

Improved application development: Part 3, Incorporate changes into requirements

Skill Level: Introductory

[Martin C. Brown](mailto:questions@mcslp.com) (questions@mcslp.com)

Author

Studio B

07 Sep 2004

The focus of this third tutorial in the Improved application development series is on change management. In this tutorial, you will learn how individual change requests are linked and traced back to the original requirements specification, and how to generate a new specification.

Section 1. Before you start

About this tutorial

In part 3 of this series, you'll see some of the issues of change management in application development. In [Part 1, Translating requirements into an application model](#), you looked at the initial stages of application development when creating requirements that will ultimately drive the project. In [Part 2, Integrating XDE Professional into WebSphere](#), you looked in more detail at the actual building of the components in your application.

In this tutorial, you will learn how individual change requests are linked and traced back to the original requirements specification, and how to generate a new specification to ensure development is building software against the latest customers needs. Change requests include defects reported by testers, end-users, or the customer support help desk, and enhancement requests from clients and other stakeholders. As an example of an enhancement request in the sample auction application, the client might decide that they really want the seller to use multiple

pictures to describe their item, or perhaps they want buyers to have an auto-bid option to automatically keep re-bidding amounts up to a maximum. Unlike enhancement requests that tend to represent stakeholder wishes, defects are bugs and faults in design or code and are specifically related to a given release of an application or one of its components.

IBM Rational ClearQuest is ideal to collect all requests for changes (defects and ERs) and then accept/reject, organize, assign and approve the changes as they are accepted, resolved and implemented. Some change requests will have an impact on the original requirements specification that is still the driving force behind the project. To manage that impact of change to requirements, Rational ClearQuest integrates with IBM Rational RequisitePro to track the link between enhancement requests and requirements specifications.

Key parts of the tutorial include:

- Collecting change requests
- Integrating Rational ClearQuest and Rational RequisitePro
- Building a new version of the requirements specification
- Modifying the application

Should I take this tutorial?

This tutorial series is primarily targeted for project managers, program managers, and developers. Others in the software development business might also find it useful. In each of the tutorials in this series, you will learn about different Rational tools and the part they play in the software development process. Different team members might be interested in some tutorials more than others, but it is worth the time to take the complete series from beginning to end to see how all the products work together to form a cohesive process.

This tutorial details how individual change requests are linked and traced back to the original requirements specification, and how you can then generate a new specification to ensure development is building software against the needs of your customers. You will see how IBM Rational ClearQuest is ideally suited to collect and handle all change requests, and manage the impact of changes to requirements with the integration of Rational ClearQuest and Rational RequisitePro.

Ideally, you should be familiar with the application development process, and be involved in at least one part of the process. Familiarization with Rational tools is helpful, but not required.

Prerequisites

To complete the steps in this tutorial, two products in the Rational Suite are required: Rational ClearQuest and Rational RequisitePro. Links to these products are below.

Rational RequisitePro is available as a [trial download](#).

Note that you must have the same releases of the tools for the integration to work. You'll be using v2003.06.12 of all the various tools in this series.

You will also need a copy of [Rational ClearQuest](#).

A demonstration of developing the Online Auction system can be downloaded from the IBM Web site using the following links:

- [Online Auction Demo Part 1](#)
- [Online Auction Demo Part 2](#)
- [Online Auction Demo Part 3](#)

You can also find demonstration artifacts supplied as the example with the WebSphere Studio Application Developer, a tool you'll actually be using later in the series. Check the Auction panel in the first section for more details about the auction system and where to obtain the artifacts.

Section 2. Collate change requests

Changing an application

In the previous tutorials, you learned how to build a requirements specification and translate that specification into an application model. Ideally the specification captured all customer/client requirements. However, in most projects, not enough time is spent identifying requirements early on, so it's likely there are going to be some changes. Either the application doesn't behave quite right, does not provide the value anticipated by clients, there are bugs, or the performance of the system is not up to the standard required by the stakeholders.

Managing changes to an application is a big challenge in today's market place. You have to be able to modify and adjust your application without affecting and upsetting

your existing customers and functionality. Any way that you can effectively manage and monitor these change requests is going to be a significant help.

As changes are accepted it's also vital that you maintain the original requirements specification in synch with these changes, as the requirements specification acts as representation to what was promised to clients. To ensure developers and testers build and test upon the promise, it is vital that accepted changes are incorporated in the initial requirements specification. In doing so, you merge all stakeholder requests (the original ones collected during the requirements gathering phase and the ones that emerged after you started your project) so that at the end of your project your software actually meets the stakeholder needs. This is done even though these needs have most likely changed during the course of the project, and you can even track and monitor the history of requirements and the change requests that altered them.

Rational ClearQuest acts as a centralized tool for collecting change requests from various sources, and as a clearing and approval mechanism for change requests before they are linked to the requirements they affect. Rational ClearQuest integrates directly with Rational RequisitePro to link accepted change requests to requirements, so that the source of requirements changes can be monitored through the reporting mechanisms in Rational RequisitePro.

Source of change requests

In Part 1, you concentrated on producing a requirements specification based on the stakeholder needs, that of an auction system for selling and buying products through a Web site.

Change requests come from a number of different sources:

- External stakeholders -- the most obvious, stakeholders can often request changes to an application. These are generally enhancement requests to the original requirements for the application. Enhancements fall into two basic categories, either they are requests to enhance an existing feature, or they are an extension of the functionality already provided. For example, an enhancement request based on existing functionality might, within the auction system be the support of multiple images for a given auction item. Defects can also be lodged by stakeholders if the defect relates to the operation or implementation of an application, rather than an all out bug.
- Competitive comparison -- you might want to add some features to bring your software on par with some of its competitive tools.
- Quality Assurance team -- testing tools provide automated reports on

defects in the system and can automatically generate a list of problems in an application as part of the testing process. Defects can be anything from bugs in the software to major faults in the supporting logic. The use of testing tools such as PurifyPlus(TM) and TestManager is covered in Part 5 of this series.

- Development team -- developers will often identify potential bugs in the system and possible avenues for enhancement purely because they are closer to the implementation than any other member of the team.

What you need is a way to collect all requests in one central place to make intelligent decisions on where to use development resources to maximize stakeholder satisfaction.

Collecting requests

You know how to collect them and where they might come from, but why do you need to track this information?

The simple answer is that unmanaged requests of this type can become a major headache for the development process of an application. Creating lists of defects and working through it is pointless unless you can identify their origin; it may well be that a fix for one defect solves or isolates another defect in the system.

Without some system in place, it can also be difficult to monitor and prevent the enhancement requests from altering the priority and progress of the existing development. I've seen projects without some kind of change management system that spent more time continually adding and implementing enhancements, before they had even achieved the goals of the original version of the project.

By managing, allocating and prioritizing all of these requests you can ensure that as a development team you are devoting your energies in the right direction, not wasting them on pointless features and fixes that may never have been part of the original goals.

To manage these change requests, you're going to use Rational ClearQuest to collect the requests and Rational RequisitePro to integrate incoming change requests with the requirements that drive your project development process.

Section 3. Rational ClearQuest

Background

Rational ClearQuest is a change and defect tracking system. It stores change requests in a flexible data structure. Although there are predefined types such as Enhancement Requests and Defects, you can create a structure to hold pretty much any type of change request you require.

Once change requests are in Rational ClearQuest, you can track the requests, generate reports and statistics on the requirements, and manage the priority and severity of the different requests within the system.

Rational ClearQuest database schemas

A Rational ClearQuest database is created based on a database schema. Each schema defines the layout and structure of the available change request record types along with fields, GUI forms, actions, states and other detail used to store and manage the Rational ClearQuest information.

Rational ClearQuest supports a number of "out of the box" database schemas that you can use directly, or through Rational ClearQuest Designer (included in Rational ClearQuest) create your own schemas and layouts. The list below shows the standard schemas (along with their descriptions) provided by Rational ClearQuest. Note that others might exist in your installation based on the toolset you used to install Rational ClearQuest (for example, the Rational Team Unifying Suite includes schemas specific to the ClassicsCD.com sample).

- **Blank.** Contains system fields only. Use Blank to create a schema from scratch.
- **Common.** Contains metadata that is common to the remaining schemas.
- **Defect Tracking.** Contains the fields necessary to start using Rational ClearQuest to track defects in a software development environment.
- **Rational Suite AnalystStudio.** Provides code for the Rational ClearQuest and Rational RequisitePro integration. Use with Rational Suite AnalystStudio.
- **Rational Suite DevelopmentStudio.** Contains fields and rules that work with Rational's Purify, Visual Quantify, and Pure Coverage. Provides code for the Rational ClearQuest and Rational RequisitePro integration. Use with Rational Suite DevelopmentStudio.
- **Rational Suite TestStudio.** Contains fields and rules that work with Rational TeamTest, Rational RequisitePro, Rational Purify, Rational

Visual Quantify, and Rational Pure Coverage. Provides code for the Rational ClearQuest and Rational RequisitePro integration. Use with Rational Suite TestStudio.

- **Rational Suite Enterprise.** Contains fields and hooks that work with all Rational products. Provides code for the Rational ClearQuest and Rational RequisitePro integration. Use with Rational Suite Enterprise.
- **UnifiedChangeManagement.** Supports the UCM process by integrating with ClearCase. Use with Rational ClearCase.

If you plan to associate Rational ClearQuest change requests with Rational RequisitePro requirements, you must create a Rational project using the Rational Administrator, shown in the next panel.

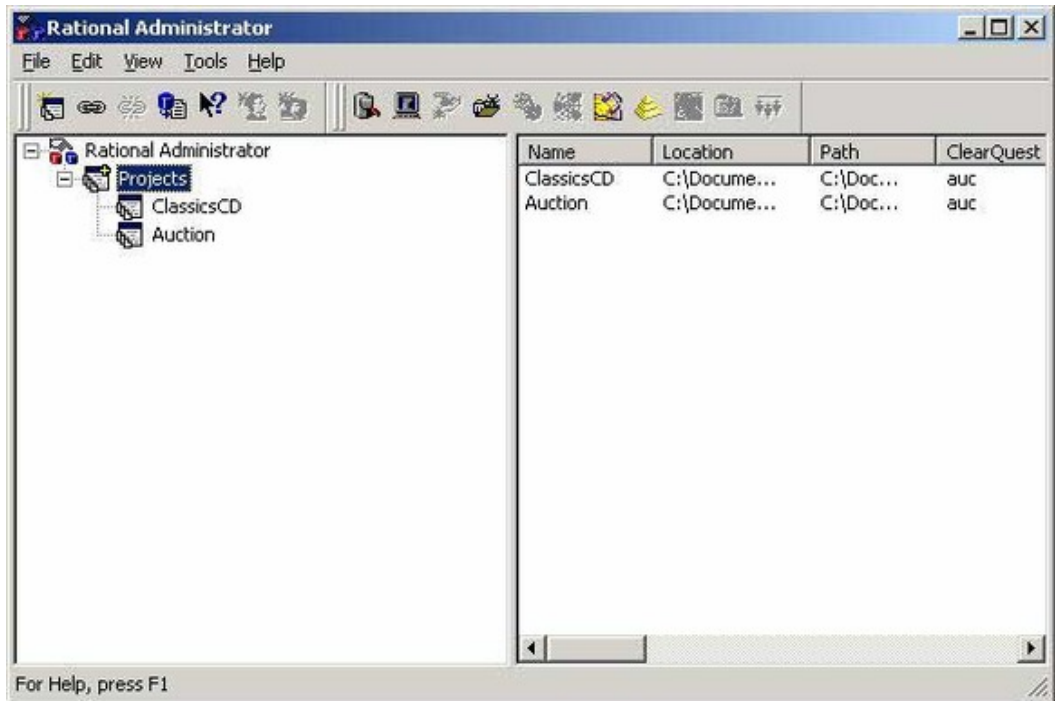
Creating a new Rational project

In Rational Administrator, a Rational project associates a Rational ClearQuest database, a Rational RequisitePro project, a Rational Rose model, and a TestManager project, all under ClearCase change management control (see [Resources](#)).

To create a new Rational project in Rational Administrator:

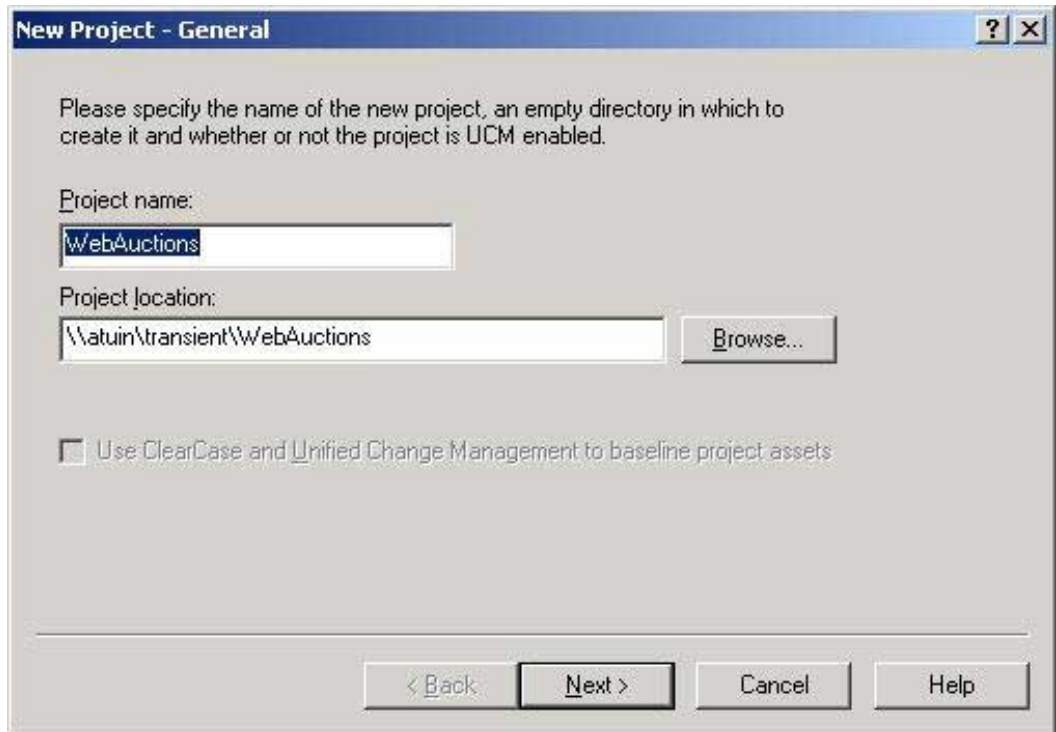
1. Open Rational Administrator. A window like the one shown in Figure 1 opens.

