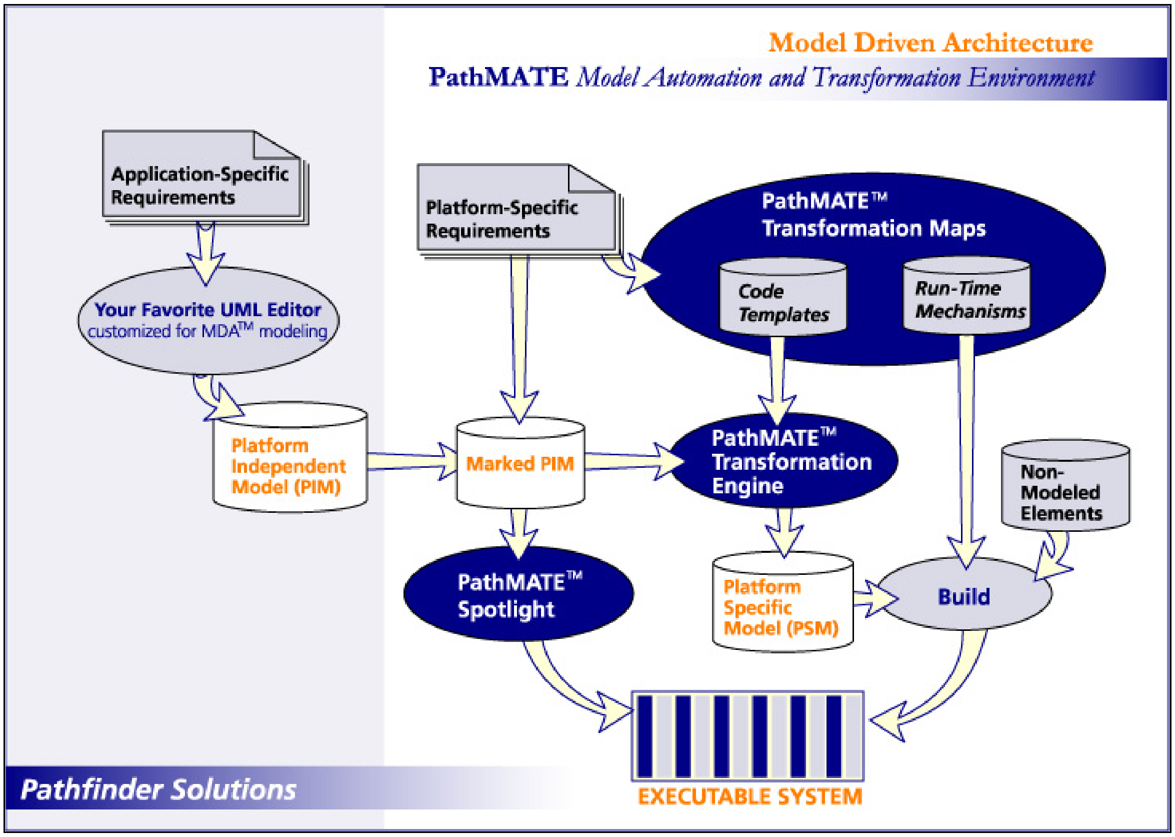
To complete the steps in this guide, you'll need the following software on your computer:

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

PathMATE MDA Toolset

Through its *Architectural Focus* and advanced automation, PathMATE is the leading MDD environment for embedded, real-time systems development with its PathMATE MDA toolset. This powerful technology executes and tests platform-independent UML 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

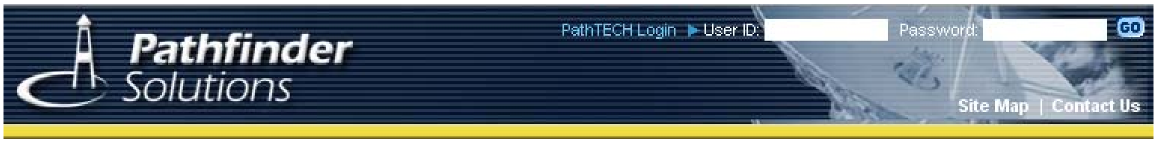
The PathMATE MDA Toolset is comprised of three parts which work together to turn your models into executable embedded systems:

- *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.

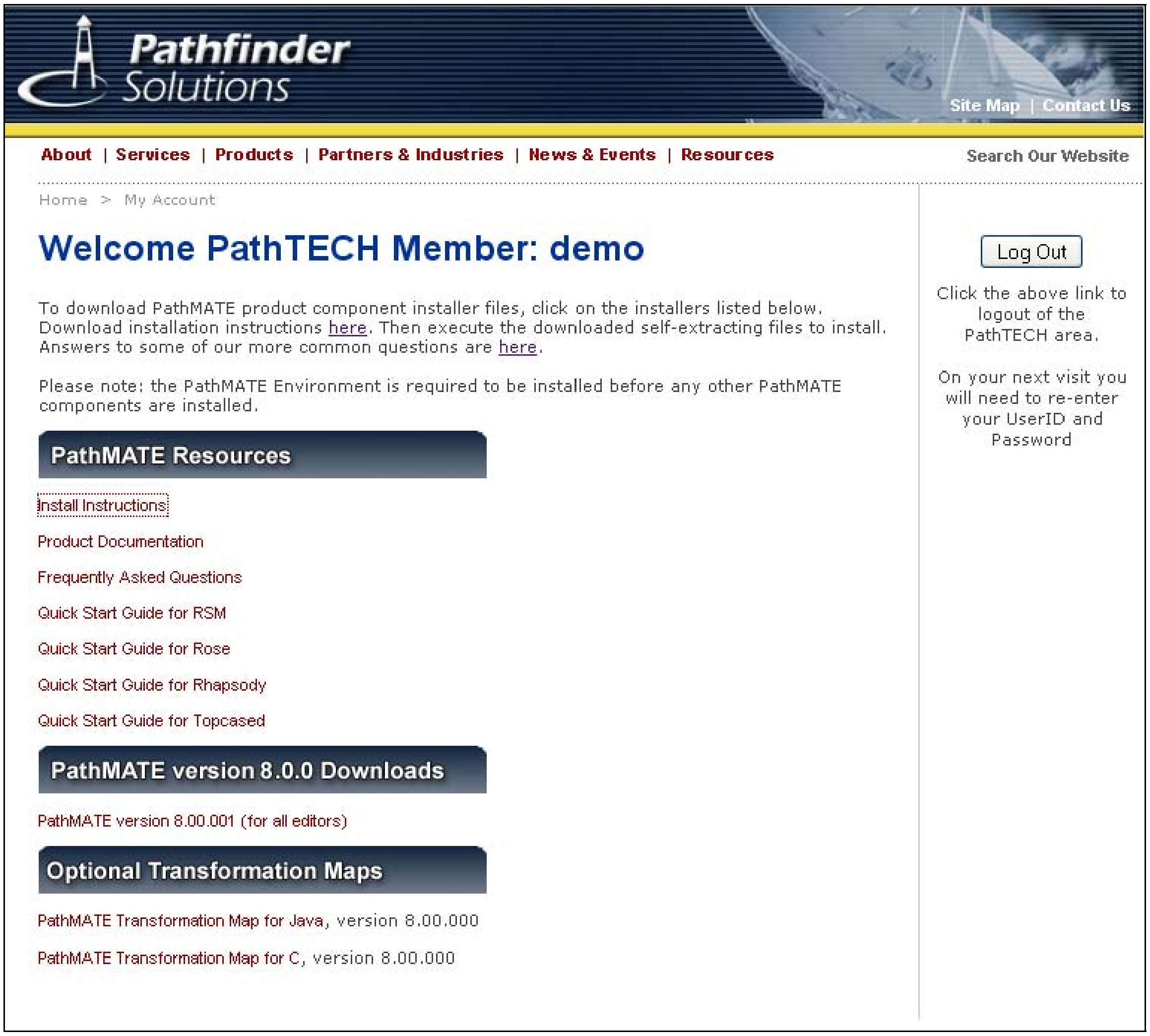
No other MDA transformation environment offers a more open or configurable set of development tools, designed to meet the requirements of embedded systems engineers.

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:



Welcome PathTECH Member: demo

To download PathMATE product component installer files, click on the installers listed below. Download installation instructions here. Then execute the downloaded self-extracting files to install. Answers to some of our more common questions are here.

Please note: the PathMATE Environment is required to be installed before any other PathMATE components are installed.

Click the above link to logout of the PathTECH area.

On your next visit you will need to re-enter your UserID and Password

PathMATE Resources

Install Instructions

Product Documentation

Frequently Asked Questions

Quick Start Guide for RSM

Quick Start Guide for Rose

Quick Start Guide for Rhapsody

Quick Start Guide for Topcased

PathMATE version 8.0.0 Downloads

PathMATE version 8.00.001 (for all editors)

Optional Transformation Maps

PathMATE Transformation Map for Java, version 8.00.000

PathMATE Transformation Map for C, version 8.00.000

3. Download the following files:
   - *PathMATE*
   - *PathMATE Transformation Map for Java (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 is you are using this version of Visual Studio.

Model a Simple Oven

The table below, outlines the model-building part of the tutorial, and introduces some PathMATE terminology:

Term	Definition
Project	Eclipse projects are containers for related development resources, often with a one-to-one correspondence with file system objects.
System Model	A PathMATE refinement of an RSM UML package container for our overall logical architecture which corresponds to the Eclipse Project.
Domain Model	A PathMATE refinement of an RSM UML package logical architectural component of our system.
Domain Services	Input/Output/Event interfaces exposed by a Domain Model.
(System) Domain Chart	Drawing showing the Domains and their and relationships that comprise the System Model.
(Domain) Class Diagram	Drawing showing the interfaces, classes, and their relationships that comprise a Domain.
Application Domain	Architecture intended to solve a specific design need. In this case, control a simple microwave oven.
Door State Machine	A logical specification of states, possible transitions between states, actions that can be performed, and the conditions or events under which transitions and actions can occur. In this case, the description of door open/closed processing.
Light State Machine	In this case, the description of light on/off processing.
Entry Actions	Detailed specification of actions that occur on the *entry* into a state. Other, possible times are while *in* a state, or when *exiting* a state.

Ready, Get Set…

When you launch RSM, designate a workspace to store all the files for the sample system. Typically you designate your workspace once, just after you install RSM. We refer to this directory as your Workspace Directory. Use the same workspace whenever you work on this tutorial.

To work with PathMATE, activate the Eclipse “Modeling” Perspective.

As with most leading software products, there are generally many ways in which a given action can be performed. We present a single, simple, and consistent way for maximum clarity. As you gain experience and comfort, explore the menus and dialogues. Try out some of the alternatives and develop the work style most comfortable to you.

Task 1: Create a UML Project Using the PathMATE System Model Template

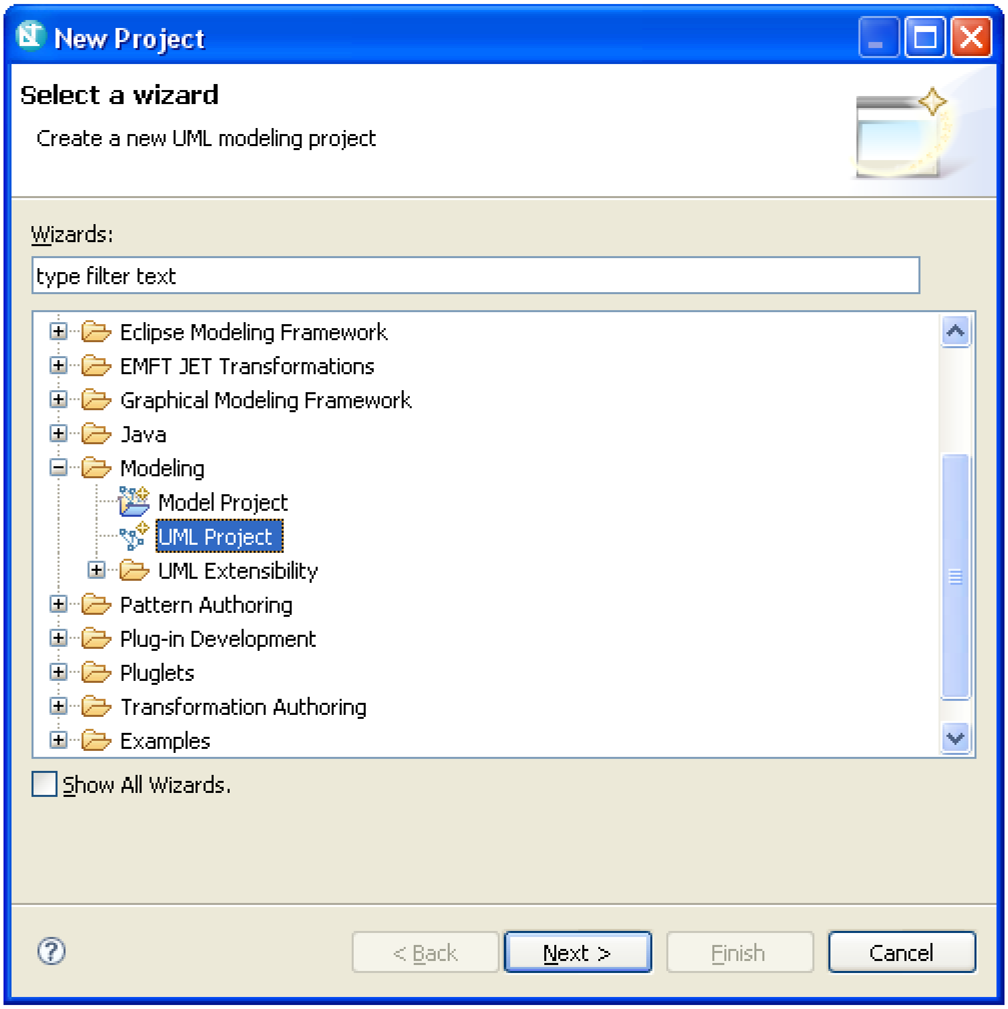
In Eclipse, all work is conducted within a project. We will create a UML Modeling Project, and create a PathMATE System Model:

PROCEDURE: Use Eclipse File > New and New Project Wizards

To create a system model:

1. Launch RSM. Select a workspace. The Welcome window appears the first time you open a workspace. If you see the Welcome window, click the x to close it.
2. Select **File > New > Project** in the top menu bar.

The New Project dialog opens.

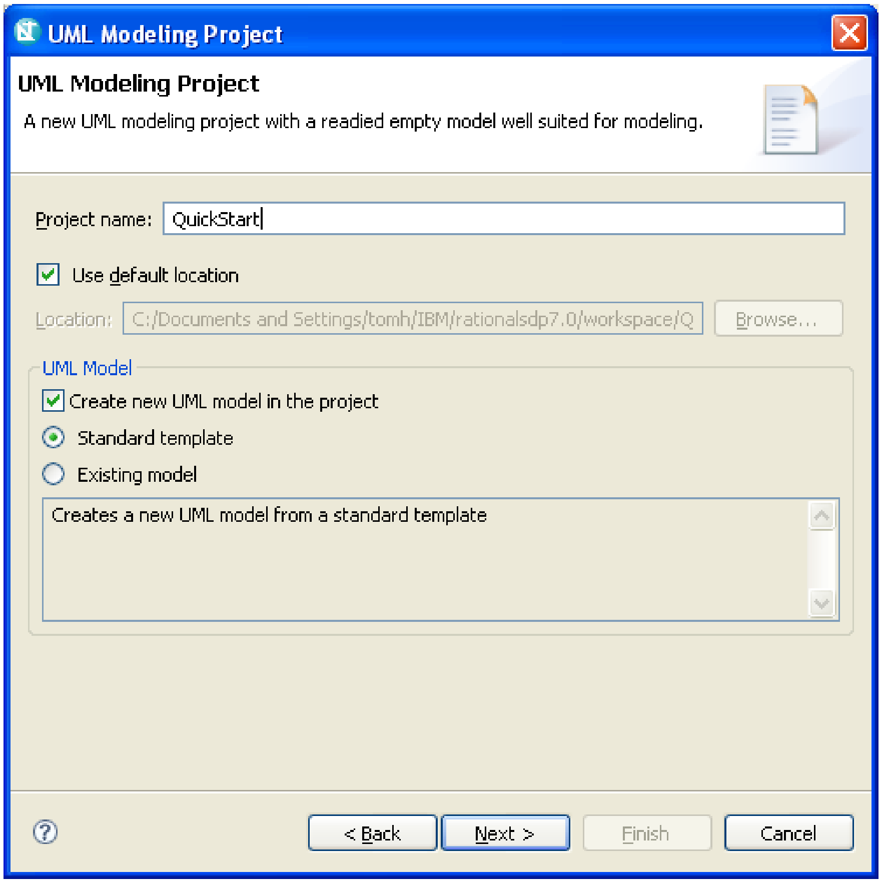


Select a wizard

Create a new UML modeling project

3. Select **Modeling > UML Project** in the list of wizards. Click **Next**.

The UML Modeling Project wizard opens.


UML Modeling Project
UML Modeling Project
A new UML modeling project with a readied empty model well suited for modeling.

Project name: QuickStart

☑ Use default location

Location: C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/Q

☑ Create new UML model in the project

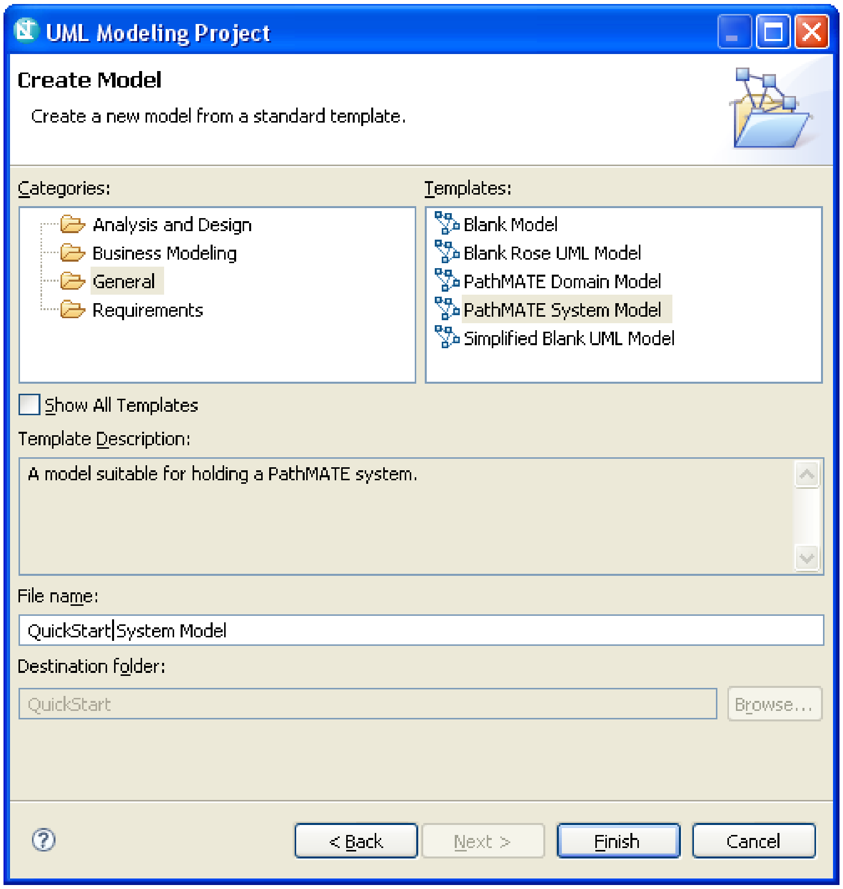
● Standard template

○ Existing model

Creates a new UML model from a standard template

4. Type *QuickStart* in the Project name field. Click **Next**.

The Create Model page opens.



UML Modeling Project

Create Model

Create a new model from a standard template.

Categories:
- Analysis and Design
- Business Modeling
- General
- Requirements

Templates:
- Blank Model
- Blank Rose UML Model
- PathMATE Domain Model
- PathMATE System Model
- Simplified Blank UML Model

□ Show All Templates

Template Description:

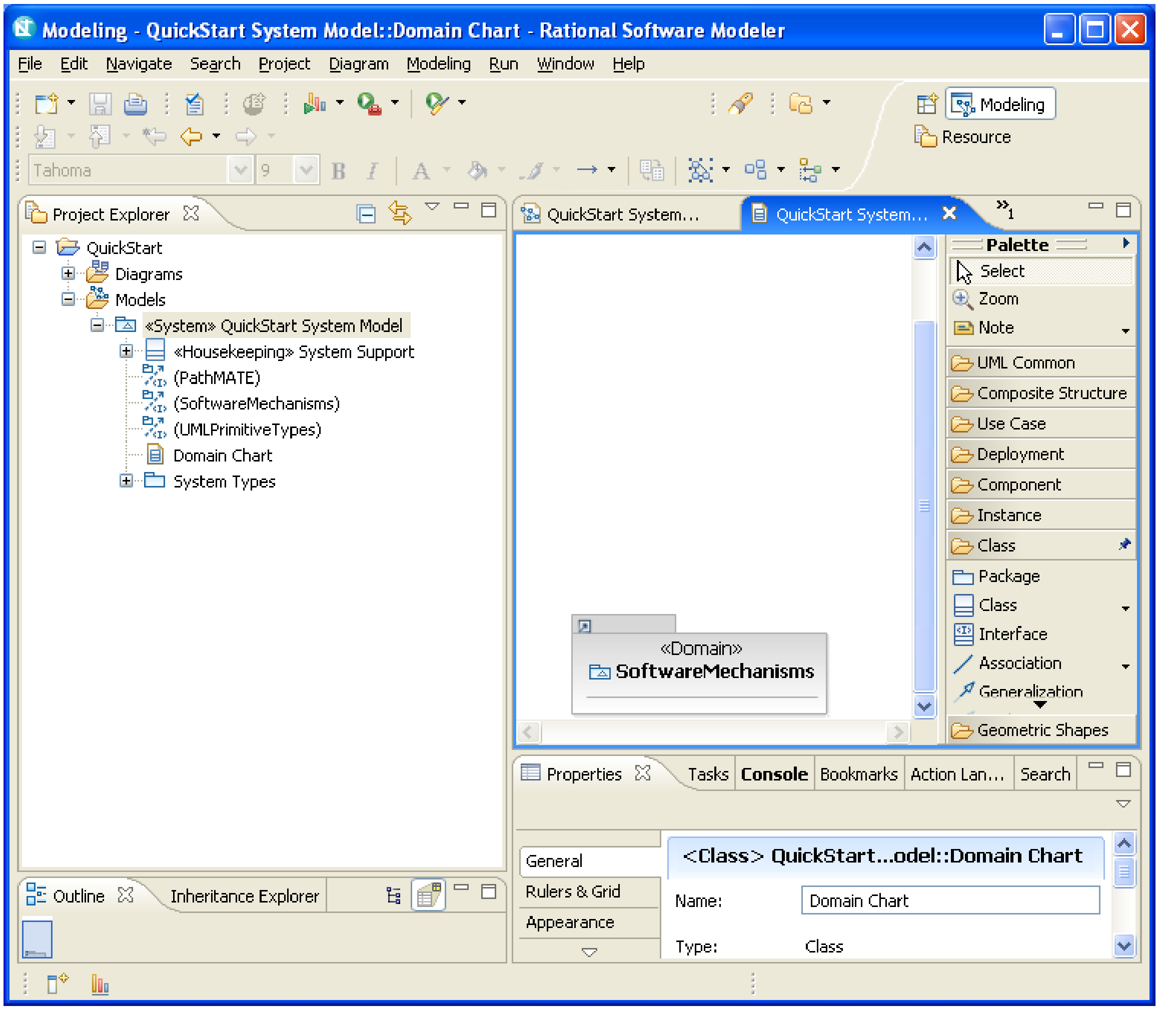
A model suitable for holding a PathMATE system.

File name: QuickStart System Model

Destination folder: QuickStart

5. Select **General > PathMATE System Model** in the list of templates.
6. In the File name field, enter *QuickStart System Model*.
7. Click **Finish**.

The QuickStart System Model appears in the Project Explorer.



<Class> QuickStart...odel::Domain Chart

Name: Domain Chart

Type: Class

Note these PathMATE System items in the Project Explorer:

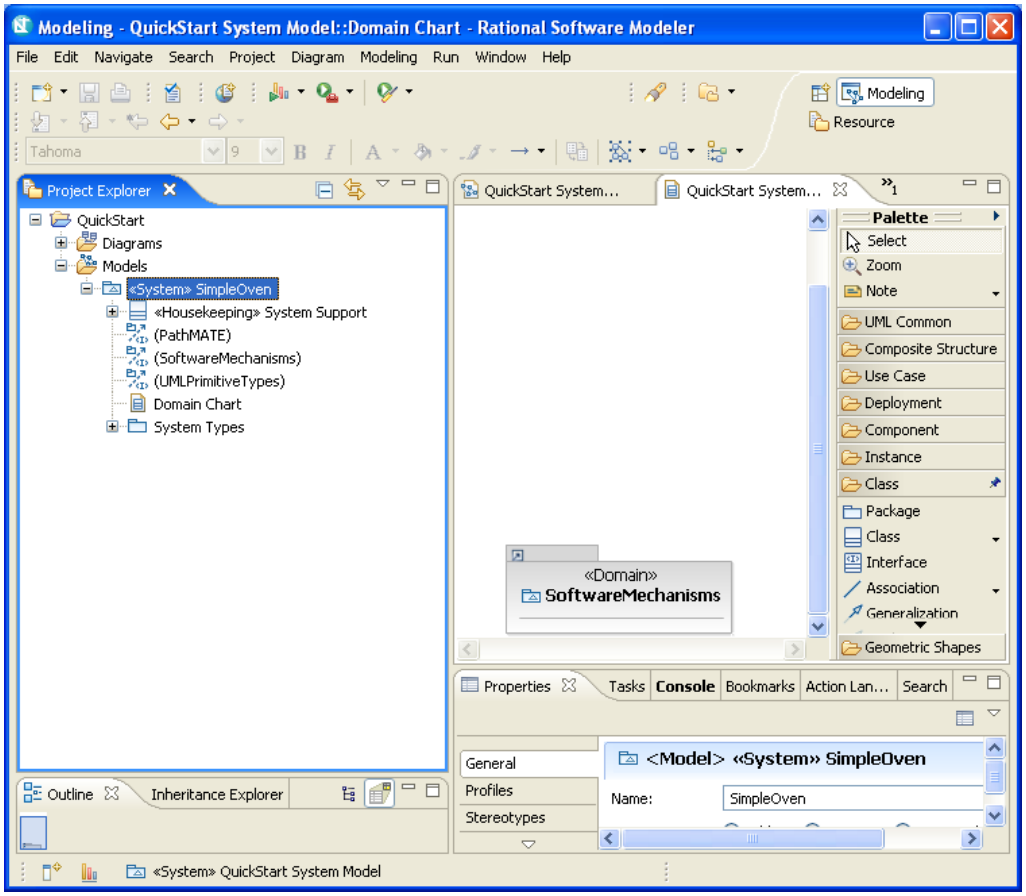
- A System Types package for user defined types.
- A Domain Chart opens in the editor pane, with a reference to the SoftwareMechanisms domain (see Task 4, Complete the System Domain Chart, below).
- A System Support class that contains the system initialization() operation.

Also loaded, but not visible in the Project Explorer, are the PathMATE-specific primitive types and PathMATE profile for the model.

PROCEDURE: Use Eclipse Refactor to do a Smart Rename

A useful capability is to rename elements and the references to those elements using a smart rename function. In Eclipse this is known as *refactoring*. To demonstrate,

1. Right-click the **<<System>> QuickStart System Model** package, select **Refactor > Rename**. The Rename dialog appears.
2. Replace *QuickStart System Model* with *SimpleOven*. Click **OK**.
3. The system model is now called SimpleOven.


Modeling - QuickStart System Model::Domain Chart - Rational Software Modeler

<Model> «System» SimpleOven

Name: SimpleOven

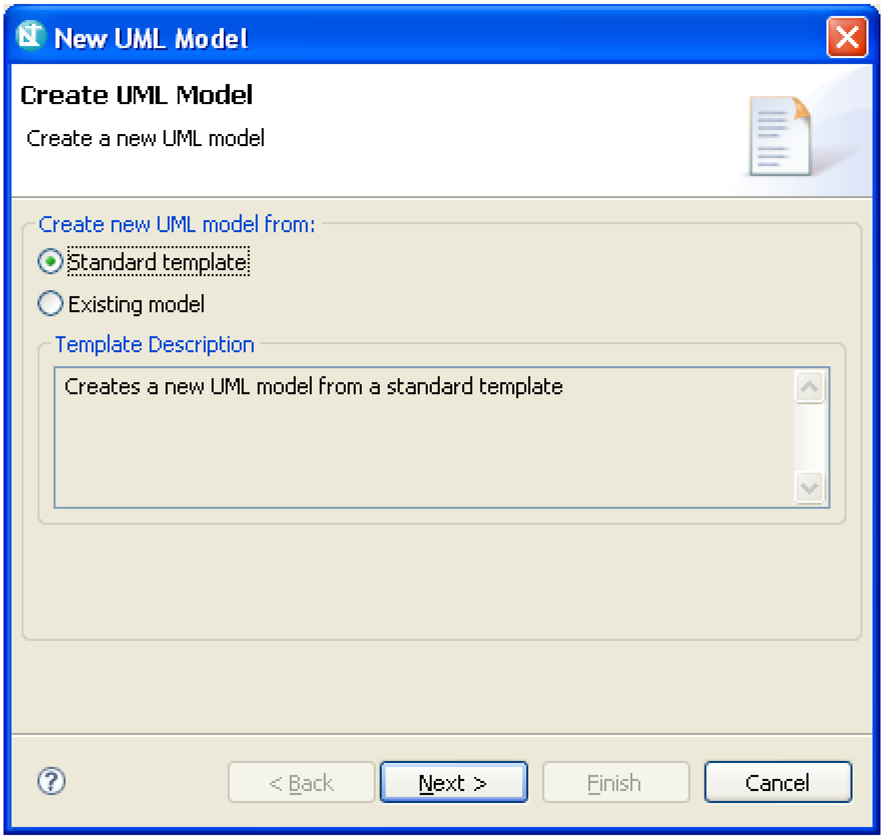
Task 2: Add a Modeled Domain

PROCEDURE: Use Eclipse File > New to Add a Domain

To create a domain model for the system's application domain, MicrowaveCooking:

1. Select **File > New > Model** in the top menu bar.

The New UML Model dialog opens.



