

# Improved application development: Part 3, Incorporating changes in requirements

Skill Level: Intermediate

[Martin C. Brown \(questions@mcslp.com\)](mailto:questions@mcslp.com)  
Freelance writer and consultant  
MCslp

28 Jun 2005

The focus of this third tutorial in the "Improved application development" series is on change management. This tutorial shows how individual change requests are linked and traced back to the original requirements specification, how you manage that information from within your development environment, and how you generate a new specification.

## Section 1. Before you start

### About this tutorial

This third part of the *Improved application development* series presents some of the issues of change management in application development. In Part 1 of this series, you looked at the initial stages of application development, when you created the requirements that will ultimately drive the project. In Part 2, you looked in more detail at the process of building the components in your application.

So far in this series, you've built the sample Auction application based on the original requirements. But what happens after you use the application and demonstrate it to the customer -- and your plans haven't quite worked out the way you expected?

In this tutorial, you will learn how you can link individual change requests to the original requirements specification for tracing as well as how you can generate a new specification to ensure that developers are building software against the latest

customer needs. Change requests include defects reported by testers, users, or the customer support help desk as well as enhancement requests from clients and other stakeholders. For instance, the client might request an enhancement to the sample Auction application because he or she really wants the seller to use multiple pictures to describe an item or perhaps because the client want buyers to have an auto-bid option to automatically keep re-bidding amounts up to a maximum. Enhancement requests tend to represent stakeholder wishes; defects, on the other hand, 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 an ideal tool for collecting all requests for changes (defects and enhancement requests); accepting or rejecting them; and organizing, assigning, and approving the changes as they are accepted, resolved, and implemented. Some change requests have an impact on the original requirements specification that is still the driving force behind the project. To manage the impact of change on requirements, Rational ClearQuest integrates with IBM Rational RequisitePro® to track the link between enhancement requests and requirements specifications.

In this tutorial, you'll learn about:

- Collecting change requests
- Integrating Rational ClearQuest and Rational RequisitePro
- Managing Rational ClearQuest data with IBM Rational Application Developer
- Building a new version of the requirements specification
- Modifying the application

## Prerequisites

The focus of this tutorial is on two products in the Rational Suite: Rational RequisitePro and Rational ClearQuest. The tutorial also covers integrating the IBM Software Development Platform (which incorporates the IBM Rational Software Modeler and Rational Application Developer components) into the process. Links to download trial copies of most of these products are below. For a trial copy of Rational ClearQuest, you must contact your local IBM office.

To complete the steps in this tutorial, you need:

- [RequisitePro Version 2003.06.13](#) or later.
- Rational ClearQuest Version 2003.06.13 or later.

- Rational ClearQuest Eclipse Client:
  - If you have Rational ClearQuest Version 2003.06.13.x, you need [Version 6.0.0x](#) of the client.
  - If you have Rational ClearQuest Version 2003.06.14.x, you must use the 6.14.x Version of the Rational ClearQuest Eclipse client. You can download this version of the client by selecting **Help > Software Updates > Find and Install** within the IBM Software Development Platform. Follow the instructions to install the client.
- [Rational Application Developer](#) Version 6.0
- [Rational Software Modeler](#) Version 6.0

You also need to install the Auction application sample from the IBM Software Development Platform Showcase samples. These samples include a pre-built version of the Auction application that you have been building in this series. By using the pre-built version, you can continue the tutorial with the remainder of the code already configured. To install the sample, perform the following steps:

- Select **Help > Samples Gallery**.
- In the Samples Gallery window, expand the Showcase samples folder.
- Expand the Auction Application folder.
- Expand the Construction folder.
- Click **Web Application**.
- Click **Import the sample** to import the code and source files into your current workspace.

Find the Enterprise JavaBean (EJB) components for the application in your EJB Projects folder as the project **AuctionV60EJB**.

---

## Section 2. Collating change requests

### Changing an application

In the previous tutorials in this series, you learned how to build a requirements specification and translate that specification into an application model. Ideally, a specification should capture all customer or client requirements. However, in most

projects, not enough time is spent identifying requirements early on, so there are often changes to be made later on in the development process. The application may not behave quite as expected; it may not provide the value anticipated by clients; there may be bugs; or the performance of the system may not up to the standard required by the stakeholders.

Managing changes to an application is a big challenge in today's marketplace. You have to be able to modify and adjust your application without affecting your existing functionality or upsetting your customers. Any tool that can help you 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 requirements specification in synch with these changes, because the requirements specification acts as a representation of what was promised to clients. It is vital that accepted changes are incorporated into the requirements specification to ensure that developers and testers build and test upon that promise. In doing so, you merge all stakeholder requests (the original ones collected during the requirements-gathering phase and those 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; 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 of this series, you concentrated on producing a requirements specification based on the stakeholder needs for an auction system for selling and buying products through a Web site. At that stage, the requirements were driven based on the original needs and requests. Now that the first stage of the development process has been completed, you need to incorporate changes.

Change requests come from several different sources:

- **External stakeholders.** This is the most obvious source. Stakeholders can often request changes to an application; these are generally requests for enhancements to the original requirements. 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, a request for enhancement to the original auction system might be one for support of multiple images for a given auction item. A defect report can also be reported by stakeholders based on their understanding of the application's expected operation if the defect is not a clear-cut bug but rather relates to the operation or implementation of an application.

- **Competitive comparison.** You might want to add some features to bring your software on par with some of its competitors.
- **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.
- **Development team.** Developers often identify potential bugs in the system and possible avenues for enhancement purely because they are closer to the implementation than any other members of the team. The use of testing tools such as IBM Rational PurifyPlus® and IBM TestManager is covered in Part 5 of this series.

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

## Collecting requests

Now you know how to collect requests 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 them is pointless unless you can identify their origin; it may well be that a fix for one defect will solve or isolate another defect in the system.

Without some system in place, it can also be difficult to monitor enhancement requests and prevent them from altering the priority and progress of existing development. I've seen projects without any kind of change management system that spent a great deal of time continually adding and implementing enhancements before the developers had even achieved the original goals of the project. This is frequently referred to as "feature creep" or "gold plating".

By managing, allocating, and prioritizing all of these requests, you can ensure that your development team is devoting their energy to the right problems, not wasting it on pointless features and fixes that may never have been part of the original goals.

To manage change requests to your sample application, in this tutorial 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 record type. Although Rational ClearQuest uses some predefined types, such as enhancement requests and defects, you can create a record type to hold pretty much any type of change request you require.

After 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 details used to store and manage the Rational ClearQuest information.

Rational ClearQuest supports several database schemas that you can use out of the box; alternatively, through Rational ClearQuest Designer (included in Rational ClearQuest), you can create your own schemas and layouts. The list below describes the standard schemas that Rational ClearQuest provides. Note that others might exist in your installation based on the toolset you used to install the product (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.
- **Rational Suite DevelopmentStudio** . Contains fields and rules that work with IBM Rational Purify, Rational Visual Quantify, and IBM Rational PureCoverage. Provides code for the Rational ClearQuest and Rational RequisitePro integration.
- **Rational Suite TestStudio** . Contains fields and rules that work with Rational TeamTest, Rational RequisitePro, Rational Purify, Rational Visual Quantify, and Rational PureCoverage. Provides code for the Rational ClearQuest and Rational RequisitePro integration.
- **Rational Suite Enterprise** . Contains fields and hooks that work with all Rational products. Provides code for the Rational ClearQuest and Rational RequisitePro integration.
- **UnifiedChangeManagement** . Supports the Unified Change Management (UCM) process by integrating with Rational ClearCase. Use with Rational ClearCase.

You'll choose a schema for your project in the next panel.

If you plan to associate Rational ClearQuest change requests with Rational RequisitePro requirements, as you will in this tutorial, you must create a Rational project using the Rational Administrator. The next panel shows you how.

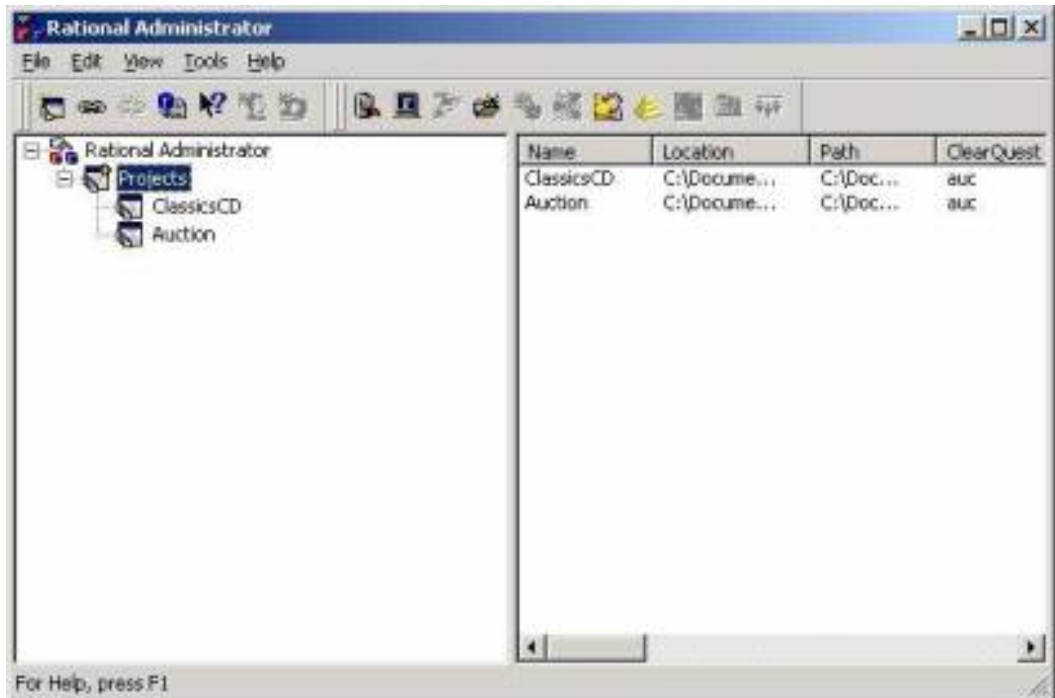
## Create a new Rational project

In Rational Administrator, a Rational project associates a Rational ClearQuest database, a Rational RequisitePro project, an IBM Rational Rose model, and a TestManager project, all under Rational ClearCase change management control.

To create a new Rational project in Rational Administrator, perform the following steps:

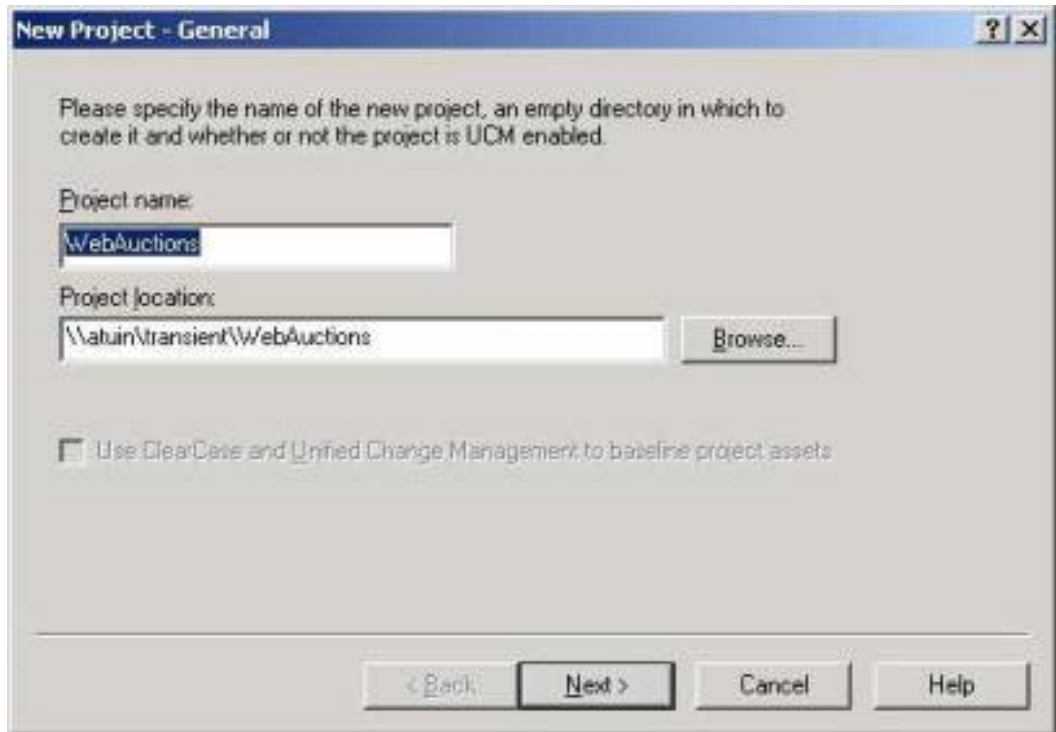
1. Open Rational Administrator. A window like the one shown in Figure 1 opens.

