The xtUML Editor Quick Start Tour: User Interface

Welcome to the xtUML Editor. This feature-rich Editor contains numerous windows, menus and selections which can be overwhelming to a new user. In this tour, the most common Editor activities will be highlighted, and much of the mystery removed. By the end of the tour, you will be ready to start your first xtUML project and the entry of your first diagram. Enjoy your tour!

A Few Terms Before We Get Started

The xtUML Editor is comprised of a set of plug-ins integrated into the Eclipse Development Environment. For embedded developers that have not used Eclipse before, some terms and concepts may be unfamiliar. Let's take a look at these terms before moving on.

- A **project** is the fundamental unit of organization in Eclipse. Most operations require the context of a project.
- A **workspace** holds one or more related projects. (A dialog will come up at the launch of The xtUML Editor, allowing the user to select the workspace.)
- The **workbench** is the outermost container of an Eclipse-based user interface, which holds all the tools being used at a given time.
- An individual tool panel is called a **view**.
- **Editors** let you manipulate a resource directly, for example a source file, or model elements, such as attributes, activities, and descriptions.
- A **perspective** contains a group of related views and editors. You can switch between perspectives to quickly access various task-centered groupings of functionality within the context of a single project or workspace.

The xtUML Editor builds upon the Eclipse fundamentals to provide projects, perspectives, views, and editors that let you build xtUML models and translate them into code.

The xtUML Modeling Perspective

The xtUML Modeling Perspective is the default perspective, shown upon opening the tool for the first time in a particular workspace.

When you view or enter a xtUML model, you do so from within the xtUML Modeling perspective. As described above, a perspective is just a task-centered collection of views and editors. The xtUML Modeling perspective provides views and editors that center around modeling in xtUML.

As we go through the Quick Tour, you will be introduced to each of the views and editors that comprise the xtUML Modeling perspective. If you are not in this perspective, select **Window > Open Perspective > xtUML Modeling**.

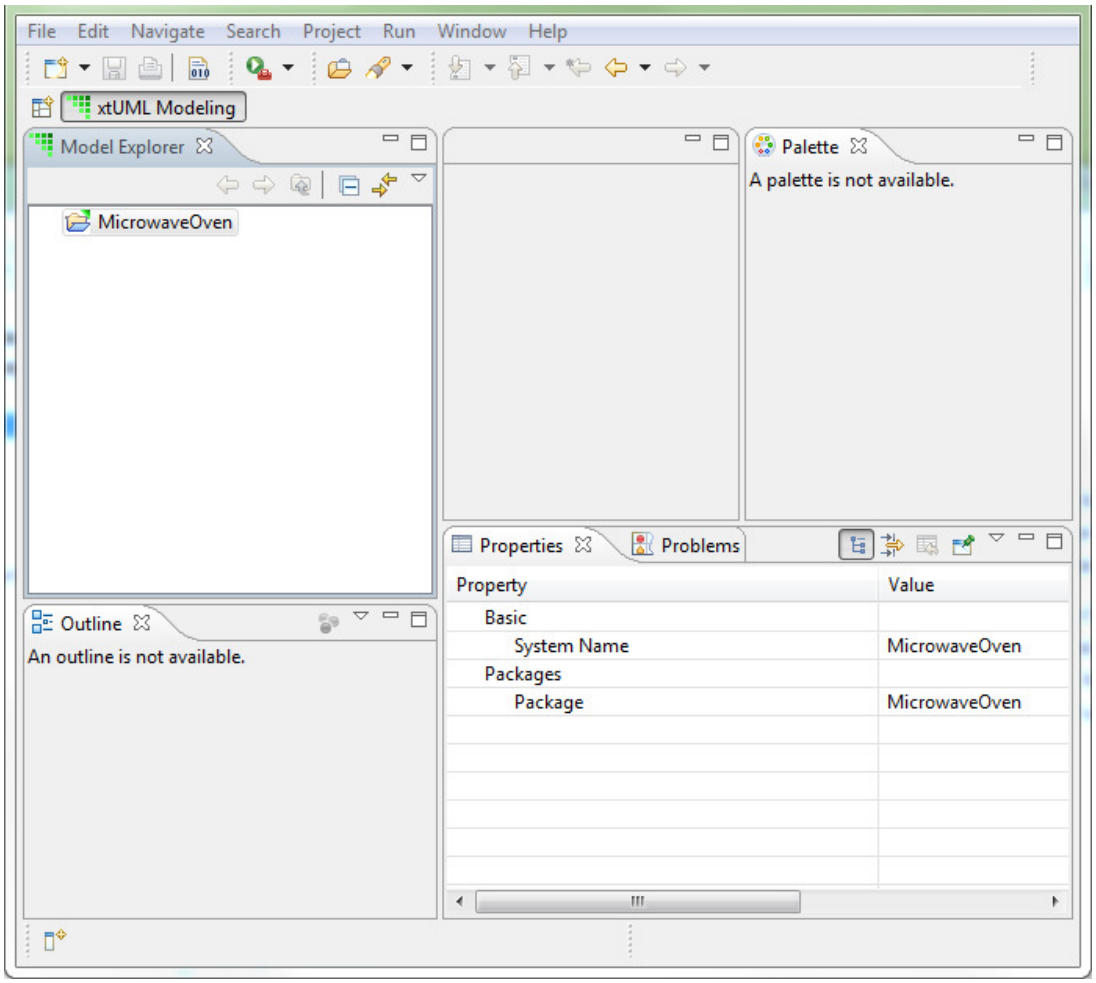
In this section of the Quick Start Guide, we take a look at the user interface through the sample model. This part of the tour requires the MicrowaveOven sample project to exist in the workspace. If you do not see the MicrowaveOven project select **Window > Quick Start > Example Application – Microwave Oven**. This will create the desired example model in the workspace.

Sample Models

The sample model for the microwave oven, though simplified for the tour, implements the operational behavior of a typical microwave oven. You open a door, put something inside to be heated, close the door, and enter the duration of time you want the oven to cook. When the oven is finished cooking, a beeper sounds. If the door is opened during cooking, the oven interrupts its operation.

Reviewing the Sample UML Model

Before we look at the sample model, make sure you are in the *xtUML Modeling* perspective. This is usually the default when the tool is brought up and can be seen from the tab just above the Model Explorer view shown in the following image. If you are not in this perspective, select **Window > Open Perspective > xtUML Modeling**.



For this tour, The xtUML Editor creates a workspace containing an xtUML project and model for the MicrowaveOven. When you change to the xtUML Modeling perspective it will show the `MicrowaveOven` project folder similar to the image above.

