Engineering 205

Model Based System Engineering

Student User Guide 2:
Papyrus Project, Model, and Diagrams Export and Imports

Myron Hecht

Version 1.1

September 26, 2015

Revision History

Date	Version	Author	Change	File Name
9/15/2015	1.0	Myron Hecht	Original	UG2_Project Model and Diagrams
9/26/2015	1.1	Yang Shen and Myron Hecht	Added figure for file export in Section 3	UG2_Project Model and Diagrams2

Table of Contents

List of Figures

1 Introduction

This assumes that you have successfully installed the Papyrus software and describes the next steps. Section 2 shows how to create a project file, section 3 describes creating package, use case, and requirements , and sequence diagrams (primarily to demonstrate linkages with use case diagrams; a more complete descriptions of sequence diagrams appears in the third user guide), and Section 4 describes importing and exporting project files.

2 Creating a Project and Model

This section describes how to create a Project and Model in Papyrus. A project is in essence a file directory into which Papyrus will write the files that store the models. Every project must have at least one model. Recall the file hierarchy for Eclipse and Papyrus: from the bottom up, SysML (and UML) model elements and diagrams are contained within Papyrus models which are contained within Eclipse projects which are finally, contained within the Eclipse workspace. Therefore, before you can create any diagrams, you need to define a project file and then at least one model.

The mouse sequence starts from the top menu bar "File" item (upper left most) and is as follows

File > New Project

The following dialog box appears.

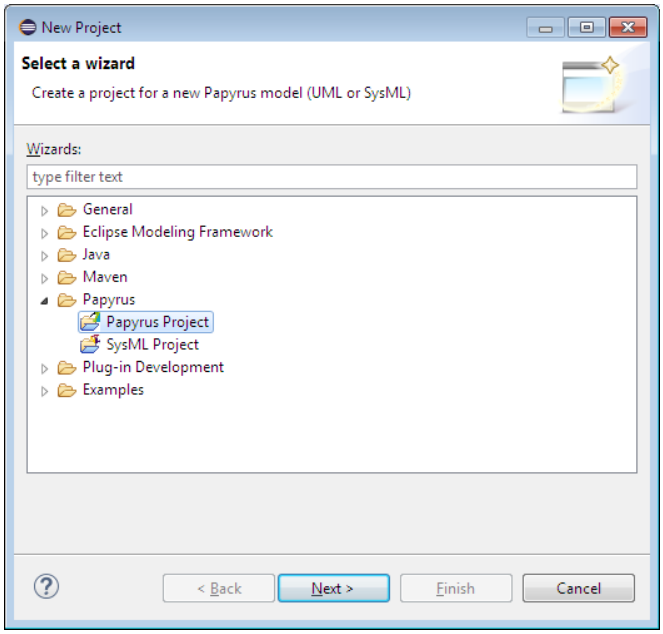

Figure 1. Wizard for Opening a Papyrus Project

Select Papyrus Project (not SysML project).

After you press next, the following dialog box pops up.

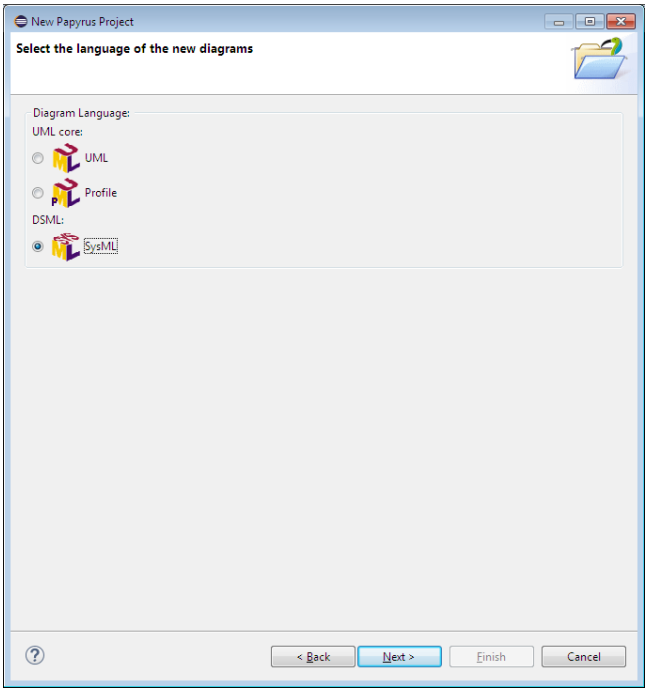

Figure 2. Selecting SysML as the Language for the Project

SysML appears as a "Domain Specific Markup Language" DSML because it is an extension of UML into the SysML domain. Select it, click “next”, give it a name, and finish.

Then go back to and select the Papyrus perspective by clicking on the perspective icon.

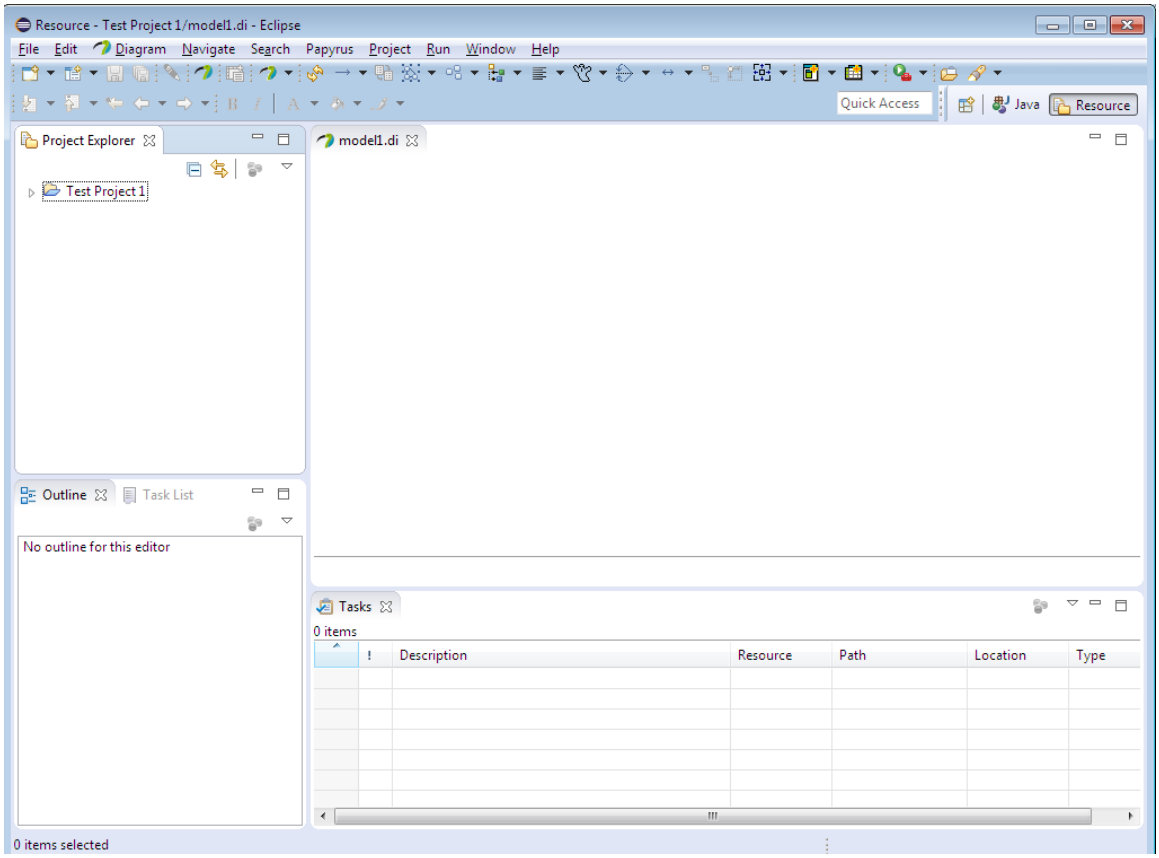

Figure 3. Eclipse Workbench before switching to Papyrus perspective

The following figure shows another way to switch perspectives, starting with the Window menu bar item.

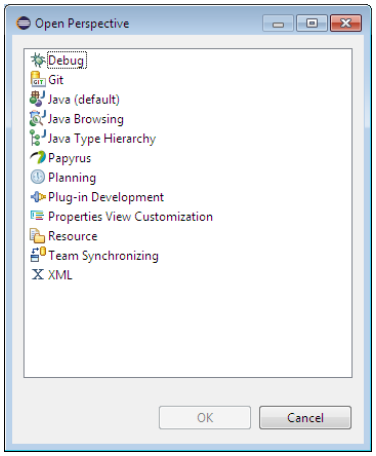

Figure 4. Another way of switching perspectives

The following figure shows the Eclipse workbench after the perspective change. Note the appearance of the Model Explorer on the left.

