IBM Rational Modeling Extension for Microsoft .NET

Visualizing Microsoft .NET applications

Lalitha K. Kishore
Darpan Saini

March 06, 2007

This article is for developers and architects who are interested in using IBM® Rational® Modeling Extension for Microsoft® .NET to model, design, and develop .NET applications in C#. It assumes that you have a basic knowledge of IBM Rational modeling tools (either IBM® Rational® Software Modeler, IBM® Rational® Systems Developer, IBM® Rational® Application Developer, or IBM® Rational® Software Architect), and of programming in the Eclipse environment.

IBM Rational modeling tools

Rational Software Modeler, Rational Application Developer, Rational Systems Developer, and Rational Software Architect, -- collectively Rational modeling tools -- are built upon the Eclipse integrated development environment (IDE). They provide visual diagramming tools to help you gain insight and knowledge of existing applications or to more easily create and understand new applications.

Rational Software Modeler (or Rational Systems Developer or Rational Software Architect) as extended by Rational Modeling Extension for Microsoft .NET (Rational Modeling Extension, for short) provides functionality to generate C# code from UML models. It also provides code diagramming tools to help you gain insight and knowledge of existing Microsoft .NET applications. This article will describe how to work with Microsoft .NET solutions in Rational Software Modeler (or Rational Systems Developer or Rational Software Architect) and how to create diagrams (Class, Sequence, and Browse Diagrams) from existing C# code and .NET assemblies using code modeling technology. This article does not talk about generating C# code from UML models. Hereafter, Rational Modeling Extension will refer to Rational Modeling Extension as installed into Rational Software Modeler.

What is Code Modeling

What we refer to as simply code modeling is a technology found in IBM Rational modeling tools that helps you to graphically depict code-level concepts using UML Notation but without relying upon separately persisted UML semantics (abstract syntax). There are many benefits of
representing code elements in UML. Most importantly, UML is an open standard and well known to
the software development community. Any complex system can easily be understood by creating
diagrams and views that visually represent the internals of a system. This can easily be done in
3GL domains such as Java and C++ using the code modeling technology.

Code modeling is different from the traditional Round Trip Engineering (RTE) capability provided
by products such as IBM Rational Rose or IBM Rational XDE™, as it enjoys the following
characteristics:

- Uses UML notation (concrete syntax)
- Requires no knowledge of UML semantics (abstract syntax)
- Requires no knowledge of how 3GL semantics map to UML semantics
- Compared to traditional RTE, eliminates overhead of 3GL-to-UML mapping a synchronization
- Depictions of code in UML notation always faithfully, and instantly, reflect changes to code
  and code is updated immediately when edited via diagrams
- Compared to traditional RTE, greatly simplifies team workflows
- Can be used for Mixed-Modeling where one can depict code elements alongside true UML
  semantic elements, as well as depictions of DDL (tables) and other semantic domains

**Overview of Rational Modeling Extension for Microsoft .NET**

Rational Modeling Extension is designed to complement the use of Microsoft Visual Studio®
2005, and the use of the Class Designer feature within Visual Studio (Class Designer is also an
implementation of code modeling as described above). Rational Modeling Extension is not meant
to replace the Class Designer feature. Accordingly, Rational Modeling Extension does not provide
full visual code editing capability like the code modeling support for Java or C++ found in Rational
Systems Developer, Rational Application Developer, or Rational Software Architect.

In the .NET world, that is the job of Class Designer, and Rational Modeling Extension only provides
read-only visualization of C# (based on source code files) and CTS types (based on binaries
contained in assemblies). This is sufficient to provide significant value, by complementing Class
Designer with visualization features that Class Designer does not support.

While Rational Modeling Extension does allow you to visually depict C# (Classes, Interfaces,
Structs, Delegates, etc), using class diagrams just as Class Designer does, it also provides
support for you to review the structure of the application by viewing relationships between
various .NET elements using Browse and Topic diagrams. In addition, it lets you depict typical
or what if interactions between .NET classes in Sequence diagrams. Class Designer does not
support Topic, Browse, or Sequence diagrams.

**Working with Microsoft Visual Studio projects in Rational Modeling Extension**

For the overwhelming majority of .NET developers, Microsoft Visual Studio is the preferred IDE
for writing and debugging C# code. Rational Software Modeler, Rational Systems Developer,
and Rational Software Architect represent the market-leading solutions for designing applications
using UML2. The design of Rational Modeling Extension anticipates this and provides a way in which users can work with Visual Studio projects within Rational Software Modeler (or Rational Systems Developer or Rational Software Architect). Using Rational Modeling Extension, users can visually design their .NET applications in UML, then use transformations to seed/generate code into Visual Studio projects, and from there proceed with code development and debugging using a combination of Visual Studio (including the Class Designer feature) and code visualization within Rational Modeling Extension.

