

Project management with Rational Team Unifying Platform

Skill Level: Introductory

[Martin C. Brown](#)
Freelance writer

20 Aug 2004

In this tutorial you'll get a look at the Team Unifying Platform (TUP). TUP is a combination of the main Rational tools used in a team environment coupled with a further management and collation tool, called the Rational ProjectConsole. ProjectConsole collects information and statistics from the other tools in the Rational suite and provides a Web-based interface to the information.

Section 1. Before you start

About this tutorial

The Rational® Unified Process (RUP) is a methodology for developing application software. The RUP defines the methods, documentation and processes required to build the application and to ensure that information is distributed to the various members of the software development team. Although you can follow the RUP processes, the development process is much easier if you use the automation tools provided by Rational Software.

Rational tools such as Rational RequisitePro, Rational XDE and Rational ClearCase are available individually. They can also integrate with each other. However, in a team environment where there are many members of the software development process who need to access and use information from the different applications, it can become difficult to effectively share the information. Furthermore, as a project or program manager, it can be difficult to collate, manage and monitor the progress of development.

In this tutorial, you're going to take a look at the Team Unifying Platform (TUP). TUP is a combination of the main Rational tools which are used in a team environment coupled with a further management and collation tool, called the Rational ProjectConsole. ProjectConsole collects information and statistics from the other tools in the Rational suite and provides a Web-based interface to the information.

Key topics covered include:

- Overview of the Rational system, including TUP and RUP.
- Walk through of a typical project and where the different components are used.
- How to use and deploy the ProjectConsole to provide status information.
- How to build and produce reports from the system that can be used by the team to monitor progress.

You're also going to get your feet a little wet by looking at some of the applications in the TUP to demonstrate how the information is built and exchanged between team members.

Throughout the tutorial, use the ClassicsCD.com sample data that is provided with the TUP.

Prerequisites

Individual tools within the Rational Suite are available from the [developerWorks download site](#). The Team Unifying Platform is only available through your local IBM supplier.

Section 2. Rational system

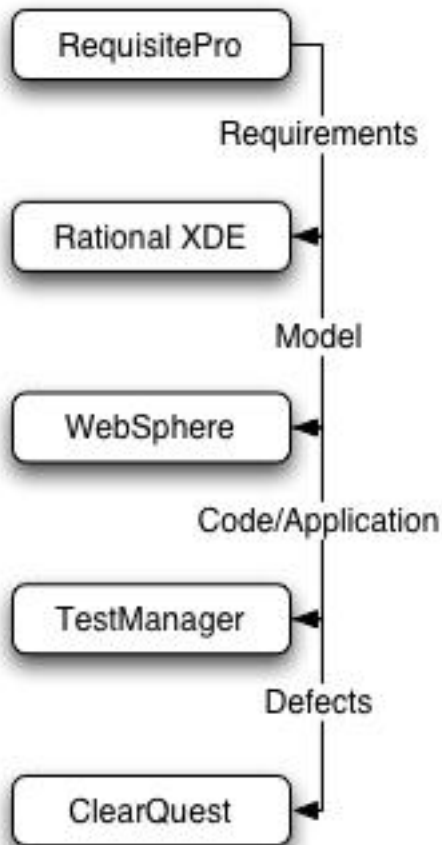
Team-based application development with RUP

Developing software in a team environment generally involves a lot of cooperation, discussion and distribution and dissemination of information between all the members of the team. At each stage of the Rational process you need to share and exchange large quantities of information among different team members. Once the requirements have been collected, these need to be passed on to the designers and analysts who build the model, which then needs to be communicated to the

developers who will turn this model into code. Even testing requires information about the components and the model to be communicated so that the faults and functional tests can be targeted towards the original requirements and components of the application design.

Figure 1 shows the basic sequence involving the packages and information distributed.

Figure 1. Information cascade in development projects



The role of the project manager is to help manage and disseminate the information between the different members, and this can easily become a full time job. Using the Rational Unified Process, you can simplify and standardize on some of the information. For example, use-cases become the standard method for defining the goals and aims of the project. Using Rational Software can also help, because it provides a mechanical method for tracking information through the various stages of the RUP.

But how do project managers and development team members make use of this information to help in the development process? One way is to use the Team

Unifying Platform.

The Team Unifying Platform (TUP) is an extension of the ideas of the RUP and incorporates the software that supports the different stages of development. In addition, the TUP also includes the ProjectConsole, a Web-based application which aggregates information from a range of sources, including but not limited to the software included in the TUP, into a single source. With all the information about a project in a single location it becomes much easier to share information about a project across the development team, and as a project manager you can build reports and graphs which will show, at a glance, the status and progress of your project.

In this tutorial we're going to have a look at the individual applications within the TUP, what information they generate, and how this information is used by the next member in the team to continue the development process. We're also going to have a look at how the ProjectConsole can be used to generate reports and information that can be used by project managers to monitor the progress and activity in a project. Let's start by covering the basic components of the TUP itself.

Rational Team Unifying Platform

The Rational Team Unifying Platform is the collective name for a suite of tools which enable you to define, control, and manage the development of an application using the techniques provided by the Rational Unified Process. Each of the tools has a specific role in the process of developing your application and as well as supporting these individual stages, the tools also communicate and integrate with each other in order to transfer and link relevant information between each stage.

For example, all projects start with a list of requirements, once the requirements are built into an application you can create a link between the original requirement and the portion of the developed application, typically a class, object instance, or method, which implements the requirement. Now when the requirements change, you know which object implementation needs to be altered. Spreading out further from this integration, changes to the implementation would be further recorded, and if defects were identified as related to a specific requirement, then these too would be traceable back to those original requirements.

These are in fact features common to all of the management tools in the Rational Suite and in the Team Unifying platform, which incorporates many of the tools into a single package. The Team Unifying Platform Suite consists of five major applications which are used during the management and development of an application:

- Rational RequisitePro
- Rational ClearCase Client

- Rational ClearQuest
- Rational TestManager
- Rational SoDA

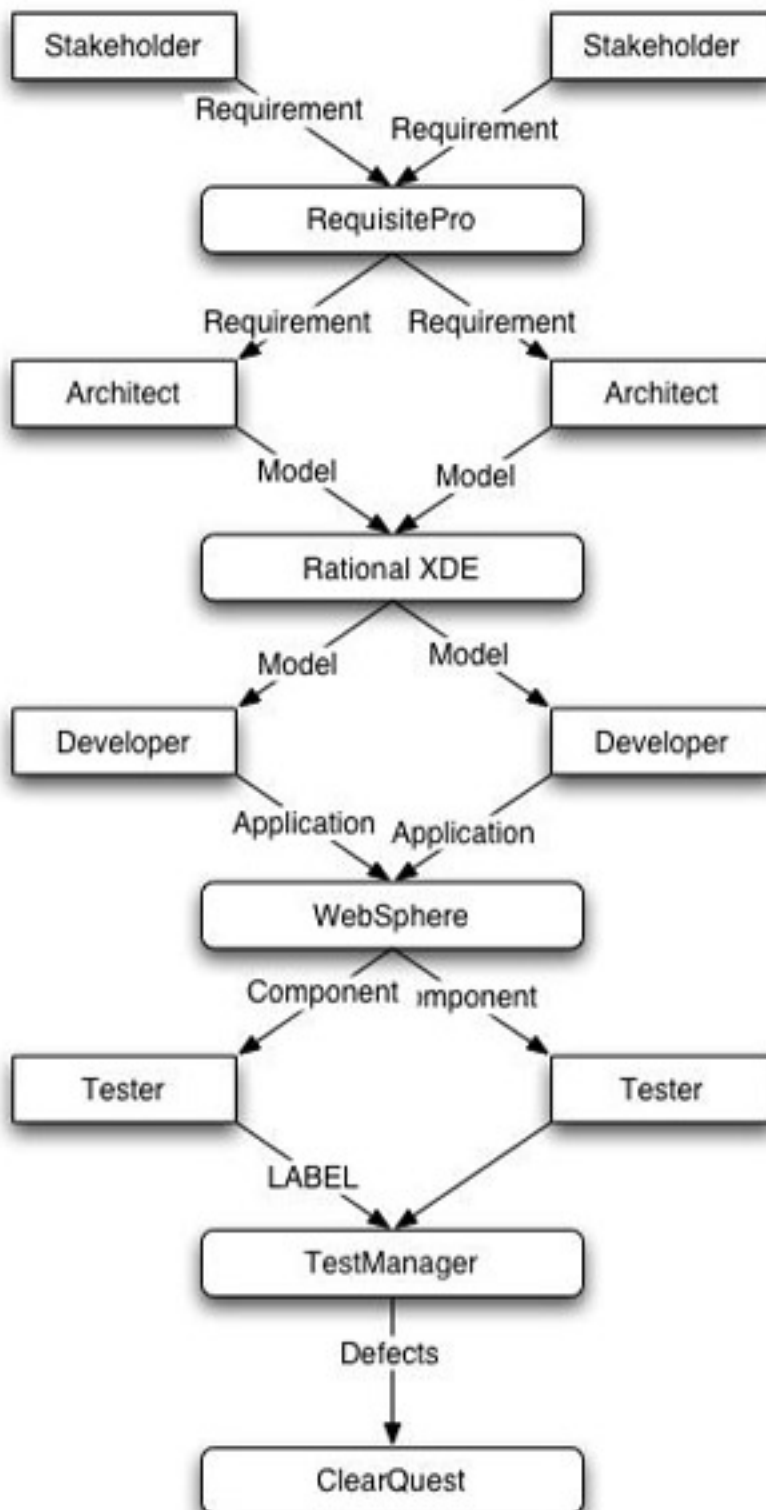
The Team Unifying Platform provides all of these tools in their full versions. A further application, the ProjectConsole, is used to pull information together from all of the applications so that you can view, manipulate, and analyze the status of a project through a single interface. You can monitor progress on different components and generate reports and statistics on different aspects of your project.