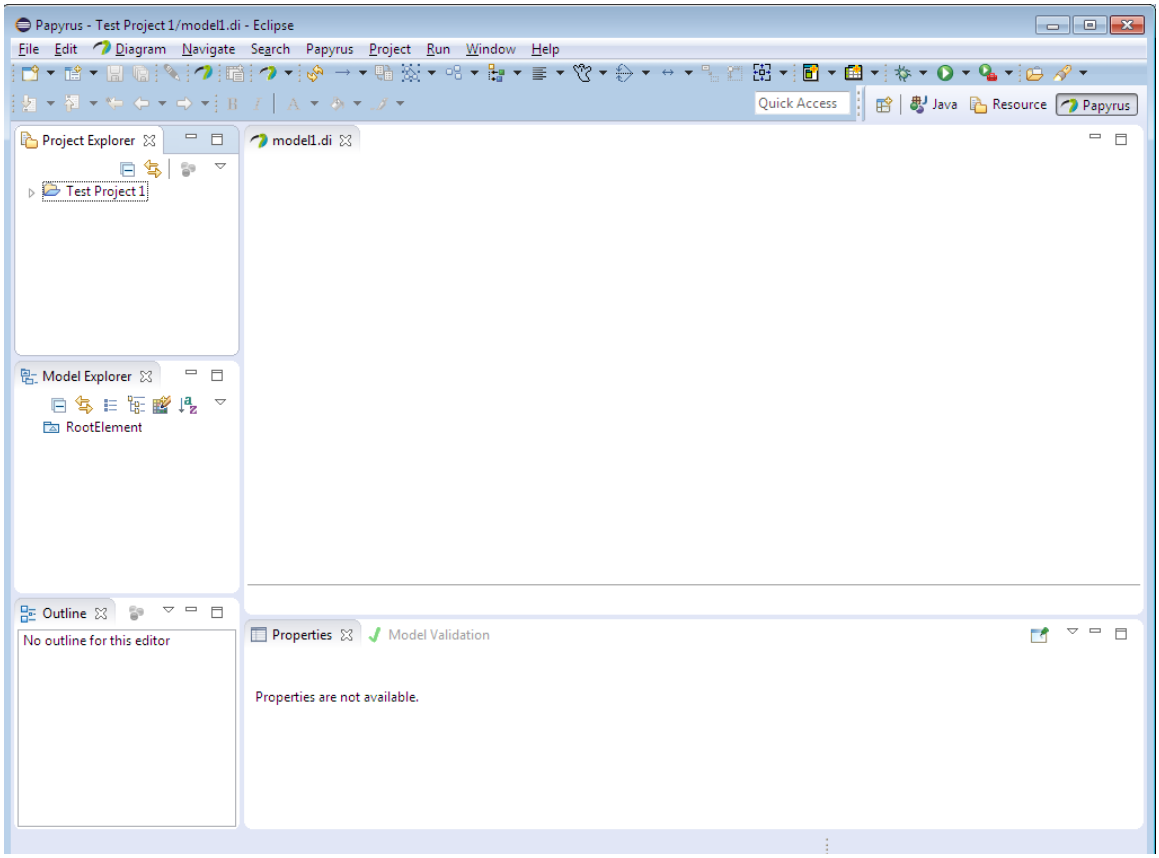

Figure 5. Eclipse Workbench with Papyrus perspective

3 Creating SysML Diagrams

In order to create diagrams in Papyrus, you must have both created a project and switched to the Papyrus perspective. Refer to the previous section if you have not completed those steps.

Once you have created a Papyrus project model and switched to the Papyrus perspective, select the root element in the Model Explorer window. The property window should now show properties of the root element. The properties window shows UML properties because Papyrus uses the same repository structure for both UML and SysML. SysML properties are displayed only for model elements that are extensions of the UML base classes (e.g., requirements or blocks).

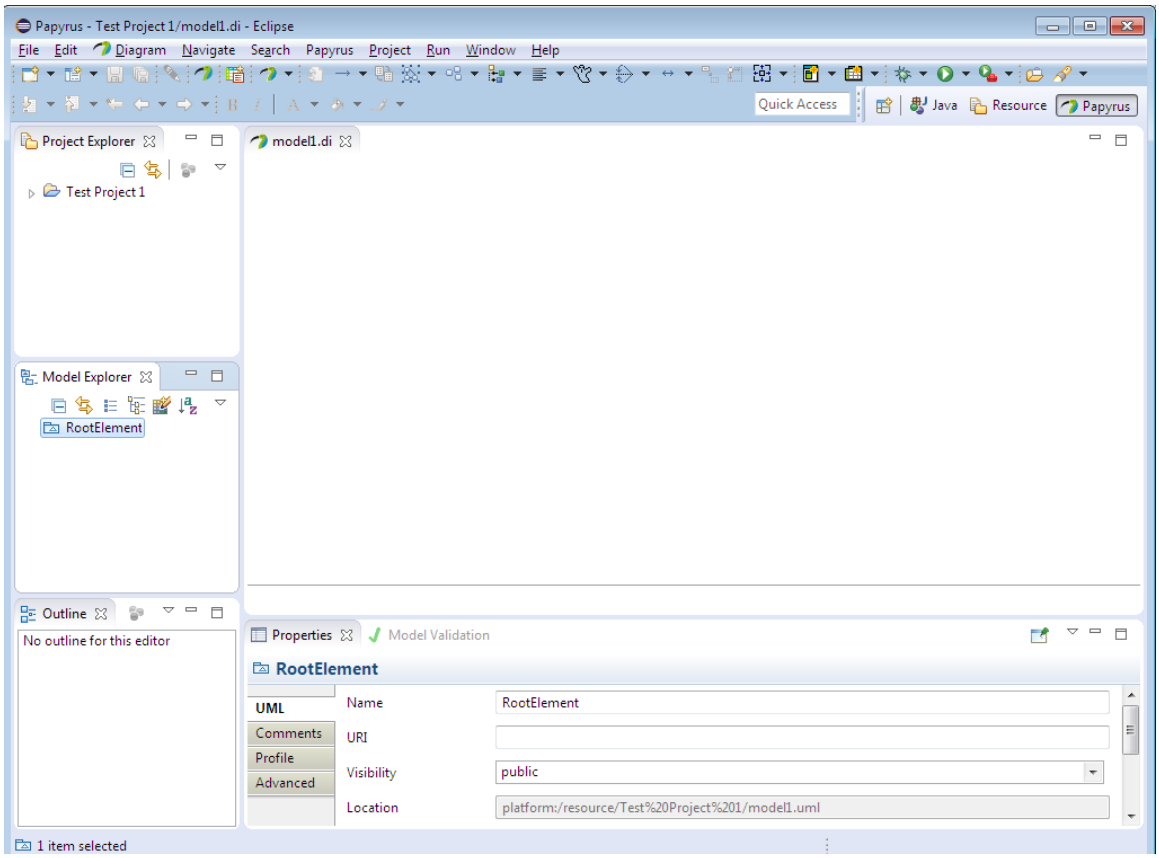

Figure 6. Selection of the Root Element

3.1 Creating a Package Diagram

You can create diagrams only when you are in the Papyrus perspective and have selected the root element. See the immediately previous description if you have not done so.

Once you have selected the root element in the Model Explorer window, go to the Papyrus menu item. Perform the following mouse sequence

PAPYRUS>New Diagram>Package Diagram.

The package diagram pallet will appear on the right as shown in the following figure.

