Building Reusable Architectures using Rational Rose Frameworks.

KhawarAhmed

Rational Software White Paper

Rational®
the e-development company™

Table of Contents

Release Better Software Faster

These four words aptly capture the competitive nature of today's software development environment. Not only must we develop better software; it must also be done in less time (and with fewer resources!). This multi-pressure reality, termed the e-software development paradox by Rational® Software, is not beyond manageability, but requires a planned approach to reuse and some careful thinking.

Rational Rose's ® framework capability can be used effectively in this respect. This paper will explain the framework capability, its advantages, and how to create and use frameworks in Rose.

So, What's a Framework?

Formally speaking, a framework is a predefined modeling element set for modeling specific systems. In other words, a framework permits you to specify, group together, and reuse model elements to effectively model some specific software system.

Lets say you are in the business of building systems that always have customer billing and account management functionality. You could start each system from scratch and rewrite the billing and account management portions, but that would mean a lot of unnecessary rework. More realistically, you would likely take the billing/accounting pieces from one of your earlier implementations, and reuse them in some fashion. Rose frameworks provide you the opportunity to formalize and better manage such reuse.

Some Rational Rose users may already be using frameworks, without realizing that they are using the Rose framework capability. For example, if you are a Rose J user, every time you choose JDK, JEterprise, or JFC to start off your Rose model, you are actually choosing to use the respective Rose frameworks built through reverse engineering the relevant libraries. In fact, the process outlined in this document can be used to create new framework for a JDK, for example, when a new release comes out.

Why Use a Framework?

A framework can be used in two basic ways. In the first approach, you use a framework for establishing a set of reusable components. In the alternate approach, a framework is used for creating a template for new models or for defining the

architecture of specific types of systems. Each approach has its advantages, and requires different levels of advance planning and effort.

Lets look at these two approaches in a little more detail.

The Library Approach

The first approach, using a framework to create a set of reusable components, is the easier one in the sense that it's very much like using a library, with the added benefit that the library is completely visible within your model. Referring back to the system with the billing and account management capabilities, you would simply take all the relevant classes, put them together and create a framework containing the classes of interest. When its time to implement your next system, it's simply a matter of using the framework and reusing the desired pieces within it to develop the billing and account management functionality.

The Framework as Template Approach

In the framework as template approach, you essentially create a framework from a model that contains assembled pieces of your typical system. Creating a new system simply requires you to use the framework as the basis of the new model, and then fill in the blanks or customize the model to the new system's requirements. Clearly, this is more work than simply putting some classes into a loosely organized library, and requires some planning in advance. However, it also yields superior results in terms of reuse because not only do you reuse the code, you also use the framework to capture and reuse key, highly scarce knowledge of the system architects. The template approach allows you to develop new systems faster because not only do you get the implementation code for the pieces, you also get an authentic blueprint for putting it together in a consistent and usable manner.

Regardless of the approach, the end result of using frameworks is an increase in the time you spend on developing the features and functionality, and less on rehashing what you've already done. In the process, you also decrease the overall software development time due to higher reuse, and less rework.

How to Build a Framework

In order to create or use frameworks, the **Framework Wizard** Add-In it must be active (Note that this add-in is only available on Windows). You can easily find out if it is active by choosing **File > New** command in Rational Rose. If it displays a **Create New Model** dialog box, the Framework Wizard is active.

If the **Framework Wizard** Add-In is not active, you can activate it by clicking **Add-Ins > Add-In Manager** in Rational Rose. Select the **Framework Wizard** option and click **OK** (if the Framework Wizard option is not present, the Framework Wizard Add-In is not installed).

The first step to using the framework capability is to create a new framework. This is a very simple process. All you need to do is save a model, and then select it as the basis for a new framework. Here's a step-by-step guide to creating a new framework:

- Create and save a model with the contents of the framework in any folder. That model will be used as the template when creating new models from this framework.
- Choose **File > New**. If the Framework Wizard Add-In is installed and activated, the Create New Model dialog box opens.
- Double-click on **Make New Framework** icon. This starts the Framework Wizard. (If the welcome page is shown, click Next.) The Framework Wizard - Specification of Framework dialog box opens.
- In the **Framework Name** field, specify the name of the new framework. The name must be unique among the existing frameworks, and it can only contain characters that are allowed in folder names.

- In the **Model File** field, specify the name and location of the file that constitutes the framework model. To browse to the file, click in the field. Then click the displayed button.
- To specify what diagram to be initially opened for models created from this framework, click in the **Start Diagram** field. The specified framework model opens. Click the arrow to the right of the Start Diagram field, and select one of the diagrams.

Optionally, you can specify the name and location of documentation and icon files in the Documentation File and Icon File fields. (To browse to a file, click in the field. Then click the displayed button.)

Click Next. A summary of the new framework is shown.

Creating a Framework from JDK

Creating a new framework from a JDK is very similar. Just drag and drop the archive file onto a Rational Rose diagram. Once it is reverse engineered into a Rose model, save that model, and use it to create the framework as detailed above.

Protecting your Framework

Of course, your framework is only good as long as it is maintained in its original form. Minor tweaking here and a slight change there and pretty soon even you won't be able to use your own framework (not to mention all the confusion it would cause). You can protect against this scenario by making model elements of your framework read-only so that any modifications are made in a controlled manner.

To write-protect your framework:

- Place the model elements you want to write-protect into a package, lets call it **myPackage**
- Select **myPackage** in the browser, and choose **Units>Control myPackage**. Enter a suitable location and file name when prompted
- Select **myPackage** and choose **Units>write-protect myPackage**
- Save the model, and use the model to create the framework (as detailed above)

You should know that each framework is stored in a separate folder in the **\Framework\Frameworks** folder (the framework library) in your Rational Rose installation folder. If you ever want to delete a framework, you must go to this directory, and manually delete the framework.

Note also that each package in a predefined framework is stored as a controlled unit in a separate file. Th access the contents of a package in a framework, you have to load the corresponding controlled unit. To load a unit double-click on the package in a diagram, or click **File** > Units and click **Load**.

Go Forth and Framework

As you can see, Rational Rose framework capability is powerful, effective, yet simple to use. Not only can it simplify your development; it is also an effective tool to use in overcoming the e-development paradox.

Corporate Headquarters
18880 Homestead Rd.
Cupertino, CA 95014
Toll-free: 800.728.1212
Tel: 408.863.9900
Fax: 408.863.4120
Email: info@rational.com
Web site: www.rational.com

For International Offices: www.rational.com/worldwide