Using the new features of UML Modeler in IBM Rational Software Architect Version 7.5

Skill Level: Intermediate

Wayne Diu (wdiu@ca.ibm.com)
Software Developer
IBM

26 Sep 2008

This article highlights some of the new functionality of the UML Modeler component that is common to both IBM® Rational® Software Architect Standard Edition Version 7.5 and IBM® Rational® Software Architect for WebSphere Software Version 7.5, and explains how to leverage several recently added features.

Improved UML modeling

The new functionality of the UML Modeler component is common to both the new IBM® Rational® Software Architect Standard Edition Version 7.5 and the new IBM® Rational® Software Architect for WebSphere Software Version 7.5. Throughout this article we will refer to both of these products as IBM® Rational® Software Architect Version 7.5 or as Rational Software Architect V7.5, for short.

This article highlights some of the new functionality of the UML Modeler component in Rational Software Architect Version 7.5, and explains how to leverage several recently added features.

There have been a lot of usability enhancements made to the UML Modeler component in Rational Software Architect V7.5. First, model management capabilities have been significantly enhanced, making it easier for you to work with larger models shared among a large number of users. In addition, for those who assume that modeling is difficult or hard to grasp, steps have been taken to reduce initial obstacles. Much effort has been placed in simplifying potentially complex areas, allowing you to achieve results faster.

Another point of emphasis is model integrity: that is, reducing the likelihood of ending up with a broken model. In fact, in Rational Software Architect, you have been given the ability to understand and fix potential problems in your models. The end result is that you will be able to get your work done more rapidly, and end up with more correct models. To that end, changes have been made to allow your models to achieve a higher level of compliance with the UML specification.

The New Model wizard

Users upgrading from earlier versions of Rational Software Architect will immediately notice a difference when using the New Model wizard:

- There are now two wizards for creating models, the Model wizard and the UML Model wizard.
- The set of model templates has been updated.
- Wizard pages have been added to allow you to customize new features (such as having a Package as a root element, UI capabilities, and working sets).
- The extensibility mechanism has been improved, which will make it evident why two wizards are present.

Package as a root

If you are creating a new model from the template, allowing a UML Package to be the root element of a resource is one of the most visible changes exposed by the New Model wizard. Instead of templates referring to *models*, you will see templates referring to *packages*.

To clarify the terminology: a Model is really a Package with two more properties, one of which is the viewpoint String. (The other property is whether it is a metamodel.)

This change is rather subtle, but it has some important implications. Historically, in all of the previous releases of the product, the Model UML element was always at the root of the .emx file resource. There is nothing in the UML specification that prescribes what has to be at the root of the resource: indeed, there is no concept of the resource in the UML specification.

Seeing that a Model is really just a specialization of a Package, in Rational Software Architect V7.5, there is now the option to have the Package as the root element of the resource. This explains why instead of seeing "Blank Model," you see "Blank Package."

A problem that you might encounter is with file compatibility, because earlier

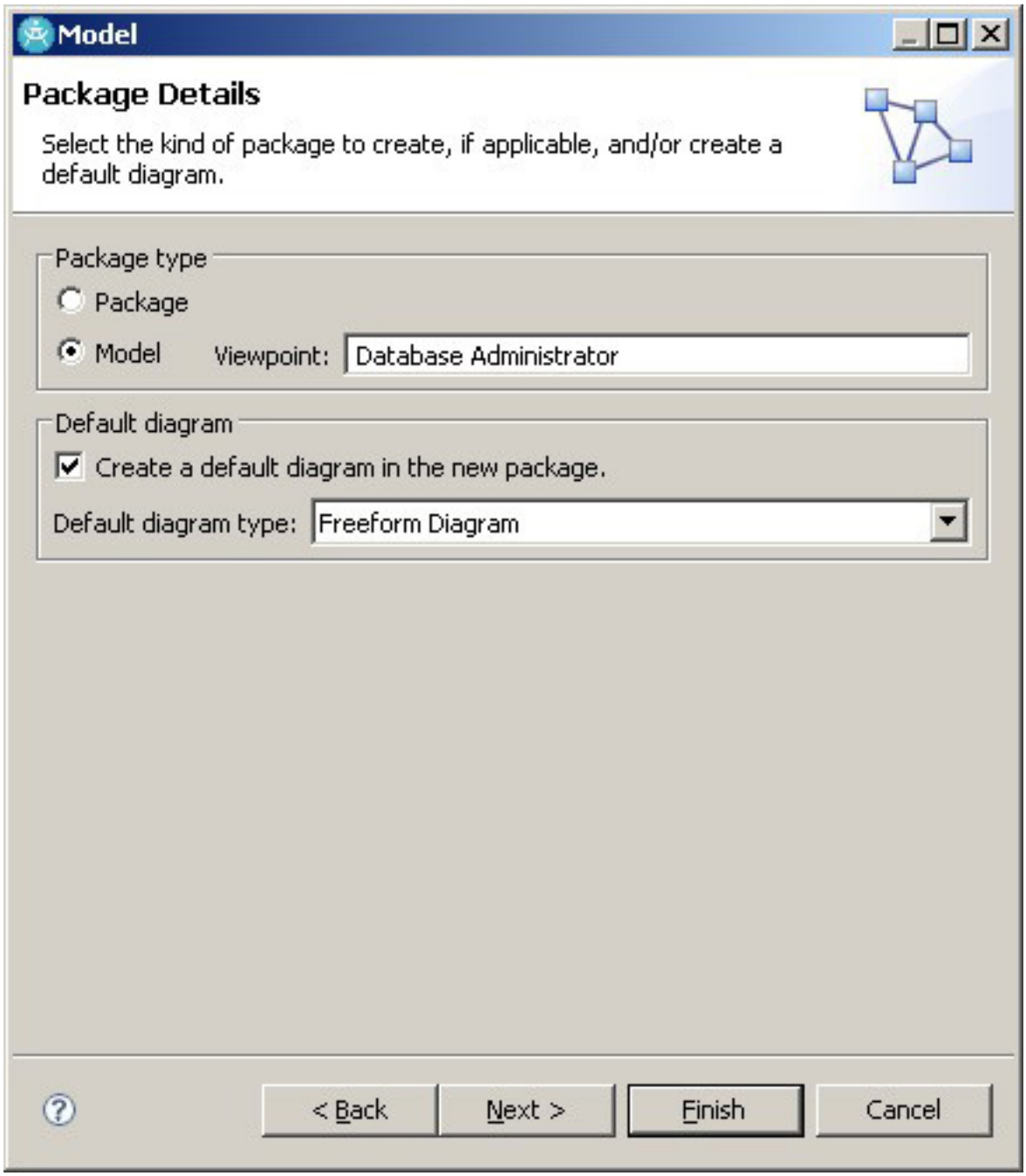
versions of the product do not support keeping the model as a root. So, how can you ensure that the file is compatible? The first way is from the ensuing wizard page.

Package Details wizard page

If you click the **Next** button after selecting a template (and if it is supported by the template) you will be presented with the Package Details page.

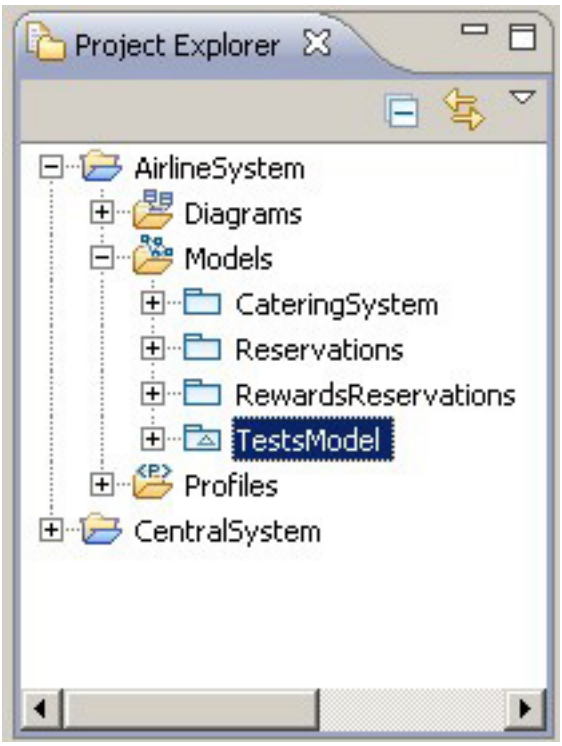
In the **Package Type** group, you are able to change the package type to **Model**, as shown in Figure 1. Then, the root of the resource will be a Model, and the file can be opened properly in previous versions. This setting does not revert. Thus, if you run through the wizard again, **Models** will be generated, not **Packages**. If you are planning on using models in a previous release of the product, choosing **Model** instead of **Package** will help prevent a backwards compatibility issue.

Figure 1. Choosing between Model and Package in the Package Type group



Tip: You can see in Figure 2 that the icon for the package (a closed folder) is slightly different than the icon for the model (a closed folder with a triangle) in the Project Explorer. This is so that you can tell them apart without having to activate the Properties view.

Figure 2. The UML icon representing a Model



If you have already created a Package and would like to convert it to a Model, you can do so easily.

1. Right-click the Package
2. Select **Refactor > Convert Package to Model**
3. Click **OK**.

Tip: You can convert a Model to a Package in the same manner. Right-click the Model, and select **Refactor > Convert Model to Package**. You are warned that you will be losing the viewpoint attribute. Recall that the viewpoint String is an additional property supported only by the package, not the model. Also, do not confuse this viewpoint String from the UML specification with the Modeling viewpoints used to hide unnecessary elements of the user interface.

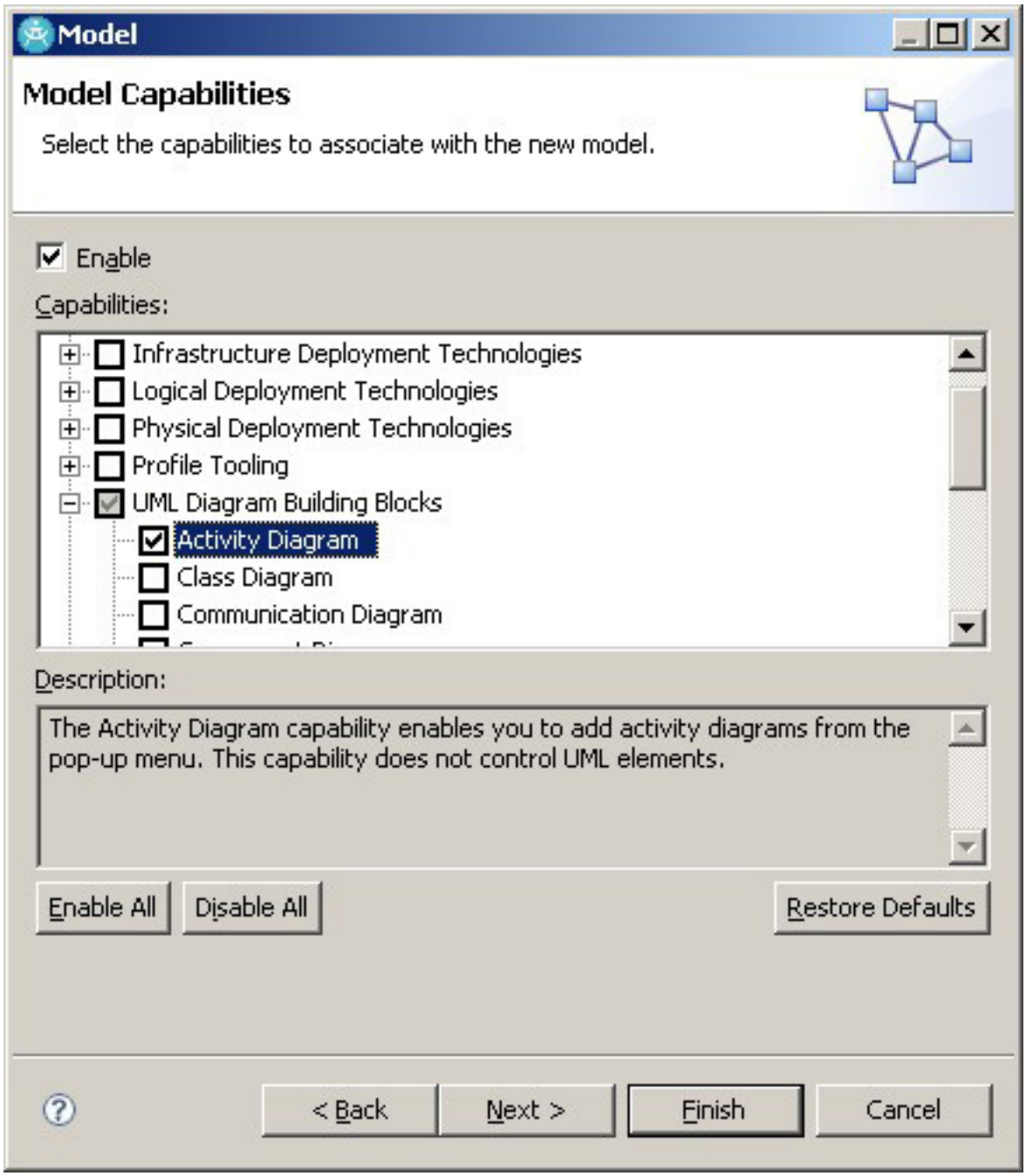
Having the root element as a package has its benefits. For example, it is possible to move one root package into another with a simple drag-and-drop operation. This is perfectly valid, because a package can contain other packages.

Model Capabilities wizard page

The Model Capabilities page allows you to hide irrelevant UI from your model. For example, suppose that you chose to create a model from the **Blank Package** template, and you will be modeling Activity diagrams exclusively. There is no need to

clutter up the menus, palettes, and diagram assistants with menu items, tools, and buttons with options to create elements that you will not be interested in. From this wizard page, you may select only the diagrams and UML elements that you would potentially use in your model.

Figure 3. Select the capabilities to associate with the new model



Tip: If you decide to change the capabilities after creating the model, simply:

1. Select the model in the Project Explorer view.

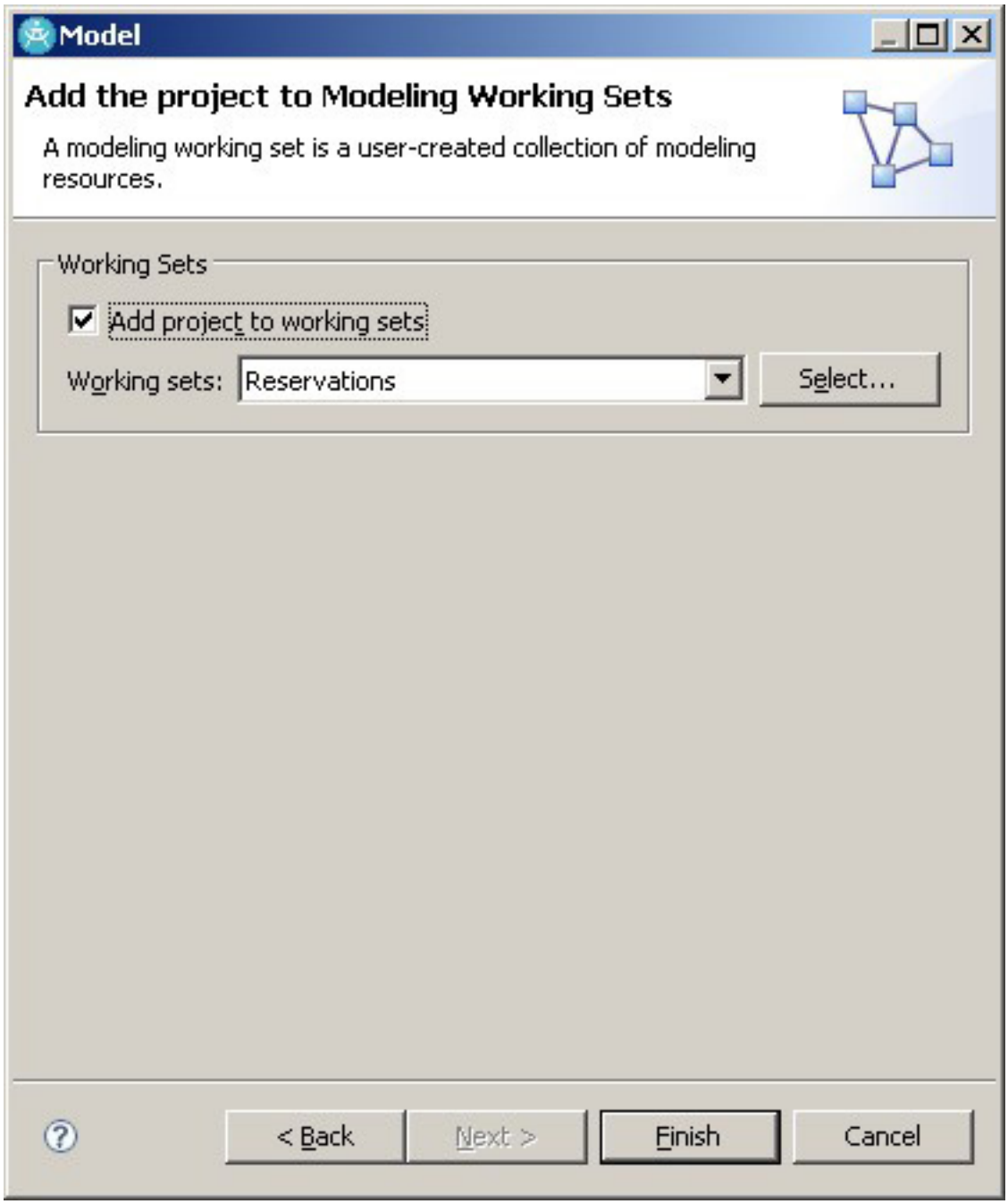
2. Then, you can modify the capabilities from the **Capabilities** tab of the Properties view.

Modeling Working Sets wizard page

A Modeling Working Set is a group of resources that consists solely of Modeling resources from your Eclipse workspace. You can think of it as a Resource Working Set that has been filtered to only include Modeling files.

The Add the project to Modeling Working Sets page allows you to add the project containing your newly created model (either from a standard template or from an existing model) into such a working set. If you do not have a working set already, you can click the **Select** button to create your own working set, as shown in Figure 4.