### **Figure 1. Rational Administrator**



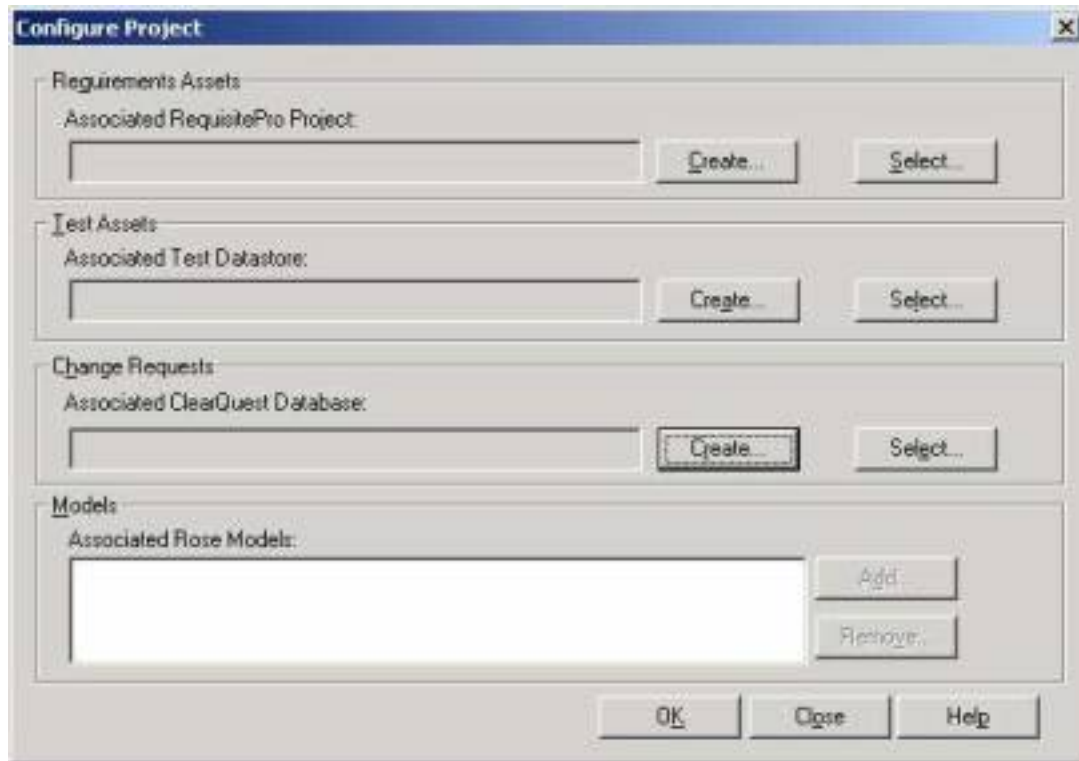
2. Right-click **Projects** , then select **New Project** .
3. Name the project and define a location, as shown in Figure 2. The location should be an empty directory. In a team environment, make sure that this directory is on a shared location on the network and is available through a UNC share (for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

**Figure 2. Name the project**



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

**Figure 3. Finishing the creation of the project**

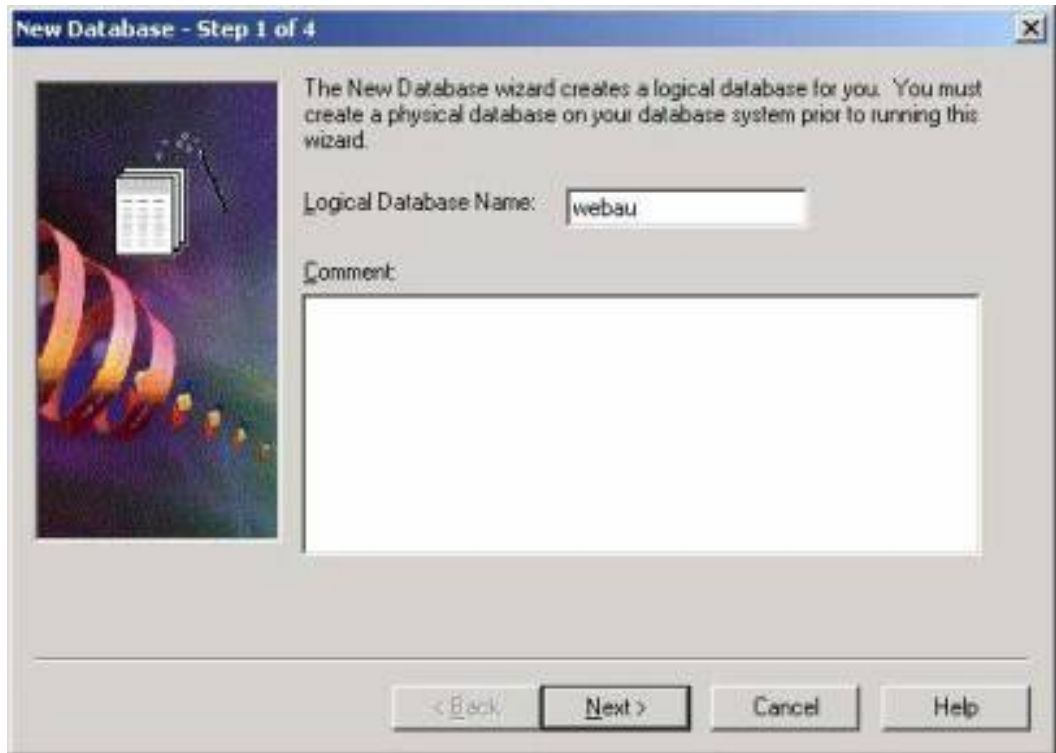


## Add a Rational ClearQuest database

In the previous panel, you created a Rational project to associate different aspects of your system. To associate a Rational ClearQuest database to store and manage change requests, perform the following steps:

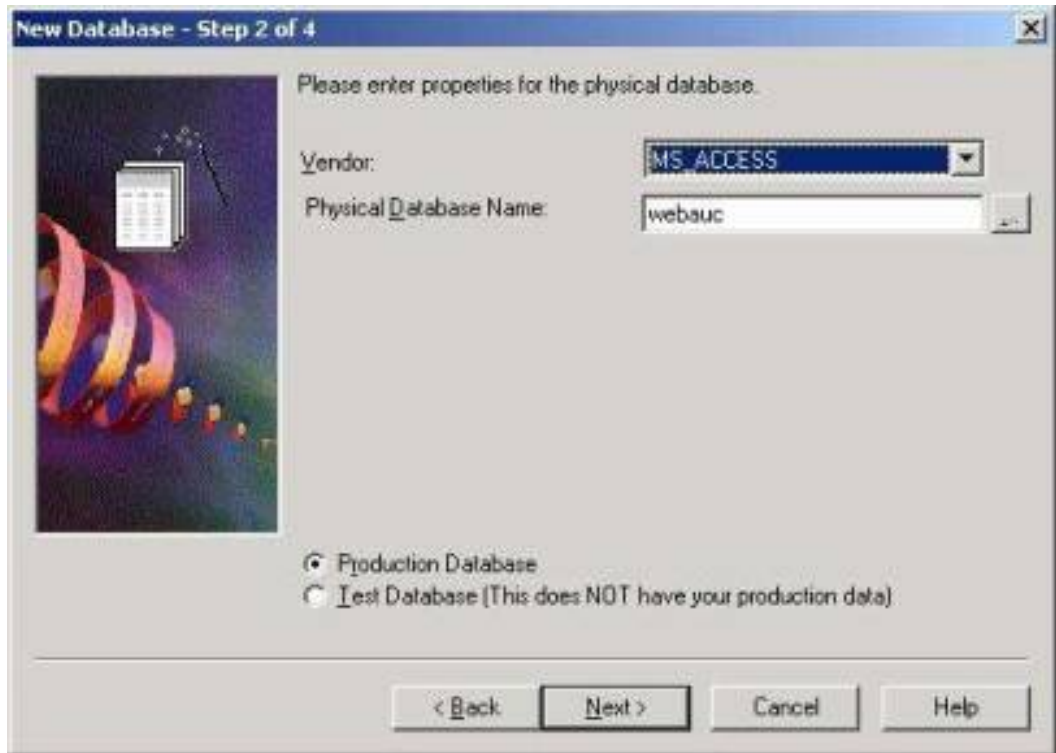
1. In the Change Requests section of the Configure Project window (illustrated in Figure 3 in the previous panel), click **Create**.
2. Choose a database connection. Several database connections are created when you installed Rational ClearQuest. These databases allow you to connect to a Rational ClearQuest datastore, but the information itself resides in a database accessed through the database connection.
3. Name the database `webau`. If you'd like, you can add a comment to describe the database content, as illustrated in Figure 4.

### Figure 4. Name the database

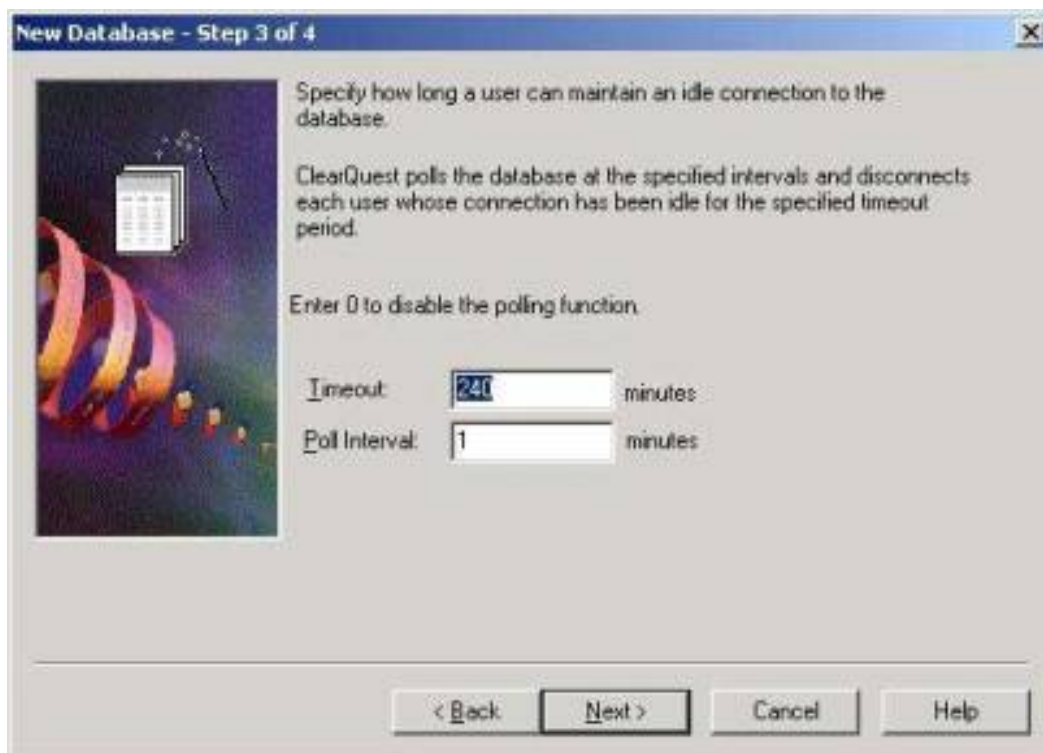


4. Choose the name of the database in which you want to store the Rational ClearQuest information for this project, as shown in Figure 5. Your choice depends on the database system you are using (for example, Microsoft Access, SQL Anywhere, Microsoft SQL Server).

**Figure 5. Choose your database**

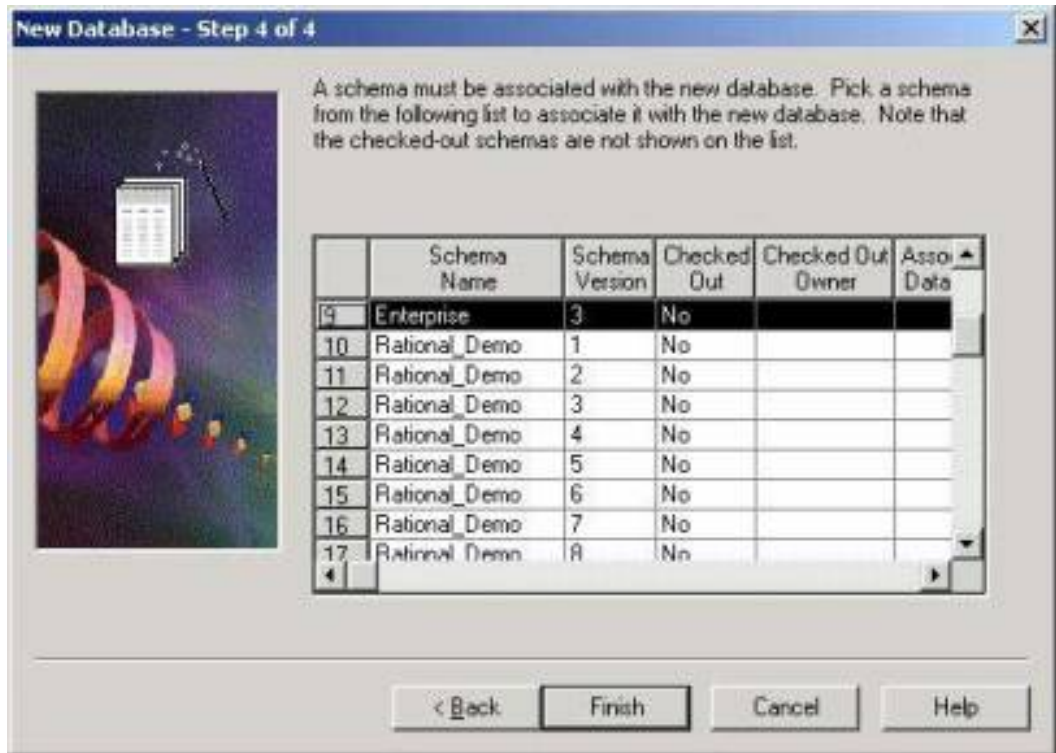


5. Specify the timeout and poll interval values. The default values, shown in Figure 6, are reasonable, so there is generally no need to change them.  
**Figure 6. Specify the timeouts**



6. Select the Rational TestStudio database schema, which Figure 7 shows. This schema 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. Choose a database schema**



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

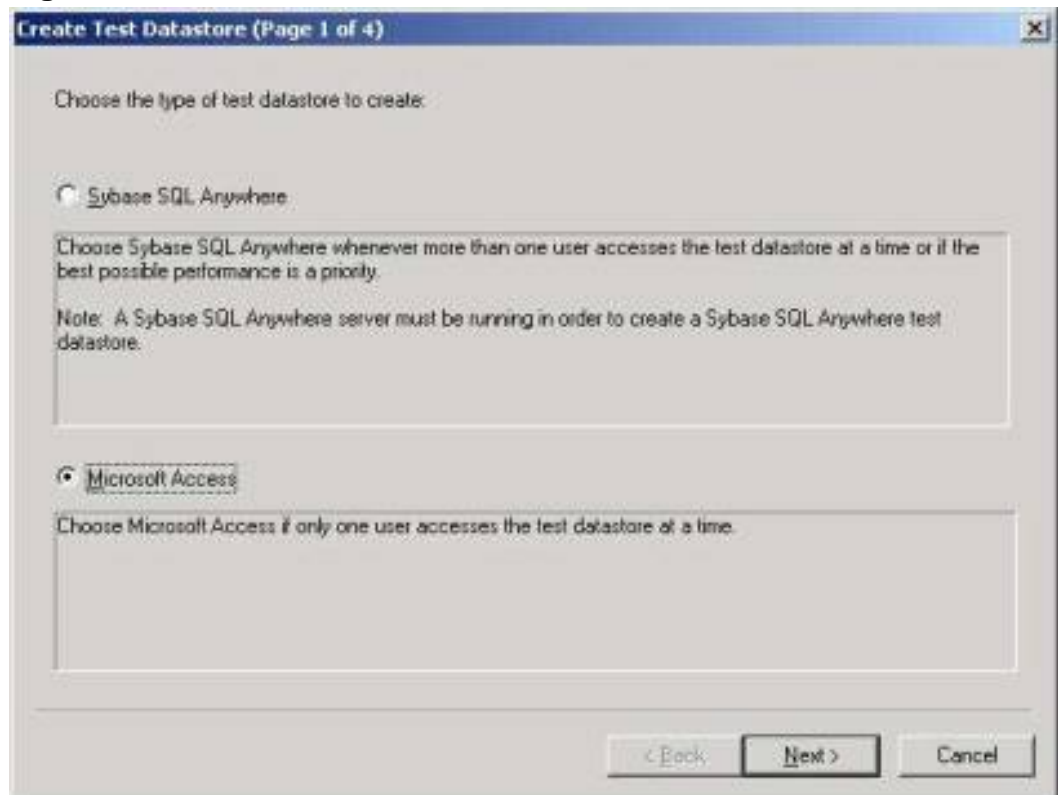
To have defects automatically created when test scripts fail, you need to create a test datastore for the project. The next panel shows you how.