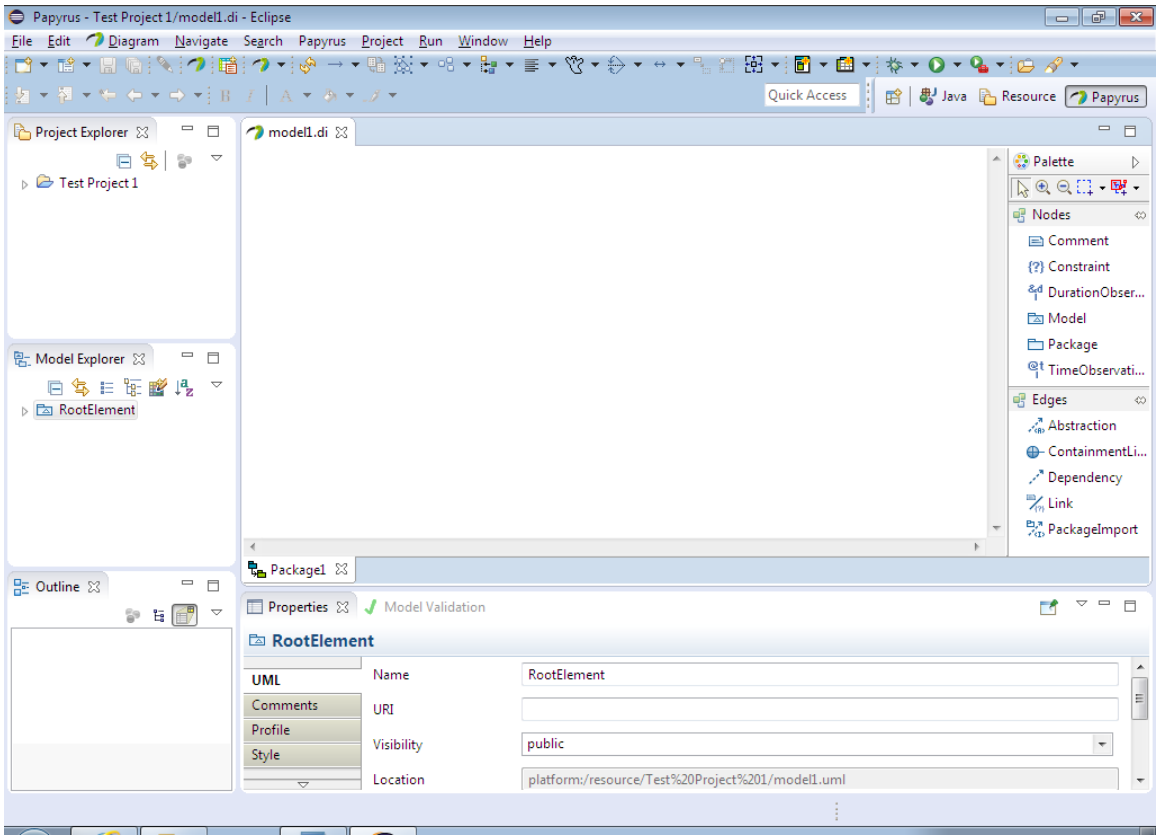

Figure 7. Package diagram pallet on the Eclipse Workbench, Papyrus perspective.

Note that the pallet has two sections, Nodes and Edges. Nodes are symbols and edges are connectors or relations. Also note that not all nodes and edges are visible on the pallet (this screen shot is at a fairly low resolution). Up and down arrows that enable you to scroll through the other elements appear in the pallet sections when you move the mouse to either the top or the bottom of each section.

Drag two packages on to the canvas and connect them with a containment edge (circle with a plus symbol). Then, in the Model Explorer window, click on the right pointing arrow to the left of the root element link and see Diagram 1 and Package1 as shown in the following figure. Clicking on the right pointing arrow of Package1 will enable you to see Package2

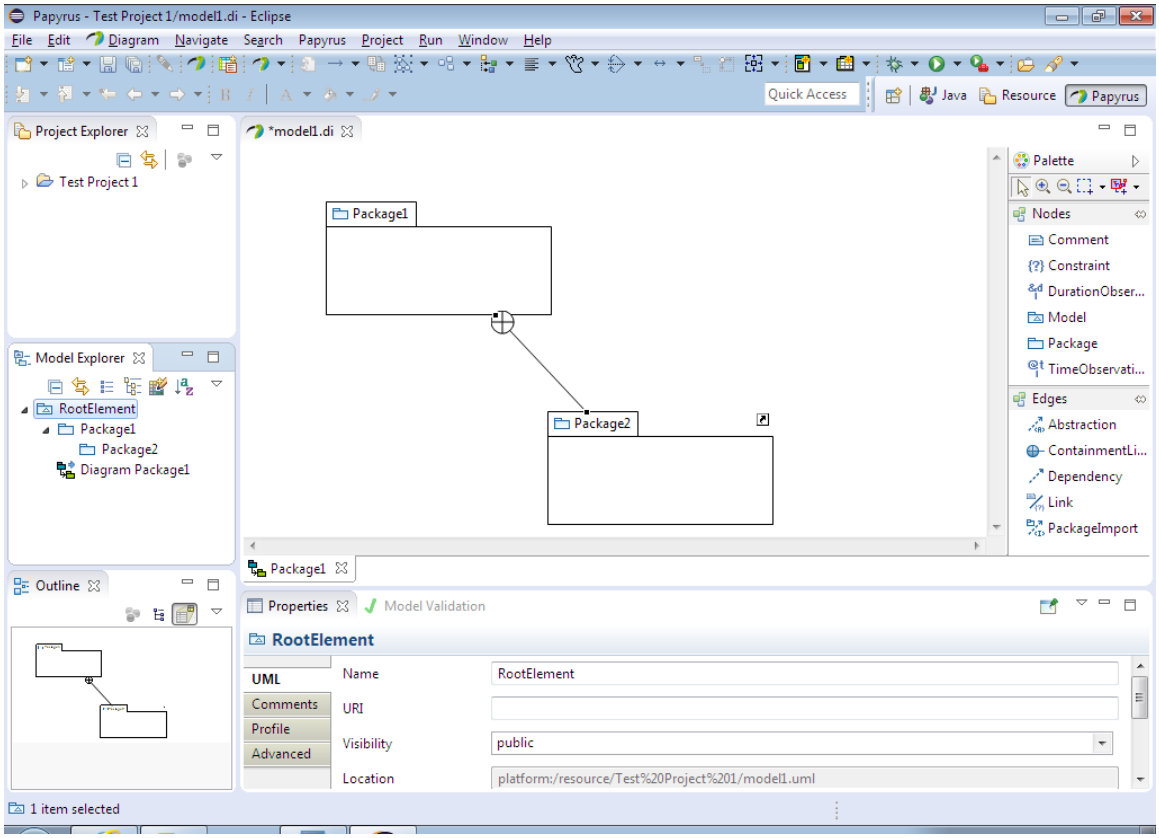

Figure 8. Creating a Package Diagram in Papyrus

3.2 Use Case Diagrams

As was the case with the package diagram, you must have the Papyrus perspective open. You must have also selected an element in the Model Explorer window that will "own" the diagram. However, it need not be the root element (note that not every model element in the model explorer can contain a diagram). For the purposes of this document, we did select root element and opened a Use Case diagram with the following mouse sequence

Papyrus>New Diagram>Use Case Diagram

The following figure shows the result.

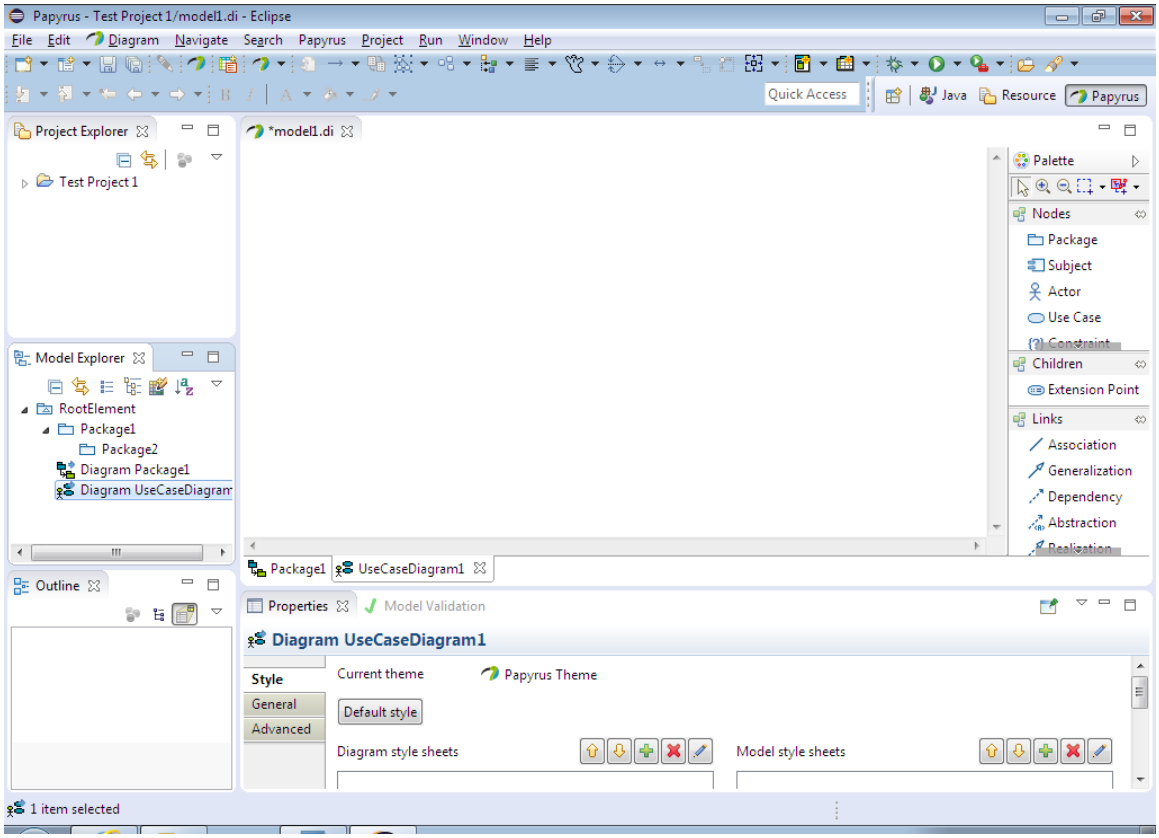

Figure 9. Creating a Use Case Diagram in Papyrus

Create the use case diagram shown in the following figure by dragging an actor (stick figure) from the pallet, and then dragging an oval (called a "use case" in this figure; we use the term "behavior" for the oval and reserve the term "use case" to refer to the combination of the actor and the behavior) and then used an association to connect them. Their designations were UseCase 1 and Actor1. Repeat the actions to create a second actor and use case called Actor2 and UseCase2 which are visible on the figure below. Drag a package on the canvass, expand it, and then drag another actor and use case and connect them as well.