Figure 4. Adding a project to a Modeling Working Set



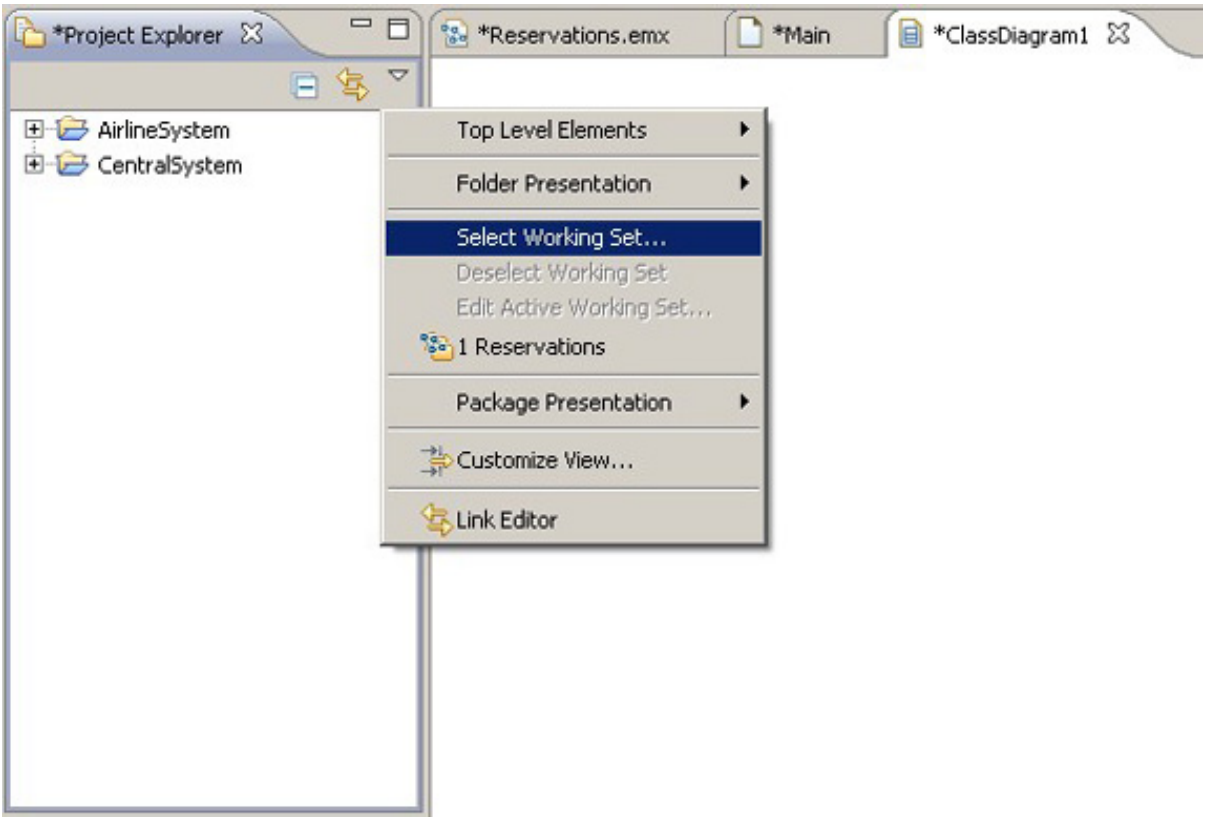
The purpose of a working set is to hide irrelevant resources from your workspace. For example, suppose that half of the time you are responsible for the Scheduling System, while the other half of the time you are responsible for the Reservations System. Both sets of resources have similar naming conventions, making it easy to confuse them. Usually, they are independent, but occasionally you would like to work with both sets at the same time.

As a solution to this problem, you could divide the resources into two working sets.

1. Then, from views such as the Project Explorer (or Package Explorer) you could press the triangle button to show the **View** menu.

2. From there, choose **Select Working Set** (as shown in Figure 5) to filter out the irrelevant resources in your workspace to avoid confusion.

Figure 5. Selecting the working set from the Project Explorer



Wizard extensibility

The extensibility mechanism of the New Model wizard has been much improved. Indeed, as a testament to its extensibility, the UML Model wizard extends the generic, non-UML specific Model wizard. This explains why there are two wizards: one is intended for models in general, while the other is intended for UML models. The UML Model wizard subclasses com.ibm.xtools.common.ui.wizards.NewModelWizard. You can do so too, to build a customized wizard.

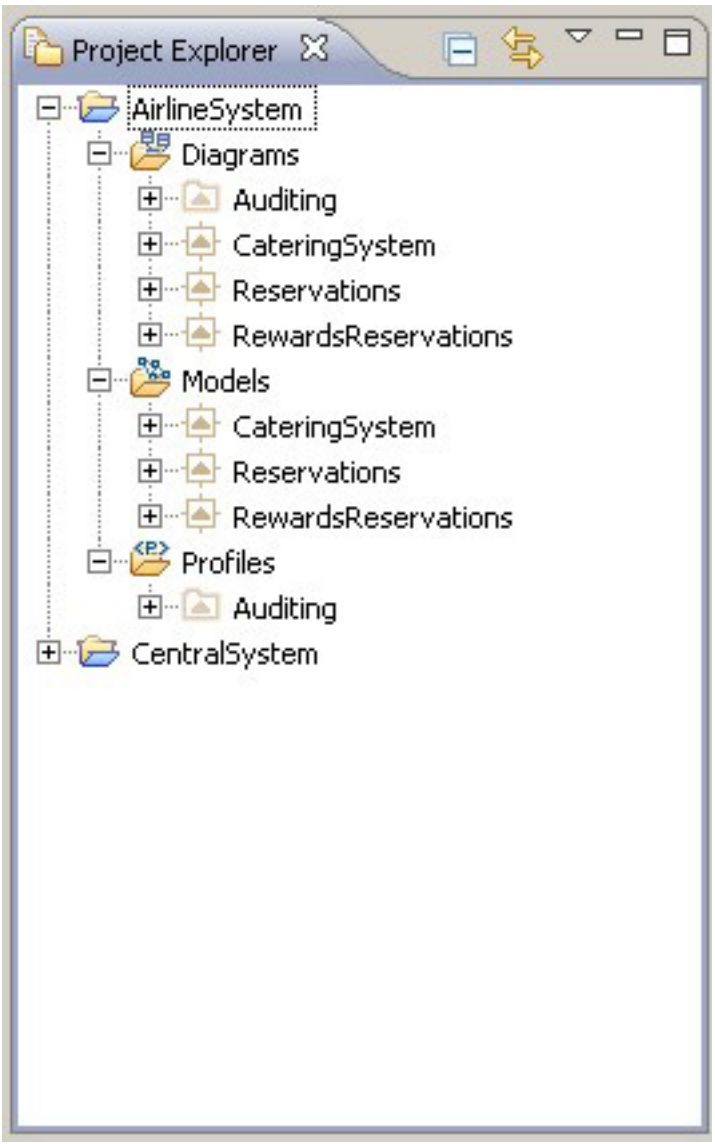
For more details on contributing the wizard through the org.eclipse.ui.newWizards extension point, please see the **Resources** section. To learn more on contributing templates, please see the product's documentation (**Extending product function > Extending IBM Rational Software Architect Software function > Extending the Rational modeling environment > Rational Modeling Platform Developer Guide**

> Programmer's Guide > Customizing the user interface > Adding Templates to the New Model Wizard).

Reducing model load time

The standard way of opening a model is by-double clicking it in the **Project Explorer** view. In Rational Software Architect V7.5, you will notice a plus sign next to each model, as shown in Figure 6. If you click the plus sign next to a closed model, the model will open without opening the main diagram or the model editor. Typically, the main diagram, which opens when the model is opened, is the diagram with the most references. Therefore, rendering that is what usually triggers referenced models to load. Clicking the plus sign instead of double clicking the model allows you to browse the model without opening the diagram, and thus potentially triggering other models to load. This can be a real time saver.

Figure 6. Plus signs appear next to closed models in the Project Explorer



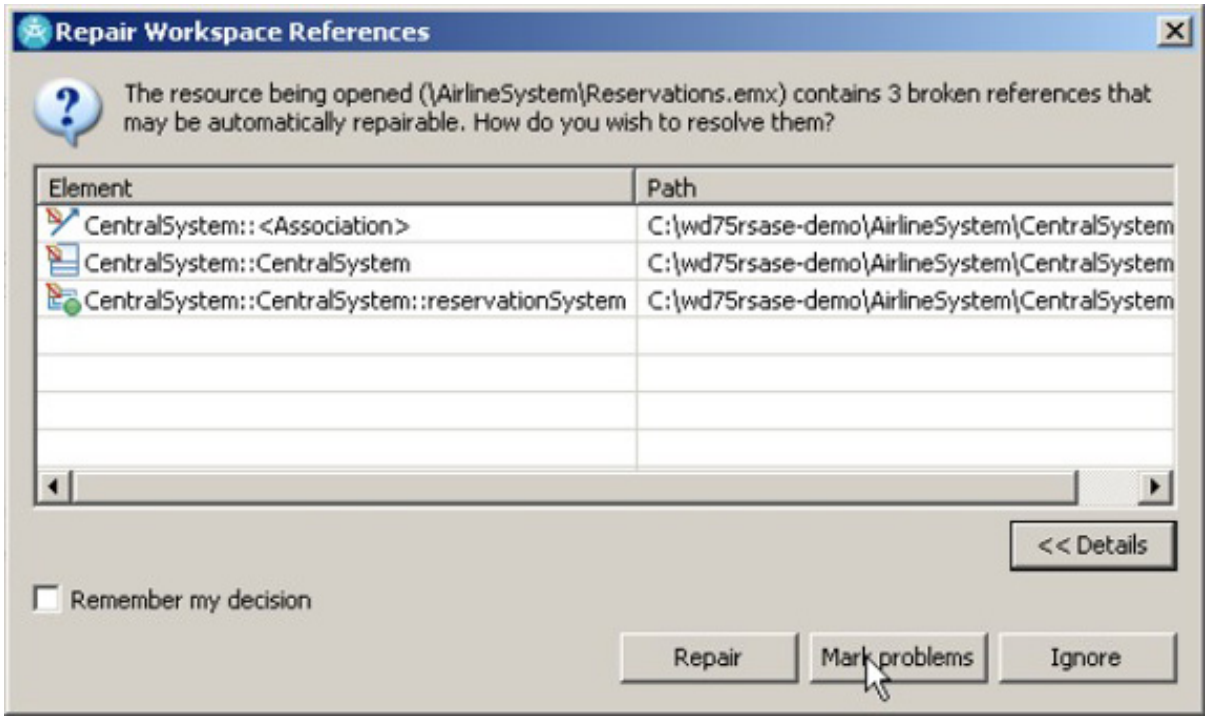
Tip: You can also achieve this using the Open Model menu item, which is available when you right-click a selected model in the Project Explorer. The Open Model menu item will not open the model editor or the main diagram.Model integrity

Have you ever opened a model that opened properly just a little while ago, only to find that now when you open it, it contains broken references? In Rational Software Architect, there is a new dialog informing you of broken references. This dialog appears whenever you open a model containing one or more broken references (for example, through a user gesture such as clicking the plus sign, double-clicking the model, or even indirectly by opening some other referencing model).

Pressing the **Details** button shows you a table of the broken references, as shown in Figure 7. There are three buttons at the bottom:

- The first is **Repair**: This repairs the references immediately subject to your workspace preferences, which will be explained in further detail. Repair will only search for elements within the model currently being loaded. Problem markers will be created for elements which cannot currently be resolved.
- The one in the middle is **Mark problems**, which adds problem markers to the Problems view for all the broken references. In addition, the model elements will be marked with a red X to indicate the location of the problem.
- The one on the right is **Ignore**. This leaves the broken references alone.

Figure 7.The Repair Workspace References dialog



Being able to mark the problems is not very useful without being able to fix them. Fortunately, you have the ability to do just that. If you choose the **Quick Fix** menu item, a dialog with two choices displays. Choosing the **Automatic Workspace Reference Fixup** choice scans all of the resources in your workspace (including the closed resources). It then attempts to fix all of the broken references by locating the element inside one of the resources. The matching algorithm is based on element ID. Alternatively, choosing the **Change resource path reference** choice will only fix the currently selected broken reference.

Two ways to fix marked problems

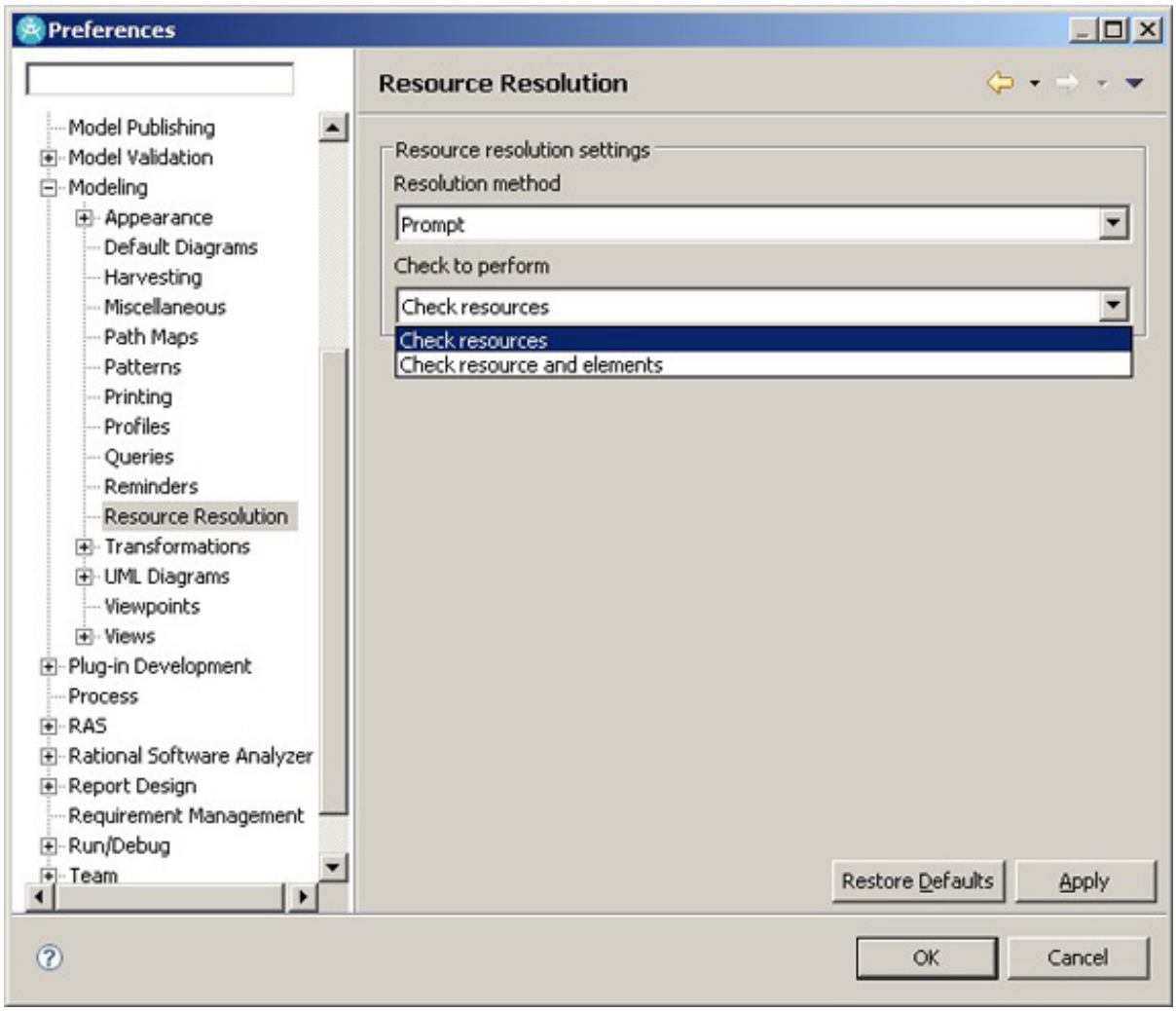
The difference between **Repair** in the Repair Workspace References dialog and **Automatic Workspace Reference Fixup** in the Quick Fix dialog is as follows. Repair will only search the

immediate model to find the broken referenced element. For example, consider a case when your team is working on a model that contains two fragmented packages, and the first fragmented package contains a class. Also, a diagram at the root level of your model contains a reference to that class. If someone else from your team moves the class from the first fragmented package to the second fragmented package, but subsequently forgets to check in the main model, the diagram's reference to that class will be broken.

Repair from the Repair Workspace References dialog will fix this scenario, assuming that you have set the **Check to perform** option as **Check resource and elements**. However, if someone were to move the class to an entirely different model, you would need to invoke the **Automatic Workspace Reference Fixup** Quick Fix in order to find and fix the broken references.

Now, look at the preferences. It is possible to change the reference resolution settings to suit your workflow. You can do this from the Preferences dialog, in the **Modeling**, **Resource Resolution** category, as shown in Figure 8.

Figure 8. The Resource Resolution preference page.



These are your choices for the first group of options (**Resolution Method**):

- **None**: Does not take any action when a broken reference is encountered. Equivalent to the **Ignore** button in the dialog.
- **Prompt** (default): displays the **Repair Workspace References** dialog, allowing you to preview the broken reference and then decide what to do.
- **Mark broken references**: Automatically creates problem markers when broken references are encountered without prompting. Equivalent to the **Mark problems** button in the dialog.
- **Repair broken references**: Automatically repairs the broken references without prompting. Equivalent to the **Repair** button in the dialog.

The second group of options (**Check to Perform**) requires some more explanation:

- **Check resources** (default): The reference is deemed to be unbroken as long as the resource exists, even if the referenced element no longer exists inside the resource.

- **Check resources and elements**: The reference is deemed to be unbroken if the resource exists **and** the referenced element exists inside the resource.

Tip: To preserve the behavior of the previous release, the Check resources option was chosen as the default. However, Check resources and elements performs a more thorough job (although it takes a little more time). If you need to fix up references after having moved numerous elements between resources, it is strongly suggested that you select the Check resources and elements option.

Profile integrity

Apart from opening models with broken intra-model references, you could open a model with a missing applied profile. Figure 9 illustrates the situation after opening a model (`CateringSystem`) where the profile applied to that model (`PilotOrderMaker`) is missing.