Section 3. Rational tools in a team environment

Overview of the sequence

The Rational Team Unifying Platform consists of a range of tools that help with the different steps in the process of developing an application. Integration and the exchange of information between these steps helps to simplify the process, even if used by a single person. Figure 2 shows a basic diagram of the sequence and how the various software tools are used in each step of the process.

Figure 2. Diagram of development sequence



Within a team environment, the ability to exchange between these tools simplifies the process. Multiple stakeholders can contribute requirements into RequisitePro. Members of the design team can access the requirements and produce the model and the code that makes the application, and other team members can use TestManager and the components produced by the developers and the requirements to build tests and strategies to use through TestManager.

Because you can spread the load over a team of people in each section and have them automatically access the information, you can skip the meetings and constant exchanges of information. Instead of meeting to discuss the model between analyst and developer, the developer can view the application model that was built, and even review the requirements that led to the model being created. Not only does it save time, it also provides a handy suite of reference material that can be referred to throughout the development process

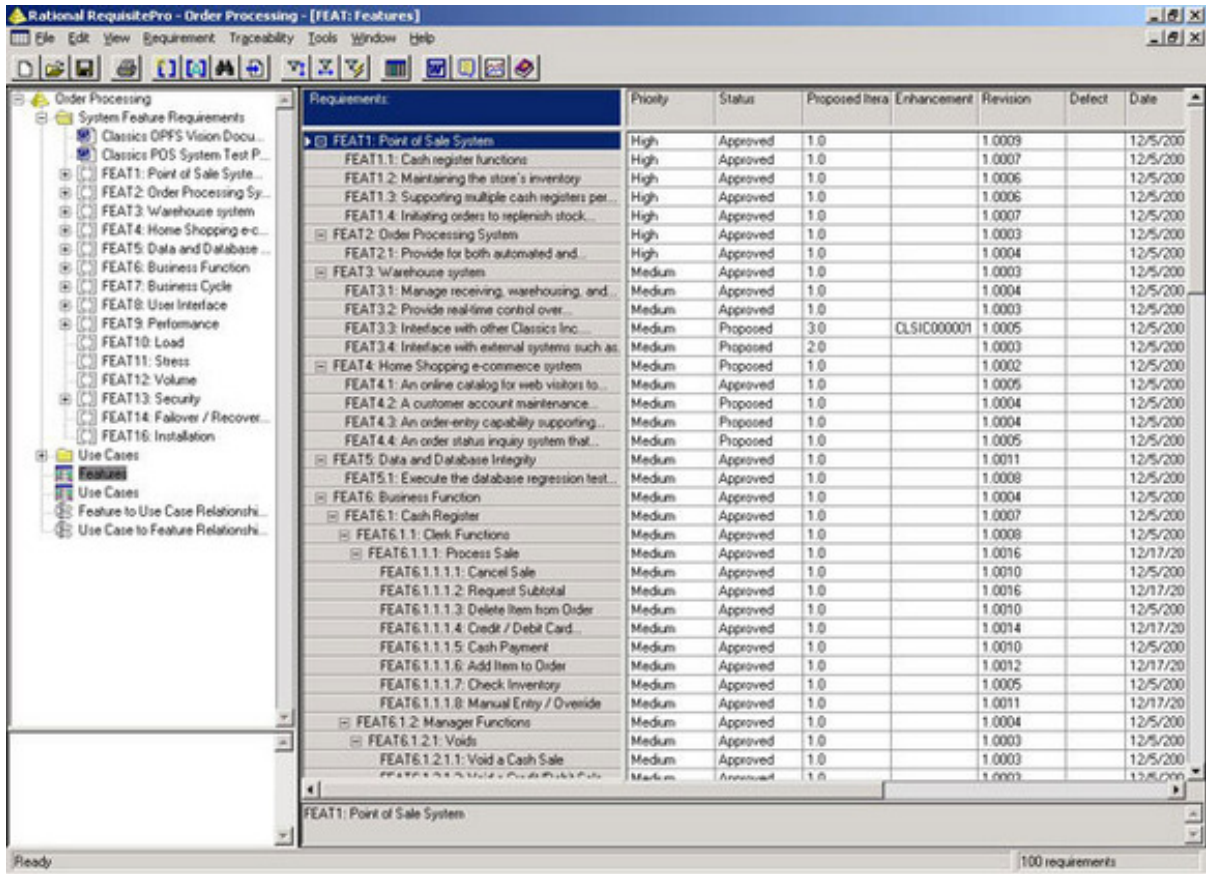
Let's start by looking at how you produce project requirements, where the requirements come from, and how they are stored in the database.

Project requirements

The application being developed in the sample files provided with the Team Unifying Platform is for ClassicsCD.com, a fictitious company that provides a Web based shop for buying Classical CDs, and a Point of Sale system to handle the needs of a real, physical CD shop. The first stage in any development is to build the list of requirements, which provides the drive and focus for the project and defines the needs and goals of the applications.

The primary goal of RequisitePro is to track these requirements and keep the team updated and on track throughout the application development process by making the individual requirements easy to write, communicate, and change throughout the life of the project and across all the different team members. Figure 3 shows the initial process of the requirements collection phase.

Figure 3. The requirement collection phase

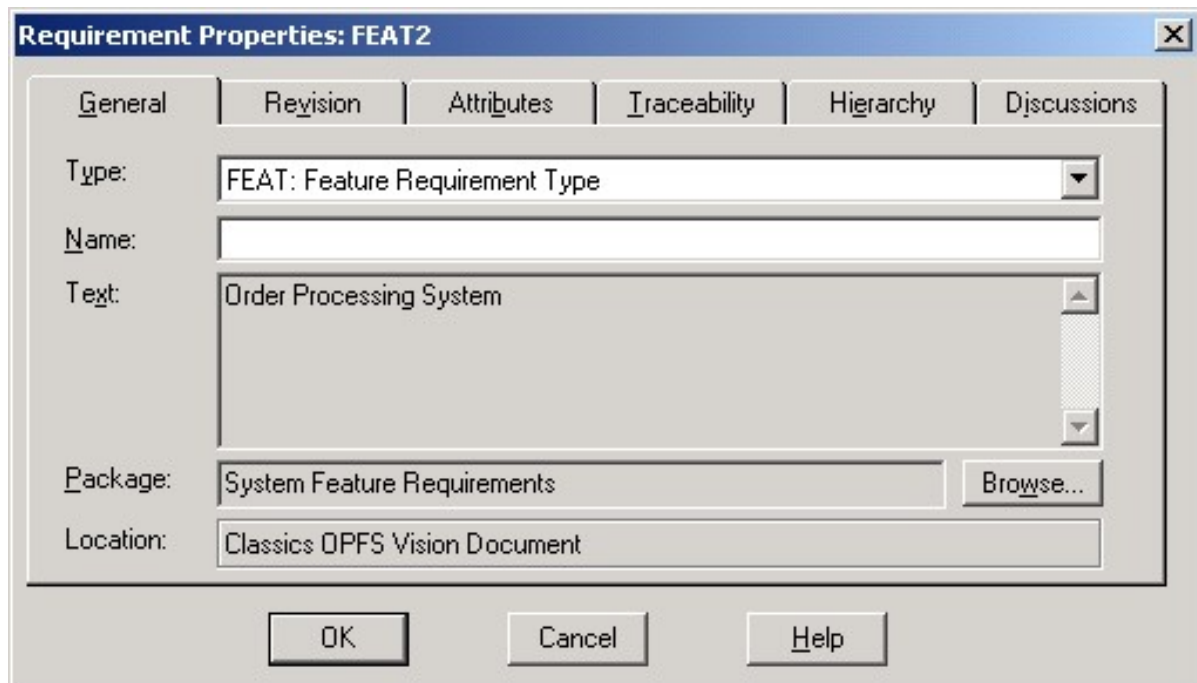


Requirements can be added to the requirements project either through the main RequisitePro interface, while building a full specification within a Word document, or through a Web interface called RequisiteWeb. Use cases are used in RequisitePro to define requirements -- although you can use other requirement types and templates. Individual requirements within a RequisitePro project are identified according to specific requirement types and can be further organized through a folder-like structure called packages.