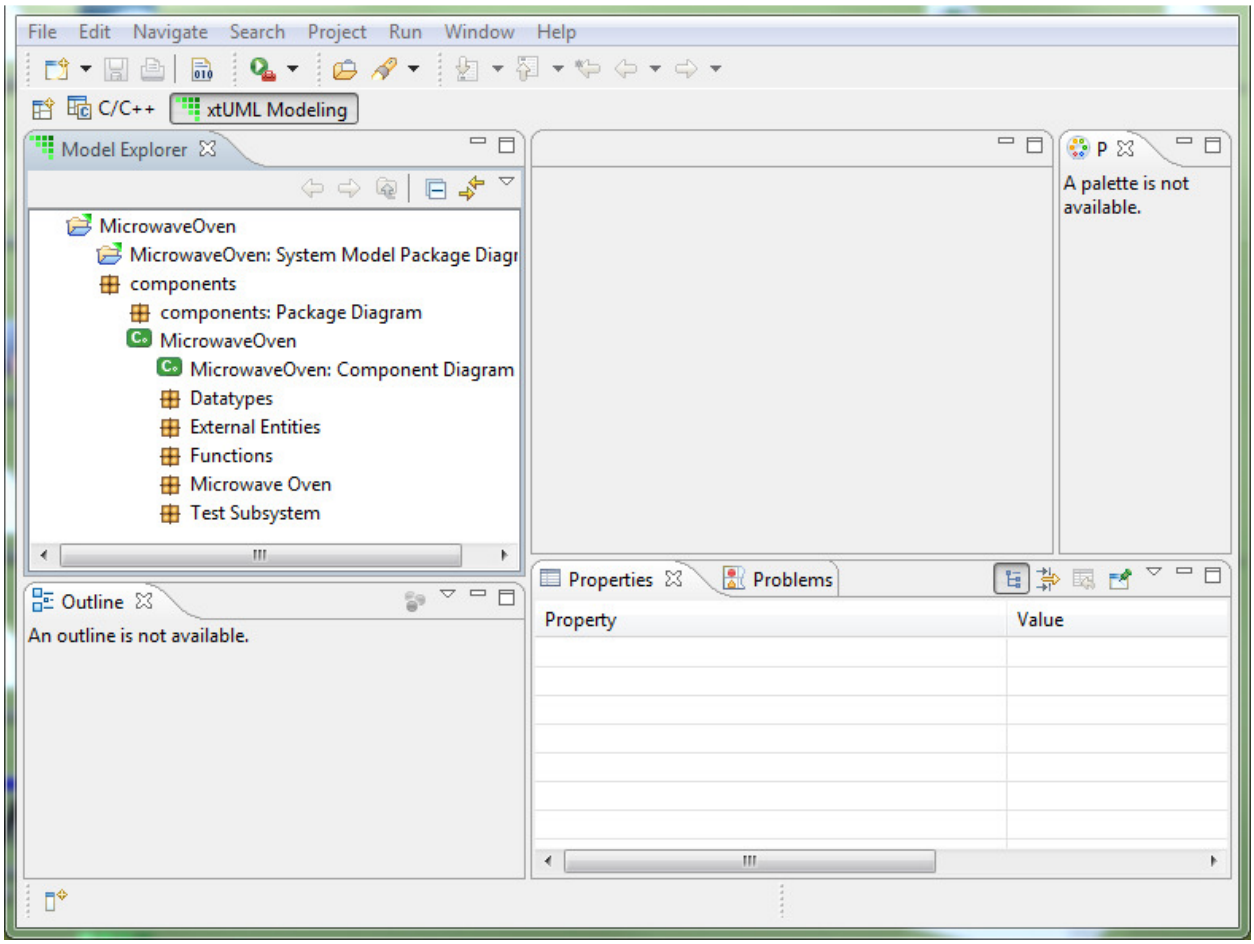
Now let's take a closer look at the models and the tool.

Model Explorer

Model Explorer is the view you use to explore and navigate the model. From this view you can drill down into the model to see all of the constituent model elements. It also provides the place from which you launch the diagram, activity, and description editors.

The Model Explorer view organizes a UML model hierarchically starting with the project, followed by one or more models. Each model contains packages that organize the model elements that make up the model.

Expand the Model Explorer view so that the contents of the `MicrowaveOven` project and model are shown. It should look similar to the image shown below.



Packages

If you drill down into the hierarchy, you will notice that there are many packages. Each of these packages contains elements used to build the microwave oven application. Note that these are so-called *generic* packages. They can be used to contain and group any sort of model element together.