## Create a test datastore for your project

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

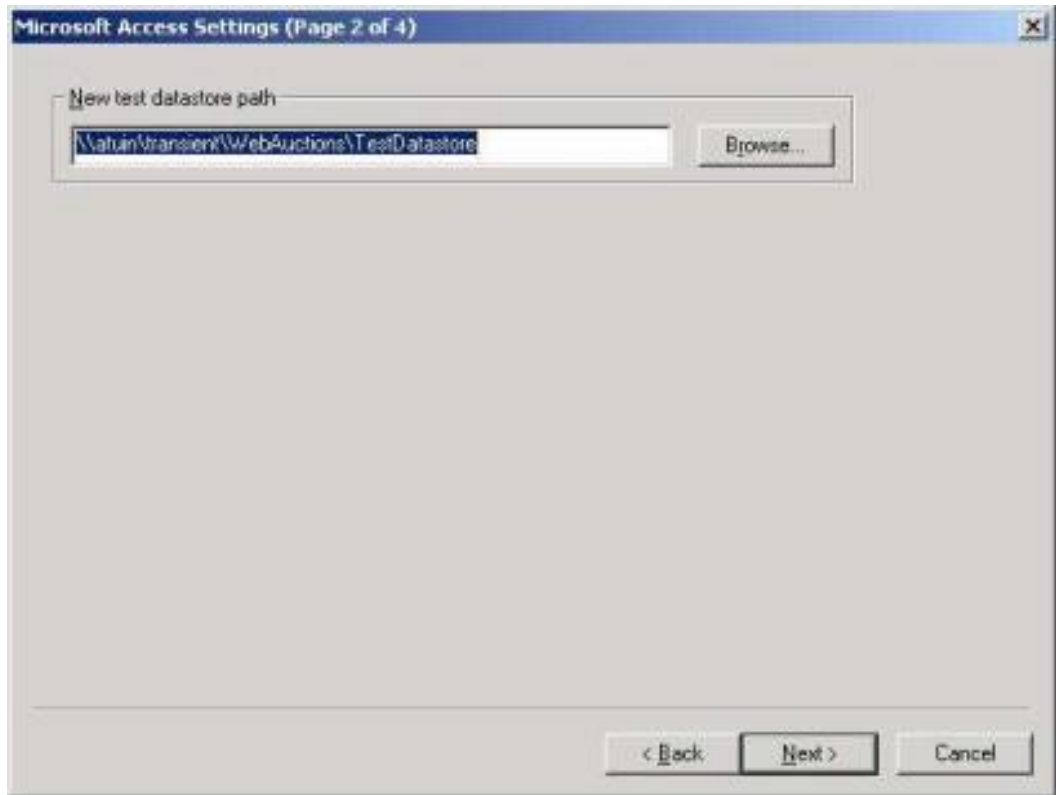
To create a new test datastore, perform the following steps:

1. From the Configure Project window, click **Create** in the Test Assets section.
2. Choose the database type to be used for the test results and tracking information, 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.

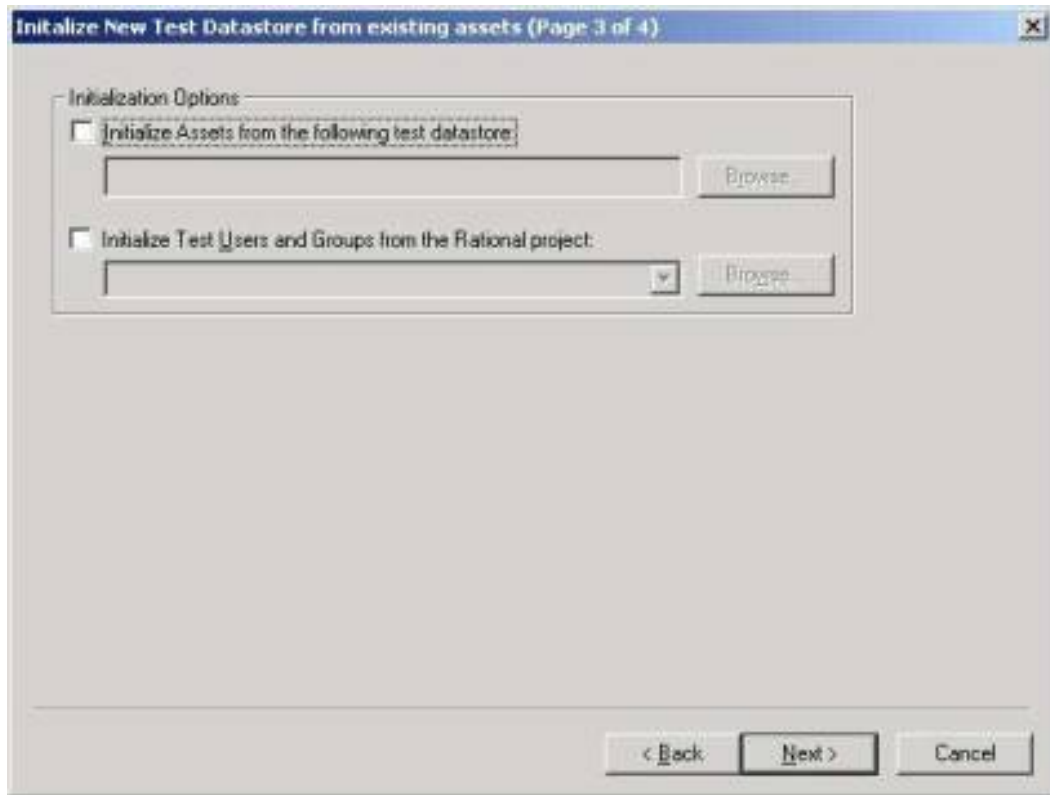
**Figure 8. Create the test datastore**

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

**Figure 9. Choose a location for the datastore database**



4. Choose an existing datastore to initialize this test datastore, as shown in Figure 10. Doing so can be useful if you have already created a test datastore before using the integration system. You can also choose to initialize the list of users and groups from an existing Rational project.  
**Figure 10. Choose 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 created and introduced into the system outside of Rational ClearQuest (from TestManager, Rational PurifyPlus, or the IBM Software Development Platform, for example), other change requests will be 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 you can use to specify very precisely the information that the analyst requires during data entry.

You can submit information into Rational ClearQuest either through the standard Microsoft 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, select **Actions > New**. A window similar to the one shown 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 covered in more 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. Your team can decide which fields should be mandatory and which ones are optional.

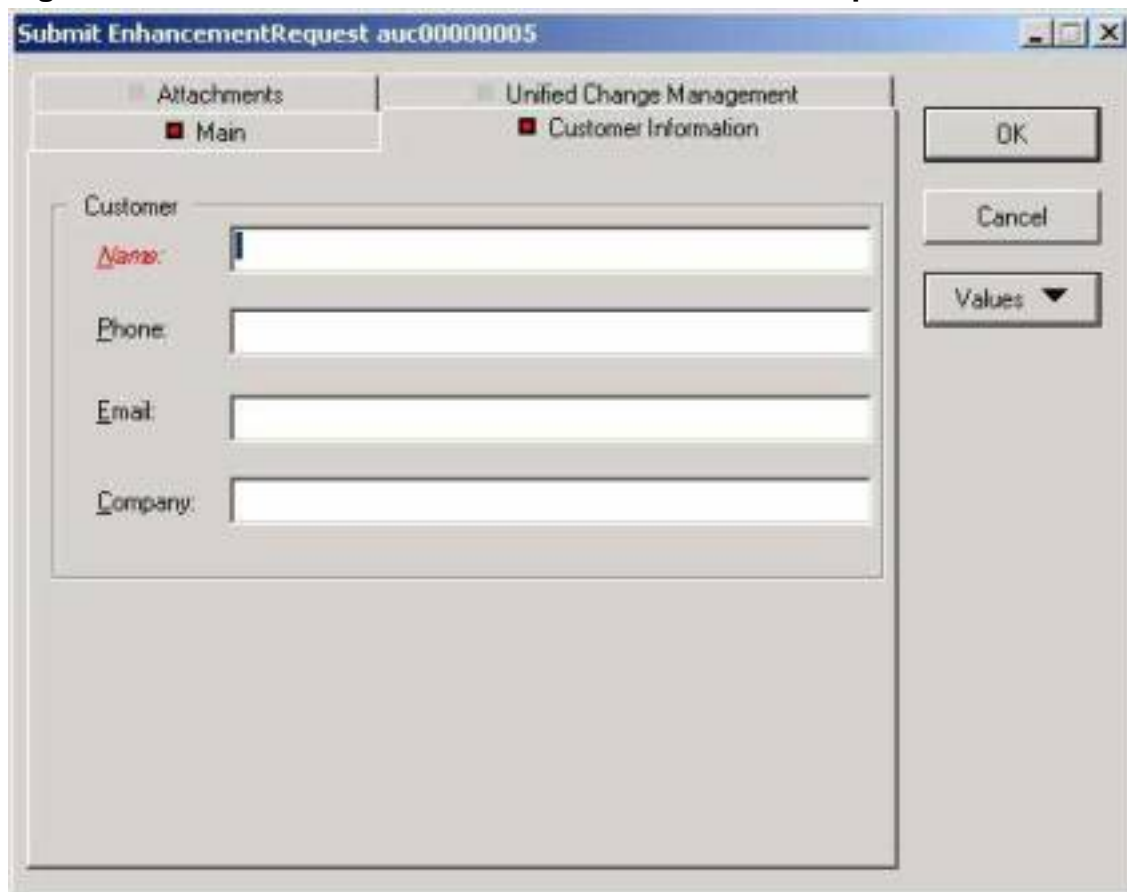
**Figure 11. A blank enhancement request**

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 (for example, customer priority, description), making it easier to evaluate and compare them than if they were left in your developers' voice mail,

presented in conversations in the hallway, or written on sticky notes or napkins.

The Customer Information tab shown in Figure 12 is used to record information about the customer who requested the enhancement. This information is crucial for cases in which the description is unclear or confusing. By having a contact, the analyst can validate his or her understanding of the request. For this tutorial, you can enter the information by hand; but keep in mind that you can also set up schemas to use a predefined list of customers and a GUI form that provides the ability to use only that predefined customer information.

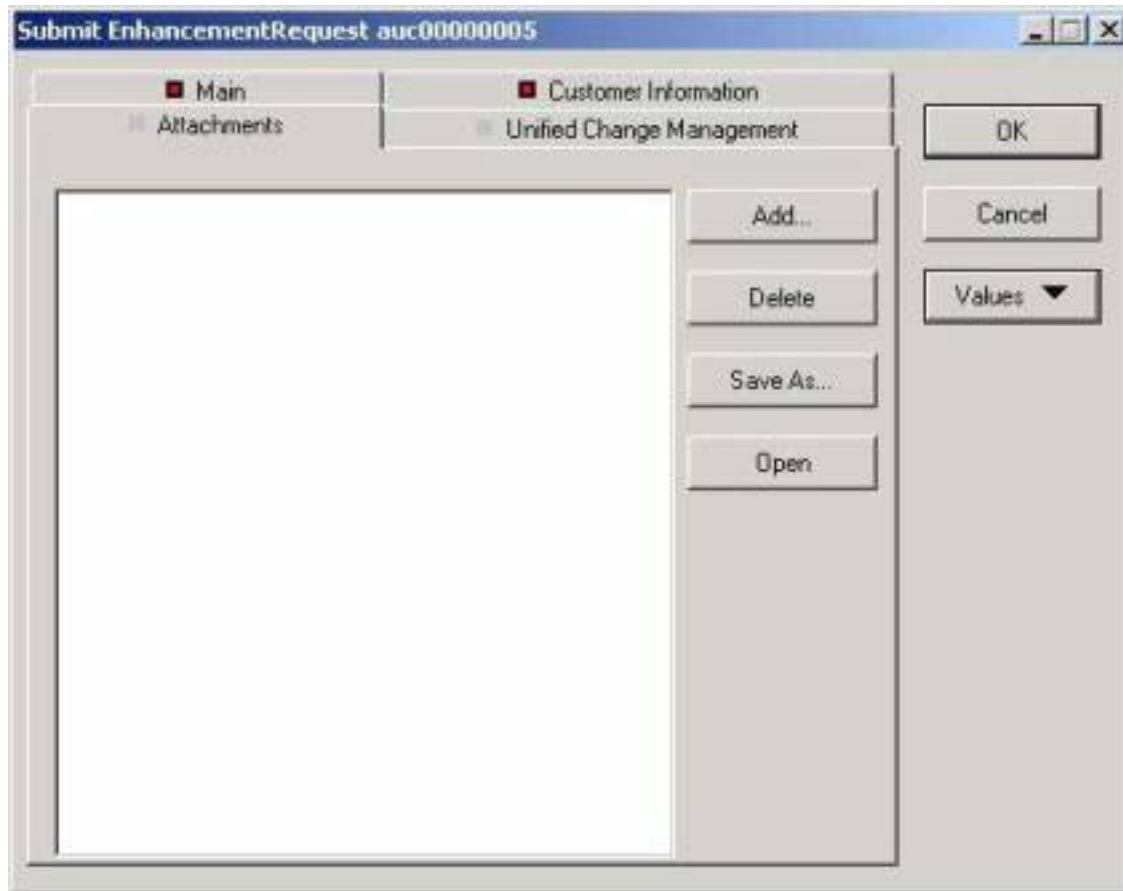
**Figure 12. Customer information in an enhancement request**



The screenshot shows a dialog box titled "Submit EnhancementRequest auc00000005". It has two tabs: "Main" (selected) and "Customer Information". The "Customer Information" tab is active, showing a form with four text input fields labeled "Name", "Phone", "Email", and "Company". To the right of the form 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 are completely flexible. You can use Rational ClearQuest to track and trace any information you need to help you or the change analyst 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. Rational ClearQuest also allows you to use the same form and database and provides a field to mark the change request as either a 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