Create UML Model

Create a new UML model

Create new UML model from:

● Standard template

○ Existing model

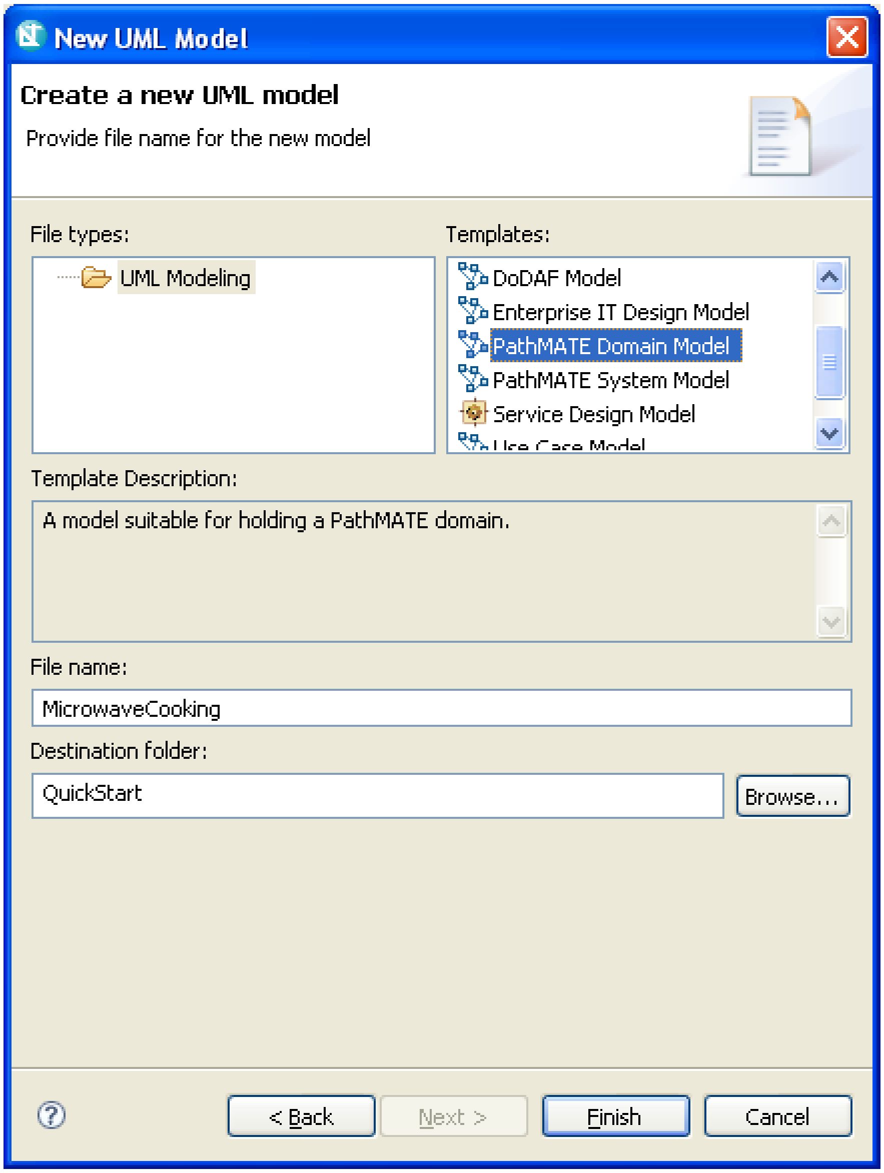
Template Description

Creates a new UML model from a standard template

2. Accept the **Standard template** option, and click **Next**.

On the next page:

3. Select **General > PathMATE Domain Model** in the list of templates.
4. Type *MicrowaveCooking* in the File name field.
5. Make sure that *QuickStart* appears in the Destination folder field. If not, click the Browse… button.
6. Click **Finish**.



Create a new UML model

Provide file name for the new model

File types:

Templates:

Template Description:

A model suitable for holding a PathMATE domain.

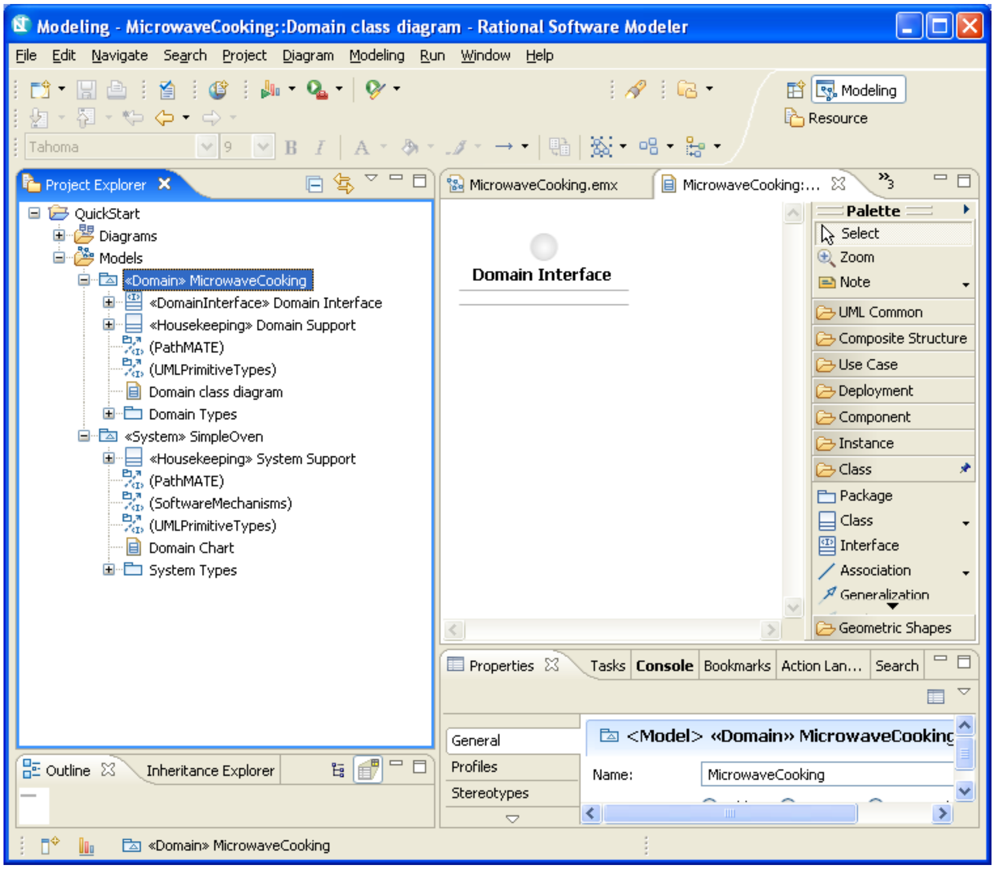
File name: MicrowaveCooking

Destination folder: QuickStart

The new domain model appears in the Project Explorer.

Expand **<<Domain>> MicrowaveCooking** in the Project Explorer to see:

- A Domain Types package for user defined types within the domain.
- A Domain class diagram. The class diagram is open in your edit pane. See Create a Class Diagram for more information about the domain class diagram.
- A Domain Support class that contains the domain initialization() operation.
- A Domain Interface class for domain services. The class is a published interface.



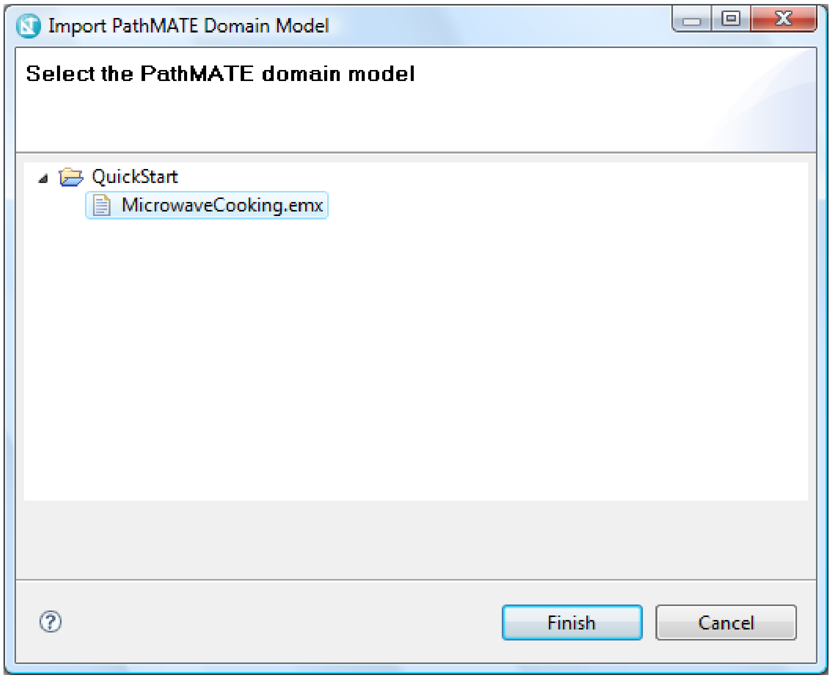
<Model> «Domain» MicrowaveCooking

Name: MicrowaveCooking

PROCEDURE: Import MicrowaveCooking domain into SimpleOven

To include a domain model in a system model:

1. In the Project Explorer, select **<<System>>SimpleOven**.
2. Right click and select **Import PathMATE Domain Model…** The Import PathMATE Domain Model dialog appears.
3. Select **QuickStart > MicrowaveCooking.emx**.
4. Click the **Finish** button.


Import PathMATE Domain Model
Select the PathMATE domain model
QuickStart
MicrowaveCooking.emx
Finish
Cancel

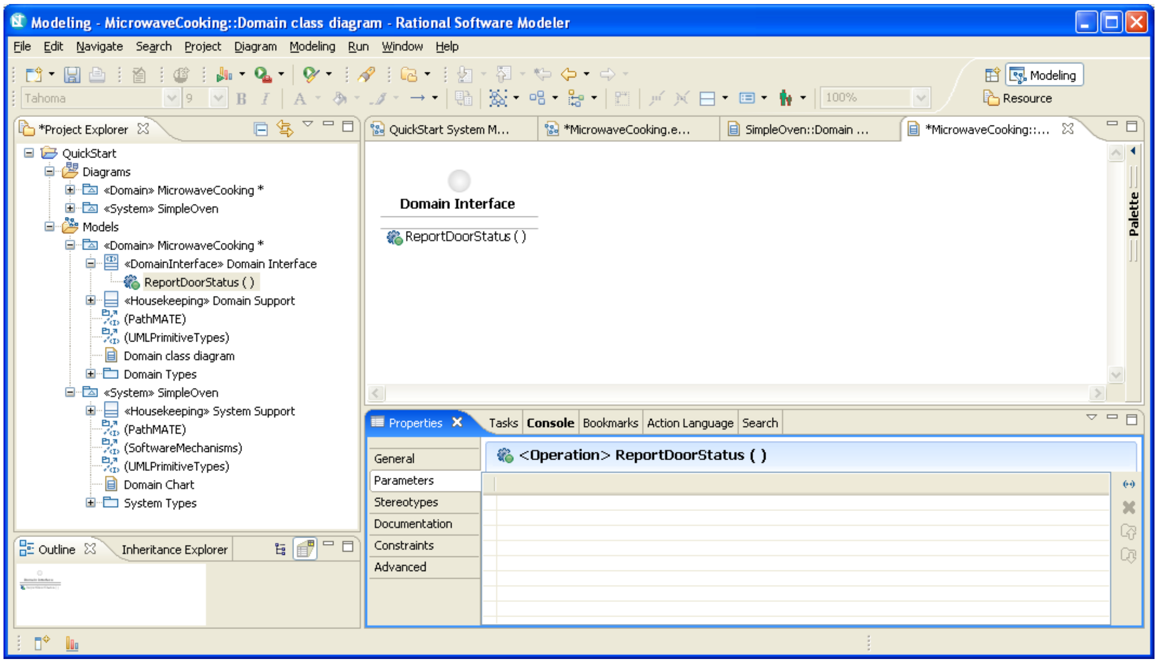
PROCEDURE: Use Eclipse Context Menu to Add a UML Operation

To define the domain interface for MicrowaveCooking:

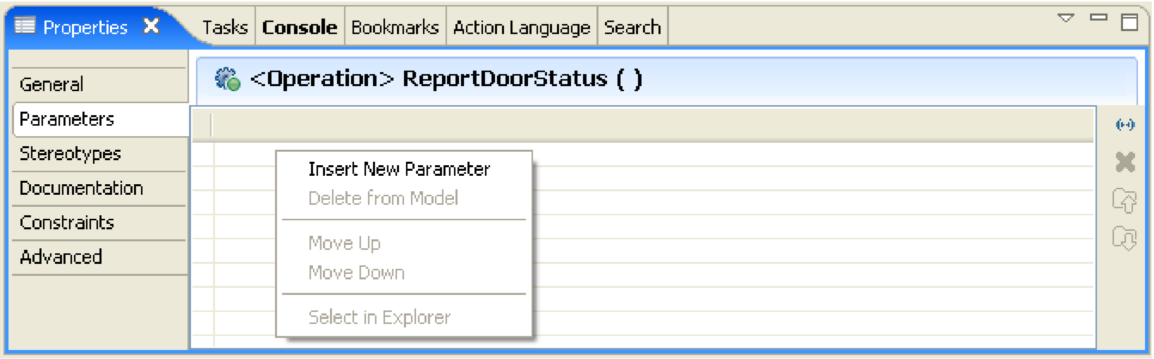
1. In the Project Explorer expand **<<Domain>>MicrowaveCooking** if necessary and perform the actions below within its context.
2. Right-click **<<Domain Interface>> Domain Interface** and select **Add UML > Operation** in the pop-up menu.
3. An Operation element is added as a child to <<Domain Interface>> Domain Interface and an edit-in-place field opens. (If not, right click on the Operation element in the Project Explorer and select Refactor > Rename.)
4. Name the operation *ReportDoorStatus*.

PROCEDURE: Use Eclipse Properties View to Add a Parameter to an Operation

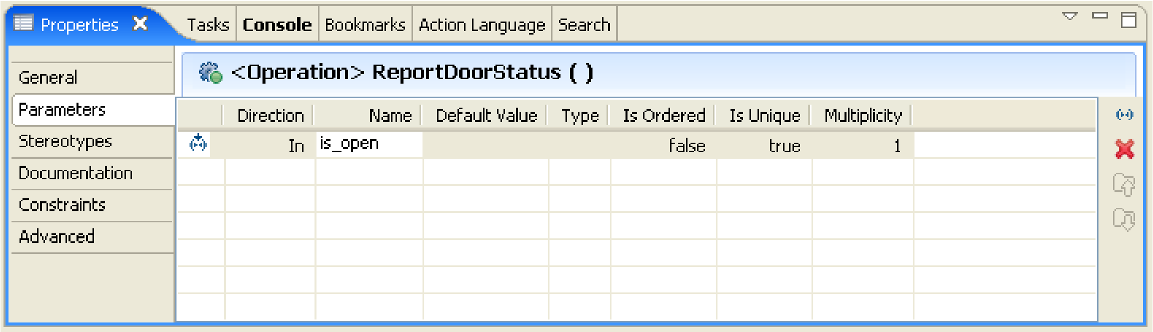
1. With the new operation selected in the Project Explorer, activate the context-sensitive Properties View by clicking on its **tab** at the top. Select the **Parameters tab** on the left.



2. Right-click in the parameters panel and select **Insert New Parameter**.

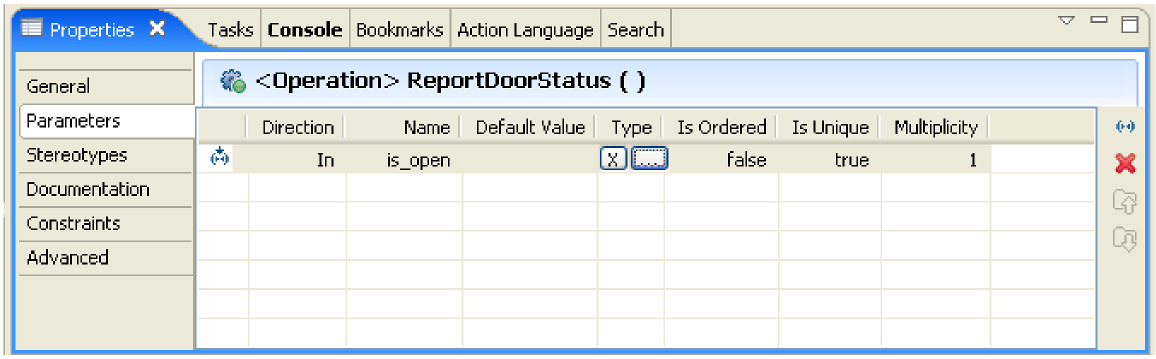


3. Enter *is_open* in the Name column.



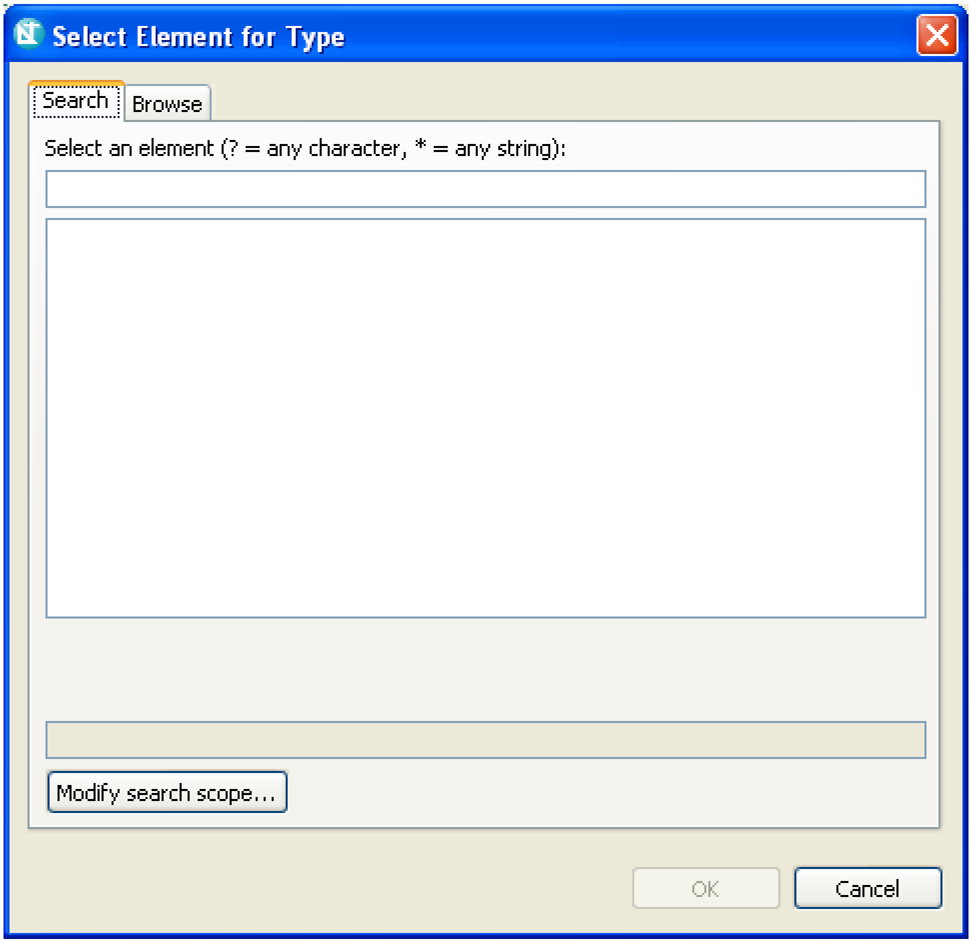
Direction	Name	Default Value	Type	Is Ordered	Is Unique	Multiplicity
In	is_open			false	true	1

4. Click on the empty cell in the Type column and click the "**...**" button in the right-hand side of the Type field.

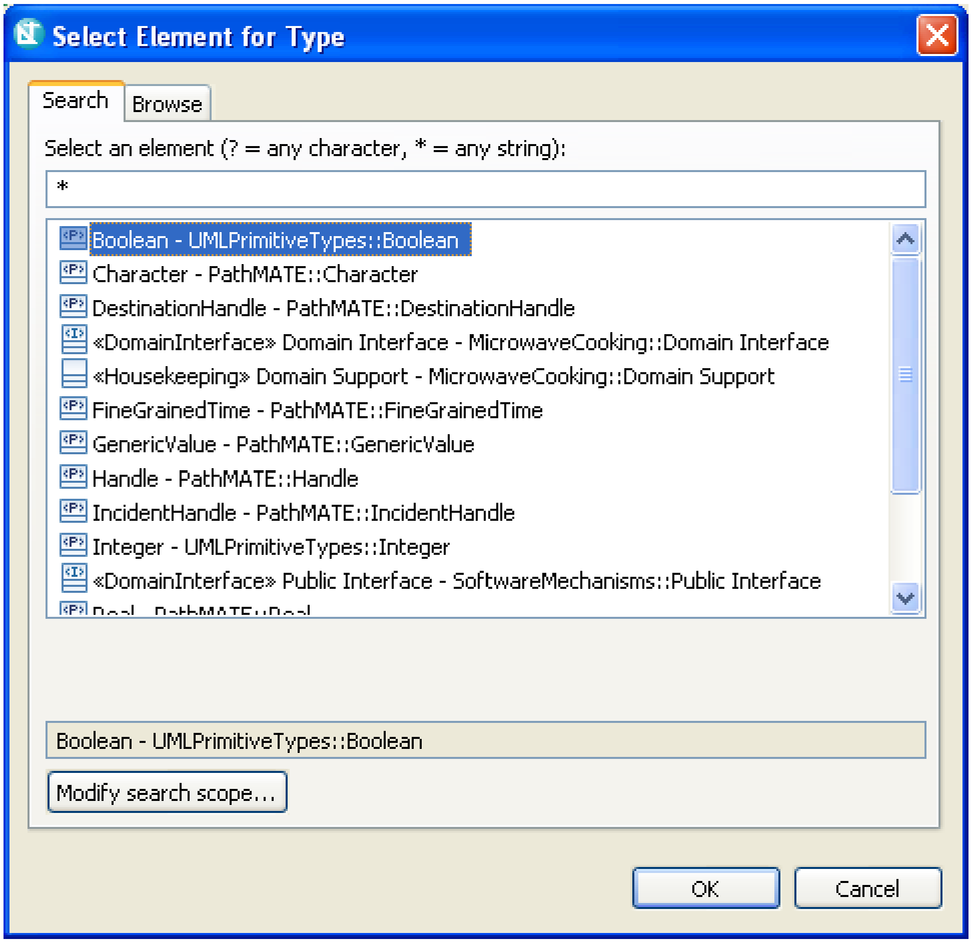


Direction	Name	Default Value	Type	Is Ordered	Is Unique	Multiplicity
In	is_open			false	true	1

5. The Select Element for Type dialog opens. But, there is nothing to select. Type an asterisk "*****" in the top edit field:



6. Now, select **Boolean** in the list of types and click OK to close the dialog.



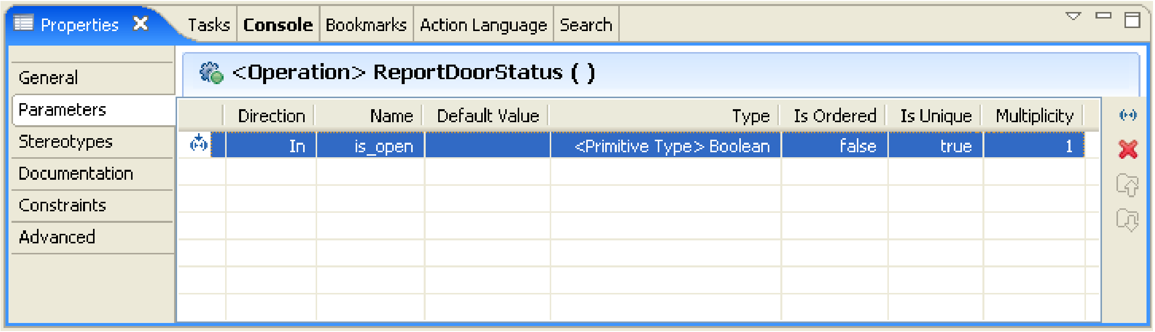
Select an element (? = any character, * = any string):

*

- Boolean - UMLPrimitiveTypes::Boolean
- Character - PathMATE::Character
- DestinationHandle - PathMATE::DestinationHandle
- «DomainInterface» Domain Interface - MicrowaveCooking::Domain Interface
- «Housekeeping» Domain Support - MicrowaveCooking::Domain Support
- FineGrainedTime - PathMATE::FineGrainedTime
- GenericValue - PathMATE::GenericValue
- Handle - PathMATE::Handle
- IncidentHandle - PathMATE::IncidentHandle
- Integer - UMLPrimitiveTypes::Integer
- «DomainInterface» Public Interface - SoftwareMechanisms::Public Interface
- Real - PathMATE::Real

Boolean - UMLPrimitiveTypes::Boolean

The figure below shows the Parameters section of the Properties tab after you define the domain interface for MicrowaveCooking.



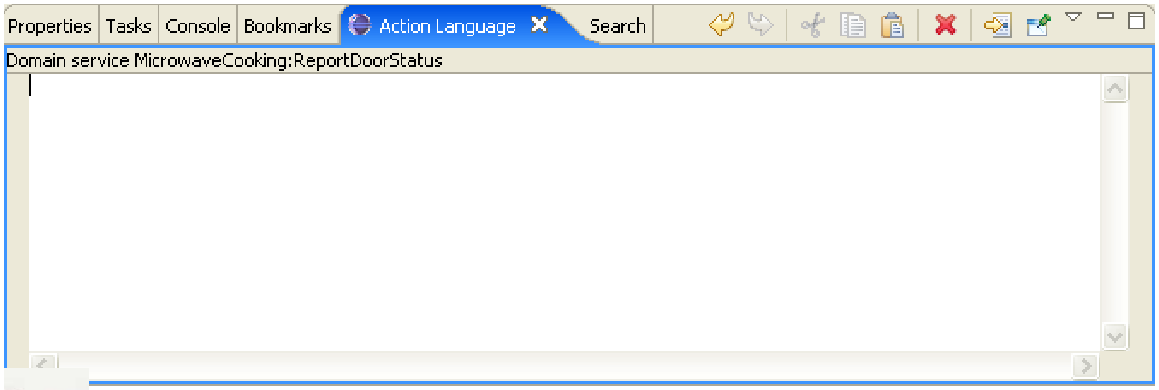
<Operation> ReportDoorStatus ( )

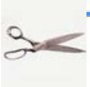
Direction	Name	Default Value	Type	Is Ordered	Is Unique	Multiplicity
In	is_open		<Primitive Type> Boolean	false	true	1

PROCEDURE: Add ReportDoorStatus Interface Service Action Language

To complete the Interface service, incorporate action language that returns the status of the door.

1. Select **ReportDoorStatus ()** under <<Domain>>MicrowaveCooking, <<Domain Interface>> Domain Interface in the Project Explorer.
2. Bring the context-sensitive Action Language View forward by clicking on its **tab** at the top. Note that it is usually in the same view group as the Properties View. If you do not see the Action Language View, select **Window > Show View > Other…** from the top menu bar. The Show View dialog appears. Select **PathMATE > Action Language**. Click **Ok**.