Figure 1. Rational Administrator



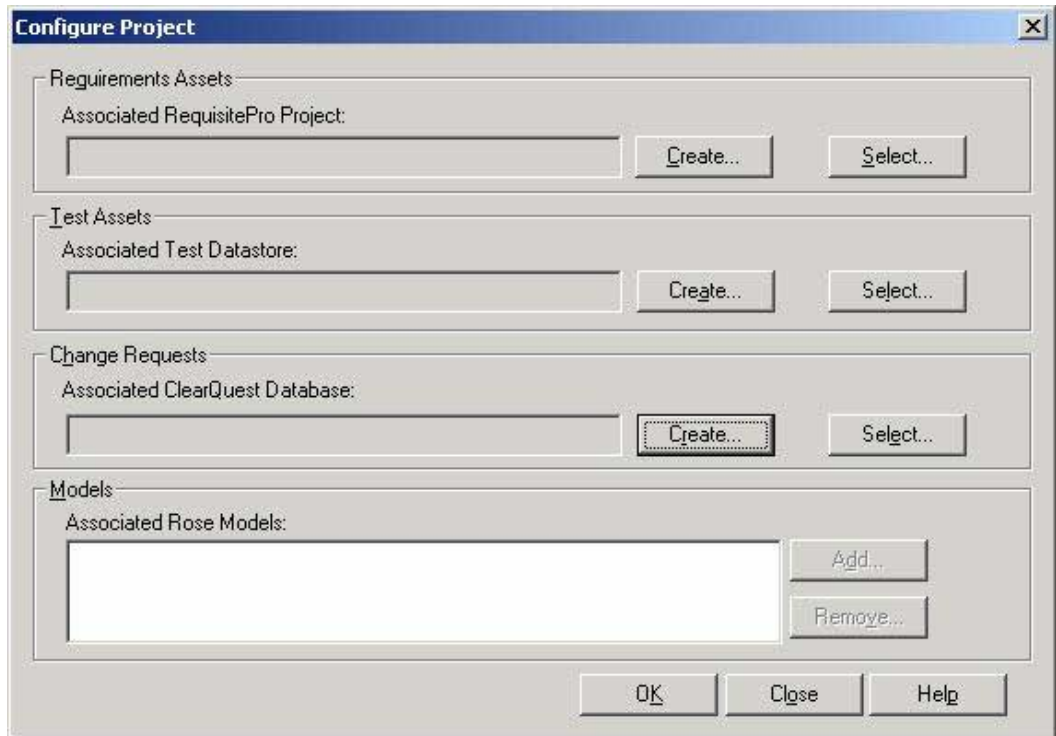
2. Right-click on Projects from the tree on the left panel and select **New Project...**
3. Name the project and define a location as shown in Figure 2. The location should be an empty directory. In a team environment, make sure that this directory is on a shared location on the network and is available through a UNC share (i.e., `\\machine\directory\project`) so that other users can access it.

Figure 2. Naming the project



4. Click **Next**.
5. Give the project a password. Note that this is just for controlling the project, not the individual entities. Rational ClearQuest, Rational RequisitePro and others preserve their existing user access mechanisms. Click **Next**.
6. Click **Finish** to finish the creation of the project. A window opens, shown in Figure 3.

Figure 3. Finishing the creation of the project



Add a Rational ClearQuest database

In the previous section you created a Rational project to use to associate different aspects of your system. To associate a Rational ClearQuest database to store and manage change requests:

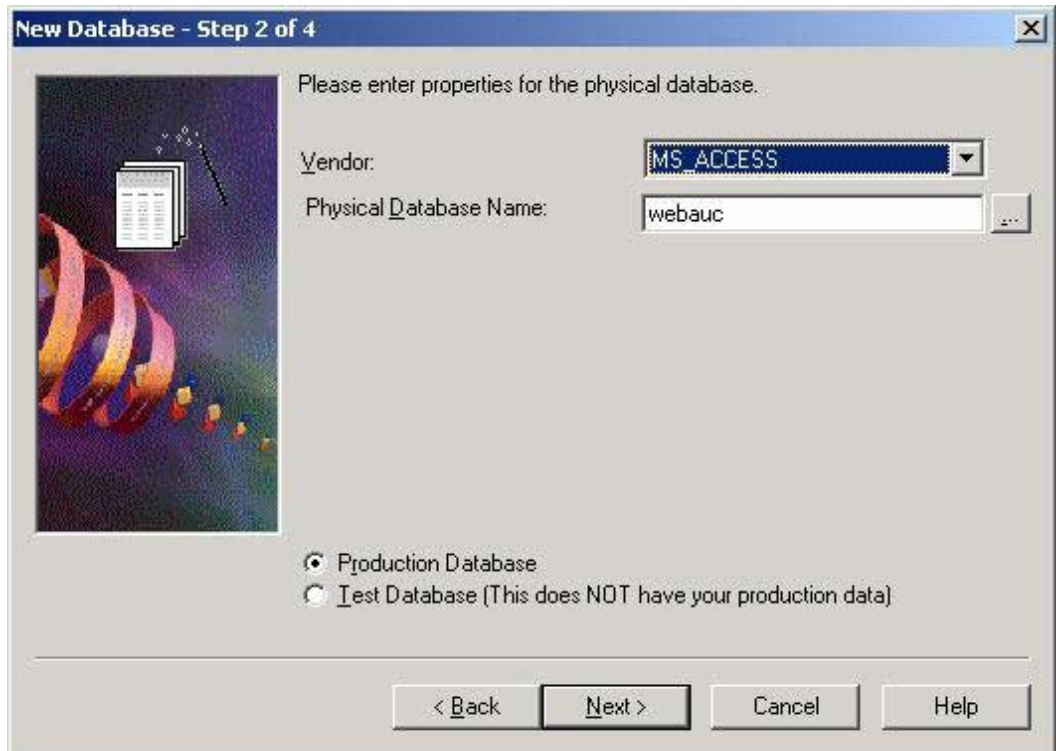
1. In the Configure Project window (Figure 3), click **Create** in the Change Requests section of the window.
2. Choose a database connection. A database was created when you installed Rational ClearQuest.
3. Name the database and if necessary, add a comment to describe the database content.

Figure 4. Naming the database



4. Choose the database in which you want to store the information (see Figure 5). Your choice depends on the database system you are using (for example, Access, SQL Anywhere, SQL Server). The only field common to all database systems is the name of the database. Other database systems might also add server names and authentication details.

Figure 5. Choosing the database

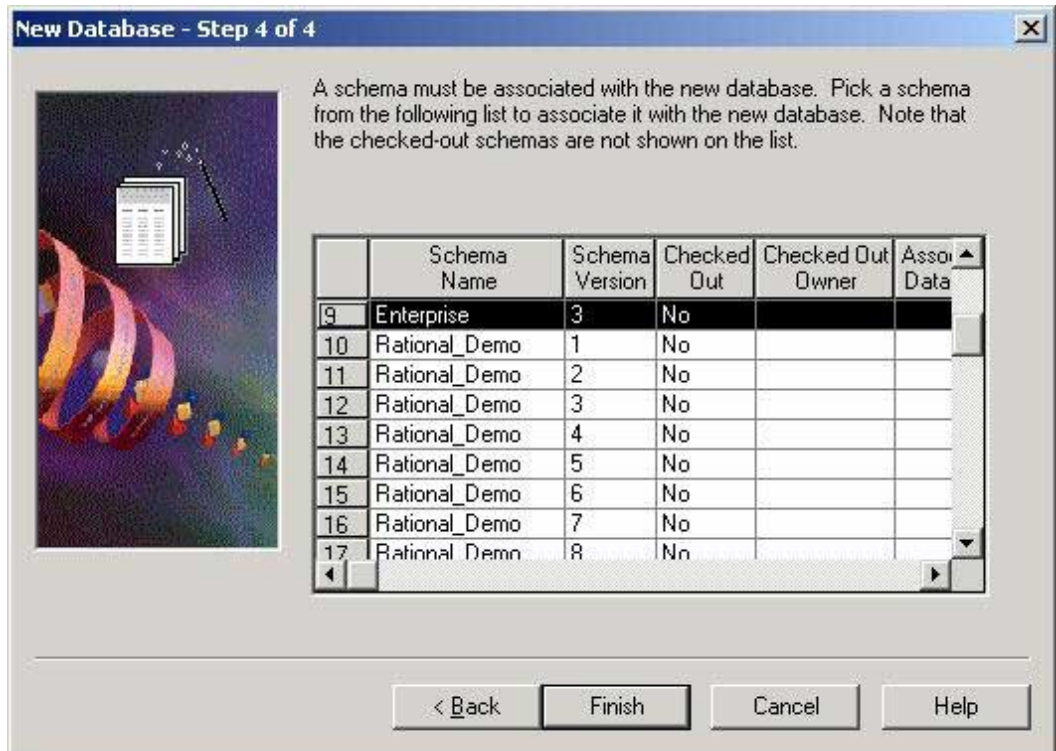


5. Specify the timeout and poll interval values. Default values are reasonable, so there is generally no need to change them (see Figure 6).
Figure 6. Specifying the timeouts



6. Choose a database schema. Figure 7 shows the Rational TestStudio schema selected. It provides the widest range of record types and information for working with all the Rational tools, including Rational TestStudio, Rational PurifyPlus and Rational RequisitePro.

Figure 7. Choosing the database schema



7. Click **Finish** to create the database.

To have defects automatically created when test scripts fail, create a test datastore for the project as described next.

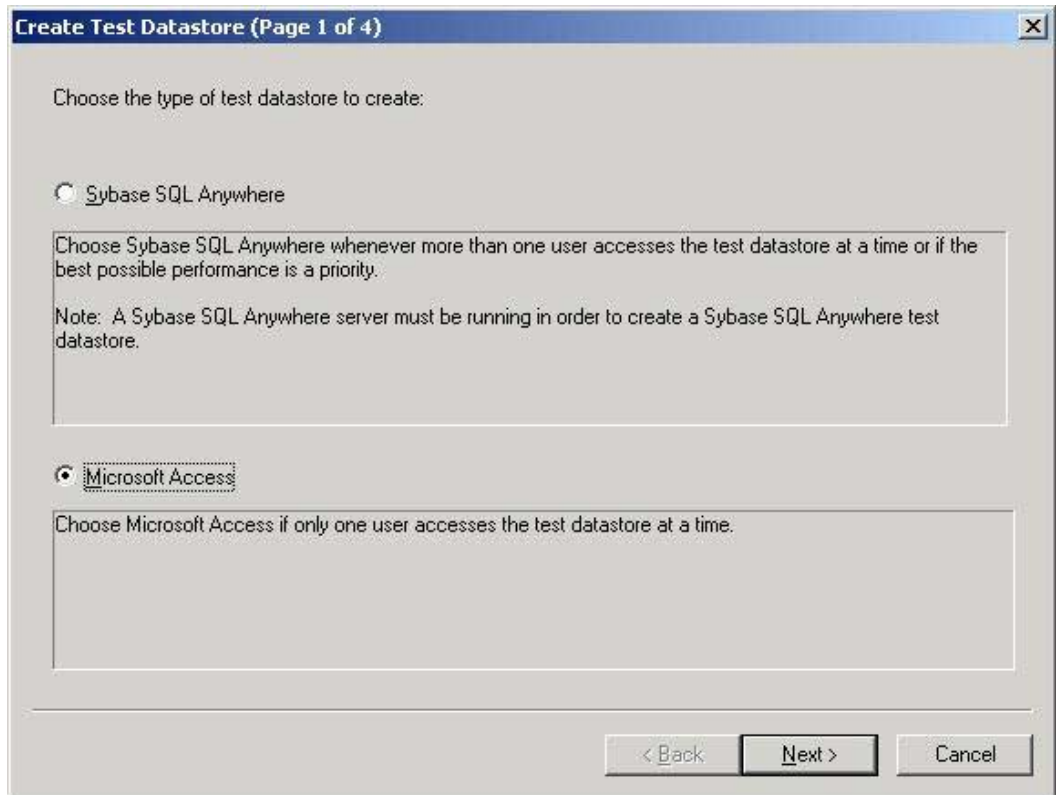
Creating a test datastore for your project

To directly create defects raised by the various test software provided in the Rational suite, create a test datastore. This holds the results and information from testing and allows you to automatically include all the testing details into the defects entered in Rational ClearQuest.

To create a new test datastore:

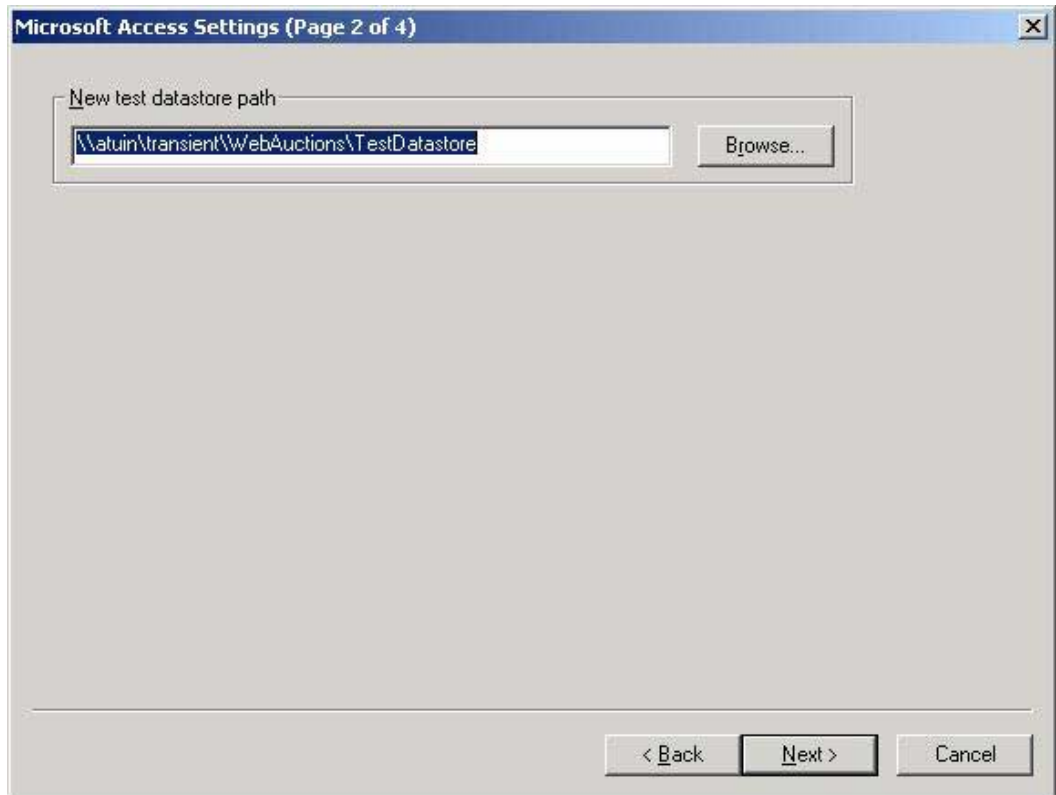
1. From the Configure Project window, click **Create** in the Test Assets section.
2. Choose the database type to be used for the datastore, as shown in Figure 8. If you have more than one tester, use Sybase SQL Anywhere so that any tester can update the information. If you are the only tester, use Microsoft Access.