Technically, Rational Modeling Extension uses Visual Studio extensibility APIs to retrieve information from Visual Studio C# projects about their contents, like C# files and assemblies referenced by the projects. The Visual Studio extensibility API can only query and return information from an open Solution. Due to this limitation, the Solution containing the C# projects that are to be modeled in Rational Modeling Extension should always be open in Visual Studio whenever Rational Modeling Extension is in use.

**Importing Visual Studio projects into Rational Modeling Extension**

To work with Visual Studio C# projects, you first have to import the projects into Rational Modeling Extension. This can easily be done using the .NET Solution Import wizard, which you can open from **File > Import > Others > .NET Solution** (as shown in Figure 1).

**Figure 1. .NET Solution Import wizard**

The path to the solution file should be provided in the first page of Import wizard (illustrated in Figure 2). The wizard will parse the solution file to find the C# projects contained in the solution,
and a corresponding Eclipse project with the same name will be created in the Eclipse workspace (illustrated in Figure 3). The importer will also create Eclipse links to the C# files and .NET assemblies used by the project. These links are used to retrieve information about the contents of the C# projects. These links are stored in the .project file of the corresponding Eclipse project.

**Figure 2. .NET Solution Import wizard, Page 1**

![Figure 2. .NET Solution Import wizard, Page 1](image)

**Figure 3. .NET Solution Import wizard, Page 2**

![Figure 3. .NET Solution Import wizard, Page 2](image)

After you successfully finish importing the Visual Studio .NET solution into Eclipse, Rational Modeling Extension will switch to the .NET perspective. You can then use the Project Explorer to view project contents. Note that the project contents can only be viewed from the Project Explorer view (regardless of the perspective), and not with any other view.
A sample view of the imported project is illustrated following. Figure 4 shows the Solution Explorer view in Visual Studio, and Figure 5 shows the corresponding view of the imported project in the Rational Modeling Extension Project Explorer view.

**Figure 4. MS Visual Studio Solution Explorer View**

![MS Visual Studio Solution Explorer View](image)

**Figure 5. Rational Modeling Extension Project Explorer View**

![Rational Modeling Extension Project Explorer View](image)

You can expand either .NET Assemblies or C# files to view the types defined in them, and you can visualize these types into Diagrams (both stand-alone diagrams, and diagrams that reside within actual UML models). You can also visually inspect C# (using statements, C# attributes, Operations, and so on) by expanding the corresponding type enclosing them.
Visualization: usability features provided by Rational Modeling Extension

Visualize menu

The Visualize menu, shown in Figure 6, allows you to add the selected element(s) to a new Sequence Diagram, Topic Diagram, or Class Diagram. You can also explore the selected element(s) using the Explore in Browse Diagram command. The available options, of course, depend on the selection. For example, as shown in Figure 6, if the selected element is a CTS Enumeration then the Sequence Diagram option is not available.

Figure 6. Visualize context menu

The amount of time taken by the Visualize command to Visualize .NET elements depends on factors like the following:

- The size of the C# project
- The number of elements selected
- The depth of the query

In some cases, due to these factors, it may take several seconds to complete the Visualize command.

Browse and Topic diagrams

As mentioned previously, the Visualize menu allows you to explore .NET elements in a Browse diagram. Figure 7 illustrates a Browse diagram, which is a temporary, non-editable diagram that provides a quick way to explore existing elements and their relationships. Browse Diagrams are a nice way to walk through a code base in order to understand its structure and the nature of its interdependencies.
Figure 7. Visualizer Browse diagram

The Browse diagram allows you to select relationships you want to view, and when you click **Apply** a new browse diagram will be generated using the new query. You can also navigate through Browse diagrams by using the navigation buttons on the toolbar. Since the Browse diagram is temporary, it cannot be saved. However, you can save a snapshot of a browse diagram as a regular UML diagram (.dnx) or as an Image file by accessing the **File > Save As** pop-up menu on the diagram surface.

A Topic diagram can be used to quickly create dynamic views of relationships between .NET elements. Figure 8 shows the predefined queries that the Topic wizard provides so that you can view the relationships.
Figure 8. Topic diagram presets

Topic diagrams -- like Browse diagrams -- are query-driven and cannot be edited, but you can customize the query (which causes the diagram to refresh). You can also just refresh the diagram to cause code changes to be reflected on it.

Sequence diagrams

The sequence diagram shows the order in which the messages flow between instances of the .NET elements for a specific interaction. The Visualize menu also allows you to create a Sequence diagram (Figure 9) and create a new interaction.