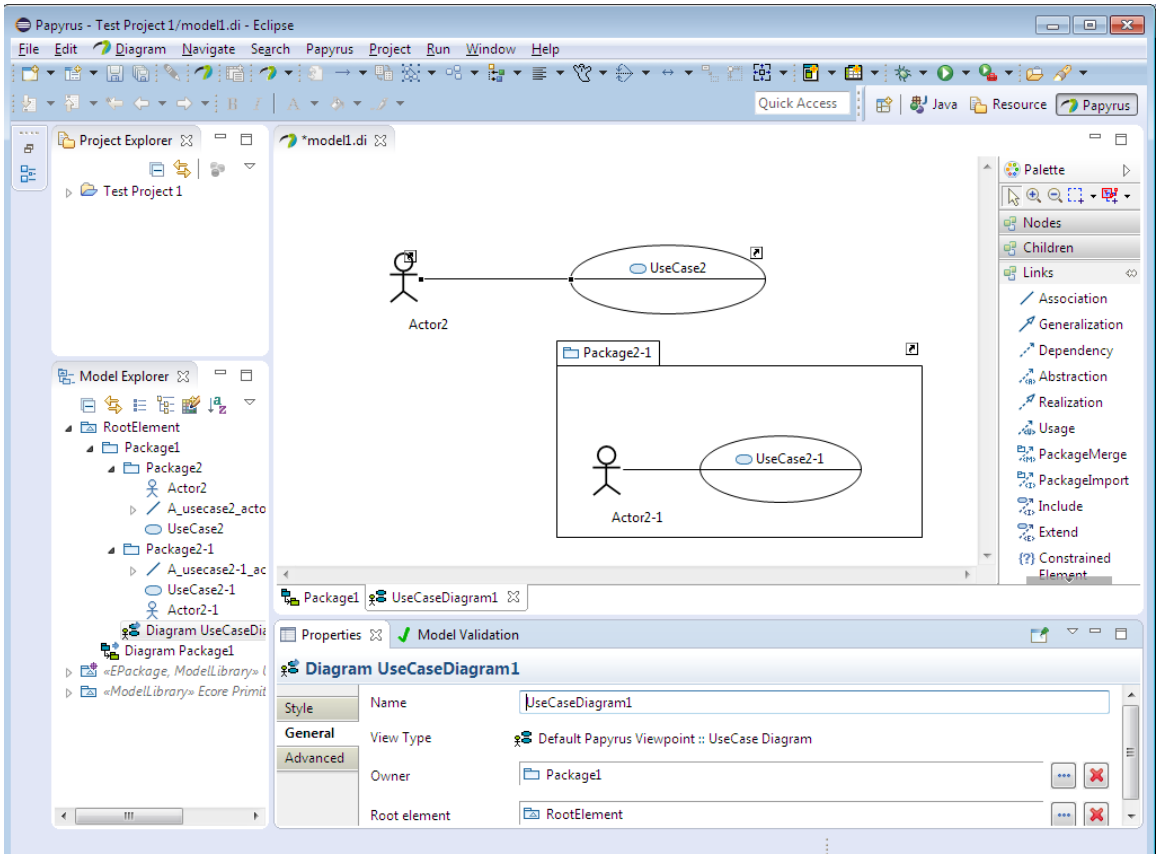

Figure 10. Creating a Use Case Diagram in Papyrus

Because we created this diagram under the root element, all the model elements (symbols) on the diagram appeared under the root element in the model tree. This is not a good practice. A better structure would be to place the entire Use Case diagram into Package1; Package 2-1, Actor2, the association, and UseCase2 into Package2; and Actor2-1, the association, and UseCase2-1 into Package 2-1. We first created new packages on the model explorer by right clicking on the root element and adding a new element through the drop down menus. We then dragged the elements in the model tree into the target packages. To drag an item to the intended target parent, you have to be sure that the intended parent is highlighted (underlined).

Another way to change the parent (actually, the owner in this case) is to change the Owner in the Property View (the window under the canvas)

3.3 Sequence Diagram

Although we present sequence diagrams in the context of Internal Block Diagrams later in this course, they can be used to represent interactions between actors and behaviors (use cases) as well. To create a Sequence diagram for Actor2-1 and UseCase2-1, the same prerequisites as were discussed for the Use Case diagram above apply. The procedure for creating a new diagram is also similar. The mouse sequence is as follows:

Papyrus>New Diagram>Sequence Diagram

Add Actor2-1 and Use-Case 2-1 by dragging them onto the canvas from the model tree. They should appear on the diagram as lifelines as shown on the following figure.

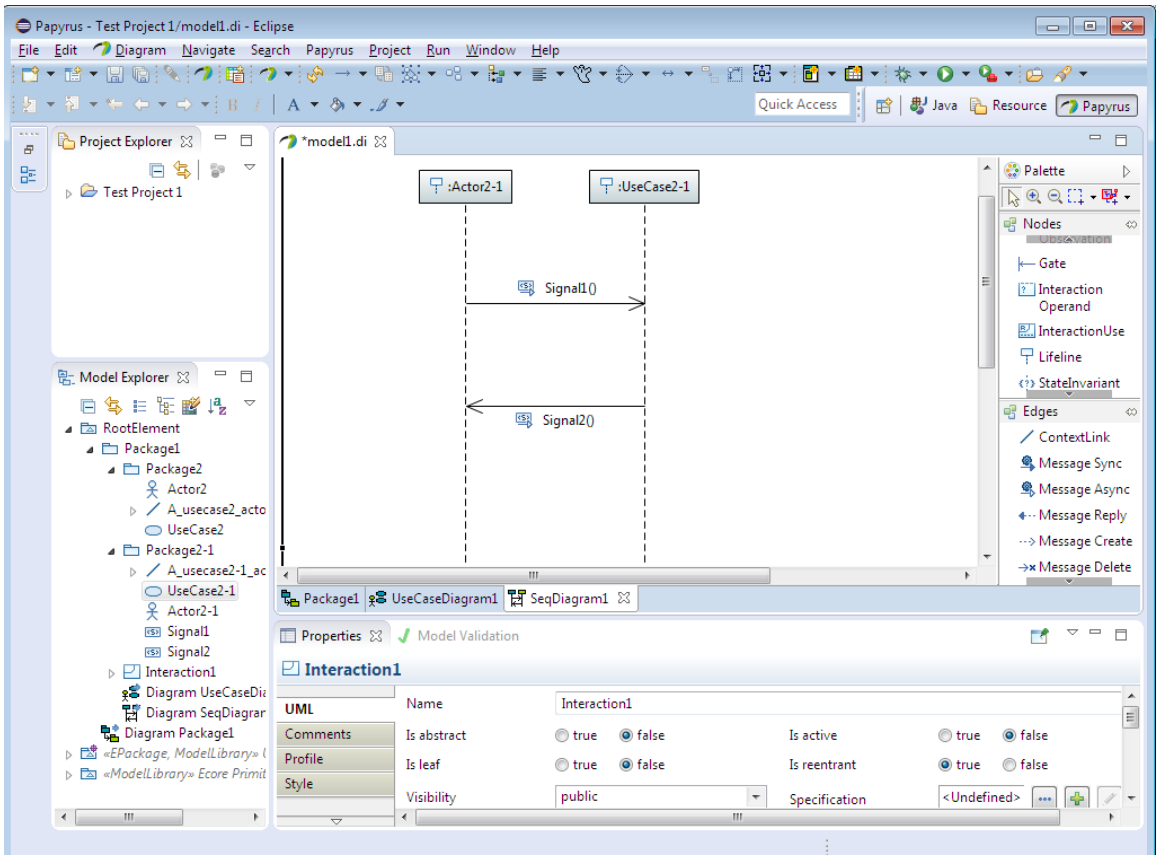

Figure 11. Sequence Diagram with Actor and Use Case

Work around note: In one case, I attempted to drag the actors onto the diagram to create lifelines but it didn't work so we created one lifeline using a lifeline item from the pallet. Then we I added actor2-1 and then deleted the "starter" lifeline and added Use-Case 2-1

Create an asynchronous signal by dragging the Message Async line (edge) from the pallet to the actor I then dragged one end to the use case. Then a "create new message" dialog box shown in the following figure appeared. The dialog gives you the options of using an existing element (signal or operation), creating a new one, or selecting no element. I chose to create a new signal.