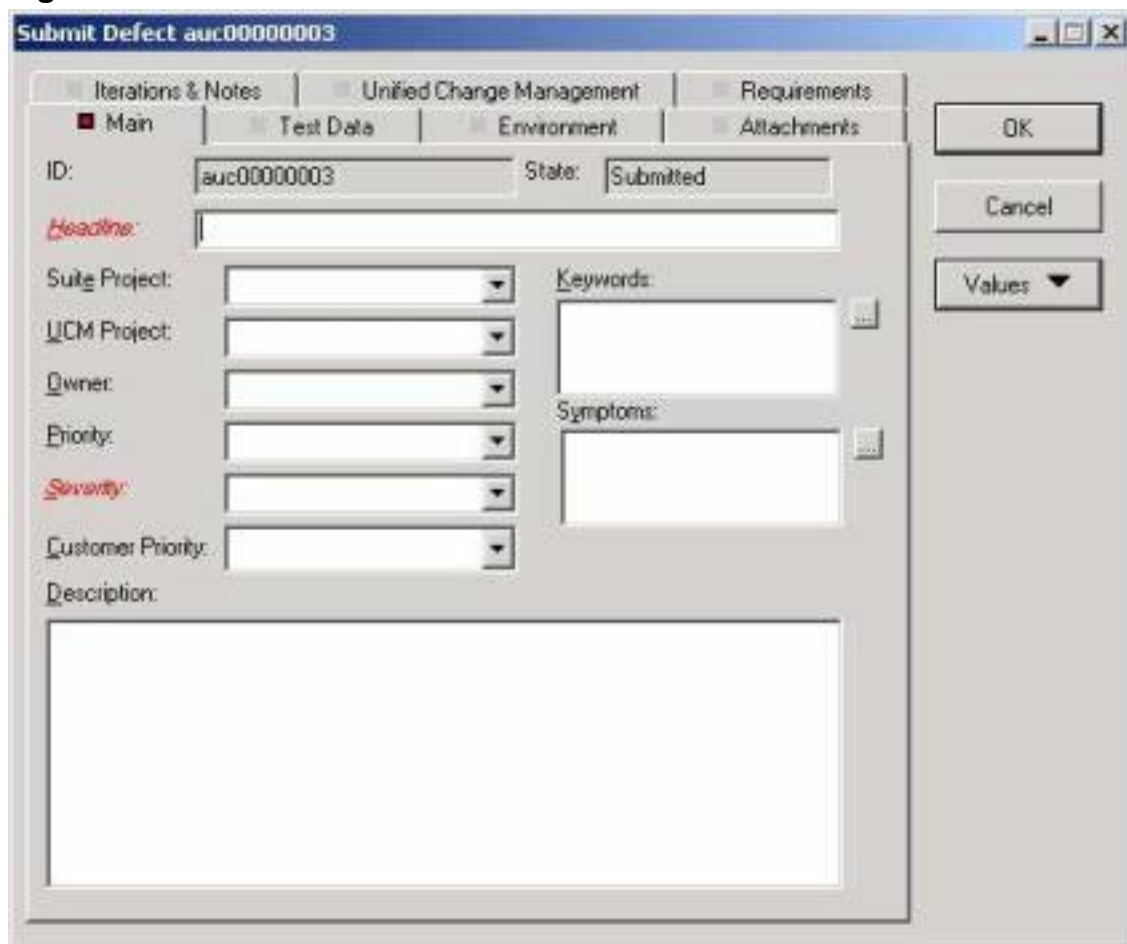
that generated that feature in the application. The defect can also come up during testing, in which case you will want to log the test information.

Furthermore, because defects will be specific to a particular iteration of an application's 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 you use an outside tool such as TestManager or Rational PurifyPlus (covered in Part 5 of this series), a lot of this information is automatically recorded and entered into the system for you, thanks to the integration between these tools and Rational ClearQuest.

To create a new defect, select **Actions > New** . The Submit Defect window is shown in Figure 14.

**Figure 14. Submit a defect**

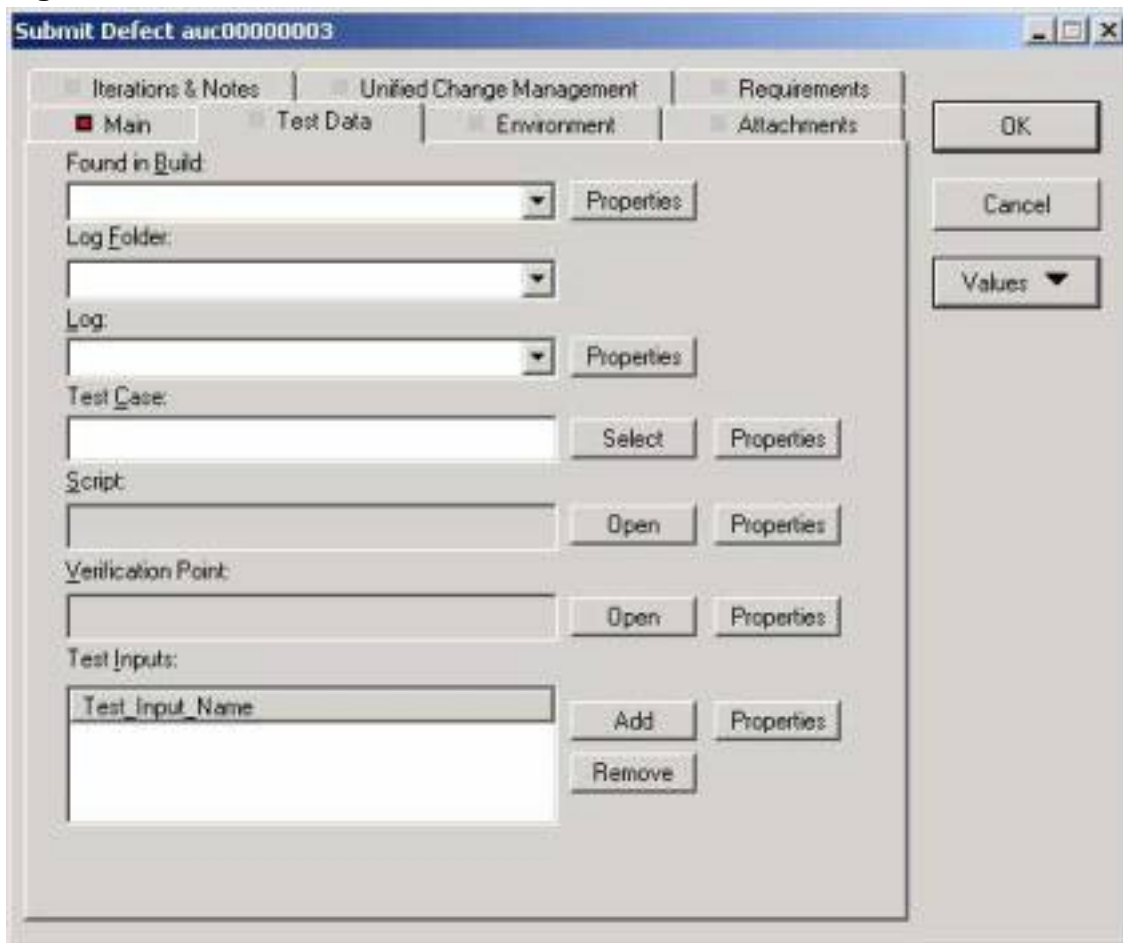


As you can see, you can enter more details for a defect than for an enhancement request, but much of the information, in this schema at least, is optional. As with the enhancement request, there are only two vital pieces of information: in this case, the

headline description and the severity. The other pieces of information -- keywords, symptoms, and the associated projects and ownership of the defect -- are optional items. If you take 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, illustrated in Figure 15, automatically lists the location of the test, the build of the application, and the location of the test logs, test case, and input data that led to the defect. That is the key value of Rational ClearQuest -- TestManager integration. No manual intervention is required to propagate the test information into the defect log.

**Figure 15. The Test Data tab**



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

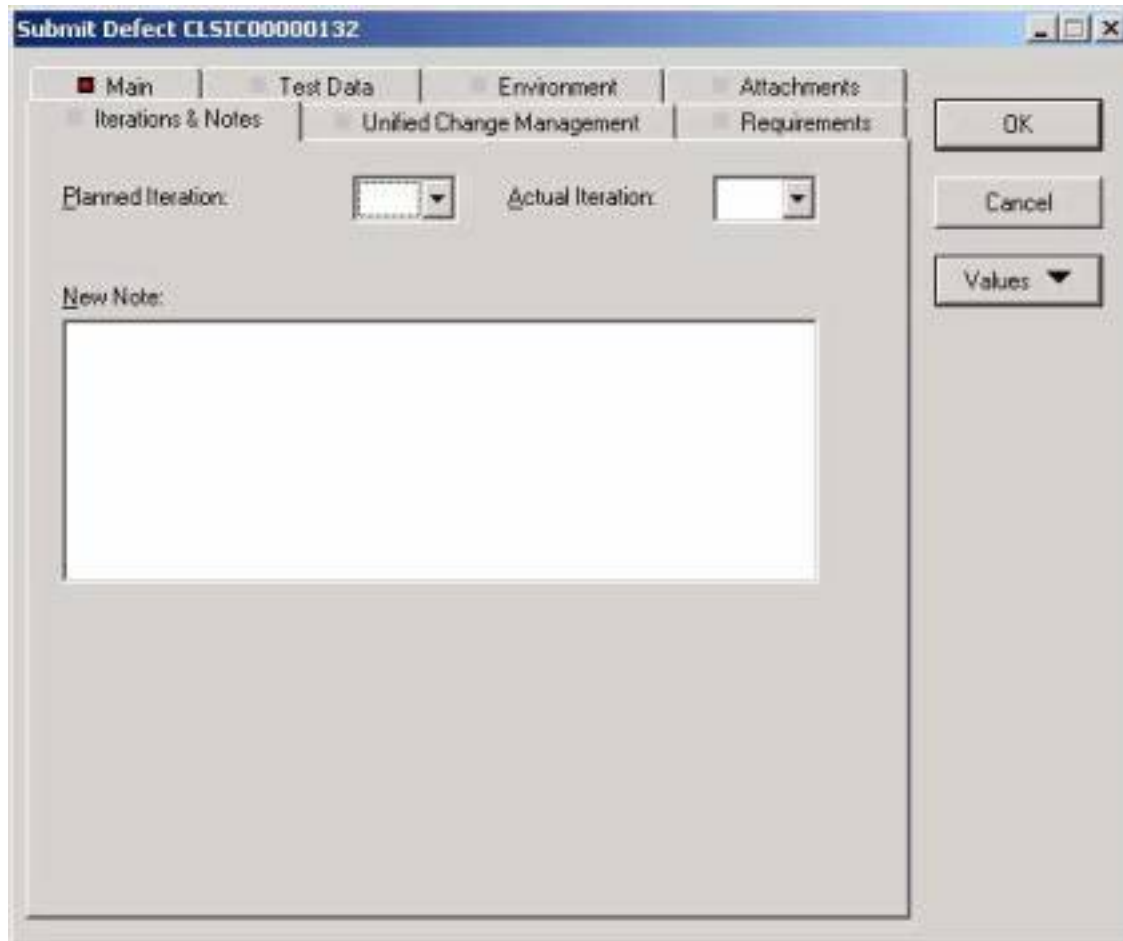
**Figure 16. The Environment tab**

The screenshot shows a 'Submit Defect' dialog box with the following structure:

- Title Bar:** Submit Defect auc00000003
- Tabs:** Iterations & Notes, Unified Change Management, Requirements, Main (selected), Test Data, Environment, Attachments.
- Reported By:** Contact (text field), Company (text field).
- Environment:** Hardware (dropdown), Computer (dropdown), Operating System (dropdown), Other Environment (text field).
- Other Fields:** Custom1 (dropdown), Custom2 (dropdown), Custom3 (text field).
- Buttons:** OK, Cancel, Values (dropdown).

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

### Figure 17. Set iterations data



All change requests, both enhancement requests and defects, along with detailed information about them, are entered into the database either manually or automatically from TestManager. You can now start organizing, prioritizing, and reporting on the information.

---

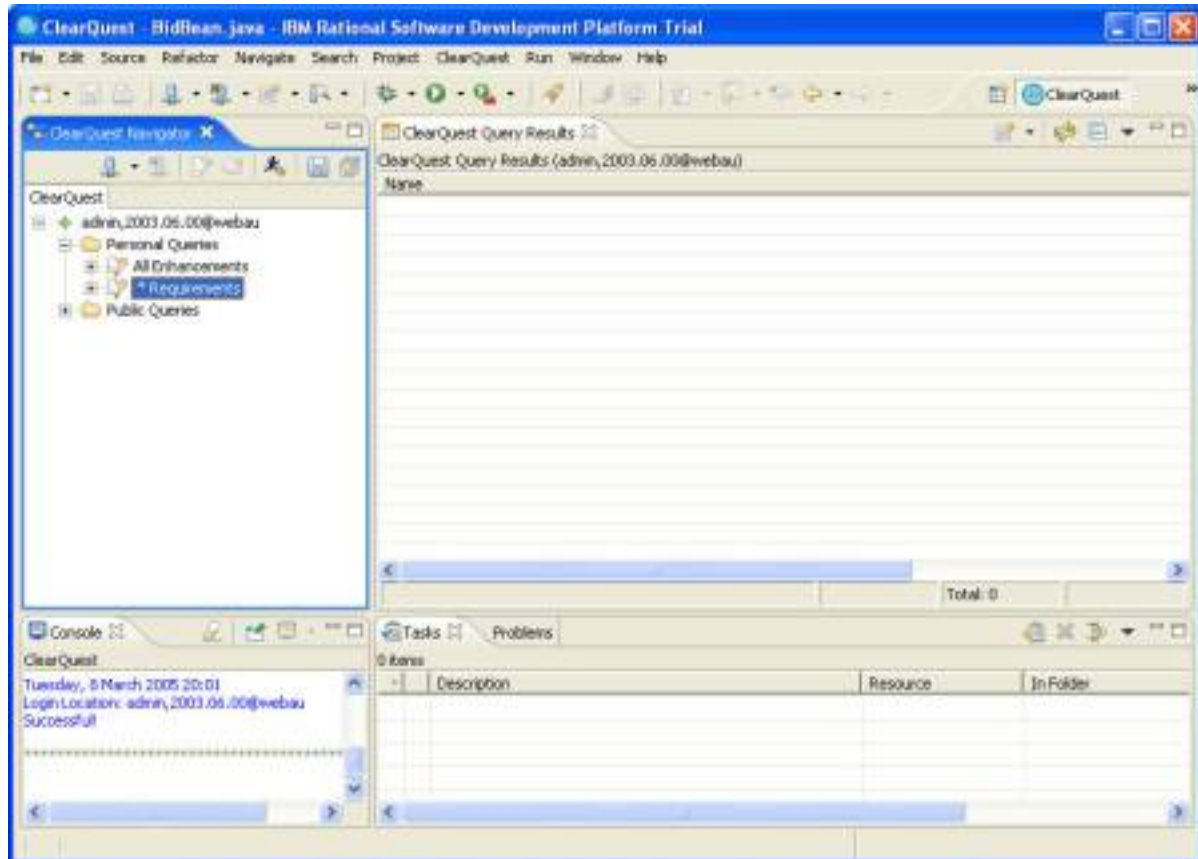
## Section 4. Using Rational ClearQuest data within Rational Application Developer

### Open a Rational ClearQuest repository

The Rational ClearQuest Client for Eclipse includes a set of additional views and a new perspective that you can use to access and interact with Rational ClearQuest

data. You can see a sample of the perspective in Figure 18.