Adding requirements to a RequisitePro Project

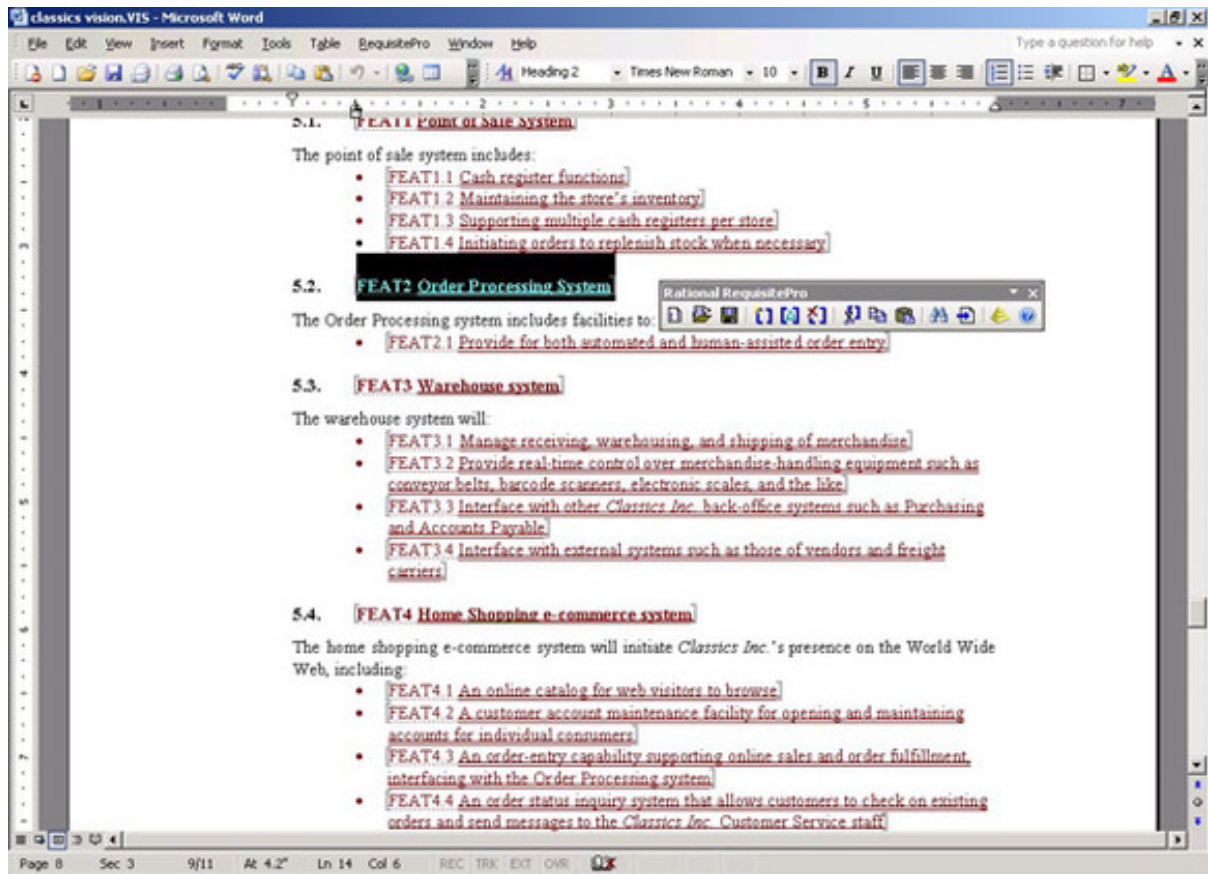
To add a requirement in RequisitePro, create a new requirement object and fill in the information described in the property window for the requirement, as shown below in Figure 4.

Figure 4. Requirement properties in RequisitePro



Alternatively, you can introduce requirements to the database from a Word document, which you can also populate with other details and other information relevant to the requirement. These documents are created within the RequisitePro structure and form part of the overall project. Word documents can also form part of the printed documentation for project management. A sample of the Word interface is shown in Figure 5.

Figure 5. Adding a requirement from Word



Let's have a look at how to build and organize these requirements into the specification for the application.

Combining and collating requirements

Requirements come from a variety of different places, usually identified by the various stakeholders in the project, including the clients, developers, and project managers. This is the first phase of the process where the team platform of the Rational suite becomes useful. Whether you are using the standalone application, the Word interface, or the Web interface, multiple users can update and add requirements into the project.

Once all the various requirements have been collected, you can start to build relationships between the requirements to show how individual requirements relate to each other. For example, a requirement for a security system might lead to another requirement for a login and authentication system. By linking and relating these requirements together, it becomes easier to trace other elements later during the development cycle. For example, if a fault has been found in the security system, you know from the requirements specification that it led to other systems being

developed that might also exhibit, or contribute to the problem.

Effective management of the project relies on the ability of the different team members to collate and collect these requirements to form the definition of the application to be developed.

To view the various requirements, including the relationships between individual requirements, use one of the views within RequisitePro. Views in RequisitePro use standard query techniques on the requirements database to show information and relationships. Each view is composed of the query that generates the information to be displayed and the view type, one of four predetermined formats for viewing the information. These view types are:

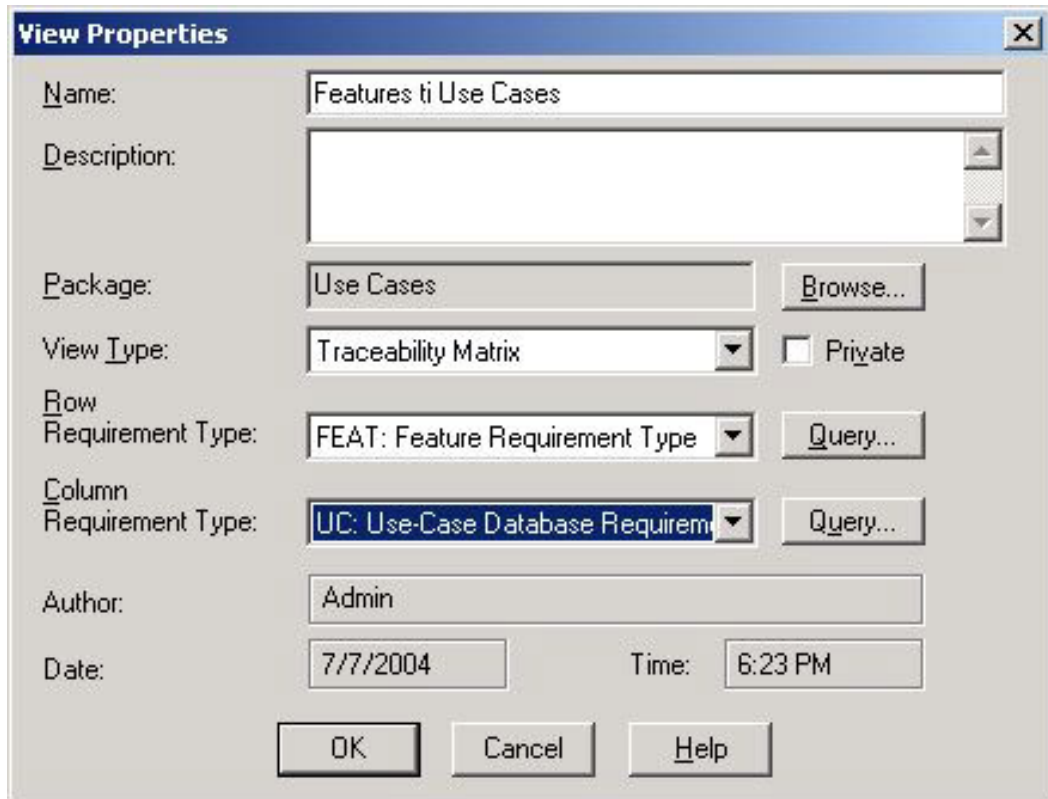
- Attribute Matrix -- a simple table of requirements and their attributes.
- Traceability Matrix -- a matrix table, showing the relationship between two requirements lists. For example, you might want to show the relationship between the stakeholder requests and the main software requirements.
- Traceability Tree (Traced into) -- a tree showing how requirements relate to the specified requirement. For example, you might want to show a tree that describes how stakeholder, developer and other requirements relate to the main software requirements.
- Traceability Tree (Traced out of) -- a tree showing the requirements traced from the specified requirement type. For example, you might want to show all the requirements traced from a stakeholder request.

Creating views in RequisitePro

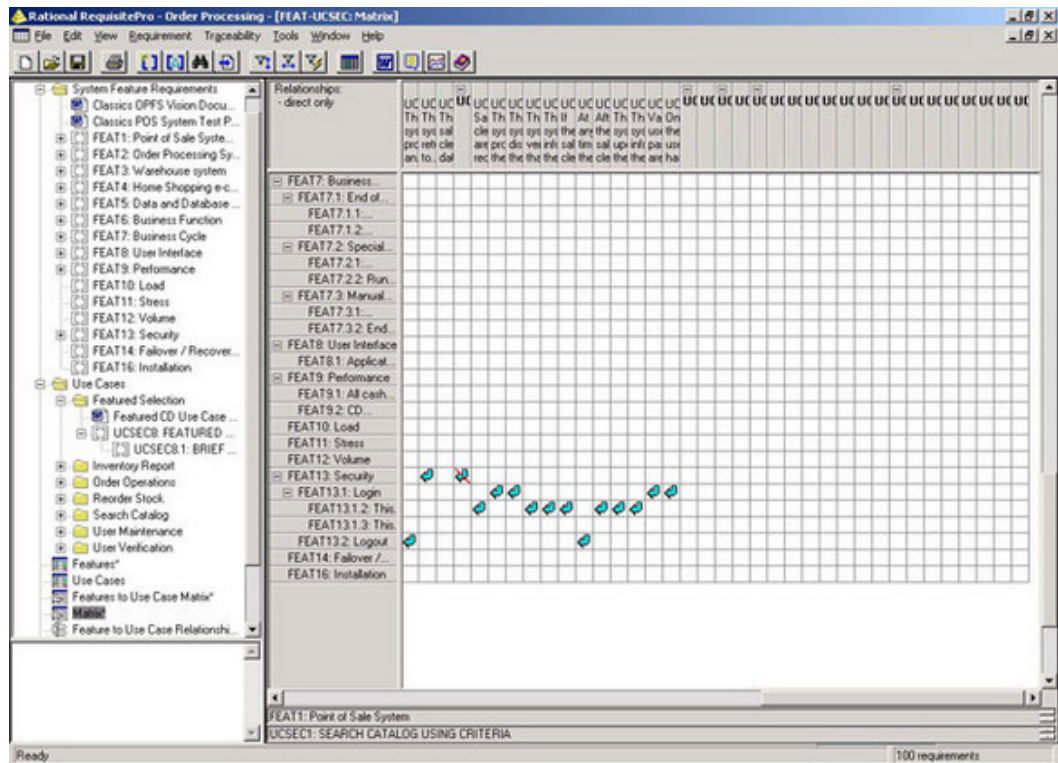
To build a new view and report information within RequisitePro:

1. Right-click on the Use Cases package and select **New > View**.
2. In the View Properties window, give the view a name and choose **Traceability Matrix** as the View Type.

