UML Model Metric Analysis rules-based generation of class diagrams

Manoj Paul
Manish Paul
August 06, 2013

Unified Modeling Language (UML) Model Metric Analysis rules-based generation of class diagrams is a new feature in Rational Software Architect. This feature helps you to compartmentalize and create diagrams to better analyze underlying architecture. It also provides a framework for you to extend and contribute custom model analysis rules that can be used for the generation of class diagrams.

Introduction

Rational® Software Architect is a modeling and development tool that uses Unified Modeling Language (UML) to design the architecture of software applications and web services. The diagram support that Rational Software Architect provides enhances the capability of software application architecture design. Rational Software Architect also provides reverse engineering support that enables you to transform your code into UML models. Reverse engineering support in Rational Software Architect lets you capture the architecture of the software application as a UML model but does not generate a diagram. It might become difficult to analyze the architecture without the diagram.

Rational Software Architect 8.5 introduced UML Model Metric Analysis rules-based generation of class diagrams. This feature lets you compartmentalize and create diagrams to better analyze the underlying architecture. It also provides a framework for you to extend and contribute custom model analysis rules that can be used for the generation of class diagrams.

Model analysis in Rational Software Architect

It is imperative that you can see the diagram of the architecture of a software application based on a parameter. This is also true for complex systems where it is necessary to be able to filter the elements that should be shown on the diagram based on some metric. For example, you might need to explore only the depth of a classifier in its inheritance hierarchy in a model. This kind of parameter can be used to seed the diagram to explore the various aspects of software design.

Model analysis is a capability in Rational Software Architect that simplifies model automation and improves design quality. You can use model analysis to access capabilities, such as automated
model review, model metrics, and structural analysis, which you can use daily in the development process.

UML Model Metrics Analysis rules provide varying types of numerical information about the model. For example, you can discover the number of attributes or operations in a classifier, the level of instability of a package, or the depth of a classifier in its inheritance hierarchy. UML diagrams can be seeded based on the parameters with respect to an analysis rule of your choice.

For further details, refer to the "Analyzing UML models with static analysis tools" topic in the IBM Rational Software Architect, version 8.0, Information Center.

**Generating class diagrams based on UML Model Metric Analysis rules**

Assume that you have a model with UML classes associated through relationships, as shown in the left side of Figure 1. You want to generate a diagram from this model based on UML Model Metric Analysis parameters because they provide a seeding element to use to generate the diagram. You also want to generate the diagram with the classifiers that have at least one ancestor (UML classes Class2 and Class3 are such elements in the model). With these elements, the result should be the association relationship up to level two. With such a requirement to generate the diagram, you can leverage the UML Model Metric Analysis support to get the seed elements for the diagram and expand the diagram with the relationship that you want up to the desired level of that relationship. As in the example, UML classes Class2 and Class3 can be picked as the seed elements for the diagram. The UML diagram in right top part of Figure 1 shows how UML class Class2 and all other elements associated with it, and in the UML diagram in right bottom part of Figure 1, UML class Class3 has been used as the seed element to generate the diagram.

**Figure 1. Diagram Generation based on seed element**

Diagram generation support is accessed through a context menu. A pop-up menu on a model or a package for the class diagram generation is provided for generating UML diagrams based on UML model metric analysis rules, as shown in Figure 2.
Figure 2. Diagram generation context menu

From the subsequent dialog (shown in Figure 3), select the rule(s) based on which diagram(s) to generate.

Figure 3. Diagram generation properties dialog
Details of the "Generate Class Diagram" dialog

UML Metric Analysis Rules section

The number of rules shown in this dialog is a smaller subset of UML Metric Analysis rules. If you want to see more UML metric analysis rules in this dialog, you need to implement and register an interpreter for the rules through extension point. Refer to the Contributing rules for diagram generation section for details.

The process to create customized UML Model Metric rules is explained in the wiki article, "Extending Model Analysis in Rational Software Architect - Part 2: Extending the UML Model Metrics." After you create the rules, you can use them to generate diagrams through interpreter registration For each selected rule, separate class diagram(s) are generated.

Properties section

Each rule has the following additional properties that you need to specify. These additional properties help to establish the nature and content of the diagram generated for the rule.

Generate diagram for individual elements

Option to indicate that for a rule whether one class diagram needs to be generated for selected elements OR if separate class diagrams should be generated for each selected element.

Diagram name

Name of "class diagram" to add to the UML model. If "Generate diagram for individual elements" is selected, then the class diagram name that corresponds to each element is: element name suffixed to entered diagram name, that is <Diagram Name>_<element_name>.

Lower bound and upper bound

These are numerical figures that filter out element(s) of interest for which diagram(s) to generate. Elements with metrics that are within the range of these values are selected. This helps to generate a precise and concise diagram of interest. You can leave these fields blank if you don't want to filter out elements based on metric value. Another acceptable filter is to have one of the bounds set. In such a condition, the system filters out elements based on the bound set.

Relationship types

You can select the relationships needed to plot on class diagrams. This property helps the underlying framework to query only those elements related to the selected elements based on the relationships selected.