**Figure 18. The Rational ClearQuest perspective**



You can also individually select any one of several views that are built into the perspective; doing so is generally the most useful way for you to interact with the Rational ClearQuest database while working with the code or beans or during execution and testing.

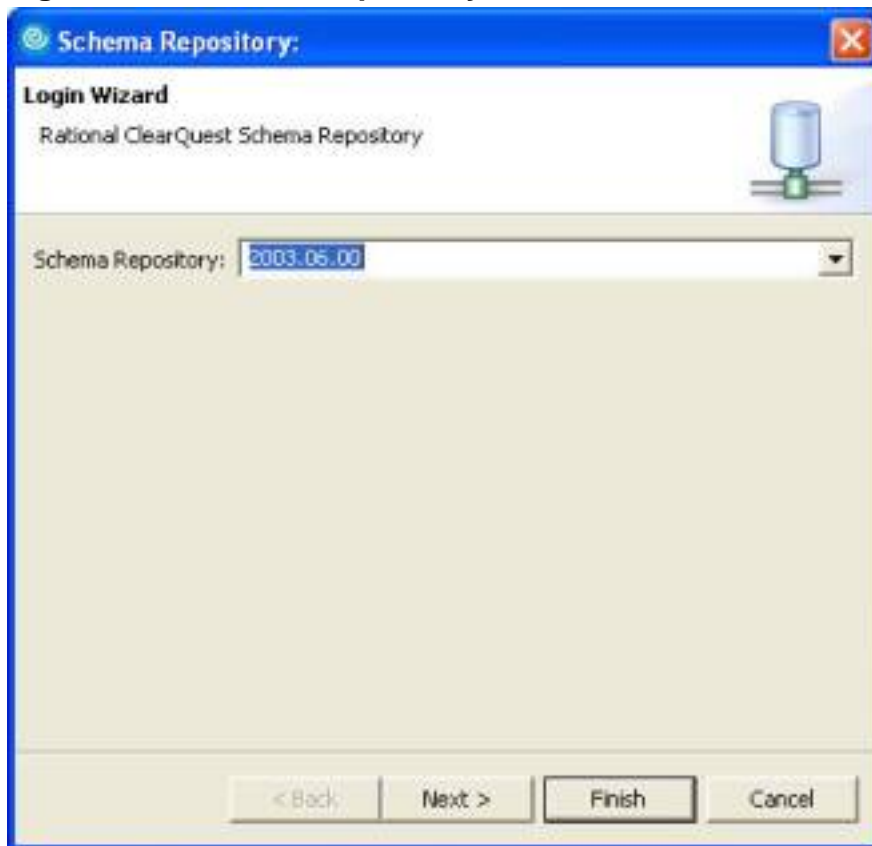
You can see the main views available by looking at Figure 18. The main interface to Rational ClearQuest is through the ClearQuest Navigator, shown here on the left. This view provides the same navigation structure available within Rational ClearQuest; you'll use this view to build queries and views to your Rational ClearQuest data.

The ClearQuest Query Results view shows the results of running these queries. The actual structure of the information in this view is entirely dependent on the queries that you create, although it will be based on a tabular structure showing the fields you've selected for display. Other views show the console (used for error messages) and the properties of individual Rational ClearQuest entries.

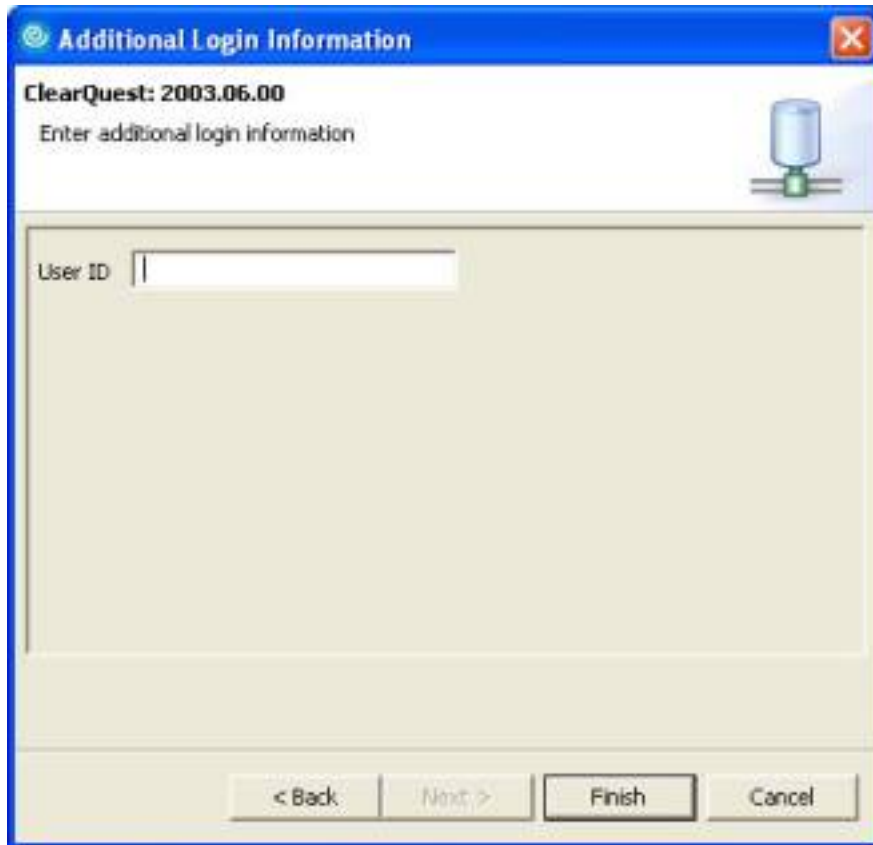
To connect to a Rational ClearQuest repository, open the ClearQuest Navigator

view, then perform the following steps:

1. Click the database connection icon in the toolbar of the ClearQuest Navigator view.
2. Select the repository you are using, as shown in Figure 19.  
**Figure 19. Select the repository**



3. Click **Next** , then enter the user ID for this repository, as shown in Figure 20. If you are using the sample database provided with Rational ClearQuest, enter `admin` as the user ID.  
**Figure 20. Set login details**



4. Enter the login details for the repository and for the database within the Rational ClearQuest repository that you can connect to. You can see the webau database selected in Figure 21.

**Figure 21. Select the Rational ClearQuest database**



After you've completed these steps, your ClearQuest Navigator view should show

the open repository.

The next panel shows how you can update the information in your Rational ClearQuest database from within Rational Application Developer.

## Create Rational ClearQuest entries

Rational Application Developer is the hub of the development process; therefore, it makes sense for you to be able to create defect reports and enhancement requests directly from within Rational Application Developer itself. Indeed, the very ability to view and create defects and enhancement requests is one of the key levels of integration.

Using the Rational ClearQuest interface, a developer can report or raise an issue, then manage and deal with that issue in the Rational ClearQuest database without ever actually leaving the development environment. This makes the process easier and therefore makes it more likely that these issues, and their resolution, will be reported and tracked through the iterations of development.

You can create new enhancement requests and defects by clicking the **New Record** icon in the toolbar for the ClearQuest Query Results view. If you click and hold the icon, you can select whether to create a new defect or an enhancement request.

In both cases, you must enter a minimum amount of information into certain fields to save the record into the database, just as you did when you used Rational ClearQuest itself. You can see a new defect request in Figure 22. The fields highlighted in red must be completed.

### Figure 22. Enter a new defect in Rational Application Developer

The screenshot shows a 'New Defect' dialog box with the following fields and values:

- ID: webau00000044
- State: Submitted
- \*Headline: (empty, highlighted in red)
- RA Project: Web Auction
- Priority: (empty)
- \*Severity: (empty, highlighted in red)
- Owner: (empty)
- Keywords: (empty)
- Symptoms: (empty)
- Description: (empty text area)

Buttons: OK, Cancel

Figure 23 shows a new enhancement request in Rational Application Developer. Again, you must complete the fields highlighted in red.

**Figure 23. A new enhancement request in Rational Application Developer**



The screenshot shows a Windows-style dialog box titled "New EnhancementRequest ... (admin,2003.06.00@webau)". The dialog has four tabs: "Main", "Customer Information", "Attachments", and "Requirements". The "Main" tab is selected. The form contains the following fields:

- ID: webau00000045
- State: Submitted
- \*Headline: (empty text box)
- \*Customer Priority: (dropdown menu)
- Description: (large text area)

At the bottom right, there are "OK" and "Cancel" buttons.

## Building queries

When the Rational ClearQuest database has been populated with defects and enhancement requests, this information must be presented to your developers in such a way that it helps them fix and address the problems within the code.

Now that you have some data in the database, you can get detailed reports. Rational ClearQuest provides customizable requests for information from the database along with in-depth reporting and statistical information.

The basic structure of the query system is actually 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, then 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 complexity has benefits and

pitfalls. The benefits are the sheer flexibility you have for reporting and structuring the information. The main drawback is that the quantity of choice can sometimes be difficult to work with when you need that quick report. To address this issue, Rational ClearQuest allows you to create and save queries in the workspace. After you've defined the fields to display and the filter parameters, you can execute that query to report on current data at any time. In addition, predefined queries are provided.

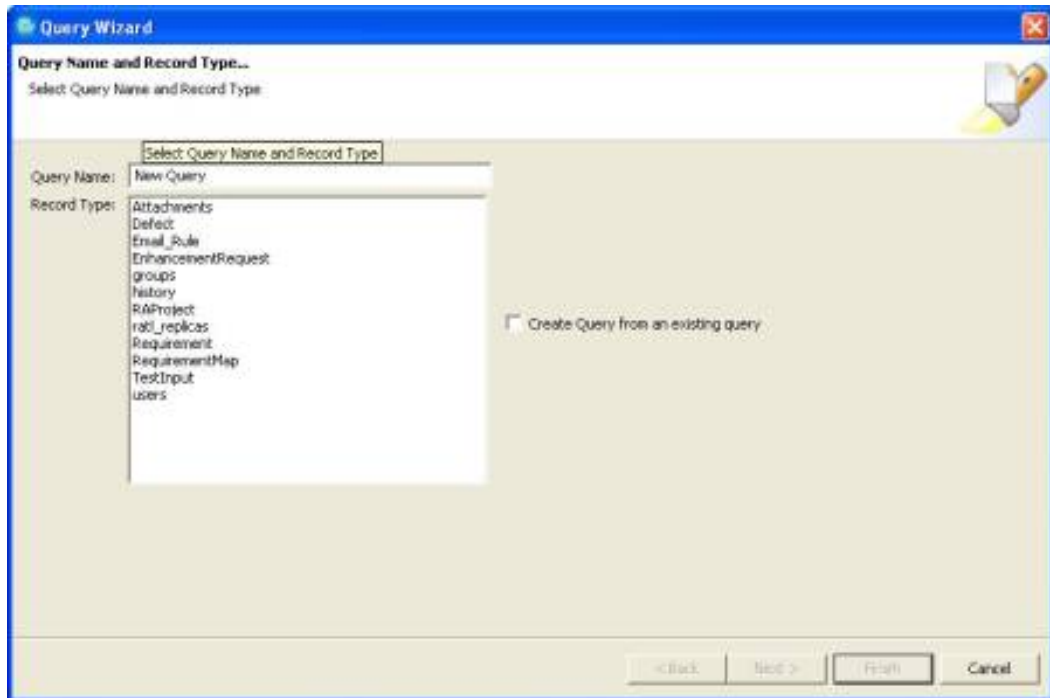
In the next panel, you'll create a simple query that shows a summary of the requests in the system, just to demonstrate the basics of creating a query. Later, you'll use this to add detailed information about the requests into Rational RequisitePro.

## Create a simple query

You can create a query either within ClearQuest or through Rational Application Developer, the basic structure and process is essentially identical. To create a public or personal query to your Rational ClearQuest database from within Rational Application Developer, open the ClearQuest Navigator view, then perform the following steps:

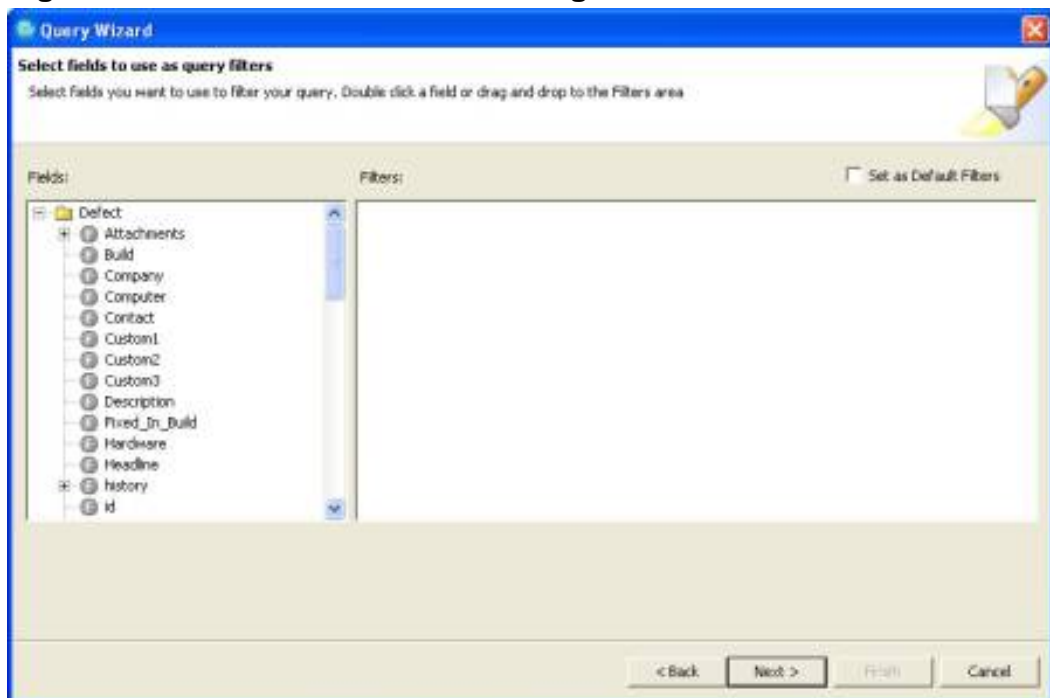
1. Right-click a folder within the ClearQuest Navigator tree, then select **New Query**. Alternatively, click **New Record** in the ClearQuest Query Results view, then select **New Query**. *Personal queries* are specific to the current user. *Public queries*, if you have the permission to create them, are available to all users.
2. Choose a record type to query against, as shown in Figure 24. You can also supply the name of the new query that will be used to identify it within the Navigator view. In this example, choose to query against defects in the database, and name the query `All Defects`. (Note that you can also choose to create a query based on an existing query definition, which can be useful if you want to duplicate a query and change the filter. Ignore this option for the moment.)