Figure 6. Setting view properties



3. Choose **Use Case** as the Row Requirement Type and **Design** as the Column Requirement Type. In a very large project, you can limit the view to specific use cases by any of its properties by using the Query button to specify which requirements to include in the view.
 4. Click **OK** to complete the view.
- Figure 7. The final traceability matrix**

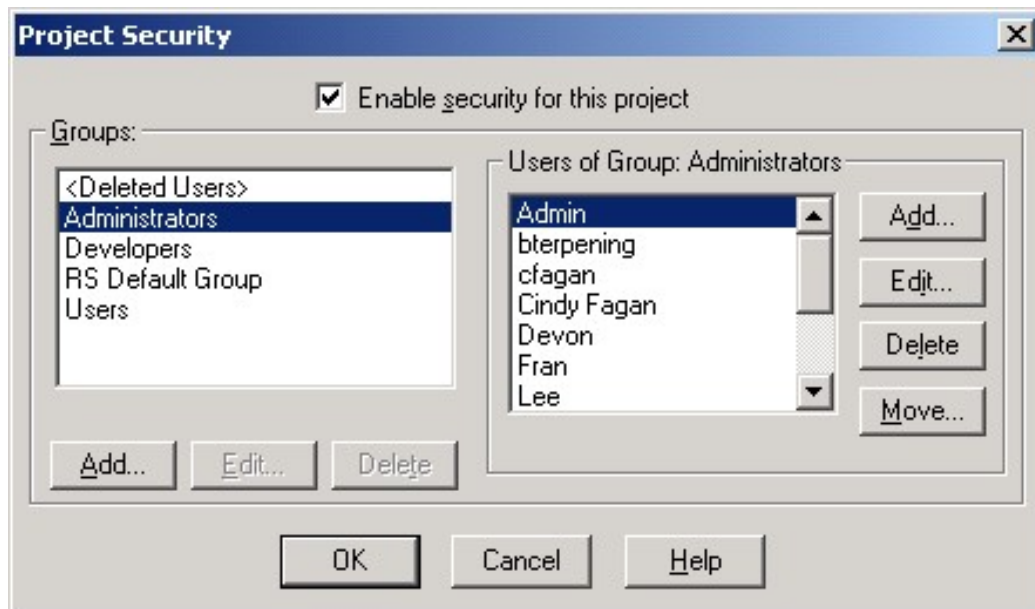


This traceability matrix -- as well as the basic attribute matrices - become a vital document which will be used by designers and other team members to help build the application. Once the application has been built, the requirements specification will also be used during testing - to ensure that functionality matches requirements - and during defect tracking to ensure that problems are associated with the proper requirement.

Team collaboration within RequisitePro

RequisitePro is a multi-user tool that can be used by a number of different people to update and manage the requirements in the project. It has its own login and user control system that can be used to identify individual users into the system. Figure 8 shows how to control security within RequisitePro.

Figure 8. Security in RequisitePro



All requirements in the database have a revision history that can be used to identify how the properties and information attached to individual requirements have been modified, and changes through the system. A sample of the change history is shown in Figure 9.

Figure 9. RequisitePro revision history



Requirements change, if not in their basic description, perhaps in their more fundamental descriptions and methods. Tracking this information helps to identify when, and why, the requirements were changed so that team members can identify the reasons behind the changes, i.e. because of a fault in the program logic, or a change of requirement from the client or other stakeholder.

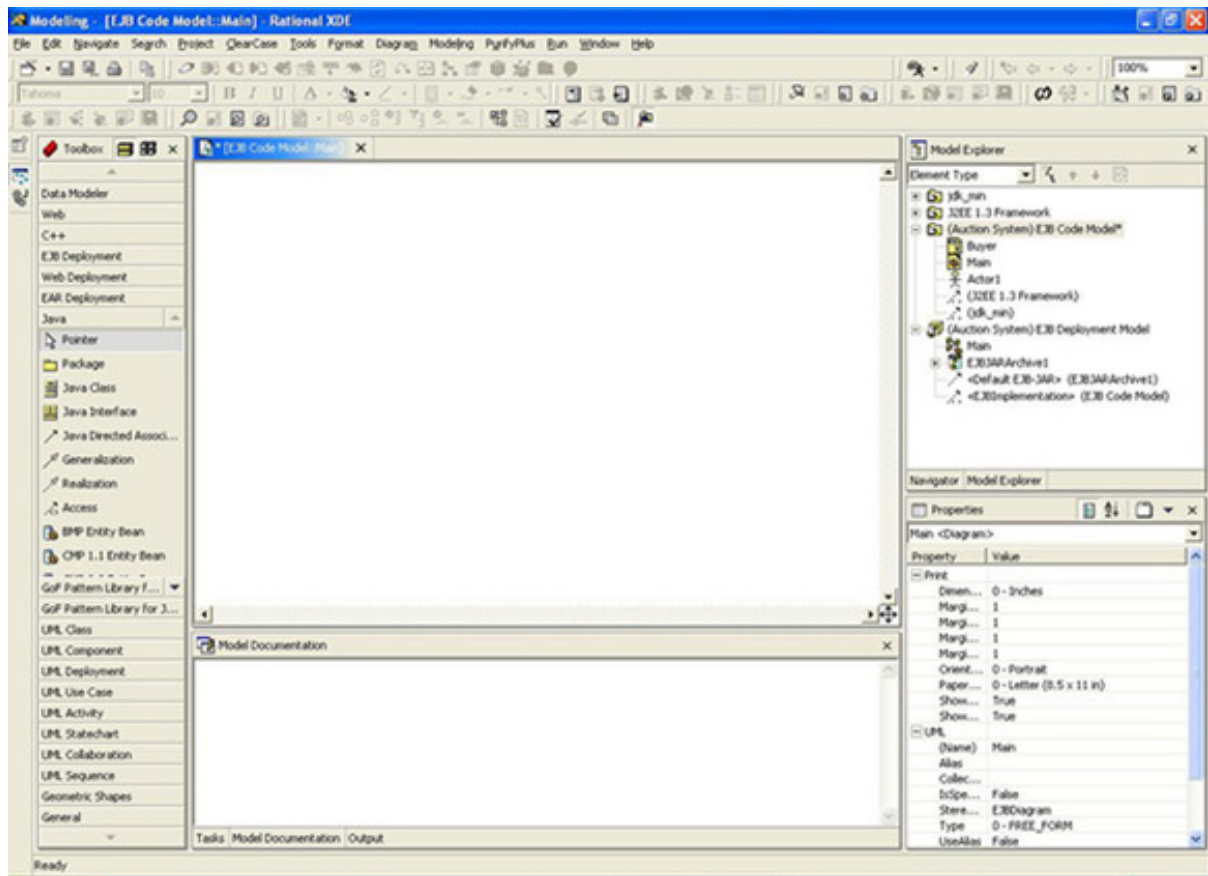
Once the requirements are, at least in their initial version, it's time for another part of the development team to use the information to build the application model, and ultimately code the application.

Translating requirements into an application model

Although Rational XDE and WebSphere Studio Application Developer (WSAD) are not a part of the Team Unifying Platform, they are team-based tools that use and integrate with the information within RequisitePro. Once the requirements are produced, it is up to the software architects and analysts to build the model based on these requirements to generate the class structure used by the application.

The implementation phase of development relies on the use Rational XDE or WSAD to start writing code, often using the patterns and classes produced during the previous design and modeling phase. A sample model is shown in Figure 10.

Figure 10. A class model in XDE



Using the integration features within Rational XDE, WSAD and RequisitePro you can build a class model of a use case, associate the model back to the original use case in RequisitePro and then use the link to help document and track the implementation of a requirement. The result is a new type of requirement in RequisitePro that you can use with a traceability matrix to identify whether individual components in the use case have been correctly modeled.

From a project management perspective you can use the traceability matrix and the identification information in the design requirement to identify the progress and to identify the member of the development team responsible for developing a given component.

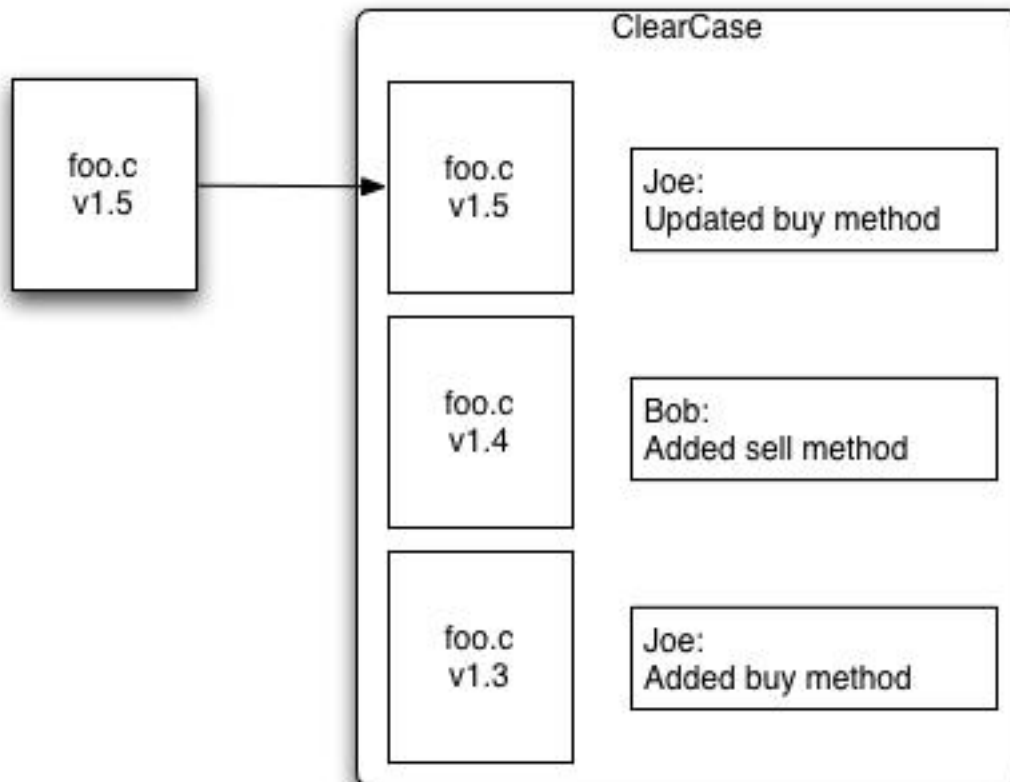
Unified configuration management with ClearCase

Throughout the process of development, you can create and more importantly modify and update a number of different items, from the original requirements documents to the code files that ultimately make up the application. ClearCase provides a method for storing changes in these documents so that you can view a history of the modifications and alterations to a particular item. In addition to the

changes, you can also create notes about the changes, and all submissions are logged to identify the user and date/time when the new version was submitted.