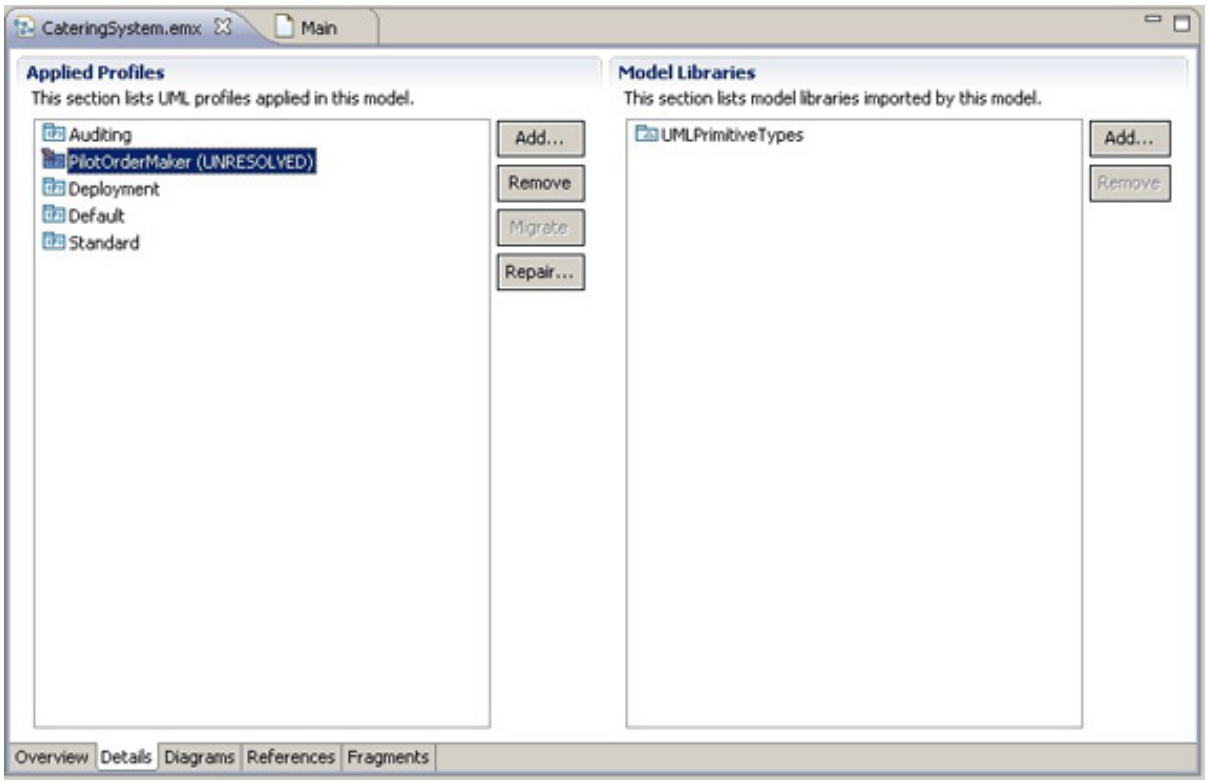
1. First, open the Profile Editor and click the **Details** tab.

2. Then select the unresolved applied profile. Both the **Repair** and **Remove** buttons will be enabled.

In Rational Software Architect V7.5, there are two actions that you may now take to help you resolve this problem. Clicking **Repair** allows you to reselect the profile to be applied. This is useful if the location to the profile has been moved. For example, this may happen if the profile used to be a file in your workspace, but now it is being deployed in your environment as a plug-in.

However, if a completely useless profile is applied (does a pilot even make catering orders?), clicking the **Remove** button will be more appropriate. **Remove** will unapply the profile application, as well as all of the applied stereotypes from the profile being unapplied. There is no way to recover this information once it is removed, so you should be careful when you use it. In cases where the profile is obviously meaningless, it is quite a handy tool to use.

Figure 9. The Profile Editor shows unresolved profiles



Activity Diagram

New developments in Activity diagrams in Rational Software Architect V7.5 include:

- Automatic synchronization of Call Behavior Actions and Call Operation Actions to the respective behavior or operation
- Enhancements to layout
- A new, unified flow tool
- The ability to control the hiding and showing of pins

Call Behavior Actions and Call Operation Actions

Starting with IBM® Rational Software Modeler V7.0.5.1 and IBM® Rational Software Developer V7.0.5.1 and new to Rational Software Architect 7.5, Call Behavior Actions and Call Operation Actions are no longer given names by default. When a Call Behavior Action or Call Operation Action is unnamed, the name of the referenced behavior or operation is used instead. Of course, if you choose to specify a name, that name is displayed.

Note: If you choose to display aliases on the diagram, and the Call Behavior Action or Call Operation Action is unnamed, the alias of the referenced behavior or

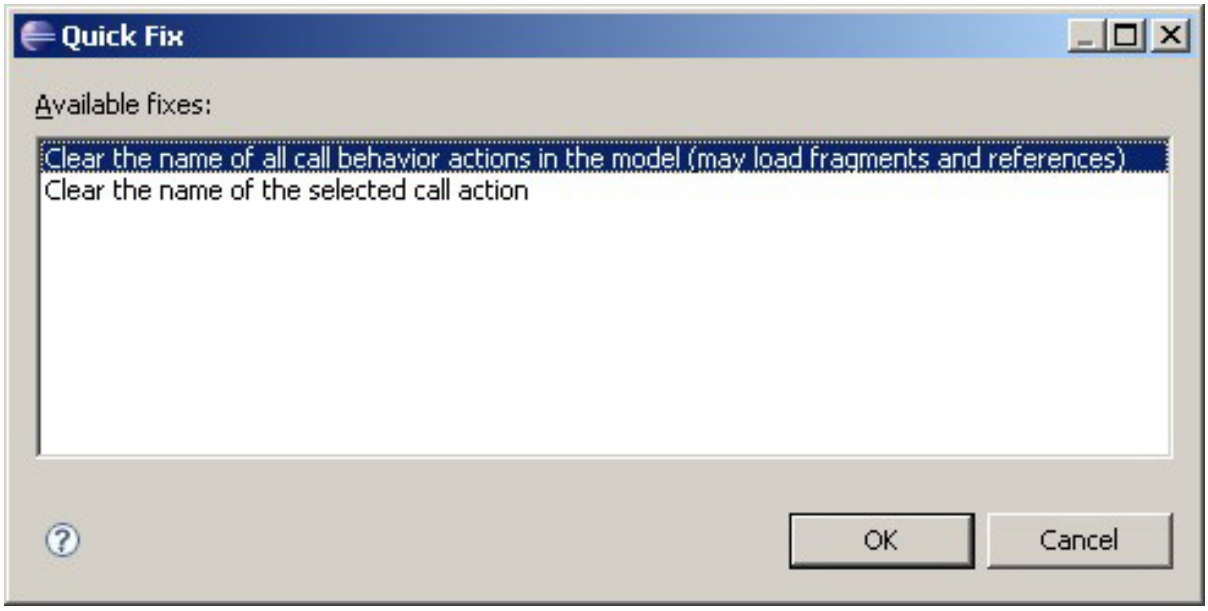
operation is displayed (regardless of whether or not the alias for the Call Behavior Action or Call Operation Action is set). Should the alias of the referenced behavior or operation be blank, the name of the referenced behavior or operation will be displayed.

Because Call Behavior Actions and Call Operation Actions created in previous versions of the product were given names by default, you may want to clear the names of those actions in the model to take advantage of the new functionality. To clear the names of all the Call Behavior Actions quickly:

1. Activate the **Tasks** view and select a task titled "It is suggested that the call behavior action "[*name*]" has a blank name to show the name of the referenced [behavior / operation]."
2. Right-click and choose **Quick Fix**.
3. Next, choose the option to **Clear the name of all call behavior actions in the model**, as shown in Figure 10.

These steps can be repeated for Call Operation Actions, substituting *Behavior* for *Operation* where necessary.

Figure 10. Options in the Quick Fix dialog to clear the name of Call Behavior Actions easily

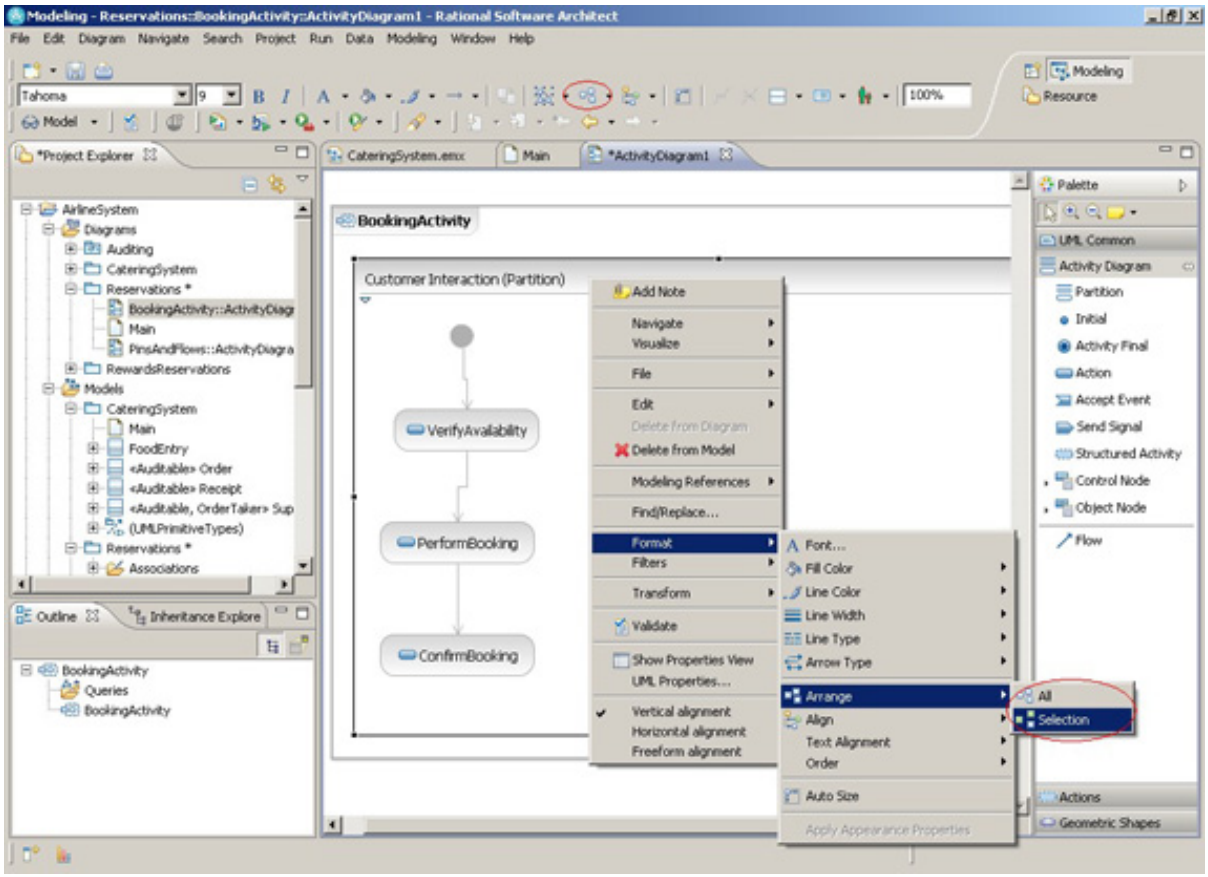


Layout

The **Arrange All** and **Arrange Selection** behavior in activity diagrams has been enhanced. In particular, elements do not overlap partitions. Also, it is now possible to arrange elements inside a partition easily. **Arrange Selection** is now enabled when

a partition is selected: it results in arranging the elements within the currently selected partition, as shown in Figure 11. On the other hand, **Arrange All** arranges the entire activity diagram independently of the selected object.

. Figure 11. Arrange can be performed via the diagram toolbar as well as through a menu item

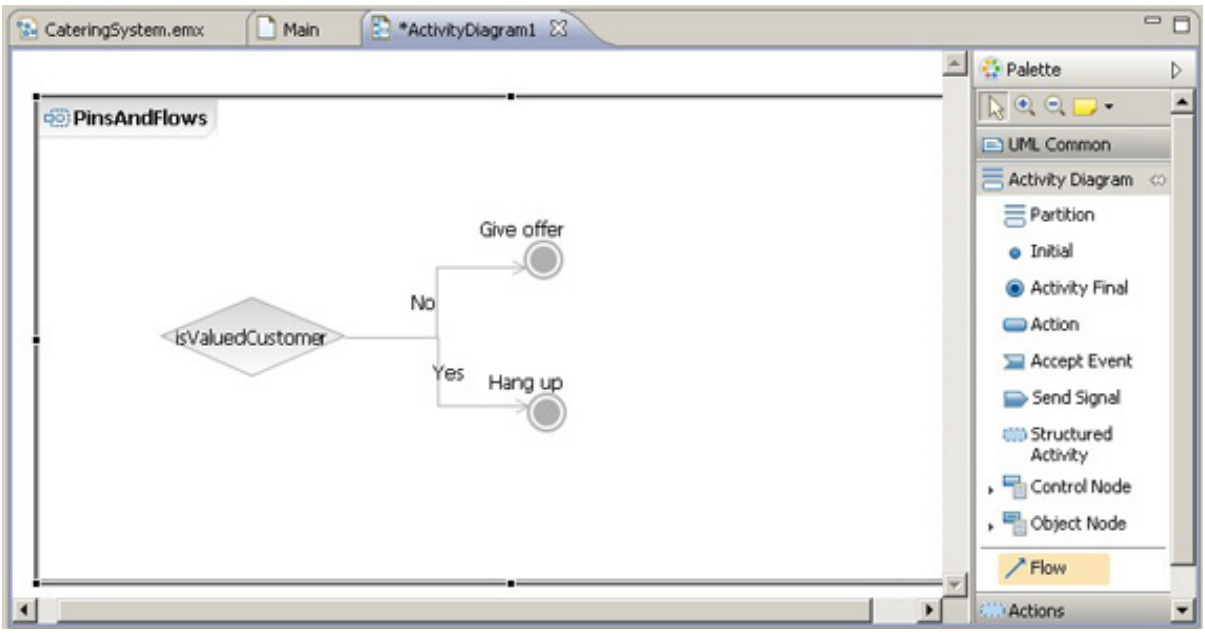


Unified Flow tool

If you worked with Activity diagrams using a previous version of Rational Software Architect, you probably noticed two flow tools in the palette: **Object Flow** and **Control Flow**. Now, in Rational Software Architect V7.5, the palette has been simplified to contain only one tool. You no longer have to worry about picking the correct one, as the tool is smart enough to determine which one you want based on UML rules and common sense.

The **Flow** tool is now even smart enough to be able to change between Control flow and Object flow when necessary. For example, it is obvious that the `Yes` and `No` flows between the `isvaluedcustomer` Decision control node and the `Hang up` and `Give offer` Activity Final nodes must be Control flows, as shown in Figure 12. However, suppose for some reason that you were to add a Central Buffer. If you then reoriented the `No` end to the Central Buffer, the Control flow will automatically change to an Object flow.

Figure 12. Modeling with the Flow tool.

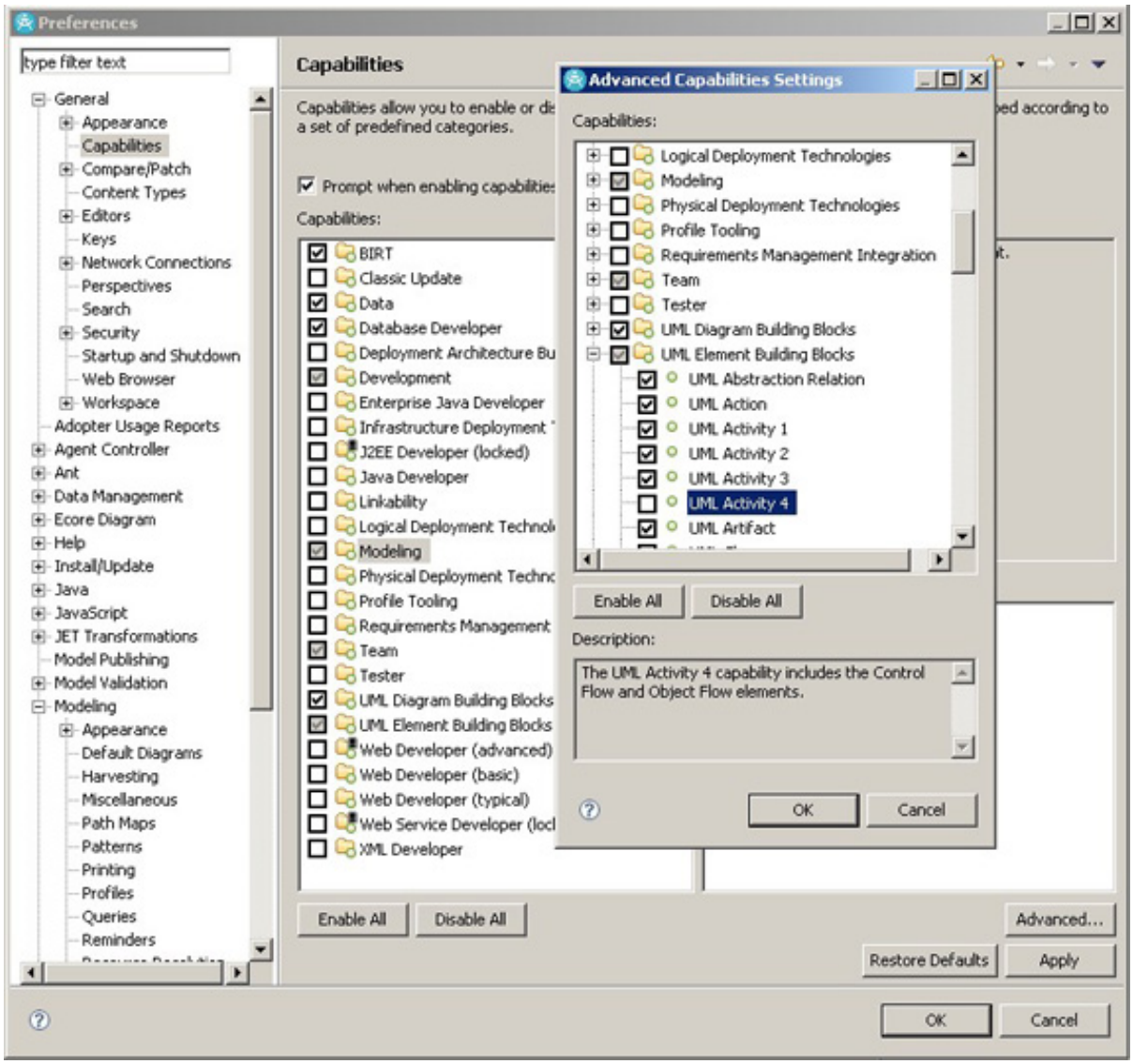


Of course, it is possible that you are very particular about which flow tool will be used.

1. Should this be the case, you still have the option of reenabling the previous two flow tools from the **Preferences** dialog in **General > Capabilities**.

2. Click the **Advanced** button and select the **UML Activity 4** option to enable it, as shown in Figure 13.

Figure 13. Enabling the capability to display both the Object Flow and Control Flow tools.