3. Note the generated signal in the following Action Language. Copy the following Action Language and paste (using Control-V) into the Action Language view:

```
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 from Adobe Acrobat, click the Select Text tool in your Acrobat toolbar. Then select the text, copy it, and paste using Control-V in the Action Language view.

Task 3: Create a Realized Domain

ExternalDeviceControl is a realized (non-modeled) domain that provides domain services for the system. These procedures explain how to abstract the interface to the ExternalDeviceControl services at the Platform *Independent* Model (PIM) level for use by modeled domains.

The first step is to create a new Domain which is by default a Modeled Domain. Then, the Domain's PathMATE "Analyzed" Property is changed from True to False, making the Domain Realized.

The interface to the ExternalDeviceControl includes an Enumeration Type which is specified first and used to define Operation arguments.

PROCEDURE: Use Eclipse File > New to Add a Domain

To create the initial ExternalDeviceControl Modeled Domain, the steps are essentially the same used to create the MicrowaveCooking Domain, above:

1. Select the *QuickStart* project from the Project Explorer.
2. From the main Eclipse menu bar select **File > New > Model**. (The Model dialog opens.)
3. Accept the **Standard template** option, and click **Next**.
4. Select **PathMATE Domain Model** in the list of templates.
5. Type *ExternalDeviceControl* for the File name field.
6. Ensure that the *QuickStart* project is specified as the destination folder.
7. Click **Finish**.

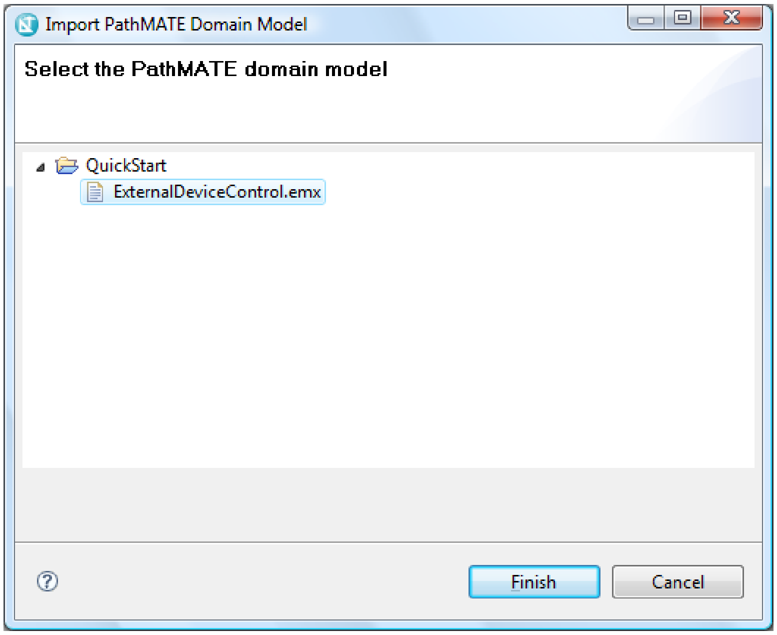
The new domain model appears in the Project Explorer.

PROCEDURE: Import ExternalDeviceControl domain into SimpleOven

To include a domain model in a system model:

1. In the Project Explorer, select **<<System>>SimpleOven**.
2. Right click and select **Import PathMATE Domain Model…** The Import PathMATE Domain Model dialog appears.
3. Select QuickStart > ExternalDeviceControl.emx.

4. Click the **Finish** button.



Import PathMATE Domain Model

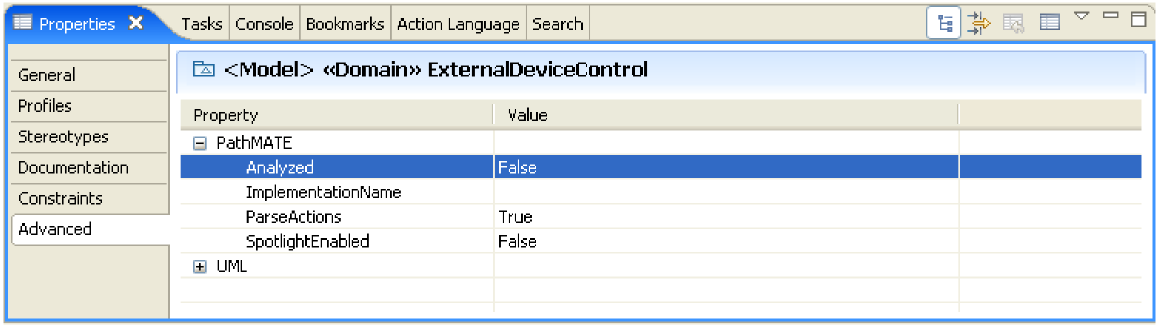
Select the PathMATE domain model

- QuickStart
  - ExternalDeviceControl.emx

PROCEDURE: Use the Eclipse Properties View to modify the "Analyzed" property

1. Select **<<Domain>> ExternalDeviceControl** in the Project Explorer.
2. Ensure that the context-sensitive Properties View is visible by clicking on its tab.
3. Click on the **Advanced tab**.
4. Expand the PathMATE properties group.
5. Select the **"Analyzed"** property from the PathMATE group and change its value from True to False.

The figure below shows the Properties tab as it appears after you complete these steps.



<Model> «Domain» ExternalDeviceControl

Property	Value
PathMATE	
Analyzed	False
ImplementationName	
ParseActions	True
SpotlightEnabled	False
UML	

PROCEDURE: Use Eclipse Context Menu to Add a UML Enumeration to a Domain's Public Types

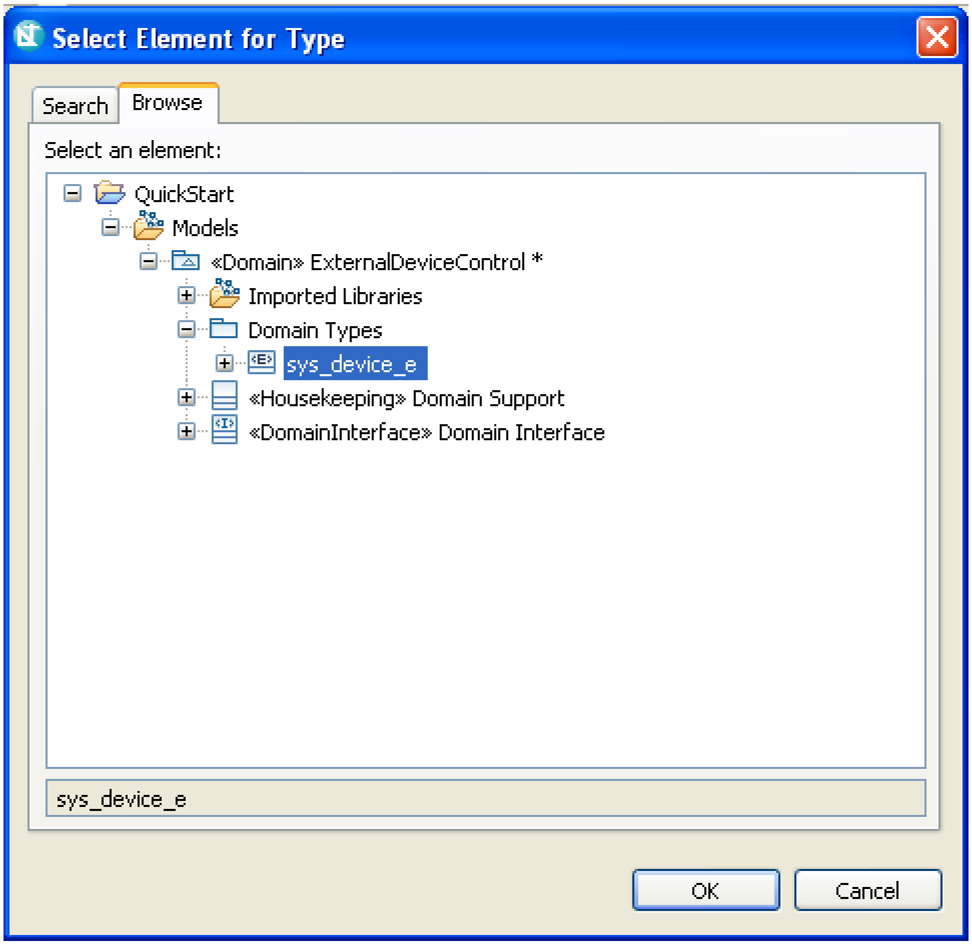
1. Expand **<<Domain>> ExternalDeviceControl** in the Project Explorer. Perform the next steps below within its context.
2. Right-click **Domain Types** and select **Add UML > Enumeration** in the pop-up menu.
3. Name the enumeration *sys_device_e*.
4. Right-click the new enumeration and select **Add UML > Enumeration Literal**.
5. Name the new literal *SYS_DEVICE_LIGHT*.

PROCEDURE: Use Eclipse Context Menu to Add a UML Operation

1. Continue to work under <<Domain>> ExternalDeviceControl in the Project Explorer.
2. Right-click **<<Domain Interface>> Domain Interface** and select **Add UML > Operation** in the pop-up menu.
3. Name the operation *ActivateDevice*.

PROCEDURE: Use Eclipse Properties View to Add a Parameter to an Operation

1. Select **Parameters** in the Properties tab.
2. Right-click in the parameters panel and select **Insert New Parameter**.
3. Enter *device_id* in the Name column.
4. Click on the empty cell in the Type column.
5. Click the "**...**" button in the right-hand side of the Type field.
6. Click on the Browse tab. Select QuickStart, Models, External Device Control, Domain Types, and select the **sys_device_e** enumeration. Click **OK**.



Select an element:

- QuickStart
  - Models
    - «Domain» ExternalDeviceControl *
      - Imported Libraries
      - Domain Types
        - sys_device_e
      - «Housekeeping» Domain Support
      - «DomainInterface» Domain Interface

sys_device_e

PROCEDURE: Use Eclipse Copy and Paste to Add a UML Operation

1. In the browser, select the ActivateDevice operation, right-click and pick **Copy**.
2. Select the <<Domain Interface>> Domain Interface, right-click and pick **Paste**.
3. Select Copy_1_ActivateDevice, right-click and pick **Refactor > Rename** (or use F2). The Save All Modified Resources dialog appears. Click **OK** to save the models before refactoring. The Rename dialog appears. Change the name to *DeactivateDevice* and click **OK**.

When you are finished, select **File > Save All** in the top menu bar (or hit Control-Shift-S) to save your changes.

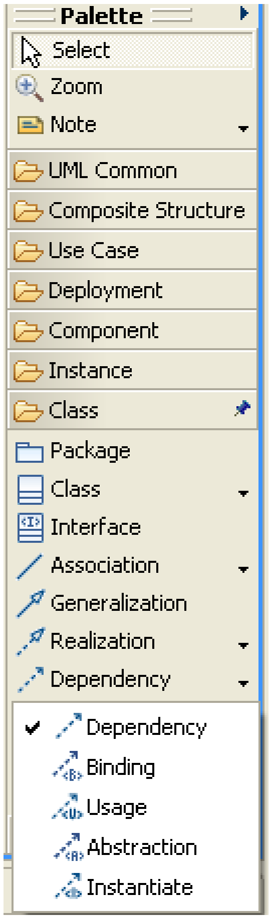
NOTE: If Save All is disabled, the model has already been saved. The Save All option will become selectable when there are unsaved changes to one or more elements in the workspace.

Task 4: Complete the System Domain Chart

Keep in mind that System and Domain models are PathMATE refinements of UML Packages as you work through this task.

To complete the System Domain Chart you'll need to locate (and select if necessary) the Dependency directed line tool on the Drawing Editor Palette. Note where the Class and Association tools are as well, you'll need them later on in the tutorial. The Palette can be positioned on the left or right of a diagram and may be minimized. In any event, the word "Palette" will be in view.

Palette with Dependencies selected



As you work with Diagrams, you will find that repositioning is necessary from time to time to maintain legibility and style.

PROCEDURE: Create the Domain Diagram

1. Open (Double-click) the **Domain Chart** in the SimpleOven <<System>> package. The MicrowaveCooking and ExternalDeviceControl domains were added to top left of the Domain Chart diagram automatically when the Import PathMATE Domain Model menu item was selected. The imported package symbols will overlap. Left click on the top package and drag it to reveal the package underneath.

   If you do not see the MicrowaveCooking or ExternalDeviceControl domains import the domain models by following the **PROCEDURE: Import MicrowaveCooking domain into SimpleOven** on page 15 or the **PROCEDURE: Import ExternalDeviceControl domain into SimpleOven** on page 21.

2. Drag **<<Domain>> MicrowaveCooking** domain to the upper center of the diagram.

3. Drag **<<Domain>> ExternalDeviceControl** below and to the right of MicrowaveCooking.

4. Select the **Dependency** tool under “Class” in the Palette. Place the cursor over the text label on the MicrowaveCooking <<Domain>> package. Left click and drag to the text label on the SoftwareMechanisms <<Domain>> package. Release the mouse. The MicrowaveCooking domain now has a dependency on SoftwareMechanisms domain.

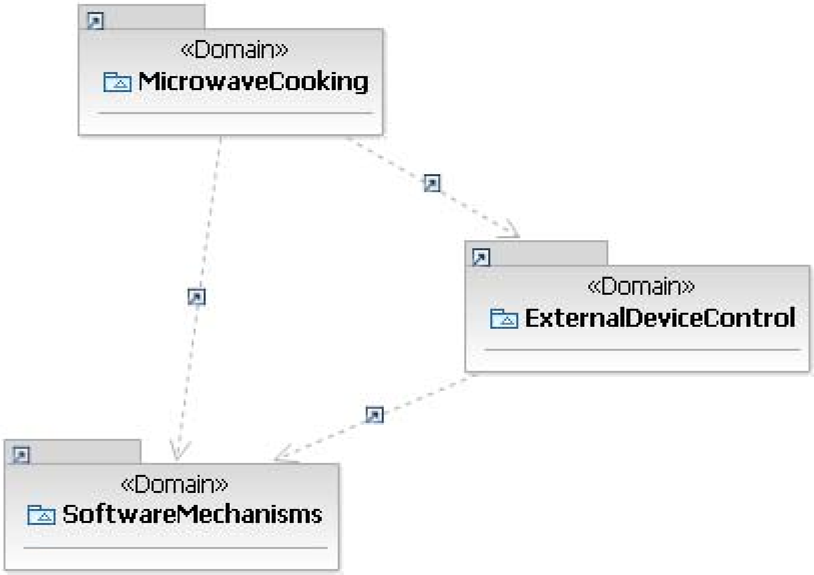
   NOTE: If the mouse is below the horizontal line underneath the text label of the domain package, you will not be able to start a dependency. Move the cursor above the horizontal line.

   If a menu item shows after dropping, click on free space to cancel. Make sure the mouse is inside the destination domain package when dropping.

5. Do the same to specify the dependency of MicrowaveCooking on ExternalDeviceControl.

6. Specify the dependency of ExternalDeviceControl on SoftwareMechanisms.

The figure below shows the System Domain Chart after you complete these steps.



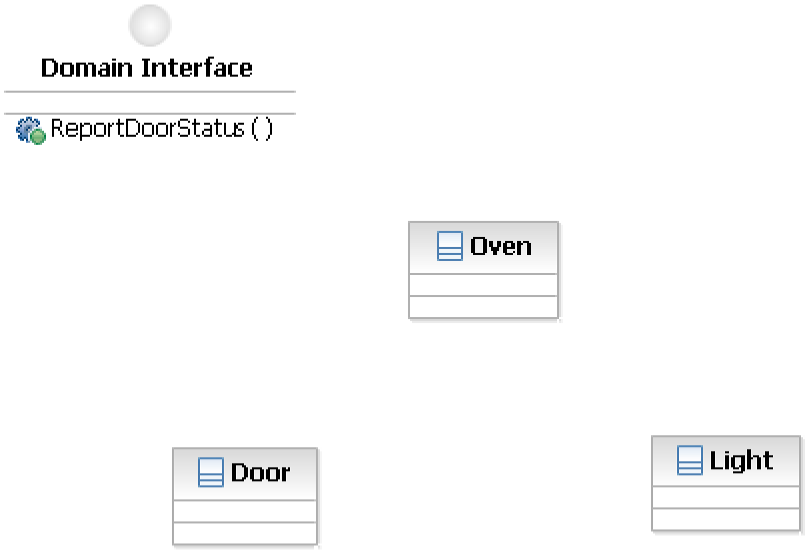
Task 5: Complete the Class Diagram for the MicrowaveCooking Domain

The MicrowaveCooking Domain is the heart of the application. In this task you will complete the MicrowaveCooking's Domain Class Diagram by adding Classes, Attributes, and Associations.

PROCEDURE: Use RSM Diagram to Create Three Classes in MicrowaveCooking Domain

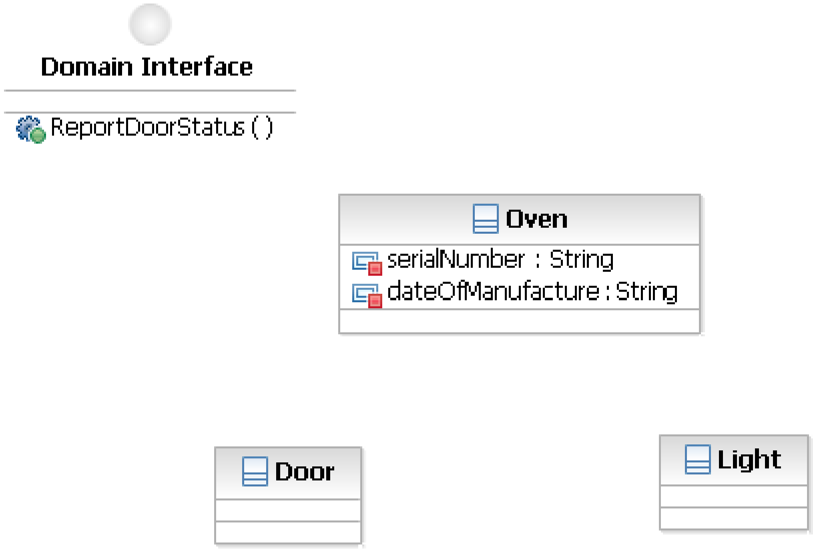
1. Open the **Domain Class Diagram** under <<Domain>> MicrowaveCooking in the Project Explorer.
2. Select the **Class** tool in the Palette and move the cursor to the upper part of the editor pane, and click the open space to place the first class.
3. Type *Oven* in place of *Class1* in the class's title bar.
4. Do the same to create a *Door* class below and to the left of the Oven class.
5. Create a *Light* class below and to the right of Oven.

The figure below shows the Domain Class Diagram after you complete these steps.



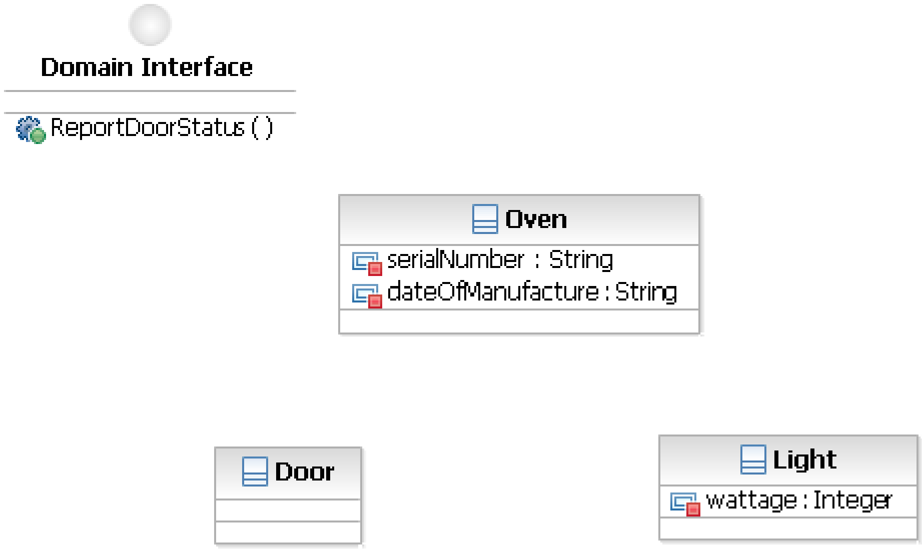
PROCEDURE: Use Eclipse Context-Sensitive Menu and Properties View to Add Attributes to the Oven Class and Set Their Values

1. In the Domain Class Diagram right-click **Oven** to access its context-sensitive menu and select **Add UML > Attribute**.
2. *Attribute1* appears in the Oven class and an edit-in-place field opens. (If not, right click on the attribute and select **Refactor > Rename**.) Name the Attribute *serialNumber*.
3. Ensure that the Properties View is visible and click **General** tab in the Properties View.
4. Click the **Select type…** button in the General Tab.
5. Enter S in the filter string at the top of the Select Element for Type dialog. Select **String** from the list and click **OK**.
6. In a similar fashion, add another attribute *dateOfManufacture* with a type **String**, to the Oven class.



PROCEDURE: Use Eclipse Context-Sensitive Menu and Properties View to Add Attributes to the Light Class and Set Their Values

1. In a similar manner, add an Integer attribute, *wattage*, to the Light class.

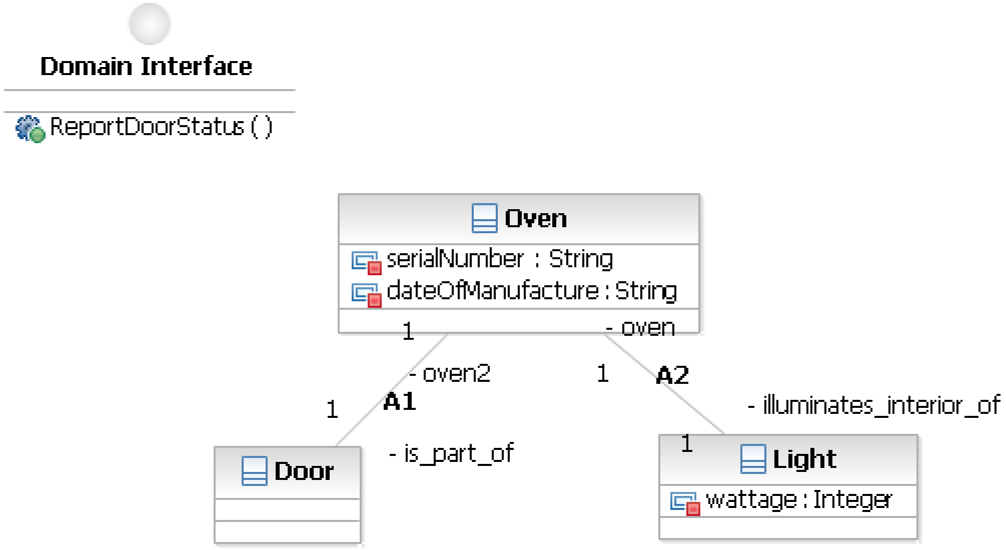


PROCEDURE: Use RSM Diagram to Associate Classes

To associate the Oven with the Door and Light classes:

1. Select the **Association** tool under “Class” in the Palette and draw a line from Oven to Door to specify an Association between the two.
2. Type *A1* in the association’s name edit-in-place title bar.
3. Select the **General** tab in the Properties view and fill in the role phrase *is_part_of* for Door.
4. In a similar manner, create the association *A2* between Oven and Light, specifying the Light role *illuminates_interior_of*.

The figure below shows the Class diagram after you complete these steps.

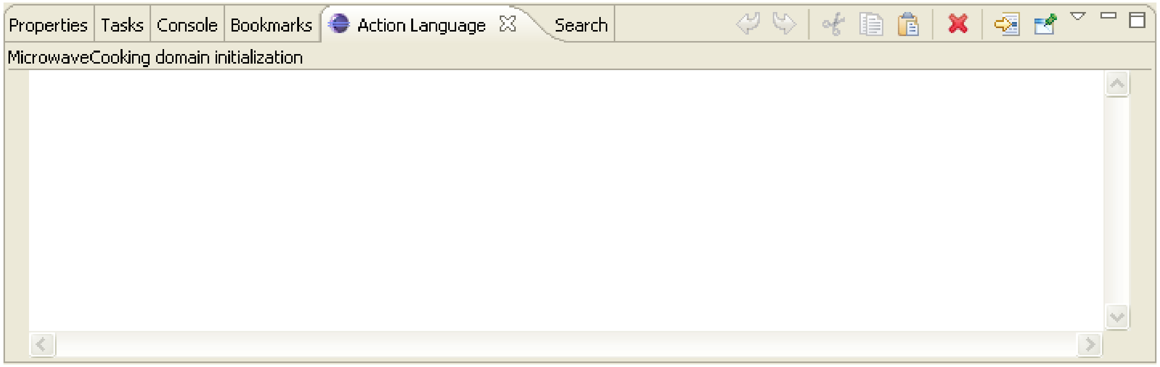


Task 6: Complete the Application Domain

PROCEDURE: Add MicrowaveCooking Initialization Action

To complete the application domain, incorporate action language that initializes MicrowaveCooking.

1. Select **initialization()** under <<Housekeeping>> Domain Support for <<Domain>>MicrowaveCooking in the Project Explorer.
2. Bring the context-sensitive **Action Language** View forward by clicking on its tab at the top. Note that it is usually in the same view group as the Properties View.



3. Please review the action language below. Note the generated IsOpen signal initiates processing in the domain by opening the Door. Copy (Control-C) and paste (Control-V) the Platform-independent Action Language (PAL) into the Action Language view:

```
// 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);
```

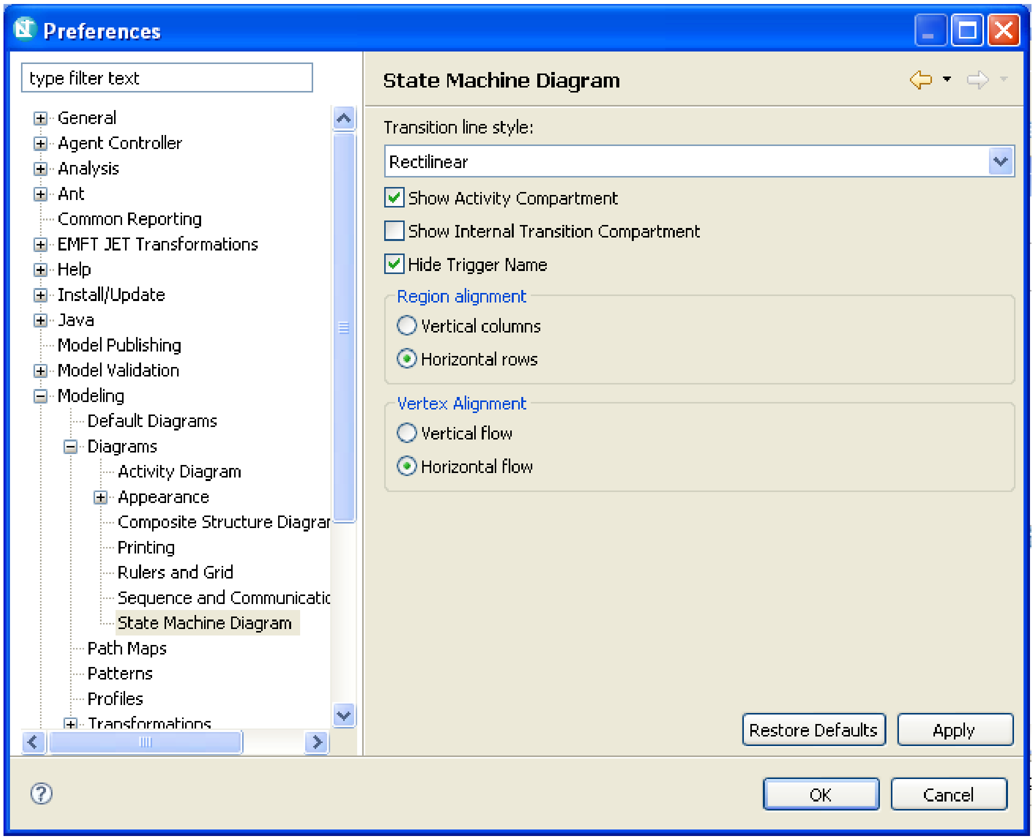
Task 7: Create the Door State Machine

The door State Machine shows transitions between closed and open. Create the door State Machine in four parts:

- Create and Name the Door State Machine diagram.
- Place the Door State Symbols
- Draw Transition Lines for the Door States
- Create Signals and Triggers for the Door State Machine

PROCEDURE: Use Eclipse Window Preferences Dialog to Configure State Machine Modeling Diagrams to Show Activity Compartment and Set Default Line Style

1. Click **Window > Preferences...** to open the Preferences dialog.
2. Expand **Modeling**, then expand **Diagrams**, then select **State Machine Diagram**.
3. Check the **Show Activity Compartment** check box.
4. On the Transition line style combo box, select Rectilinear.



State Machine Diagram

Transition line style: Rectilinear

☑ Show Activity Compartment

□ Show Internal Transition Compartment

☑ Hide Trigger Name

Region alignment

○ Vertical columns

● Horizontal rows

Vertex Alignment

○ Vertical flow

● Horizontal flow

5. Click **OK** to save the preferences.

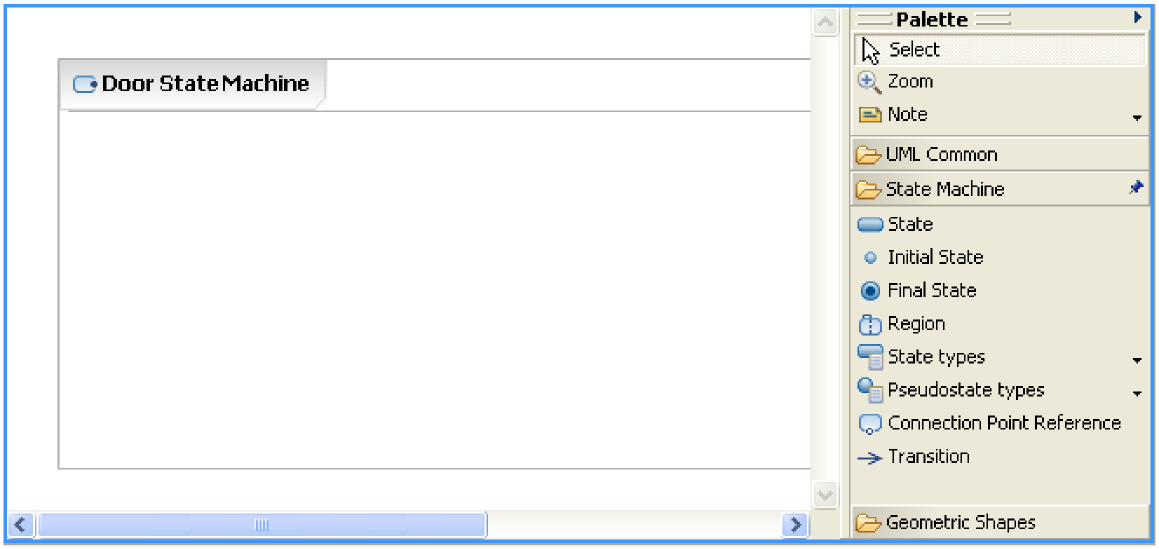
6. These global settings affect newly added states and transitions. Existing drawings will not be modified.

   To show the action compartment for an existing state, select the state on the State Machine Diagram. Right-click and select **Filters > Show / Hide Compartment > Show Action Compartment**.

   To change the style of a transition, select the transition. From the main menu bar, select **Diagram > Line Style > Rectilinear Style Routing**.

PROCEDURE: Create and Name the Door State Machine Diagram

1. Right-click the **Door** class in the Project Explorer and select **Add Diagram > State Machine Diagram** in the pop-up menu. A blank state machine diagram named *StatemachineDiagram1* appears in the editor pane.
2. Expand the Door class and rename the diagram *StatemachineDiagram1* to *Door_State_Machine*.
3. Select State Machine *StateMachine1* and rename it to *Door State Machine*.

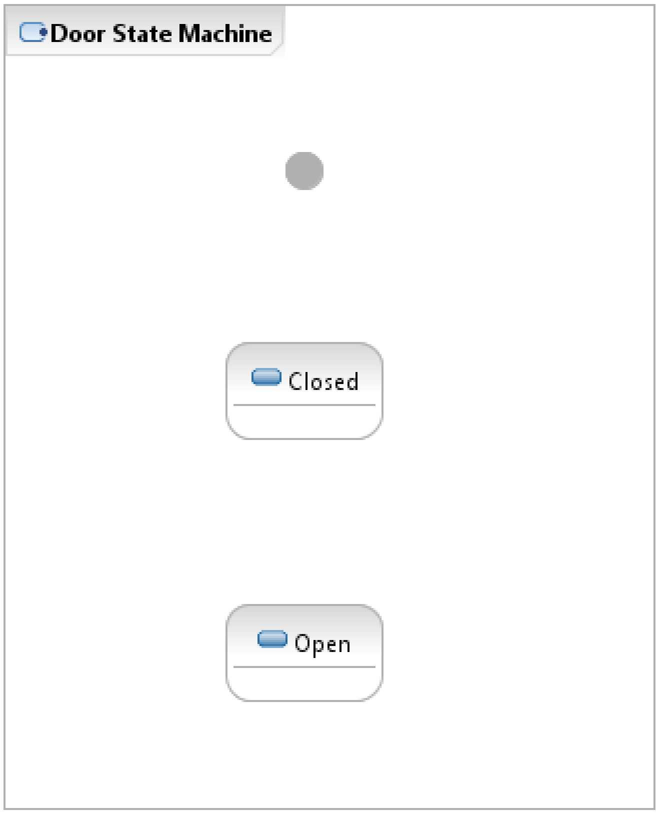


Note the Palette. You’ll be using the Initial State, State, and Transition tools in the following Procedures.

PROCEDURE: Place the Door State Symbols

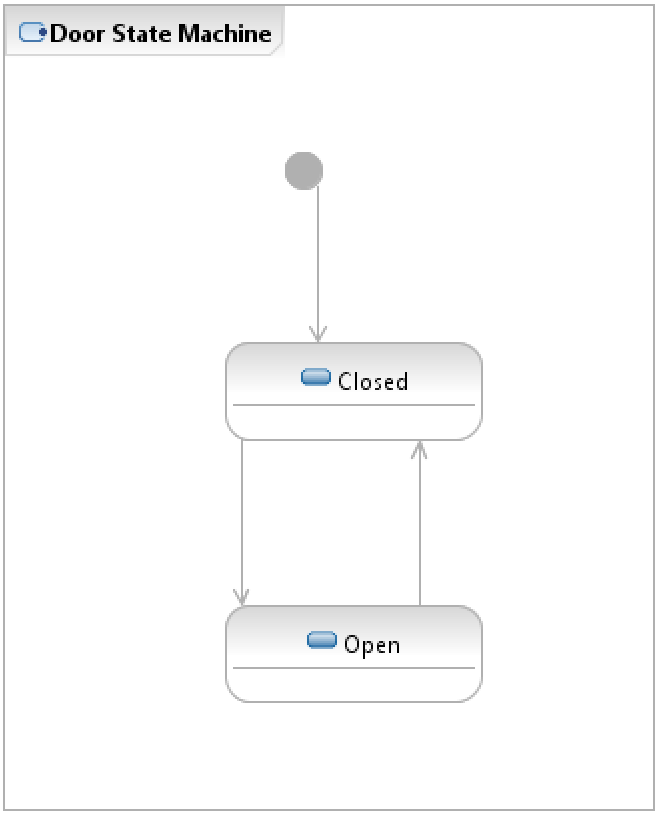
To place the state symbols in the new State Machine:

1. With the techniques learned in completing previous diagrams, use the Initial State and State tools on the Palette to populate the Door State Diagram with the following:
   - (unnamed) Initial State
   - *Closed* State, and
   - *Open* State.
2. Name and position the states as in the diagram below:



PROCEDURE: Draw Transition Lines for the Door States

1. Using the Transition tool in the Palette. Draw the following transitions:
   - From the initial state to the Closed state,
   - From the Closed State to the Open State, and
   - From the Open State to the Closed State.
2. Resize the states selecting them and using their “grab handles” (small squares along the perimeter) and position the transitions as shown in the drawing below.



NOTE: To improve Diagram layouts, explore tools in the Diagram menu on the main menu bar.

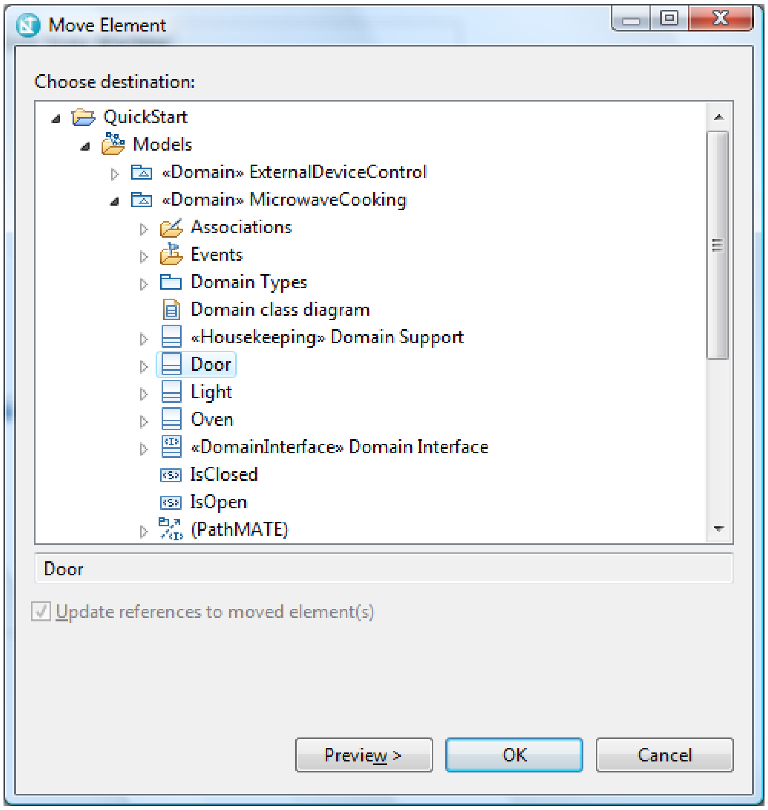
Task 8: Create Signals for the Door State Machine

UML2 defines Signals and Signal Triggers. A Signal describes an event and its parameters (if any). When created, a Signal is created within the enclosing Domain. By convention, PathMATE modelers then move them to the receiving class. A Signal trigger attaches to a State Transition and indicates which transitions may occur when the class receives the event.

PROCEDURE: Use Project Explorer to Add Signals to Domain Microwave Oven

1. In Project Explorer right-Click on **<<Domain>> MicrowaveCooking** to access its context menu and select **Add UML > Signal**.
2. Use edit-in-place to rename the Signal element so added under the Domain from *Signal1* to *IsOpen*.
3. In a similar manner, create the Signal *IsClosed*.
4. Use control click to select the IsOpen and IsClosed signals.
5. Right click and select **Refactor > Move**.
6. The Save All Modified Resources dialog appears. Click **OK**.

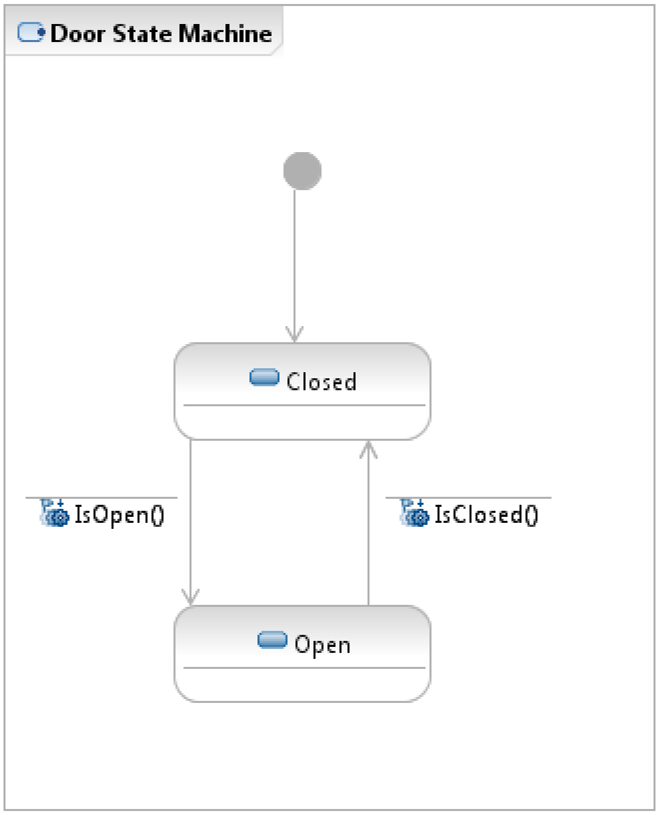
7. The Move Element dialog appears. Select **QuickStart > Models > MicrowaveCooking > Door**.
8. Click **OK**.
9. The Signals now appear in the Project Explorer under the Door class.



Move Element
Choose destination:
QuickStart
Models
«Domain» ExternalDeviceControl
«Domain» MicrowaveCooking
Associations
Events
Domain Types
Domain class diagram
«Housekeeping» Domain Support
Door
Light
Oven
«DomainInterface» Domain Interface
IsClosed
IsOpen
(PathMATE)
Door
☑ Update references to moved element(s)
Preview >
OK
Cancel

PROCEDURE: Use Drag & Drop to Add Signal Triggers to Transitions

1. Drag the *IsOpen* Signal in Project Explorer and position over the Transition line from *Closed* to *Open* in the diagram. Drop when you see the "+" Copy Cursor.
2. Select the **Transition** and in the context-sensitive Properties View select the **Triggers** tab. Verify that the Trigger has been added.
3. In a similar manner, add the *IsClosed* trigger to the transition from *Open* to *Closed*.



Task 9: Create the Light State Machine

The Light State Machine shows transitions between on and off. Create the Light State Machine in four parts:

- Create and Name the Light State Machine diagram
- Place the Light State Symbols
- Draw Transition Lines for the Light States
- Create Signals and Triggers for the Light State Machine

PROCEDURE: Create and Name the Light State Machine Diagram

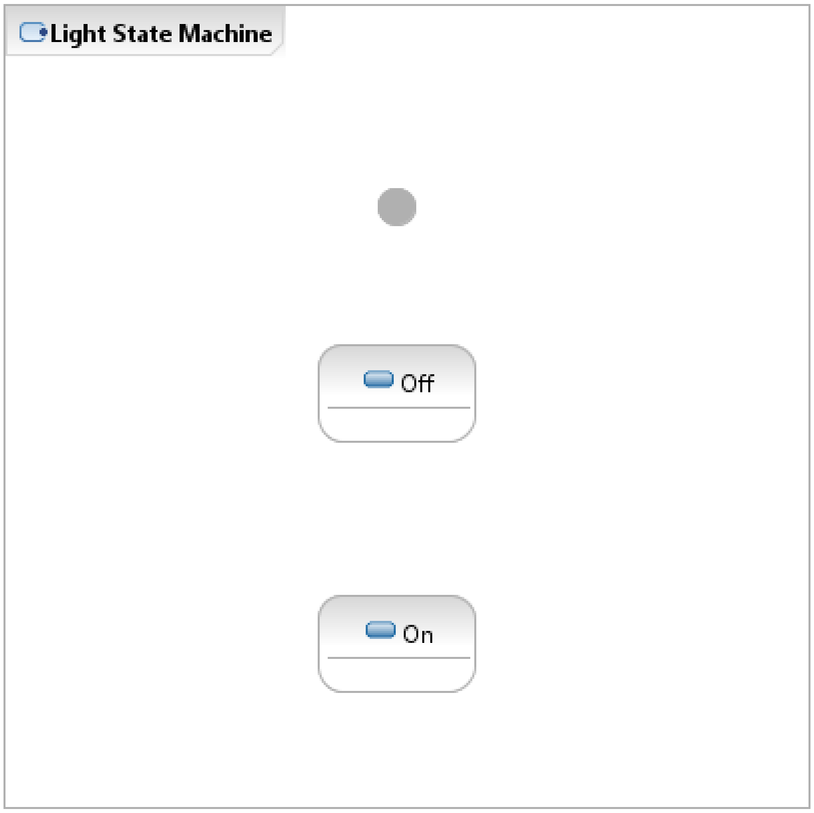
1. Right-click the Light class in the Project Explorer and select **Add Diagram > State Machine Diagram** in the pop-up menu. A blank state machine diagram named *StatemachineDiagram1* appears in the editor pane.
2. Expand the Light class and rename *StatemachineDiagram1* to *Light_State_Machine*.
3. Select State Machine *StateMachine1* and rename it to *Light State Machine*.

PROCEDURE: Place the Light State Symbols

To place the state symbols in the new State Machine:

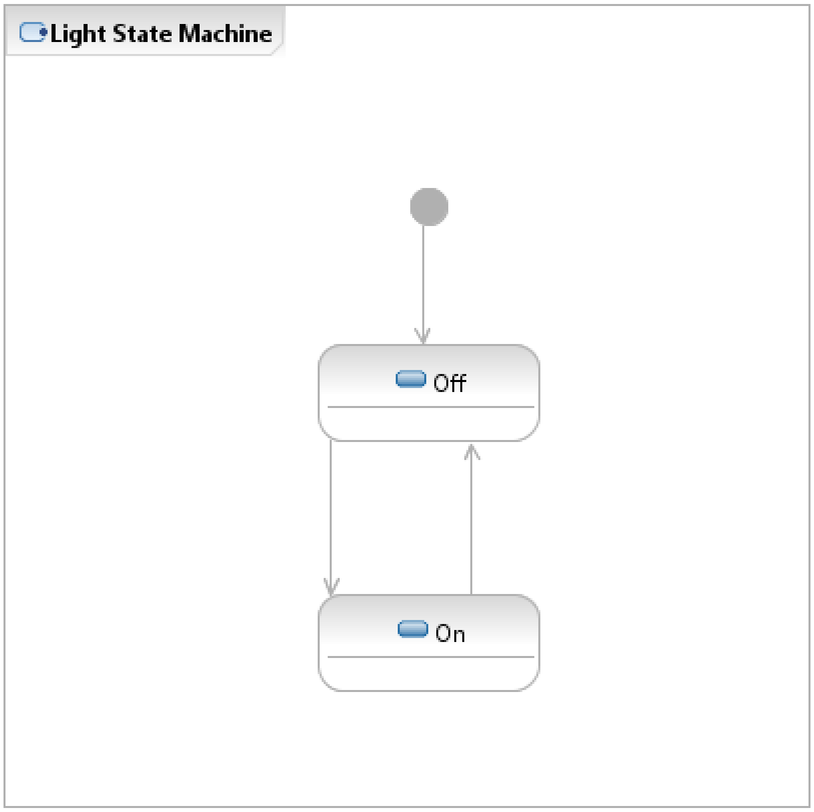
1. In the same way you added states to the Door State Machine Diagram, populate the Light State Machine with the following:
   - (unnamed) Initial State
   - *Off* State, and
   - *On* State.
2. Name and position the states as in the diagram below:

The figure below shows the Light State Machine after you create the initial, Off, and On states.



PROCEDURE: Draw Transition Lines for the Light States

1. Using the **Transition** tool in the Palette. Draw the transitions:
   - From the initial state to the Off state,
   - From the *Off* State to the *On* State, and
   - From the *On* State to the *Off* State.
2. Resize the states using their "grab handles" and position the transitions and transition names as shown in the drawing below.

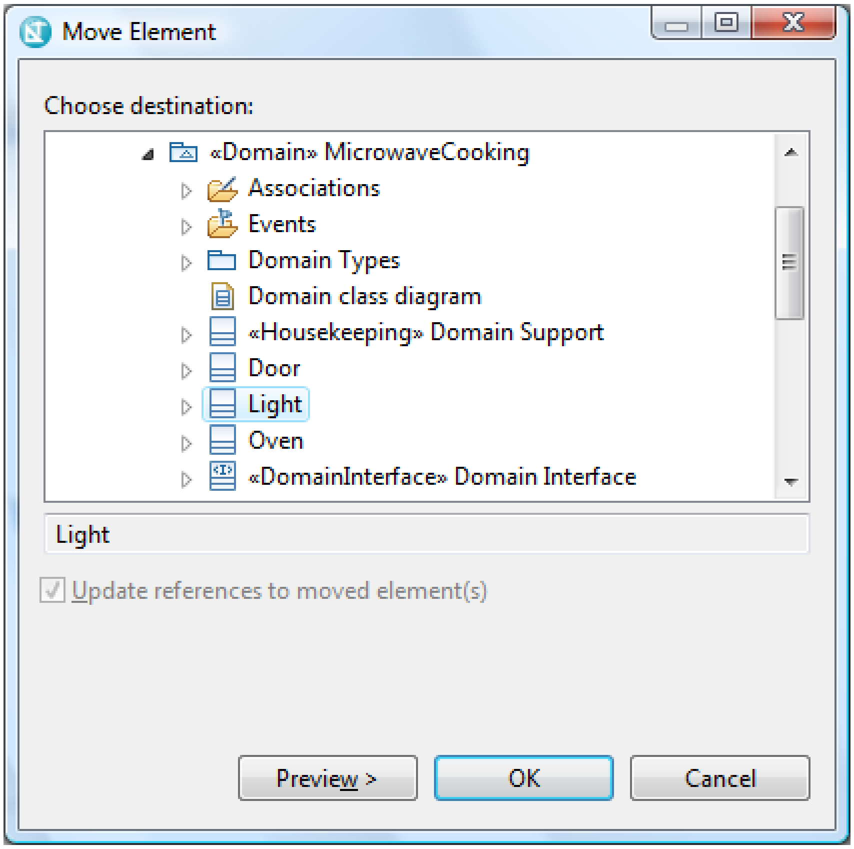


Task 10: Create Signals for the Light State Machine

PROCEDURE: Use Project Explorer to Add Signals to Domain Microwave Oven

1. In Project Explorer right-Click on **<<Domain>> MicrowaveCooking** to access its context menu and select **Add UML > Signal**.
2. Use edit-in-place to rename the Signal element so added under the Domain from *Signal1* to *TurnOn*.
3. In a similar manner, create the Signal *TurnOff*.
4. Use control click to select the TurnOn and TurnOff signals.
5. Right click and select **Refactor > Move**.
6. The Save All Modified Resources dialog appears. Click **OK**.

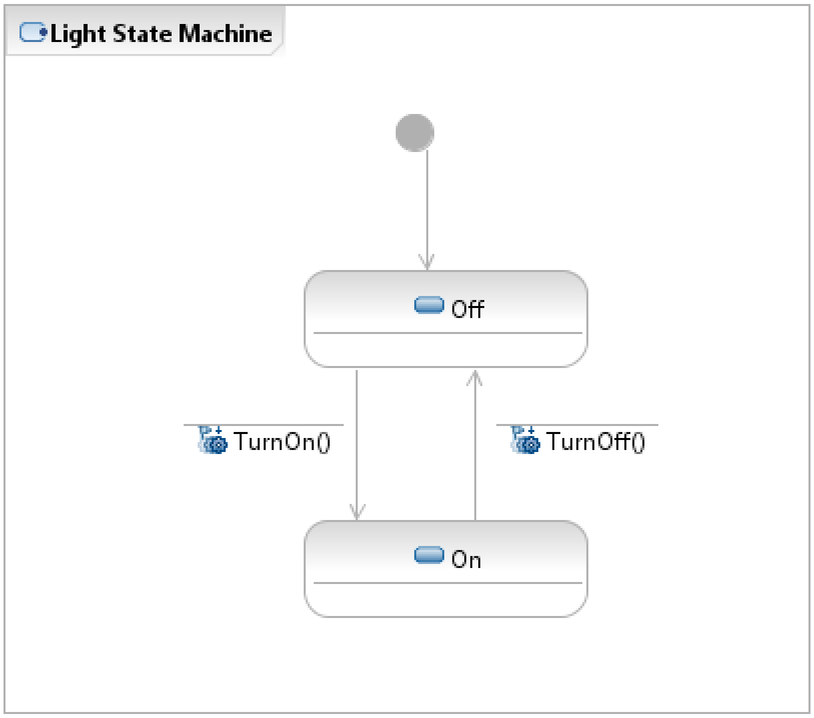
7. The Move Element dialog appears. Select **QuickStart > Models > MicrowaveCooking > Light**.
8. Click **OK**.
9. The Signals now appear under the Light class.



Move Element
Choose destination:
«Domain» MicrowaveCooking
Associations
Events
Domain Types
Domain class diagram
«Housekeeping» Domain Support
Door
Light
Oven
«DomainInterface» Domain Interface
Light
☑ Update references to moved element(s)
Preview >
OK
Cancel

PROCEDURE: Use Drag & Drop to Add Signal Triggers to Transitions

1. Drag the *TurnOn* Signal in Project Explorer and position over the Transition line from *Off* to *On* in the diagram. Drop when you see the "+" Copy Cursor.
2. Select the Transition and in the context-sensitive Properties View select the **Triggers** tab. Verify that the Trigger has been added.
3. In a similar manner, add the *TurnOff* trigger to the transition from *On* to *Off*.


Light State Machine
Off
TurnOn()
TurnOff()
On

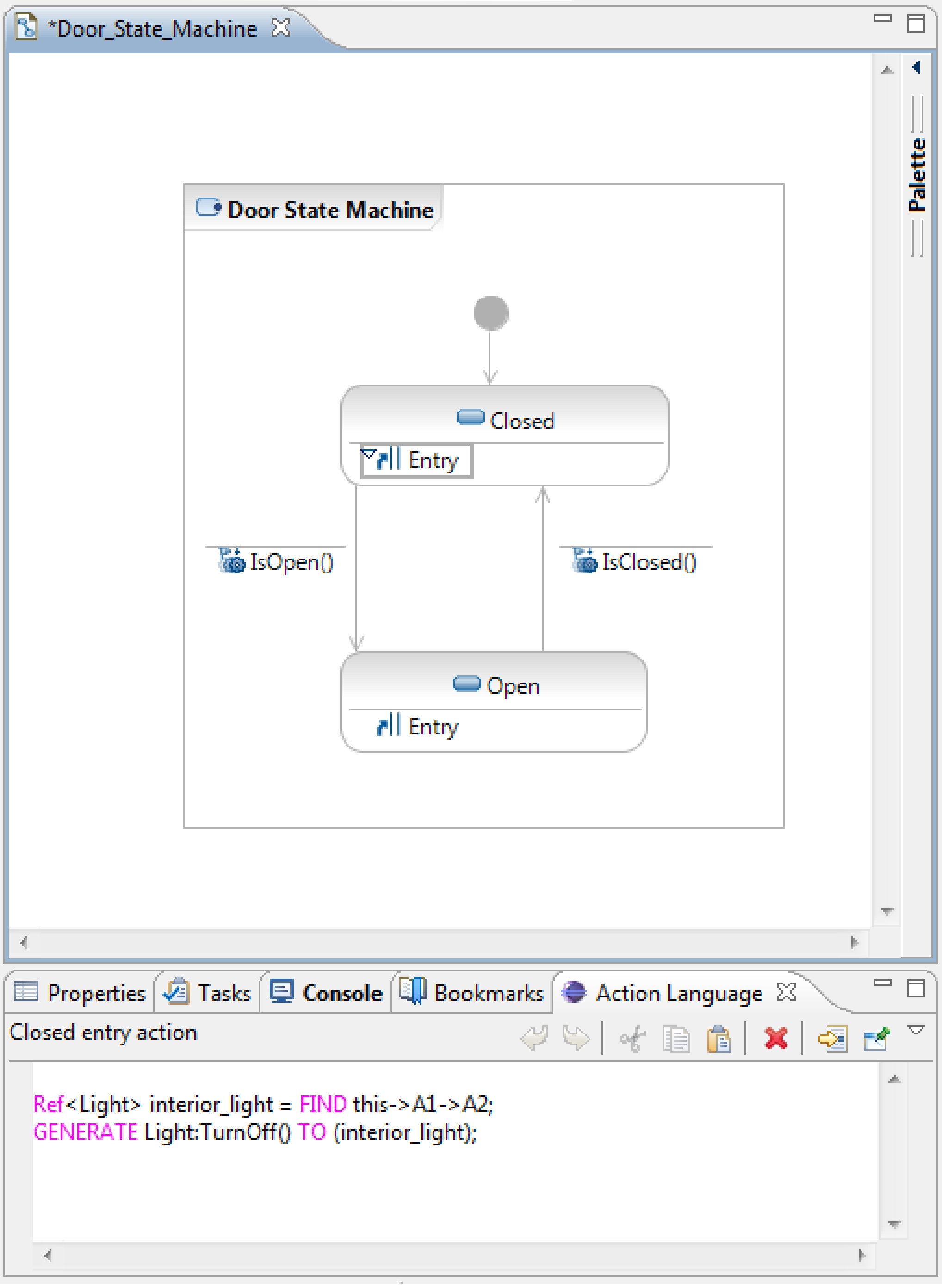
Task 11: Add Entry Actions

Entry actions establish a relationship between the Door state and the Light state. When the oven door is closed, the interior light is off. When the oven door is open, the interior light is on.

PROCEDURE: Use context menu and PathMATE Action Language View to Create an Entry Action for the Door State Machine's Closed State

1. Click on the Door_State_Machine tab in the editor pane to bring the Door state machine diagram to the top.
2. Right-click the Door State Machine's *Closed* State in the diagram and select **Add UML > Entry** and then pick **Create Activity** from the menu. Type **Enter** to accept the *Entry* name. Click on the label **Entry** in the state's activity compartment. Click on the Action Language View to bring it to the top. This will cause the context label *Closed entry action* to be displayed in the top of the Action Language View.
3. Copy the following snippet of PAL and paste it using Control-V in the Action Language view:

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



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

4. Repeat Step 1 to create an entry activity for the *Open* State, select it to make it active in Action Language view and copy in the following PAL:

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

5. Click on the Light_State_Machine tab in the editor pane to bring the Light state machine diagram to the top.
6. In the Light State Machine, create an entry activity for the *Off* State. Copy the following snippet of PAL and paste it in the Action Language view:

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

7. Create an entry activity for the *On* State, and copy the following snippet of PAL into the Action Language view:

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

8. From the main menu select **File > Save All** to save the model.

Congratulations! Your QuickStart project has a Real Model.

Prepare for Transformation

You now have a complete, PathMATE-ready MDA 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:

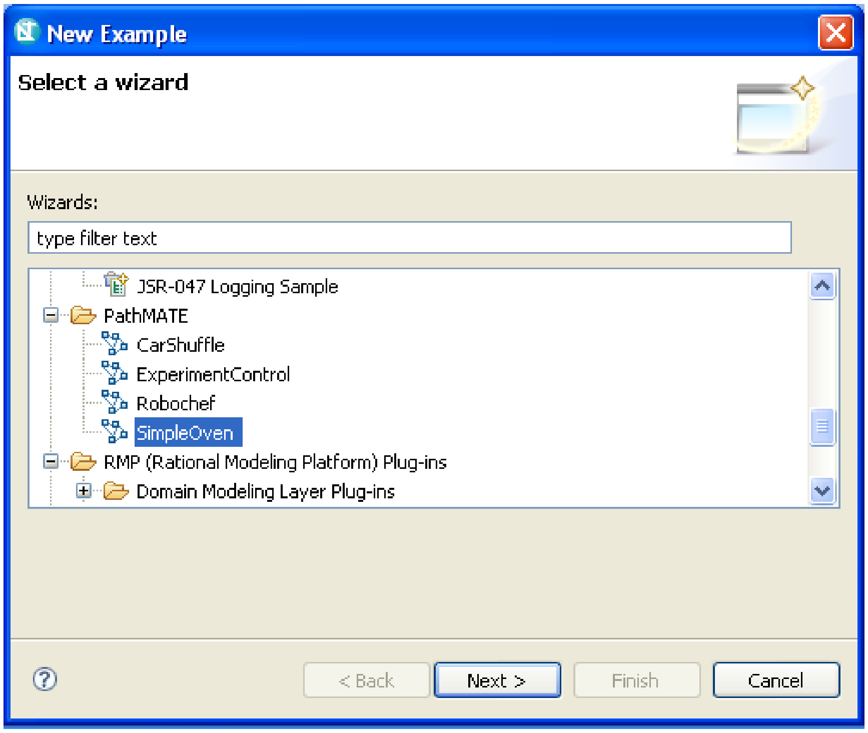
- Instantiate sample SimpleOven Reference Project and copy C++ and Java transformation properties and realized code from the Reference Project
- Add a PathMATE Project to the QuickStart project which will control transformation and deployment.

Task 1: Reuse elements from Example Project

SimpleOven exists as a sample model in the PathMATE examples library - part of the PathMATE Profile data in RSM. Instantiate SimpleOven as a reference project in order to copy the realized ExternalDeviceControl implementation and properties that guide the automatic generation of a Visual C++ project for this system.

PROCEDURE: Use Eclipse to Instantiate a Reference Project for SimpleOven

1. **Select File > New > Example** in the top menu bar. The New Example wizard opens.
2. Expand **PathMATE**, select **SimpleOven**, and click **Next**.

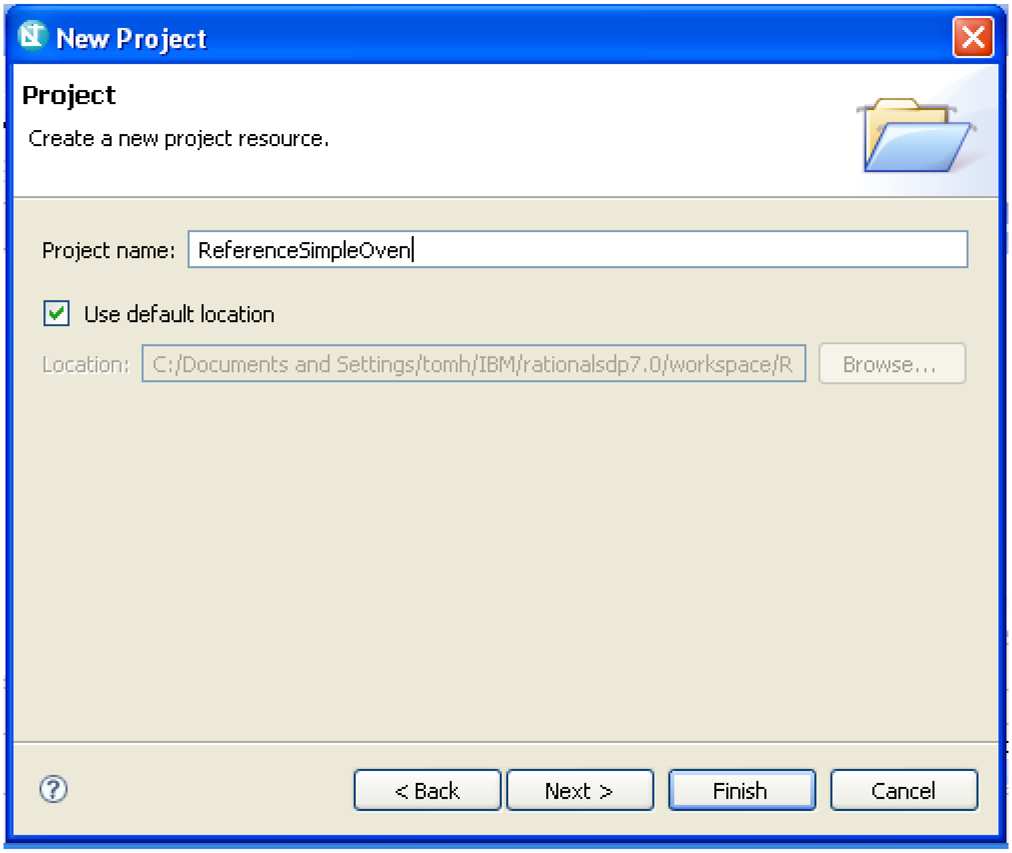

New Example

Select a wizard

Wizards:

type filter text

3. Type *ReferenceSimpleOven* in the Project name field.



New Project

Project

Create a new project resource.

Project name: ReferenceSimpleOven

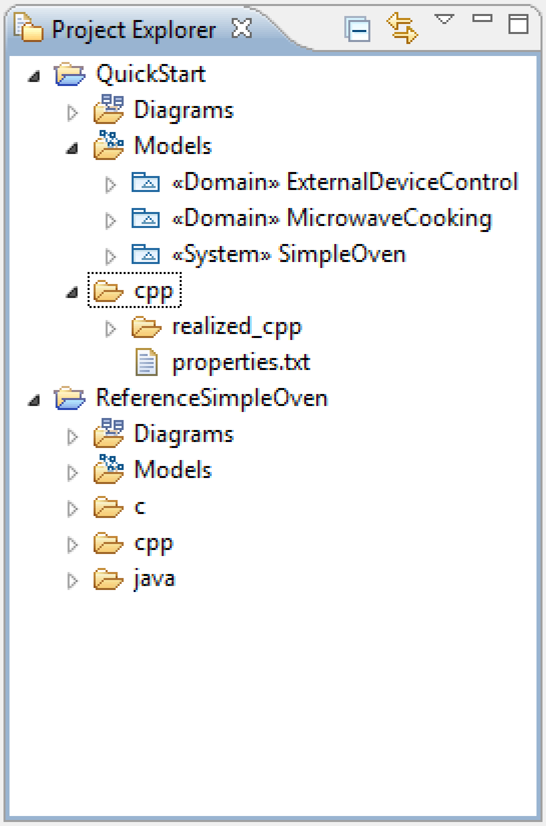
☑ Use default location

Location: C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/R

4. Click **Finish**. ReferenceSimpleOven appears in the Project Explorer.

PROCEDURE: Use Eclipse to Copy Realized Code Files

1. In the *ReferenceSimpleOven* project, select the cpp subdirectory, right click and select **Copy**.
2. Select the *QuickStart* project, right click and select **Paste**.
3. Expand the cpp folder in the resource browser:

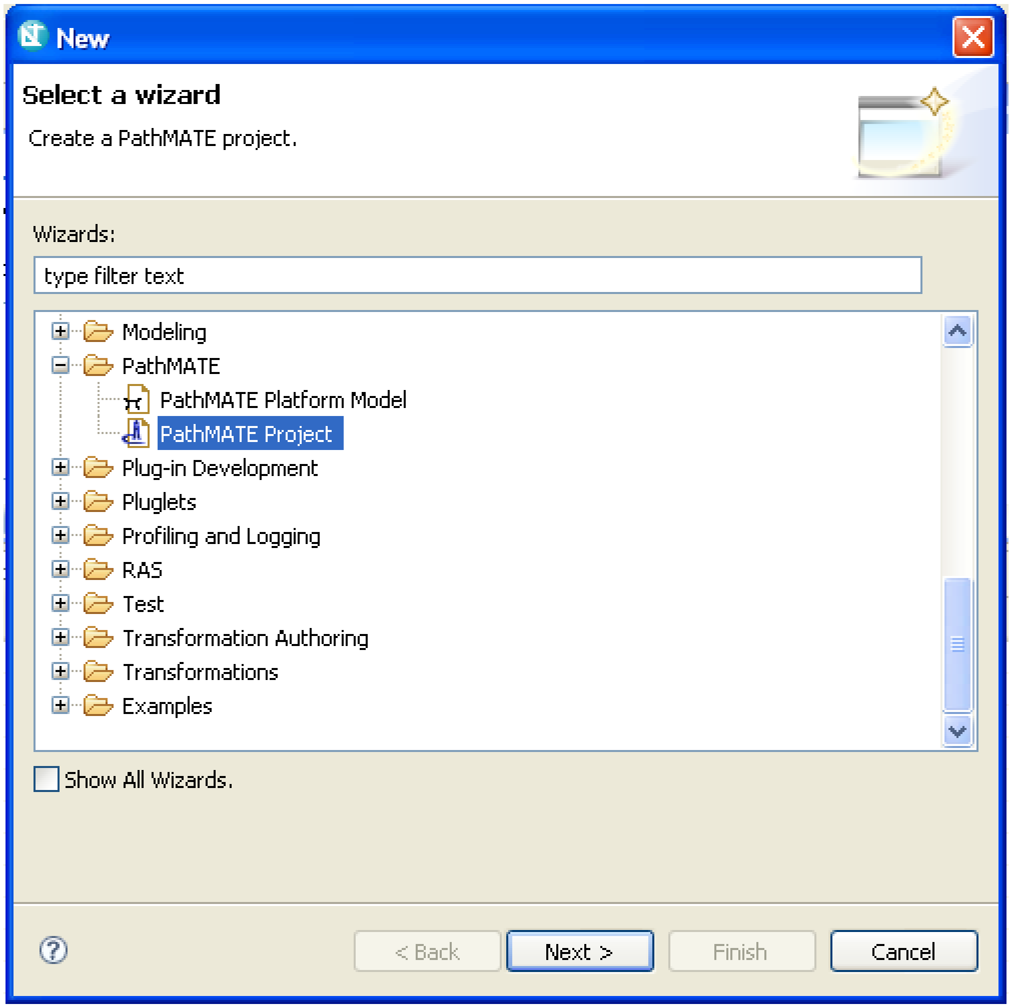


4. Repeat steps 2-4 to copy the *ReferenceSimpleOven java* subdirectory to the *QuickStart* project.

Task 2: Create PathMATE Project

PROCEDURE: Use Eclipse to Create a New PathMATE Project

1. Select **File > New > Other...** The New dialog box prompts you to select a wizard.
2. Enter Path in the wizard filter text.
3. Select **PathMATE Project** in the list of wizards. (If the PathMATE folder is not visible, click **Show All Wizards**.)



New

Select a wizard

Create a PathMATE project.

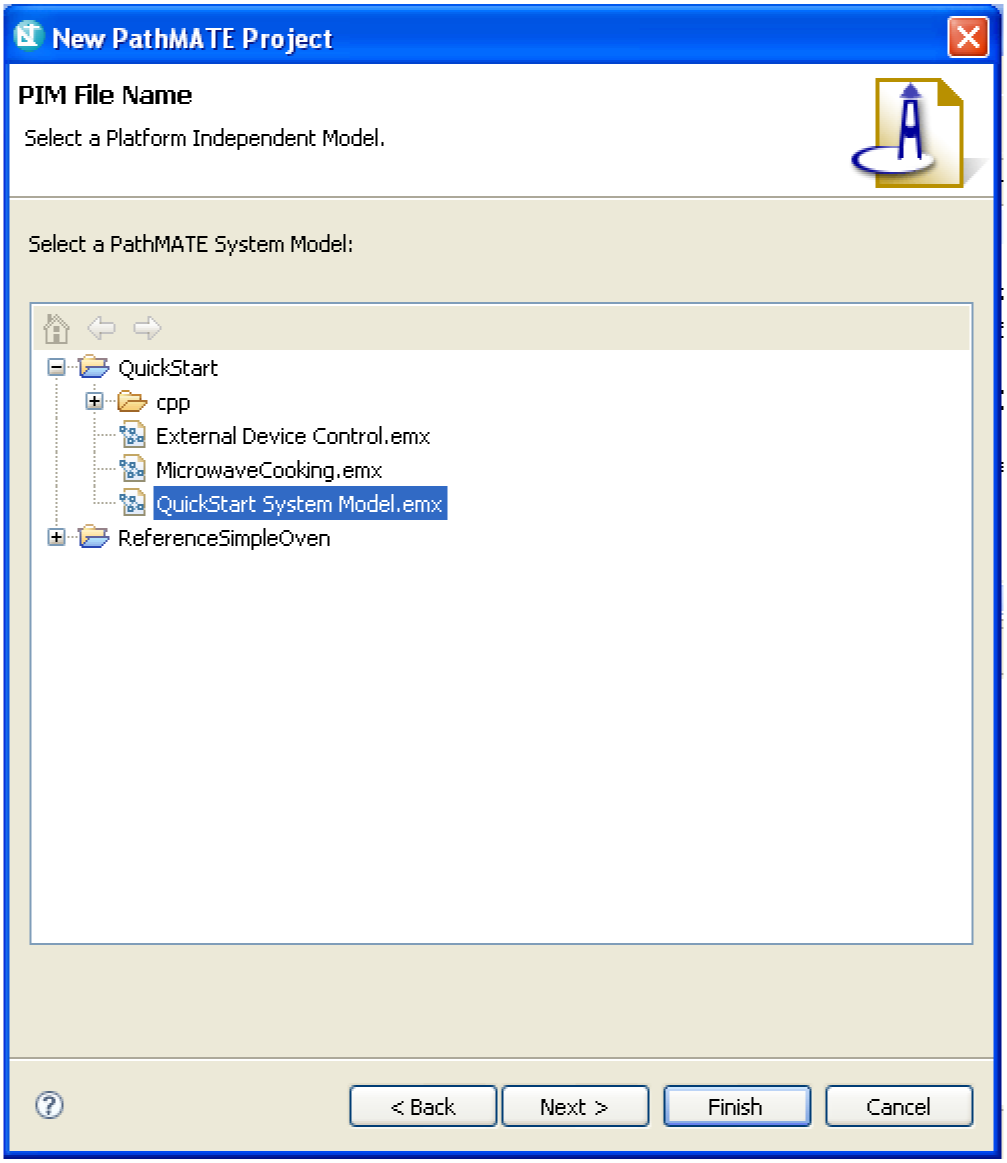
Wizards:

type filter text

4. Click **Next.**

The New PathMATE Project dialog box opens.

5. Click on the **QuickStart** project before expanding it. Select the Platform Independent Model named **QuickStart System Model.emx**.


New PathMATE Project

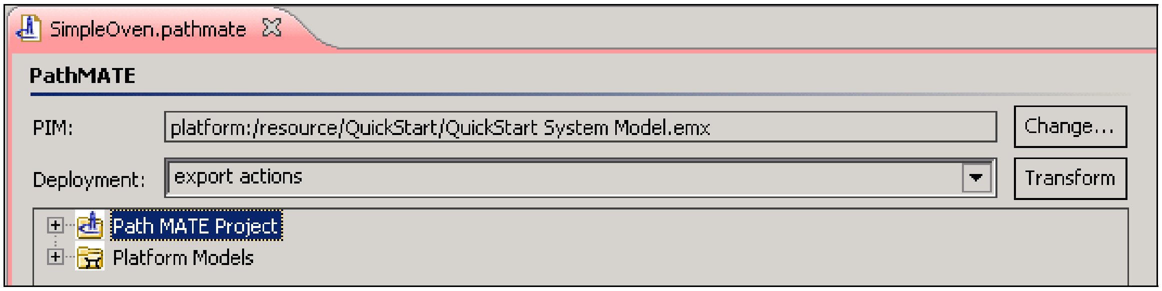
PIM File Name

Select a Platform Independent Model.

Select a PathMATE System Model:

- QuickStart
  - cpp
  - External Device Control.emx
  - MicrowaveCooking.emx
  - QuickStart System Model.emx
- ReferenceSimpleOven

6. Click **Next**. In the **File name** field enter *SimpleOven.pathmate* and click **Finish**. SimpleOven.pathmate opens in the PathMATE editor pane.


SimpleOven.pathmate

PathMATE

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

Deployment: export actions

Task 3: Check Your Model

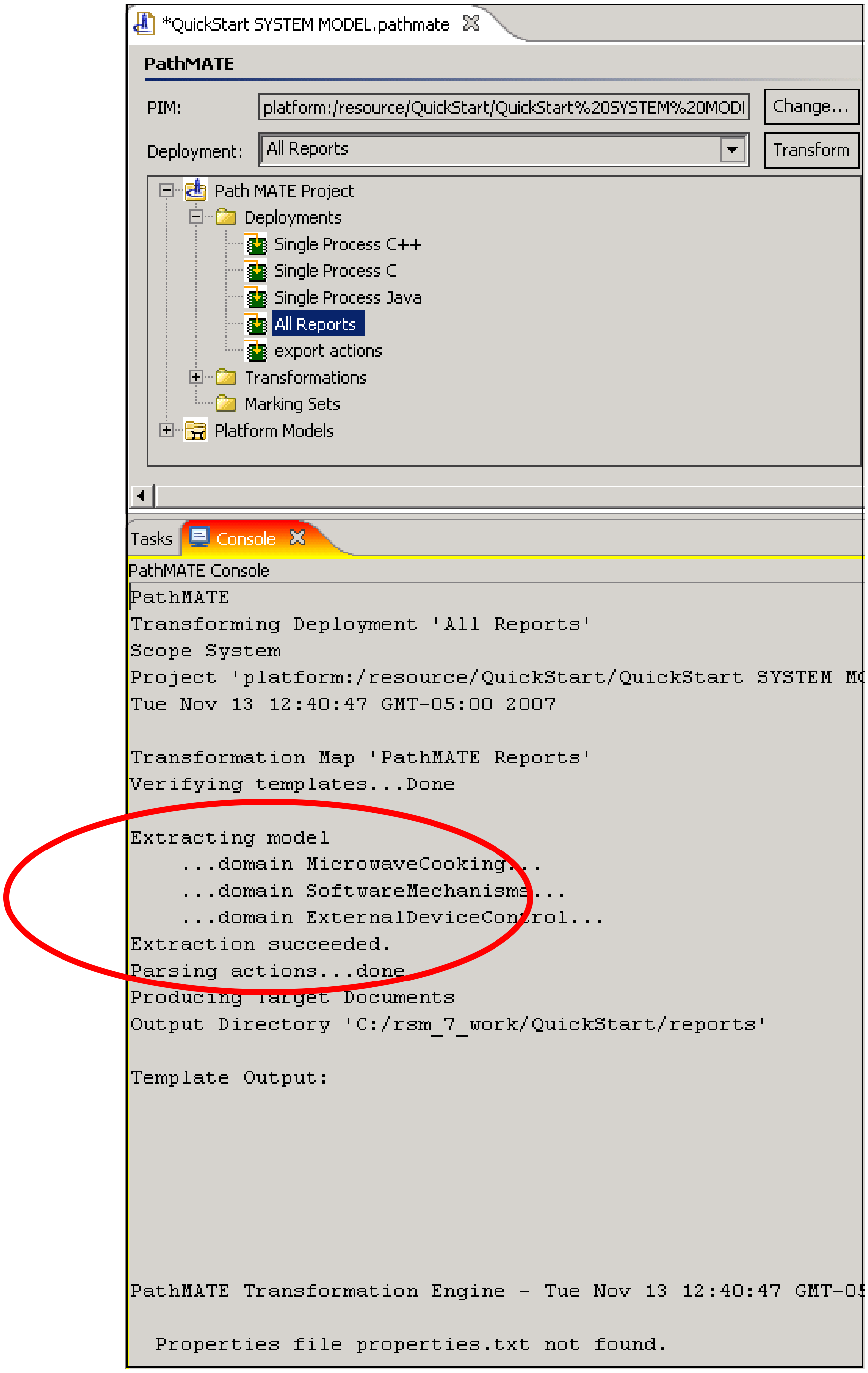
Before going on, this is a good point to check for any errors that may have occurred in entering your model.

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: Transform the new model to reports

1. Open **SimpleOven.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.

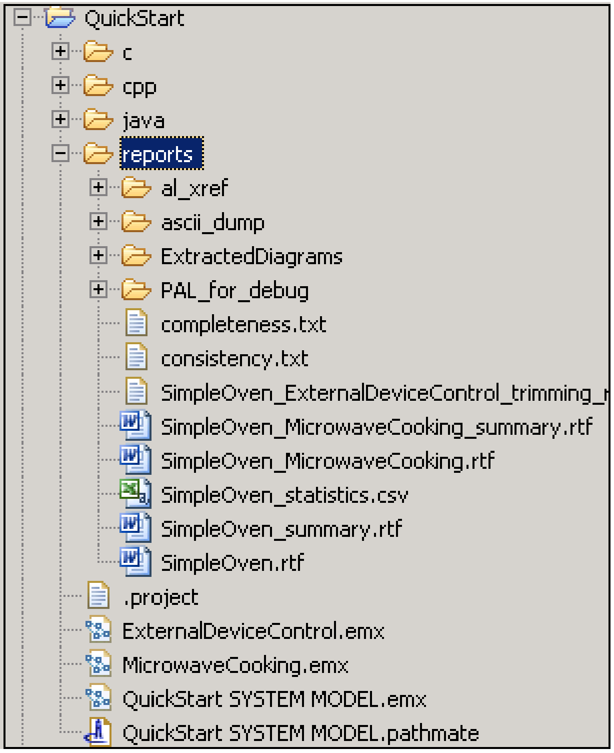


PIM: platform:/resource/QuickStart/QuickStart%20SYSTEM%20MODI

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 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. You may wish to use the Problems View to quickly navigate to the source of the error (**Window > Show View > Problems**).
8. Correct any problems identified in the messages, then return to SimpleOven.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

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

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

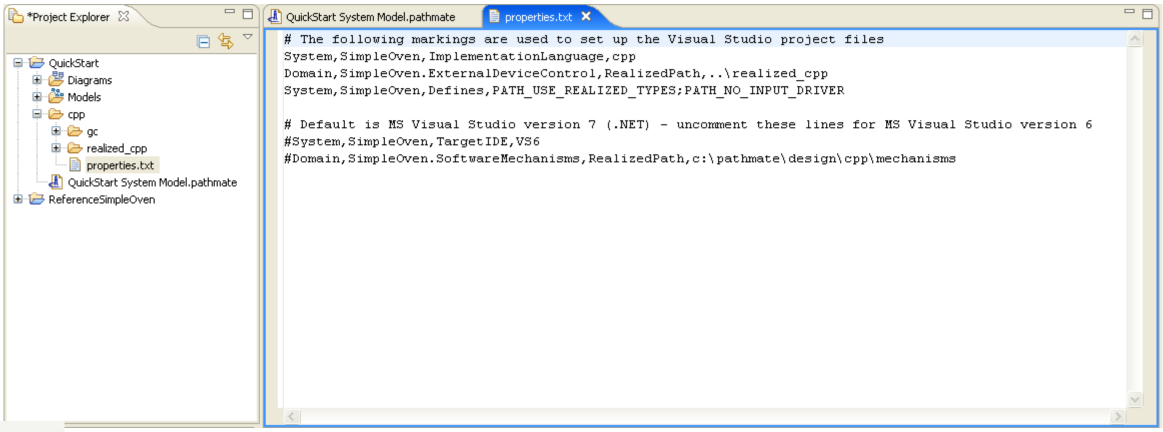
- Configure Development Environment Options,
- Transform Your Model to Code and Project Files

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. Open (double click) **QuickStart > cpp > properties.txt**