The information is stored in a Versioned Object Base (VOB). Each VOB holds all the information for a single project and consists of all the files for the project, along with the various comments and other metadata about the submission. Figure 11 shows a diagram detailing the basic structure of just one element in ClearCase.

Figure 11. A ClearCase element



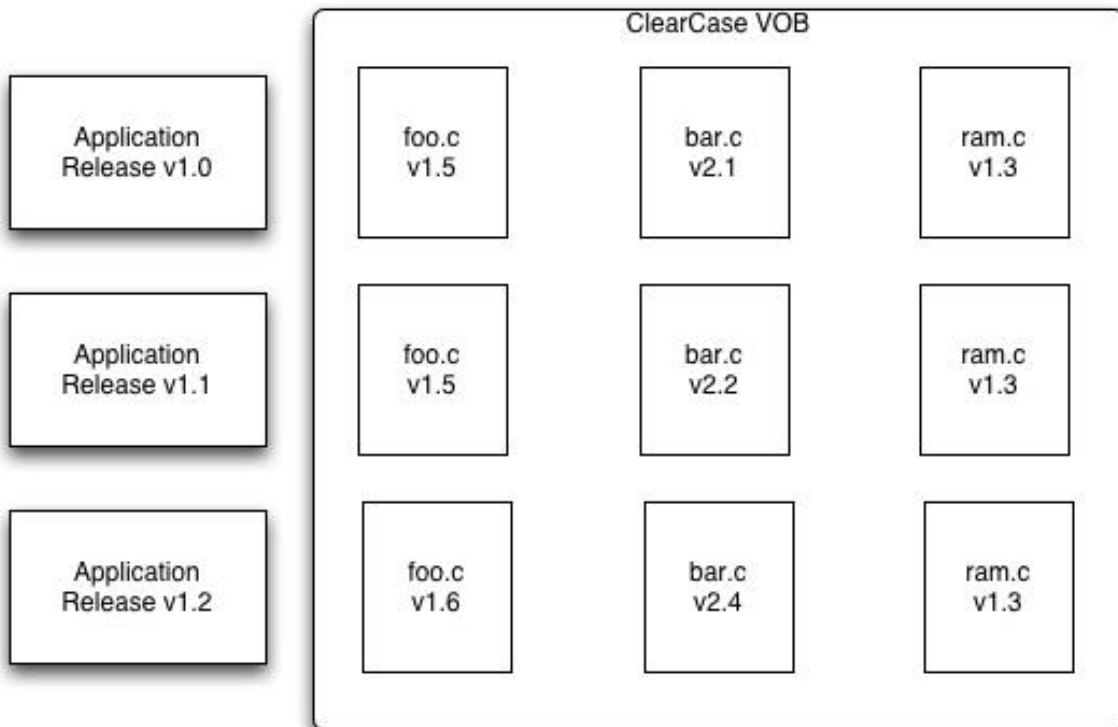
Within a team, different team members work on different versions of different documents all the time. ClearCase is used to centralize the source files, while allowing individual developers to edit the parts of the application without interfering with other user's work. For example, Bob might be working on foo.c while Alice is working on bar.c. Their work can be independent, but once they check in the latest version to the VOB, the new release becomes the latest version of the corresponding file.

Individual users must check-out a file from the VOB to start working on it. Checking the file back in implies changes and usually a note to identify what the changes were.

ClearCase in a team environment

Collectively, all the different versions of the various files in a project make up the application being developed. With ClearCase you can create a Baseline, an object that represents the stable configuration for one or more components. For example, in Figure 12 you can see that Application v1.0 relies on different versions of the source files, each of which can be edited and modified by a different developer.

Figure 12. Versioning with ClearCase



Application v1.0 is a baseline, a recognized working example of the application that you can use as a baseline for future revisions and changes. If this application is sent to the testing team, they can identify exactly which files make up the application release, and therefore which versions of the files are likely to have bugs and faults that need to be addressed.

The act of fixing and modifying a specific release of the application (and therefore, its corresponding source file versions) is called a Stream. Streams define the exact set of files in the VOB that you can view, modify, and build. Each stream defines not only the collection of files that are affected, but also the activity. Hence, you might have a stream which defines the activity related to fixing a bug identified during testing.

ClearCase can be used with a variety of tools within the Rational Suite including RequisitePro, where it's used to record changes to the requirements project and Rational XDE and WebSphere Application Developer, where it can store changes to the models and source files.

A versioning tool is vital tool used to describe the history of the development without having to constantly update each user on the changes - they can just get the information from the ClearCase database.

From a project management perspective, by tracking and monitoring the activity within the ClearCase system you can determine which files in the system are likely to be causing the most problems and therefore whether you should devote more time to a specific component of the system for most extensive testing and investigation.

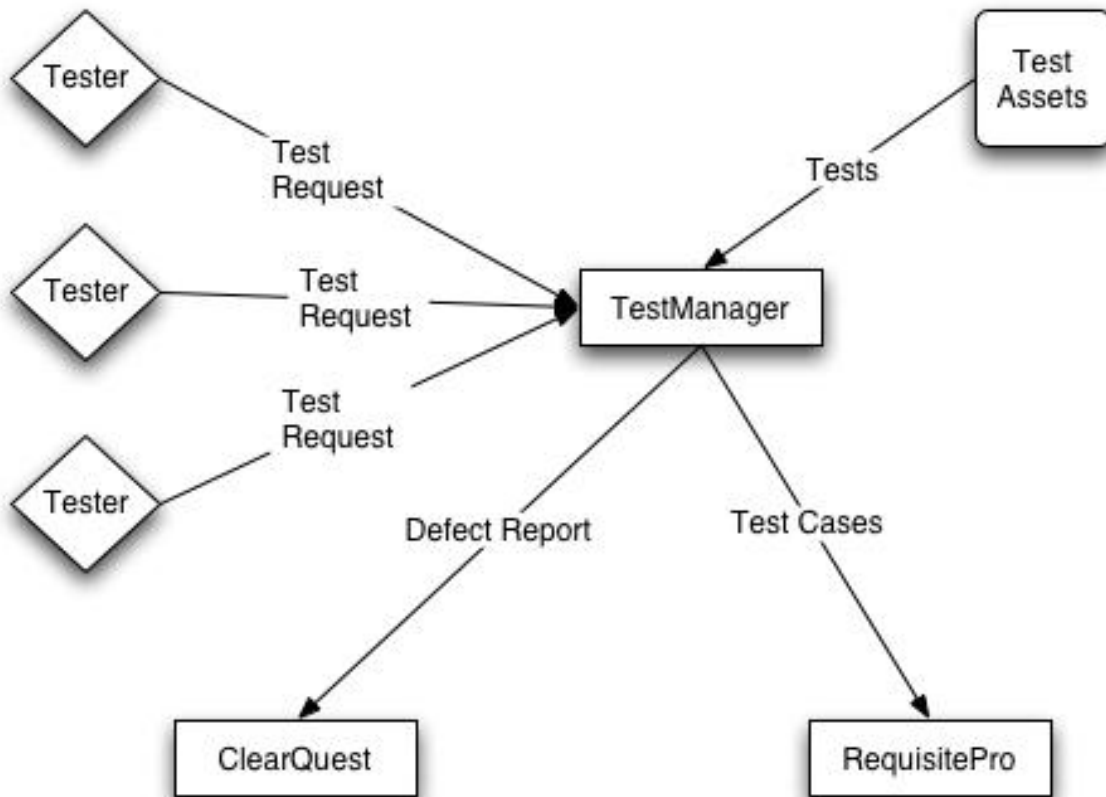
Testing

Once the application has been completed, it's time to test the application to determine its suitability against the original list of requirements. Testing is handled by different teams and team members throughout the course of the applications development. It can be a complicated process as you handle functional testing of the application, as well as the user interface, performance and load testing of your application.

Within the Team Unifying Platform, Rational TestManager provides a central interface for the management and testing of applications. The TestManager supports functional, performance, manual, integration, regression, configuration and component testing. You can create tests and save the test specifications so that they can be used by the entire team. Through TestManager, individual members of the test team can request the execution of one or more tests on the application automatically.

TestManager also communicates with RequisitePro and ClearQuest. With RequisitePro you can integrate between the tests and the original use-cases in the requirements specification to verify the operation. With ClearQuest, you can automatically submit defect reports to the system. Figure 13 shows a diagram of the basic integration.

Figure 13. Testing with TestManager



Each iteration of each test case is recorded and each is mapped against the release of the application, so you can use the reporting functionality of ProjectConsole that we've seen elsewhere in this tutorial to provide metric information about the effectiveness of each release of software. In most cases, the number of test failures should decrease as the number of releases increase, due to the iterative improvements in the process of the Rational Unified Process.

Defect and enhancement tracking with ClearQuest

Once the application has been built, and the testing has started, you need to start recording defects, realized or otherwise, about the application. This is the role of ClearQuest, which receives defect information automatically from TestManager. TestManager collects information from the tests submitted from individual users. By using the implementation details and the links back to the original requirements specification for the project within RequisitePro you can identify when a defect affects a particular requirement.

