from The Rational Edge: Conducting an impact analysis before making changes to an existing system is a recommended practice to ensure that all system interdependencies are known, and that changes to one or more components will be understood in terms of the effects on other components. This article describes specific techniques for using the IBM Rational toolset for performing an impact analysis.

If you have the luxury, or the requirement, to do an impact analysis before changes are made to your application, your project is likely to be more successful than if you did none. An impact analysis will confirm what part of the application you need to change, and it will tell you how much of the application may be affected by that change. It will also help you determine what level of regression testing may be required. In addition, an impact analysis will help you estimate the time required to make the proposed enhancement.

Imagine a situation where an architect or developer has been working on an application for quite a long time, and therefore has a lot of this kind of information stored in his or her head. In this case, this worker may not need to do a formal impact analysis. But in many cases applications are very large and have many developers working on them. Also, if the application has been around for a while the architecture can become unwieldy. In this case, an impact analysis is a valuable way to gain a better understanding of how a proposed enhancement will effect the other parts of the application.
While there is not an "Impact Analysis" feature in any Rational Architecture Management tool, there are some very nice capabilities in the Rational toolset that will help you determine the effects of the proposed changes on your application. There are specific analysis features in various Rational tools that you can use, and the integration between these tools enhances the impact analysis capability. However, even if you don't own all of these tools you can use them independently to assess the impact of a change.

**Rational tools for impact analysis: An overview**

If you use visual modeling, and your UML models are in sync with the application code, you can start looking at your models to ascertain where a potential change might be made and what other parts of the application would be affected. UML 2 structure diagrams, such as a Class diagram, show relationships between classes or components. You can then see what other components might be affected by the change. If there are many other components that use the component to be changed, then you can see the extent to which the change will potentially impact the larger application.

Let's take a look at what IBM® Rational™ tools are available for impact analysis; in the next section, I will discuss the use of each tool in more detail.

**IBM Rational RequisitePro**

RequisitePro is a requirements management tool that integrates with other architecture management tools. You can greatly enhance your ability to assess change to your application by using your UML models along with the integration of RequisitePro and IBM Rational Software Architect (or IBM Rational Software Modeler). This integration allows you to easily link requirements in RequisitePro to model elements in Rational Software Architect or Rational Software Modeler. Rational Software Architect also has architectural discovery tools that I will talk about later.

RequisitePro also integrates with WebSphere® Business Modeler and IBM Rational Application Developer (version 7). The integration of RequisitePro with any of these four tools allows Analysts, Architects, and Developers to maintain traceability from requirements directly to models and code. Using the RequisitePro interface within Rational Application Developer, for example, a developer can see which requirements are traced to various parts of the application. From this they can determine what impact on other parts of applications may result from a change and what risks are associated with that. RequisitePro keeps track of the linkage between model or code artifacts and requirements. So when you find where you need to make changes in your application, you can follow the traceability to see what components (as requirements in RequisitePro) trace to that component you want to change.

**IBM Rational Application Developer**

Developers can use Rational Application Developer to look at their application code visualized as UML to gain a better understanding of existing relationships between classes and/or interfaces. This helps them understand where potential changes might need to be made and what effects they might have on other classes and interfaces. They can view their code in Class Diagrams to
see these relationships. They can also view complex methods via Sequence Diagrams so that they can quickly see what other methods, on what classes, are called from a complex method. If developers need to further see how the application reacts at runtime, they can use the runtime analysis feature to profile their application. Describing precisely how to do that is outside the scope of this article, but I have included a reference in the resources section below for more information.

There is also a static code analysis feature in Rational Application Developer that will look at your existing code for best practice violations. While this may not seem applicable to impact analysis, this features gives you the ability to create your own rules that can help you better understand your application. For instance, you can create your own static analysis rule to run against a large application to find where a particular interface or class is used. You can visualize all of your code and look for references, or you can do a Find, looking for text in your code. However, creating a rule is repeatable and can be shared with other developers.

Rational Software Architect

Rational Software Architect has all of the same features mentioned for Rational Application Developer and also extends the code analysis feature by adding rules to look at the structure of your application code. This allows architects to find whether any patterns or anti-patterns exist. It also contains a category of rules to look at UML Models for traceability.