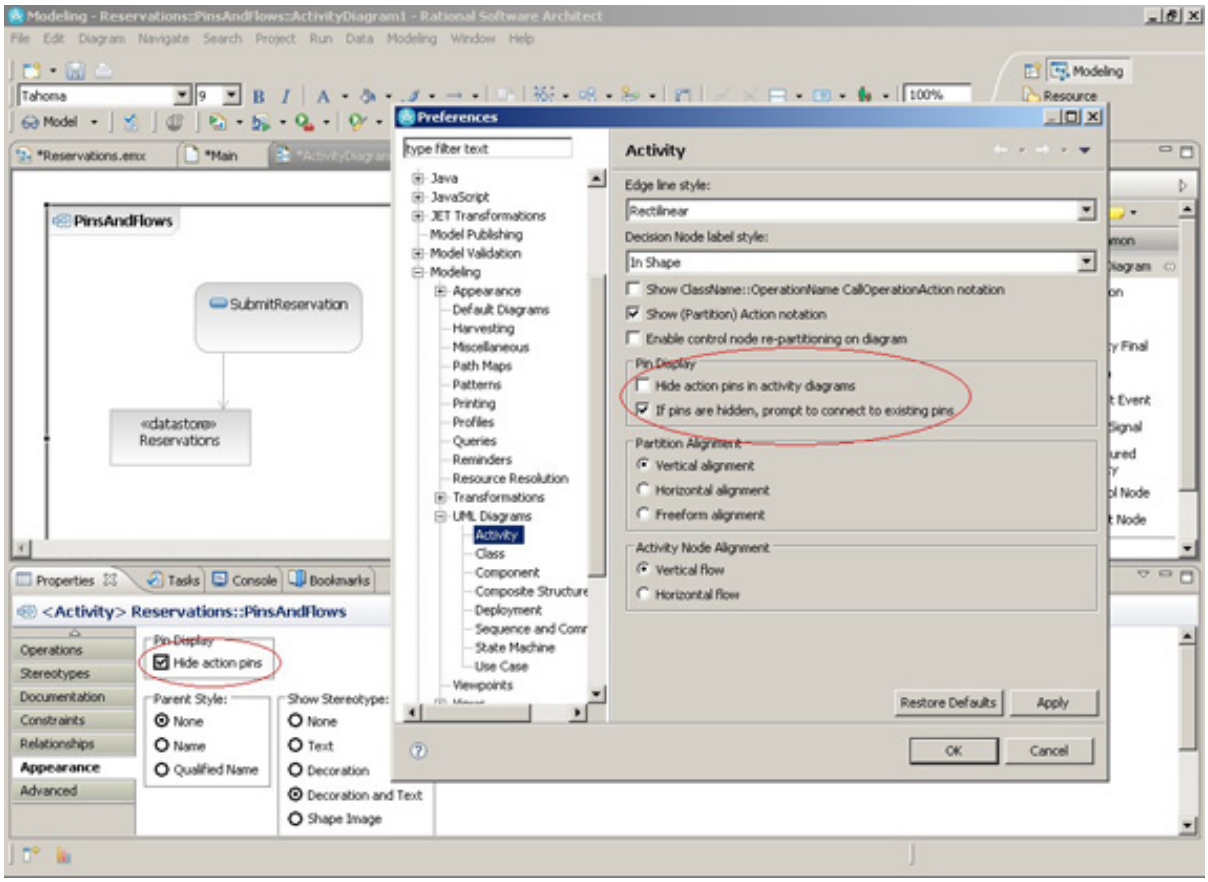


Showing and hiding pins

Some people find pins very distracting and (even though they are semantically necessary) want them hidden from the Activity diagram.

1. Click the diagram surface, then on the **Appearance** property tab.
2. Select the **Hide action pins** check box. The pins will now become hidden.
3. You can also choose to hide pins for all diagrams from the Preferences dialog in **Modeling > UML Diagrams > Activity > Pin Display**, as shown in Figure 14.

Figure 14. Showing and hiding pins in the Preferences dialog



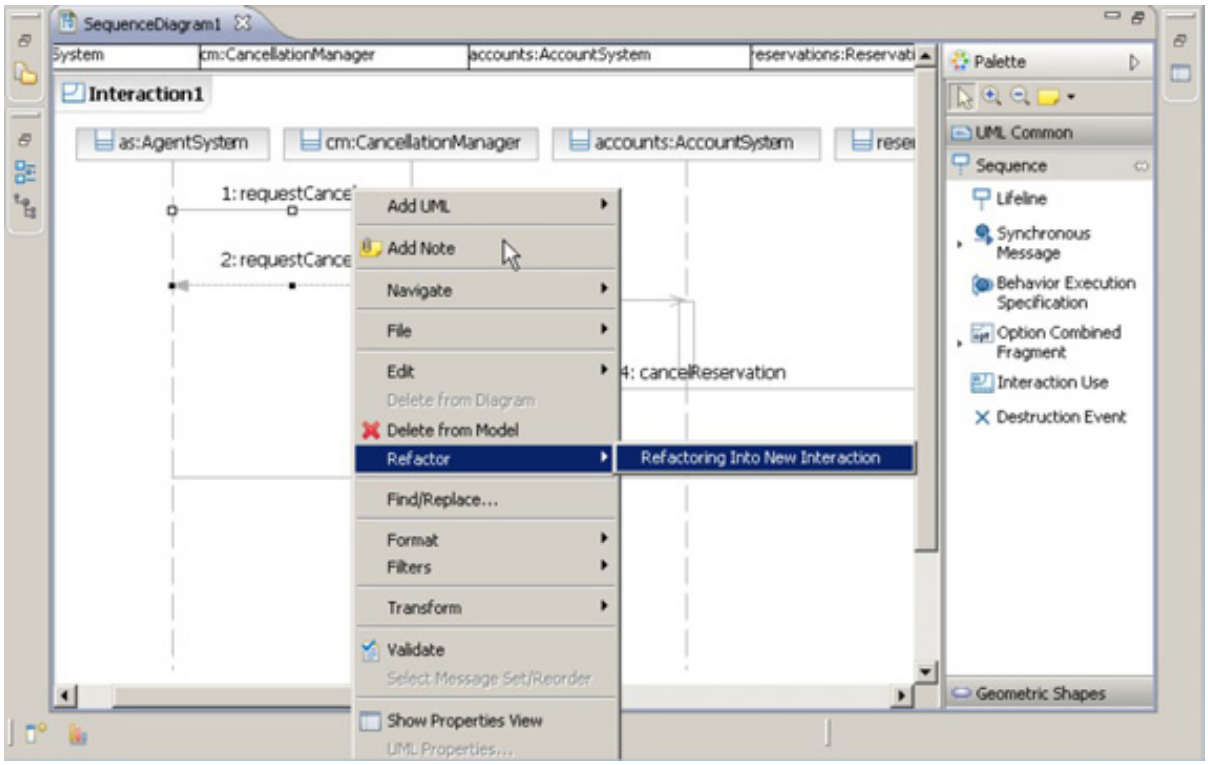
Sequence diagrams

You can now refactor portions of a sequence diagram into an InteractionUse. In addition, refactoring an operation will update the corresponding message in a Sequence diagram, even when the Sequence diagram is in a closed model. Also, Sequence diagrams now support Formal and Actual Gates for better compliance with the UML specification, and copy and paste support has been much improved.

Sequence diagram refactoring

Sometimes, Sequence diagrams contain a lot of detail that you are just not interested in. You can refactor a section of the diagram into an InteractionUse. In the diagram shown in Figure 15, the `AgentSystem` sends a cancel request to the `CancellationManager`. This portion can be refactored into an InteractionUse. To do so, right-click and select **Refactor > Refactoring Into New Interaction**. The ability to perform this refactoring is new to Rational Software Architect V7.5.

Figure 15. Refactoring a section of the Sequence diagram into an InteractionUse.

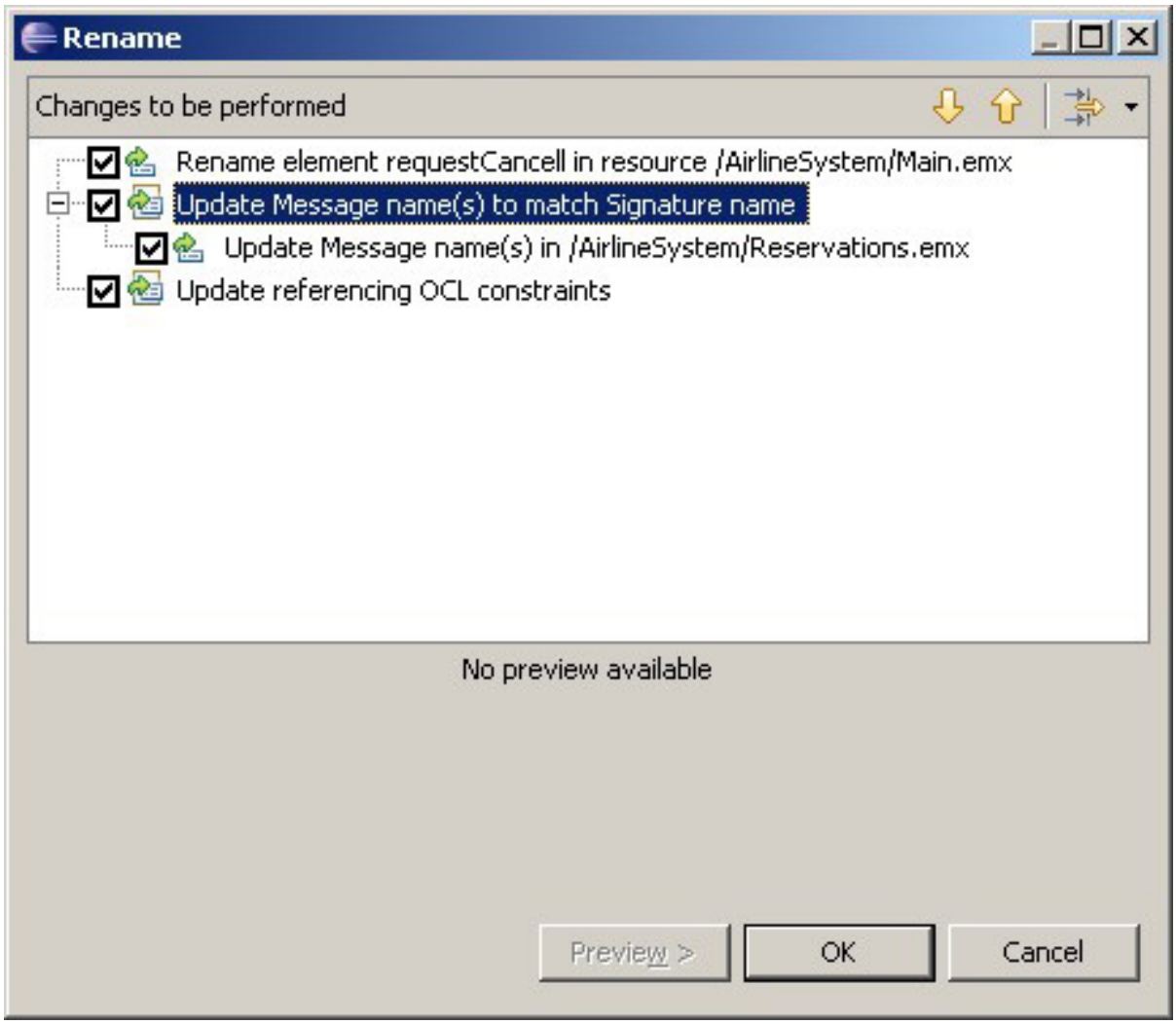


Double-clicking the InteractionUse opens the new Sequence diagram that results from the refactoring.

In addition, if a Sequence diagram message references an operation, and the operation is subsequently refactored, the message name will be updated automatically. The update will occur even when the Sequence diagram is in a closed model. This functionality is new to Rational Software Architect V7.5, but is available in IBM® Rational® Software Modeler Version 7.0.5.1 (and later) and IBM® Rational® Software Developer Version 7.0.5.1.

Tip: If you want the closed models to be considered, you need to perform a refactoring, not a simple rename (as shown in Figure 16). Be sure to choose **Refactor > Rename** from the context menu of the operation in the Project Explorer, instead of simply renaming the operation (for example, from the diagram surface or from the **Properties** view). Selecting **Refactor > Rename** on an operation will update the message names, even when the Sequence diagram containing the message is in a closed model.

Figure 16. Fixing a misspelled operation name



Gate support

Gate support has been added to Rational Software Modeler V7.0.5.1 and Rational Software Developer V7.0.5.1, but is new to Rational Software Architect V7.5.

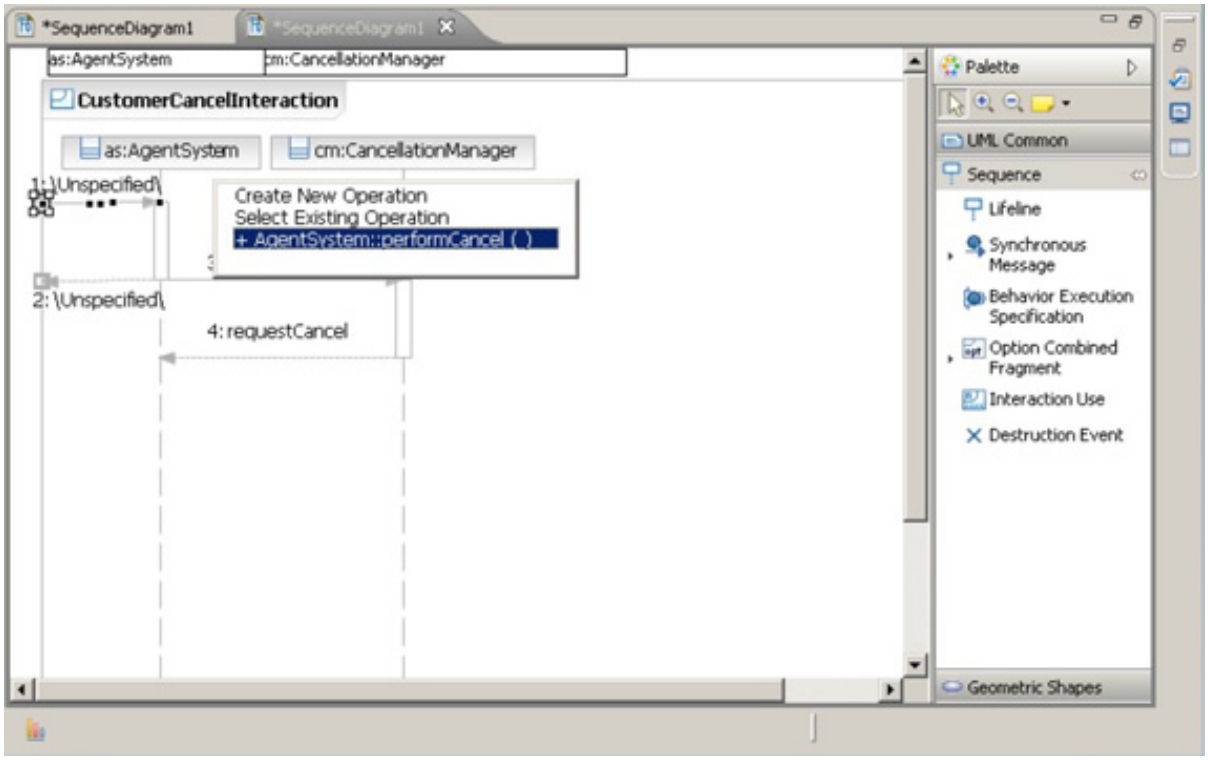
1. To add a Formal Gate on the interaction frame, click a message palette creation tool.
2. Next, click the diagram frame and drag it to the lifeline.

In the example shown in Figure 17, a synchronous message has been created from the diagram frame to the lifeline, and the existing operation, `performCancel`, has been selected. The messages are added to the bottom of the lifeline. The simplest way to move them into the proper position is by message reordering.

1. To do this, press the Alt key while selecting the message in Windows (or select the messages, right-click and choose **Select Message Set > Reorder**).

2. Then drag the `requestCancel` into the `performCancel` block.

Figure 17. Modeling a Formal Gate.

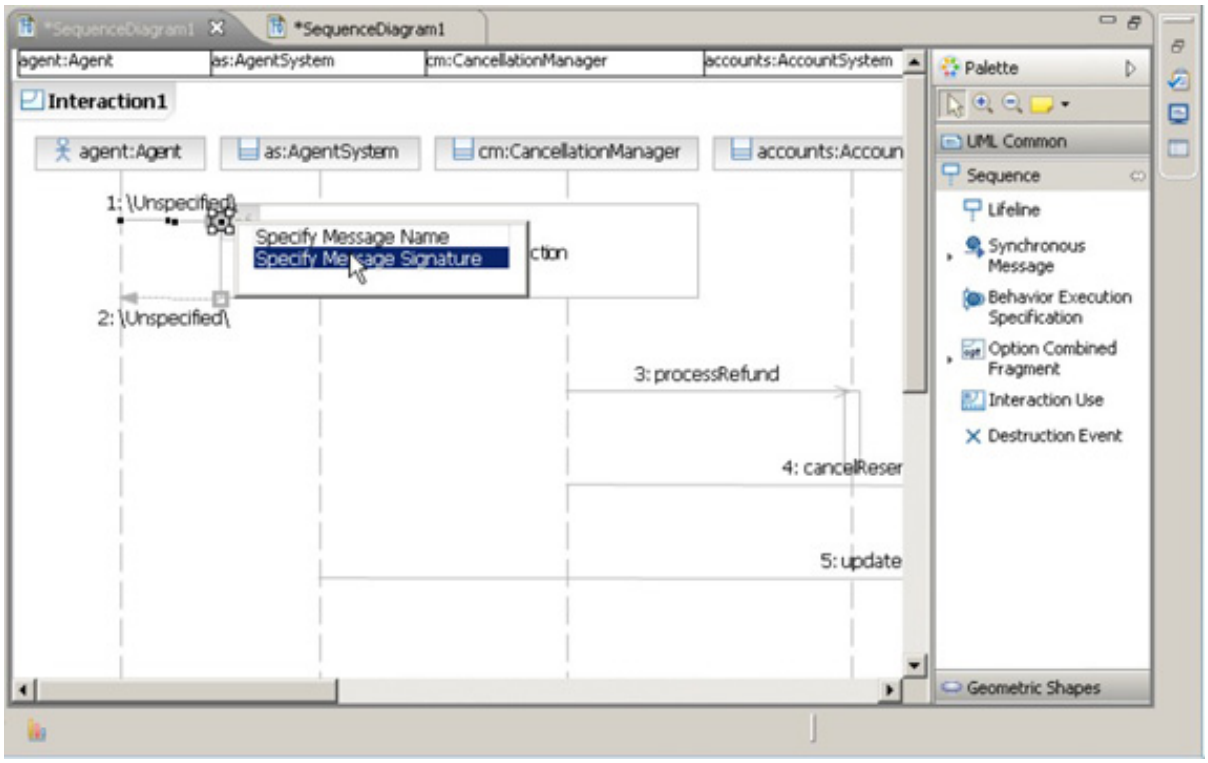


To model the Actual Gate to the InteractionUse in the original diagram, perform the following steps.

1. Drag a message from another lifeline (say, an Actor representing the `Agent` in Figure 18) to the InteractionUse.

2. Select the same operation as the one you chose for the Formal Gate.

Figure 18. Modeling an Actual Gate.



Tip: Rational Software Architect V7.5 contains validation rules to detect dangling and mismatched gates. The errors and tasks will show up in the **Problems** view and the **Tasks** view, respectively. The rule names are **Interaction gates** and **Actual gates – formal gates matching rule**.

Copy-and-paste enhancements

Copy-and-paste support has been much improved for sequence diagrams in Rational Software Architect V7.5. You can now copy Sequence diagram elements within and between Sequence diagrams, and you can copy Interactions within the **Project Explorer**. Probably the most useful enhancement relating to this functionality is that the paste operation is context aware.