Also handled by ClearQuest are enhancement requests. These are the changes to the application requested by stakeholders in the project. These requests are

changes to the operation of the system: Perhaps a change of execution order is required, or the way a system works is not quite what the stakeholder imagined.

Defects can be introduced into a ClearQuest project automatically from one of the various testing solutions or through a manual entry system. Similarly, enhancement requests can also be introduced into the project manually. The automatic option doesn't make sense, as there is no way to automatically detect a change to the original requirements. A sample defect request entry form is shown in Figure 14.

Figure 14. A sample defect request entry

The screenshot shows a 'Submit Defect' window for ID 'CLSIC00000131' with state 'Submitted'. The form is organized into tabs: 'Iterations & Notes', 'Unified Change Management', 'Requirements', 'Main' (selected), 'Test Data', 'Environment', and 'Attachments'. The 'Main' tab contains several input fields: 'ID' (pre-filled with 'CLSIC00000131'), 'State' (pre-filled with 'Submitted'), 'Headline' (empty), 'Suite Project', 'UCM Project', 'Owner', 'Priority', 'Severity', and 'Customer Priority' (all dropdown menus). There are also 'Keywords' and 'Symptoms' sections, each with a text area and a browse button (...). On the right side, there are three buttons: 'OK', 'Cancel', and 'Values' (a dropdown menu).

Reports can also be made through ClearQuestWeb, which provides a Web interface. Figure 15 shows a sample of the Web-based enhancement request form.

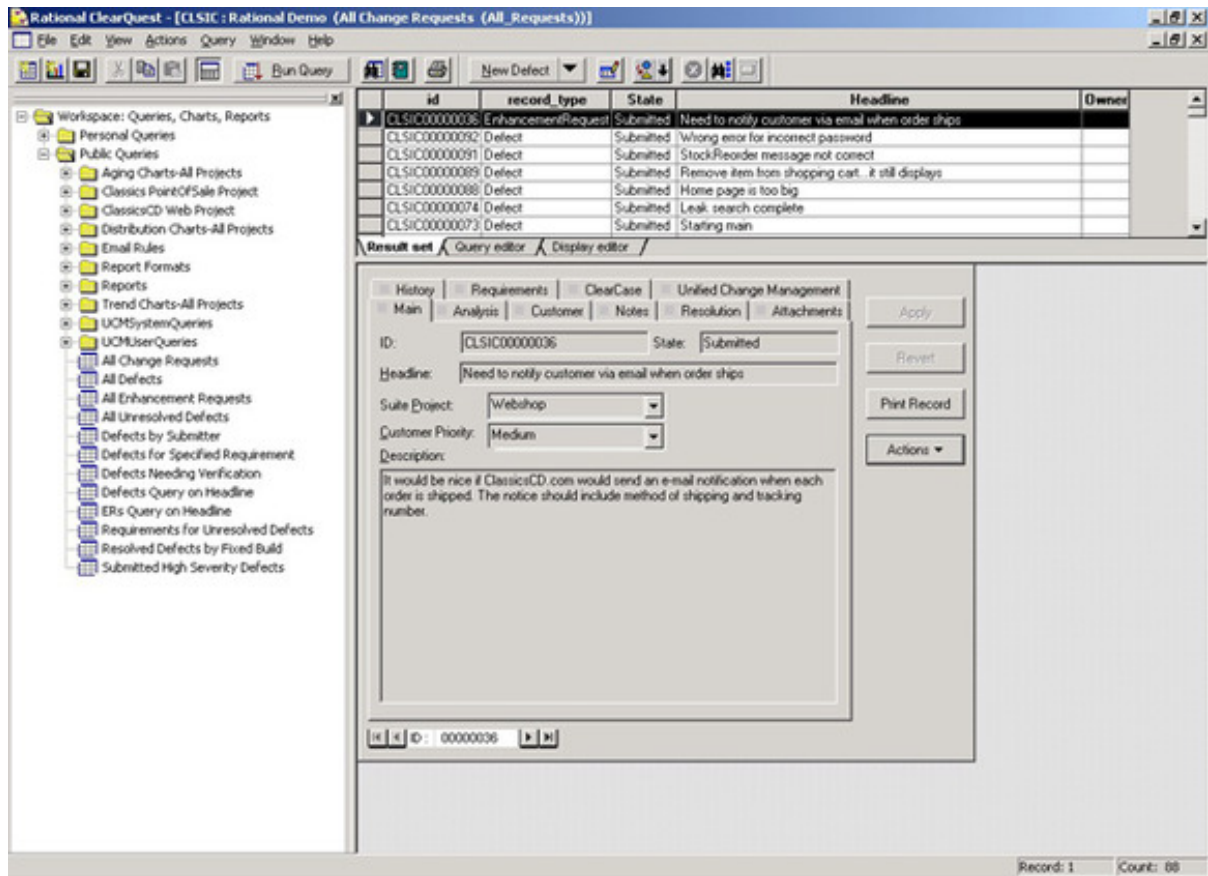
Figure 15. The Web-based enhancement request form

Main			
Unknown field type: StaticText			
Headline:	<input type="text"/>		
ID:	RMBU00011334	State:	Submitted
<i>Project:</i>	<input type="text"/>	Customer Priority:	<input type="text"/>
Origin:	<input type="text"/>		
<i>Description:</i>	<input type="text"/>		
CQ Schema Version: 112			

Reviewing requests in ClearQuest

ClearQuest is aware of multi-users and can accept reports and information from a variety of sources. Once the information is recorded within the ClearQuest database, queries can be used to help report on the status of the system. You can see in Figure 16, for example, a very simple report showing the status of the two projects within our ClassicsCD project. The Open_High_Priority_Ch columns shows the number of outstanding change requests based on defect reports. This indicates that you have a high number of faults that must be addressed.

Figure 16. Reporting in ClearQuest



More complex reports can be generated. The ClearQuest application allows for a combination of public and personal queries and reports, so you can generate a series of reports for your team, while individual members can build reports relevant to their area of responsibility.

Review of the team process

Development in a team is all about communication. If you can standardize on that communication then you can simplify and ease the process for developing an application. As you've seen in this section, the tools in the Team Unifying Platform and the functionality provided by other tools, coupled with a level of integration that allows components to be tracked from inception at the requirements stage to their refinement and bug fixing means that the Rational tool set significantly eases the sharing of information during the execution of the project.

From a project managers point of view, browsing all of the information in the separate tools can be a nightmare. Even for individual team members, finding out a fairly trivial piece of information can be time consuming if you have to load up one of the applications first. This is where the ProjectConsole is most useful because it

aggregates information from all the different tools and provides a method for building views and reports on the information. Let's start by looking at how the ProjectConsole collects information.

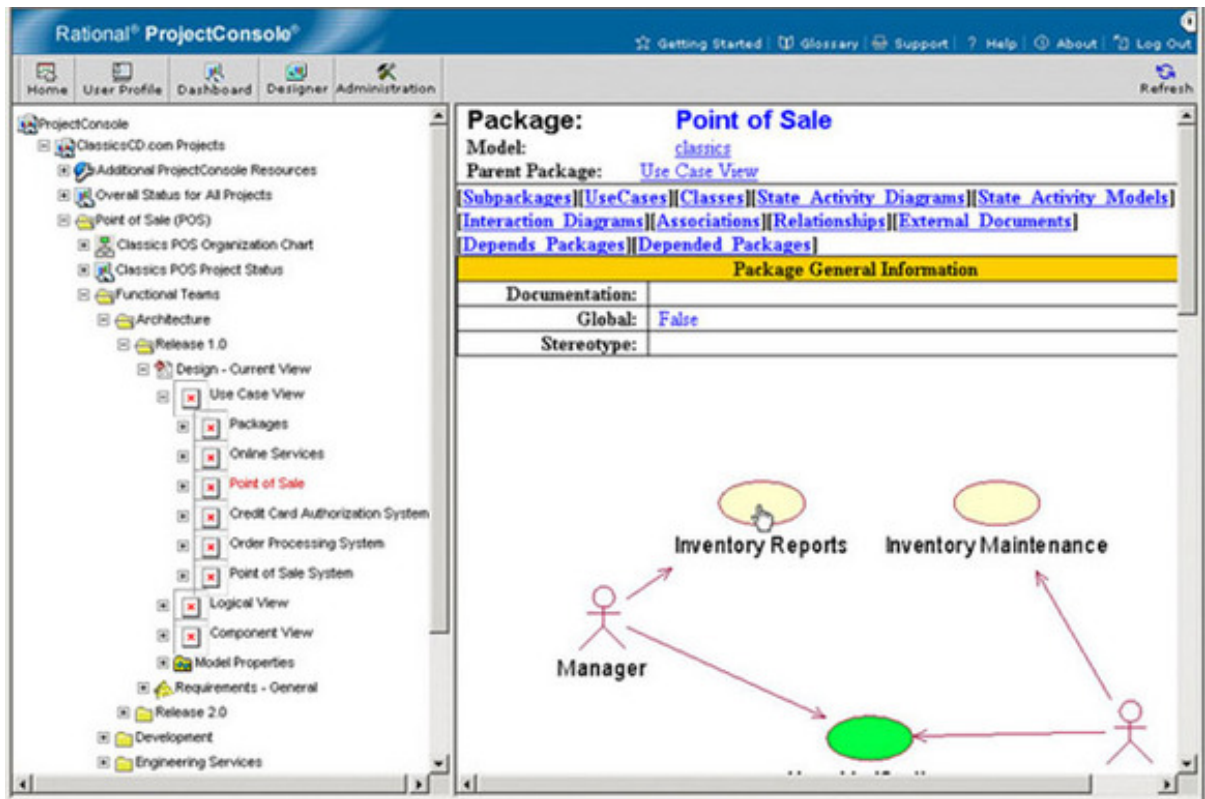
Section 4. Using the ProjectConsole

Overview

ProjectConsole works through a Web interface and uses a Java based server system to provide an interactive interface to your project data. The ProjectConsole combines all of the information from the different applications used during the development of an application and puts it into one place and interface that you can use to monitor the progress of development.