IBM Rational Data Architect

Rational Data Architect allows you to discover and visualize the structure of existing data sources in order to help you evaluate what effects the proposed changes will have. See the demo referenced in the Resources section at the end of this article for more information on Rational Data Architect.

Exploiting traceability with Rational RequisitePro

Where is the first place to look when investigating an enhancement request? If you have a requirements management database, it should be there. Rational RequisitePro allows you to keep track of project and application requirements at many levels. It also allows you to trace different types of requirements to each other and view those relationships in a matrix. In this way, you can follow the traceability as business goals are traced down to application features.

The article from the February 2004 issue of The Rational Edge that I list in the References at the end of this article contains information specific to the use of RequisitePro and impact analysis. What I will focus on here is using RequisitePro and the integration with Rational Software Architect (RSA) or Rational Application Developer (RAD) to create the traceability links that RequisitePro manages.

To enable the integration of RequisitePro with Rational Application Developer, Rational Software Architect, Rational Software Modeler (RSM), or WebSphere Business Modeler (WBM), you only need to open the Requirements perspective from within these tools. To ensure the capability is enabled, you can open the Preferences window by choosing the Windows dropdown menu and then "Preferences." From there expand "General" and click on "Capabilities." You can then enable the Requirements Management integration as shown in Figure 1.
You will also want to ensure that label decoration for RequisitePro is enabled. With the Preferences window open, expand "Appearance" and click on "Label Decorations." Ensure that "RequisitePro Requirement Decorator" is selected. In Figure 2, you will notice the elements in the workspace that are linked to a requirement in RequisitePro by the little arrow decorating the label for that element's type icon. If there is a link in an unknown state, there will be a red question mark on the type icon. If you see a red question mark on an element in your workspace, check the Requirement Link Problems view. It will give you more information regarding what is wrong with the link.

When you open the Requirements perspective from within RAD, RAD, RSA, or WBM, you will be able to link projects, models, model elements, or code to requirements that are stored in the RequisitePro database. This means that going forward, you will have traceability from that requirement to the model or code (or better yet, both) that implements that requirement. You can build traceability matrices in RequisitePro that you can view during an impact analysis from within RAD, RSM, RSA, or WBM. These queries can be run from within the RSA/RSM/RAD/WBM Requirements perspective and the results will show in the Requirement Query Results view. This is where you look for associations between the requirements in RequisitePro and the elements that implement that requirement. Figure 2 illustrates an example: Assume you need to make a change to add support for a new credit card. You could follow the traceability link between Business Rule 7 and the files and models that implement that Business Rule.

When you open the Requirements perspective you will see a new view, the Requirements Explorer view. This will allow you to open RequisitePro databases to be used in the Requirements perspective. You can configure the link policy and other properties by right-clicking on the open RequisitePro project and choosing "Properties." Other views that are used in the Requirements perspective are the Requirement Trace, which shows the trace relationship for the requirement.
that is highlighted in the Requirements Explorer view, and the Link Clipboard view, which is used to see the elements that you are saving for linking.

![Image of Requirements Explorer view](image)

**Figure 2: Following the traceability link between Business Rule 7 and the files and models that implement that business rule**

To add a link between a requirement in RequisitePro and an artifact in your RSA, RSM, RAD, or WBM workspace, all that you have to do is drag and drop the requirement onto the artifact. This will create the link in RequisitePro, not in the RSA/RSM/RAD/WBM. It will create a new "proxy" requirement in RequisitePro that represents the artifact in your workspace and will create the link between the proxy requirement and the requirement. See Figure 2 for an example of a proxy element "FILE3", "FILE4", and "DIAGRAM1." An exception to this is if you create a link between a use case in a UML model and a requirement in RequisitePro. In that case, there is no proxy requirement created.

There is much more I could describe regarding RequisitePro and impact analysis, but the goal of this article is primarily to present a few general techniques with specific tools you can use for impact analysis, not to go in depth with RequisitePro.