Figure 8. Creating the test datastore

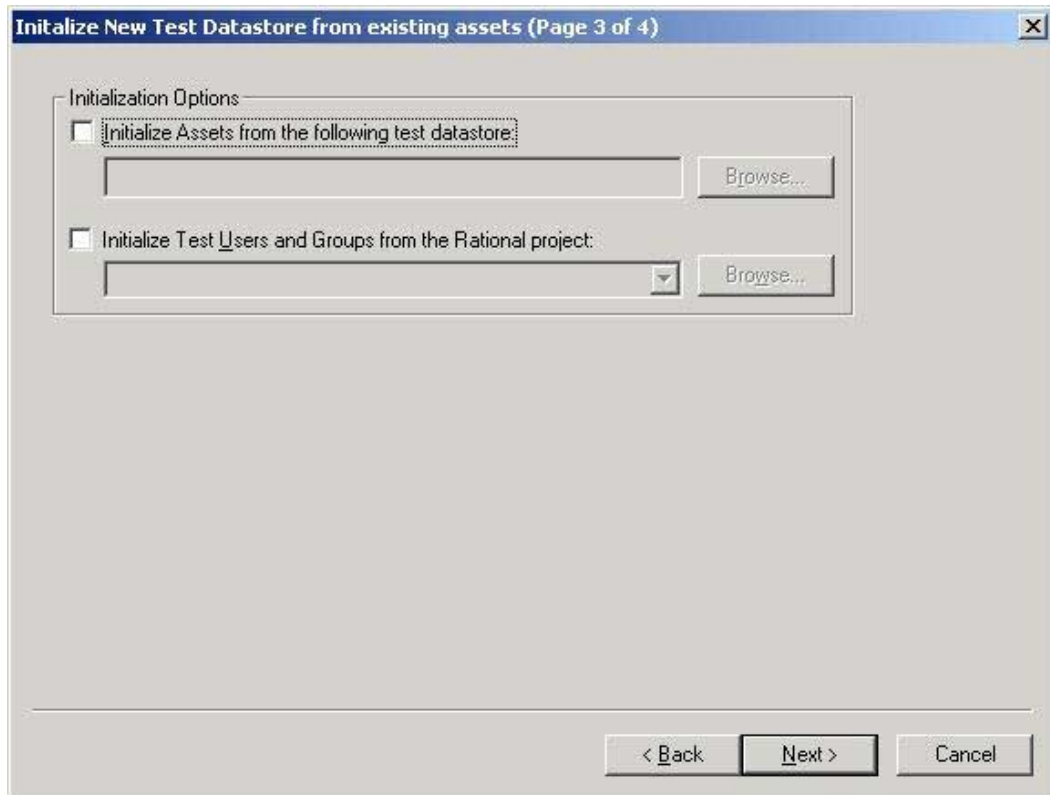


3. Choose a location for the datastore database (if using MS Access) or specify the connectivity options for one of the other database solutions supported, as shown in Figure 9.

Figure 9. Choosing a location for the datastore database



4. Choose an existing datastore to initialize this test datastore, as shown in Figure 10. This can be useful if you have created a test datastore in a previous phase before using the integration system. You can also choose to initialize the list of users and groups from an existing Rational project.
Figure 10. Choosing an existing datastore to initialize this test datastore



5. Click **Finish** to create the database.

Now when you test, the test report information can automatically be logged into defects. This is a big time-saving step, as you would otherwise have to manually enter the defect detail.

You are now ready to start collecting change requests and charting their progress through the development process.

Logging enhancement requests

Although some change requests are going to be created and introduced into the system outside of Rational ClearQuest, for example from TestManager, PurifyPlus or other Rational Components, some change requests are recorded directly within the Rational ClearQuest system as part of the role of the project manager or analyst responsible for change management.

In general, data collection through Rational ClearQuest solves many of the problems inherent in collecting enhancement requests because of the flexibility of the schema system, which can be used to specify very precisely what information is required by the analyst during data entry.

Information submitted into Rational ClearQuest can be provided either through a standard Windows interface or through the Rational ClearQuest Web interface allowing both internal users and external stakeholders (such as the customer) direct access into the system.

To create a new enhancement request using the Windows interface, click the button on the toolbar or select **Actions > New....** A window similar to the one shown here in Figure 11 opens. Note that this window is generated from a restricted user account, designed for entering data into the system at a basic level. Analyst level tools are described in detail later in this section. Fields shown in red are required, while tabs that have required but unpopulated elements are similarly highlighted with a red square. As a team, decide which fields should be mandatory and which ones optional.

Figure 11. Submit enhancement request

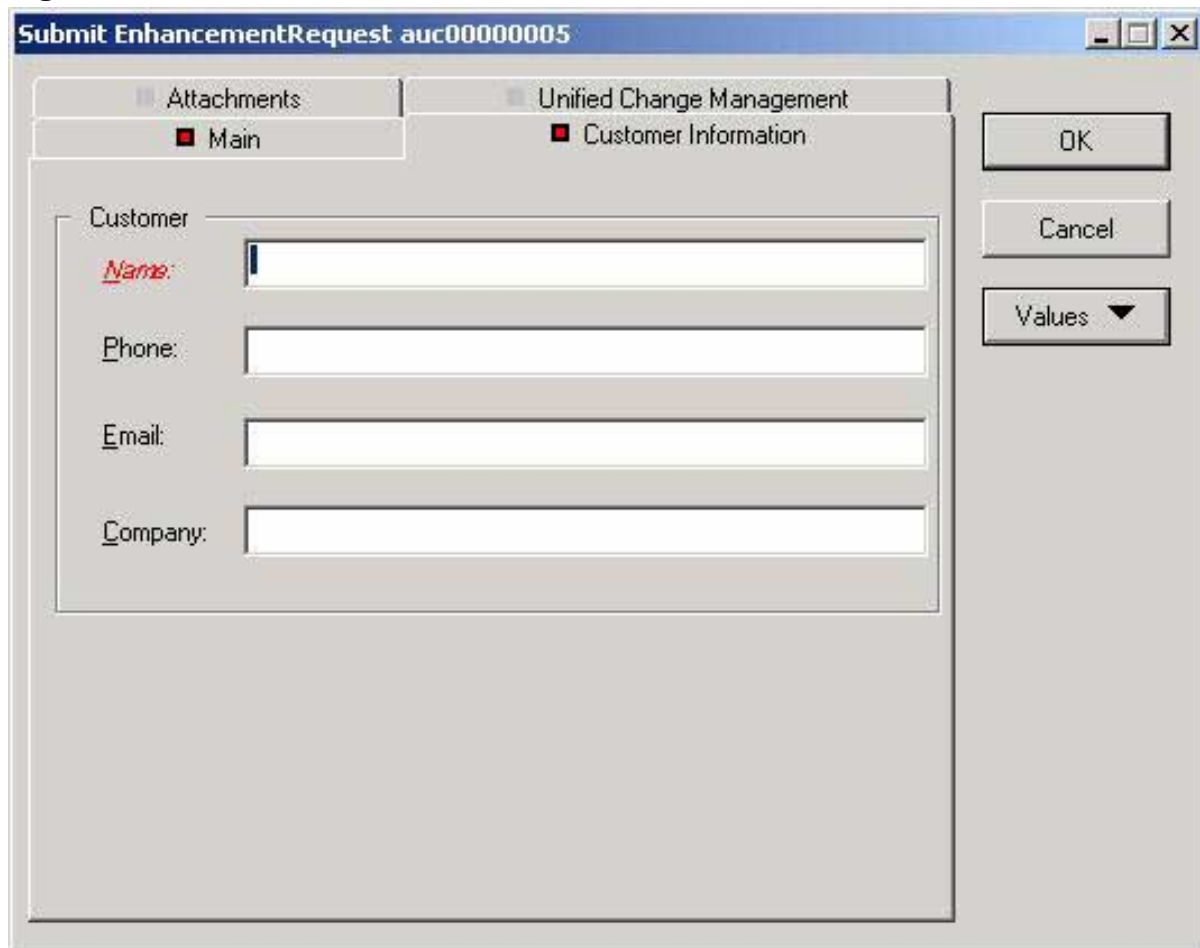
The screenshot shows a Windows-style dialog box titled "Submit EnhancementRequest auc00000005". It has a tabbed interface with "Attachments" and "Unified Change Management" tabs at the top, and "Main" and "Customer Information" tabs below. The "Main" tab is selected. The "ID" field contains "auc00000005" and the "State" field contains "Submitted". The "Headline:" field is empty and highlighted in red. The "Customer Priority:" field is a dropdown menu, also highlighted in red. The "Description:" field is a large text area. On the right side, there are "OK", "Cancel", and "Values" buttons.

In the example here, you can see that the basic information required for an enhancement request is very simple: A headline identifies the entry and a customer priority defines how important the request is. There's also space for a more detailed

description of the enhancement. The key is that all change requests contain the same information (customer priority, description, etc.) making it easier to evaluate and compare them than if they were stated in voice mails, hallway conversations, sticky notes, or napkins.

The Customer Information tab shown in Figure 12, is used to record the information about the customer who requested the enhancement. This is crucial in the case when the description is unclear or confusing. By having a contact, the analyst can validate their understanding of the request. In this case, you can enter the information. But, you can also set up schemas to use a predefined list of customers and a GUI form that provides the ability to use only this customer information.

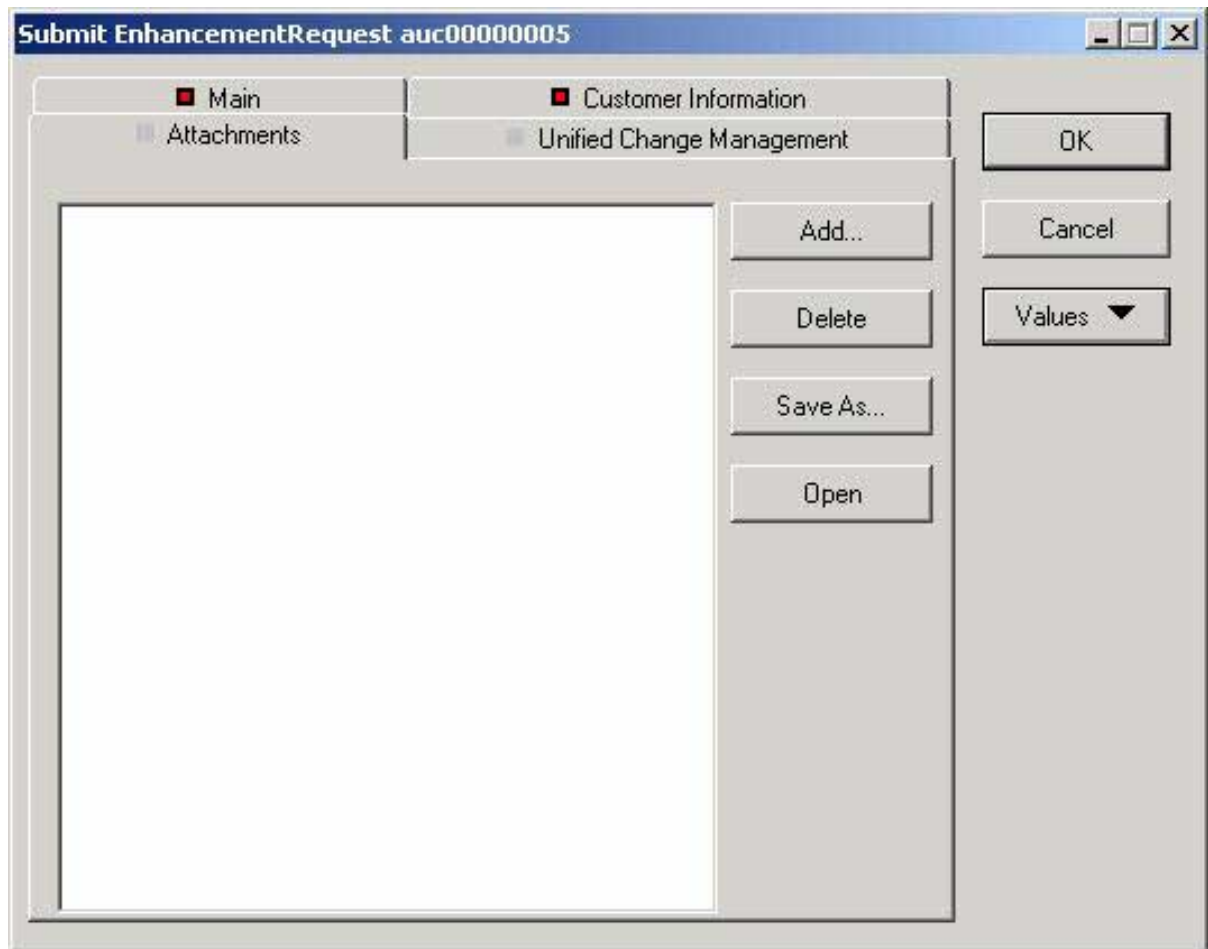
Figure 12. Customer Information tab



The screenshot shows a dialog box titled "Submit EnhancementRequest auc00000005". It features two main tabs: "Attachments" and "Unified Change Management". The "Unified Change Management" tab is selected and contains a sub-tab labeled "Customer Information". This sub-tab includes four text input fields for "Name", "Phone", "Email", and "Company". To the right of these fields are three buttons: "OK", "Cancel", and "Values" with a dropdown arrow.