```
# 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. **File > Save** to save the changes

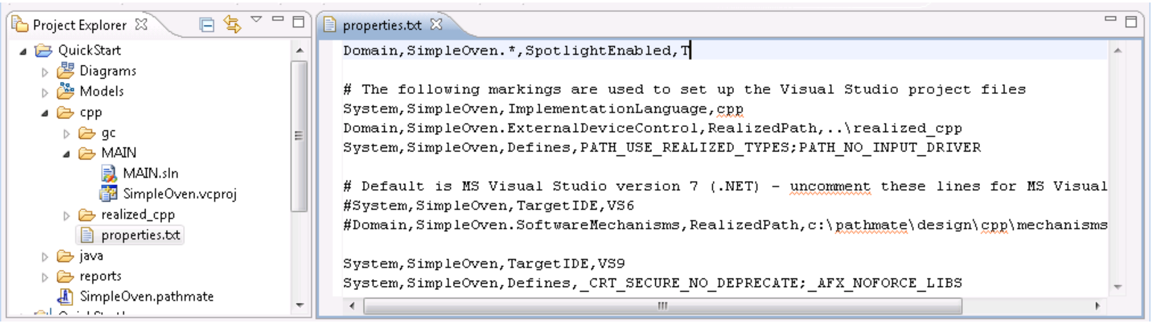
PROCEDURE: Change Markings for Visual Studio 2008 Express Edition ONLY

The default target when generating IDE project files is to support Visual Studio version 7 (SimpleOven.vcproj). To generate Visual Studio 2008 Express Edition project files change properties.txt to specify Visual Studio 9 as the Target IDE.

If you are not using Visual Studio 2008 Express Edition, skip this procedure.

1. Add the follow as the bottom line:

```
System,SimpleOven,TargetIDE,VS9
System,SimpleOven,Defines,_CRT_SECURE_NO_DEPRECATE;_AFX_NOFORCE_LIBS
```



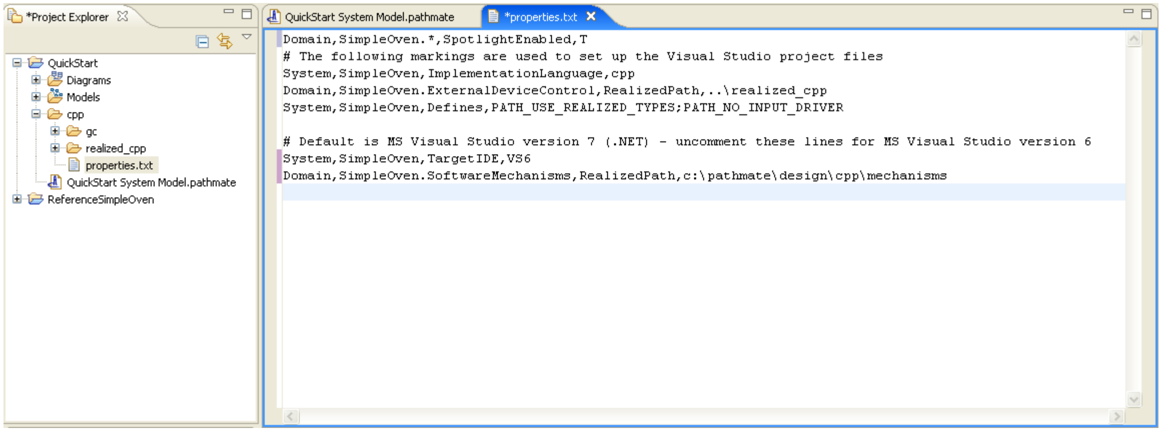
```
Domain,SimpleOven.*,SpotlightEnabled,T

# 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
#System,SimpleOven,TargetIDE,VS6
#Domain,SimpleOven.SoftwareMechanisms,RealizedPath,c:\pathmate\design\cpp\mechanisms

System,SimpleOven,TargetIDE,VS9
System,SimpleOven,Defines,_CRT_SECURE_NO_DEPRECATE;_AFX_NOFORCE_LIBS
```

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 (SimpleOven.dsp and SimpleOven.dsw) project files change properties.txt to specify Visual Studio 6 as the Target IDE.

If you are not using Visual Studio version 6, skip this procedure.

1. Uncomment the last 2 lines in the file



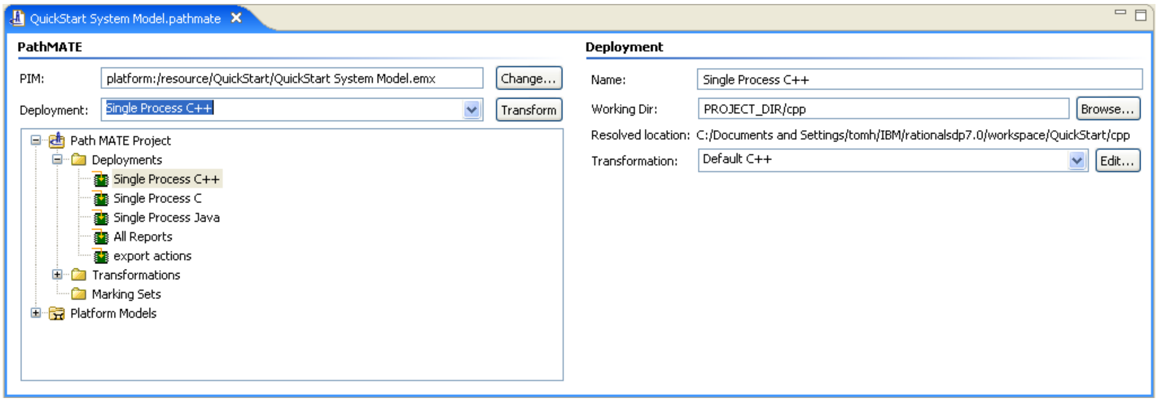
```
Domain,SimpleOven.*,SpotlightEnabled,T
# The following markings are used to set up the Visual Studio project files
System,SimpleOven,ImplementationLanguage,cpp
Domain,SimpleOven.ExternalDeviceControl,RealizedPath,..\realized_cpp
System,SimpleOven,Defines,PATH_USE_REALIZED_TYPES;PATH_NO_INPUT_DRIVER

# 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. **File > Save** (or Control – S) to save the changes

Task 2: Transform Your Model to Code and Project Files

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

1. Open **SimpleOven.pathmate** in the editor pane.
2. Select **Single Process C++** from the Deployment pick list.



PathMATE

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

Deployment: Single Process C++

Deployment

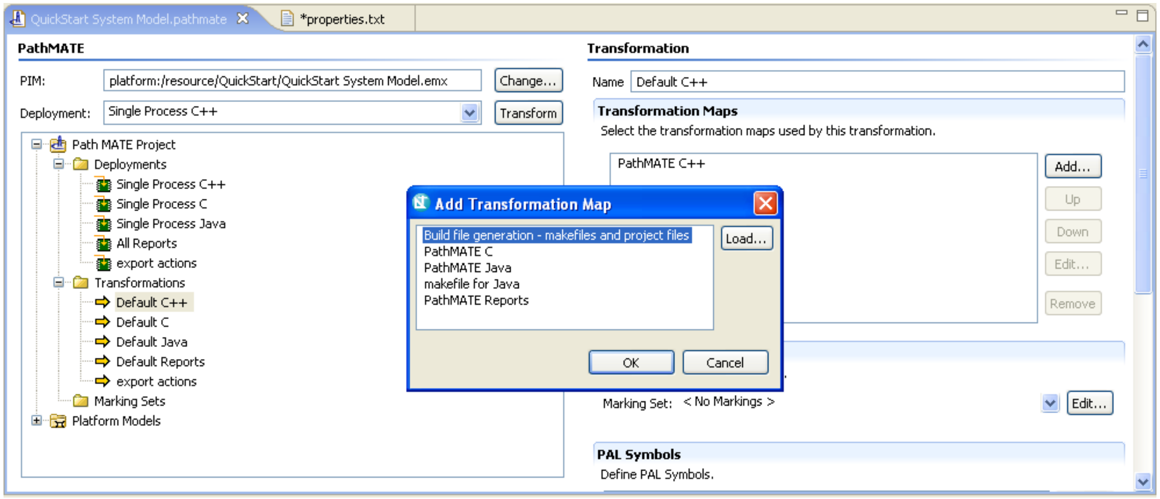
Name: Single Process C++

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.
4. In the Transformation area on the right side of the PathMATE Editor, click **Add**… The Add Transformation Map dialog appears.
5. Select *Build file generation – makefiles and project files*. Click **OK**.
6. In the Transformation Maps section ensure the PathMATE C++ map is first in the list and Build file generation Maps appears second. If not, select one of the maps and use the **Up/Down** buttons to correct the order.



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

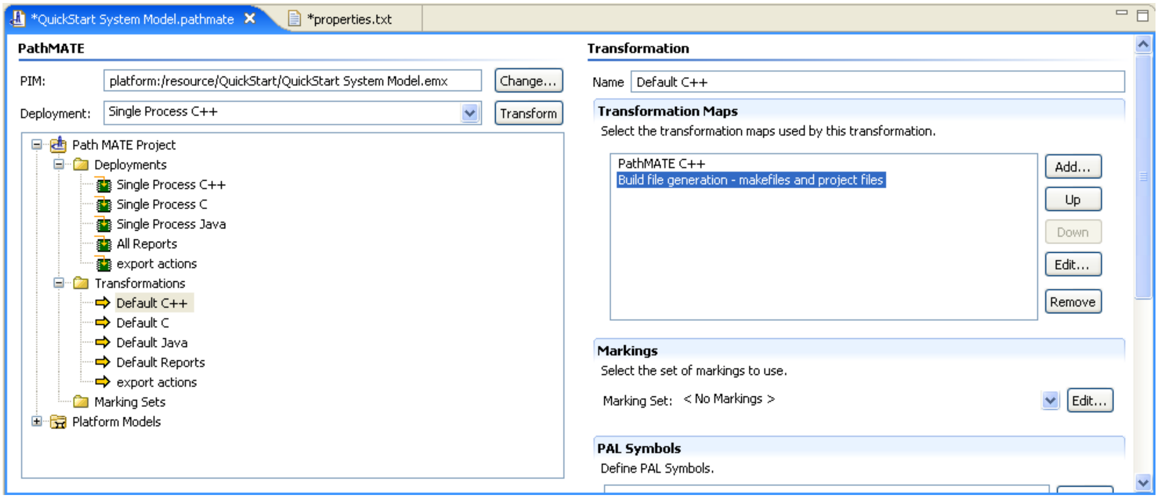
Deployment: Single Process C++

Name: Default C++

Transformation Maps: PathMATE C++

Add Transformation Map: Build file generation - makefiles and project files (selected); PathMATE C; PathMATE Java; makefile for Java; PathMATE Reports

Marking Set: < No Markings >



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

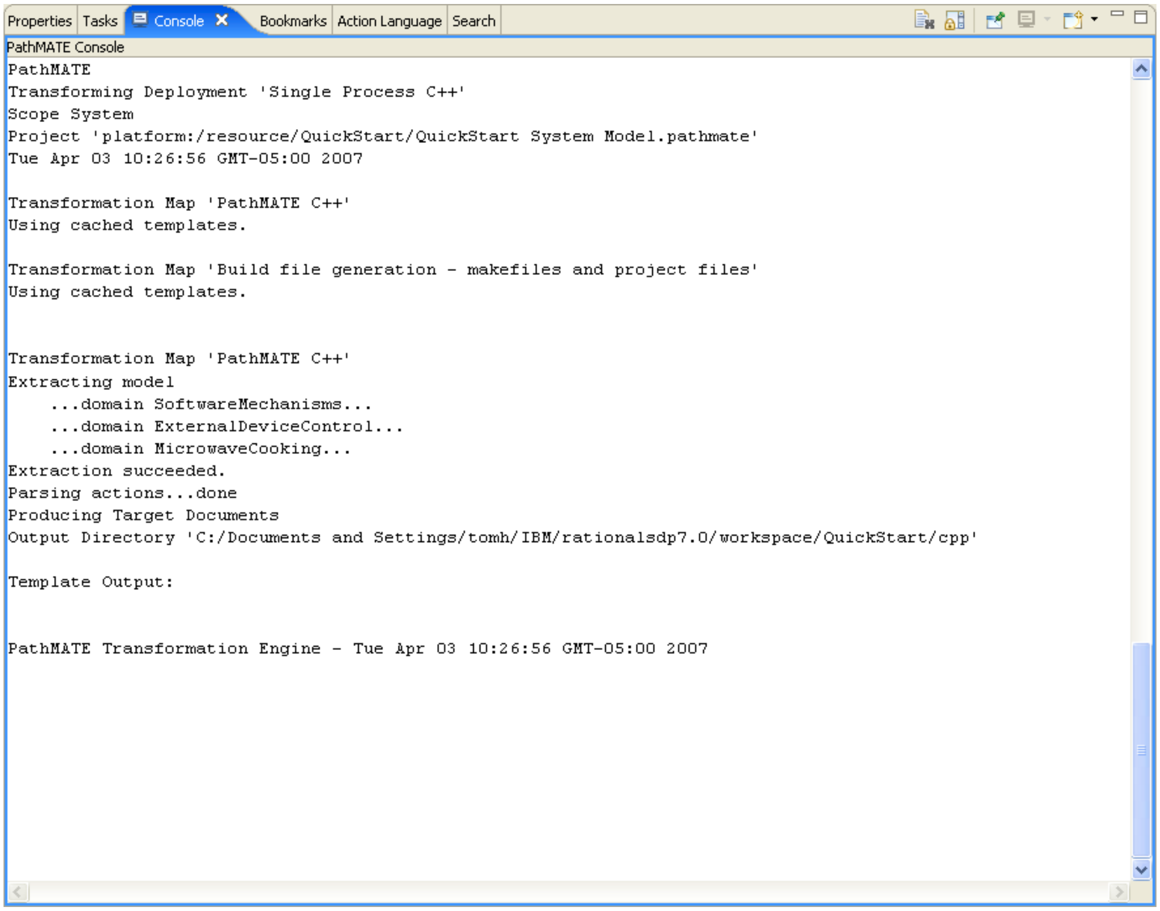
Deployment: Single Process C++

Name: Default C++

Transformation Maps: PathMATE C++; Build file generation - makefiles and project files (selected)

Marking Set: < No Markings >

7. Click **Transform**. A progress bar appears.
8. 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
```

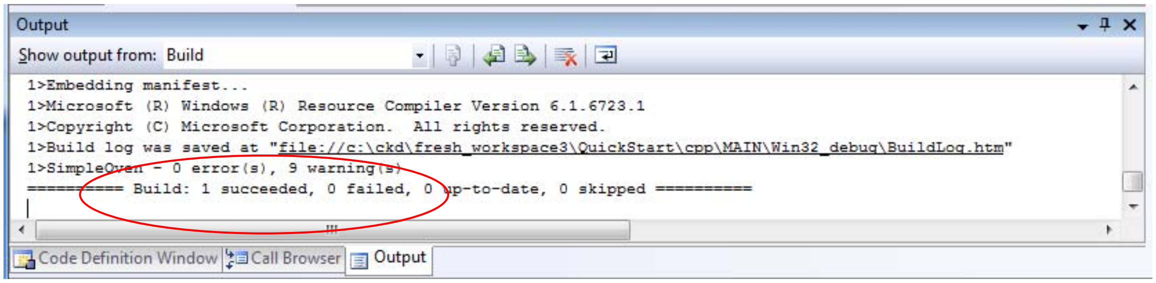
Task 3: Build an Executable System

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 2008 Express Edition

1. In the Project Explorer, right-click **QuickStart > cpp > MAIN > MAIN.sln**. Select **Open With > System Editor**. The generated Visual Studio Solution opens.