This means that you can copy an Optional Combined Fragment into another. Arguably more useful, existing lifelines and elements are reused whenever possible. For example, if you copy messages, they appear at the bottom of the original lifelines. You can even copy lifelines and only a portion of the messages. These enhancements have made sequence diagrams much easier to edit.

Other new features and improvementsPrevious sections of this article discussed how certain validation rules have been added, along with gate support.

Identifying validation rules

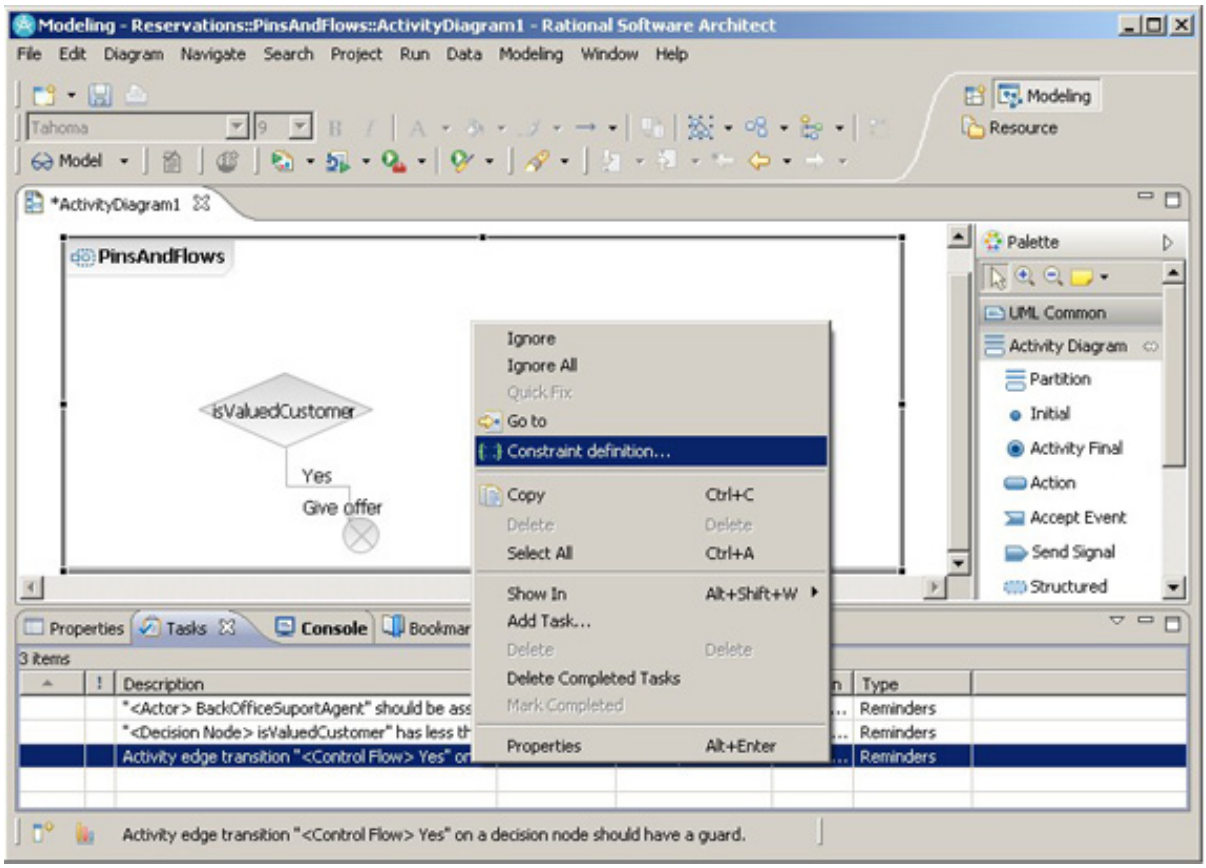
However, the reason for certain problems, warnings, and tasks may not be

immediately apparent to you. Now, there is an easy way to identify the rule corresponding to the validation failure.

1. Select a problem, warning, or task from the **Problems** view or the **Tasks** view and right-click it.

2. You will see a new menu item called **Constraint definition**, as shown in Figure 19. Choosing this menu item will give you detailed information about the task, so that you can better understand the suggested action. If the constraint is not mandatory, you have the option of disabling it by clearing the checkbox and clicking **OK**.

Figure 19. Get more details about the rule causing the validation failure.

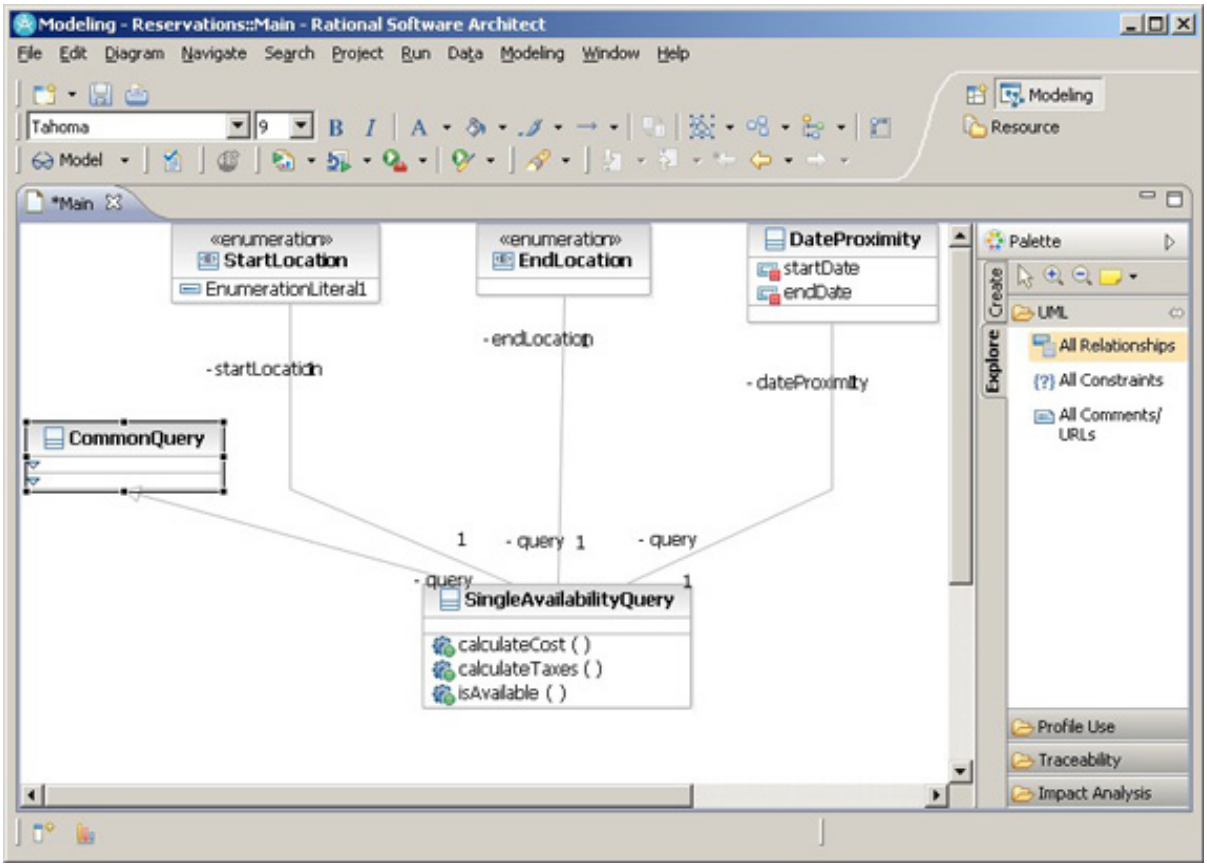


The Explore view palette

In Rational Software Architect V7.5, you have the ability to explore relationships easily. The **Explore** tab in the diagram palette allows you to explore different kinds of relationships. To see the elements related to a particular element, click a palette item, then the target element on the diagram surface. The relationships and the related elements all appear automatically.

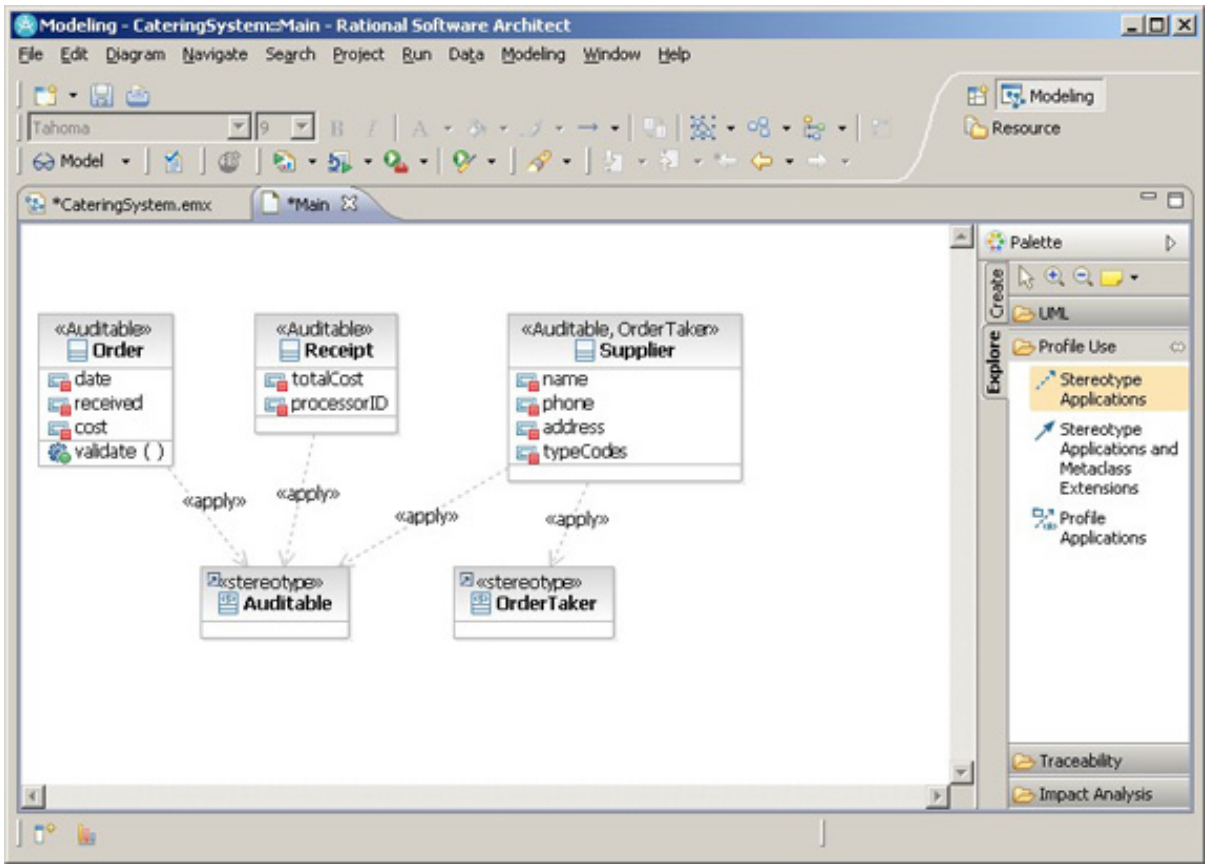
Figure 20 shows the relationships and related elements of `SingleAvailabilityQuery` (choose the **All Relationships** tool from the **UML** drawer of the **Explore** tab in the palette, and click the `SingleAvailabilityQuery` class).

Figure 20. Using the All Relationships tool to visualize relationships.



In addition, from the **Profile** drawer, you can visualize applied stereotypes and applied profiles, as shown in Figure 21. To do so, use the **Applied Stereotypes** tool from the **Profile** drawer of the **Explore** tab in the palette.

Figure 21. Using the Applied Stereotypes tool to visualize relationships.



Grouping

You may want to keep certain diagram elements together. For example, the `StartLocation` and `EndLocation` enumerations are similar concepts, so it would make sense that they appear close to each other on the diagram. To do so:

1. Select the shapes
2. Right-click them, and then choose **Filters > Group…**.

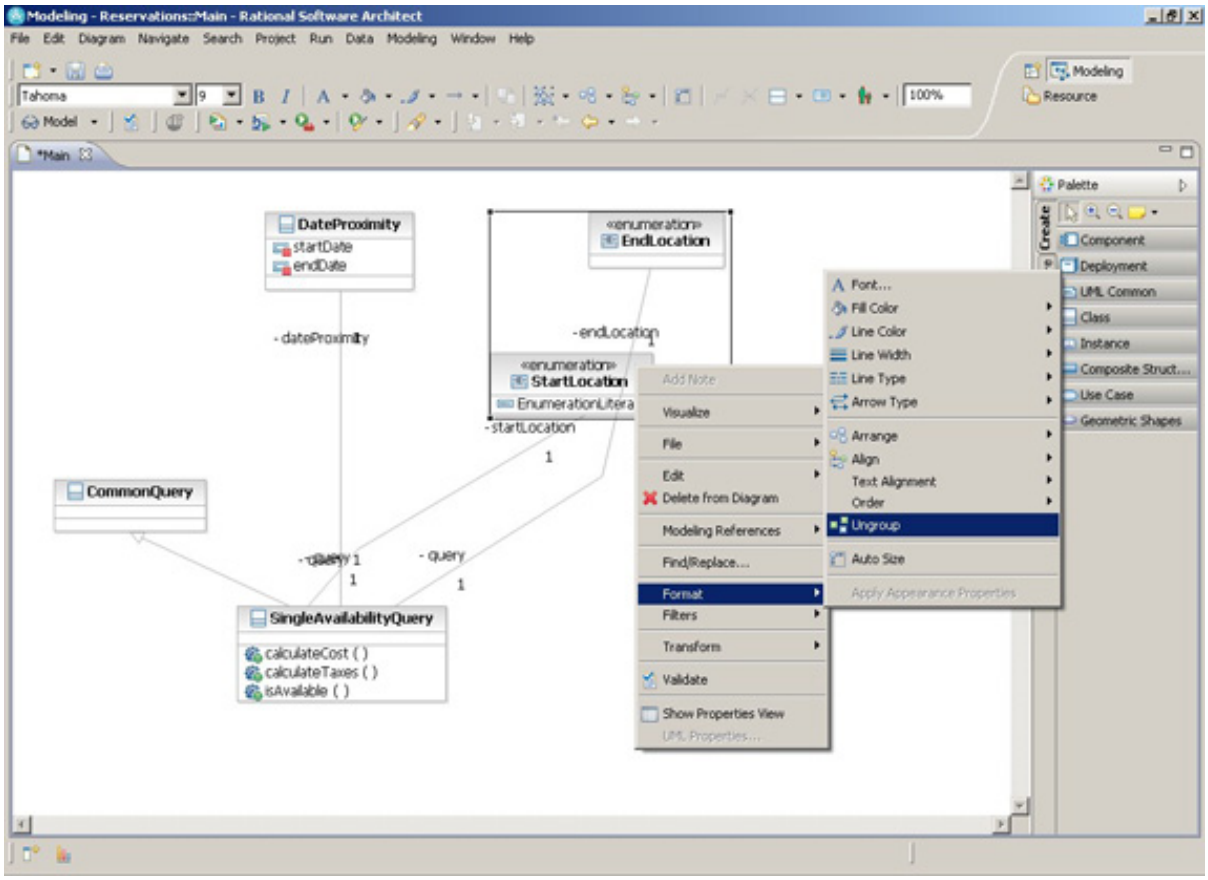
After they are grouped together, the elements are treated as one inseparable block, even during arrange operations. In addition, if the group is selected, **Arrange Selection** will arrange the objects inside the group.

Another benefit of grouping elements is the ability to modify the appearance of the grouped elements quickly. For example, the visibility of the compartments of `StartLocation` and `EndLocation` can be modified from the **Filters** pop-up menu.

Other properties can be set from the **Appearance** tab of the **Properties** view, or from the **Format** pop-up menu when the group is selected. This allows you to apply

a consistent look to multiple elements without having to reselect the same elements over and over again before making changes. You can un-group elements using the pop-up menu, as shown in Figure 22.

Figure 22. Ungroup grouped elements

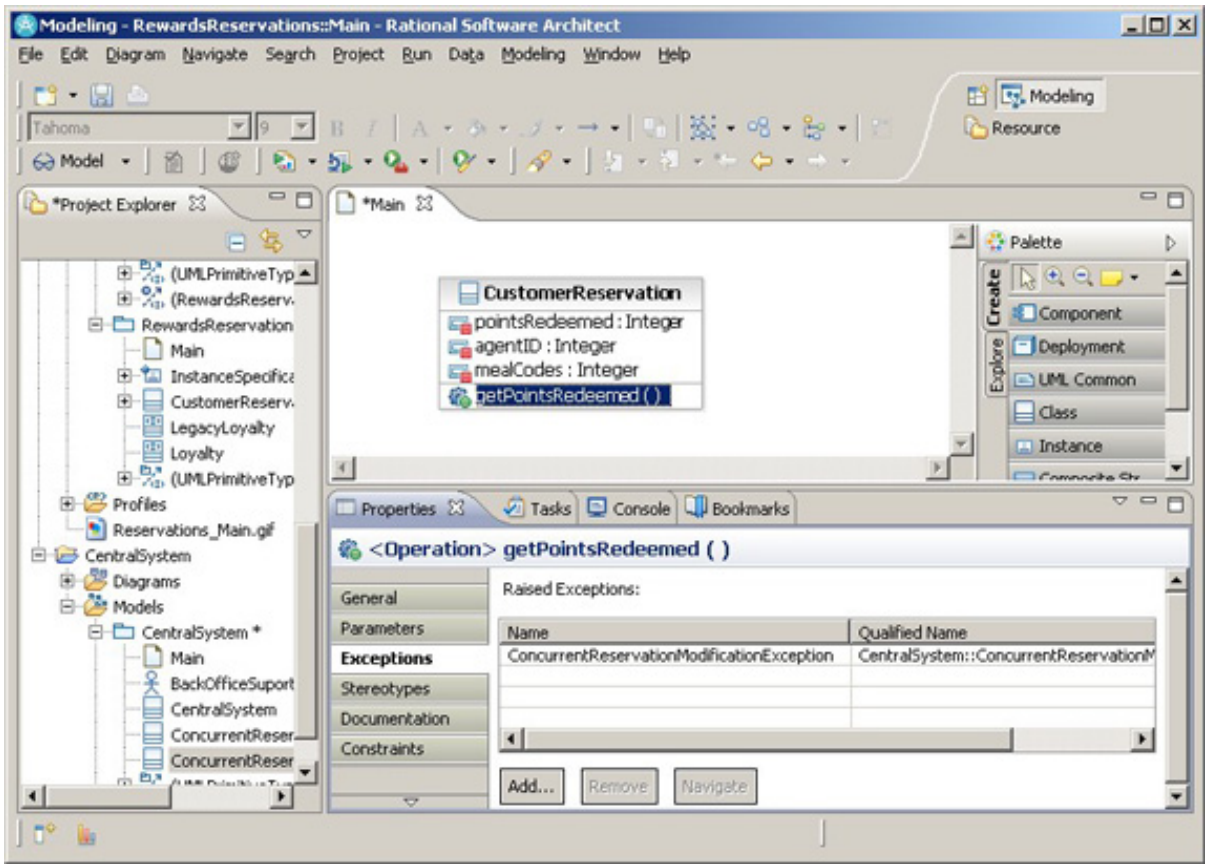


Modeling exceptions

In previous versions of Rational Software Architect, it was not possible to model exceptions. Rational Software Architect V7.5 adds support for modeling exceptions. You can try this out by doing the following.