**Using diagrams with Rational Application Developer**

So, you need to look at your application to see what classes and interfaces are related to a particular class that you think you need to change. There are a couple of ways to visualize your Java or C++ code to view it as UML. Please note that C++ code visualization is only supported in RSA, whereas Java visualization is supported in both RAD and RSA.
One easy way to do this is by using the Browse Diagram. This is a diagram that is not saved, but it allows you to browse through your code in a browser-like fashion. To view your code in a Browse diagram, right-click on your class and choose Visualize > Explore in a Browse Diagram. When the diagram opens, you can select which relationships you want to view (Extends, Uses, Implements, References, and Declares) and the depth of relationships. Then click Apply. The diagram is clickable and you can browse to see the relationships for another class by double-clicking on that class. Then you can use the Forward, Backward, and Home buttons to navigate through the diagrams you have viewed, as shown in Figure 3.

![Figure 3: Using the Browse diagram to visualize code as UML notation](image)

The Topic diagram offers another approach. A Topic diagram is similar to a Browse diagram, except that a Topic diagram is saved -- as a .tpx file -- in your project. The Topic diagram is actually a query into your application code, so if your code changes the Topic diagram will reflect that change. You create a Topic diagram the same way you create a Browse Diagram, from the context menu of a Java type.
You also have the ability to create a Class diagram that is saved into your project. The difference is that a Class diagram will only show the classes you add to it and is a way to visually edit your Java code. You can make changes to the classes in the Class diagram, which will automatically make changes to your code. Remember that in Rational Application Developer all these diagrams -- Browse, Topic, and Class -- are not models. Your Java code is behind the diagram, not a UML model.

What if you want to dive deeper into a method, but don't want to navigate the source code? You can create a Sequence diagram of the Java code in that method. This is a great tool for investigating long methods that have a lot of interaction with other classes, as well as for documentation of your methods. To view the method as a Sequence diagram, expand the class in the package explorer, right-click on a method, and choose Visualize > Add to New Diagram File > Static Method Sequence Diagram. It works with both static and regular methods, but the Sequence diagram is static. Figure 4 shows a static method Sequence diagram for a method in a Swing Class, and you can see all of the other methods that this method calls on the same class and other classes. Notice the Outline view shows you a smaller picture of your diagram, and a gray box shows you where you are within that diagram.

Figure 4: A static method Sequence diagram for a method in a Swing Class

Another way to visualize your code is to generate UML diagrams with your JavaDoc. Just like a Browse diagram, all of the classes, interfaces, and packages are clickable, and you can browse to the JavaDoc documentation for the class, interface, or package that you clicked on. To get UML embedded in your JavaDoc, do the following: From the Java perspective, make sure your project is selected and then choose the Project drop-down menu. Select "Generate JavaDoc with Diagrams" > Automatically. You will see a window similar to Figure 5.
Figure 5: Generating UML diagrams with your JavaDoc

You need to tell RAD what javadoc.exe you want to use. The example shown earlier in Figure 2 uses the javadoc.exe that ships with the WebSphere Application Server version 6.1 test environment. Select the diagrams you want generated and also the image type. If you click "Contribute diagrams and diagram tags to source," it will place the diagram images in the source code and include new diagram image tags. This means that in the future you can run "Project > Generate JavaDoc with Diagrams > From Existing Tags" instead of "Project > Generate Javadoc with Diagrams > Automatically."

Click Finish and a new folder will be created in your project named "doc". You can view the generated JavaDoc by expanding the doc folder and opening the index.html with a Web browser. The UML diagrams are listed on the JavaDoc page for each Class and Interface and shows the relationships that Class or Interface has with other Classes and Interfaces. You can navigate through the JavaDoc by double-clicking on the UML Diagram. Figure 6 shows an example of the JavaDoc with UML Diagrams.
Rational Application Developer also has an analysis feature that interrogates your source code, looking for areas where you have not applied best practices or where you have introduced performance issues. This static code analysis has 212 rules in the following categories: Design Principles, Globalization, J2EE Best Practices, J2EE Security, J2SE Best Practices, J2SE Security, Naming, Performance, and a Private API check. You can also add your own categories and rules. If you go to Windows > Preferences and expand Analysis, you will see Custom Rules and Categories. From there you can Add a Category, then add rules to that category. There is a Custom Rule wizard to walk you through the process of creating new rules.