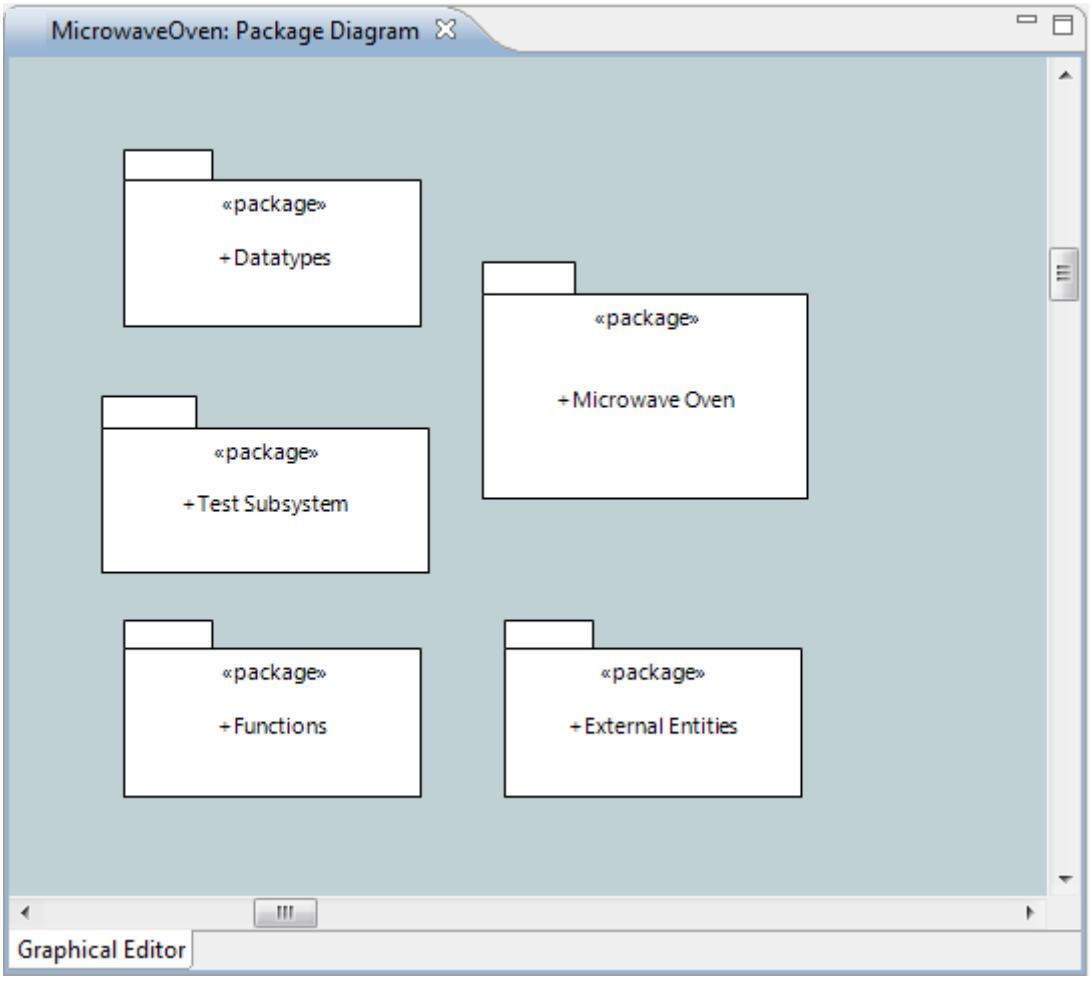
Taking a Look at the Diagrams

Each package has an associated diagram which can be accessed through the Model Explorer by double-clicking on the package or right-clicking on the package and selecting **Open**.

Package Diagram

Open the main Package Diagram by double-clicking on the `MicrowaveOven` package. This is the package icon () labeled “Microwave Oven”.

Packages are shown on diagrams as tabbed folders. There are packages for the data types, external entities, and functions. There are also two packages for the two subsystems that make up the microwave oven model: *Test Subsystem* and *Microwave Oven*.



All diagrams have a diagram toolbar which is located above the diagram. The left-most tools are common to all diagrams and are related to performing zoom operations on the diagram.

From left to right, the tools are:

Magnifying Glass - Plus Sign – Zooms in on the diagram. The short-cut key is Ctrl-Shift + Up Arrow.

Magnifying Glass - Minus Sign – Zooms out on the diagram. The short-cut key is Ctrl-Shift + Down Arrow.

Magnifying Glass - Page – Zooms all elements on the diagram.

Magnifying Glass - Elements – Zooms the currently selected elements only.

Zoom Percentage – Sets the zoom level to a predefined percentage.

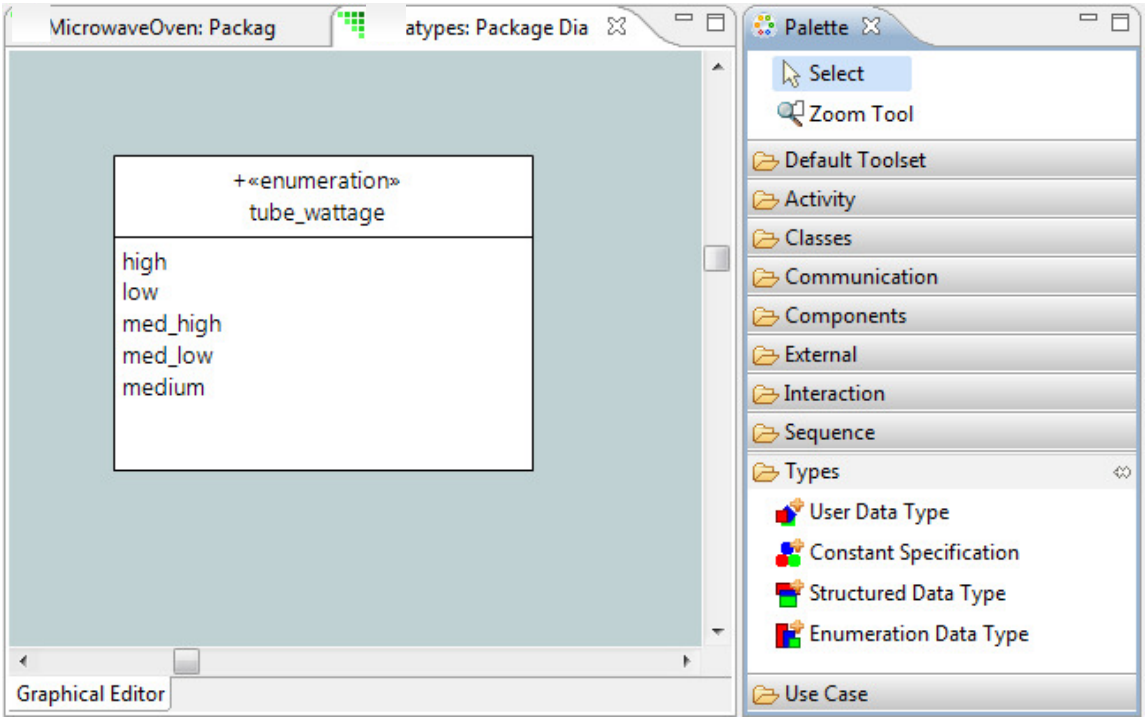
The button with the arrow is the selection tool. Use this tool to select an element on the diagram.

Double-clicking on a package inside a diagram brings up the diagram editor for the package.

Open the Datatypes package by double-clicking on it.

Data Type Package Diagram

The Data Type Package Diagram for the microwave oven looks similar to this:



Notice that the tools to the right of the package diagram are located in a window known as the **Palette**. The tools are used to draw user data types, enumeration data types, and data type packages. A package can contain any drawn element from any subset within the palette, but we can use the Types subset to draw datatype elements on this diagram.

The Datatypes Package Diagram above shows only a portion of the data types for the microwave oven. The other datatypes are not listed and are global datatypes, which do not need to be defined on this diagram. These are familiar primitives like *integer*, *Boolean*, *string*, etc. Once defined on this diagram, user data types can be used to type class attributes and parameters of UML elements.

Note that primitive data types are provided on the diagram for you and do not need to be defined.