1. Add an Operation to a Class.
2. Then, with the Operation selected, invoke the **Properties** view. You will see a tab titled **Exceptions**, as shown in Figure 23.
3. Click the **Add** button to choose an exception to add.

Figure 23. The Exceptions property tab.



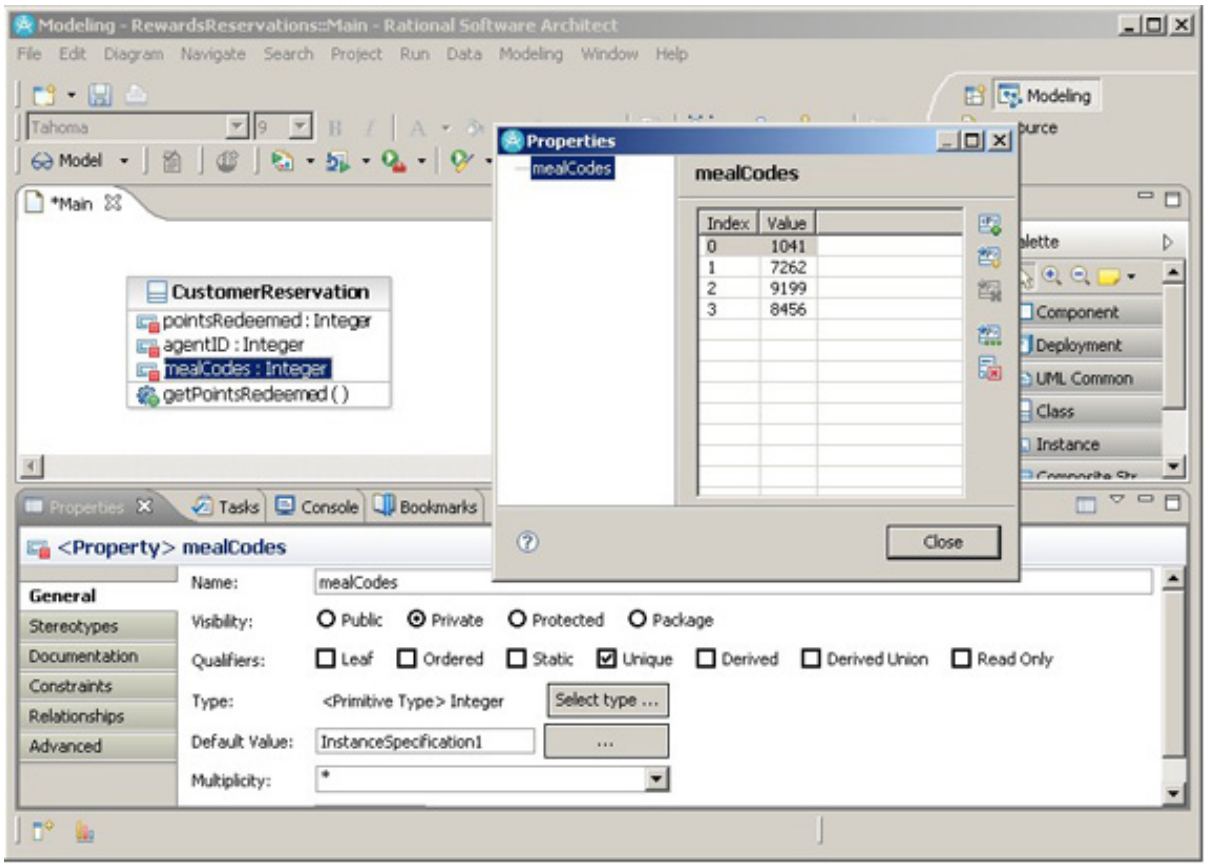
Modeling default values for multivalued attributes

You can specify multiple default values for an attribute in Rational Software Architect 7.5.

1. After creating your attribute, select the **Properties** view.
2. Click the **ellipses (...)** button. You will be presented with an interface for modeling multivalued attributes, as shown in Figure 24.

Tip: If the **ellipses** button is not active, be sure that you have set the multiplicity to ***** or a number greater than **1**, and that you have set a type for the attribute. Different types will result in different buttons appearing in the interface for modeling multivalued attributes. For example, setting the attribute to be of type `Enumeration` will result in different buttons than setting the attribute to be of a primitive type.

Figure 24. Modeling multiple default vales.

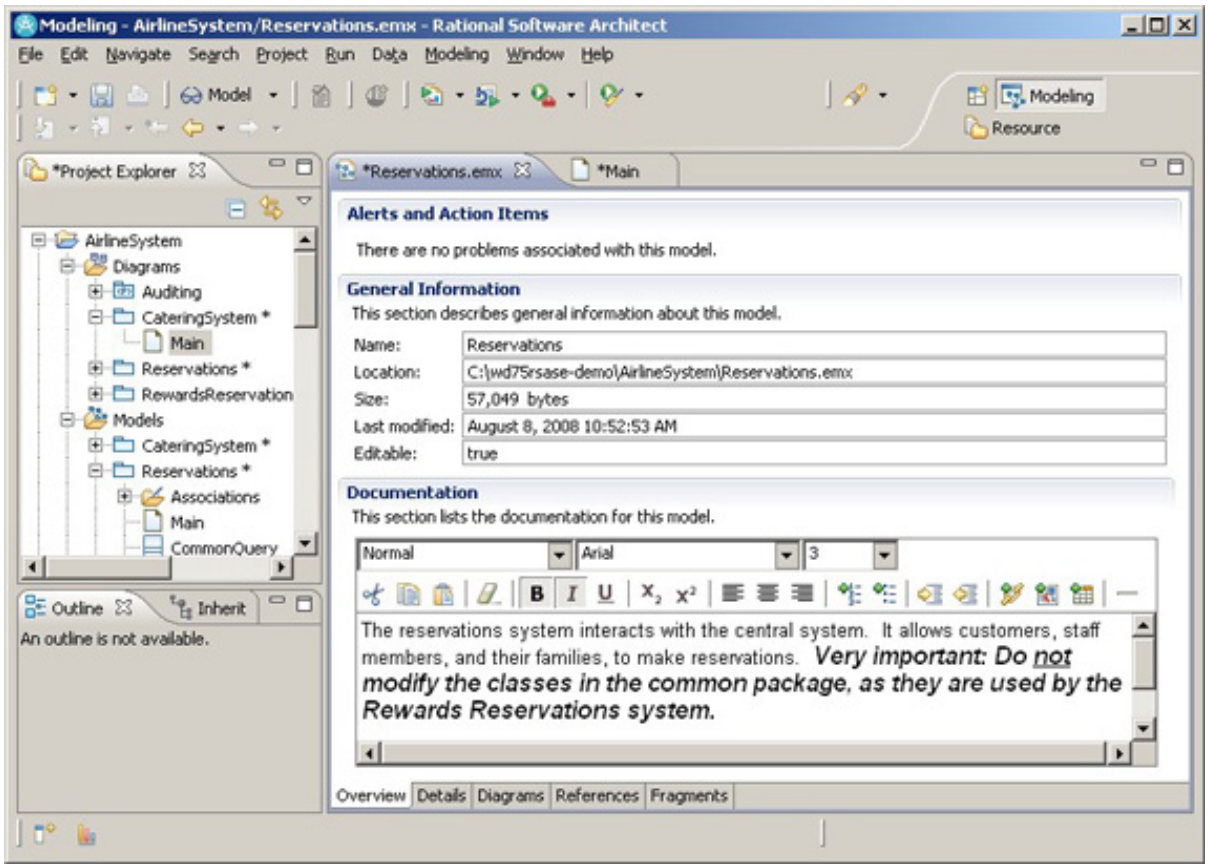


Likewise, if you perform Instance modeling, you will be pleased to find that Rational Software Architect V7.5 has much-improved support for slot values through a similar interface. The same improvements were added to Rational Software Modeler and Rational Systems Developer V7.0.5.1.

Rich text

The rich text functionality is immensely improved and more robust in Rational Software Architect V7.5, as shown in Figure 25. Of note, the behavior in Linux is slightly different than in Windows. For example, to edit the documentation in the **Model Editor** using Linux, click the **Edit** button underneath the documentation block to display a dialog that will allow you to edit the rich text.

Figure 25. Using rich text for model documentation



Element and relationship functionality

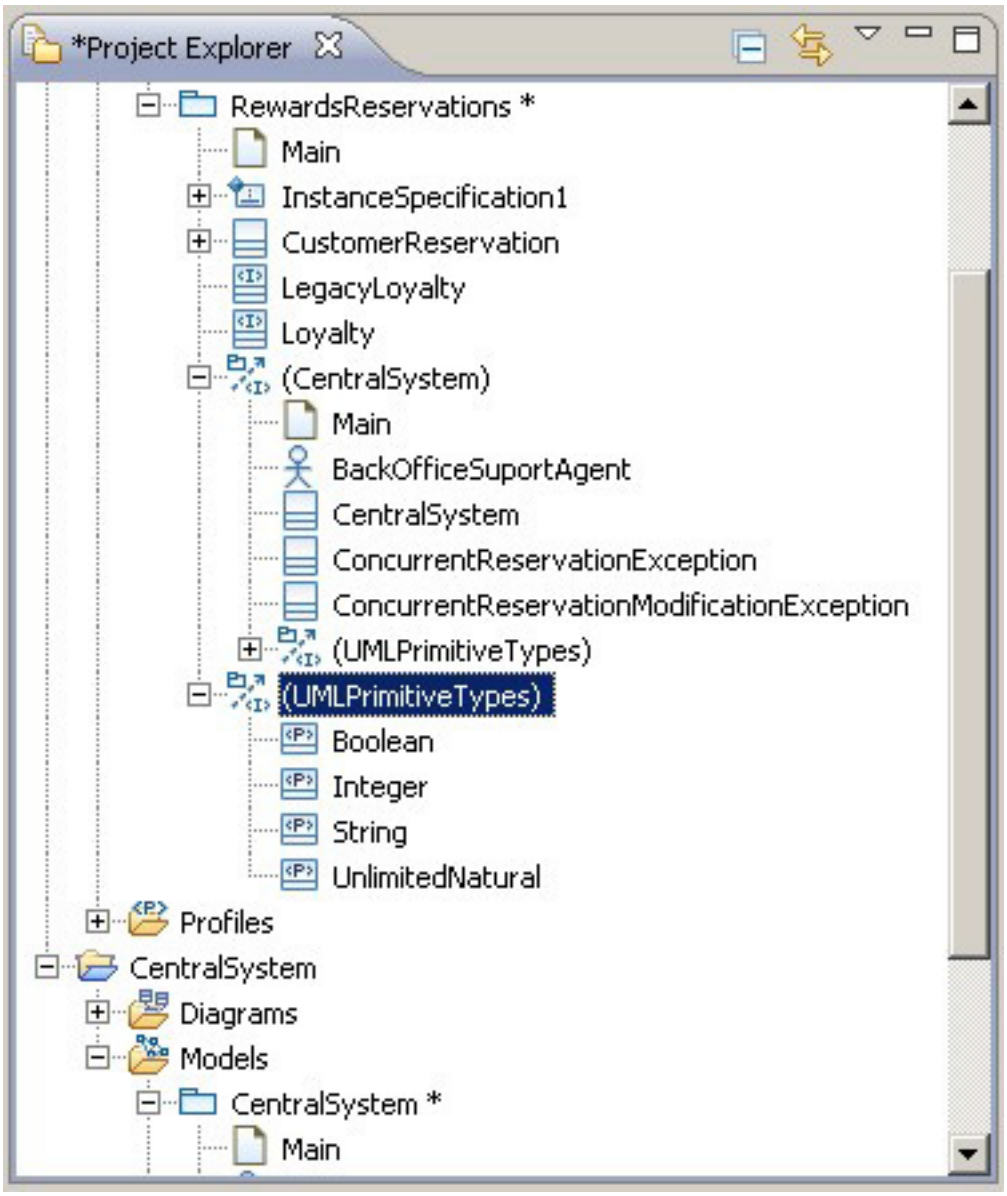
Rational Software Architect V7.5 has several other important new features.

Expanding element imports and package imports in place

You can easily expand element imports and package imports by clicking the plus sign, as shown in Figure 26. This functionality is new to Rational Software Architect V7.5, but was available in Rational Software Modeler and Rational Systems Developer V7.0.5.1. Being able to expand elements in place enables you to share manageable-sized models among larger teams and being able to reference them easily.

It is important to be clear that these imports do not necessarily have to be system libraries. They can be another model that you are sharing with another team member (or even another team), or they can point back to another package in the same model. As long as you can create an element import or package import to that element, it can be expanded in place.

Figure 26. Expanding package imports in the Project Explorer.



Select Element dialog

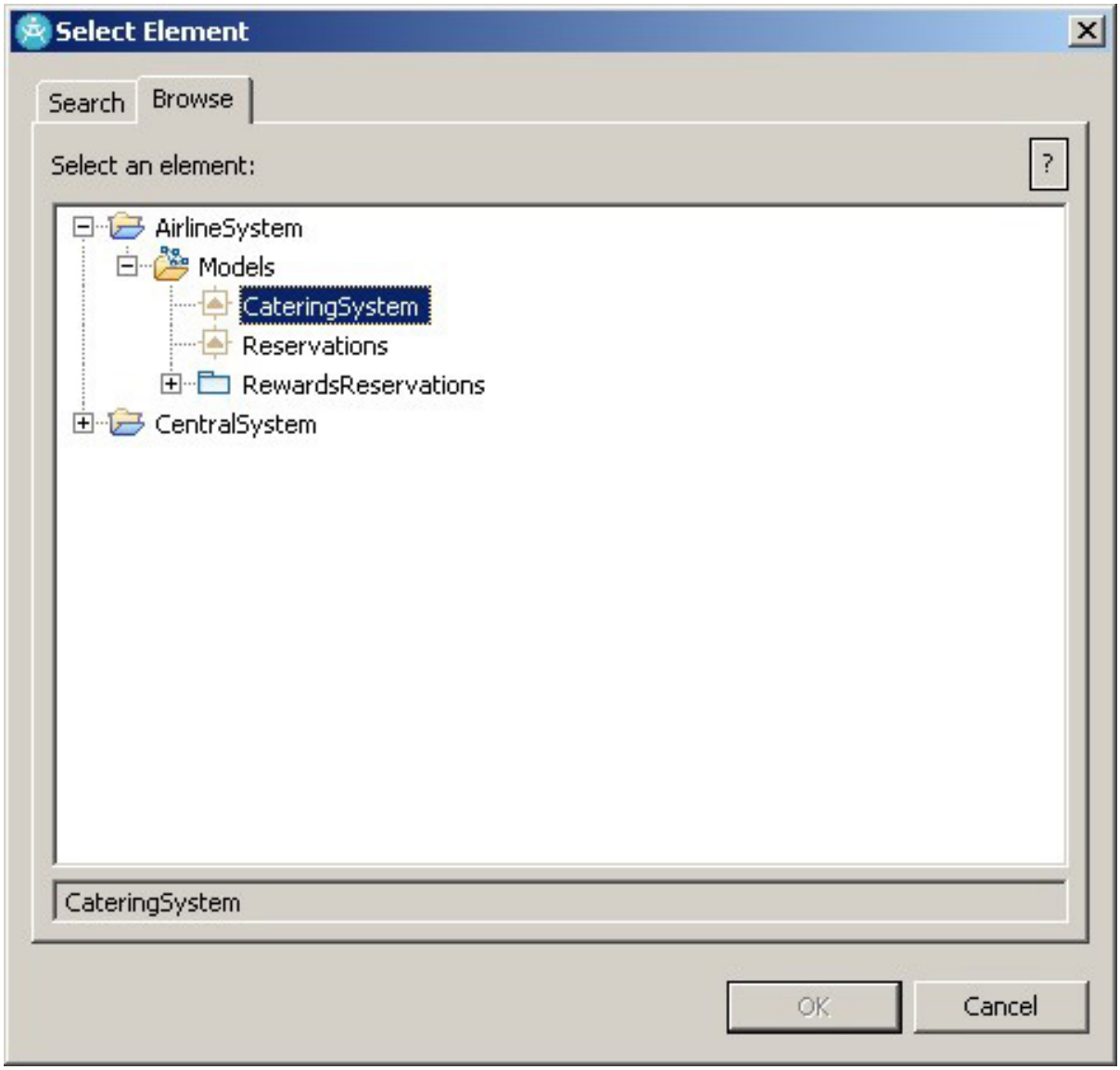
When you have a large number of models in your workspace, it is a good idea not to open them all at the same time, because this will negatively impact performance Indeed, the plus sign enhancement discussed previously helps reduce the number of open models in your environment.

In previous versions of Rational Software Architect, you were unable to work with closed models in the Select Element dialog. In fact, closed models were filtered out. In order to select something from a closed model, you had to cancel out of these Select Element dialogs, manually open the model, and then reinvoke the dialog to select the element.

Not anymore! First, the Select Element dialog shows the closed models in the **Browse** tab, as shown in Figure 27. Second, double-clicking the closed model will open it.

Tip: Of course, in the **Browse** tab, you can also expand element imports and package imports in the same way as in the **Project Explorer**. Being able to expand imports replaces the **Model Imports** virtual folder from previous releases, and is how you will now locate and select primitive types.