This is a very helpful feature for reviewing your code quality. Rational Software Architect extends this code analysis capability by adding an "Architectural Discovery" ruleset. I will go through this Analysis feature in the next section.

There is also Java Runtime Profiling in Rational Application Developer that performs runtime analysis of Java code. This is outside the scope of this article, but I highly recommend that you investigate this feature on your own.

**Architectural impact analysis with Rational Software Architect**

Rational Software Architect allows you to run a code review on your application that will help you understand the strength of your architecture. The "Architectural Discovery" ruleset contains twenty-four rules. The subcategory titled "Design Patterns" contains rules that will search your code for the implementation of seven "Gang of Four" design patterns. The subcategory "Object-Oriented Patterns" will search your code looking at the OO structure and showing you abstraction and an
inheritance tree. The anti-patterns provided in the "Structural Patterns" subcategory offer rules that help you look for implementations of things such as Breakable, Butterfly, and Hub that may result in brittle architecture. The "System Patterns" category contains one rule that will show you the package structure in your application.

To use this analysis feature, select the project that you want to review and then right-click and choose "Analyze." Some perspectives allow you to choose the Run dropdown menu and then choose "Analysis." Either way you will get a window similar to Figure 7.

![Figure 7: First step in running a code review in Rational Software Architect](image)

From this window click the New button; then you will have the option to select the name and set the scope of the analysis. If you click on the Rules tab, in RSA you will see the option for Architectural Discovery for Java. If you select the category, then all twenty-four rules within that category are selected. Click Apply, and then Analyze to run the Analysis. The analysis configuration will be saved and can be easily re-run from the Analyze option of the context menu. The Analysis Results view, as shown in Figure 8, will display when the Analysis has finished running.
Figure 8: The Analysis Results view will display when the Analysis has finished running

The Analysis Results view shows the categories chosen for Analysis; the bracketed information shows how many results were found and how long the analysis took to run. When you expand the category, you will see the sub-categories with the same information for that sub-category. If you expand the sub-category further you will see each of the results listed. You can open the result by double-clicking on it and you will see a visualization as shown in Figure 9.
As noted earlier, structural patterns are what we call "anti-patterns." In some cases you have built or use a framework that your whole application uses and it may show up in one of these categories. In that case you can right-click and choose to Ignore the result. Otherwise, you should take note if the implementation of three common anti-patterns is detected: Breakable, Butterfly, and Hub. A "Breakable" is a component or package that depends on too many other components. Conversely, a "Butterfly" is a component or package that has too many dependencies. Combining both, a "Hub" is a component or package that has both too many dependencies and too many dependants. These are determined at the global level by a percentage of components in the project and at the local level by immediate dependencies. The defaults are 15% and 10% respectively, but are both configurable in the analysis configuration that I described earlier and illustrated in Figure 7.

**Database impact analysis with Rational Data Architect**

Here is what the IBM Web site says about Rational Data Architect: "Impact analysis will list the dependencies on a specific element. The results are visually represented and also displayed in report format for easy viewing." The demo I've listed in the Resources section, "Managing Database Design Changes with Rational Data Architect," provides good information on doing database impact analysis.

**Summary**

Rational RequisitePro is not only a great tool for requirements management; it is also good for doing impact analyses. The analysis tools in Rational Application Developer and Rational Software Architect are code-based, which means they look at your code to do the analysis or to build the visualizations. Additionally, Rational Software Architect can be used to look at Models to investigate the impact of a proposed change. While Rational Software Modeler does not contain an IDE and therefore doesn't have the code based analysis tools mentioned in this article, you can still do impact analysis if your models are representative of your application code.

**Resources**

Discuss this article now!

Dear Reader: A new forum has been created specifically for Rational Edge articles, so now you can share your thoughts about this or other articles in the current issue or our archives. Read what your colleagues the world over have to say, generate your own discussion, or join discussions in progress. Begin by clicking HERE.

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