2. From Visual Studio 2008 Express Edition, select Build > Build Solution.
3. Examine the Output window to determine if the build succeeded.



```
1>Embedding manifest...
1>Microsoft (R) Windows (R) Resource Compiler Version 6.1.6723.1
1>Copyright (C) Microsoft Corporation.  All rights reserved.
1>Build log was saved at "file://c:\ckd\fresh_workspace3\QuickStart\cpp\MAIN\Win32_debug\BuildLog.htm"
1>SimpleOven - 0 error(s), 9 warning(s)
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

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

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

Congratulations! You now have a complete C++ executable 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:

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

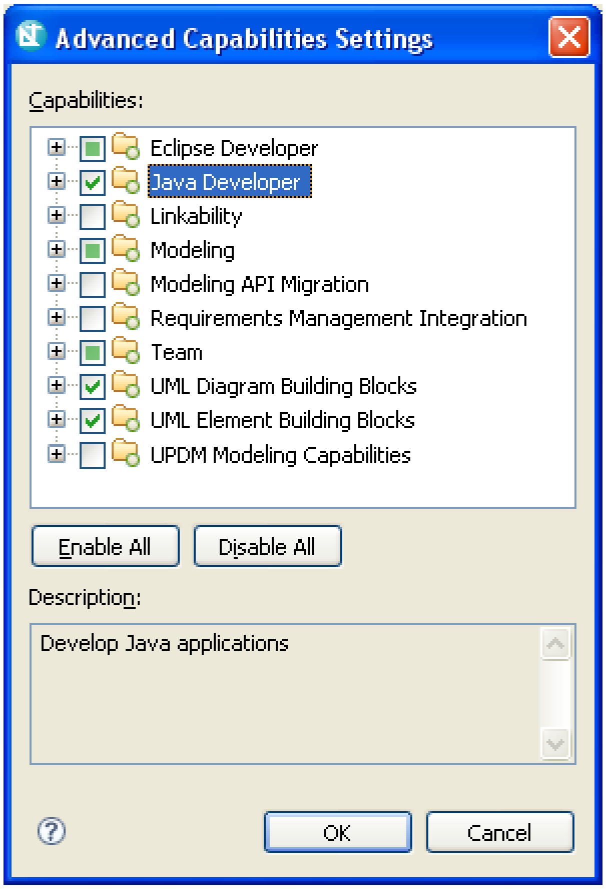
Task 1: Set Up Java, Create a Java Project, and Build

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. Ensure **Java Developer** is checked.



Advanced Capabilities Settings

Capabilities:

- ■ Eclipse Developer
- ☑ Java Developer
- □ Linkability
- ■ Modeling
- □ Modeling API Migration
- □ Requirements Management Integration
- ■ Team
- ☑ UML Diagram Building Blocks
- ☑ UML Element Building Blocks
- □ UPDM Modeling Capabilities

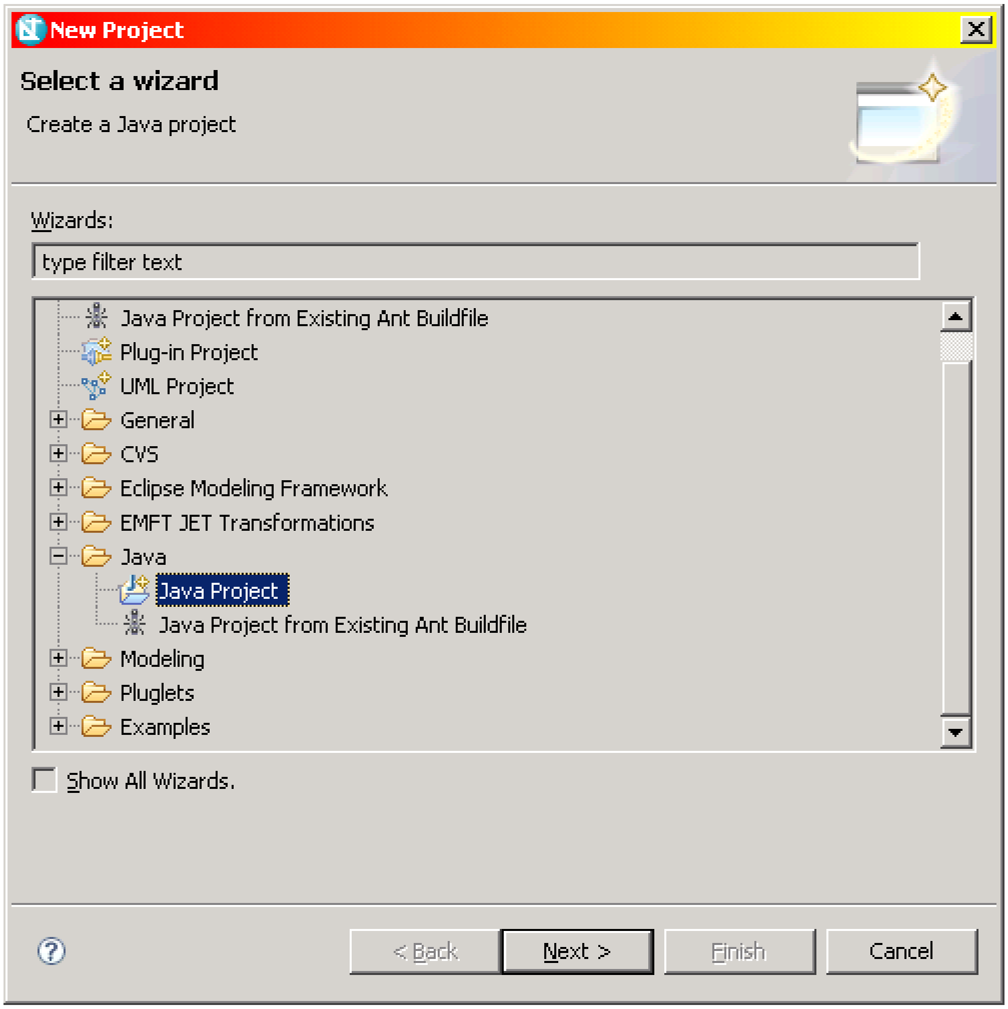
Description:

Develop Java applications

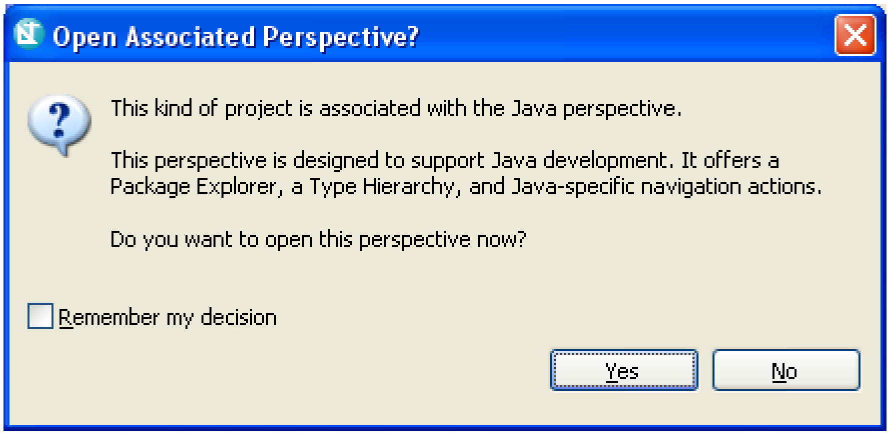
3. Click **OK** on both dialogs.

PROCEDURE: Create a Java Project

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 **Finish**. Eclipse may display the Open Associated Perspective? Dialog. Click **Yes** to switch to the Java perspective.



This kind of project is associated with the Java perspective.

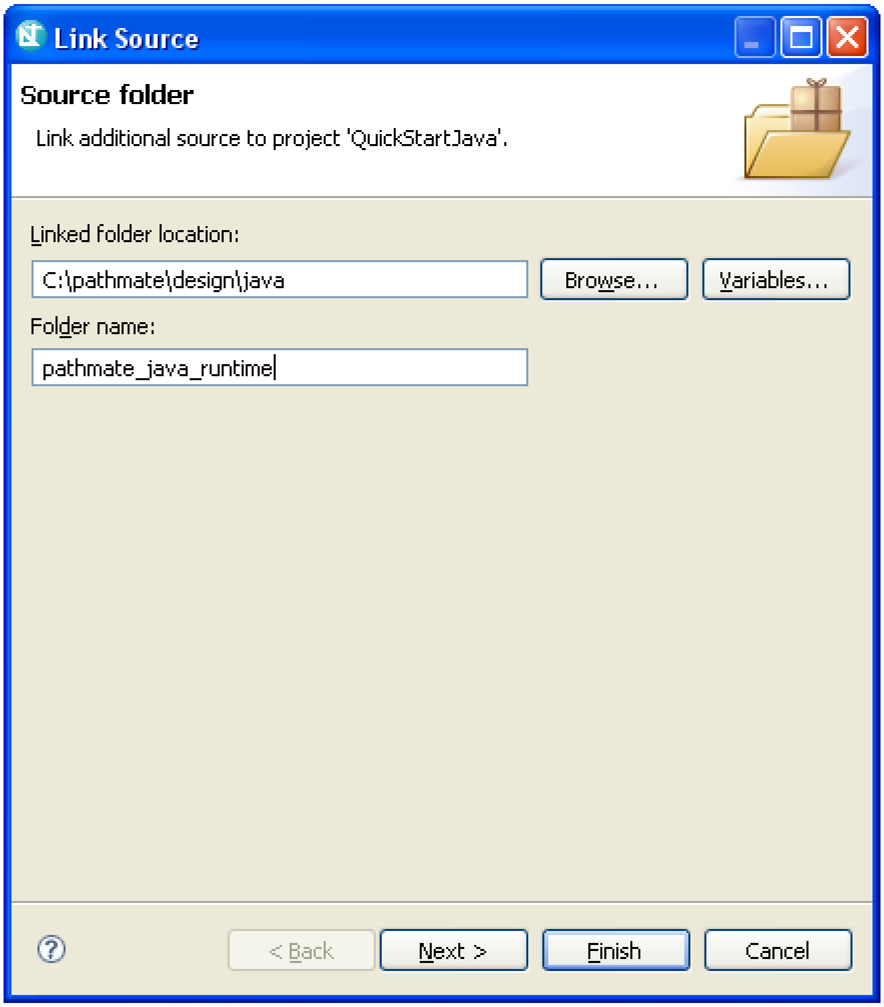
This perspective is designed to support Java development. It offers a Package Explorer, a Type Hierarchy, and Java-specific navigation actions.

Do you want to open this perspective now?

☐ Remember my decision

6. In the Package Explorer, select **ReferenceSimpleOven > java > properties.txt**.
7. Hit control-C to copy it.
8. Select *QuickStartJava*.
9. Hit control-V to paste it.
10. In the Package Explorer, select **ReferenceSimpleOven > java > realized_java**.
11. Hit control-C to copy it.
12. Select **QuickStartJava > src** source folder.
13. Hit control-V to paste the realized_java code.
14. In the Package Explorer browser, select your *QuickStartJava* project, right click, and select **Build Path > Link Source…**

15. Click **Browse…**, navigate to c:\pathmate\design and select the **java** folder.
16. Click **OK**.
17. Enter *pathmate_java_runtime* in Folder name.



Source folder

Link additional source to project 'QuickStartJava'.

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

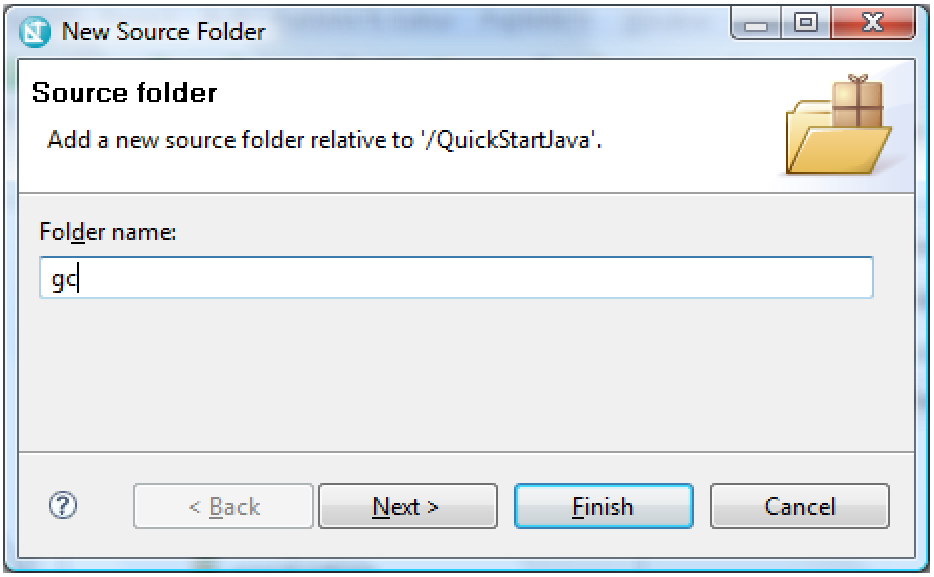
Folder name: pathmate_java_runtime

18. Click **Finish**.

19. In the Package Explorer browser, select *QuickStartJava* project, right click, and select **Build Path > New Source Folder**. The New Source Folder dialog appears.

20. Enter *gc* in the Folder name field. The gc folder will hold the Java code generated from the model. This directory will be populated by a transformation in a subsequent step.

21. Click **Finish**.



New Source Folder

Source folder

Add a new source folder relative to '/QuickStartJava'.

Folder name: gc

PROCEDURE: Change Markings and Defines to Enable Spotlight Instrumentation

1. Open **QuickStartJava/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**.

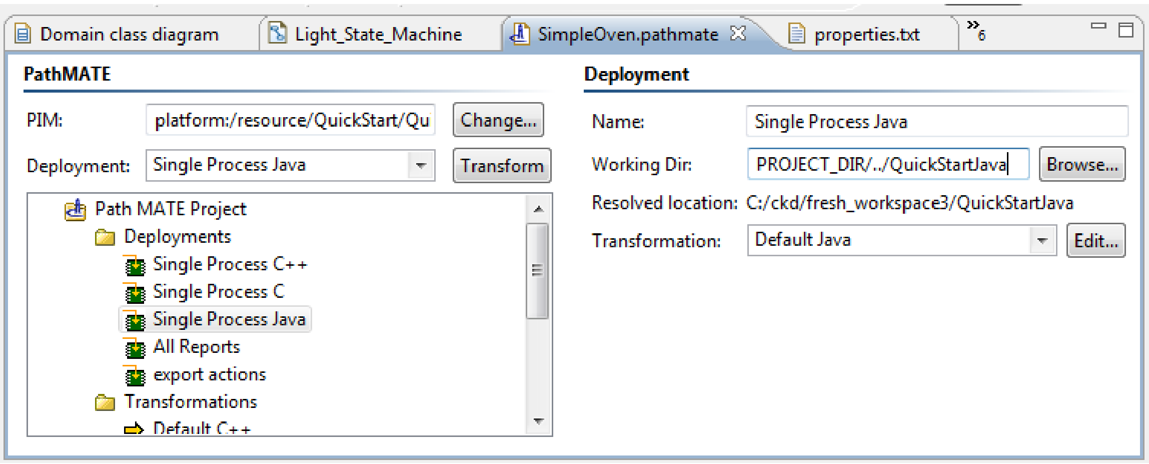
2. Double click to open **PfdDefine.java**. The file opens in the Editor pane. Set the *NO_PATH_IE* to *false* as shown below:

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

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

PROCEDURE: Generate Java Code

1. Open **QuickStart > SimpleOven.pathmate**. Select **Single Process Java** under the Deployments.
2. On the right hand side of the PathMATE editor, enter in the Working Dir field *PROJECT_DIR/../QuickStartJava*.



PathMATE

PIM: platform:/resource/QuickStart/Qu

Deployment: Single Process Java

- Path MATE Project
  - Deployments
    - Single Process C++
    - Single Process C
    - Single Process Java
    - All Reports
    - export actions
  - Transformations
    - Default C++

Deployment

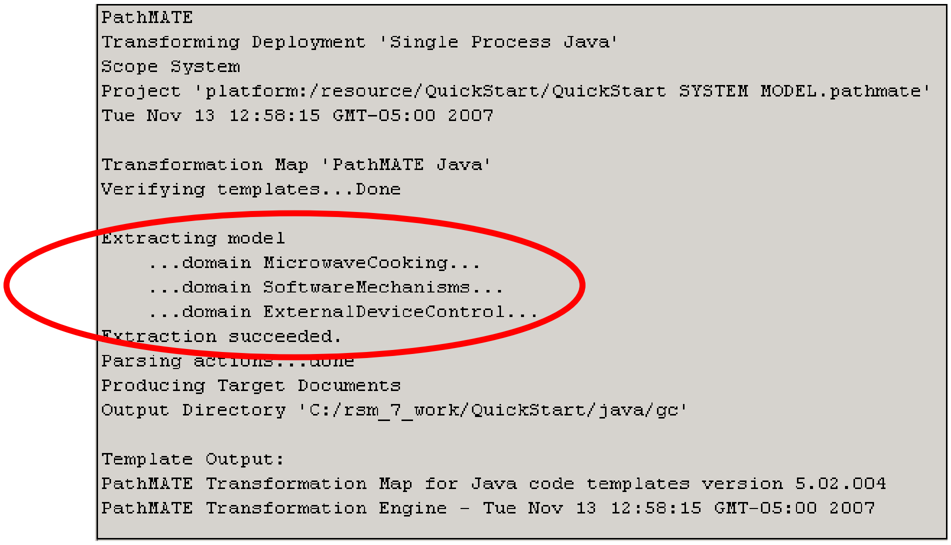
Name: Single Process Java

Working Dir: PROJECT_DIR/../QuickStartJava

Resolved location: C:/ckd/fresh_workspace3/QuickStartJava

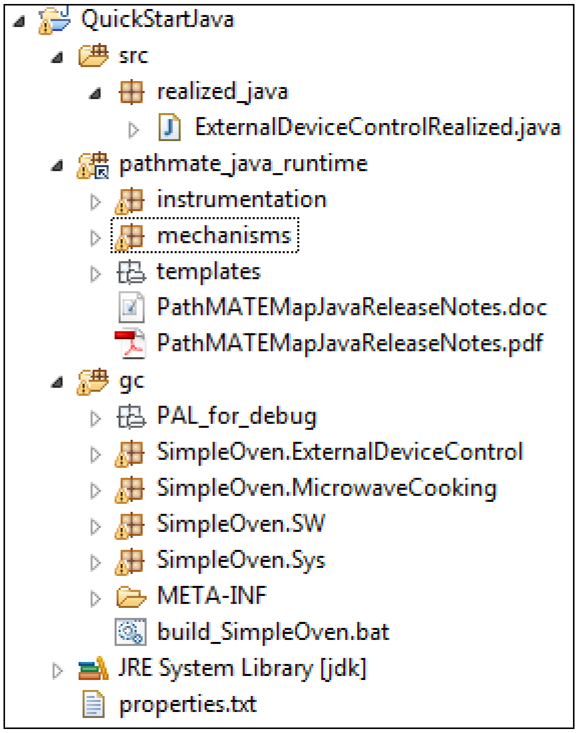
Transformation: Default Java

3. Click **Transform**.
4. Verify no model extraction errors, or other errors are listed.



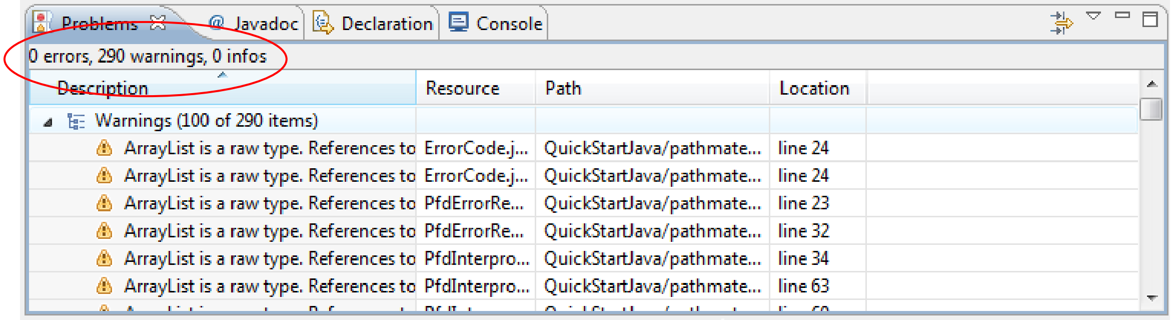
```
PathMATE
Transforming Deployment 'Single Process Java'
Scope System
Project 'platform:/resource/QuickStart/QuickStart SYSTEM MODEL.pathmate'
Tue Nov 13 12:58:15 GMT-05:00 2007