### Figure 24. Create a new query



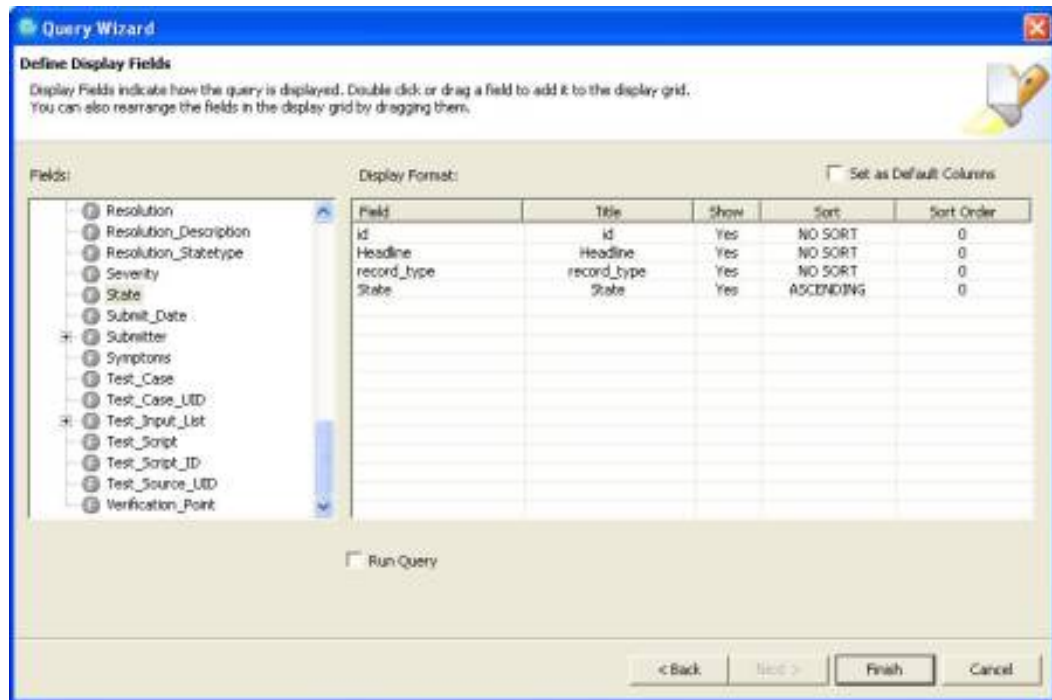
3. As you can see in Figure 25, you next have the opportunity to select the fields to use to filter your query results; you won't be filtering the results in this query, so you can ignore this screen and click **Next**. The issue of filtering is addressed later in this tutorial.

**Figure 25. Select the fields for filtering**



4. 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 the list of fields available here depends entirely on the record type you selected in this first window of the wizard.) Drag the **Headline**, **id**, **record\_type**, and **State** fields from the panel on the left to the Display Format table on the right; alternatively, you can double-click the fields to move them.
5. Click in the Sort column next to **State** to select the sort order as ascending. Because this is the only field you're going to use for sorting, you don't have to worry about the sort order, but you can use this column in the query definition to prioritize columns in the results. When you've moved everything, your screen should look like Figure 26.

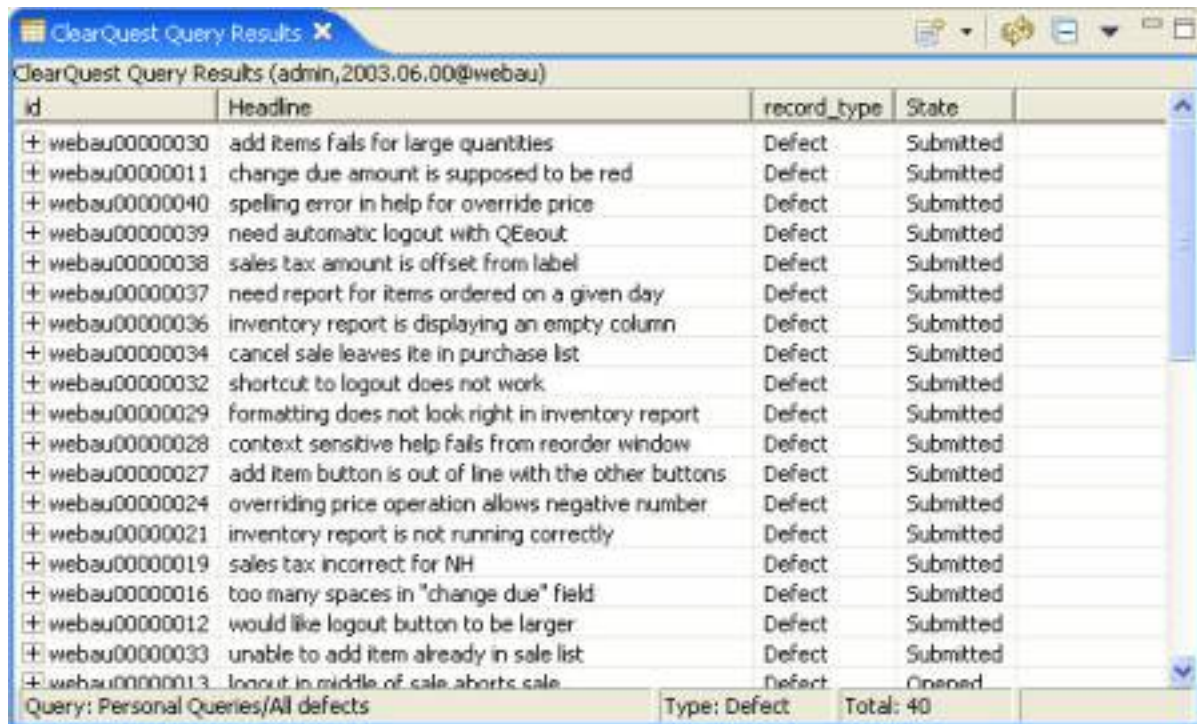
**Figure 26. Defining how the query is displayed**



6. Click **Finish** to save the query.

To actually run the query, you need to double-click the query name within the Navigator view. You can see the results from the sample database in Figure 27.

**Figure 27. The query results**



ClearQuest Query Results (admin,2003.06.00@webau)

| id              | Headline  | record_type | State     |
|-----------------|---|-------------|-----------|
| + webau00000030 | add items fails for large quantities                  | Defect      | Submitted |
| + webau00000011 | change due amount is supposed to be red               | Defect      | Submitted |
| + webau00000040 | spelling error in help for override price             | Defect      | Submitted |
| + webau00000039 | need automatic logout with QEeout                     | Defect      | Submitted |
| + webau00000038 | sales tax amount is offset from label                 | Defect      | Submitted |
| + webau00000037 | need report for items ordered on a given day          | Defect      | Submitted |
| + webau00000036 | inventory report is displaying an empty column        | Defect      | Submitted |
| + webau00000034 | cancel sale leaves ite in purchase list               | Defect      | Submitted |
| + webau00000032 | shortcut to logout does not work                      | Defect      | Submitted |
| + webau00000029 | formatting does not look right in inventory report    | Defect      | Submitted |
| + webau00000028 | context sensitive help fails from reorder window      | Defect      | Submitted |
| + webau00000027 | add item button is out of line with the other buttons | Defect      | Submitted |
| + webau00000024 | overriding price operation allows negative number     | Defect      | Submitted |
| + webau00000021 | inventory report is not running correctly             | Defect      | Submitted |
| + webau00000019 | sales tax incorrect for NH                            | Defect      | Submitted |
| + webau00000016 | too many spaces in "change due" field                 | Defect      | Submitted |
| + webau00000012 | would like logout button to be larger                 | Defect      | Submitted |
| + webau00000033 | unable to add item already in sale list               | Defect      | Submitted |
| + webau00000013 | logout in middle of sale aborts sale                  | Defect      | Opened    |

Query: Personal Queries/All defects      Type: Defect      Total: 40

The ClearQuest Query Results display is static; it doesn't automatically update if the information in the database changes (as it might if other developers are changing information, for instance). You can click **Refresh** in the toolbar to update the display or simply double-click the query in the Navigator view again.

The only problem with this query is that it is showing all defects. It would be more useful if you could create a query that displays all this information only for those defects that have been submitted into the system but have not yet been opened. In the next panel, you'll see how to create just such a 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. *Filters* do exactly what you would expect: they filter the content by selecting only the records you want. Note the distinction, though; a filter filters the contents of a query, and a query defines the fields and filter settings you want to use. This is slightly (but not entirely) different from the terminology used when building a database query.

Creating a new query based on your existing query but with the addition of a filter is quite straightforward. You can create the query in one of several ways, including following the wizard you completed in the previous section and using your previous query as the basis for a new one.

However, you can also copy queries, just as you would copy other components, and then duplicate them. You can then modify these new queries. This latter method also helps demonstrate how you can modify existing queries.

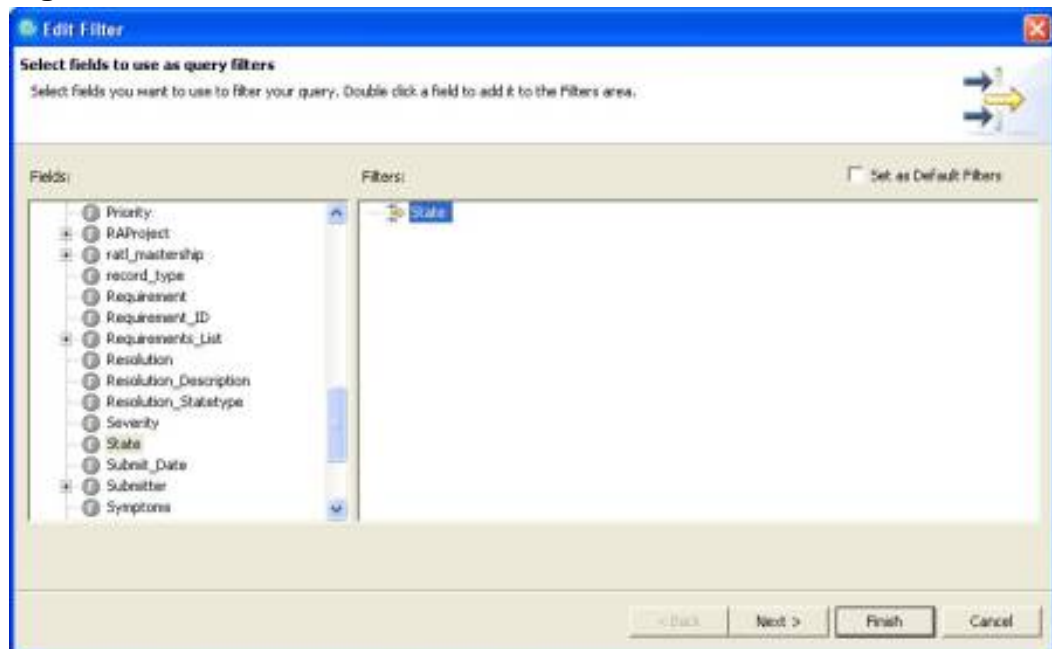
To copy a query:

1. Right-click the query you want to copy in the Navigator, then select **Copy**.
2. Right-click the folder in the Navigator into which you want to create the new query, then select **Paste**.
3. Right-click the new query, then select **Rename**. Call the new query Submitted defects.

Now, you need to add the filter to your new query:

1. Click the box with the plus sign next to the query to expand the query definition.
2. Double-click **Filters** to edit the filters. You should get a window like the one shown in Figure 28. In the figure, we've already added the **State** field to the filter list. Add as many additional fields as you'd like to this list. Click **Next**.

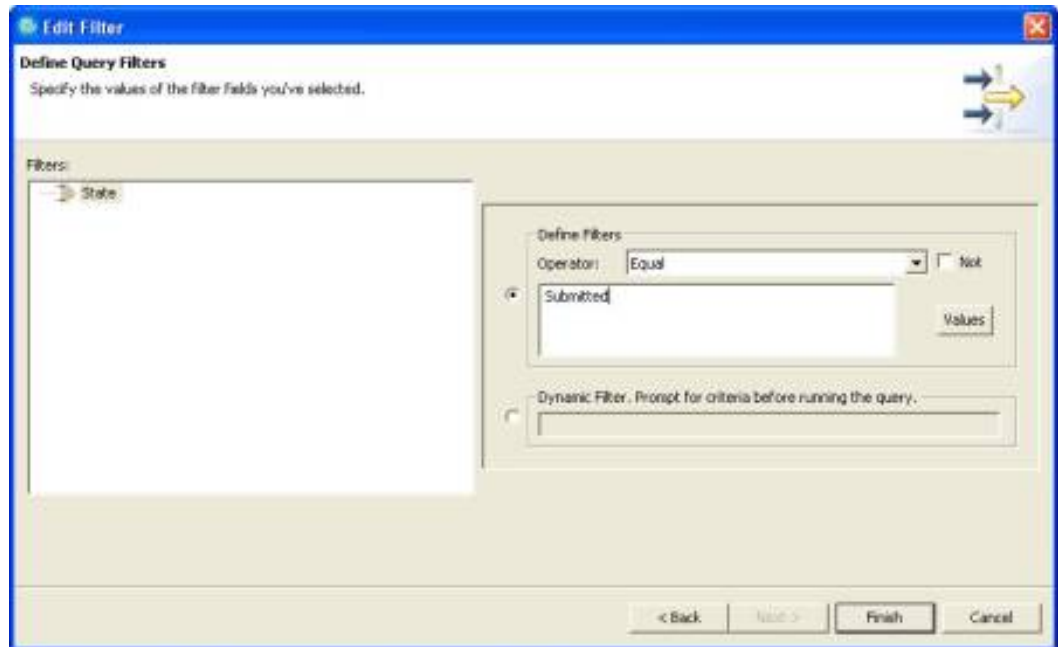
**Figure 28. Edit filters**



3. When prompted, specify the filter settings for each field. Select the **State**

field. The window shows the filter settings for this field. Select **Equal**, then enter `Submitted` into the value box. (Note that you can click **Values** to get a list of potential values for this field if you need guidance.) A populated version of the filter settings is shown in Figure 29.