Expansion direction

Indicates whether the incoming, outgoing, or both, are considered for querying related elements.
Level
Indicates the level up to which underlying framework query for related elements. With an increase in level, the performance might be affected, depending on the size and complexity of the model.

Selection diagram generation section
These options allow you to generate a fresh diagram or merge the changes.

Overwrite existing diagram
Selecting this option overwrites any diagram in the model with same name as specified in the "Diagram Name" field with the new diagram. If there is no existing diagram, the system creates a new diagram. You should select this option when you generate a fresh class diagram.

Figure 4. Generating a new diagram

Merge with existing diagram
This option uses the compare-merge functionality to show differences between existing and newly generated diagrams. You can choose to accept or reject the deltas, as shown in Figure 4. Select this option when you want to preserve the modifications made on previously generated diagrams while also reconciling the existing diagram with the latest changes in the model.
Figure 5. Accept or reject the delta

Figure 6. Updated diagram after accepting the change and successful merge

Contributing rules for diagram generation

The number of rules appearing in the diagram generation UI is a smaller subset of the UML Metric Analysis rules. If you want to see more UML Metric Analysis rules in this dialog, implement and register an interpreter for the rules through the extension point provided by the diagram generation framework. A UML Metric Analyzer rule is available in the Diagram Generation UI for selection, after an interpreter is registered against the rule. The next section, Concept of the interpreter, provides more detail on this and an example scenario for contributing Model Metric rules for diagram generation is also explained.

Concept of the interpreter

Analysis rules, when executed, return the analysis results in some format. The results refer to the model elements along with the metric values. They do not provide the actual UML element for
which the metric value was determined, nor the element type (such as generalization, realization), which was used in determining the metric value. To generate diagrams out of the analysis result, you need the actual UML elements and the element kind used by the analysis rule. You also need an interpreter to do this job. The extension point, described in the next section, allows you to register interpreters against analysis rules.

**Extension point for registering the interpreter**

An extension point `com.ibm.xtools.analysis.interpreter` is provided to allow you to register an interpreter to interpret the result of an analysis rule. An interpreter is registered against each analysis rule. The extension point needs to be provided with the rule id against which the analysis result was obtained, along with the kind of relationship used in the analysis rule and the direction of the relationship. The interpreter's job is to retrieve the actual UML element from the selected model from the element reference in the analysis result.

**Example**

The UML Model Metric Analysis framework contributes several rules. One rule provides the number of interfaces that a class implements. If you want to leverage this metric for diagram generation, you need to register an interpreter for this rule with the id `com.ibm.xtools.analysis.uml.metrics.rule.NumberOf Interfaces`. The interpreter class `com.ibm.xtools.auto_diagram.services.interpreter.UMLAnalysisResultInterpreter` interprets the results fetched by the analysis rule. The extension point also says that the `elementKind` is `Realization` and `direction` is `OUT`.

**Listing 1. Extension point to register an interpreter**

```xml
<extension point="com.ibm.xtools.analysis.interpreter">
  <analysisInterpreter
    class="com.ibm.xtools.auto_diagram.services.interpreter.UMLAnalysisResultInterpreter"
    id="com.ibm.xtools.analysis.umlmetric.interpreter.realization">
    <analysisRule
      direction="OUT"
      elementKind="Realization"
      ruleId="com.ibm.xtools.analysis.uml.metrics.rule.NumberOfInterfaces">
    </analysisRule>
  </analysisInterpreter>
</extension>
```

After you have the registered the interpreter against a rule, that rule becomes available in the Diagram Generation user interface as a checkbox item. It can be used for generating a diagram that satisfies the properties specified in the UI against that rule, as shown in Figure 6. After you select the rule, the properties specific to this rule are enabled. You can specify the Lower Bound and Upper Bound for that metric. In this example, Lower Bound is specified at 2 and Upper Bound to 4. This gives all the classes in the model two to four interfaces. The classes are fetched by the interpreter and are used to seed the diagram to be generated. You can choose to generate individual diagrams for individual seeded elements or a combined diagram. If desired, select additional relationship types for those elements and the level up to which you want to see them in the generated diagram.
Figure 7. Contributing rules to diagram generation

Figure 7 shows a sample model that has classes with different numbers of interfaces implemented. Class1 implements one interface, Class2 implements two, and Class3 implements five. If you choose the option specified in Figure 6, the result is only Class2 as the seed element for diagram generation. Figure 8 shows the resulting diagram.

In addition to the default relationship ("Realization"), the diagram also shows other relationships such as "Generalization" as specified in the diagram generation properties.

Figure 8. Sample model for diagram generation
Conclusion

Inheriting a project from another team is challenging. It is difficult to decipher and analyze the underlying architecture without the supporting artifacts. Rational Software Architect provides an out-of-the-box capability to reverse-engineer the code. This feature helps you generate diagrams and compartmentalize the elements based on topics (defined by model analysis). This helps you comprehend the design. Because the diagram-generation capability is extensible, you can specify custom rules for diagram generation.