Figure 9. Sequence diagram
As you can see, you can create various types of messages between lifelines in a sequence diagram, and select an existing operation as a signature of the message. The limitation is that you cannot create a new message type, but can only choose existing messages.

**Show Related Elements**

**Show Related Elements** is a very useful feature that helps you query for related elements on a diagram. In Rational Modeling Extension, the **Show Related Elements** action is available by right-clicking any .NET element on the Class Diagram. The **Show Related Elements** option is present in the **Filters** submenu.

Rational Modeling Extension also provides an enhanced Show Related Elements dialog, which is shown in Figure 10. The dialog provides predefined custom Queries that you can use to quickly view the related elements. By default, the **Details** are collapsed, and only the left pane in the dialog is visible. By clicking the **Details** button, you can view the actual relationships, along with other settings related to the selected query. Any changes made to the queries can be saved for future use.

**Figure 10. Show Related Elements dialog for .NET elements**

You can select **Incoming**, **Outgoing**, or **Both** directions. For example, if you wish to see the super class hierarchy of the selected element, select **Outgoing**. In order to view the subclass hierarchy, select **Incoming**.

Also of note is the **.NET Show Related Elements Filters** Preferences page, as shown in Figure 11.
Figure 11. Preference page for .NET Show Related Elements Filters

The preferences give you the option to filter out .NET Assembly types when executing the Show Related Elements action. Note that this setting will affect all operations that query for related elements (for instance, queries on Browse diagrams and Topic diagrams).

Visualize existing > .NET type menu

A Visualize existing > .NET type menu, shown in Figure 12, is available on the pop-up menu for the diagram surface. This allows you to Visualize any existing .NET type onto the diagram.
When you select this menu, it launches a Visualize .NET type dialog that allows you to select the .NET type that you want to visualize onto the diagram. This menu saves you from having to first find the .NET type in the Project Explorer and then drag it onto the diagram, or accessing the Visualize > Add to current Class Diagram command. The Visualize .NET type dialog only allows you to select a type that is part of the workspace.

Navigating to C# code from Rational Modeling Extension

You can easily navigate to the appropriate line in the C# file in Visual Studio by either double-clicking the element in the Project Explorer, or double-clicking a visualized C# element from a Class diagram (note that this does not work in a Browse Diagram where the double click gesture has a specialized meaning for navigating through a series of diagrams). When you double-clicked a C# element in Rational Modeling Extension, the control will switch to Visual Studio and the cursor will be placed at the appropriate line in the C# file.

Getting C# code updates into Rational Modeling Extension

As stated earlier in the article, the current version of Rational Modeling Extension does not support visual editing of C# code but the user can refresh the imported projects and diagrams in Rational Modeling Extension to quickly reflect any code changes made in Visual Studio. To do this, follow these simple steps.

1. First, make your modifications to the C# files in Visual Studio.
2. Save the project in Visual Studio.
4. Right-click the root folder of the Visual Studio project that needs updating and click Refresh.

When you click Refresh, the updated project contents are retrieved from Visual Studio, and the corresponding elements in the Project Explorer and in the diagrams are updated.
Conclusion

Rational Modeling Extension offers a host of features with respect to visual modeling for MS .NET. As noted earlier, in addition to the C# and CTS diagramming and visual exploration capabilities described in this article, there are also the following capabilities, which are topics for other articles:

- transforming from UML to C#
- transforming from C# to UML
- using the combination of UML models, transformations, and code modeling to practice a Mixed Modeling approach to Architecture Management
- using a combination of UML to C# and C# to UML transformation together with IBM Rational model fusing technology to practice an Architecture Reconciliation approach to Architecture Management

Rational Modeling Extension makes .NET development a whole new experience, with great support for designing the application using UML Modeling techniques. Understanding the structure and design of an existing application is made easy by using Class diagrams, Sequence diagrams, Topic and Browse diagrams, in addition to commands such as Show Related Elements.
Related topics

- **Rational Modeling Extension for Microsoft .NET page**: Find product information about IBM Rational Modeling Extension for Microsoft .NET.
- Download **evaluation versions** of IBM Rational Software Modeler, Rational Software Architect, Rational Systems Developer. IBM Rational Modeling Extension for Microsoft .NET should be available as a trial download soon.
- **IBM Rational Software Modeler product page**: Find technical documentation, how-to articles, education, downloads, and product information about Rational Software Modeler.
- Visit IBM's **Pattern Solutions** and find out what IBM is doing around patterns and reusable assets.

© Copyright IBM Corporation 2007
**Trademarks**
(www.ibm.com/developerworks/ibm/trademarks/)