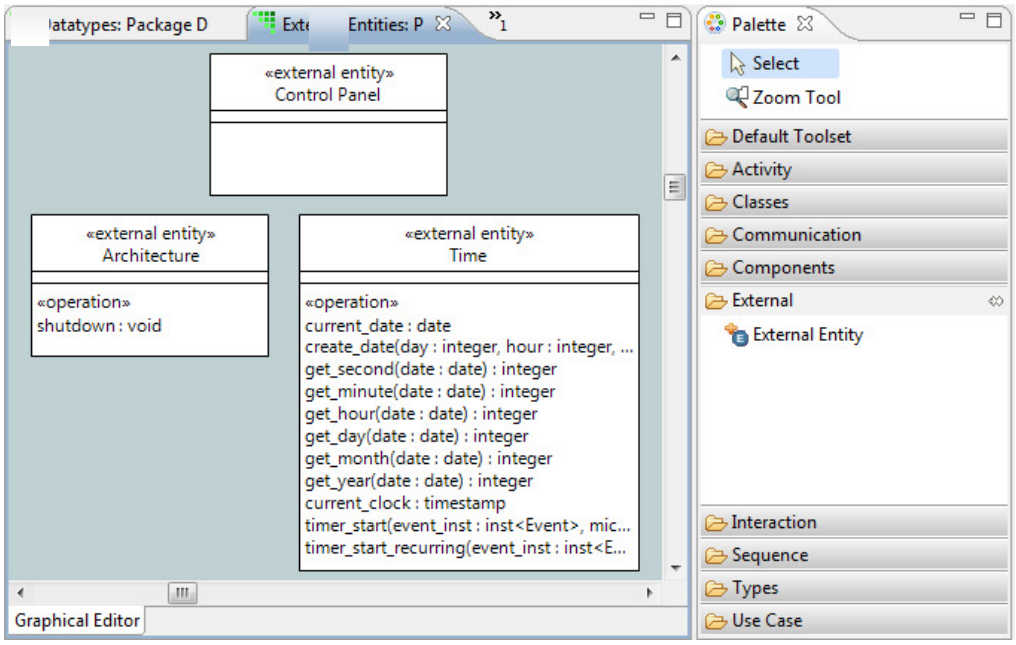
External Entities Package Diagram

Open the External Entities Package Diagram. You can do this by either going back to the Model Explorer or the Domain Package Diagram.

The External Entities Package Diagram shows those things, or entities, that are outside (external) to what we are modeling. EE's can represent other models, or code that our model needs to interact with.

The microwave oven model has three EE's: Control Panel, Architecture, and Time. The Control Panel represents the control panel of the microwave. The Architecture EE represents the model compiler and contains one bridge operation named shutdown which is used to shutdown the microwave application (more on this later).

The Time EE provides time and date bridge operations so that our microwave oven model can have access to dates, time, and timer (i.e., delayed) events.



Functions Package Diagram

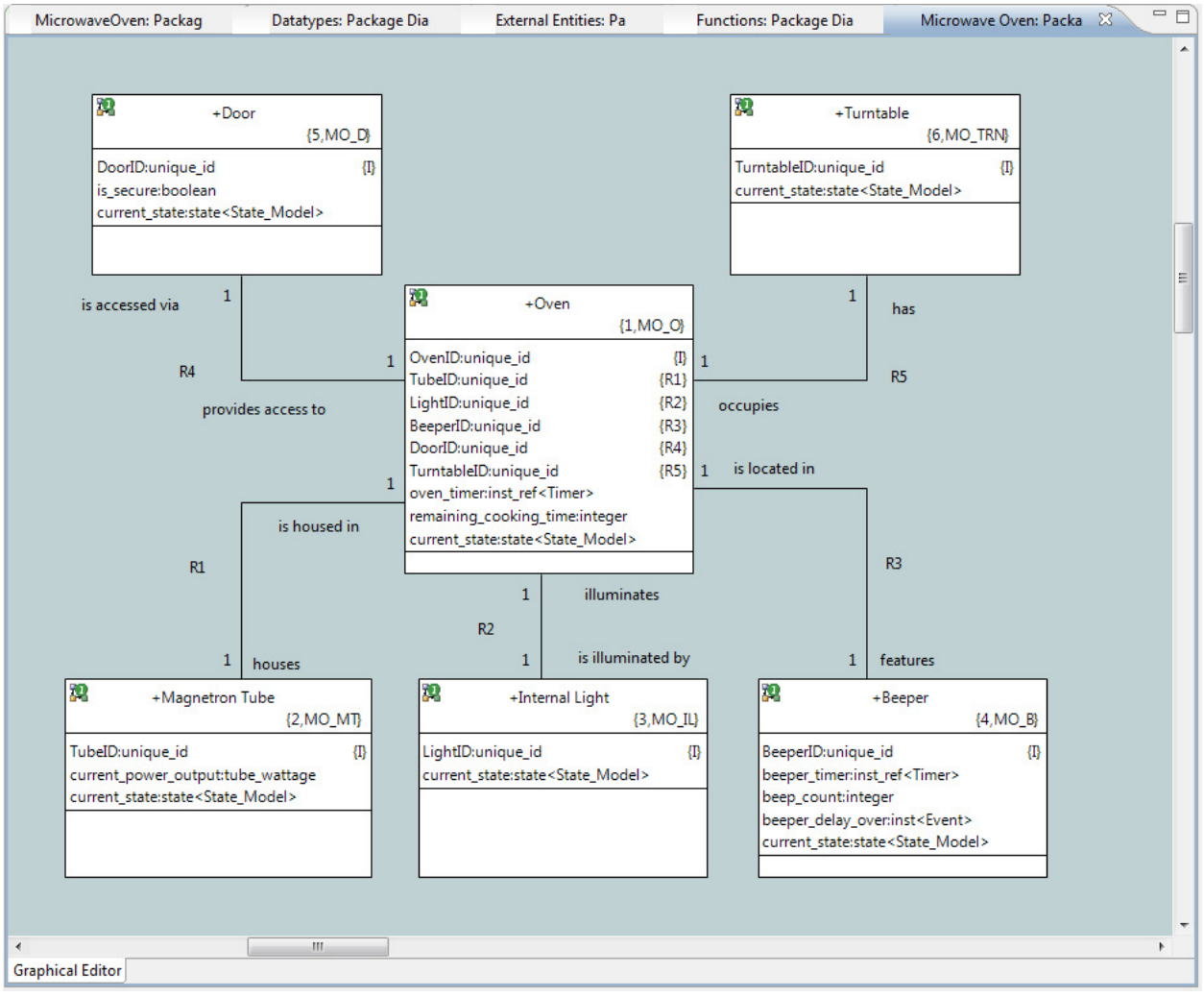
The Function package provides a way to organize functions. Functions do not have a representation on the diagram – there is no UML notation for them. They can only be seen in the Model Explorer.

The microwave oven model uses several functions to provide initialization and simulate the operation of the microwave.

Microwave Oven Class Diagram

Packages can also provide a convenient way to break up a model into manageable chunks. The microwave oven model contains two subsystems, one that models the microwave oven, and one that tests it.

Double-clicking on a package containing classes will show what is known as a class diagram. The class diagram for the Microwave Oven package is shown below.



