Introducing IBM Rational Software Architect
Improved usability makes software development easier

Kunal Mittal

February 15, 2005
(First published February 15, 2005)

This article introduces IBM Rational Software Architect and the modeling capabilities that Rational Software Architect provides. It guides the beginner through the basic views, and explains how Rational Software Architect leverages the Eclipse 3.0 framework for usability. You'll learn how to use Rational Software Architect to create simple class and use-case diagrams, as well as how to generate code from the class diagrams. Finally, it briefly covers how you can perform an architectural analysis of your code. Follow-up articles will elaborate in more detail on what is covered in this introductory article.

What is Rational Software Architect?

IBM® Rational® Software Architect -- part of the IBM Software Development Platform -- is the first major release of Rational products since IBM's acquisition of Rational in February 2003. The revamped Software Development Platform features a new level of integration and ease-of-use that developers working with IBM products will enjoy.

The Software Development Platform consists of several IBM Rational products:

- **IBM® Rational® Portfolio Manager**: A project management tool that encompasses the full life cycle of a project, Rational Portfolio Manager helps provide real-time visibility into portfolio investments. This in turn allows managers to make well-informed decisions. This product can also be used to increase collaborations between various development teams.

- **IBM® Rational® Application Developer and Rational Web Developer**: This is a re-branding, respectively, of the IBM® WebSphere® Studio Application Developer and IBM®WebSphere® Studio Site Developer integrated development environment (IDE) tools. These newer products are now the baseline on which IBM® Rational® Software Modeler and IBM Rational Software Architect are built.

- **IBM Rational Software Modeler**: Rational Software Modeler is a tool based on Unified Modeling Language (UML) 2.0 that allows architects and analysts to create different views of the system. This tool fosters good communication between architects and designers on one hand and the development team on the other hand.
• **IBM Rational Software Architect:** This tool allows architects to design and maintain the application architecture. Rational Software Architect is the focus of this article.

• **IBM® Rational® Manual Tester:** This tool allows analysts and testers to create and execute test cases. Rational Manual Tester is aimed at unit testing.

• **IBM® Rational® Functional Tester:** This tool can be used to automate functional and regression tests.

• **IBM® Rational® Performance Tester:** This tool enables automating load and performance tests for your applications.

Because Rational Software Architect is a superset of Rational Application Developer, Rational Web Developer, and Rational Software Modeler, you get all of these tools when you download Rational Software Architect. For more information about these tools, visit IBM's Rational Web site--see the Related topics section below.

**Rational Software Architect's usability**

As a long-time user of the IBM Rational Rose tool, my pet peeve with the tool has always been its usability. This IBM release of the Rational products, however, is built entirely on top of Eclipse 3.0, which not only provides a unified interface for all the technical folks (architects, designers, and developers) on your team, but also allows the tools to leverage the usability features that Eclipse provides. (For more information about Eclipse, visit the Related topics section later in this article.) Figure 1 shows the basic user interface (UI) for Rational Software Architect.

**Figure 1. The Rational Software Architect UI**
Because Rational Software Architect is built on Eclipse, you can use Rational Software Architect for Java 2 Platform, Enterprise Edition (J2EE Platform) technology. In addition, using the code-generation features that are a part of Rational Software Architect, you can convert designs and UML diagrams drawn in the Modeling perspective into code. Finally, the underlying Eclipse platform also provides a robust and feature-rich IDE for developers.

However, Rational Software Architect is not limited to Java technology or the J2EE platform. You can also convert the UML models you create into C++ code as needed. The code-generation features of the tool allow you easily to generate base code that you can then customize. This feature allows developers to leverage the modeling and design pattern capabilities of Rational Software Architect for non-J2EE platform development.

The basics of Rational Software Architect

This article is very hands-on. I recommend that you launch Rational Software Architect on your machine while reading this article. If you don't already have Rational Software Architect on your machine, you can download an evaluation copy from the "Trials and betas" page—see the Related topics section below.