The final tab of particular interest is shown in Figure 13. Through the Attachments tab the requestor can add documents, diagrams, and any other type of file to help describe the contents of the enhancement request in more detail.

Figure 13. The Attachments tab



Note that this format is just an example. The schema and the GUI used to populate the information is completely flexible. You can use Rational ClearQuest to track and trace any information you need to help you or the change analyst to manage the requests.

Logging defects

The process for logging defects is identical to the basic method for logging enhancements; typically you just use a different form and update the information into a different database with the Rational ClearQuest database schema that you selected. Although Rational ClearQuest also allows you to use the same form and database and provides a field to mark the change request as Defect or Enhancement Request.

The difference between enhancements requests and defects is the amount of information and the level of detail that you typically need to track. In an enhancement, the key data is the description of the enhancement, for example, "I

want to add other pictures to an auction item". In a defect, the problem will be related to a specific part of the application and through association, a specific requirement that generated that feature in the application. The defect can also come up during test, in which case you will want to log the test information.

Furthermore, because defects will be specific to a particular iteration of the application development, you want to be able to identify which iteration and version of the application and its associated component led to the defect being reported.

When used with an outside tool such as TestManager or PurifyPlus (covered in Part 5 of this series) a lot of this information is automatically recorded and entered into the system for you, that is the role of the integration between these tools and Rational ClearQuest.

To create a new defect, use the button on the toolbar, or choose **Actions > New....** The Submit Defect window is shown in Figure 14.

Figure 14. Creating a new defect

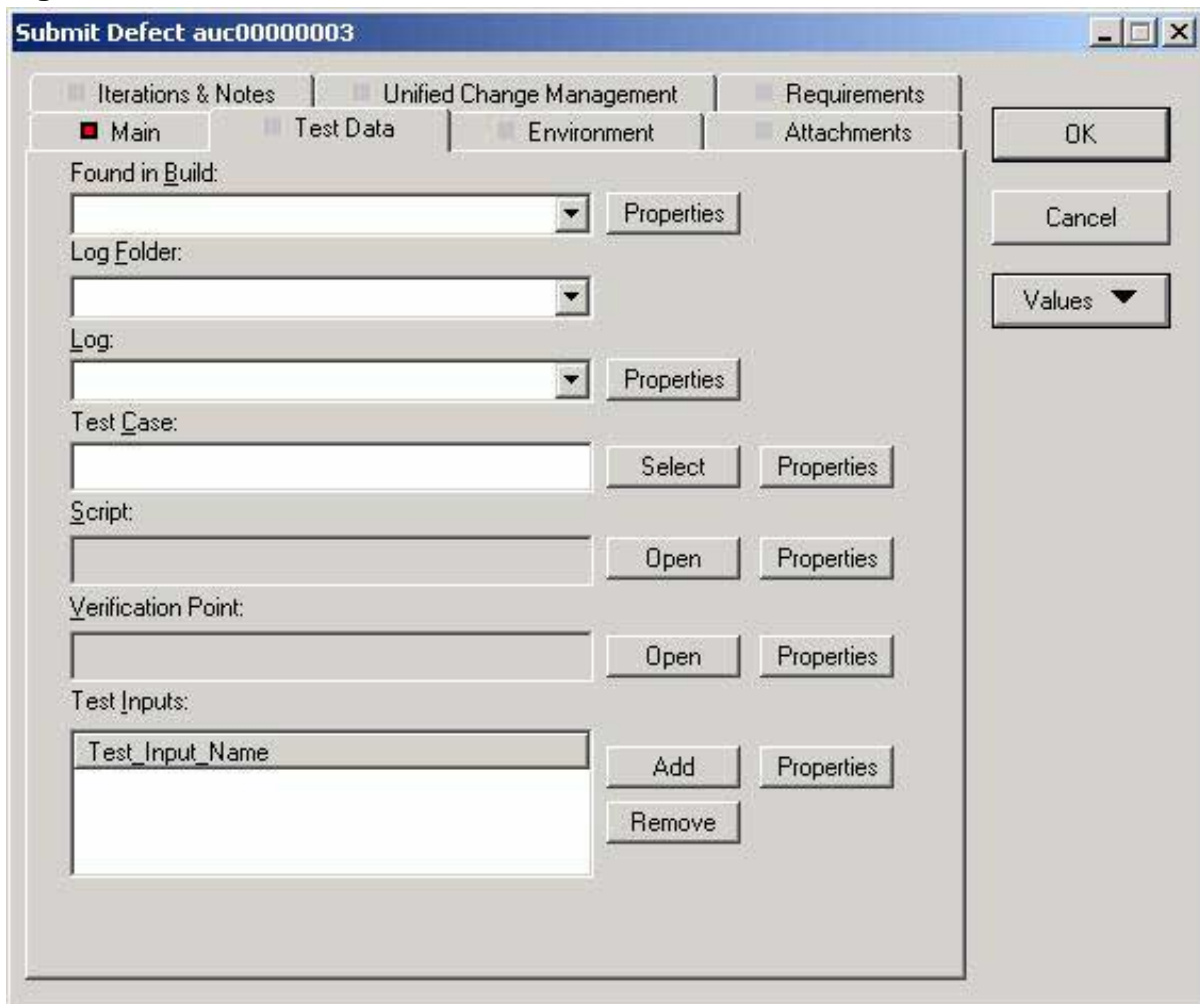
The screenshot shows the 'Submit Defect' dialog box with the following fields and controls:

- Window Title: Submit Defect auc00000003
- Navigation Tabs: Iterations & Notes, Unified Change Management, Requirements, Main (selected), Test Data, Environment, Attachments
- ID: auc00000003
- State: Submitted
- Headline: (empty text field)
- Suite Project: (dropdown menu)
- UCM Project: (dropdown menu)
- Owner: (dropdown menu)
- Priority: (dropdown menu)
- Severity: (dropdown menu)
- Customer Priority: (dropdown menu)
- Description: (large text area)
- Keywords: (text field with browse button ...)
- Symptoms: (text field with browse button ...)
- Buttons: OK, Cancel, Values (dropdown menu)

As you can see, the level of detail required is higher, but much of it, in this schema at least, is optional. As with the enhancement request, there are only two vital pieces of information, this time the headline description and the severity. The other information, keywords, symptoms and the associated projects and ownership of the defect are optional items. If you have a quick look at the other tabs, you can see how Rational ClearQuest starts to integrate into the rest of the Rational testing products.

When a test script fails while using Rational TestManager, the Test Data tab automatically lists the location of the test, the build of the application, the location of the test logs, test case, and input data that led to the defect. That is the key value of the Rational ClearQuest - TestManager integration. No manual intervention is required to propagate the test information into the defect log.

Figure 15. Test Data tab



The Environment tab (shown in Figure 16) records the operating system and hardware that reported the fault.

Figure 16. Environment tab

The screenshot shows a dialog box titled "Submit Defect auc00000003". It features a tabbed interface with the following tabs: "Iterations & Notes", "Unified Change Management", "Requirements", "Main" (selected), "Test Data", "Environment", and "Attachments". The "Environment" tab is active, displaying the following fields:

- Reported By:** A group box containing "Contact:" and "Company:" text labels above two text input fields.
- Environment:** A group box containing "Hardware:" and "Computer:" text labels above two dropdown menu fields.
- Operating System:** A text label above a dropdown menu field.
- Other Environment:** A text label above a text input field.
- Other Fields:** A group box containing "Custom1:", "Custom2:", and "Custom3:" text labels above three input fields (two dropdown menus and one text field).

On the right side of the dialog, there are three buttons: "OK", "Cancel", and "Values" (with a downward arrow).

The Iterations & Notes tab (shown in Figure 17) records the iteration of the application that triggered the defect.

Figure 17. Iteration & Notes tab

The screenshot shows a dialog box titled "Submit Defect CLSIC00000132". It features a tabbed interface with the following tabs: "Main" (active), "Test Data", "Environment", "Attachments", "Iterations & Notes", "Unified Change Management", and "Requirements". In the "Main" tab, there are two dropdown menus for "Planned Iteration:" and "Actual Iteration:". Below these is a large text area for "New Note:". On the right side, there are three buttons: "OK", "Cancel", and "Values" with a dropdown arrow.

All change requests, enhancement or defect, and their detailed information are entered into the database either manually or automatically from TestManager. You can now start organizing, prioritizing and reporting on the information.

Building queries

Now that you have some data in the database, you can report on the detail. Rational ClearQuest queries provide customizable queries into the database and also in depth reporting and statistical information.

The basic structure of the query system is actually very similar to the Rational RequisitePro views that you looked at in Part 1 of this series. When defining a query, specify the set of fields you want to display in the query result set and specify the filtering mechanism used to select the desired records from the database.

Rational ClearQuest, of course, has more data to deal with and typically (depending

on the schema you use) a more complex structure. This has benefits and pitfalls. The benefits are the sheer flexibility you have for reporting and structuring the information. The pitfall is the sheer quantity of choice can sometimes be difficult to work with when you need that quick report. Predefined queries are provided. To address this, Rational ClearQuest allows you to create and save queries in the workspace. Define the fields to display and the filter parameters once, and then you can execute that query to report on current data at any time.

You'll create a simple query that shows a summary of the requests in the system, just to demonstrate the basics of creating a query in the system. Later, you'll use this to add detailed information about the requests into Rational RequisitePro.

Creating a simple query

To create a public or personal query within Rational ClearQuest:

1. Right-click on a folder within your workspace tree and select New Query. Personal Queries are specific to the current user. Public Queries, if you have the permission to create them, are available to all users.
2. Choose a record type to query against, as shown here in Figure 18.

Figure 18. Choosing a record type



3. In this example, choose to query against All Requests in the database,

regardless of whether they are defects or enhancement requests.

4. Click **OK**. The Rational ClearQuest Query wizard opens. On the first screen, choose an existing query on which to base your new query, as shown in Figure 19. To create a new one, click **Next**.

Figure 19. Rational ClearQuest Query wizard



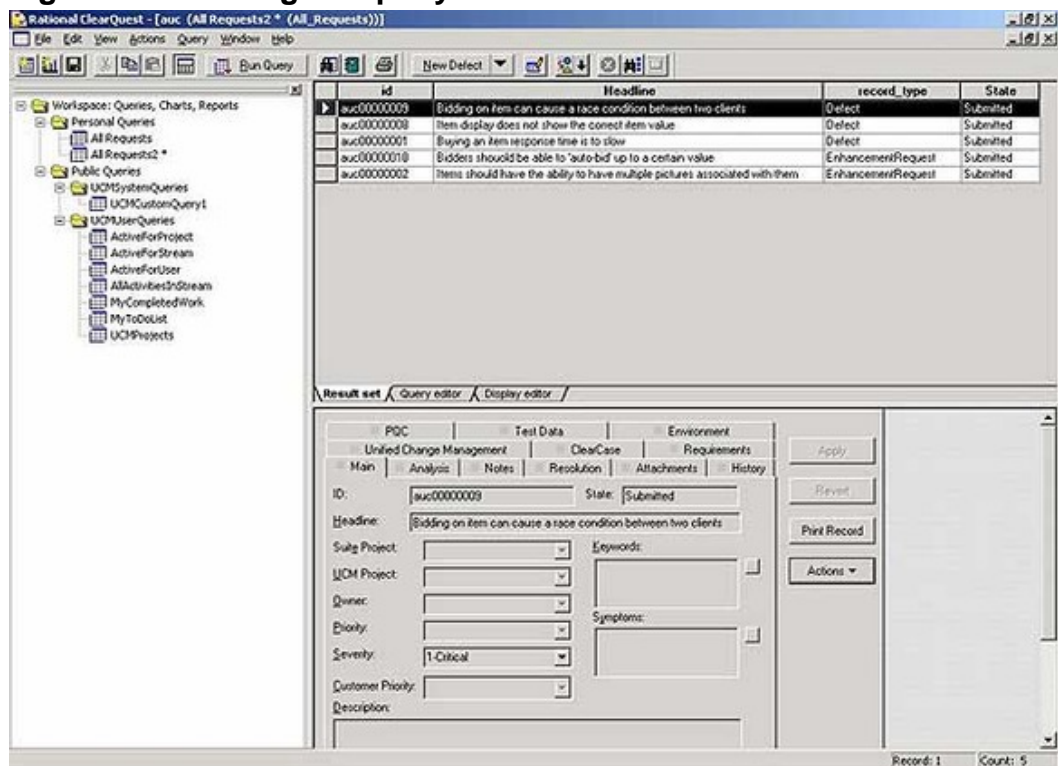
5. The display configuration page opens. This page defines the fields from the record type you specified in the first step to include in the query results. (Note that since you chose "All Requests", only those field names that exist in both the Defect and Enhancement Request record types are available to this query.) Drag the Headline, id, record_type and State fields from the panel on the left to the Display Format table on the right. Click in the box in the Sort column next to State to specify the State field as the primary sort field, as shown in Figure 20.

Figure 20. Defining how the query displays



7. Click **Run** to run the query and display the results. A window like the one in Figure 23 opens.

Figure 23. Running the query



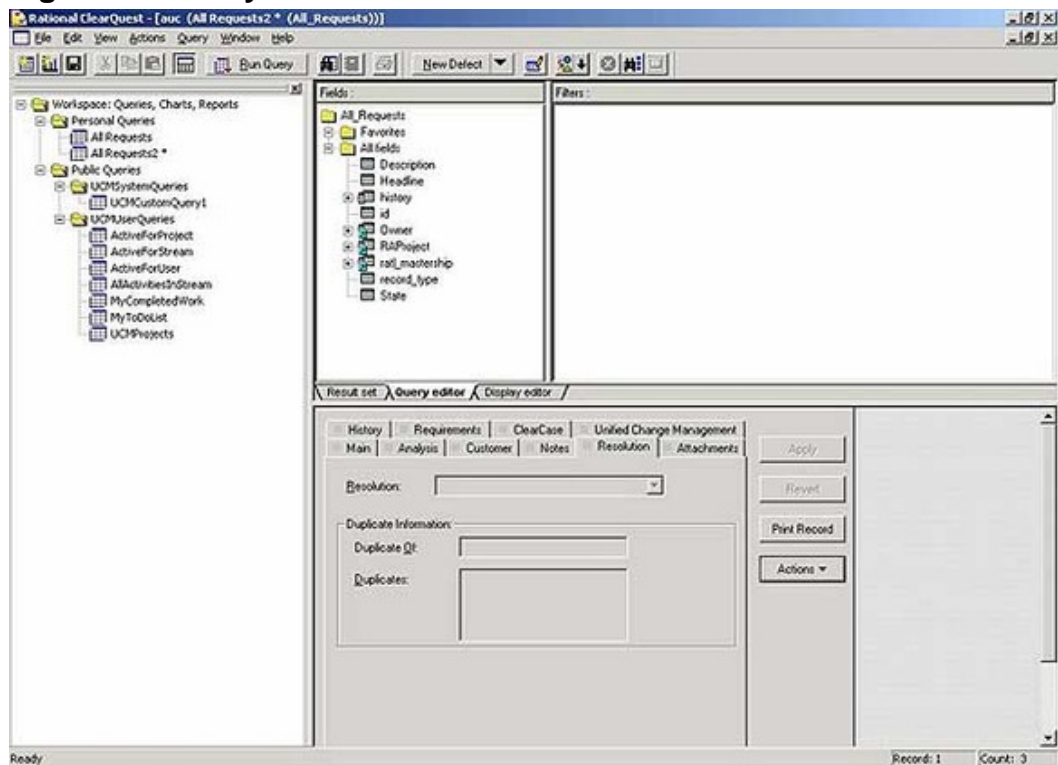
Filtering queries

The previous example only built a basic query on your requests that listed all the available entries. More useful are targeted queries that highlight specific sets of information, such as all the requests that have been submitted, but not assigned.