When the microwave oven was modeled, the following six classes were identified:

- Oven
- Door
- Turntable
- Beeper
- Internal light
- Magnetron tube

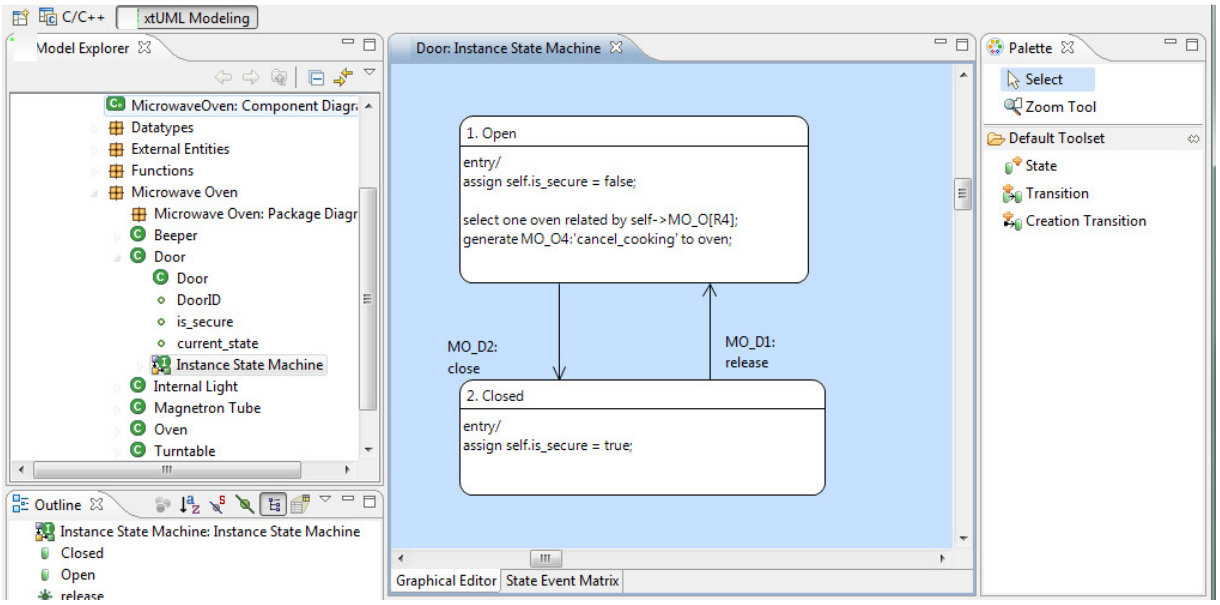
The Class Diagram represents the classes as boxes. The class attributes are the variables listed in the boxes, and the associations (relationships) between individual classes are the lines between the boxes. Associations have multiplicity and are named with text phrases.

The `Oven` class (at the center of the screen) has an association with several of the other classes. For example, there is an association, labeled `R4`, between the `Oven` and the `Door`. The `R4` association is drawn as a solid, undirected line with a 1 at both end points, which denotes a one-to-one association. In our example, all of the associations are one-to-one.

The phrases “is accessed via” and “provides access to” describe the association between the two classes.

State Machine Diagram

A State Machine Diagram is used to represent the behavior of a class. Let’s take a look at the State Machine Diagram for the `Door` class. To open the State Machine Diagram for the `Door`, navigate inside the Door class in the Model Explorer and double-click on Instance State Machine.



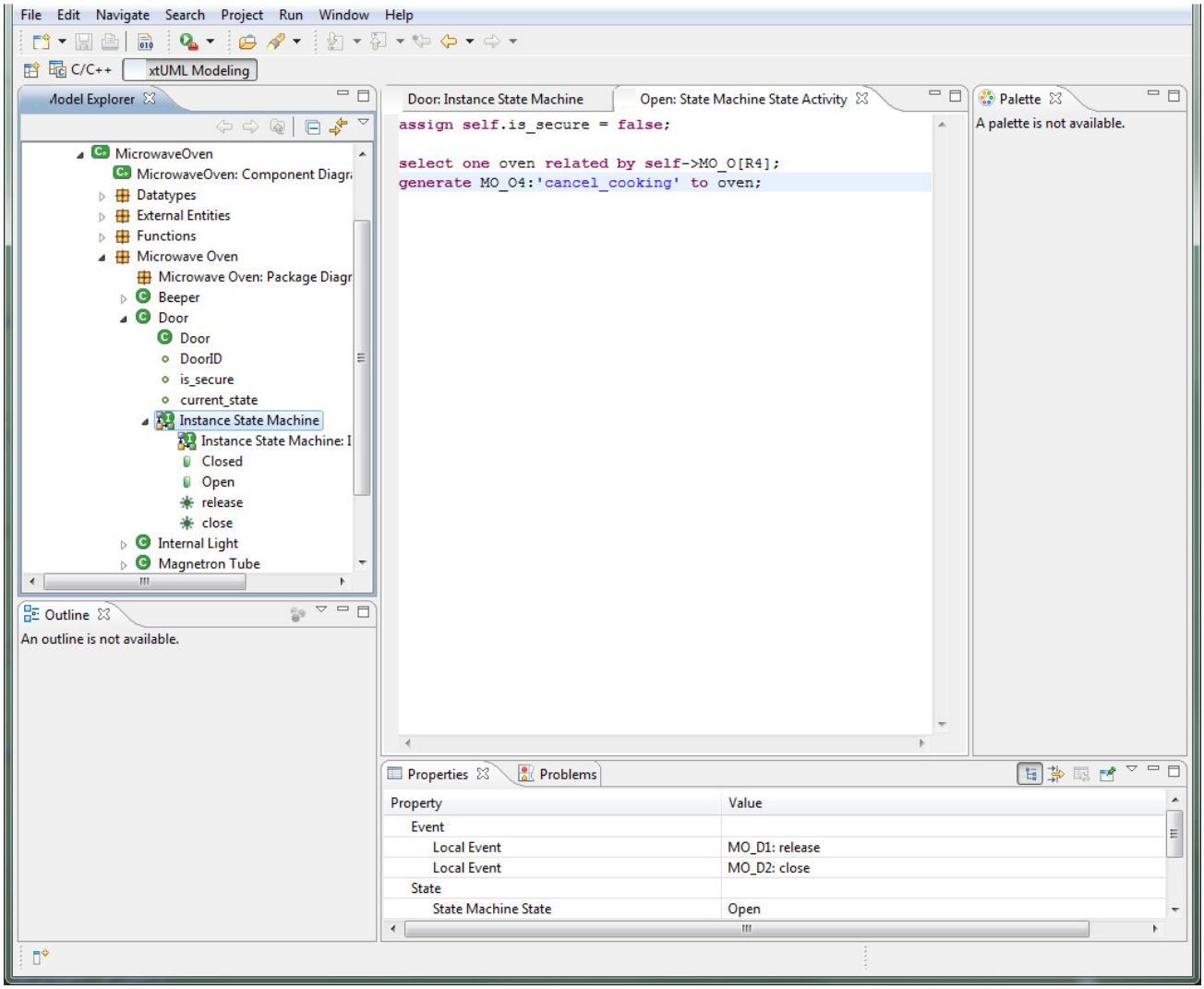
A state is represented by a rectangle with rounded corners. Each state has a name and number. The directed arrows represent state transitions and are labeled with the event that causes the transition. Activities are shown inside the rectangle and represent the processing that occurs on entry to a state.