**Figure 29. Set filter parameters**



4. Click **Finish** to save the filter definition.

That's it! If you run the query, you should see a reduced list of defects in the Rational ClearQuest database containing only those defects that have been submitted.

The most valuable queries are those that give you, at a glance, a list of the currently open, submitted but unassigned, closed, or resolved requests, along with other similar straight-answer reports. These help you monitor the development status and workload within your project.

## Classifying requests

After 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) should meet regularly to evaluate change requests that were posted and logged since the last

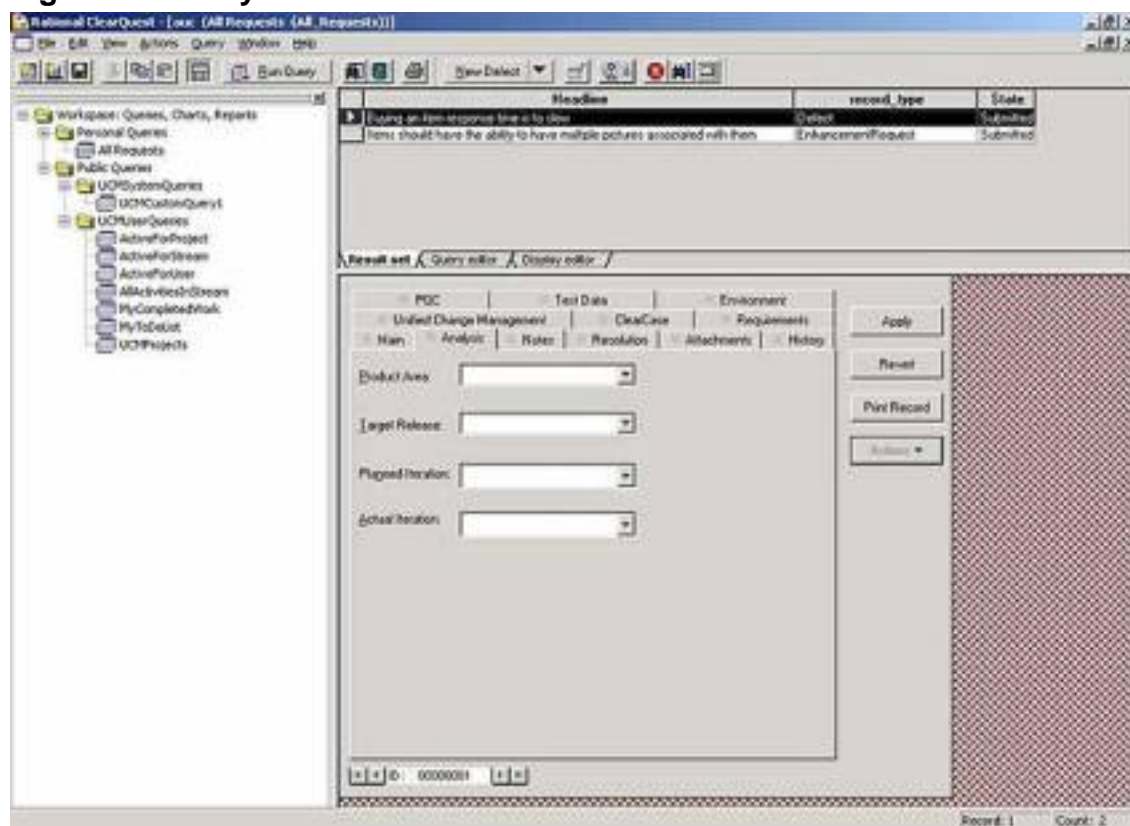
meeting. In those meetings, team members can complement the change request information with their own perspectives.

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

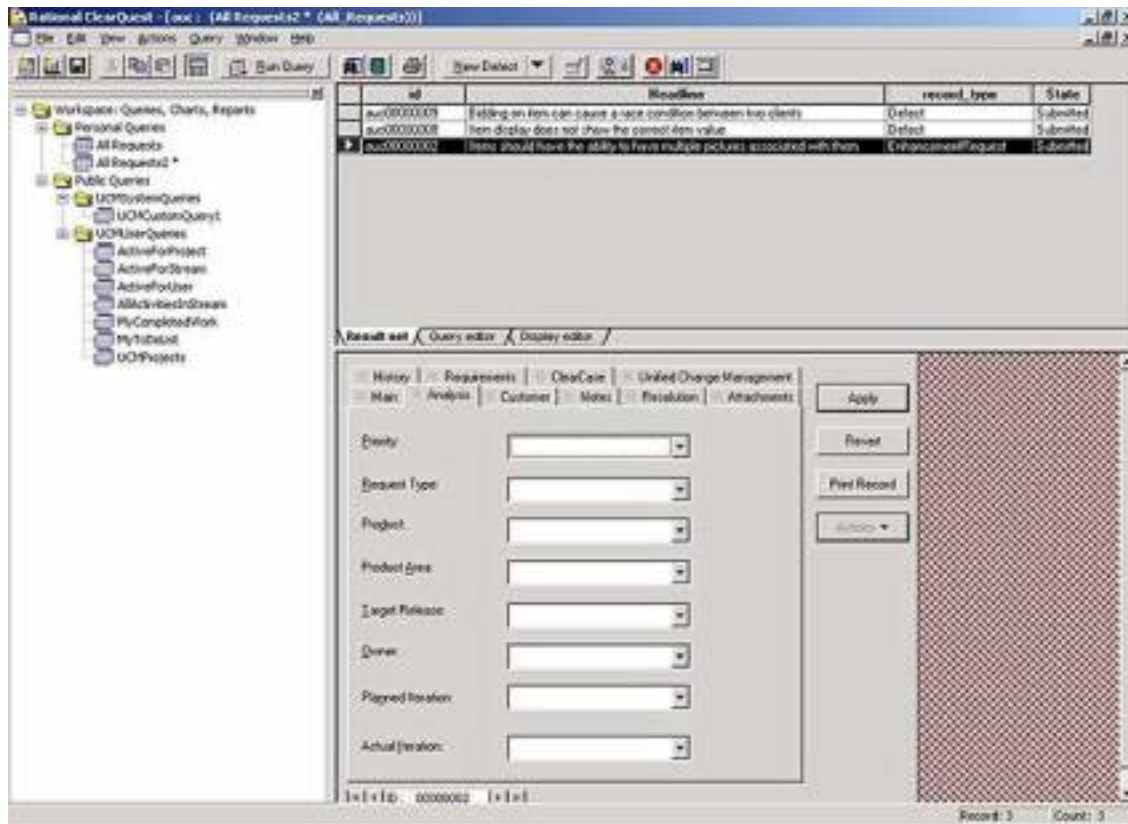
This classification process can occur within Rational ClearQuest or Rational Application Developer; changing the type of classification that is occurring will generally alter which tool is being used. For example, developer classification would probably occur within Rational Application Developer, while manager classification would take place within Rational ClearQuest. The screens and windows are essentially the same; they both ask for the same information, and only the presentation differs between applications.

You can see the Analysis tab for a defect in Figure 30 and for an enhancement request in Figure 31. Both of these are taken from Rational ClearQuest.

**Figure 30. Analysis tab for a defect**



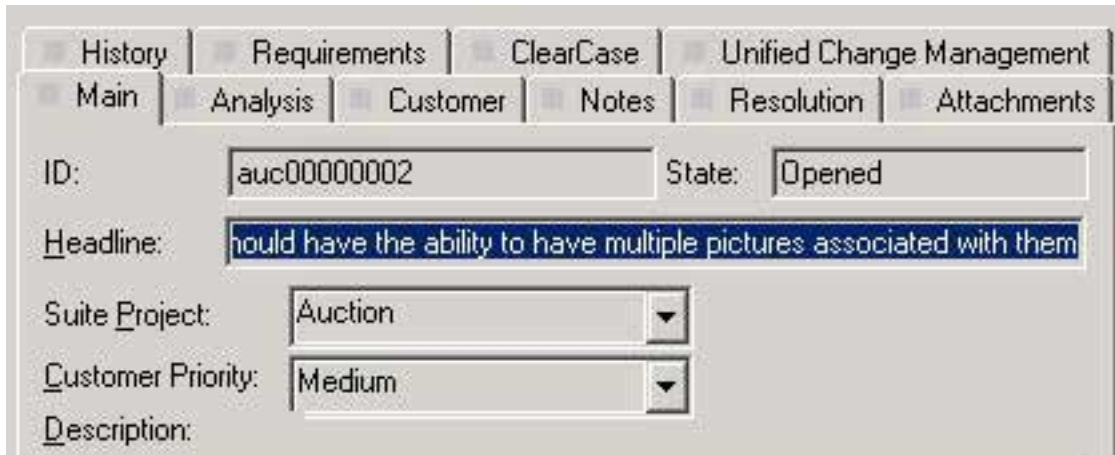
**Figure 31. Analysis tab for an enhancement request**



## Accepting or rejecting requests

When all requests have been analyzed and the team perspective has been recorded, Rational ClearQuest queries provide a project manager with a way to look at a group of requests at once, based on the fields established in previous panel. Because each team has limited resources, managers need to decide which requests can be worked on and which ones are either rejected (for enhancement requests) or deferred (for enhancement requirements and defects), as Figure 32 shows.

**Figure 32. Setting priorities**



The screenshot shows a software development tool interface with a request form. At the top, there are several tabs: History, Requirements, ClearCase, Unified Change Management, Main, Analysis, Customer, Notes, Resolution, and Attachments. Below the tabs, the form contains the following fields:

- ID: auc00000002
- State: Opened
- Headline: ould have the ability to have multiple pictures associated with them
- Suite Project: Auction (dropdown menu)
- Customer Priority: Medium (dropdown menu)
- Description: (empty text area)

You can set this information by opening a request or defect and then adjusting the priority (to set the severity of the issue) or by setting the *State* of the request or defect in the database.

## Assigning requests

You should assign requests for defects that are selected to be fixed in the next release 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 that 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 still needs to be identified or assigned to a developer for resolution.

The way you would assign change requests differs depending on the schema. In the schema for our sample application, the process is simply a case of assigning the defect to a member of the development team. The assigning of an enhancement request 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 or her queue.

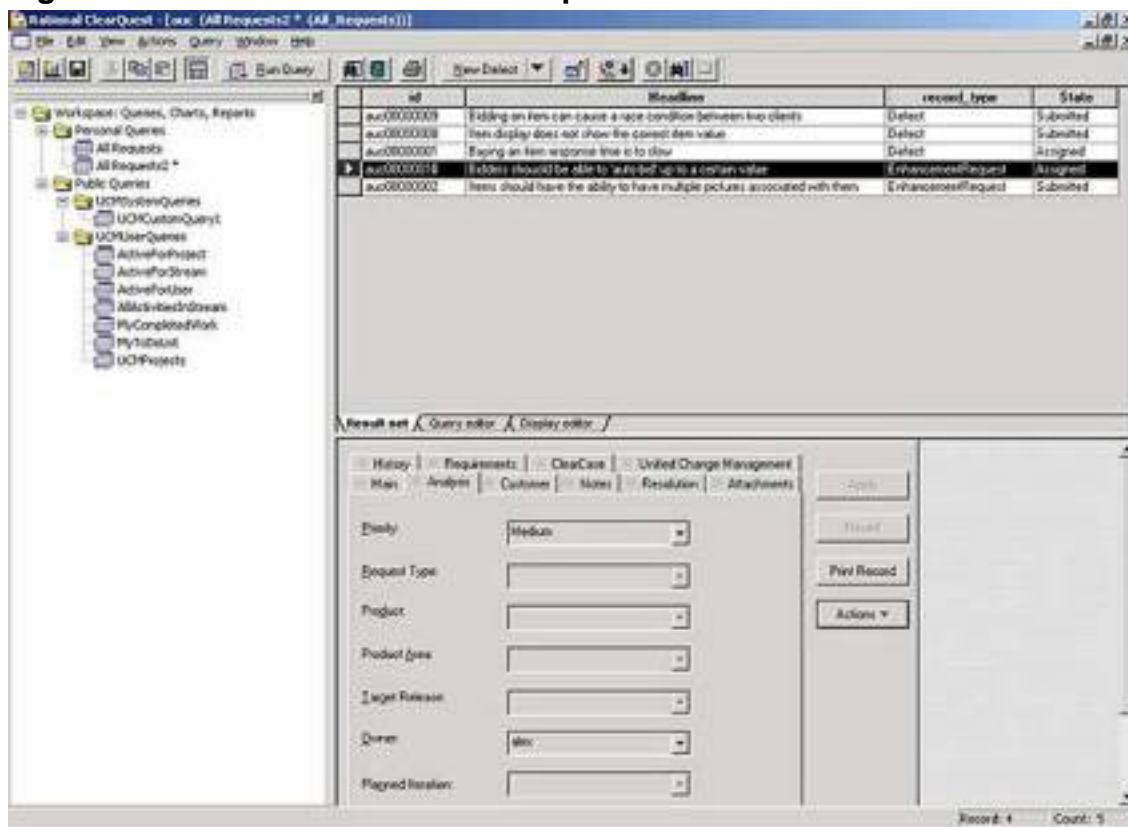
Minor enhancement requests to existing functionality can similarly be assigned to developers. These do not affect the requirement specification that describes the software delivered at the end of the project. However, enhancement requests that reflect new functionality first need to be incorporated into the requirements specification before being assigned to a designer to work on. (See [Integrating enhancement requests into a requirement specification](#) for more on this.) For minor enhancement requests, you need to provide a priority for the request so that the assignee can prioritize the enhancement requests with other tasks in his or her 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 is still in the Submitted state.

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

The results of assigning your auto-bid enhancement request is shown in Figure 33.

**Figure 33. The new enhancement request list**



## Resolution of requests

After a developer has fixed a particular defect or implemented an enhancement, the process has one more stage: The request must be highlighted as being resolved.