To add a filter, create an entirely new query basing it on the query you just created, or use the Query editor within the Query view to alter the parameters of an existing query.

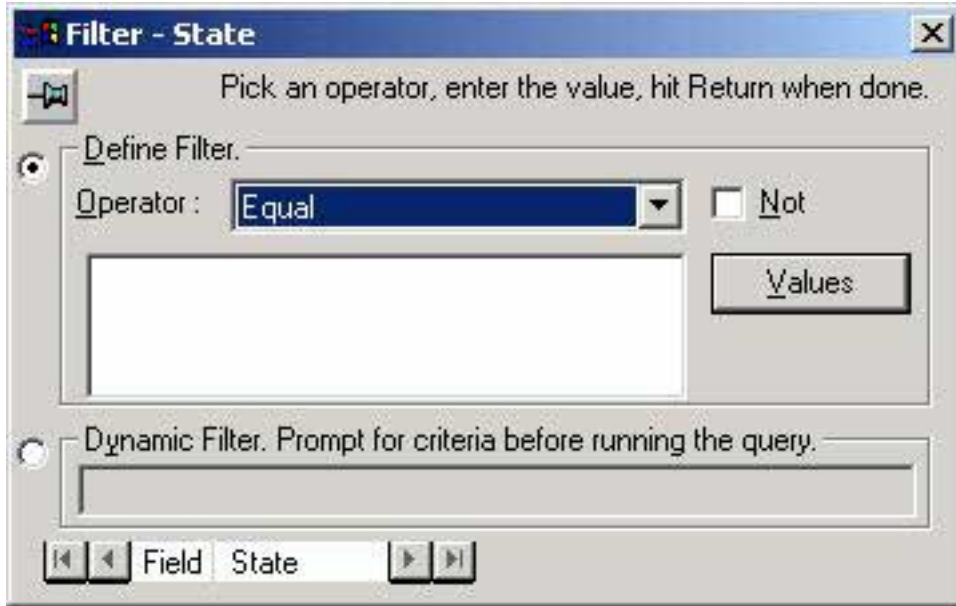
1. Alter the original query to show only requests that have been submitted to the system, but not assigned, resolved or closed.
2. Change to the Query editor tab.
3. Drag the State field from the Fields panel on the left to the Filters panel on the right. The window shown in Figure 24 opens.

Figure 24. Query editor tab



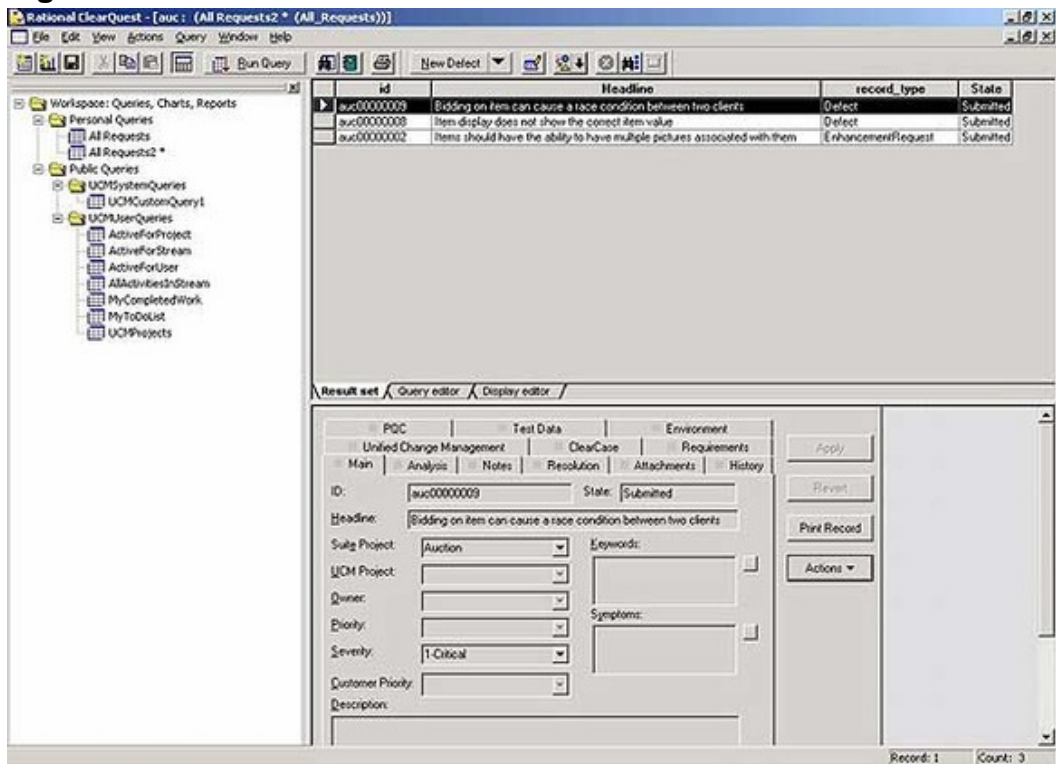
4. Adjust the filter to show States equal to Submitted by choosing Equal as the operator and then clicking the Values button to select from one of the predetermined State values (see Figure 25).

Figure 25. Adjusting the filter



5. Click the close box to close the filter definition.
6. Select the Result set tab to change to the result view and click on the Run Query button in the toolbar. Your display changes to show the reduced set of requests, as shown in Figure 26.

Figure 26. Result set tab



You can create other queries and filters based on the information above. Valuable queries are those that give you, at a glance, a list of the currently open, submitted but unassigned, closed/resolved and other similar straight answer reports. These help monitor the development status and workload within my project.

Classifying requests

Once a request has been submitted into the system, it is the role of the project manager or analyst to start further qualifying and organizing the requests. Especially in the case of enhancement requests, where the original submission provides a stakeholder/external view of the request, the software team needs to provide an internal perspective (is the request impossible to implement? Is it in line with your project objectives?) Ideally a change control board (a group of software team members representing testing, training, support and designers/coders) meet regularly to evaluate change requests that were posted/logged since the last meeting. In that meeting, they complement the change request information with their team perspective.

To further qualify enhancement requests, add whether it is a new feature or an enhancement to an existing feature, a target release version, product detail or other information. For a defect, specify a target iteration for the defect to be rectified or target release in which the defect should have been addressed.

You can see the Analysis tab for a defect in Figure 27, and an enhancement request in Figure 28.

Figure 27. Analysis tab for a defect

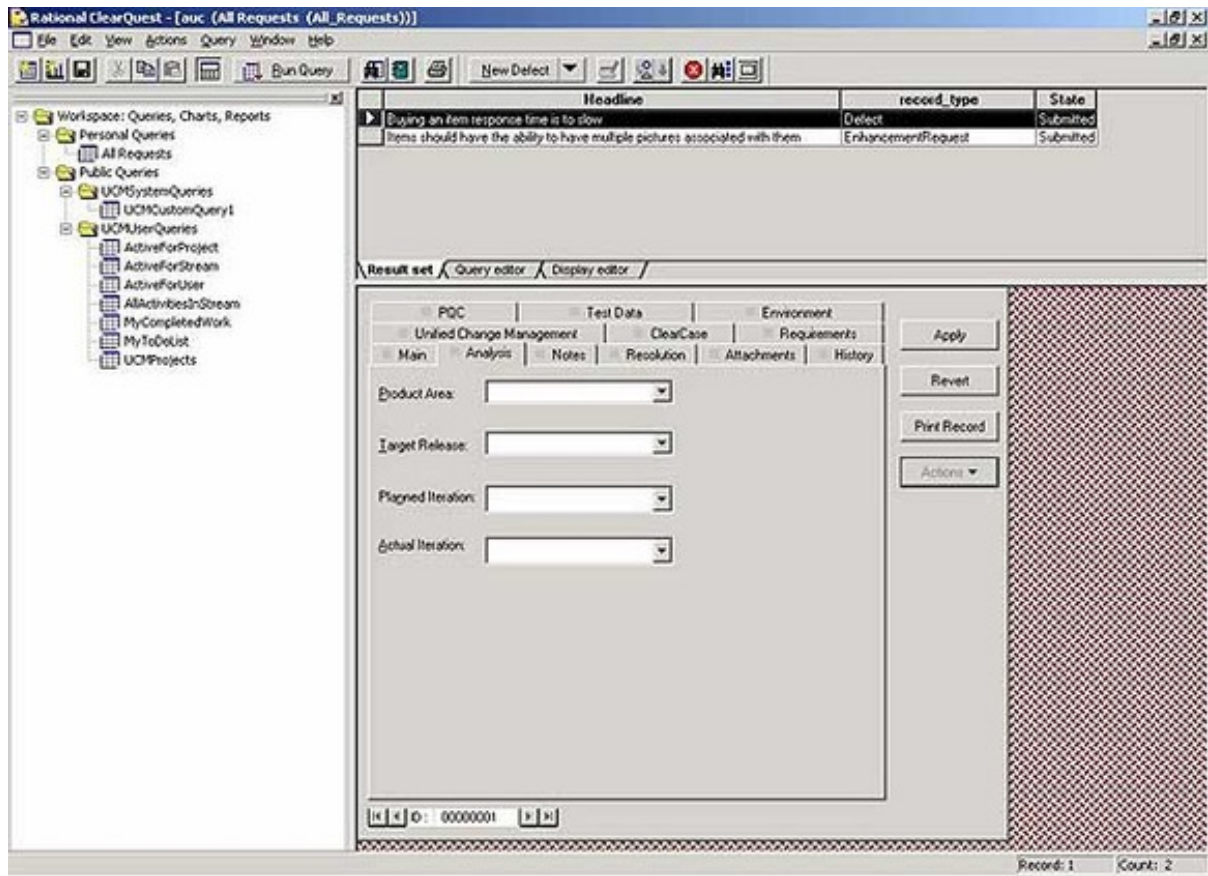
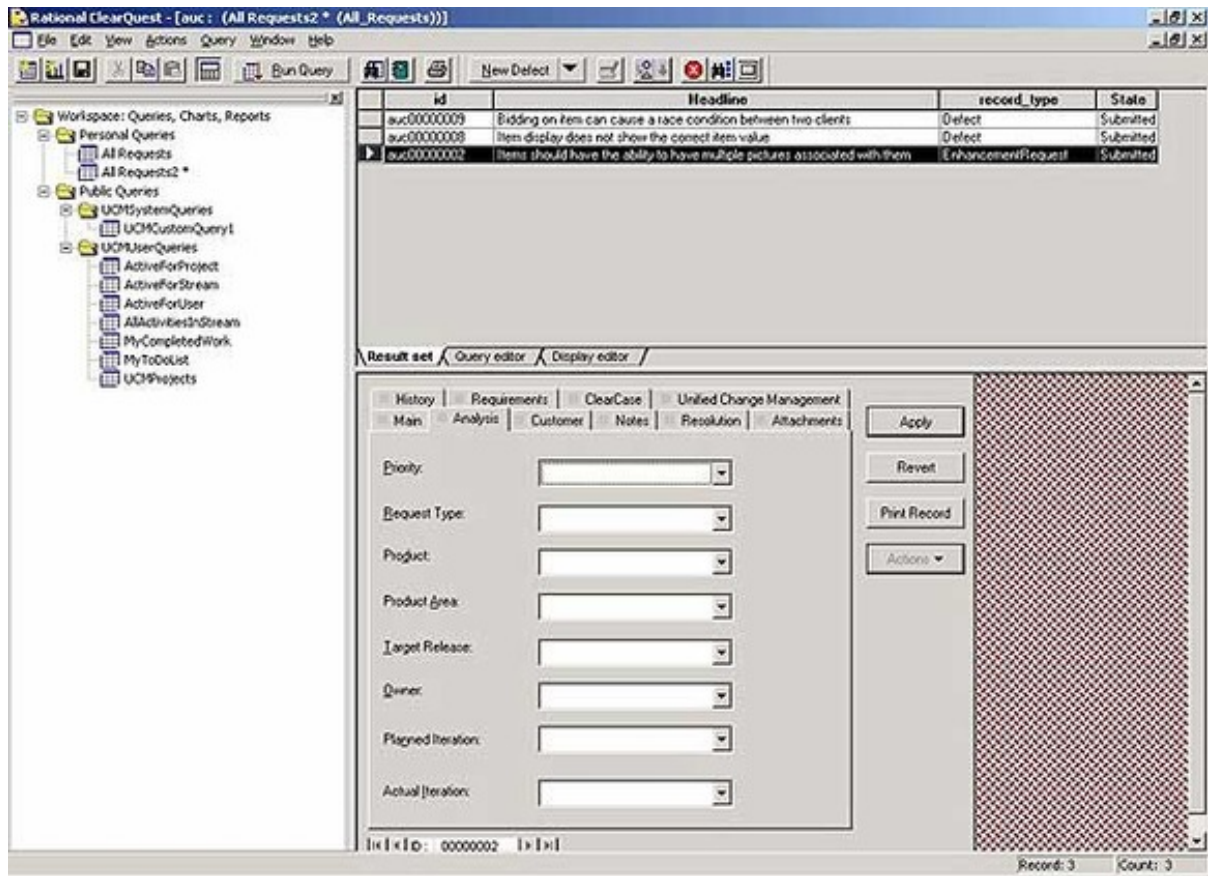


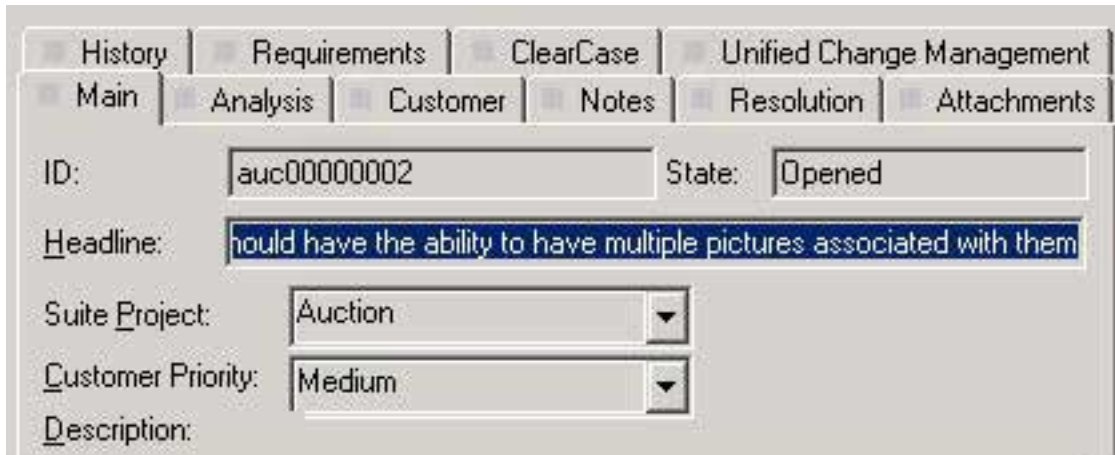
Figure 28. Analysis tab for an enhancement



Accepting or rejecting requests

Once all requests have been analyzed and the team perspective has been recorded, Rational ClearQuest queries provide a project manager a way to look at a group of requests at one time, based on the fields established in the Classifying Requests step above. Because each team has limited resources, decisions need to be made on which requests can be worked on and which ones are either rejected (for enhancement requests) or deferred (for enhancement requirements and defects).