The `Door` state machine has two states, `Open` and `Closed`. When the `release` event is received and the door is closed, it transitions to the `Open` state. Likewise when a `close` event is received in the `Open` state the door transitions to the `Closed` state.

Activities

The processing for each state in the State Machine Diagram is specified with activities written in OAL (Object Action Language). As an example, let's look at the activities for the `Open` state of the `Door` state machine.

Accessing the activity for a state can be done in one of two ways: by double-clicking on the state in either the Model Explorer or the State Machine Diagram.



The behavior being modeled for the `Door` is this: when the door is opened, the oven is not secure and must be told to stop cooking. The activities written for the `Open` state reflect this by setting the attribute variable `is_secure` to `false` and sending the event `MO_O4:'cancel_cooking'` to an `Oven` instance.

The Activity Editor is a context-sensitive editor that provides syntax and semantic highlighting, showing you errors as they are typed. Errors are reported in the Problems View.

Testing our Microwave Oven

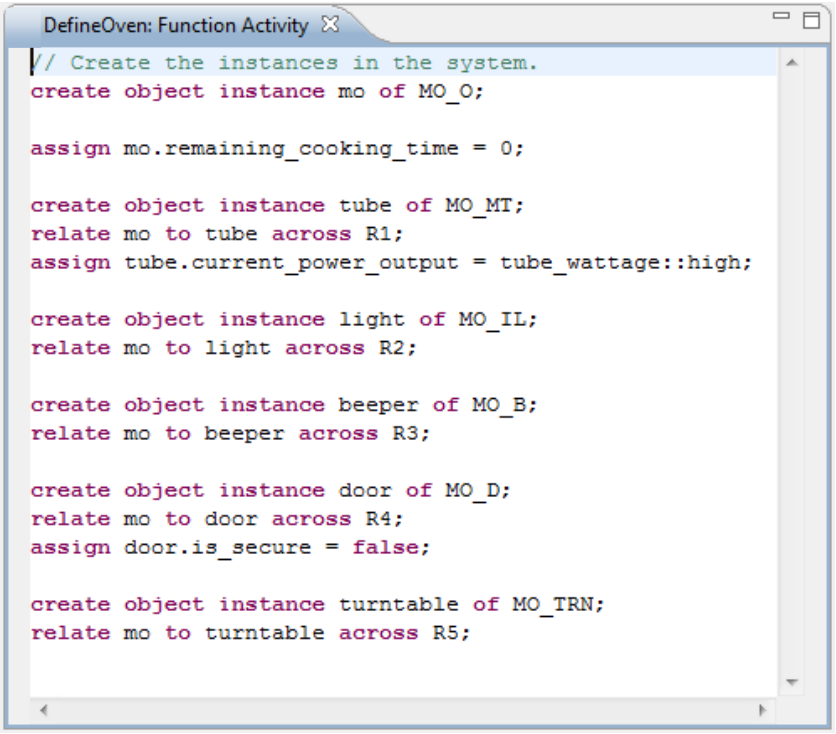
So far, we have seen the modeling elements for the microwave oven application. In order to test the application model, we need to see how it operates in typical situations. For that purpose we've created a few functions and a subsystem dedicated to testing the microwave.

Functions are one of the places where we can model activities. This makes them a convenient place to create instances of the classes in the microwave oven model; oven, door, tube, and the associations between them. We can also generate events to kick off the various scenarios.

The "`Test Subsystem`" subsystem models a set of test scenarios used to exercise the microwave oven models. One of the nice things about using a subsystem to model the scenarios is that our tests are modeled at the same level of abstraction as the microwave oven model.

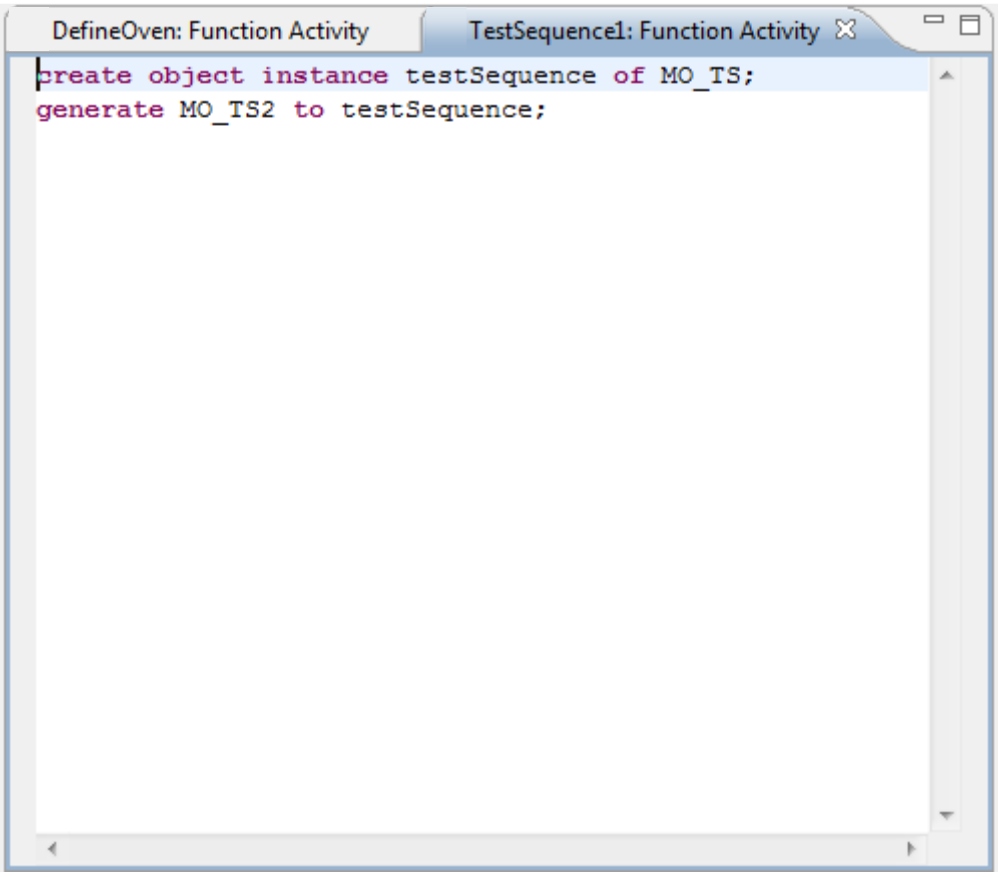
Initialization

Take a look at the function `DefineOven`. You can access the activities for this function by double-clicking on it in the Model Explorer. This brings up the Activity Editor for the function.