Rational Software Architect introduces a modeling perspective and several views. The view into all of the features of Rational Software Architect is the modeling perspective, which is where architects and designers can build UML diagrams, apply design patterns, and create detailed specifications for developers. You can also generate code in this perspective. To do so, you perform the following steps:

1. Launch Rational Software Architect, click File, click New, and then select Project.
2. In the New Project wizard, select UML Project.
3. Type a name for the project, and then click Next.
4. Create a blank model by selecting Blank Model from the drop-down list, and then click Finish.

Views in Rational Software Architect

Several modeling views are available in Rational Software Architect, as Figure 2 shows:

- The Diagram Navigator allows you to navigate through some basic patterns, as well as the different diagrams you can create.
- The Model Explorer functions like Microsoft® Windows® Explorer: It allows you to navigate through your projects and the different assets you create.
- The Pattern Explorer allows you to apply the Gang of Four patterns and use them in your design. You can also create custom design patterns. (My next article will discuss design patterns in IBM Rational Software Architect and contain information about Gang of Four patterns.)
- The UML Editor view is your workspace: You create and work with diagrams in this view.
UML modeling using Rational Software Architect

Rational Software Architect is compliant with the UML 2.0 specification; using Rational Software Architect, you can create several different types of UML diagrams. Let's look at a few examples of the types of diagrams that you can create. (For more information and a link to the Modeling Structure Guidelines whitepaper, see the Related topics section at the end of this article.)

Create a UML class diagram

You use class diagrams to describe the types of objects in a system and their relationships. For a single class, you can define the name of the class, its attributes, and its operations. In addition, you can define the relations between classes. The easiest way to understand this process is to create a simple class diagram. I created one that has one interface and two implementers. Perform the following steps to re-create this class:

1. In the Model Explorer, right-click Blank Model, and then add a new class diagram.
2. Type a name for the new class diagram. (For this example, use the name Main.) A new tab called BlankModel::Main will appear (as shown in Figure 3).
3. Right-click anywhere in the BlankModel::Main window, and then add a new interface.
4. Right-click again anywhere in the window, and then create a class.
5. Repeat step 4 to create a second class.
Define a relationship which indicates that the two newly created classes are implementers of the interface. Click the name of the class and you'll see that up and down arrows appear on the right side of the dialog box. Click an arrow, and then click **Create Implementation**. You've now created your first classes in Rational Software Architect. To see all of the options available to you for classes you create, right-click any class, and then select **Properties** (as shown in Figure 4).
Figure 4. Class properties

UML to code

Now, let's generate code out of the UML class diagram you just created. To do so, perform the following steps:

1. Select all the classes in the diagram shown in Figure 3.
2. Right-click your selection, and then click Transform.
3. Select the UML to Java option.
4. In the Transformation wizard, create a new target container.

Because the original project you created was a UML project, you now need to create a Java project in which to generate the code. Leave the defaults on the other tabs, and click Run in the Transformation wizard. You’ll see all the code being generated and displayed in the Explorer window on the left. Click the code to see how it looks. Listing 1 shows the code generated from one of the classes defined in the UML.
**Listing 1. Code generated from a class defined in UML**

```java
/*
* Created on Nov 20, 2004
*
* TODO To change the template for this generated file go to
* Window - Preferences - Java - Code Style - Code Templates
*/

/*
* @author Kunal Mittal
*
* TODO To change the template for this generated type comment go to
* Window - Preferences - Java - Code Style - Code Templates
* @uml.annotations
* derived_abstraction="platform:/resource/DevWorksProject/Blank Model.emx
* #_PnydKDsQEdmyqsBhJhf6RA%2cuml2.Interface%2cBlank+Model%3a%3amyClassInRSA"
* @generated "UML to Java
* (com.ibm.xtools.transform.uml2.java.internal.UML2JavaTransform)"
*/

public interface myClassInRSA {
}
```

**Code to UML**

Now, let's do the reverse: Create a UML model out of the code you have. This process is actually quite easy. Simply create a blank class diagram. Drag your code from the Java project into the blank diagram. The UML class diagrams are generated automatically. As you're performing this task, note that there are subtle differences between this process and previous versions of Rational Rose or XDE:

- If your class diagram is in the UML model, you're actually referencing the source code in your Java Project.
- You can also create and view class diagrams from within your Java project. In this case, the diagrams are just visualizations of the code.