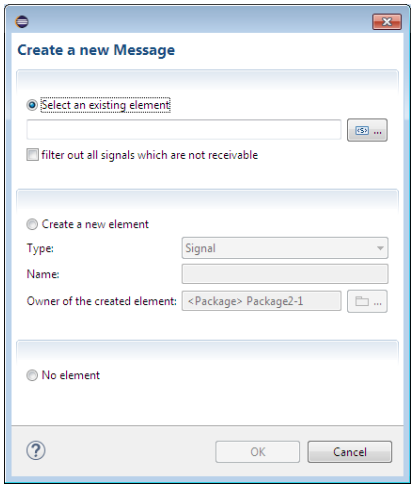

Figure 12. Dialog Box for creating a new message

Workaround Note: You have to be careful with the connection but when the mouse is over the right place, the "prohibited" (circle with a slash) symbol disappeared and I could left upon the button. Try zooming in (increasing the magnification) if you are having difficulty. You can do this by pressing “control” simultaneously with moving the mouse wheel (or slide control on a touch control pad of a laptop computer)

3.4 Requirements Diagrams

Package, use case, and sequence diagrams are common to both SysML and UML. We will not create our first SysML-unique diagram: the Requirements diagrams as shown in the following figure

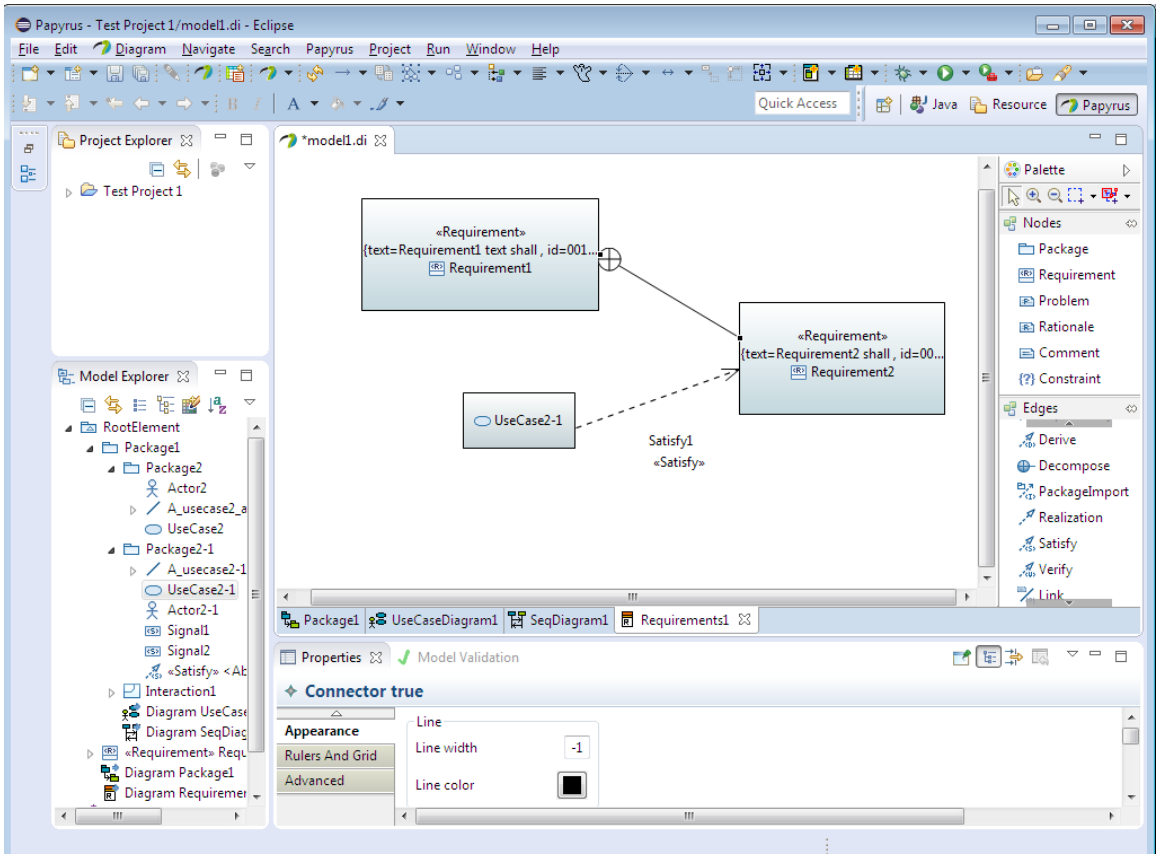

Figure 13. Completed SysML requirements Diagram

Create a Requirements Diagram by selecting RootElement on model explorer, then

Papyrus>New Diagram>Requirements diagram

Create Requirement 1 by dragging a requirement node onto the canvass. In the properties view with the SysML (not UML) tab pressed as shown in the figure below, enter the requirement ID and text. You use the SysML tab because the requirements diagram is a SysML extension of a UML base class but is not a part of UML.

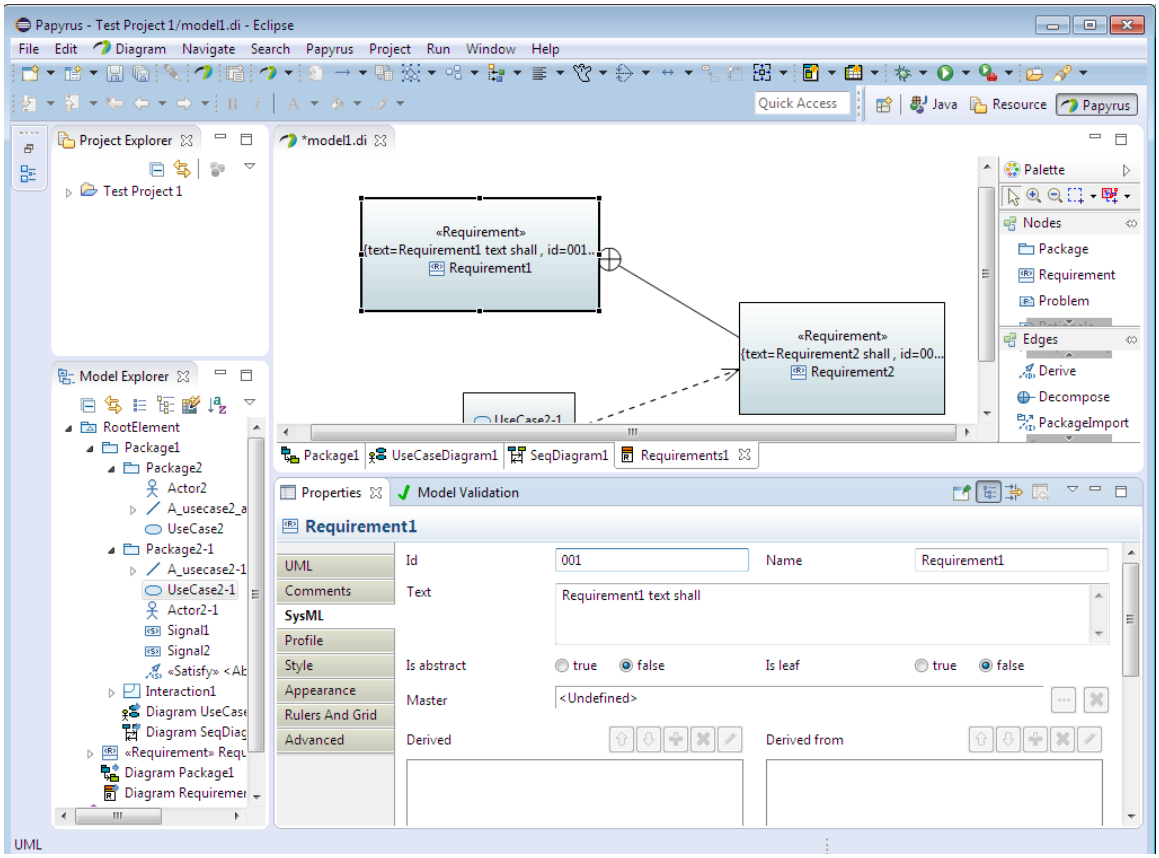

Figure 14. Inserting a requirement ID and text in the property view (with the SysML option selected)

Next, create a second requirement and use a decompose relationship (one of the "edges" on the diagram pallet) to connect to the first. We now want to link this requirement to another element of the model that doesn't appear on the diagram. Let's use UseCase2-1. Drag UseCase2-1 *from the model explorer* window onto the canvas and used a satisfy relationship to connect it to requirement 2. This should show you the importance of the Model Explorer – it is your view into the model repository that is independent of the diagram on the canvas. When you look at the model explorer, you should see the following