Figure 27. Closed models are shown in the Browse tab of the Select Element dialog

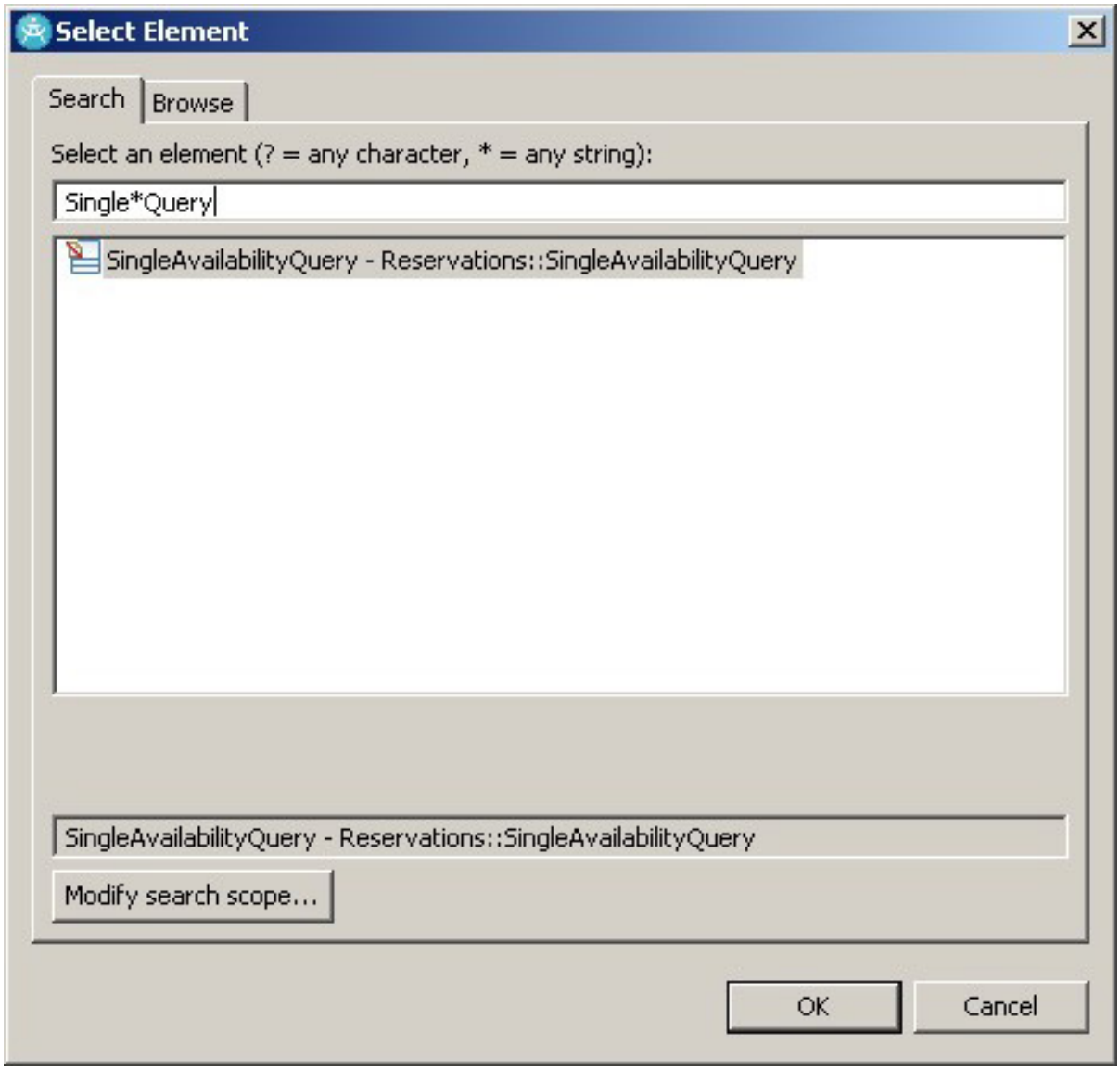


Similarly, in the **Search** tab of the **Select Element** dialog, elements from closed models are returned in the list of matches. Their icons are shown with a decorator at the top left, as shown in Figure 28.

1. Double-clicking one of these matches (or selecting it and pressing **OK**) will load the model and select the element.

2. To exclude closed models from your search scope, click **Modify search scope** and clear the **Search entire workspace** check box.

Figure 28. Elements from closed models in list of matches

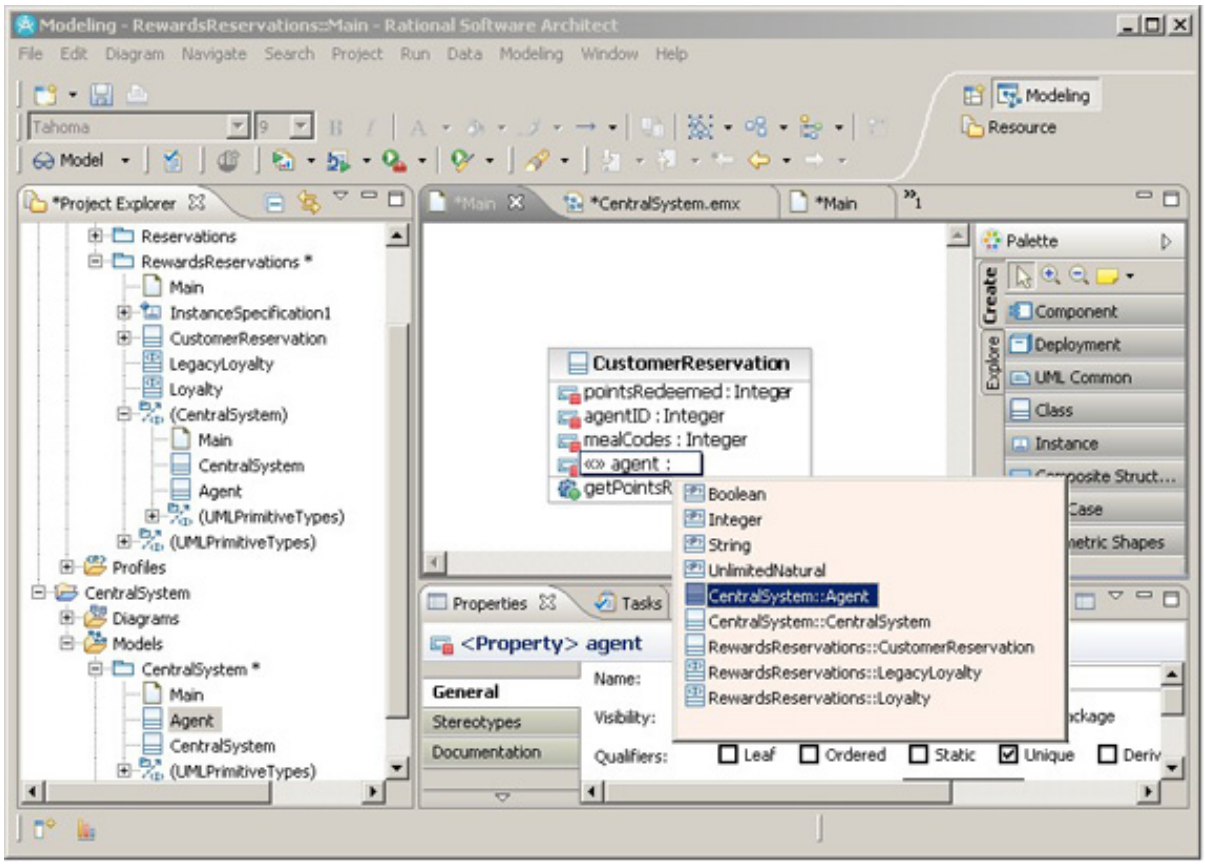


Content assist and creating unknown types

Element imports and package imports are not only helpful when you expand them in place. If you import an element and invoke content assist, types from the imported package will be included in the results. Content assist shows types from imported models. The `CentralSystem` package has been imported (see the Project Explorer

view shown in Figure 29), so its types show up in content assist.

Figure 29. Content assist

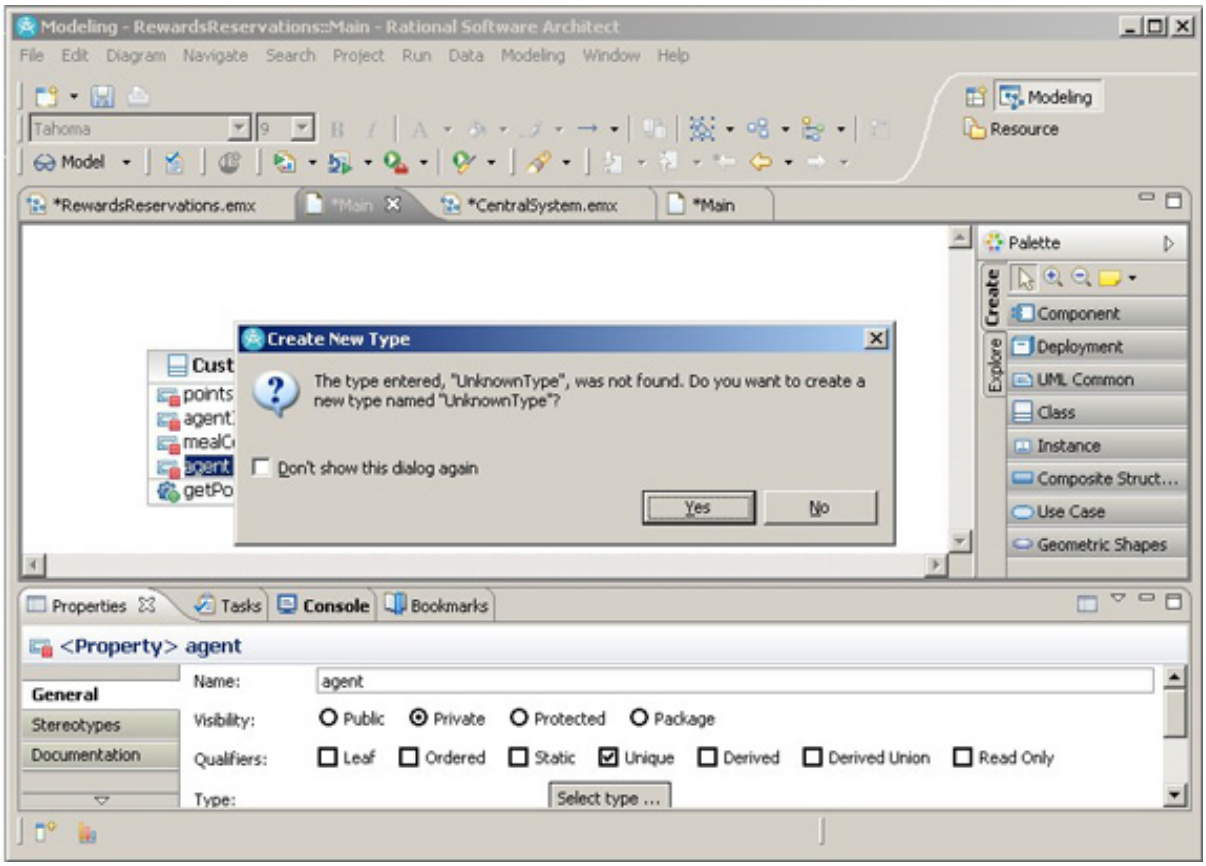


Notice that the Select Element dialog is more relaxed on visibility rules. In effect, it behaves like a mini Project Explorer, showing closed models and (depending on how it is invoked) possibly even elements that may not be valid selections.

Content assist, however, shows valid selections from the current model (where *current model* is in relation to the location of the actual element, not to a diagram on which the element may reside), and from elements which would be made available by an import, regardless of visibility. Elements from closed models and fragments are also shown, as long as they are valid selections.

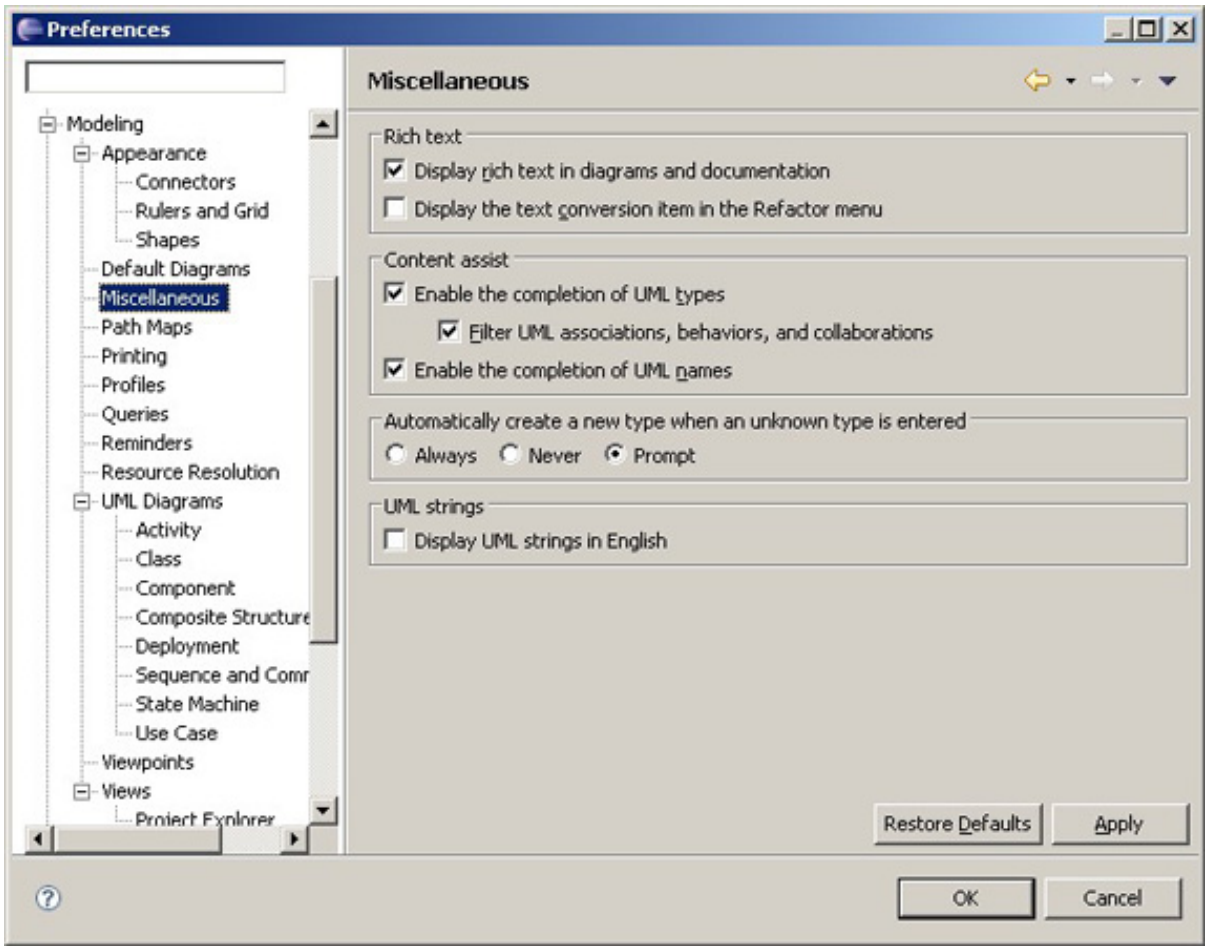
On the other hand, if you were to enter an unrecognized type, you will be prompted to create a primitive type, as shown in Figure 30. Created types will be added into a `PrimitveTypes` package at the root of your model. If his package does not already exist, it is created.

Figure 30. A dialog appears when an unknown type is entered.



You can control the settings of the Create New Type dialog from the Preferences dialog by adjusting the **Automatically create a new type when an unknown type is entered** setting on the **Modeling > Miscellaneous** page. In addition, you can also modify the content assist preferences from that preference page, as shown in Figure 31.

Figure 31. The Modeling > Miscellaneous preference page



Relationships

In previous versions of Rational Software Architect, in order to model a relationship, you had to drag at least the source or target onto a diagram, then draw a relationship using a relationship palette tool or connector handles. Also, if you did not want the elements on the diagram, you had to delete the two elements and the relationship.

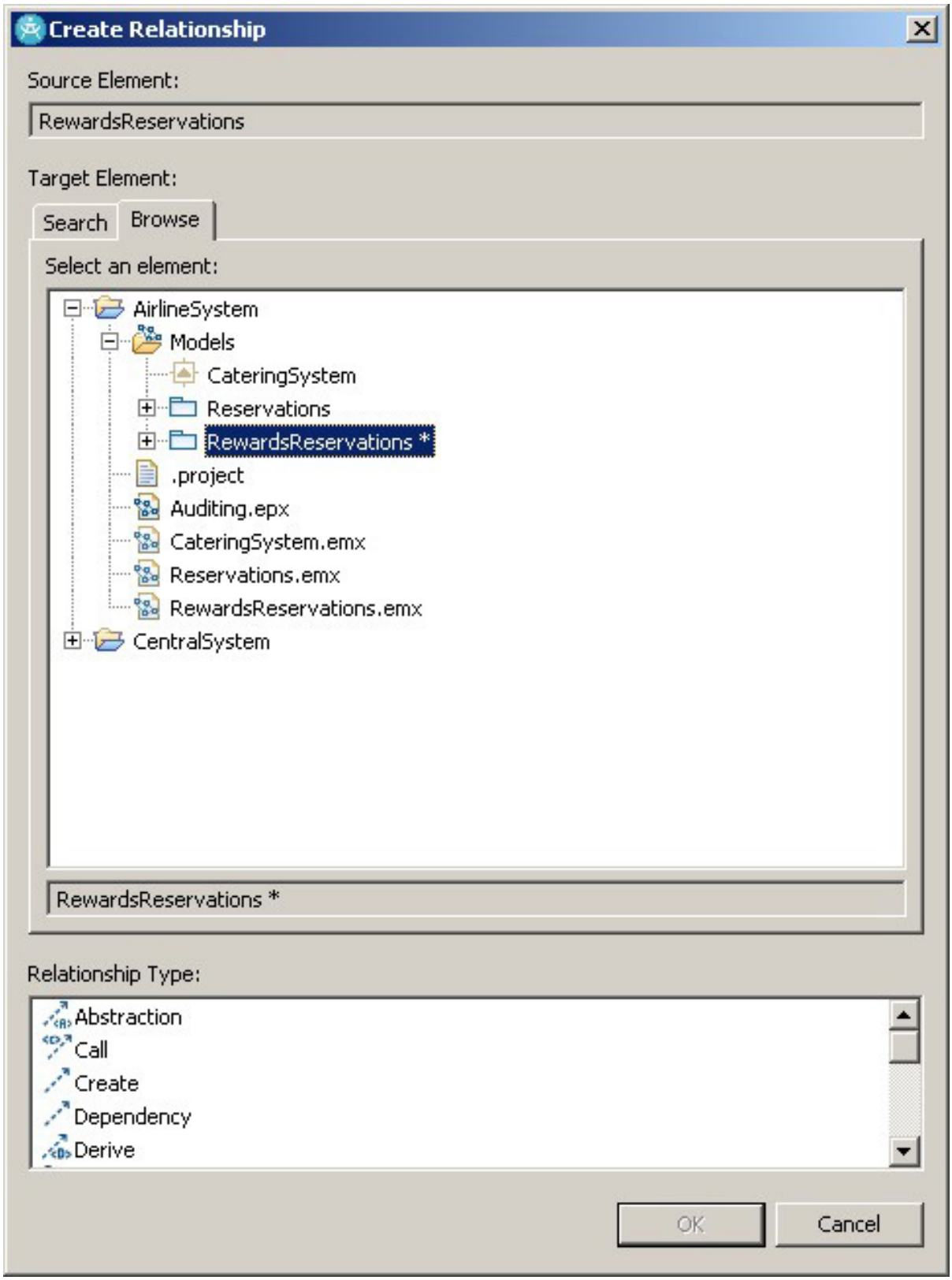
In Rational Software Architect V7.5, you can model a relationship without visualizing any of the elements on the diagram.