The `DefineOven` function contains activities that create the instances of the classes, set attribute values, and establish the associations between them. Note that the Activity Editor highlights keywords in red and comments in light green.

Now take a look at the `TestSequence1` function. This function is used to generate an event that kicks off the simulation.

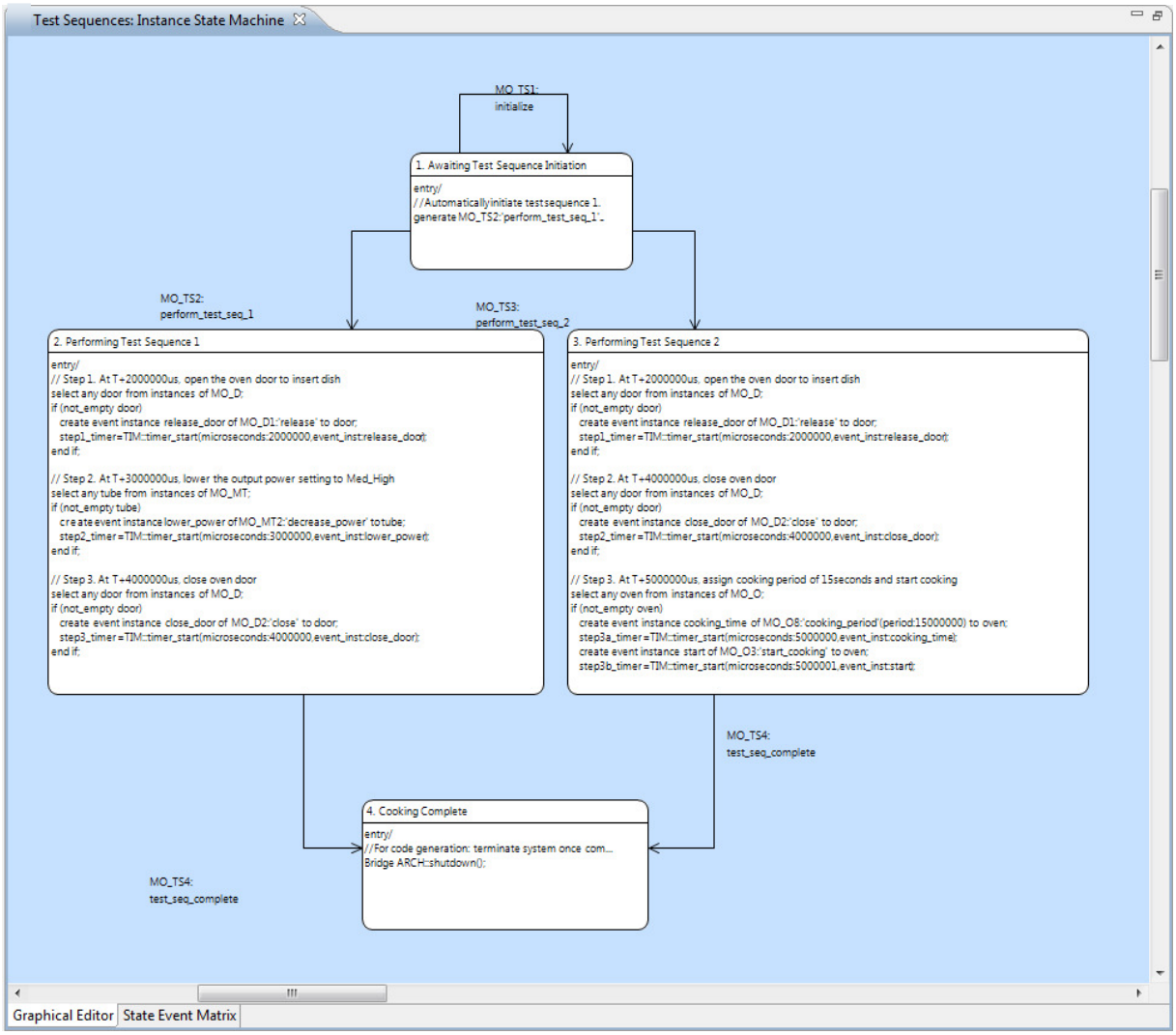


When we create the instance of the class `MO_TS`, we get a handle back (`testSequence`) which can then be used to generate the `MO_TS2` event to the instance of `Test Sequences`.

Test Subsystem

Open up the state machine for the `Test Sequences` class so we can see what happens in our model when the `MO_TS2` event is received. Remember that the class you are looking for is in the `Test Subsystem` subsystem.

The State Machine Diagram for the `Test Sequences` class is shown below:



There are four states in the diagram. State 1 (`Awaiting Test Sequence Initiation`) sends an event to the next state to start a specific test sequence. States 2 and 3 (`Performing Test Sequence 1` and `2`, respectively) are specific test scenarios for operating the oven. State 4 (`Cooking Complete`) takes steps necessary to terminate the application (note the bridge call to the `ARCH` external entity).

You can see a portion of the activities for state 2 in the diagram above. Essentially, it specifies the following processing:

- Open the door.
- Lower the power setting to medium high.
- Close the door.
- Select a cooking time of 10 seconds.
- After 20 seconds, open the door.