Figure 29. The State field



The screenshot shows a software requirements management tool interface. At the top, there are several tabs: History, Requirements, ClearCase, Unified Change Management, Main, Analysis, Customer, Notes, Resolution, and Attachments. Below the tabs, there are several input fields and dropdown menus. The ID field contains 'auc00000002' and the State field contains 'Opened'. The headline field contains 'ould have the ability to have multiple pictures associated with them'. The Suite Project dropdown menu is set to 'Auction' and the Customer Priority dropdown menu is set to 'Medium'. The Description field is empty.

Assigning requests

For defects that are selected to be fixed in the next release, assign the request to a member of the development team who can be tasked with fixing or resolving the issue.

Assigning a change request to a developer involves changing the state of the request. The state of a change request indicates where that particular request is in the development lifecycle. Until a request has been assigned (or resolved) you must assume from a management standpoint that the request is still to be identified or assigned to a developer for resolution

Depending on the schema, the way assignment works differs. In the sample schema, assignment of a defect is simply a case of assigning the defect to a member of the development team. Assignment of enhancement requests also applies the requirement to a development team member, but you'll need to provide a priority for the request so that the assignee can prioritize the request with other tasks in his queue.

Minor enhancement requests to existing functionality can similarly be assigned to developers. These do not affect the requirement specification that describes the software delivered at the end of the project. However, enhancement requests that reflect new functionality first need to be incorporated into the requirements spec before being assigned to a designer to work on (see the section on Rational ClearQuest-Rational RequisitePro integration below). For minor enhancement requests, you'll need to provide a priority for the request so that the assignee can prioritize the enhancement requests with other tasks (defects) in his queue.

To assign a request, use a query to find the request you want to assign. For example, your previous query that returned all requests still in the Submitted state.

1. Select the request from the query result set.
2. Click the Actions button to the right of the record form.
3. Choose Assign.
4. Choose the Owner of this request from the drop-down user list.
5. Select Apply.

The results of assigning your auto-bid enhancement request is shown below.

Figure 30. Result of assigning the auto-bid enhancement request

The screenshot shows the Rational ClearQuest interface. On the left is a tree view of queries. The main area displays a table with the following data:

id	Headline	record_type	State
auc00000009	Bidding on items can cause a race condition between two clients	Defect	Submitted
auc00000008	Item display does not show the correct item value	Defect	Submitted
auc00000007	Buying an item response time is too slow	Defect	Assigned
auc00000010	Bidders should be able to 'auto-bid' up to a certain value	EnhancementRequest	Assigned
auc00000002	Items should have the ability to have multiple pictures associated with them	EnhancementRequest	Submitted

Below the table is a form for assigning the request. The 'Owner' field is set to 'Alex'. Other fields include Priority (Medium), Request Type, Product, Product Area, Target Release, and Planned Iteration. The 'Apply' button is visible on the right side of the form.

Resolution of requests

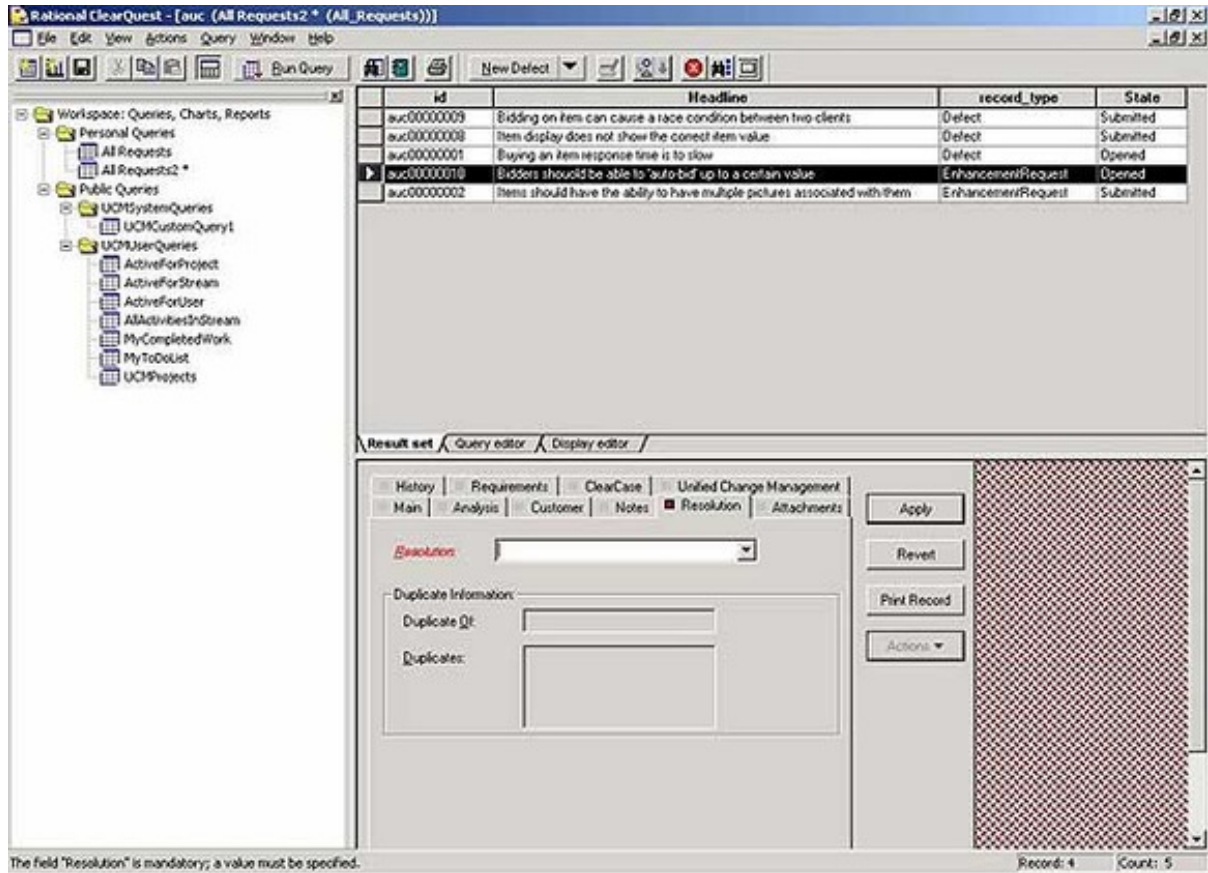
Once a particular defect has been fixed or an enhancement has been implemented, the final stage is to highlight the request as being resolved. This can be done automatically from TestManager or PurifyPlus if you are specifically dealing with a known and correctly tagged defect request.

In other situations, you'll need to modify the request manually to demonstrate the resolution of the request. First, you need to open a request. This marks the request as open (and therefore being worked upon) within Rational ClearQuest. Generally, users (developers) will manually mark the request as Open during the course of fixing or identifying the bug. This also changes the state of the request to Opened.

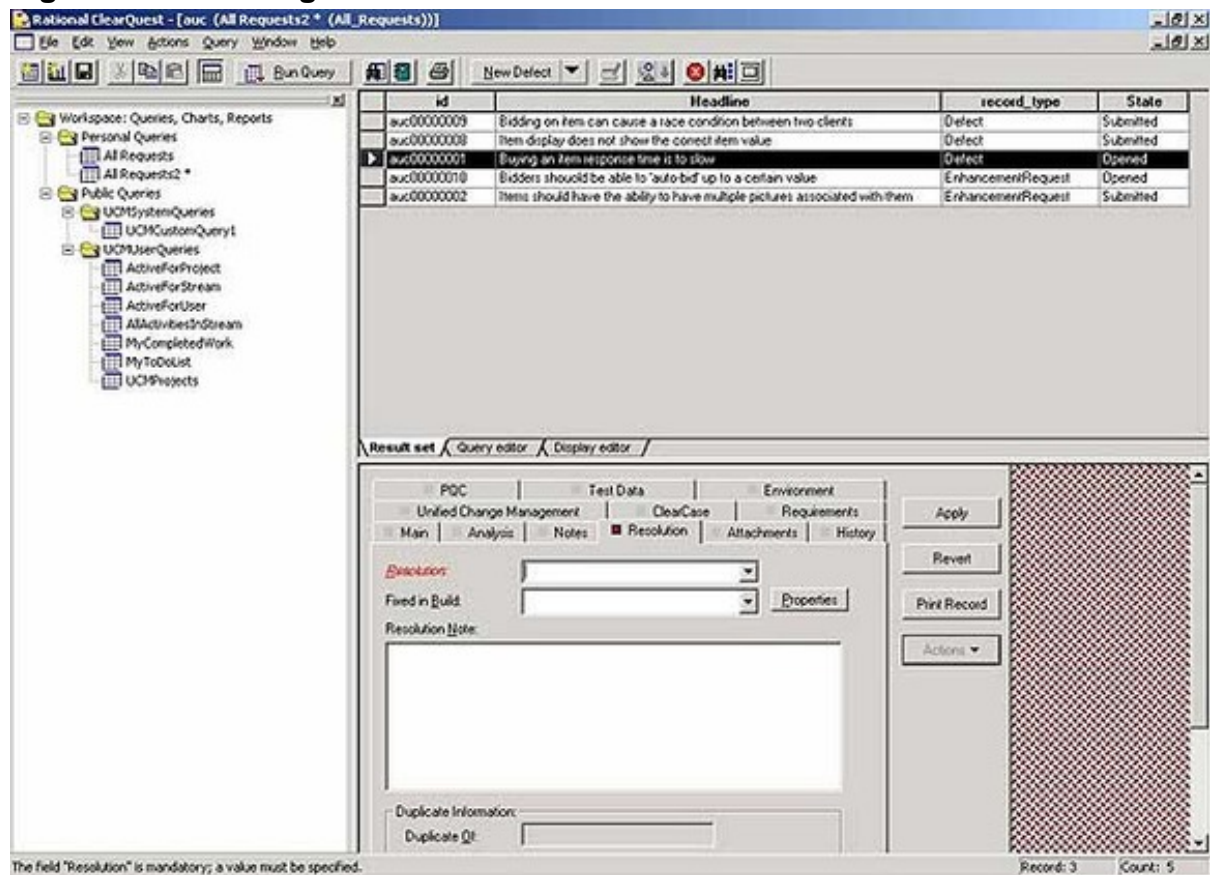
A developer typically uses a "My ToDo List" query to find work assigned to them. This query shows all requests in the Assigned state, where the owner of the request is the CURRENT_USER. CURRENT_USER is a keyword that can be used in Rational ClearQuest queries and matches the name of the currently logged-in user. To Open a request, find the request you want to work on and select **Open** from the Actions button. You can fill in some information about what you are doing, or click Apply to mark the item as open.

Once the problem has been rectified, or the enhancement request implemented, select **Close** from the Actions button and fill in the form shown below in Figure 31.

Figure 31. Resolution request form



If the request was a defect, provide details of the resolution. You can see a sample of this in Figure 32.

Figure 32. Providing details of the resolution

You've now followed a request from its inception through to reporting and finally closing the request within the system. But what if a request is related to an original requirement in your requirements specification?

To share information between these packages, integrate with Rational RequisitePro.

Section 4. Integrate enhancement requests into a requirement spec

Requests and requirements relationship

As mentioned earlier, once enhancement requests logged in Rational ClearQuest are accepted by the team, make sure the requirement spec is updated with that

newly added functionality so that developers and testers relying on the requirement spec to do their job (model, classes, code, as well as validation procedures) actually work on the latest set of requirements, and not an obsolete specification.

The Rational ClearQuest-Rational RequisitePro integration enables you to link enhancement requests with their associated requirements. Generally the two main request types integrate in different ways:

- Enhancement requests -- these are nearly always tied to an existing feature or requirement of the system, or are an enhancement of existing functionality. For example, the auto-bid feature is an enhancement of the original 'Buyers can bid on an item' feature. Enhancement requests can also relate to missing functionality, in which case a new requirement will be created in the requirement spec to represent that missing functionality.
- Defects -- these are attached to the use case definition for the object or class that is known to be causing the problem. Tracking defects in this way can be used to highlight which requirement the defect is related to. This is beneficial in analyzing patterns in poorly defined requirements. If lots of defects are attached to a use case, this tells you that the use case might not have been flushed out enough before the implementation work started. A lesson learned for your next project.

Integration between Rational ClearQuest and Rational RequisitePro works by creating an association between a Requirement type within Rational RequisitePro and a change request type in Rational ClearQuest.