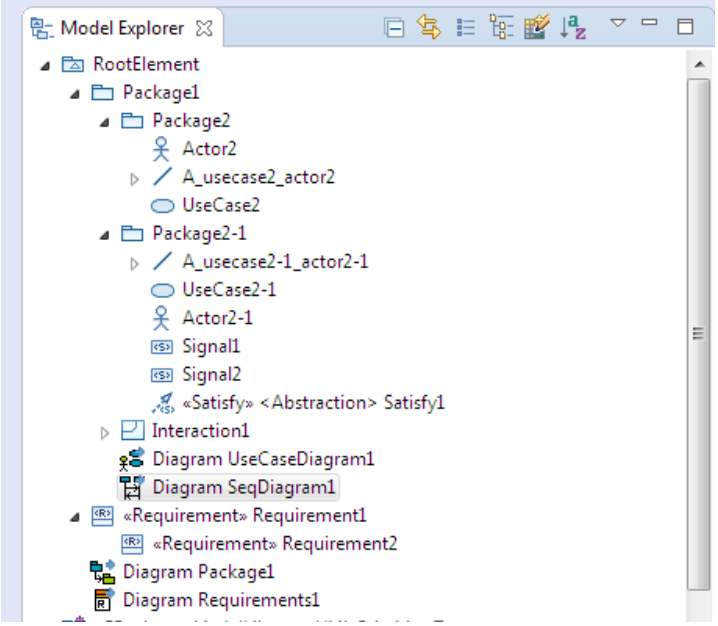

Figure 15. Model view of the requirements diagram with only Requirement1 visible

You have to "expand" Requirement1 by clicking on the right facing arrow in order to display Requirement2. This is because Requirement 2 is a "child" of requirement 1. For the sake of neatness and organization, you can now put this in its own package.

Selecting and editing items on the model explorer using the Properties View will change the property throughout the model. Similarly, if you want to delete an element entirely from a model, delete it in the Model Explorer. If you want to delete it only from a diagram, then bring up the diagram on the canvas and perform the deletion.

Workaround Note: I experienced a bug in Papyrus. The requirement would not display the text and the ID when I entered it. To solve that problem, I entered a command to display the compartments, then, because the expanded blocks was so large, I suppressed them again. However, the text remained. The procedure for hiding and displaying compartments is described in the next section.

3.5 Showing and Hiding Compartments

Many SysML model elements have “compartments” that are used to store additional properties. To save space on a diagram, Papyrus (as well as most other SysML modeling tools) do not show them by default. However you do can control the display of compartments. We’ll demonstrate this with the requirements diagram.

Select a requirement, then

Right Click>Filters>Show/Hide Compartments

The Filters menu item is about the 5th one down on the pop-up menu that displays after right clicking on the requirement.

After that mouse sequence, Papyrus displays the following

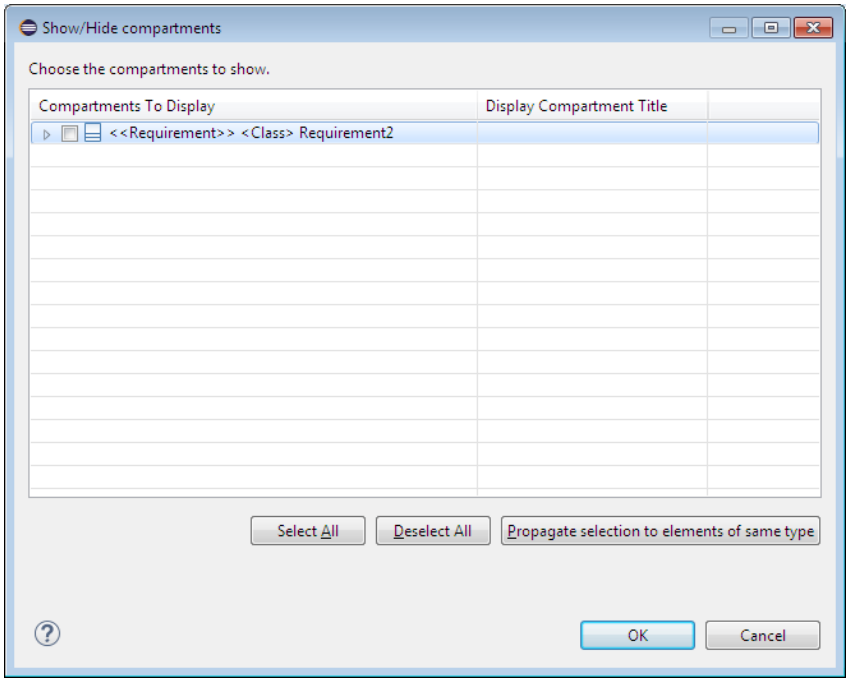

Figure 16. Show/Hide Compartments Initial Dialog Box

Select the checkbox and then click on the right facing arrow on the left to reveal all the compartments. This is how the resultant display looks

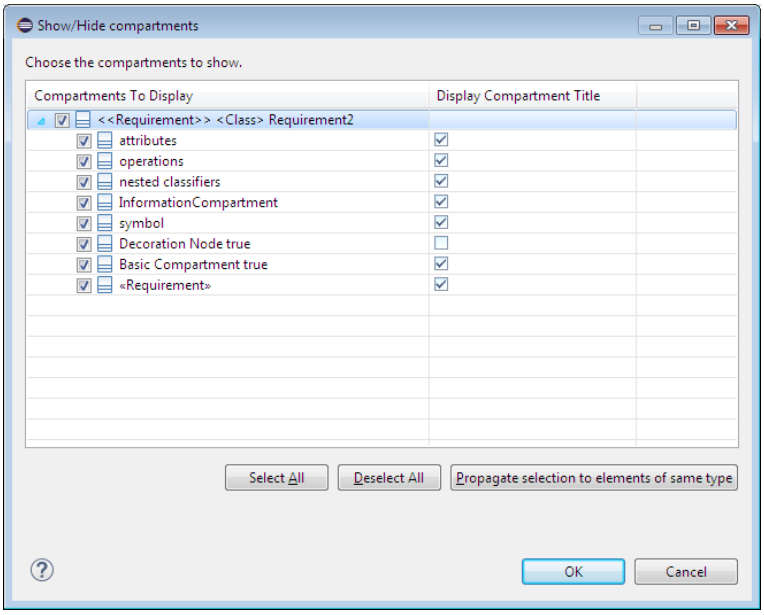

Figure 17. Compartments visible after clicking on the requirement checkbox

Select all, and then click OK. The requirement block expands (and may take up most of the canvas with all compartments displayed. Repeat the mouse sequence to get to the show/hide compartments again, and then deselect all. You should get the requirement ID and part of the text as shown above.

4 Model Export and Import

In order to enable another Papyrus user to use your model, it is necessary to export it from the workspace and for the other user to import it. The procedures are described below.

4.1 Exporting a Model

Go to the Project Explorer view. Select the project and the model within the project

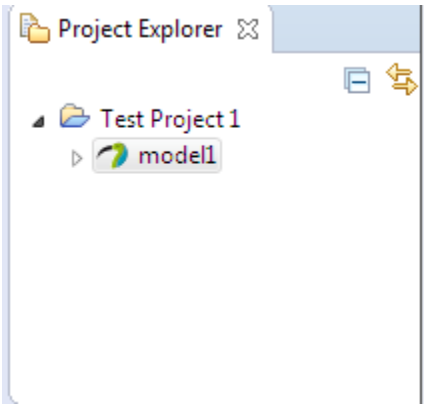

Figure 18. Project Explorer Tree containing the Papyrus model

Then, to export, execute the following mouse sequence

File>Export

and get the following dialog box.

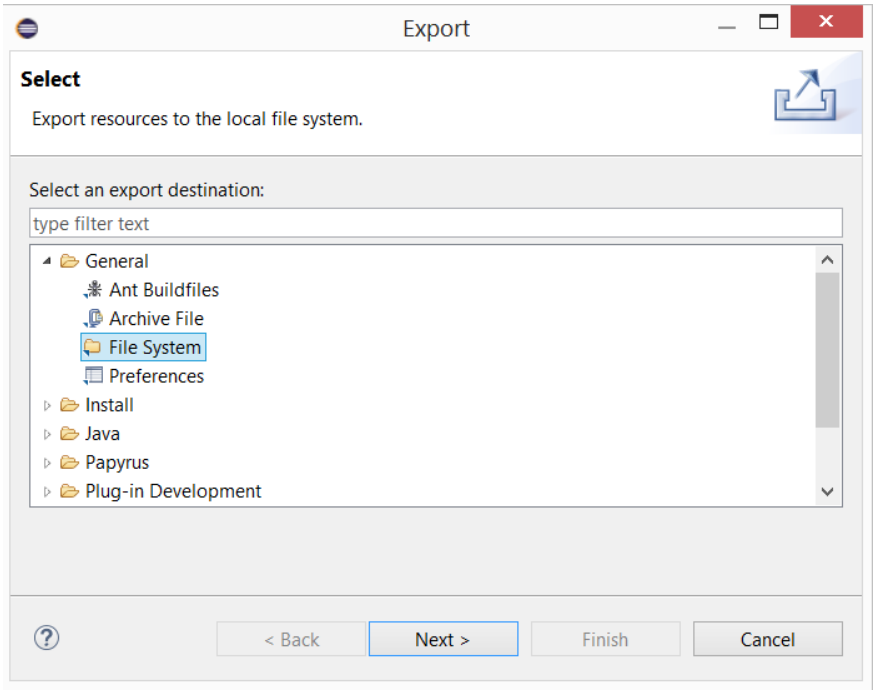

Figure 19. Dialog for Selecting File System (in Mac OS X)