Because all of the information for the project is in the same place, everybody involved in the project can get in and view information about the project, the state of different components and even view individual items to gain an understanding of the project. For example, your developers might not have, or indeed need, access to the database of requirements that are built with RequisitePro, but by using the ProjectConsole, they can view the data, and all the related components, during the development. Alternatively, they might want to view the use case model for a given component, without having to load Rational XDE. A sample of the ProjectConsole window is shown in Figure 17.

Figure 17. Model viewing in ProjectConsole



From an administration point of view, this approach greatly simplifies the deployment of software to clients, allowing you to share information from the different software in the Rational Suite without having to provide each person with access to every software package.

Aggregation

To report the information in all of the different tools covered in the previous section, ProjectConsole needs to import data - artifacts - to extract from the various projects, models and other sources. These artifacts are stored in a database warehouse. Individual schemas define the format of the information that will be stored in the warehouse, and maps are used to translate the information from the source material into the corresponding schema.

The sequence for collecting information is:

1. Identify the source material.
2. Choose the artifacts to extract.
3. Define a warehouse schema to hold the relevant information.

4. Define a mapping to translate source artifacts into our schema.
5. Create a collection task that performs the import.

Collection tasks can be scheduled to occur automatically at periodic intervals. You can, for example, schedule an update of all the information from the various applications used in your company to run overnight to update the details.

By automating the process of collecting the information, you can ensure that the information in the ProjectConsole databases is always up to date. Now let's have a look at how you can view the information that is collected to get more meaningful information about the progress of a project.

The collection process is different for each tool and you should refer to the ProjectConsole documentation for more details on extracting the information in each case.

Environment

The main interface into the ProjectConsole environment consists of a navigation tree on the left, with a main viewing panel on the right. The tree works like most tree systems, clicking on the + character expands a particular folder or entry to display the contents. The basic window is shown in Figure 18.

Figure 18. The ProjectConsole interface



The information in the tree is organized through a series of folders, which hold collections of nodes. The nodes define points of information as extracted from the ProjectConsole schema. Panels within the nodes are used to display information, either graphs, text, or other components that you want to display in your console. Individual panels can display different types of information, including text, graphs and graphical representations of objects, and one node can consist of a number of panels. This enables you to create a node that gives you an instant overview of the project that is made up of a number of panels which have sourced their information from across the database.

The contents of the tree - and ergo the reports and information that is accessible through the ProjectConsole for viewing by the user - is controlled centrally according to the collectors and warehouses defined when the information was collected. You can create custom reports and views on the information on a global scale through the ProjectConsole. That information will be available to all the users of the project.

In addition, the tree is customizable through a custom area called My Workspace. Each user has their own workspace and they can control the layout of the tree and the contents of the tree view based on the information stored in the ProjectConsole database. The tree can include reports, summary data and other components relative to their role in the organization and the project development.

Deployment

ProjectConsole is only useful to the team if you make it available to all members involved in development of the application or applications within your organization. Because ProjectConsole is a Web application, the deployment itself is easy - simply install the ProjectConsole on to a machine that will be used to hold the ProjectConsole software and databases. All users should then be provided with a login into the system, provide a separate login for each user so they can each customize their own environment.

You also need to determine what information to collect from the various applications and how frequently the information is collected. You've already looked at the basics of the collection process and how it is automated to update information at regular intervals.

Finally, you need to determine the layout and organization of the ProjectConsole Web site itself. You can organize the tree navigation system to follow whatever system you prefer. You might want to organize it by functional teams, project sequence or even the departments that will ultimately be responsible for the application. Nodes and even entire folder structure can be copied between projects, and ProjectConsole is capable of recording information from multiple projects.

Section 5. Build reports and graphs in ProjectConsole

Viewing requirements statistics

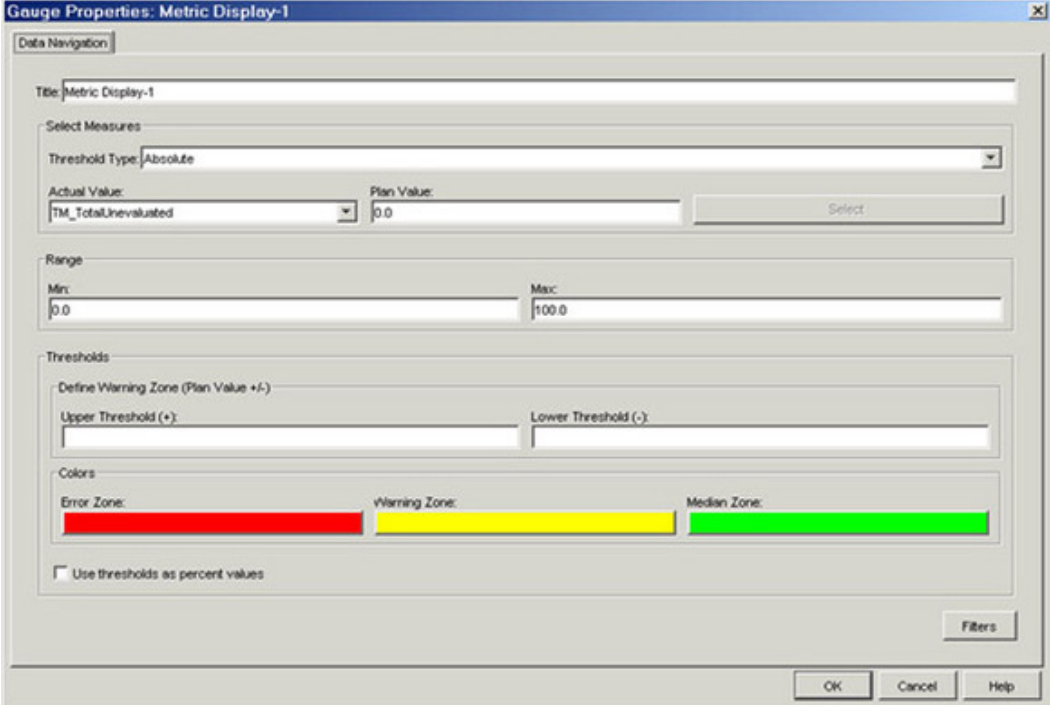
From a project management perspective, the requirements help drive the general progress of the project. Requirements help to define the major features of the project, and therefore the milestones and goals of the project.

Keeping track of the number of requirements (metrics) can be a useful way of studying the effective load of developmental effort required to complete the project. For example, by tracking the number of feature requests you can create a graphical representation of the growth of requirements as follows:

1. In ProjectConsole, expand the navigation tree so that the **ClassicsCD.com Projects > Point of Sale (POS) > Functional Teams > Architecture** folder is visible.
2. Select the Architecture folder and then click on the New Metrics Panel (the bar graph icon) within the Dashboard toolbar to create a new metrics panel.

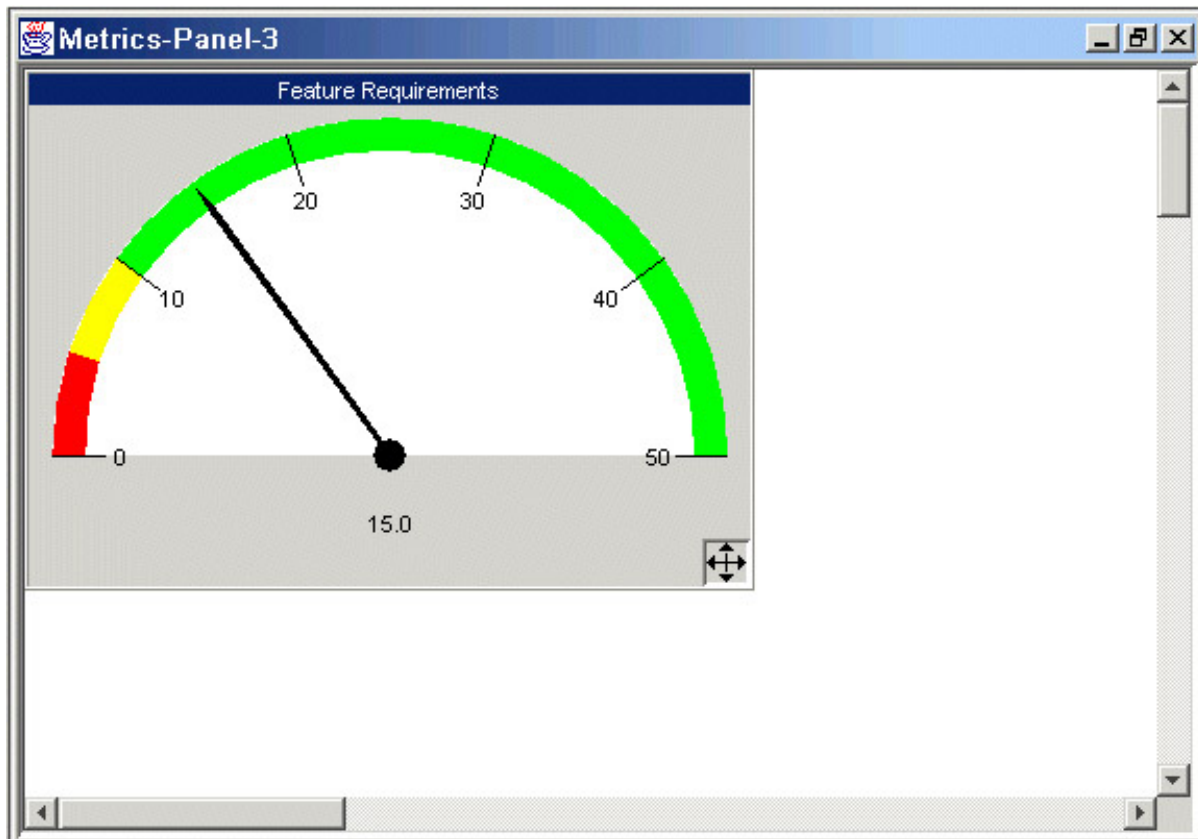
3. Click **Insert > Gauge** to add a gauge to the panel.
4. You'll see a properties box like the one shown in Figure 6. Give the gauge a name. Under Select Measures, choose **Num FEAT Reqs.** From the Actual Value list, and in the Plan Value box enter the number **10**. This sets the expected value for the gauge and is used with the colors to give an indication of the current value against its expected one. You can also give the gauge a minimum and maximum range. The Thresholds features allows you to add warnings when the value exceeds different thresholds. For example, you might want to set a limit of 30 feature requests to ensure that the project does not become bloated. Once you're finished, click **OK** to create the gauge.