Associating a Rational RequisitePro project with your Rational project

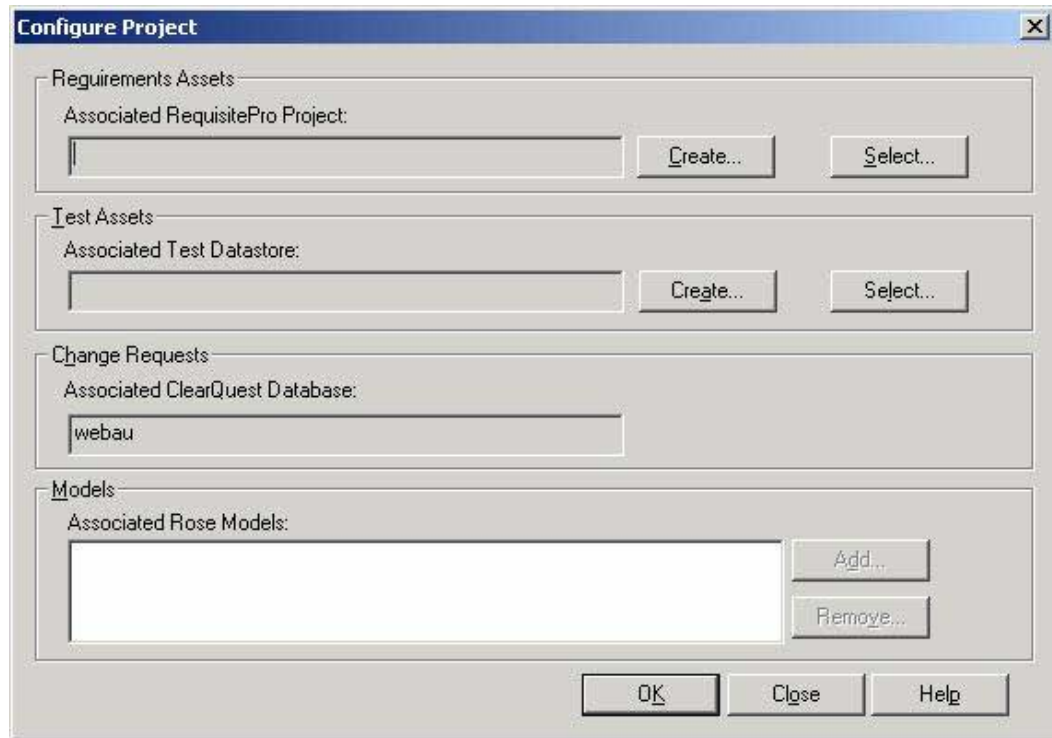
You're going to use Rational Administrator again to associate the Rational RequisitePro project you created in Part 1 with the Rational ClearQuest database you created in Part 3. The Enterprise Schema that you used to create the database already includes the necessary code to integrate with a Rational RequisitePro project.

To associate your Rational RequisitePro project:

1. Close Rational ClearQuest if it's open (not required, but recommended).
2. Open Rational Administrator.
3. Right-click the WebAuction project you created and choose **Configure**. You'll be asked to enter the password if you created one for the Rational

project. A window like the one shown in Figure 33 opens.

Figure 33. Configure Project window



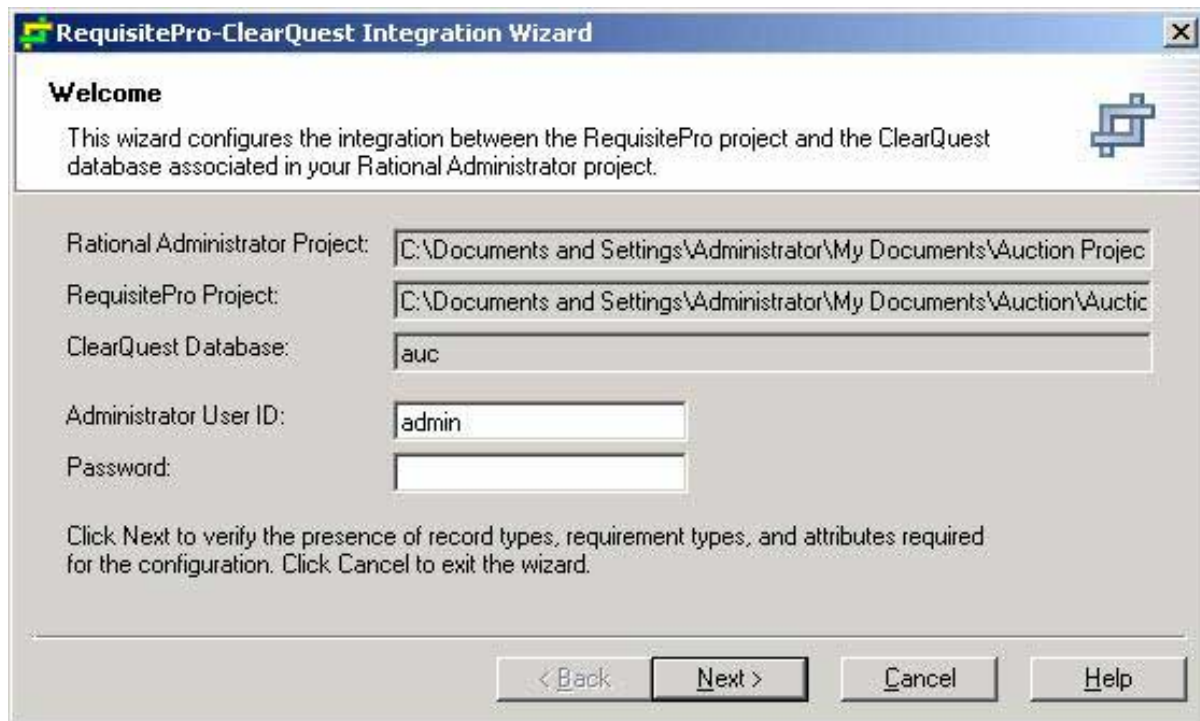
4. Click **Select** in the Requirements Assets section to select the Rational RequisitePro project that you want to associate with the Rational ClearQuest database. Use the browser to select the Rational RequisitePro project file.

Setting up the Rational ClearQuest-Rational RequisitePro integration

Once the Rational RequisitePro project has been associated with your Rational project, the Rational ClearQuest-Rational RequisitePro Integration wizard automatically starts. The wizard takes you through the basics of associating the Rational RequisitePro project with the Rational ClearQuest database, including modifying the Rational RequisitePro project to incorporate the new requirement types that are used to help trace and track the integration between the two systems.

The first screen summarizes the main information about the integration. You'll need to supply the appropriate login and password for the Rational ClearQuest database. You don't need to change anything here (see Figure 34).

Figure 34. Main information about the integration



The Choose Setup Type panel controls how information is associated between the two tools.

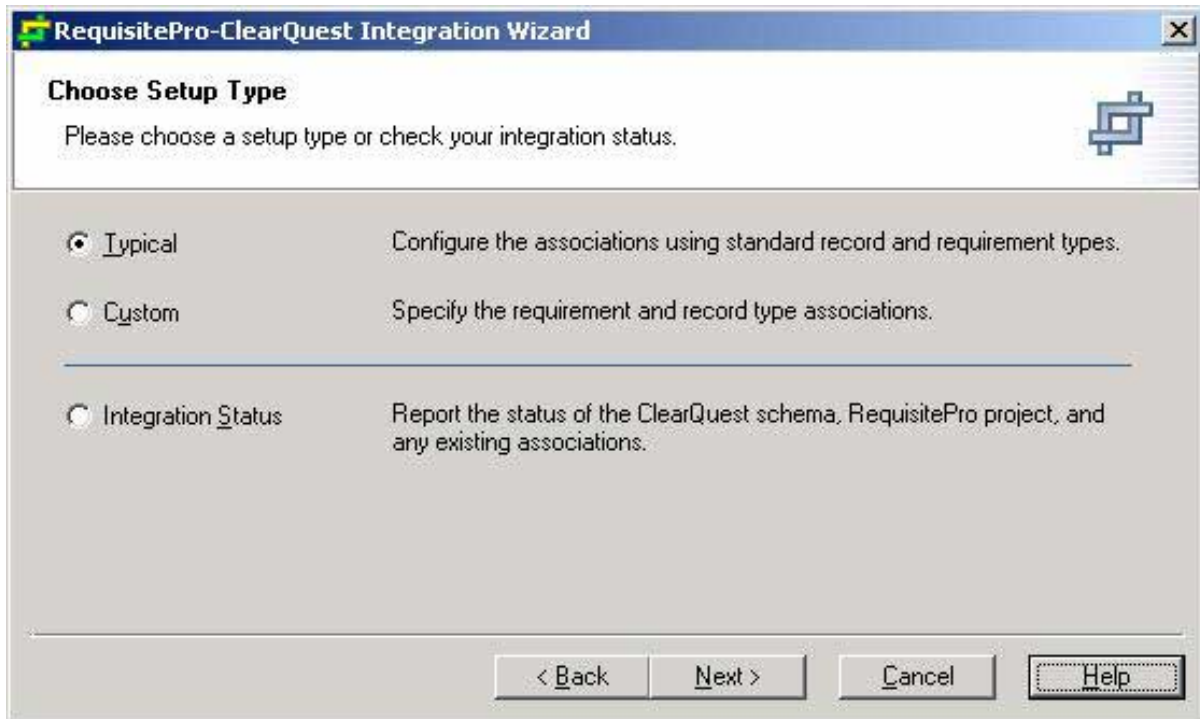
The Typical option links Rational ClearQuest Defects and Enhancement Requests records types to either Feature or Use Case requirement types.

If you use other record or requirement types, or want to establish other associations, select the Custom option to select which Rational ClearQuest record types to associate with which Rational RequisitePro requirement types.

Select **Integration Status** to report on the state (and any issues) of the integration.

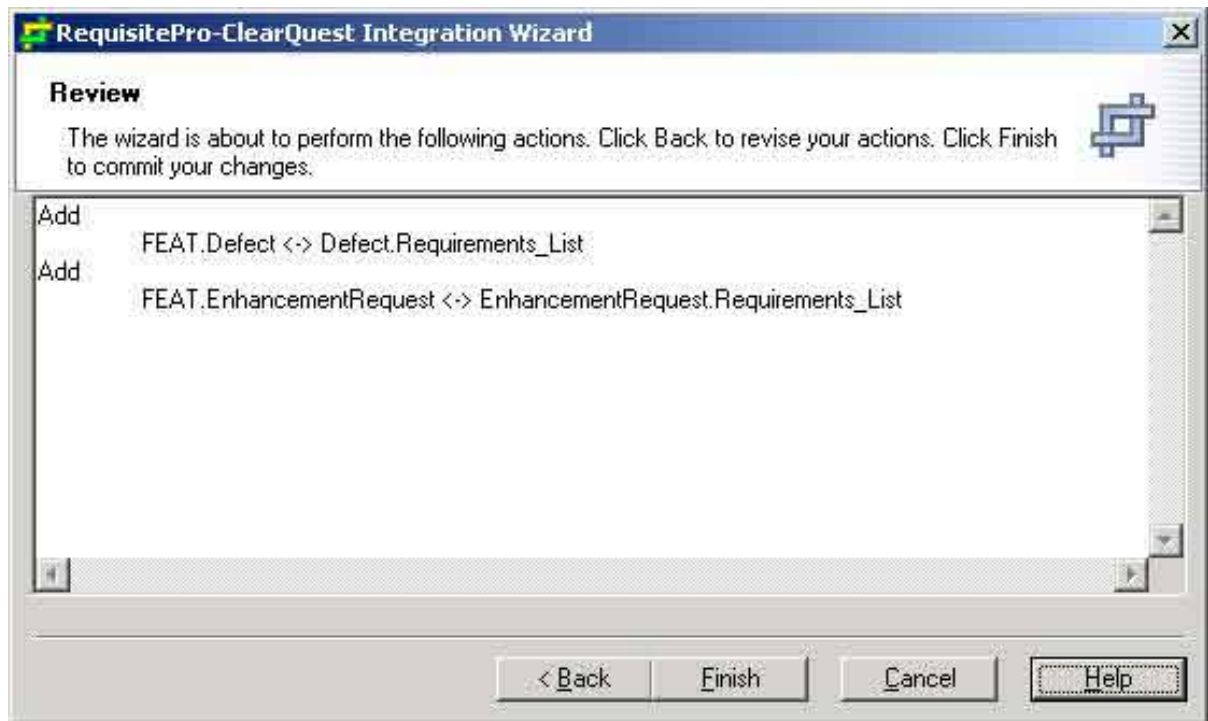
For now, use the Typical setting, as shown in Figure 35.

Figure 35. Choosing the setup type



The final screen summarizes what the integration wizard is going to do, as shown in Figure 36. Because you selected the Typical option, the wizard establishes associations between Rational RequisitePro Feature and Use Case requirement types and Rational ClearQuest Defects and Enhancement requests.

Figure 36. Summary of what the integration wizard is going to do



Let the wizard do its work. Once it's finished, you are returned to the main Project Configuration window.

You're now ready to start associating the features definitions in your original requirements specification with requests in Rational ClearQuest. Start by associating an enhancement request with one of the original requirements.