For more information on this topic, see the article, *How to migrate from code-centric to model-centric development using Rational Software Architect*, listed in the Related topics section at the end of this article.

**Create a UML use case diagram**

A *use case* is a sequence of actions that an actor performs on a system and the response generated. A *use case diagram* is a high-level view of all the use cases within a system and the actors that interact with them. (You can read more about use-cases by consulting the references in the Related topics section at the end of this article.)

Use case diagrams are easy to create. To create a use case diagram, perform the following steps:

1. Create a diagram called *Diagram1*.
2. Using the palette, create an actor called *Actor 1*.
3. Create two use cases called *Use Case 1* and *Use Case 2*.
4. Click each use case and drag it to the actor to connect the actor to the use cases.

Figure 5 shows a completed use case diagram.

**Figure 5. Completed use case diagram**

![Completed use case diagram](image)

**Architectural discovery**

Rational Software Architect includes functionality to help you perform architectural discovery on your model or code, which can help you understand your model from an architectural perspective. Figure 6 shows a simple UML model that I created. Bring up the Diagram Navigator view, and then expand some of the properties. Notice how the analysis is performed, and you start seeing more information about the simple class diagram you created. The left pane shows the architectural discovery. Look at the sections under Object-oriented Patterns: you'll see the abstract classes and inheritance tree represented in the UML class diagram.
The architectural discovery feature is important because it provides valuable insight into the overall quality of your code. It can help you identify potential problem areas and structural issues. Identifying and resolving these problems early can lead to a more predictable, robust, and maintainable product in the long run. (For more information see the Related topics section at the end of this article.)

Summary

Rational Software Architect is a tool built on Eclipse that allows developers and architects to leverage the usability features of the Eclipse Platform. However, Rational Software Architect goes beyond the features of a typical IDE -- it provides rich modeling and architectural design and discovery capabilities. In this article, I introduced you to basic concepts about how to use Rational Software Architect, and talked briefly about the Modeling perspective and the different views attached to this perspective. This information is your window into all the modeling capabilities provided with Rational Software Architect.

Nevertheless, I've only skimmed the surface of Rational Software Architect. The simple UML use case diagram creation that I showed you will serve as the foundation for what'll I'll discuss in later articles. In the meantime, you can leverage this information to practice creating other UML diagrams, including sequence diagrams and collaboration diagrams, in Rational Software Architect.

If you're a developer, I recommend that you explore the architectural discovery features of Rational Software Architect in more depth. These features are valuable and help improve the overall
robustness and maintainability of your code. To truly understand and explore all that Rational Software Architect has to offer, consider applying these basic modeling capabilities to your project.
Related topics

- **How to migrate from code-centric to model-centric development using Rational Software Architect** (developerWorks, January 2005) shows how you can develop higher-quality software more efficiently using a model-driven development approach.
- **OO design process: Use cases, an introduction** provides the basics of use-case analysis.
- **The Modeling Structure Guidelines** (developerWorks, January 2005) provides information that will help you better use the concepts described in this article.
- **Driving Iterative Development With Use Cases** explores how to plan a project based on the IBM Rational Unified Process (RUP) using a use-case based approach.
- You can read more about UML 2.0 at the [UML Resource Center](#).
- Get the evaluation version of Rational Software Architect from the [Trials and betas](#) page.
- For technical resources about Rational's products, visit the [developerWorks Rational content area](#). You'll find technical documentation, how-to articles, education, downloads, product information, and more. For specific information about Rational Software Architect, visit the [RSA technical resources page](#).
- Find more product related information by visiting the Rational Software Architect [product page](#).
- For details and more information about the Eclipse 3.0 platform, visit the [Eclipse home page](#).
- Purchase [Rational books at discounted prices](#) in the Rational section of the Developer Bookstore.

© Copyright IBM Corporation 2005

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)