1. In the **Project Explorer**, locate the object that will be at the source end of the relationship.
2. Then, right-click and choose **Add UML**, **Relationship**.
3. You can pick from one of the common relationships in the menu or click **Advanced**. In either case, you are presented with a modified version of the familiar Select Element dialog, which you will use to choose the

relationship's target element. If you chose **Advanced**, you are also given the option of choosing the relationship type, as shown in Figure 32.

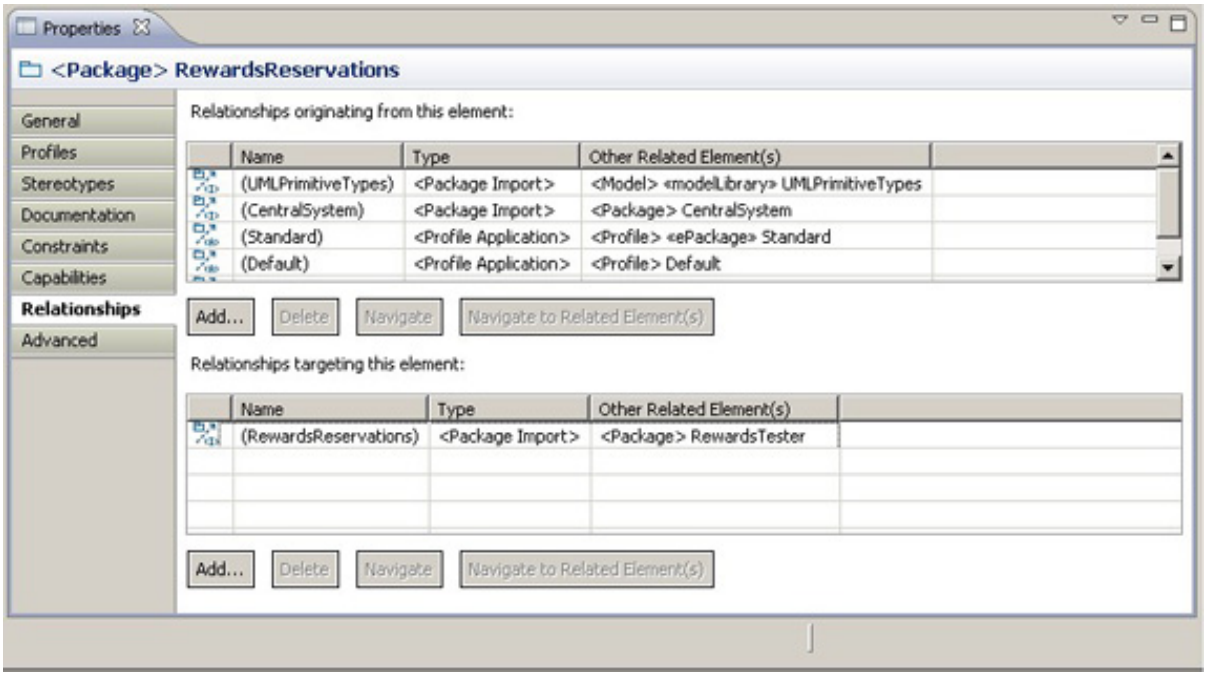
Tip: If **Advanced** does not appear in the **Add UML > Relationship** menu, enable the **UML Relationship 3** capability from **UML Element Building Blocks**.

Figure 32. Choosing the target end and the type of the relationship



Alternatively, you can also select a source or a target element, and then show the new **Relationships** property tab in the **Properties** view. From this tab, you can choose to create a new relationship and choose the target element or the source element of the relationship. (This is unlike the **Add UML** menu, which assumes that the selected element is the source element and only allows you to choose the target element.) The **Relationships** tab also shows elements participating in relationships, as shown in Figure 33. If you click the **Navigate** button, you will see the selected relationship(s) in the **Project Explorer**. On the other hand, if you click the **Navigate to Related Element(s)** button, you will see the other elements participating in the relationship(s).

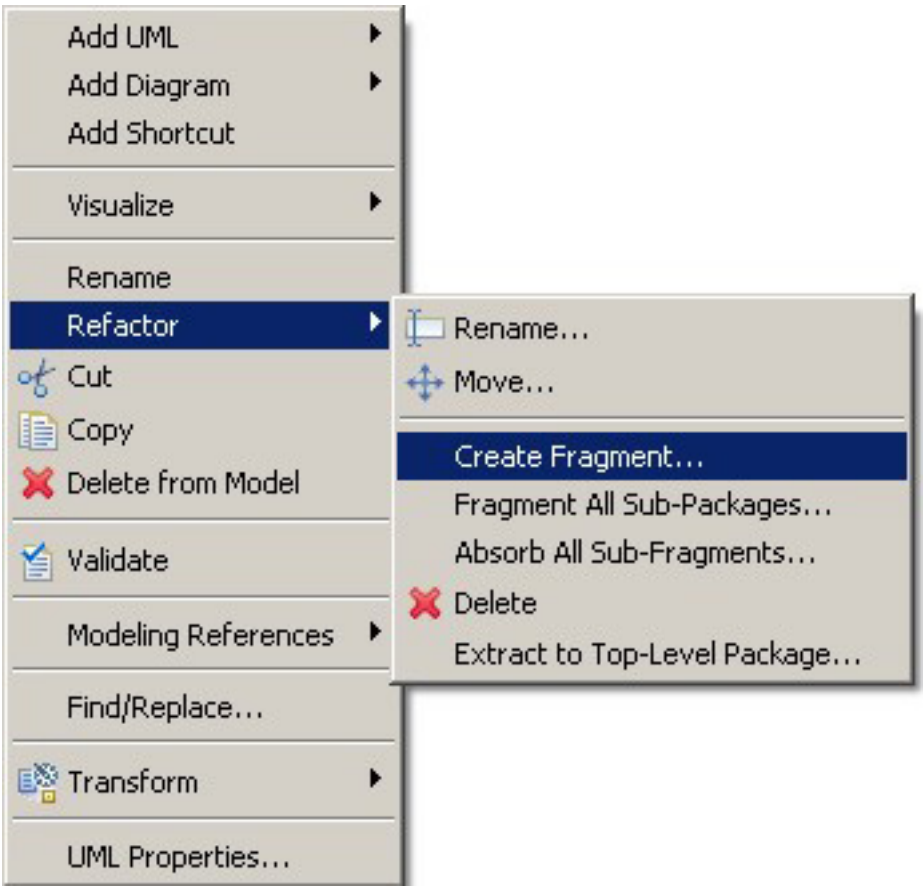
Figure 33. The Relationships tab in the Properties view



Model fragments

The most obvious change in fragments is that when you right-click the package in the **Project Explorer**, the **Create Fragment** and **Absorb Fragment** menus have moved under the **Refactor** submenu, as shown in Figure 34.

Figure 34. The Refactor submenu contains menu items to create and absorb fragments



In addition to these, you will find new menu items, including **Fragment All Sub-Packages** and **Absorb All Sub-Fragments**. These allow you to perform batch creation and absorption of fragments. Previously, fragments would have to be created or absorbed individually.

Tip: The selected item is not fragmented or absorbed when you choose **Fragment All Sub-Packages** and **Absorb All Sub-Fragments**. Only the nested child elements are fragmented or absorbed.

Refactoring

You have seen how refactoring portions of a Sequence diagram into an `InteractionUse` can simplify things. In addition, you saw how Rational Software Architect helps keep model elements in sync upon refactoring to preserve model integrity: namely, updating message names in Sequence diagrams, and having the Call Behavior Actions and Call Operation Actions use the referenced Behavior and Operation names. However, there are several more refactoring enhancements in Rational Software Architect V7.5.

Refactor, delete

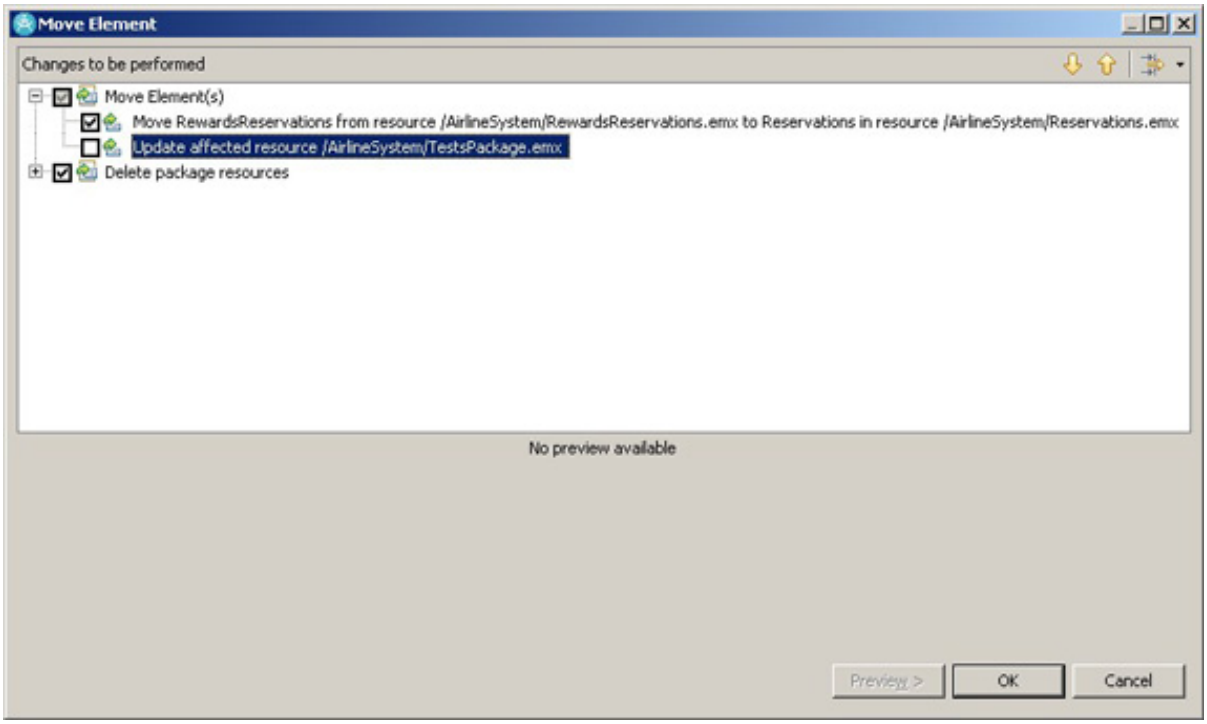
A new **Delete** menu item has been added to the **Refactor** pop-up menu in the Project Explorer. **Refactor > Delete** opens closed models (with references to the element being deleted), and updates the references. For example, diagram elements referencing the deleted element are removed.

Granular preview and the ability to disable non-compulsory updates

In Rational Software Architect V7.5, when you perform a refactoring (including creating and absorbing fragments), you are presented with a granular preview that lists all of the affected resources. Perhaps more importantly, you now have the ability to turn off non-essential updates, as shown in Figure 35. This is as opposed to being forced to abort the entire refactor operation, as in previous releases.

Therefore, this preview does not only serve as a warning as to whether the refactoring should be performed at all. In addition, it provides the ability to turn off a particular update, which can be essential in team scenarios (where not all files are writable). There can be times when you really do need to perform a refactoring, even when you are unable to update a dependant file (which you may not have write access to). An example of this is making an API change between versions, even though you know that you will be breaking an external client.

Figure 35. You can disable unwanted changes in the Refactoring preview window

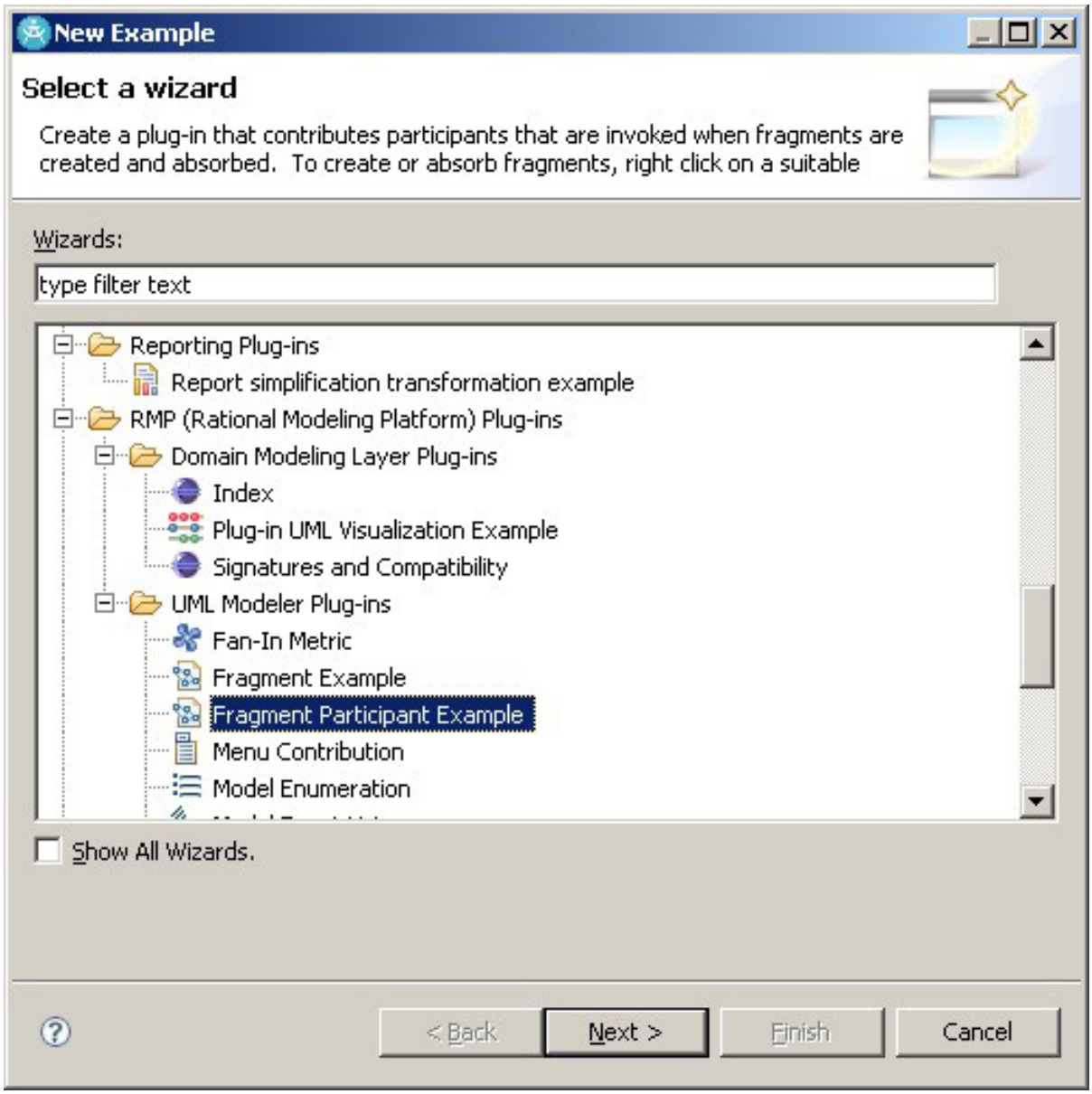


Adding fragment creation and absorption participants

Custom participants for fragment creation and absorption are now supported. This

parallels Eclipse's refactoring participants. To obtain details on the API usage, please choose **File > New > Example**, and then select the **Fragment Participant Example** from the **UML Modeler Plug-ins** category, as shown in Figure 36.

Figure 36. The wizard used to create the Fragment Participant Example in your workspace.



Tip: If you do not see the sample in the New Example wizard, be sure to install the **Modeling Extensibility** component from the Installation Manager.

New functionality in Rational Software Architect

In terms of enabling you to work in larger teams, model repair and profile application repair have been enhanced, both of which are instrumental in ensuring model consistency and integrity. The ability to expand element imports and package imports in place makes model sharing scenarios in larger teams far easier to implement than it has been.

In addition, offering a refactoring preview makes it easier to identify which changes will break, especially among larger teams with many unfamiliar models. With this feature, you need to understand that (on occasion) you must perform some changes that break things, which is why you have the option to disable certain updates.

As for ease of use, Modeling Working Sets help streamline the resources in your environment. This offers a new way of creating relationships without involving the diagram, clearly indicating what elements can be the target given the source (and vice versa). Grouping is now supported. Enhancements have been made to the Activity diagram, reflecting an understanding that not everyone wants to get into the details of UML semantics of flows and pins.

Arrange support has been improved. It is now easy to identify the validation rules corresponding to validation failures quickly. Model load times have also been made quicker, because the product enables you to browse through closed models quickly, using the plus sign. Content assist has been enhanced to consider imports, and a type creation feature has been added to help you quickly create types. Again, it focuses more on your workflow, rather than on the semantics.

The same can be said regarding the enhancements to the Select Element dialog: it focuses on your workflow. Sequence diagram copy functions have been enhanced to make it easier to edit the diagrams. Also, the **Explore** tab makes it quicker to find relationships between elements.

For compliance with the UML specification, formal and actual gates are supported in Sequence diagrams, and it is possible to have the Package as the root element.

As for general enhancements:

- The extensibility mechanism of the **New Model** wizard has been improved
- Rich text support has been enhanced
- It is now possible to model exceptions and multivalued attributes.

Furthermore, various enhancements to update the message names and to Call Behavior Actions and Call Operation Actions have made it easier to keep your models in sync, and to preserve model integrity, even when the inevitable refactorings take place.

Acknowledgements

The author would like to thank Dusko Misic and Michael Hanner for reviewing this article.

Resources

Learn

- Visit the Rational software area on developerWorks for technical resources and best practices for Rational Software Delivery Platform products.
- Subscribe to the IBM developerWorks newsletter, a weekly update on the best of developerWorks tutorials, articles, downloads, community activities, webcasts and events.
- Subscribe to the developerWorks Rational zone newsletter. Keep up with developerWorks Rational content. Every other week, you'll receive updates on the latest technical resources and best practices for the Rational Software Delivery Platform.
- Subscribe to the Rational Edge newsletter for articles on the concepts behind effective software development.
- Browse the technology bookstore for books on these and other technical topics.
- Learn more about UML at http://www.uml.org/: The Object Management Group's UML site.
- Understand more about authoring UML profiles by reading this article -- Authoring UML Profiles: Part 1
- Understand more about authoring UML profiles by reading this article -- Authoring UML Profiles: Part 2
- Discover more Eclipse wizards at -- The org.eclipse.ui.newWizards extension point

Get products and technologies

- Download trial versions of IBM Rational software.
- Download these IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Tivoli®, and WebSphere®.

Discuss

- Join the developerWorks Community in forums, blogs, podcasts, wikis, and more.
- Join the Rational Software Architect forum on developerWorks.

About the author

Wayne Diu
Wayne Diu is a software developer at IBM (Rational tools). He has developed several features for the Rational Modeling Platform (such as the Browse Diagram infrastructure), and was one of the developers responsible for the platformization of the metamodel integration framework. He has also worked on a diverse collection of other features, including the original implementation of the Profile Tooling feature to allow custom domain modeling, and he has developed some of the features discussed in this article.