Section 5. Build a new requirements specification

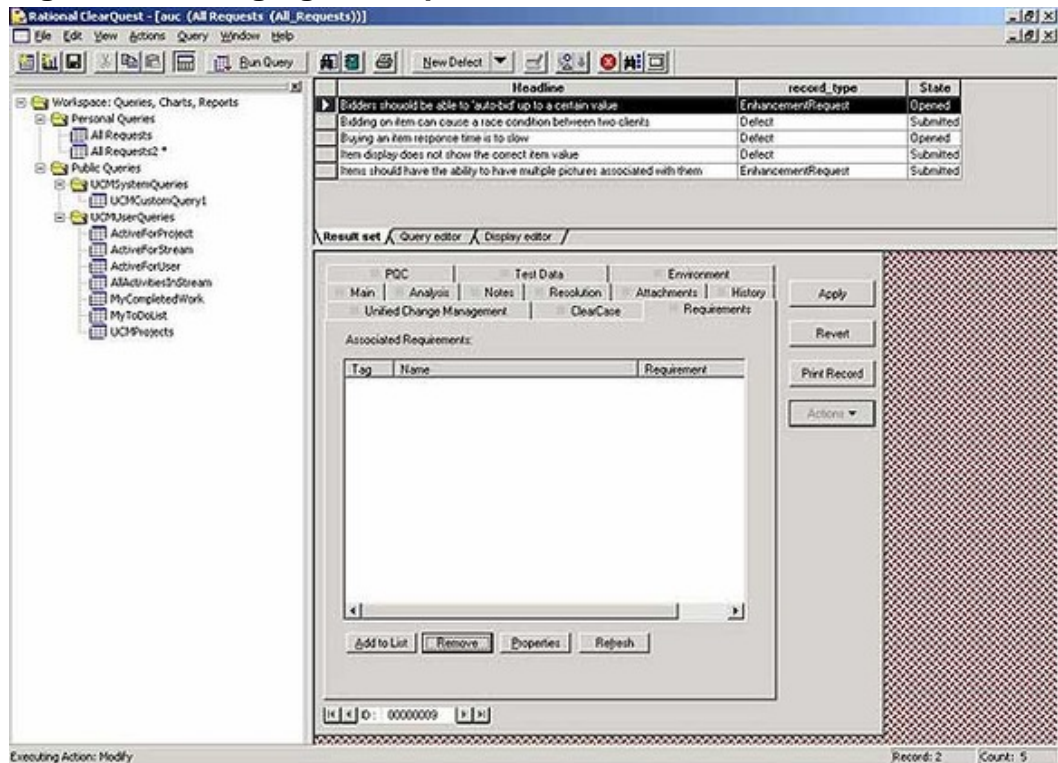
Creating a request-requirement association

Associating a request to a requirement gives you visibility into the origin of requirement and requirement changes. The information produced can also be used to provide metric information, such as the requirement or feature with the highest number of faults or enhancements that can be used by managers to target resources and prevent 'feature creep' in a product.

To create an association:

1. In Rational ClearQuest, locate the enhancement or defect you want to associate to a requirement. In this example, associate the auto-bid enhancement to a feature request. Use one of the queries you created earlier to find the request.
2. Click **Modify** from the Actions button.
3. In the Main tab select the name of the Rational project that you want this request to be related to. The Rational project is called Auction.
4. Change to the Requirements tab as shown in Figure 37.

Figure 37. Changing the requirements tab

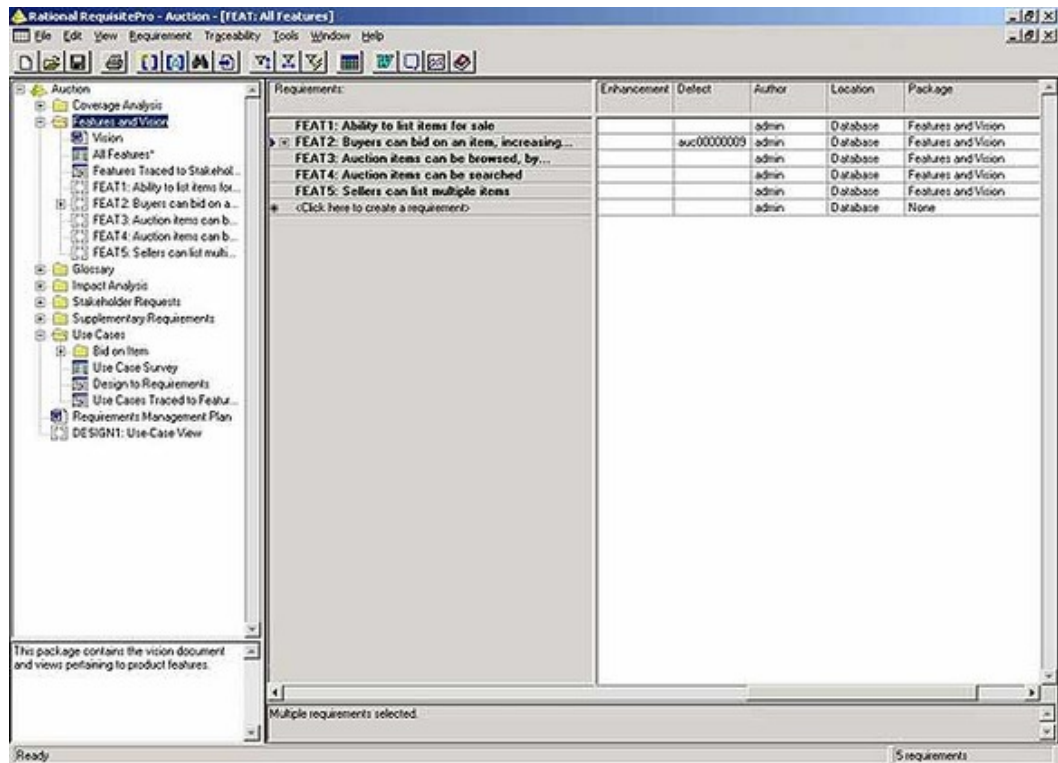


5. Click **Add to List**. A list of all the requirements in the Rational RequisitePro project associated to this Rational ClearQuest database is displayed (see Figure 38). Select **FEAT: Feature** requirement type and select **Buyers can bid on an item** feature requirement.

Figure 38. Associating requirements



6. Close this window and click **Apply**.
 7. Now, if you go in Rational RequisitePro and select a Rational RequisitePro view that contains the list of features in the Defect or Enhancement Request attribute (shown in Figure 39), you'll see that the Rational ClearQuest change request number has been entered.
- Figure 39. List of features in Rational RequisitePro**



- Click on the cell that lists the Rational ClearQuest CR. A ... button is displayed. Click on it. The detailed change request information stored in Rational ClearQuest is displayed in a window. The integration provides dynamic access to Rational ClearQuest data from Rational RequisitePro. The value is to see the details of the enhancement requests that resulted in a change of requirement without leaving Rational RequisitePro.

Creating a new requirement

Some enhancement requests result into a change in existing requirements. Often, enhancement requests require the creation of new requirements in Rational RequisitePro. These new requirements are detailed further to create updated use cases and requirements specifications used to define the application.

Rational ClearQuest and Rational RequisitePro are typically connected through the Feature Requirement type, allowing you to create FEAT requirements directly from within Rational ClearQuest. Generally, enhancement requests are based around top-level features but they can also be related to more detailed enhancements. A good example of this is the auto-bid enhancement you've been working with in this tutorial.

To create a new requirement from within Rational ClearQuest related to a request:

1. Find the enhancement or defect that you want to associate to a requirement. You'll associate the auto-bid enhancement request to a new feature, so use one of the queries you created earlier to find the request.
2. Click **Modify** from the Actions button.
3. In the Main tab, select the name of the Rational project you created earlier (the one that connects the Rational RequisitePro project with the Rational ClearQuest database). The Rational project is called Auction.
4. Change to the Requirements tab and click **Add to List** to open the list of available requirements.
5. Click **Create** to create a new requirement. The Rational RequisitePro properties window opens, shown in Figure 40.

Figure 40. The Rational RequisitePro properties window

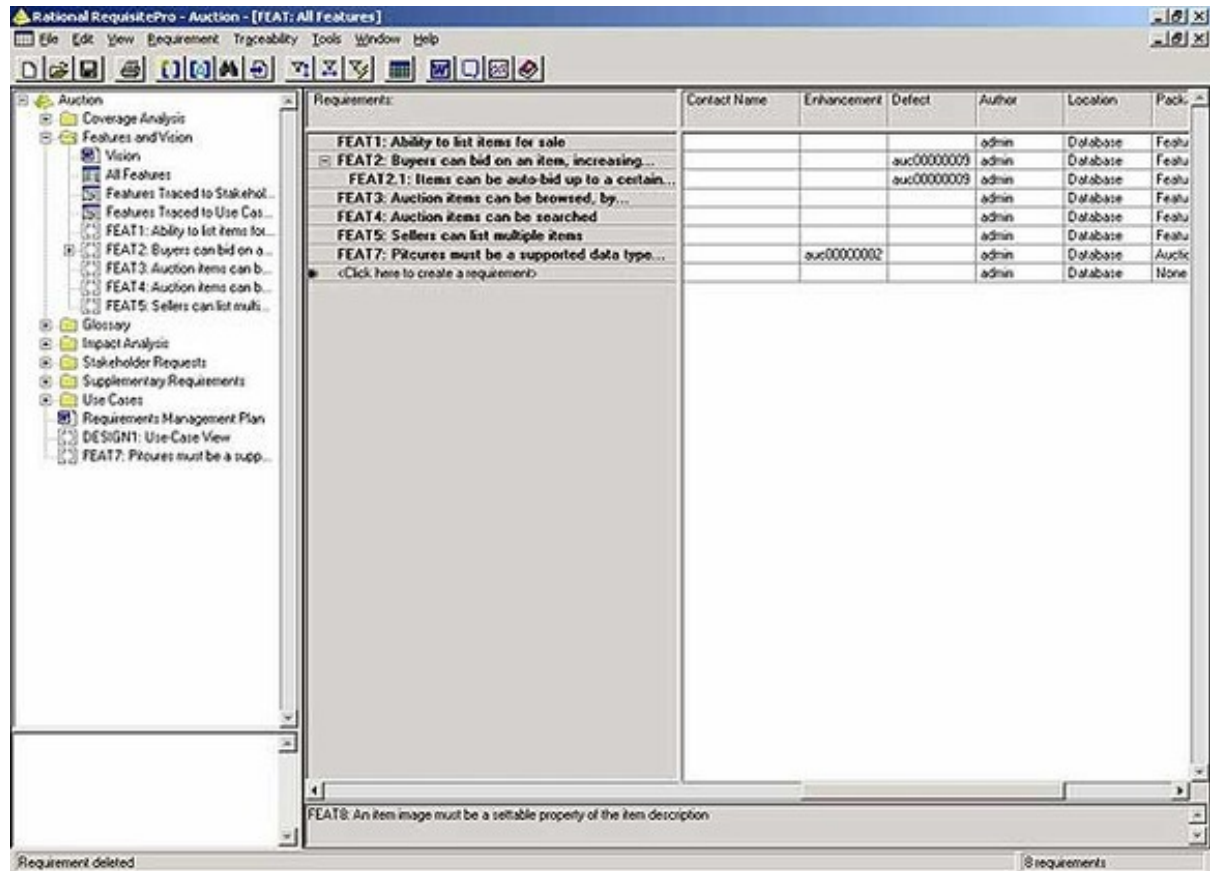


6. Set the properties of the new requirement. Take some care in writing the text of the new requirement. A typical pitfall is to simply copy the text of the enhancement request into the text of the requirement. Requirements must be testable, unambiguous, and compliant to the team glossary. You run the risk of adding ambiguity to your requirement set if you don't review the completeness and potential conflicts in the enhancement request text.
7. Select a parent requirement if relevant.
8. Click **OK**

- Click **Apply** to save the changes and close this window.

Once again, you can see the enhancement request Rational ClearQuest tag added as a requirement attribute for the newly created requirement, shown in Figure 41.

Figure 41. Viewing Rational RequisitePro



To list the new requirement in a Rational RequisitePro requirement document (which is likely how your requirement specification is created):

- Right-click on the new requirement in the view and select **Cut**.
- Open the Word document, position your cursor where you want to see that new feature in the document.
- Select **RequisitePro > Requirement > Paste** (be careful not to use the Word paste function, but rather the Rational RequisitePro paste). Cut does not really cut the requirement from the database, but rather allows you to display a requirement in a Word document.

Once a new feature is added to the original requirement spec, there are two final steps: Creating detailed requirements (use cases) to ensure the new feature gets implemented and updating the model and design to reflect the new use cases.

Integrating changes in the detailed requirements

Creating detailed requirements to implement the new feature can involve updating an existing use case or creating a new one in Rational RequisitePro. Creating use case requirements is similar to the steps you've already covered in Part 1 to create any requirement. Once you modified/created the use cases, establish a traceability link between the newly added feature and any use cases it affects, so that if the feature later changes, you can quickly pinpoint which use cases are affected.

When the original enhancement request was associated to a UC requirement (instead of a FEAT requirement), the association created in Rational ClearQuest between the CR and the UC requirement is visible in a simple view that lists the use cases that have an associated defect or enhancement request. From that view, you can access the detail of the enhancement request that is impacting the use case, to retrofit the accepted request into the use case specification document in Rational RequisitePro.

Modifying the application

If no new use cases have been created, then the design model in XDE does not need to be updated, as the Rational XDE-Rational RequisitePro integration dynamically links the use case model in XDE to the latest state of the use case specification in Rational RequisitePro. However designers should be alerted of the use case specification changes by the project manager to avoid working off an obsolete printed copy.

If a new use case was created (as a result of incorporating an enhancement request), then the new use case should be added to the XDE use case model. To update the model:

1. Open Rational XDE and open the models associated with the Rational RequisitePro project.
2. Create the new use case in the XDE model explorer or the use case diagram.
3. Right-click the use case and select to link the use case to a Use Case Specification. The XDE linkage between use cases and implementation classes should give you the information you need to update the class,

model, and ultimately the code.

Since these are stages that you have already covered in this series, consider these as exercises.

Section 6. Wrap up

Summary

In this tutorial you've seen how you can manage application changes by centralizing the collection of all change requests into Rational ClearQuest, to assess all changes in a consistent format and evaluate which changes your team has resources to implement. The integration between Rational ClearQuest and Rational RequisitePro protects requirements from unaccepted changes, provides visibility into the origin of requirement changes and allows you to mechanically update your requirements specification with the latest changes.

Rational ClearQuest in this case is merely a funnel and sorting tool, collecting the requests and making them ready for assessment. IBM Rational RequisitePro is still the driving force behind the development progress, since it holds the definition of requirements for the project. IBM Rational ClearQuest and the integration available are merely a way of further enhancing and refining the requirements as the development of the application progresses with the long term goal to deliver software that stays inline with the ever evolving stakeholder needs.

As you look forward to Part 4, you'll be concentrate on managing and tracking the changes themselves, rather than the requests that might have been triggered. These changes are recorded in IBM Rational ClearCase, which integrates with the other tools by providing a method for tracing changes in code, requirements, requests and other entities during the life of the project. In Part 5, you'll look at the role of testing packages in the development of an application.

Resources

Learn

- - Part 1 of this series: [Translate requirements into an application model](#).
 - Part 2 of this series: [Integrating XDE Professional into WebSphere](#).
 - Part 4 of this series: [Tracking changes during the application lifecycle](#)
- 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](#) can help you get started with Rational products.

Get products and technologies

- Download a [trial copy](#) of IBM Rational 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 this Web site at <http://www.mcslp.com>.