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
```

5. Verify that the *QuickStartJava* project has these contents:



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

6. The java code will build automatically. Click on the Problems view to determine if there are any compilation errors. You will see warnings about ArrayList raw types. These warnings are expected.



0 errors, 290 warnings, 0 infos

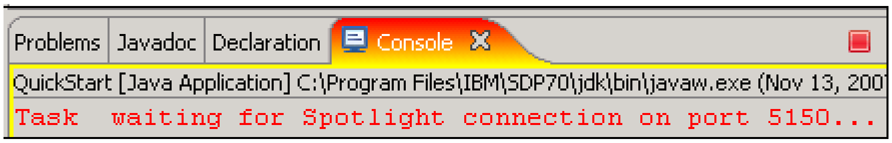
Description	Resource	Path	Location
Warnings (100 of 290 items)			
ArrayList is a raw type. References to	ErrorCode.j...	QuickStartJava/pathmate...	line 24
ArrayList is a raw type. References to	ErrorCode.j...	QuickStartJava/pathmate...	line 24
ArrayList is a raw type. References to	PfdErrorRe...	QuickStartJava/pathmate...	line 23
ArrayList is a raw type. References to	PfdErrorRe...	QuickStartJava/pathmate...	line 32
ArrayList is a raw type. References to	PfdInterpro...	QuickStartJava/pathmate...	line 34
ArrayList is a raw type. References to	PfdInterpro...	QuickStartJava/pathmate...	line 63

Task 2: Ready the System for Debug

PROCEDURE: Create a Debug Configuration and Smoke Test it.

1. Go to **Window / Open Perspective** to switch to the **Debug** perspective.
2. Pull down **Debug…** from the debug toolbar menu ✱▾. Select the **Open Debug Dialog** menu item.
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.

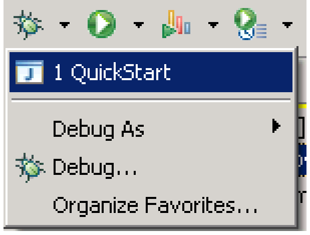
7. Select *QuickStartJava* for the Project if it is not selected already.
8. In the Main Class section, press the **Search…** button. The Select Main Type dialog appears.
9. Select **SimpleOvenApp** from the Matching Types and click **OK**.
10. Click **Debug**. This starts the system. In the Console 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: Exercise Simple Oven with Spotlight

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

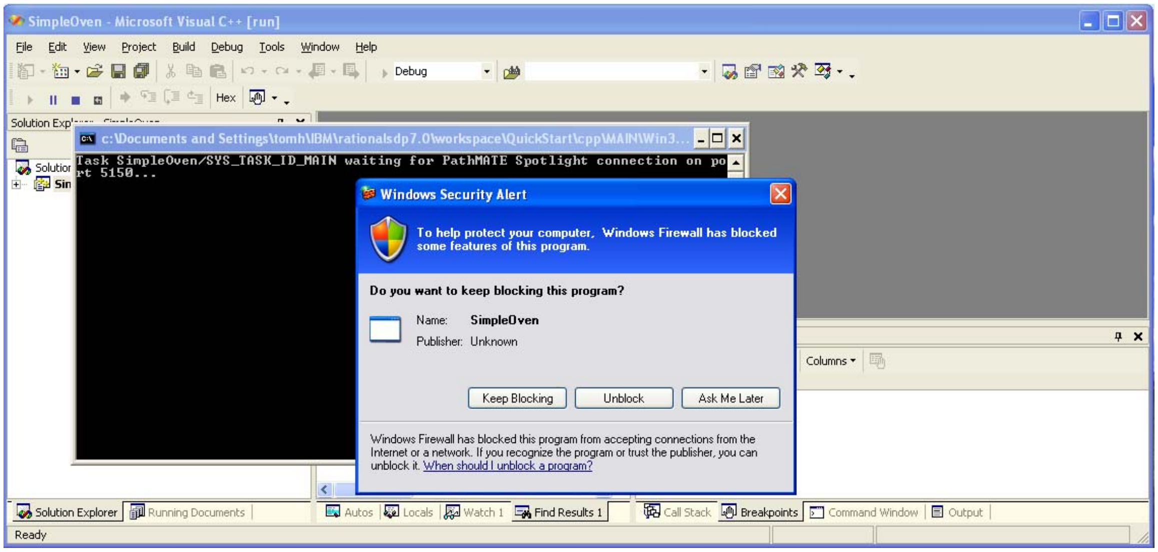
A command window opens to say that the application is running and waiting for a connection to Spotlight.

Please choose the appropriate Procedure below to start the SimpleOven executable, 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

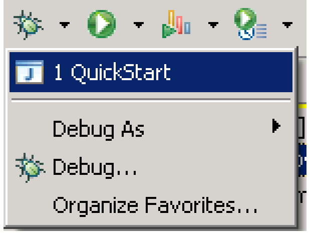
Windows Firewall has blocked this program from accepting connections from the Internet or a network. If you recognize the program or trust the publisher, you can unblock it. When should I unblock a program?

2. If a Windows Security Alert appears, click **Unblock**.

PROCEDURE: Run Java SimpleOven from the Eclipse JDT

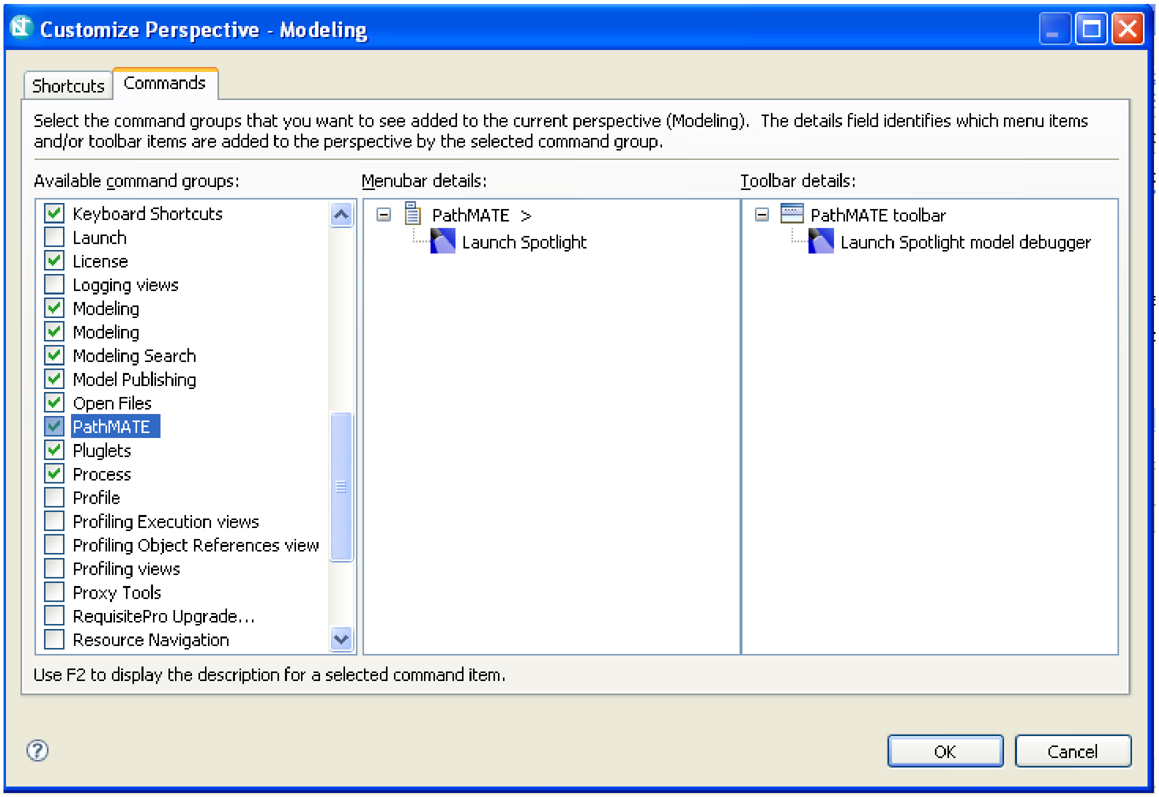
If you built C++, please skip this procedure.

1. Go to the Debug button and select the QuickStart configuration: **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 dialog by clicking on the **Window > Customize Perspective…** option in the main menu. Select the Commands tab. Check the **PathMATE** command group on the Commands tab. Click **OK**.



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:

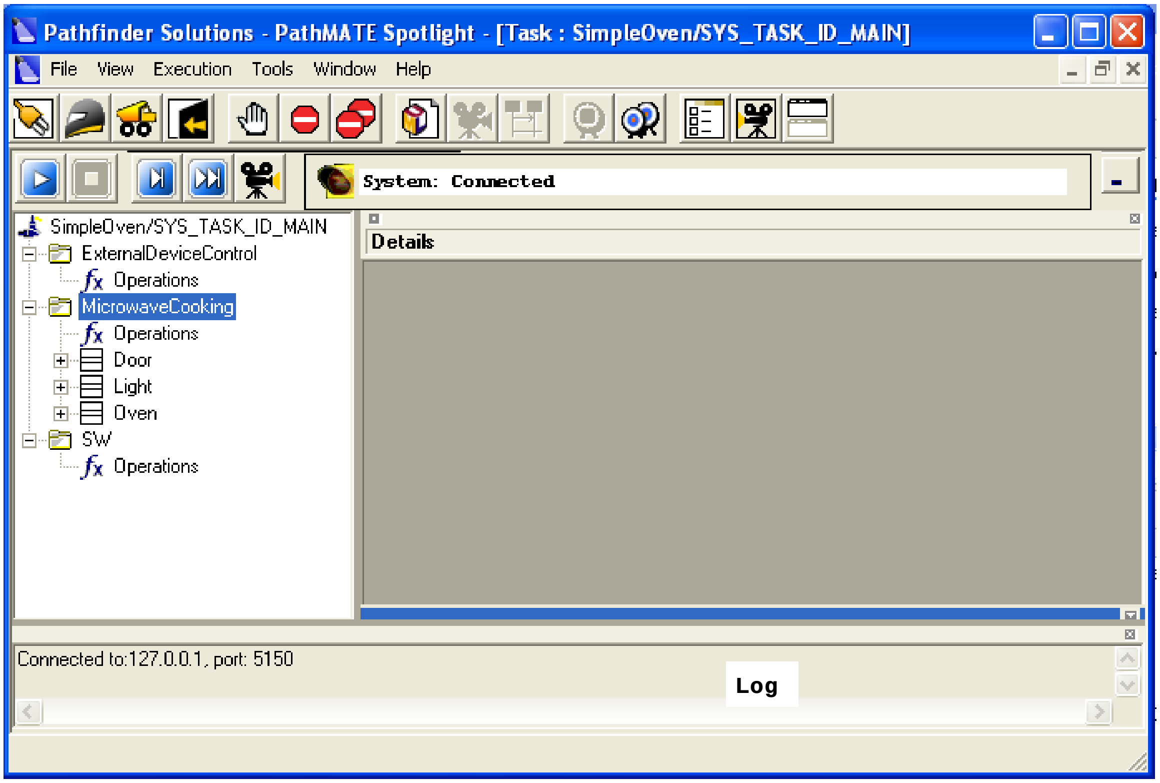
- PathMATE toolbar
  - Launch Spotlight model debugger

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

2. 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:

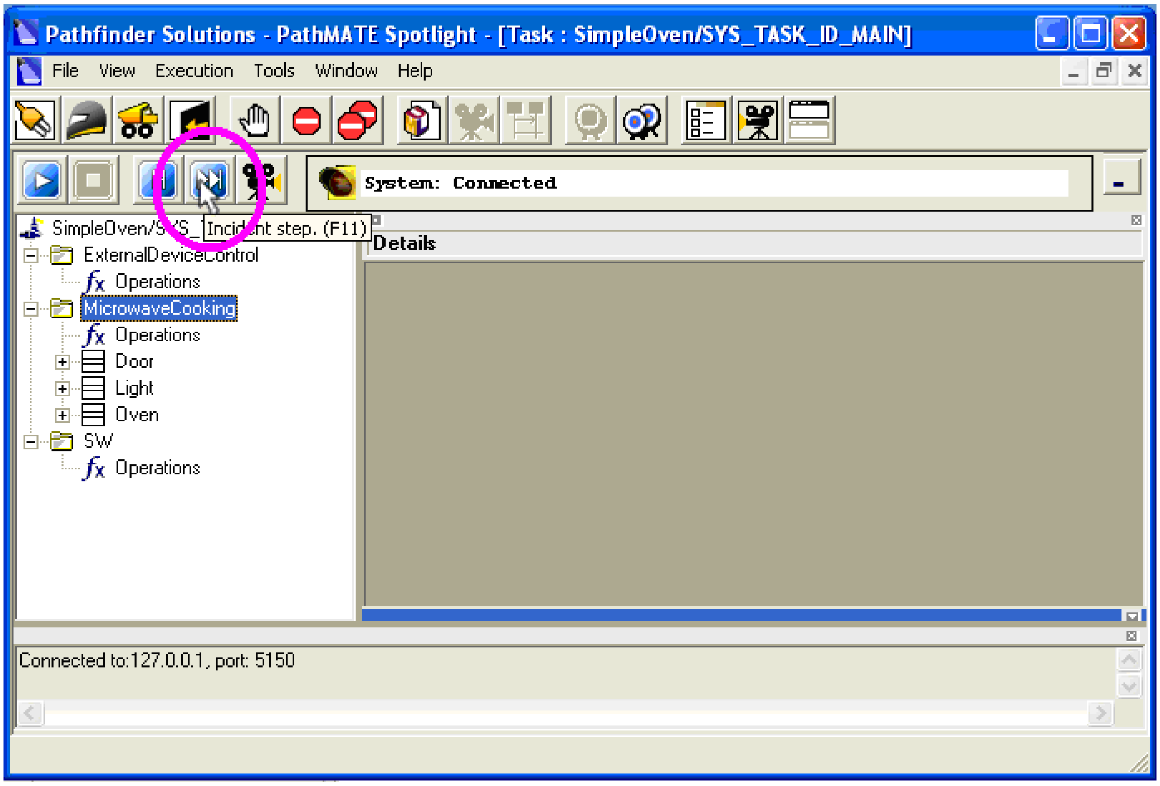


System: Connected

Connected to:127.0.0.1, port: 5150

PROCEDURE: Use Spotlight to Initialize the SimpleOven System

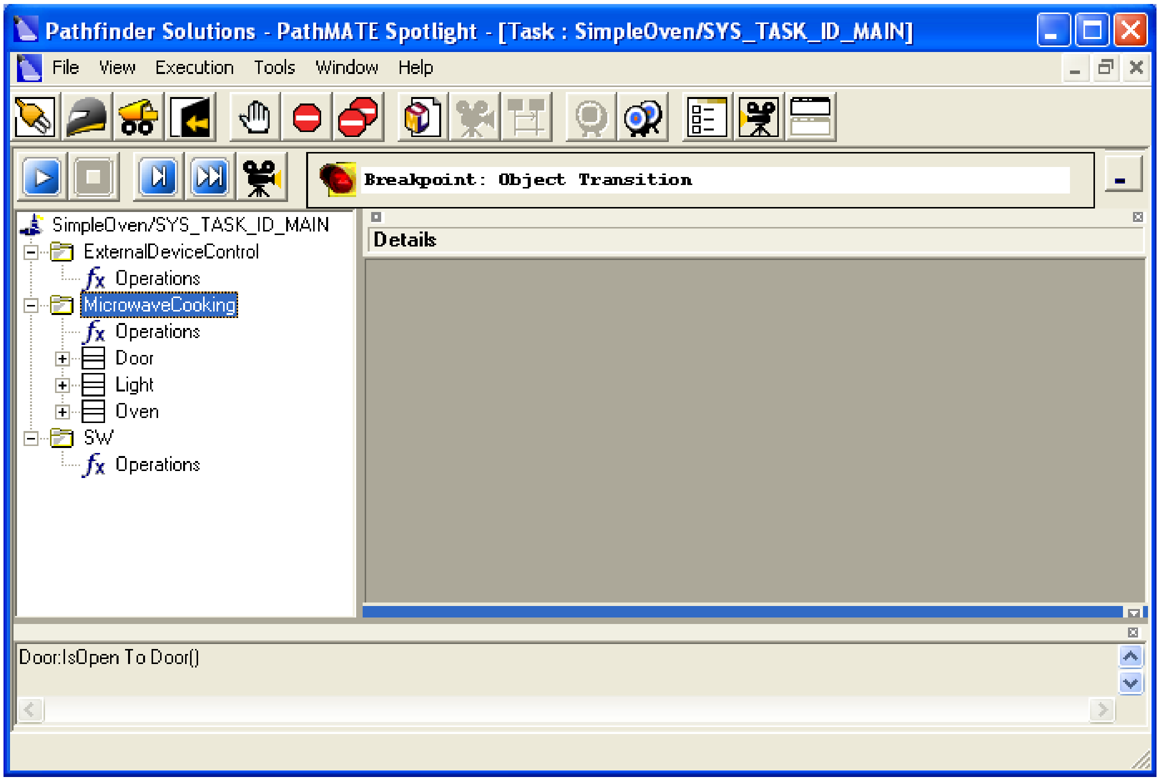
1. Locate the Incident Step button :



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:



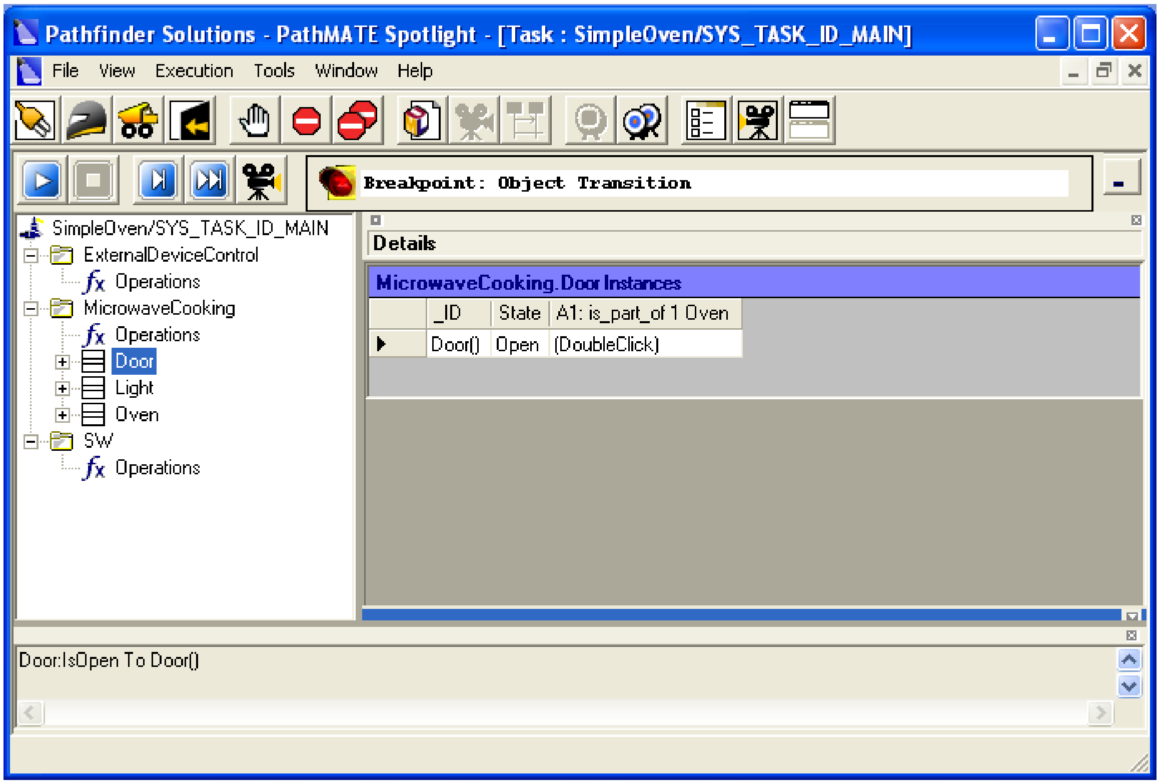
Breakpoint: Object Transition

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*).



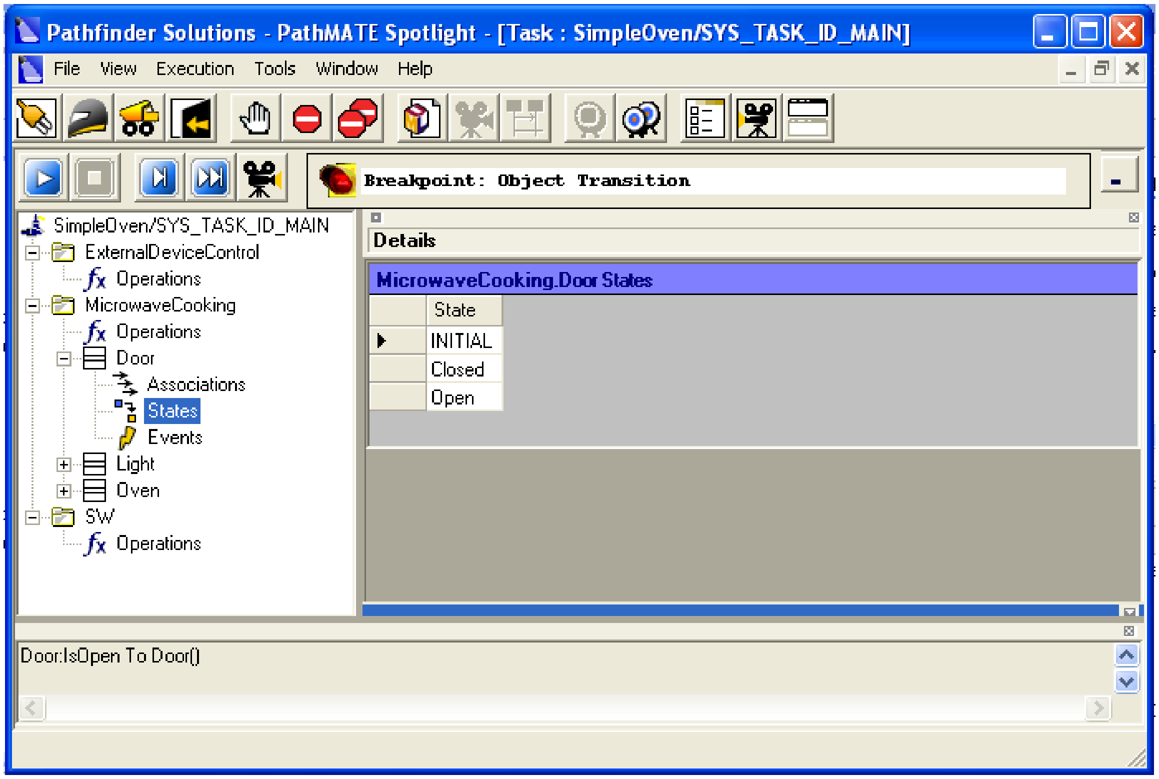
Breakpoint: Object Transition

MicrowaveCooking.Door Instances

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

Door:IsOpen To Door()

2. Expand the **Door** class in the browser and click on **States** to review the possible run-time states of this class. Note the Door instance 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.



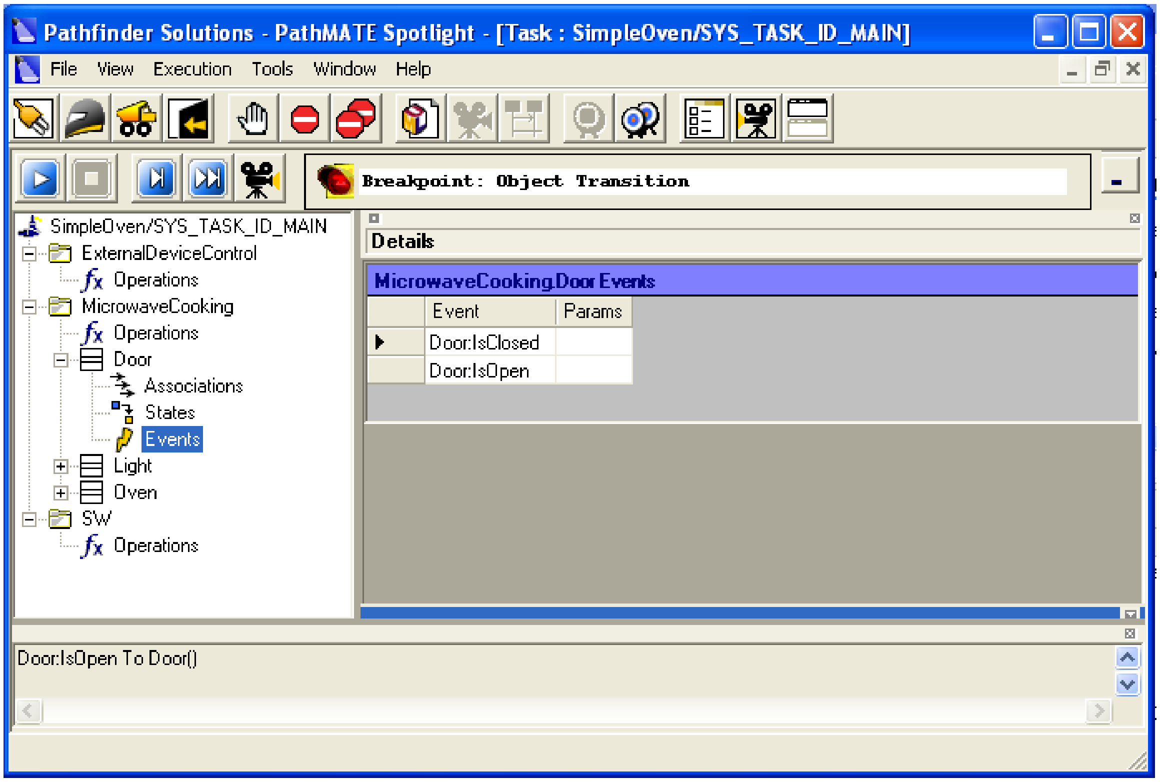
Breakpoint: Object Transition

MicrowaveCooking.Door States

	State
▶	INITIAL
	Closed
	Open

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

3. 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

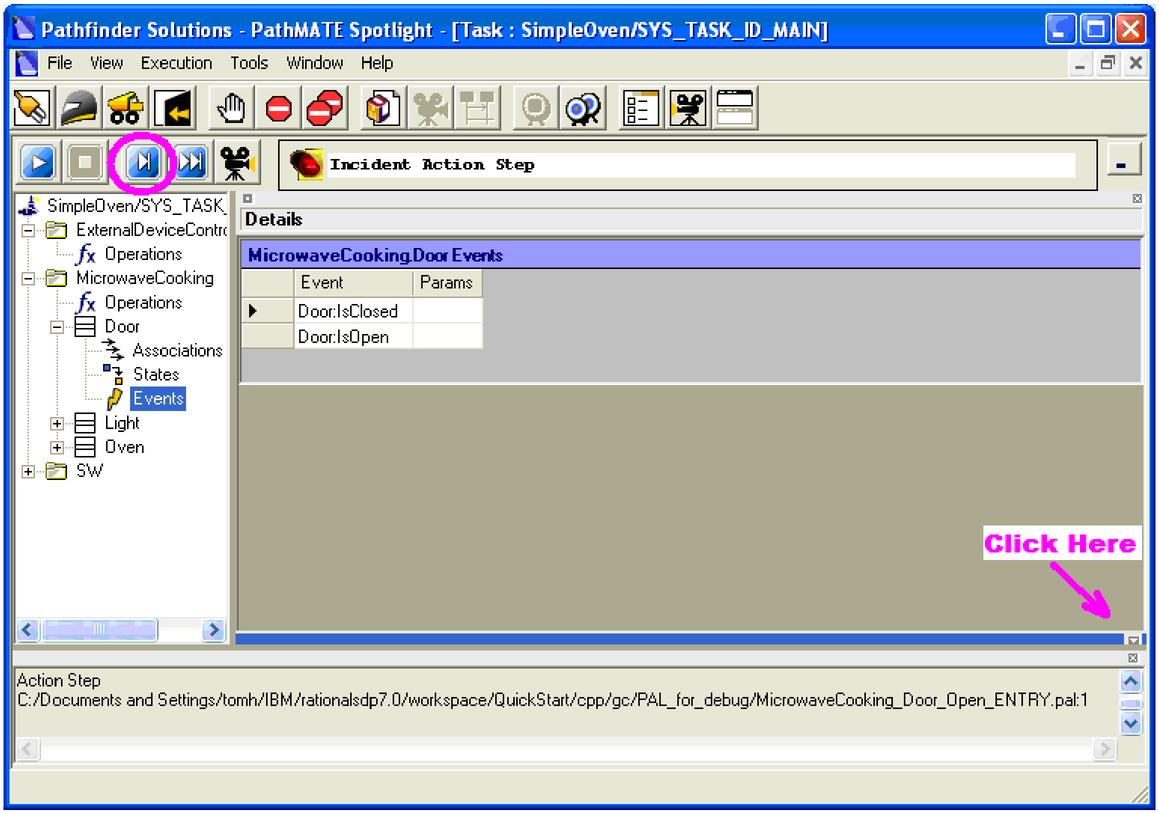
MicrowaveCooking.Door Events

Event	Params
Door:IsClosed	
Door:IsOpen	

Door:IsOpen To Door()

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

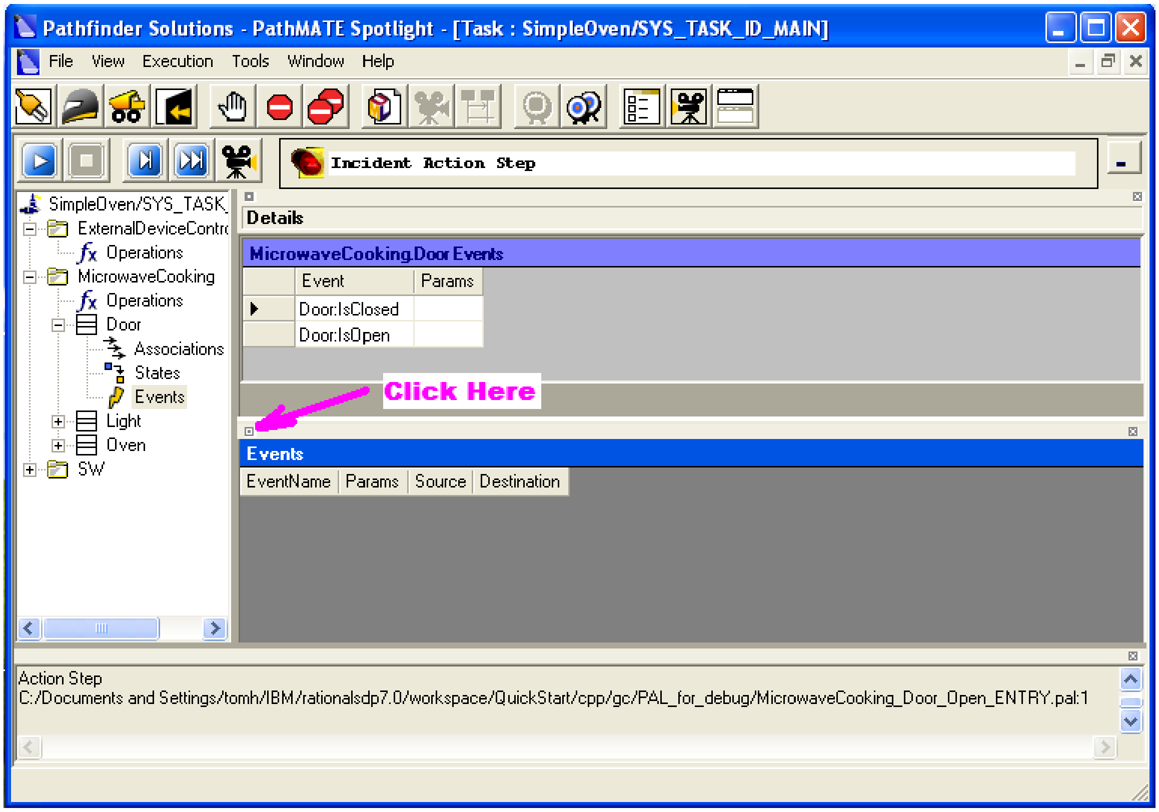
1. Locate the Action Step button . It is the button to the immediate left of the Incident Step button.
2. Click the **Action Step button** .
3. Open the State Window. The figure below shows where to click in the lower right to open and close the action/event window:



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

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



Click Here

MicrowaveCooking.Door Events

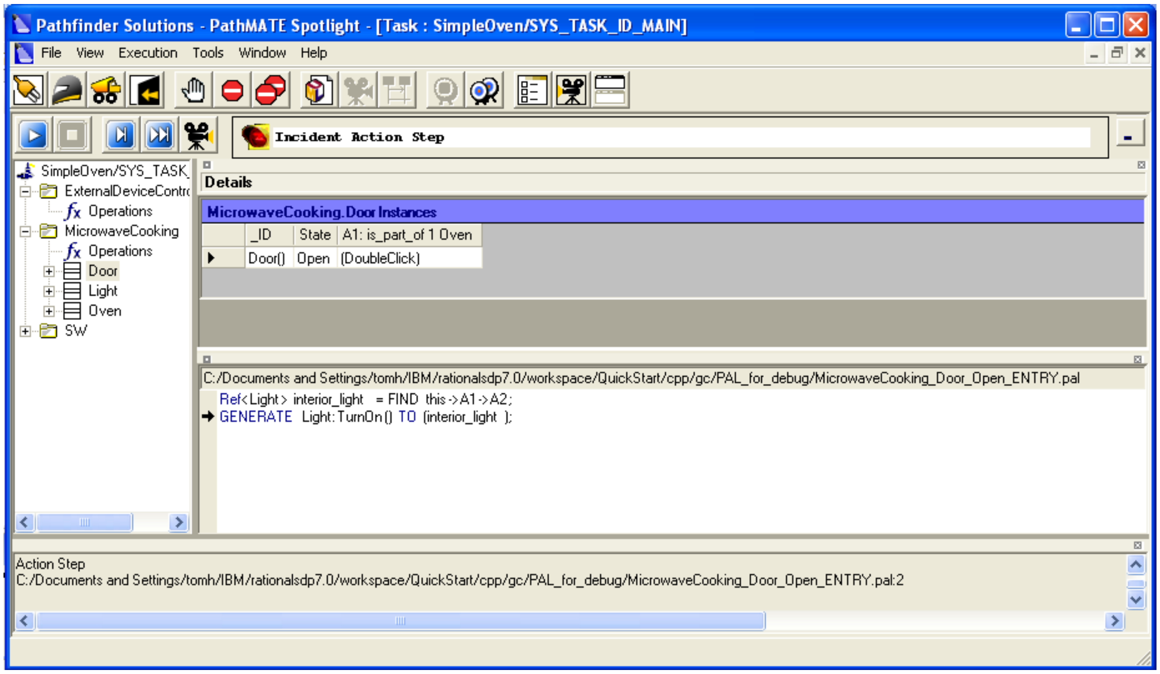
Event	Params
Door:IsClosed	
Door:IsOpen	

Events

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

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



Incident Action Step

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 );
```

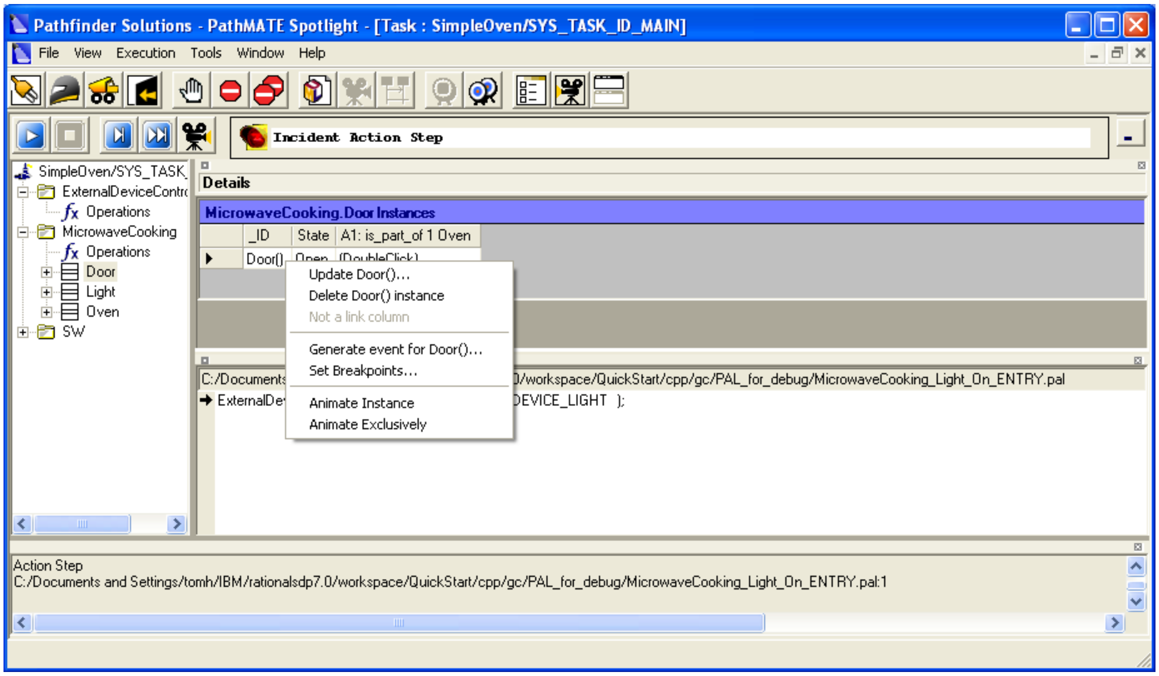
```
Action Step
C:/Documents and Settings/tomh/IBM/rationalsdp7.0/workspace/QuickStart/cpp/gc/PAL_for_debug/MicrowaveCooking_Door_Open_ENTRY.pal:2
```

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

(Please 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:



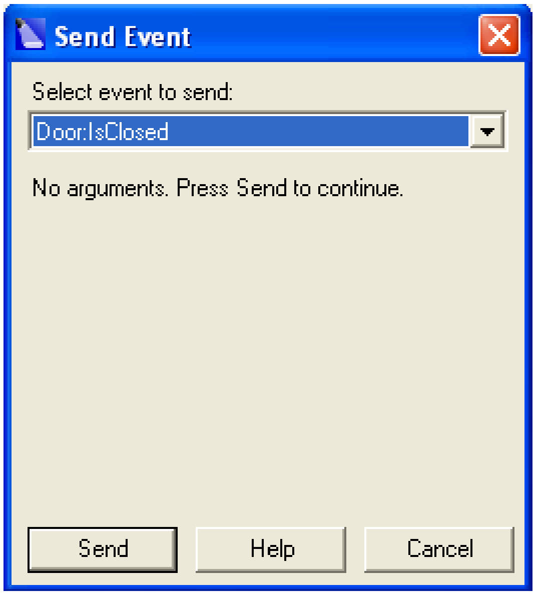
MicrowaveCooking.Door Instances

_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
```

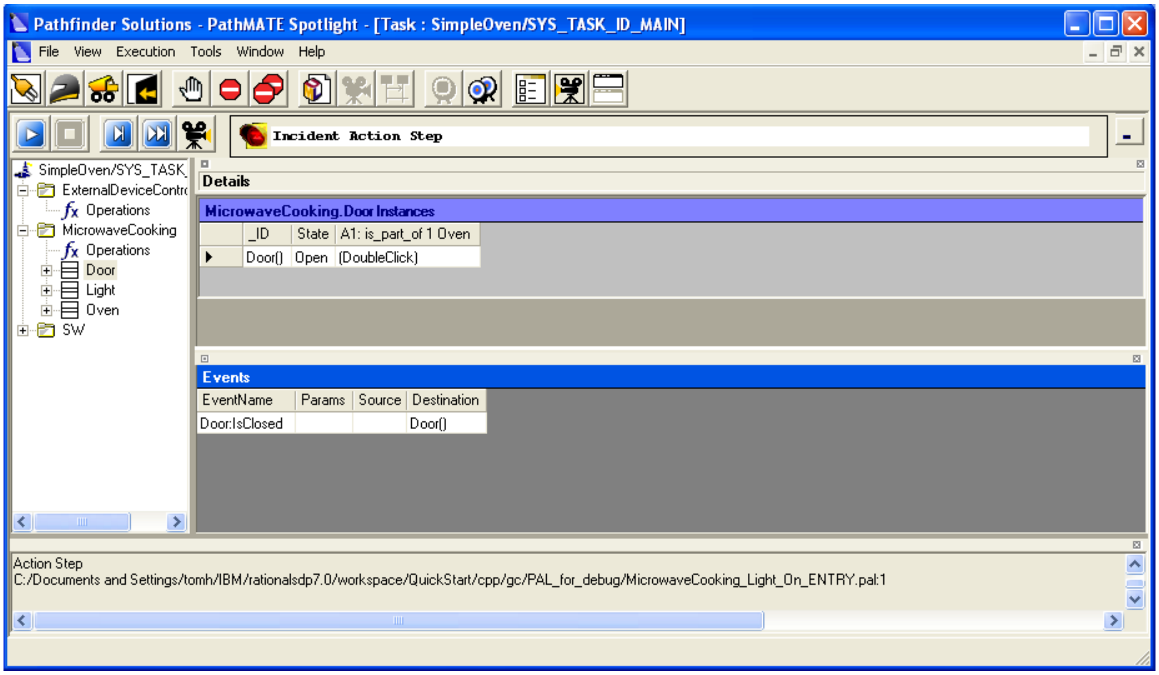
2. Select **Generate event for Door()** in the pop-up menu. The Send Event dialog opens. Select **Door:IsClosed** in the drop-down list.



No arguments. Press Send to continue.

3. Click **Send**.

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


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

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

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.

Task 2: State Machine Animation

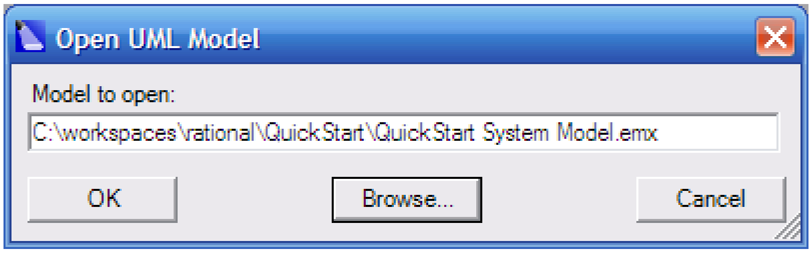
PROCEDURE: Restart Spotlight Session

1. Click the **Stop all Tasks** button .
2. Re-launch the application. A command window opens to say that the application is running and waiting for a connection to Spotlight.
3. Click the **Connect button** to connect Spotlight to the target application. Look for System: Connected on the Spotlight status bar.

PROCEDURE: Connect Spotlight to SimpleOven Model in RSM

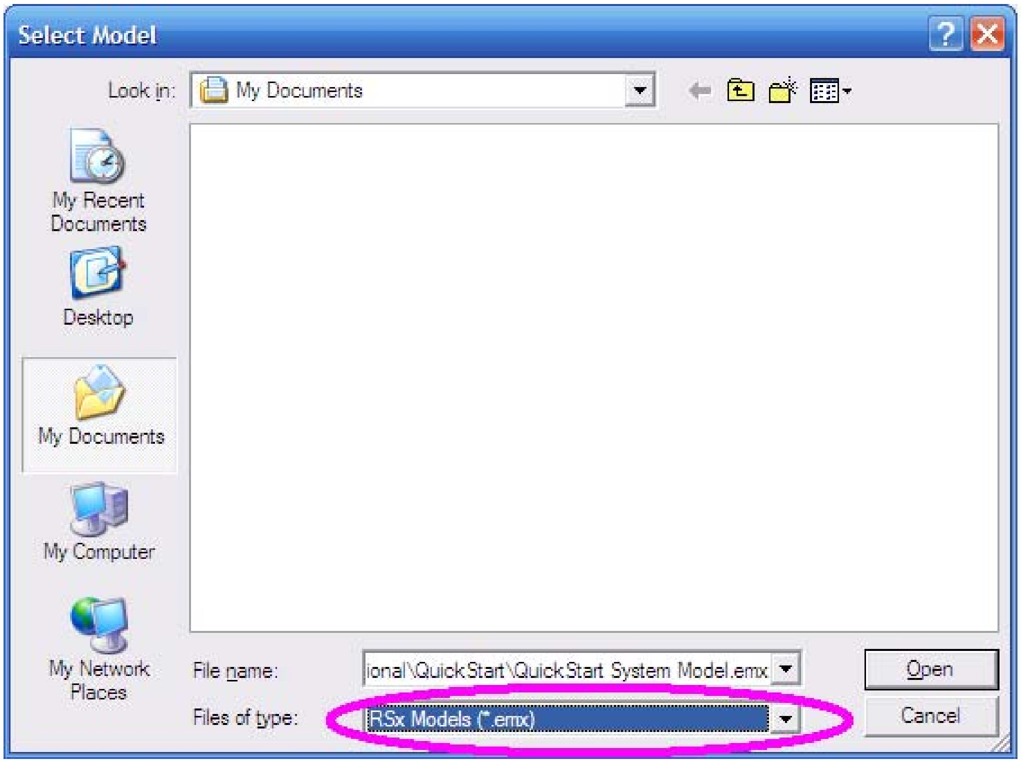
These steps establish a run-time connection between Spotlight and the Rational Software Modeler process so that Spotlight can drive state machine animation and sequence chart capture:

1. Initiate the open UML model dialog by clicking the Spotlight **UML button** on the top row of buttons. If your system model file model appears in the Model to open field, just click **OK** and go on to the next procedure. Otherwise, Click **Browse**.



Model to open: C:\workspaces\rational\QuickStart\QuickStart System Model.emx

2. Under Files of type: select **RSx Models (*.emx)**.

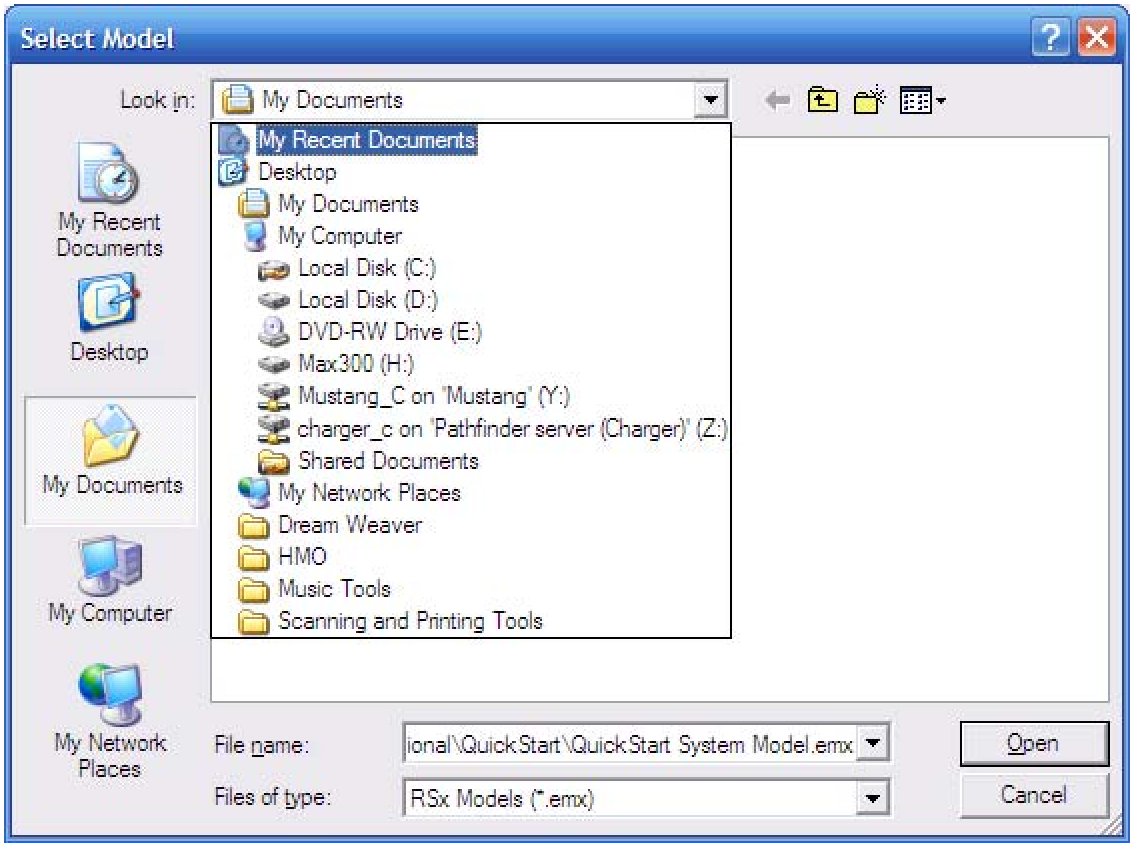


Look in: My Documents

File name: ional\QuickStart\QuickStart System Model.emx

Files of type: RSx Models (*.emx)

3. If you need to navigate to a different directory root, use the **Look in:** drop down, rather than the parent directory button .

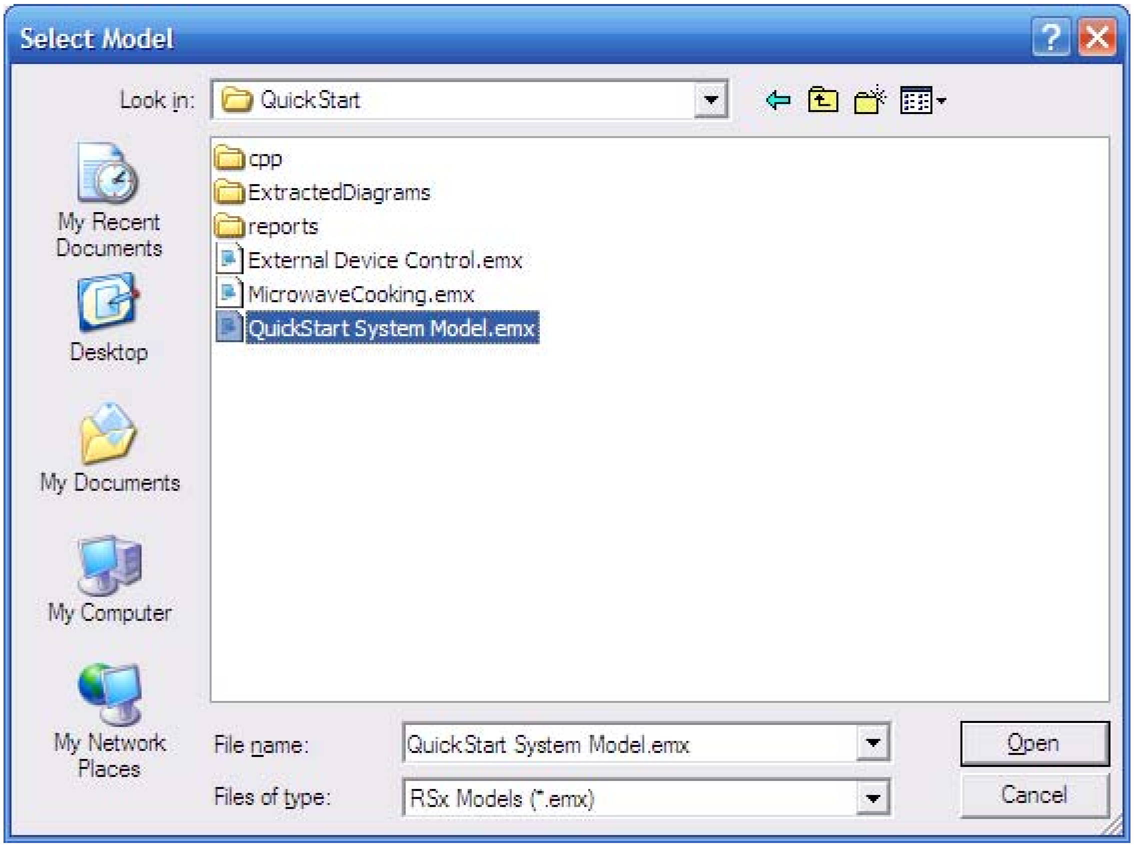


Look in: My Documents

File name: ional\QuickStart\QuickStart System Model.emx

Files of type: RSx Models (*.emx)

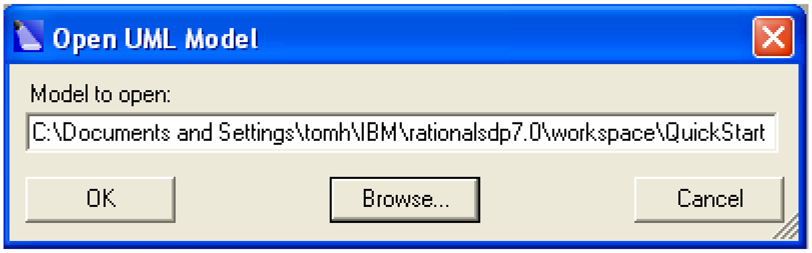
4. You may need to locate your Eclipse workspace by selecting the main menu File-Switch Workspace command in Eclipse, copying the location, and hitting Cancel.
5. Browse to the QuickStart directory in your Eclipse workspace and select **QuickStart System Model.emx**. Click **Open**.



File name: QuickStart System Model.emx

Files of type: RSx Models (*.emx)

6. Your selection willl appear in the Open UML Model dialog:



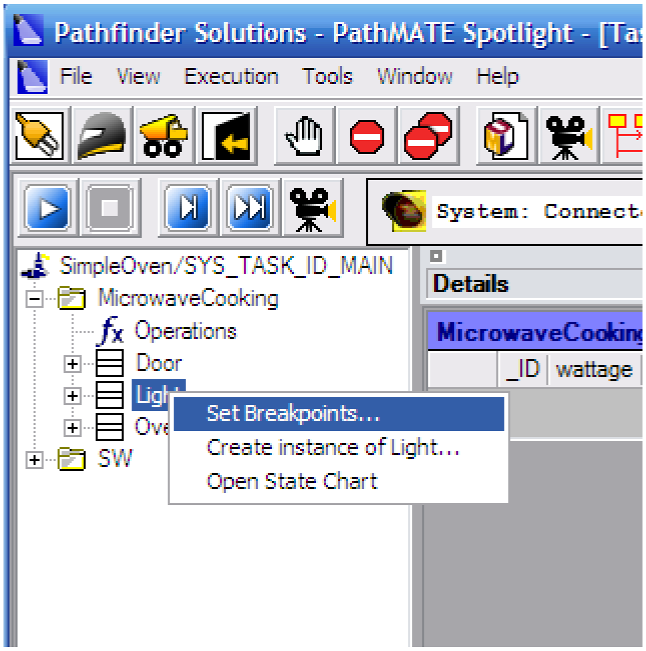
Model to open: C:\Documents and Settings\tomh\IBM\rationalsdp7.0\workspace\QuickStart

7. Click **OK**, and Spotlight will establish a connection to this model within RSM.

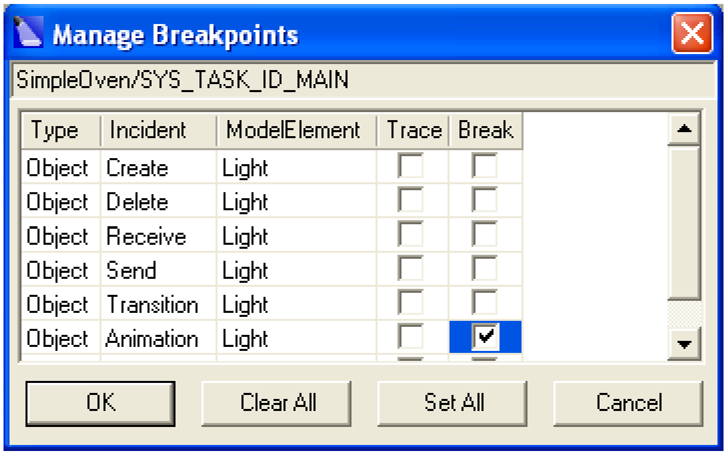
PROCEDURE: Select Light State Machine for Animation

These steps identify that the Light class state machine is to be animated.

1. **Expand** the MicrowaveCooking domain in the Spotlight browser
2. Select the **Light** class, right click, and pick **Set Breakpoints…**



3. In the Manage Breakpoints dialog check the **Animation/Break** box and click **OK**:



SimpleOven/SYS_TASK_ID_MAIN

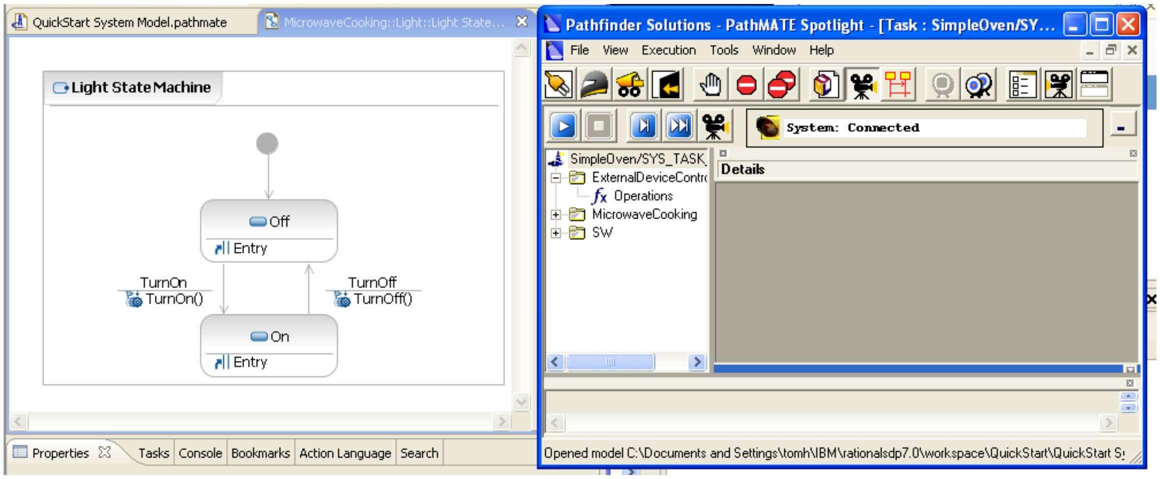
Type	Incident	ModelElement	Trace	Break
Object	Create	Light	□	□
Object	Delete	Light	□	□
Object	Receive	Light	□	□
Object	Send	Light	□	□
Object	Transition	Light	□	□
Object	Animation	Light	□	☑

4. Locate the Animate state button on the top row of buttons. Note there is another similar button on the second row, to the right of the Go button:

   - Use this one:
   - Do NOT use the Animate Action Steps button on the second row:

5. Activate state animation from this point in execution forward by clicking the **Animate state button**:

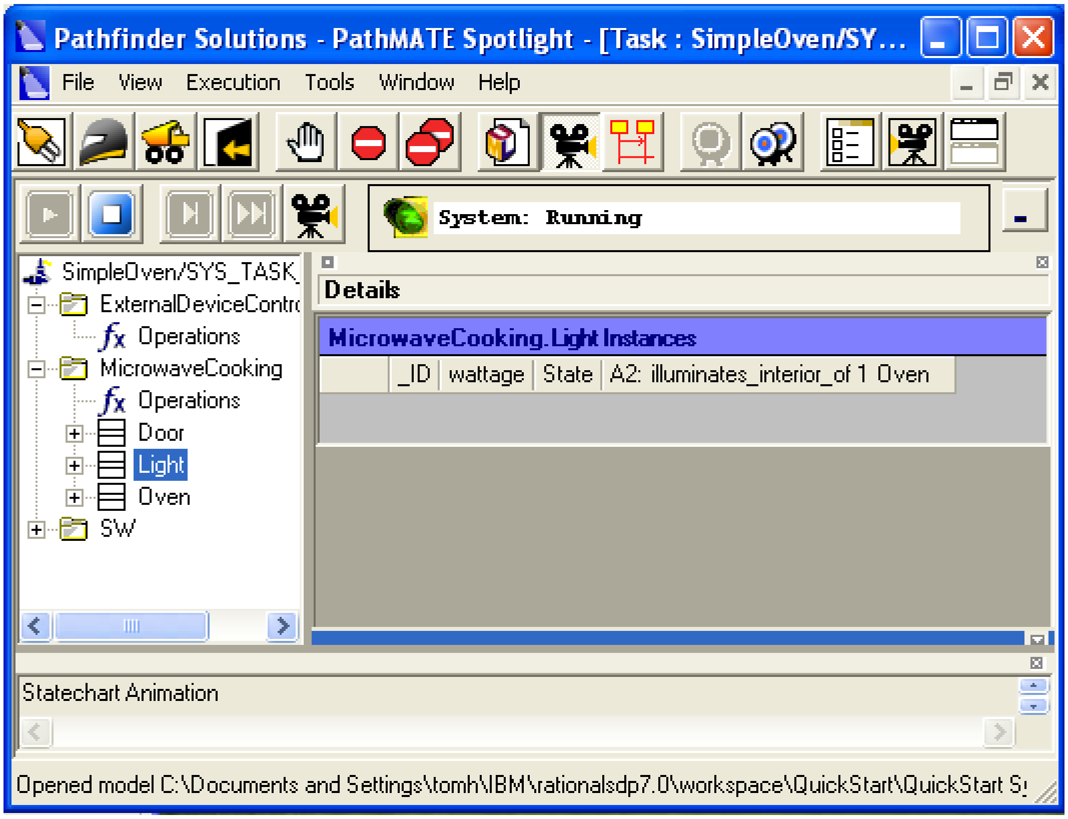
6. Position your Spotlight window over/around your RSM window so the RSM edit pane is visible



System: Connected

Opened model C:\Documents and Settings\tomh\IBM\rationalsdp7.0\workspace\QuickStart\QuickStart S

7. Press the Spotlight **Go button** to start SimpleOven execution. The System is now running:



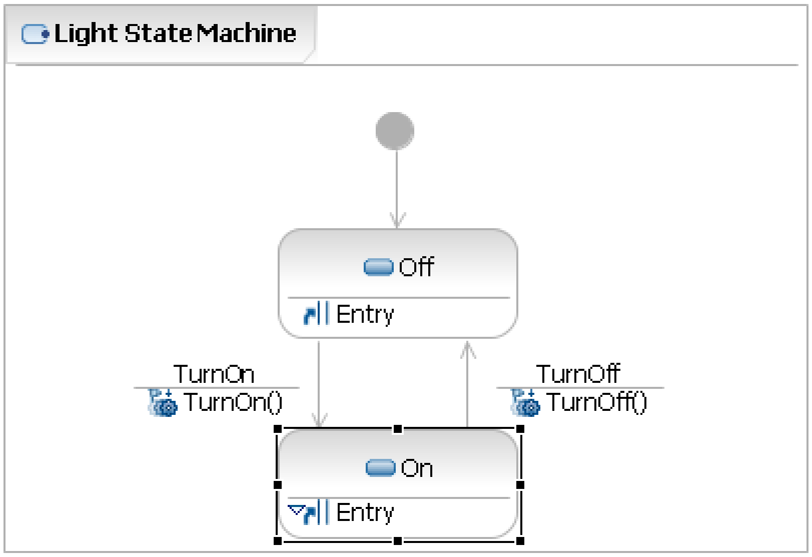
System: Running

MicrowaveCooking.Light Instances

	_ID	wattage	State	A2: illuminates_interior_of 1 Oven

Opened model C:\Documents and Settings\tomh\IBM\rationalsdp7.0\workspace\QuickStart\QuickStart S

In RSM, the Light state machine diagram the current state is highlighted. As expected, this is the On State because of the DoorIsOpen Signal generated in the MicrowaveCooking *Domain* Initialize() code:



8. You may choose to continue execution by sending Door:IsOpen and Door:IsClosed events as specified in *Task 1: Exercise Simple* Oven with Spotlight. As the Light object state changes within the SimpleOven executable the selected state will change on the Light state machine diagram.

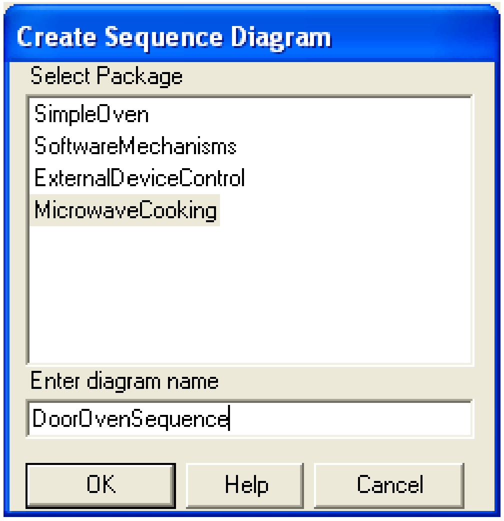
Task 3: Sequence Chart Generation

PROCEDURE: Restart Spotlight Session

1. Be sure the SimpleOven program from previous Tasks has been stopped and Spotlight is clear of all tasks connections by clicking the **Stop all Tasks** button .
2. Re-launch the application. A command window opens to say that the application is running and waiting for a connection to Spotlight.
3. Click the **Connect button** at the left end of the toolbar to connect Spotlight to the target application. Look for "System: Connected" on the Spotlight status bar.
4. Initiate the Open UML model dialog with the Spotlight **UML button** .
5. Using the same procedure used for Animation above, open SimpleOven.emx. Note that the previously opened model is the default entry in the Open UML Model dialog so this time you shouldn't have to go through all the steps.

PROCEDURE: Create a New Sequence Diagram

1. Create a new sequence diagram with the **Create New Sequence Diagram button**.
2. Select the package the diagram will go in (we selected the MicrowaveCooking domain because we will be tracing MicrowaveCooking classes), enter DoorOvenSequence in the diagram name field and click **OK**:



Create Sequence Diagram

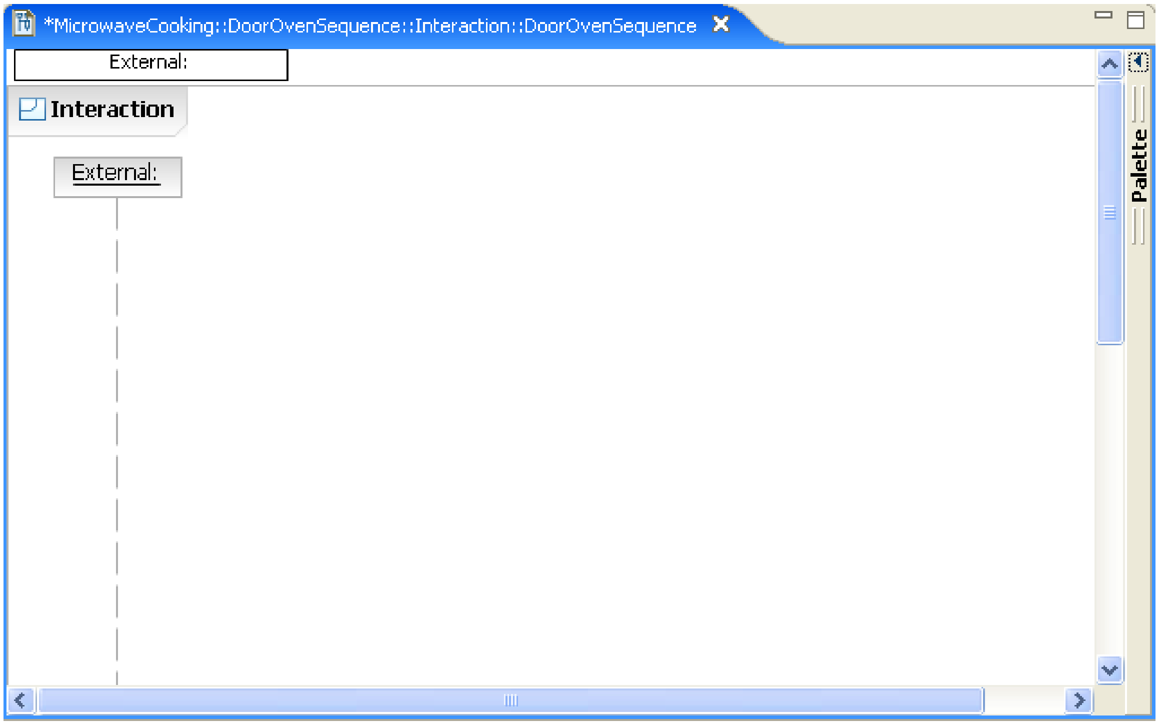
Select Package

SimpleOven
SoftwareMechanisms
ExternalDeviceControl
MicrowaveCooking

Enter diagram name

DoorOvenSequence

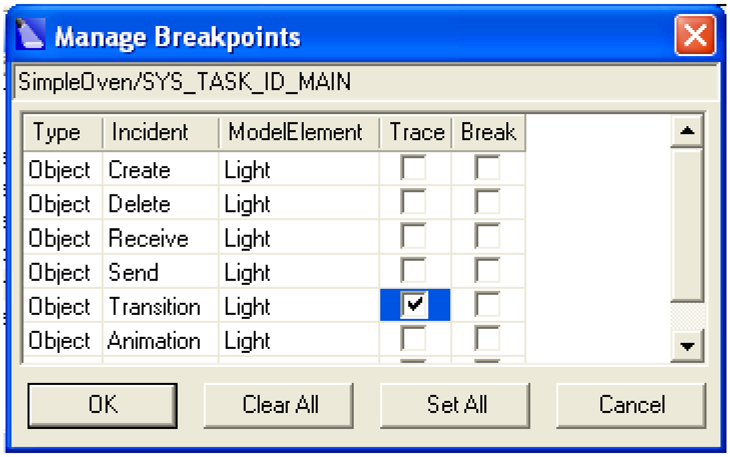
Look in RSM and see the new empty interaction diagram:



*MicrowaveCooking::DoorOvenSequence::Interaction::DoorOvenSequence

PROCEDURE: Set Transition Traces for All Classes to be Included

1. Select Light class in the MicrowaveCooking domain in the Spotlight browser
2. Right click and pick **Set Breakpoints…**
3. Check the **Transition / Trace** box and click **OK**:



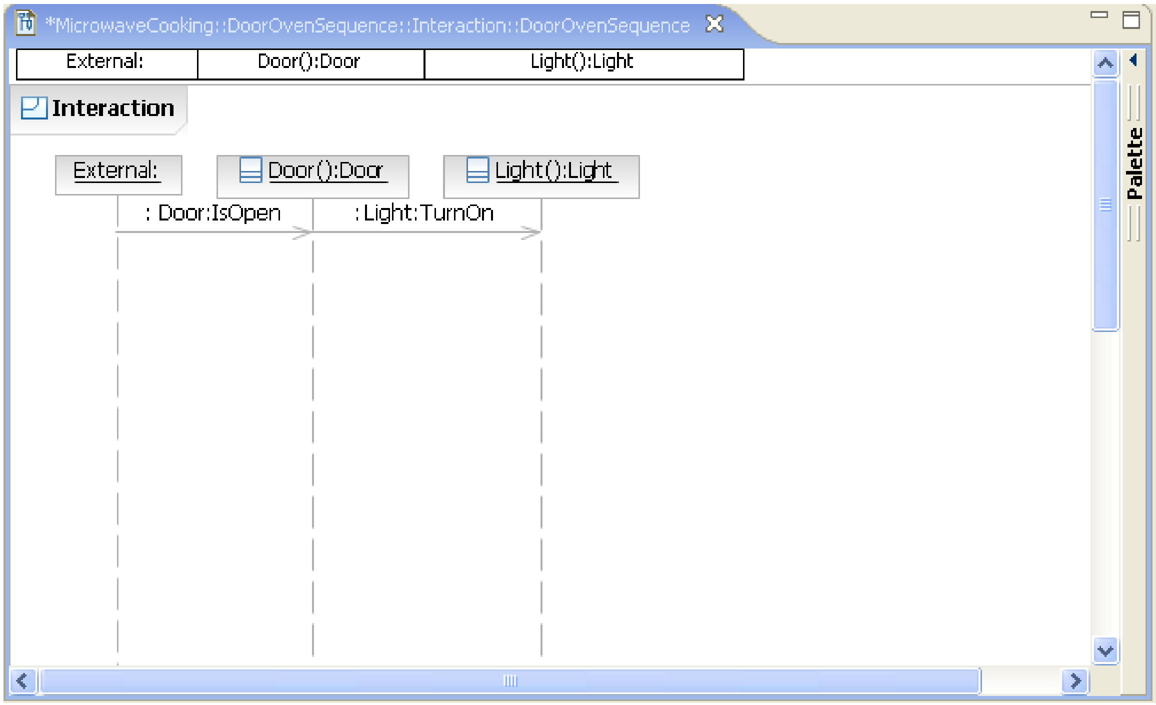
SimpleOven/SYS_TASK_ID_MAIN

Type	Incident	ModelElement	Trace	Break
Object	Create	Light	☐	☐
Object	Delete	Light	☐	☐
Object	Receive	Light	☐	☐
Object	Send	Light	☐	☐
Object	Transition	Light	☑	☐
Object	Animation	Light	☐	☐

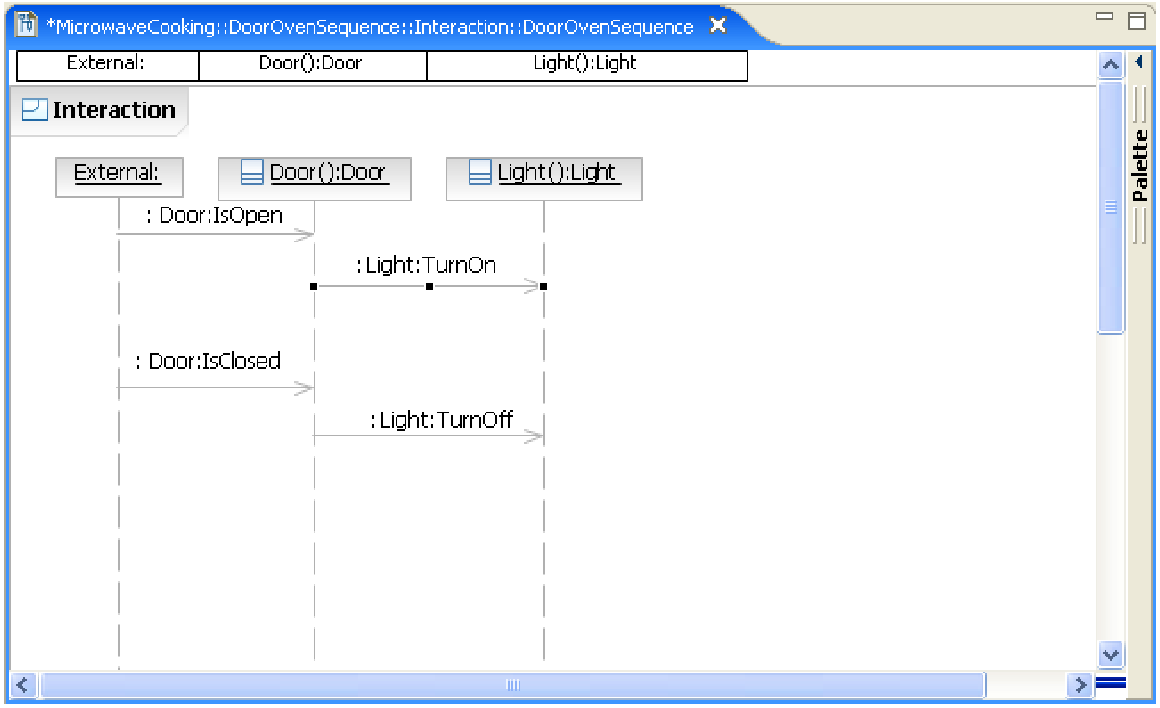
4. Select the Door class and set a **Transition / Trace** break for it as well.

PROCEDURE: Run SimpleOven and Capture the Sequence

1. Press the Spotlight **Go button** ▶ to start SimpleOven execution. The MicrowaveCooking Domain Initialize code generates a DoorIsOpen signal. The effect is immediately drawn in the interaction diagram:



2. Send a Door:IsClosed event using the techniques you learned earlier in the tutorial.
3. The interaction diagram captures the Door closing, with the resulting Light activity. You can reposition and make other changes to your diagram using all the tools available in RSM.



4. Be sure to save the MicrowaveCooking.emx model to preserve the generated sequence.

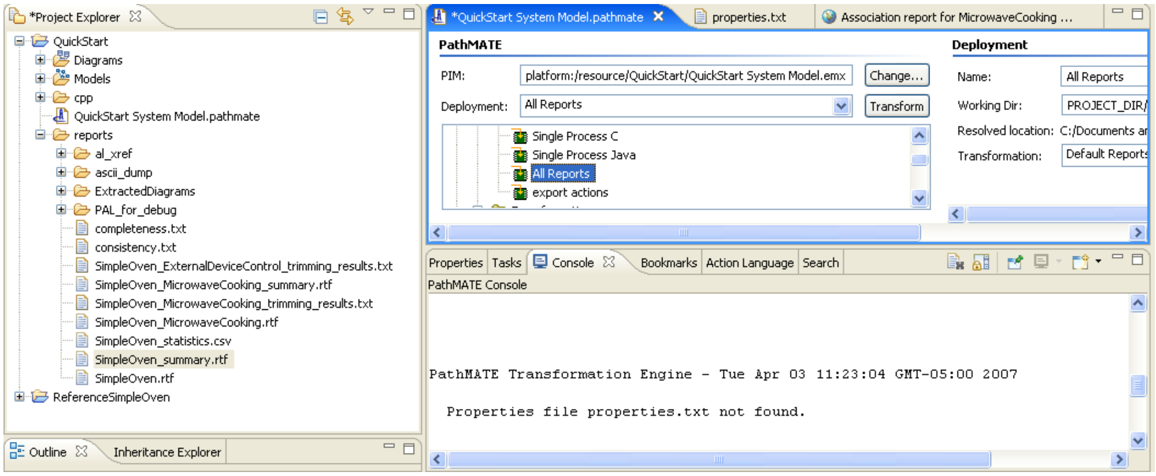
Generate System Documentation

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.)



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

Deployment: All Reports

Name: All Reports

Working Dir: PROJECT_DIR/

Resolved location: C:/Documents ar

Transformation: Default Reports

```
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
IBM	International Business Machines
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
RSA	Rational Software Architect
RSD	Rational Systems Developer
RSM	Rational Software Modeler
RSx	Any variant of RSM, RSD, or RSA
UML	Unified Modeling Language