Then select File System and the "Next" button

Select the "resources" (in essence, files). Select all of the items shown in the figure below. Enter the destination directory (i.e., the directory to which you want the files exported) and then click on the "Finish button,

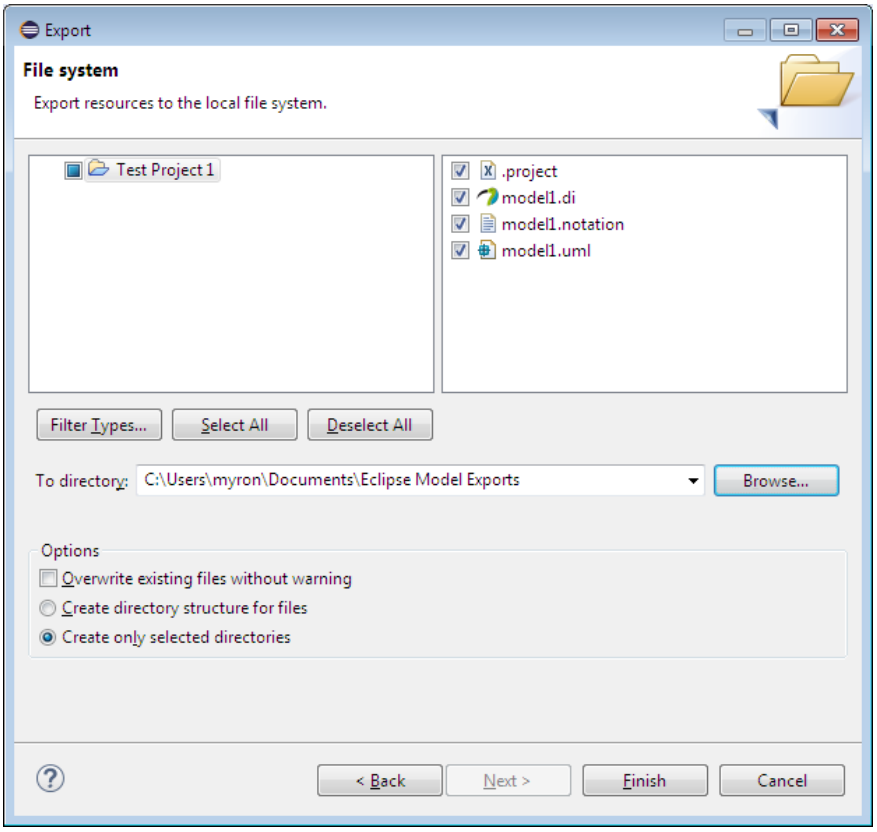

Figure 20. Dialog Box for identifying resources to export

Note: In order to get the "Finish" button, you have to enter a directory

4.2 Model Importing

To demonstrate the procedure, we are going to re-import the model we just exported, but in order not to lose our original model, create a new project if necessary using the procedure described in section 2 (e.g., Test Project 2)