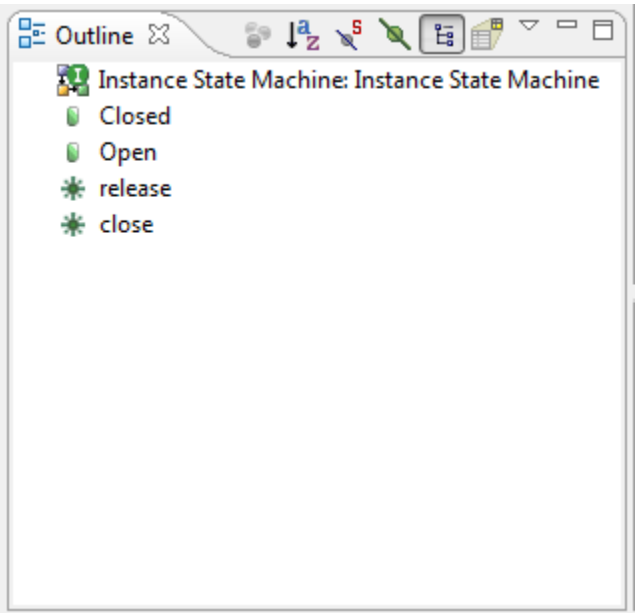
By default, all state machines start out in the lowest numbered state. This means that the event sent by the `TestSequence1` causes this state machine to transition from state 1 to state 2. Upon entering state 2, the activities are executed and the models begin to execute.

A Few Other Views and Editors

There are a few other views and editors that you need to be aware of when using the tool.

Outline View

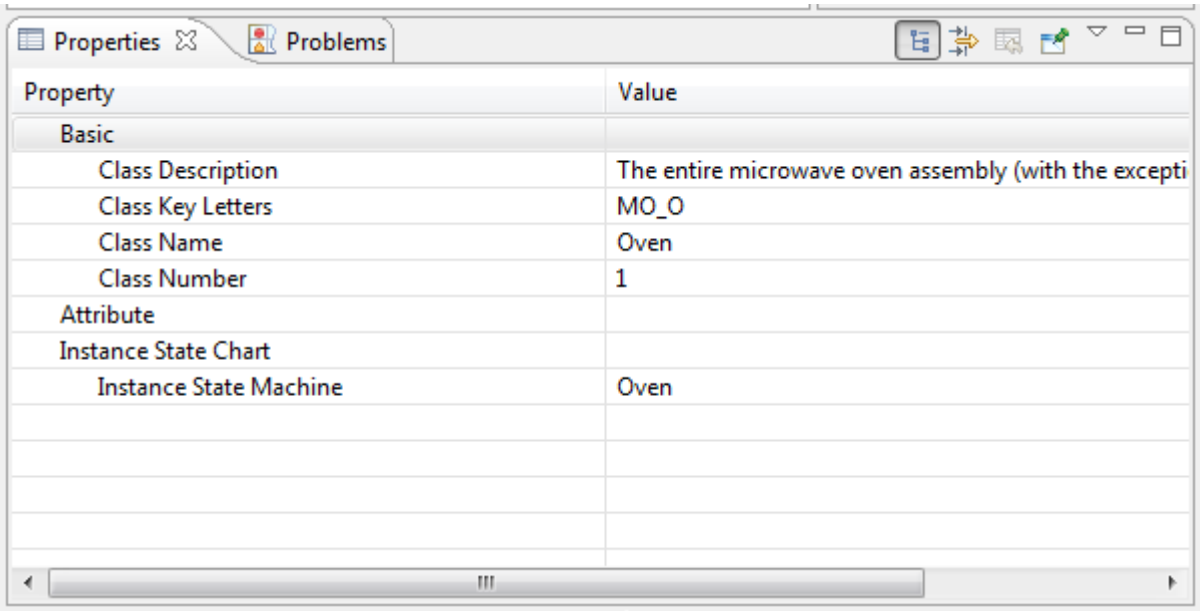
The Outline view is active when a diagram editor has focus. It shows an outline of the model elements on the diagram. Here is what the Outline view looks like for the `Door` state machine.



Bring up the Outline view and see how it changes when different diagram editors are selected.

Properties View

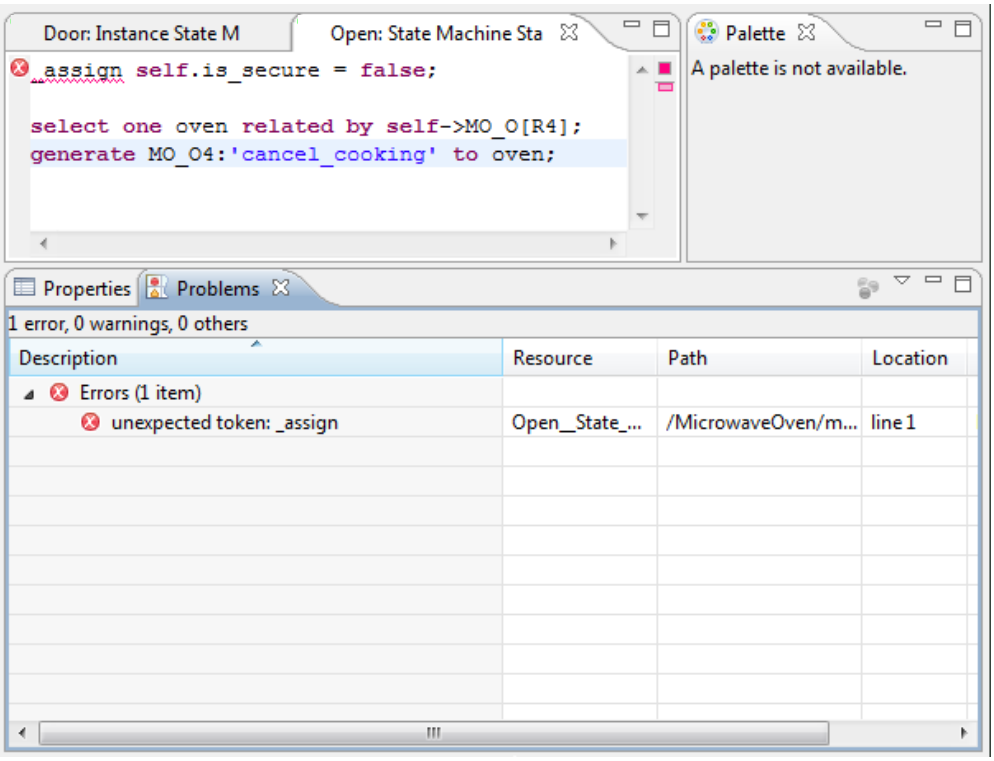
You can use the Properties view to change the attributes of the currently selected model element. There are basic and advanced properties. The following is what the Properties view looks like for the `Oven` class:



Problems View

Problems with activities (parsing errors) are reported on the Problems view. A brief description of the problem is noted along with the location. Double-clicking on a problem in the Problem view brings up the editor for the activity associated with the problem.

The "`unexpected char`" problem appears in the Problems view when an underscore is pre-pended to the `assign` keyword.



Description Editor

Virtually every model element can have a description. Descriptions are entered using the Description Editor. To access the description editor for a model element, select it (in the Model Explorer or Diagram Editor), right-click, and select **Open With > Description Editor**. The following shows the Description Editor for the `Oven` Class.