Figure 19. Creating a viewable gauge



The resulting gauge looks something like Figure 20. It's a simple display, but it provides an "at a glance" overview of the status of requirements for the project. To save the gauge panel that you just created, click **Save**.

Figure 20. Feature requirements gauge



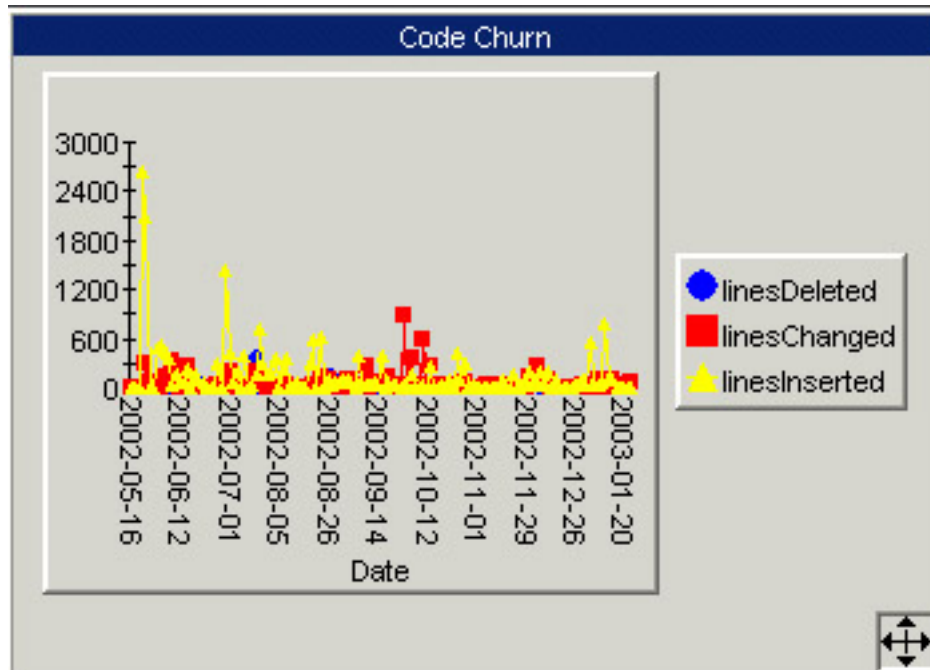
This is an example of just one simple element that you can create. All the attributes of a requirement are importable into ProjectConsole, and you can view and report on the information in a variety of ways just by using the graphs, gauges, and comparison tables that are built into the ProjectConsole application.

ClearCase and ProjectConsole

In the wider scope of measuring and monitoring the project on a team basis, ProjectConsole collects the history of changes for the different files in the system so that you can report the changes, their frequency, and who is making the most changes.

You can also use ProjectConsole to provide metric data on change frequency, who is changing the most files, and how active different files and parts of the projects are. This can provide a useful tool for monitoring the activity on a project. For example, Figure 21 contains a graph showing the activity of changes (the Code Churn) in the project.

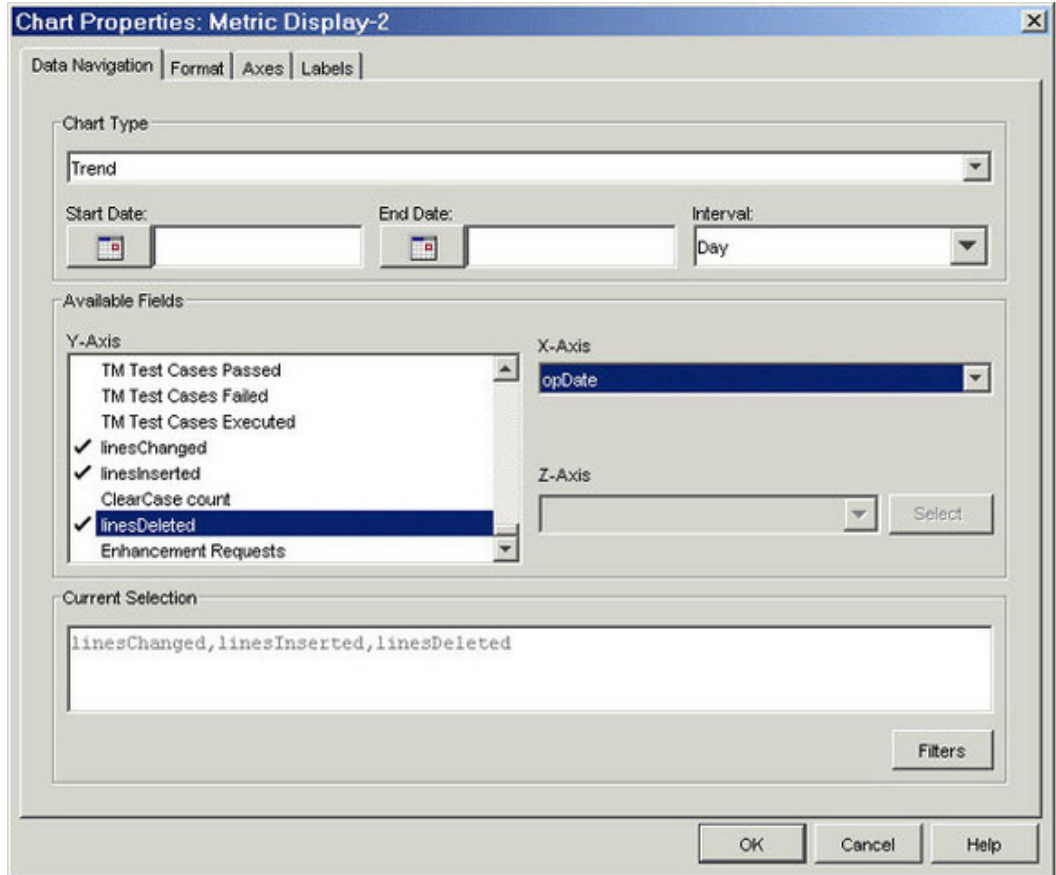
Figure 21. Code Churn



To produce this chart using the sample ClassicsCD.com data:

1. Choose a node in which to store the chart. Right-click on a suitable folder and then choose **New Metrics Panel**. To rename the panel, right-click on the new panel and choose **Rename**.
2. On the Dashboard toolbar, click **Insert Chart**. You'll be prompted with the window shown in Figure 22.

Figure 22. Chart properties



3. Choose **Trend** as the Chart Type and use the Y-Axis list to choose the information to be summarized. For this chart use **linesChanged**, **linesInserted**, and **linesDeleted**. All metrics are recorded by the ProjectConsole from ClearCase when changes are made to the source. For the X-Axis on this chart, you can change the labels displayed by each item by clicking on the Labels box.

You now have a clear view of the changes to the code over a period and how the team can use this to determine the activity on the project.

The project manager has complete control over the entire project and access to all of the information that is needed. Furthermore, individual members of the team can access the information they need at the different stages of the project development. By using the ProjectConsole in combination with the built-in facilities of the different tools, you provide an effective communication, control and management environment for the duration of the project, with the added benefit of ongoing documentation.

Section 6. Wrap up

Summary

The Team Unifying Platform provides two major components. First, it provides access to a suite of software packages that can be used to help manage the development of an application. This includes RequisitePro for collecting requirements, ClearCase for tracking changes, and ClearQuest for managing change requests (both defects and enhancements). Individually these tools provide everything you need. Using the integration features built into these products you can also share information between products to ease the process.

The second part of the Team Unifying Platform is the ability through a Web interface for project managers and team members to view and summarize information and data produced by the individual products to make it easier to share and exchange information. By using TUP and the ProjectConsole it's possible for all members in the project to follow the application development process, eliminating the need for constant meetings and automating the process of viewing and disseminating project information and documentation.

Resources

Learn

- Find more information on the entire suite of [Rational products](#).
- The [Rational developerWorks](#) site is a portal for more information, tutorials and articles on the Rational environment.
- New to Rational? [Rational developerWorks](#) can help you get started with Rational products.

Get products and technologies

- Download of trial version of [RequisitePro](#).

Discuss

- [Participate in the discussion forum for this content](#).

About the author

Martin C. Brown

Martin C. Brown, a [Studio B](#) author, is a former IT Director with experience in cross-platform integration. A keen developer he has produced dynamic sites for blue-chip customers, including HP and Oracle and is the Technical Director of Foodware.net. Now a freelance writer and consultant, MC, as he is better known, works closely with Microsoft as an SME, is the LAMP Technologies Editor for LinuxWorld magazine, a core member of the AnswerSquad.com team and has written a number of books on topics as diverse as Microsoft Certification, iMacs and open source programming. Despite his best attempts, he remains a regular and voracious programmer on many platforms and numerous environments. MC can be contacted at questions@mcslp.com, or through the Web site, <http://www.mcslp.com>.