Then execute the following mouse sequence

File>Import>General>File System

The following dialog box appears

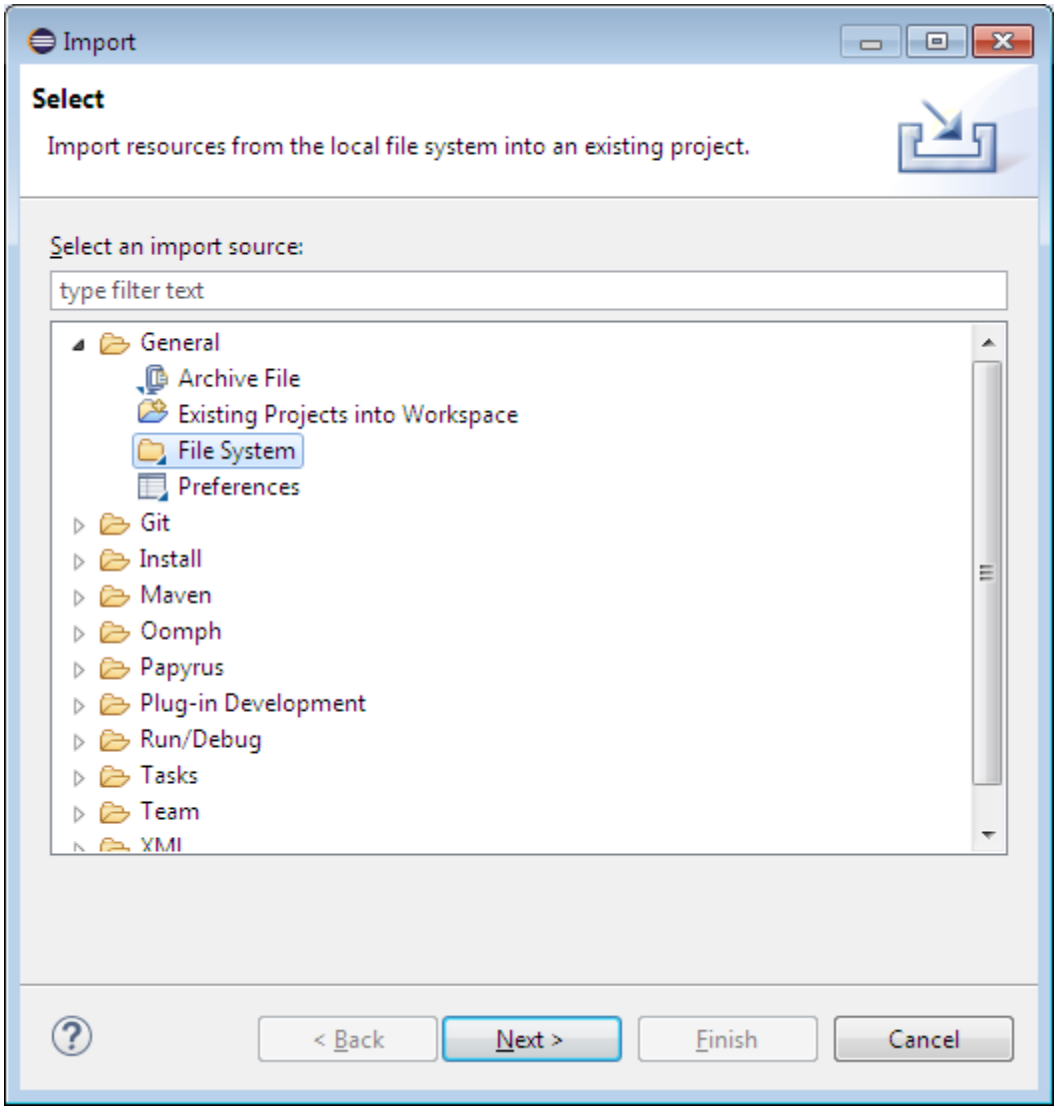

Figure 21. Import Model dialog box

You then get the following dialog box

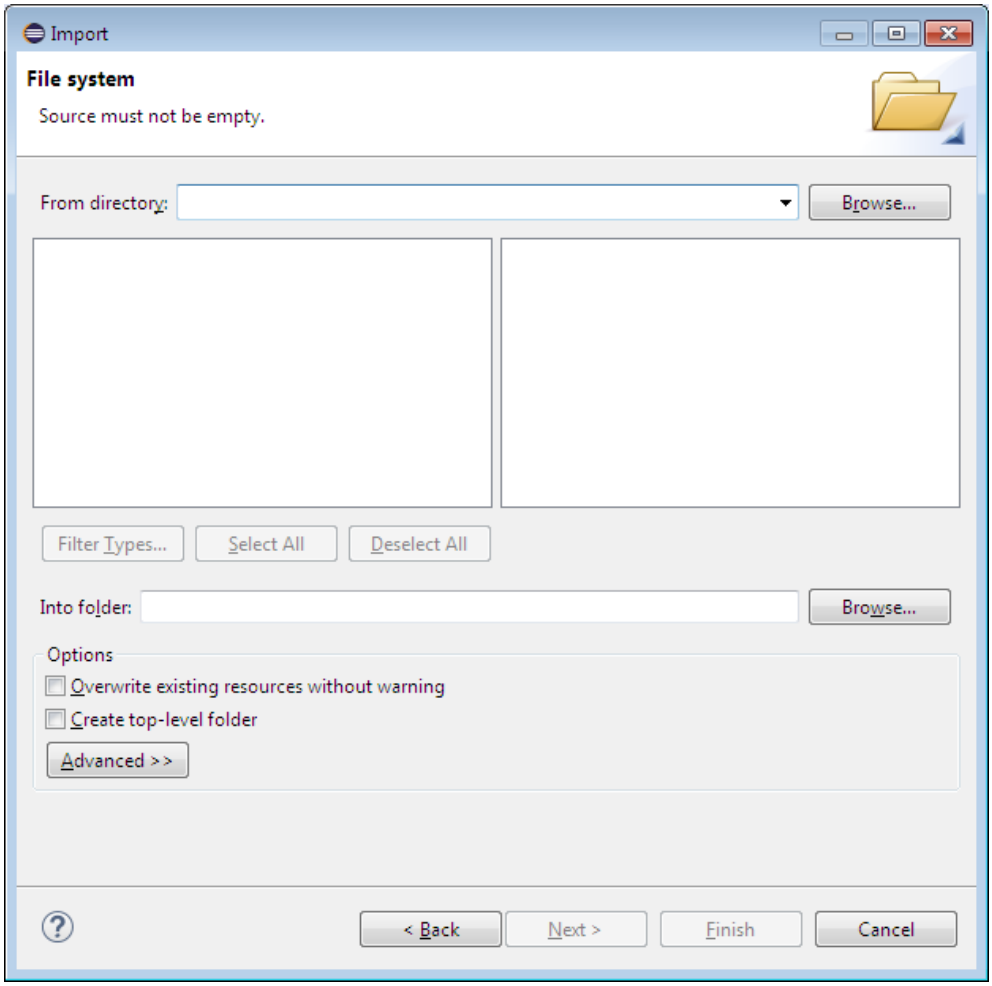

Figure 22. Dialog Box to identify directory from which to retrieve the model

Then enter the source directory (which, in our case, was the directory that we just exported, hence the name "Eclipse Model Exports") and the following four directories appear. Click the check box. Papyrus will then select all of the subdirectories. Enter the name of the destination folder (a new project in our case) and then press finish. Answer yes to the question that appears next

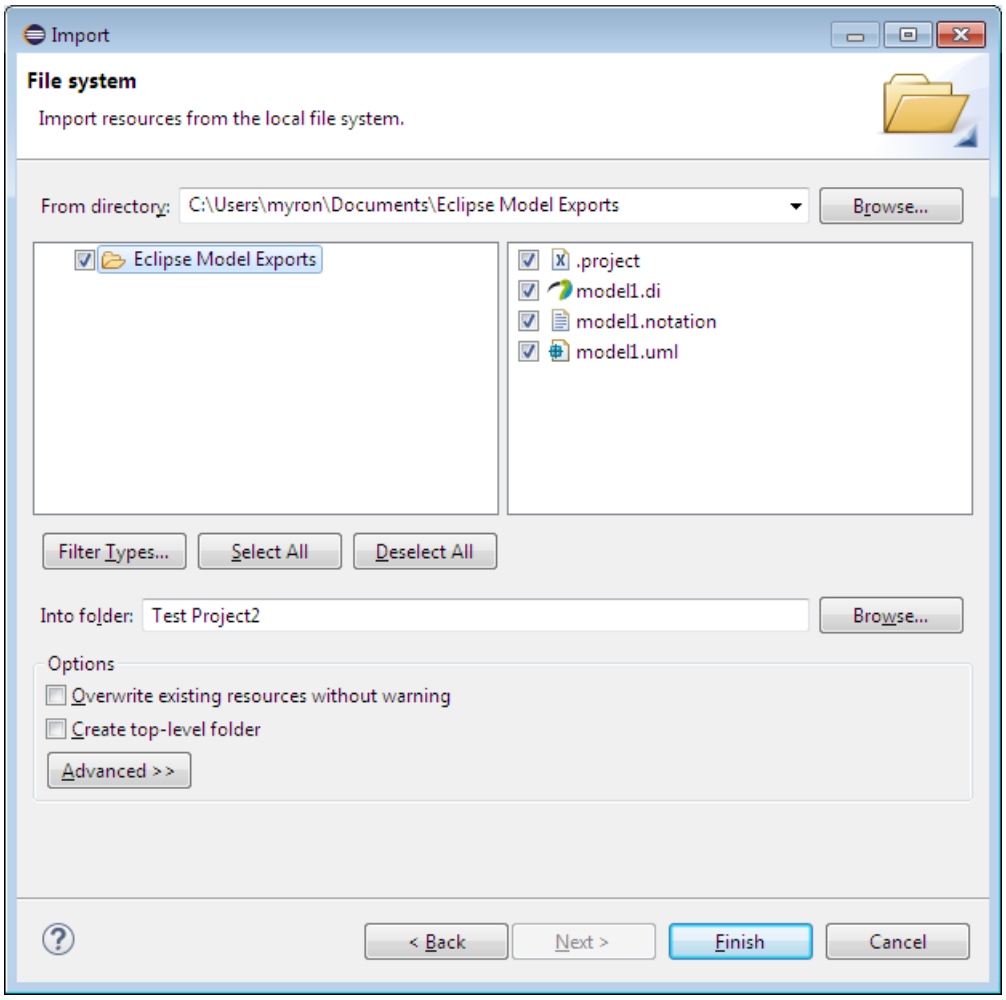

Figure 23. Dialog Box with resources to be imported identified.

The model should be imported and you should see the following into the project explorer

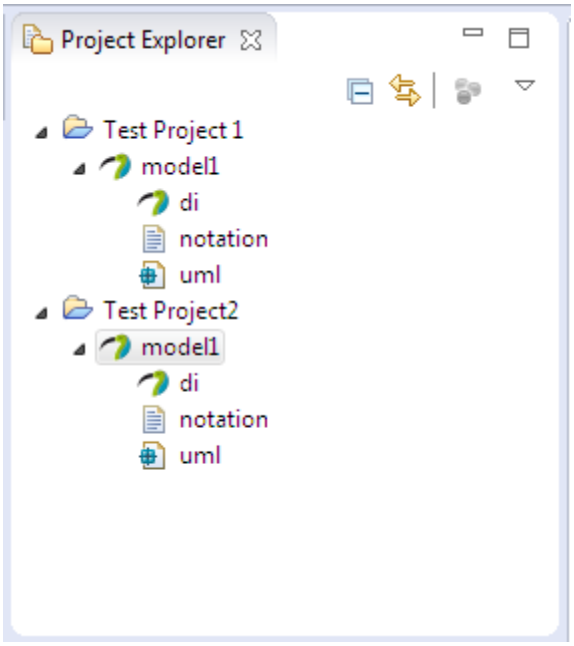

Figure 24. Project Explorer with Imported Model

Double click on the copy of model1 in Test Project2 and you should see the root element. You can then expand the root element to see the model hierarchy unfold. After you see the UseCaseDiagram1 icon, double click and it should open.

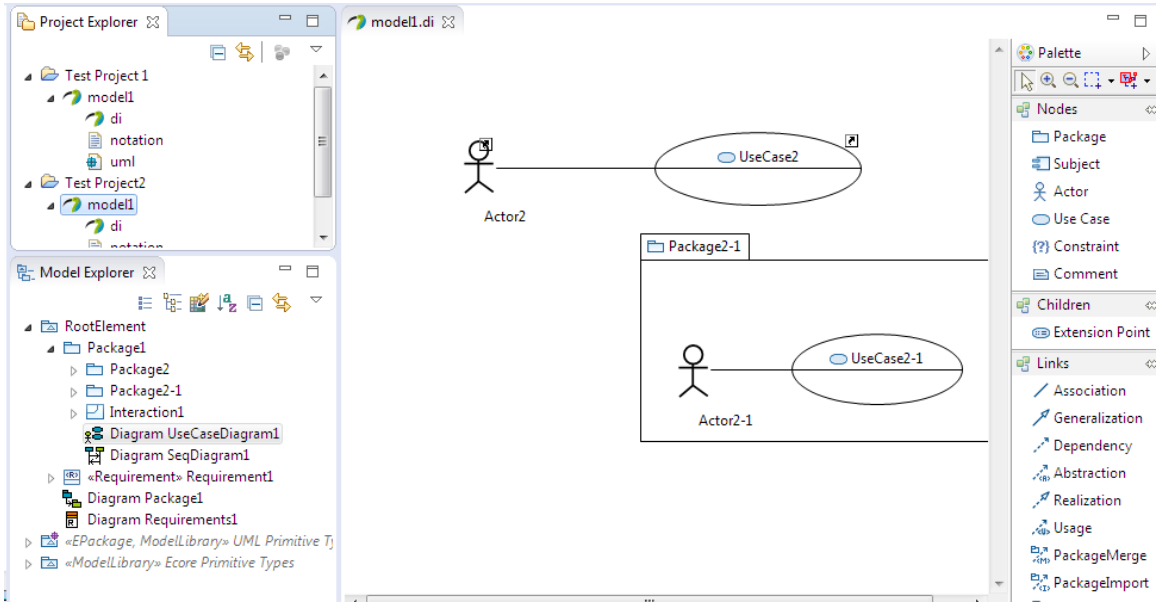

Figure 25. Use Case Diagram from imported model