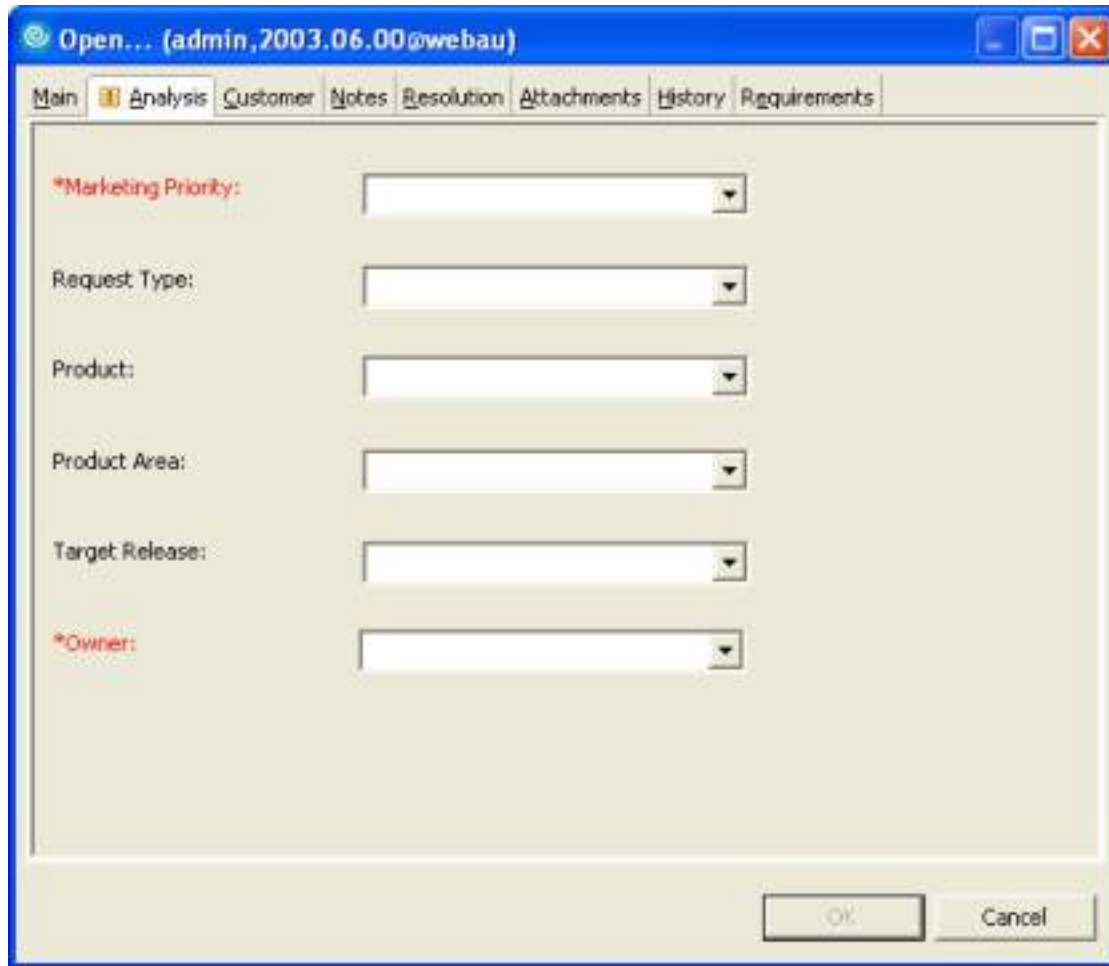
You can do so automatically from TestManager or Rational PurifyPlus if you are specifically dealing with a known and correctly tagged defect request.

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

Developers typically use a "My ToDo List" query to find work assigned to them (using the techniques seen earlier in this tutorial). This query shows all requests in the Assigned state, where the owner of the request is the CURRENT\_USER. CURRENT\_USER, which is itself 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 from within Rational ClearQuest or within the ClearQuest Query Results window. Then, click **Actions > Open**. You can fill in some information about what you are doing or click **Apply** to mark the item as open.

When the problem has been rectified or the enhancement request implemented, select **Action > Close** and fill in the form shown in Figure 34.

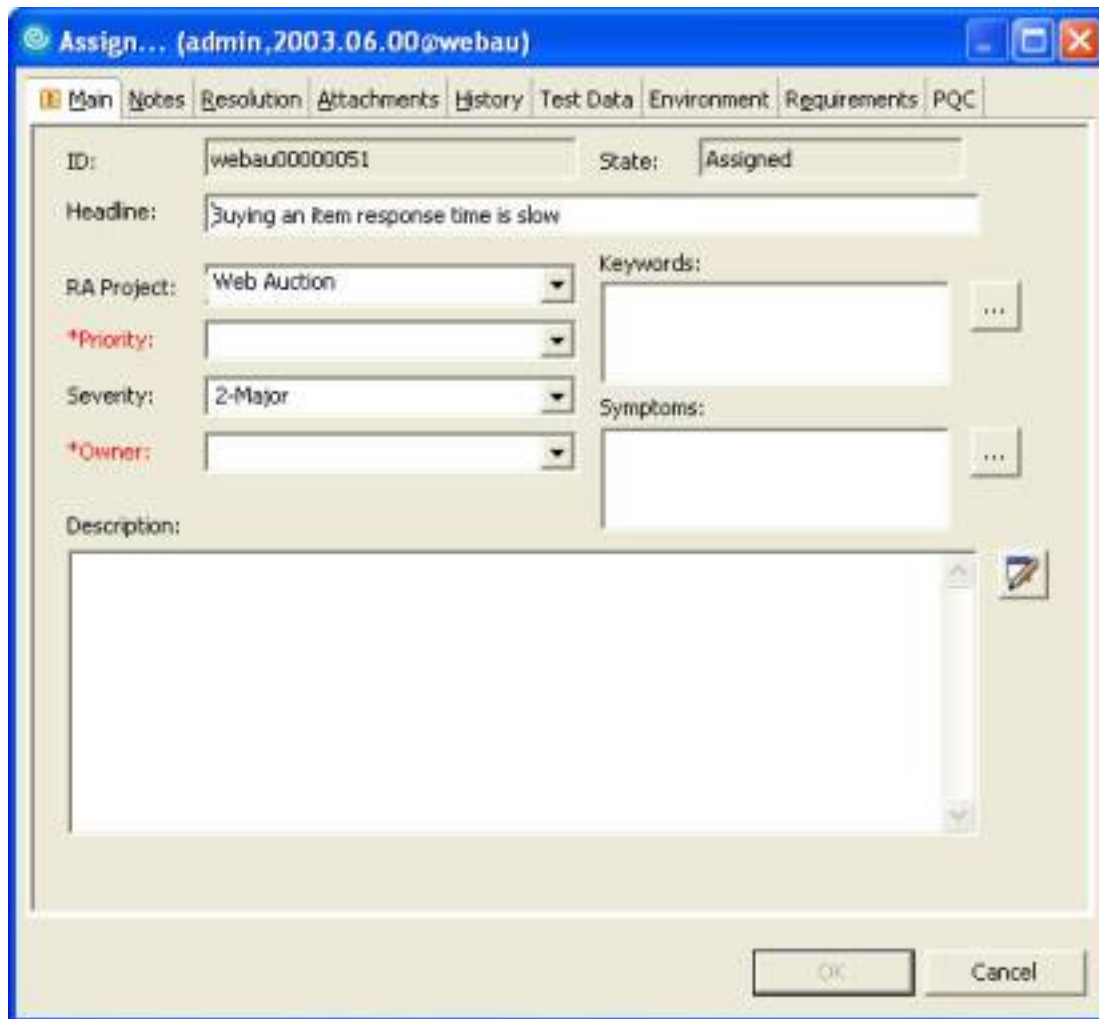
### Figure 34. Resolution request form



The screenshot shows a software application window with a blue title bar and a menu bar. The title bar text is "Open... (admin,2003.06.00@webau)". The menu bar includes "Main", "Analysis", "Customer", "Notes", "Resolution", "Attachments", "History", and "Requirements". The main content area is a light beige color and contains six dropdown menus, each with a small downward arrow on the right side. The labels for these dropdowns are: "\*Marketing Priority:", "Request Type:", "Product:", "Product Area:", "Target Release:", and "\*Owner:". The asterisks on the first and last labels indicate they are required fields. At the bottom right of the window, there are two buttons: "OK" and "Cancel".

If the request was a defect, provide details of the resolution. You can see a sample of this in Figure 35.

**Figure 35. Resolution details for defects**



You've now followed a request from its inception through reporting, then closed 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, you need to integrate with Rational RequisitePro. The next section shows how this integration works.

---

## Section 5. Integrating enhancement requests into a requirement specification

## The requests and requirements relationship

As mentioned earlier, when enhancement requests logged in Rational ClearQuest are accepted by the team, you need to make sure that the requirement specification is updated with that newly added functionality so that developers and testers relying on the requirement specification (model, classes, and code, as well as validation procedures) to do their job actually work on the latest set of requirements and not an obsolete specification.

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:

1. **Enhancement requests.** These requests are typically 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 is created in the requirement specification to represent that missing functionality.
2. **Defects.** Defects are attached to the use case definition for the object or class that is known to be causing the problem. By tracking defects in this way, you can highlight the requirement that the defect is related to. This highlighting is beneficial in analyzing patterns in poorly defined requirements. If lots of defects are attached to a use case, you know that the use case might not have been fleshed out enough before the implementation work started -- a lesson learned for your next project.

You can integrate Rational ClearQuest and Rational RequisitePro by creating an association between a requirement type within Rational RequisitePro and a change request type in Rational ClearQuest.

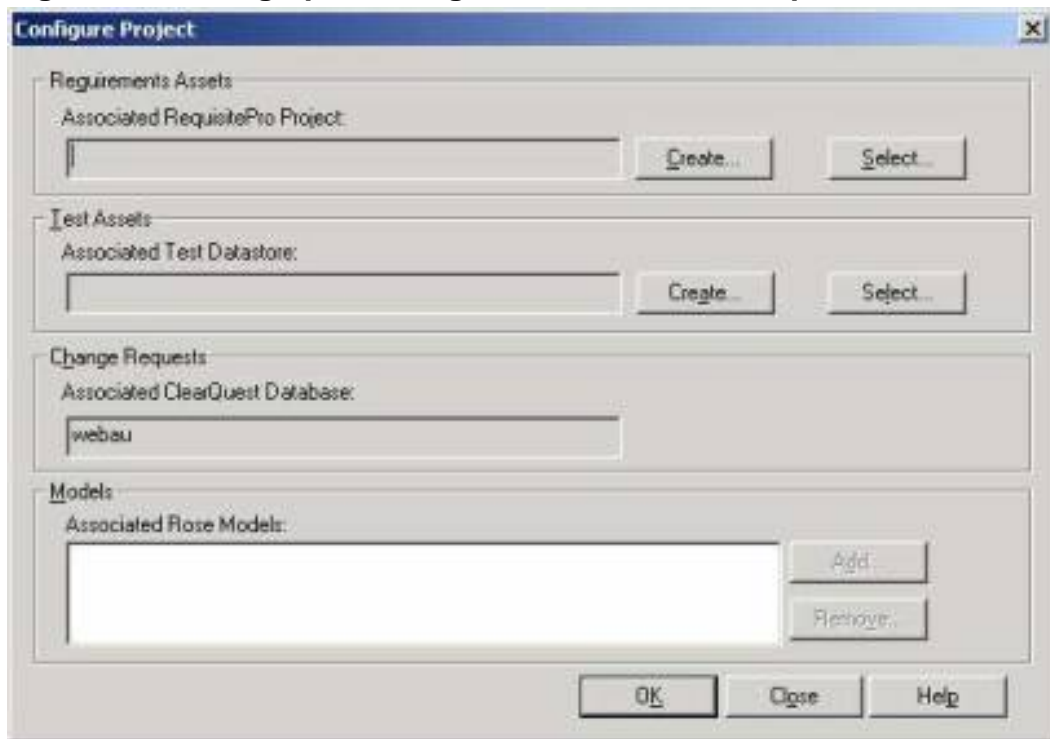
## Associating a Rational RequisitePro project with your Rational project

In this panel, you're going to use Rational Administrator again to associate the Rational RequisitePro project you created in Part 1 of this series with the Rational ClearQuest database you created earlier in this tutorial. 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, perform the following steps:

1. Close Rational ClearQuest if it's open (not required, but recommended).
2. Open Rational Administrator.
3. Right-click the **WebAuction** project you created, then select **Configure** . If prompted (and if you created one for the Rational project), enter the password. A window like the one shown in Figure 36 opens.

**Figure 36. Setting up the integration to Rational RequisitePro**



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

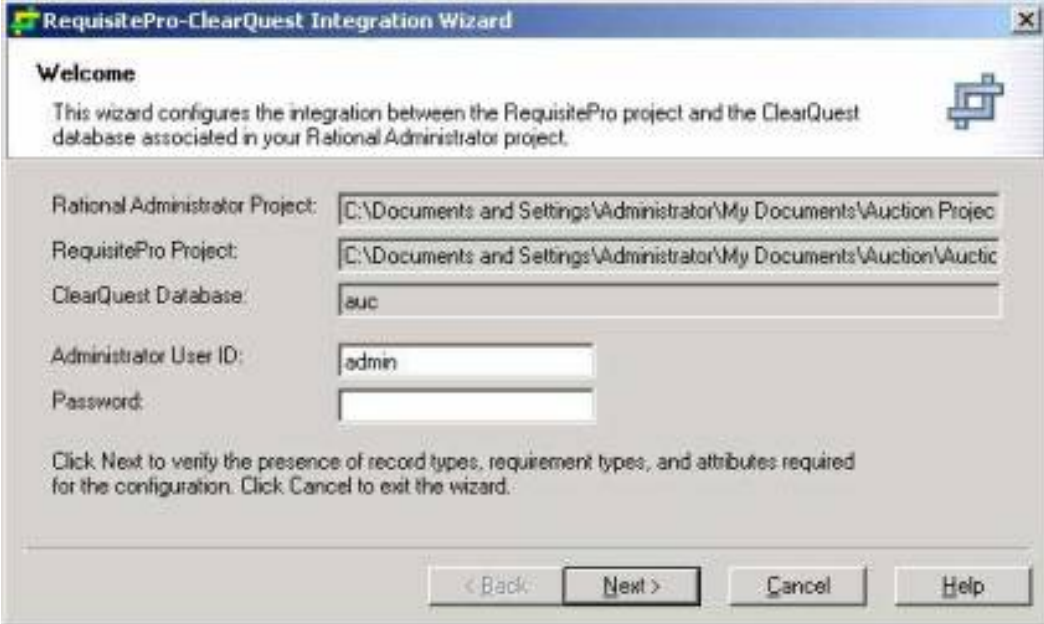
When you've associated the Rational RequisitePro project with your Rational project, the Rational ClearQuest-Rational RequisitePro Integration wizard automatically starts. The wizard takes you through the basics of associating the Rational RequisitePro project with the Rational ClearQuest database, including modifying the Rational RequisitePro project to incorporate the new requirement types used to help

trace and track the integration between the two systems.

Perform the steps below to complete the wizard:

1. The first screen, illustrated in Figure 37, summarizes the main information about the integration. Supply the appropriate login and password for the Rational ClearQuest database. You don't need to change anything here, so click **Next**.

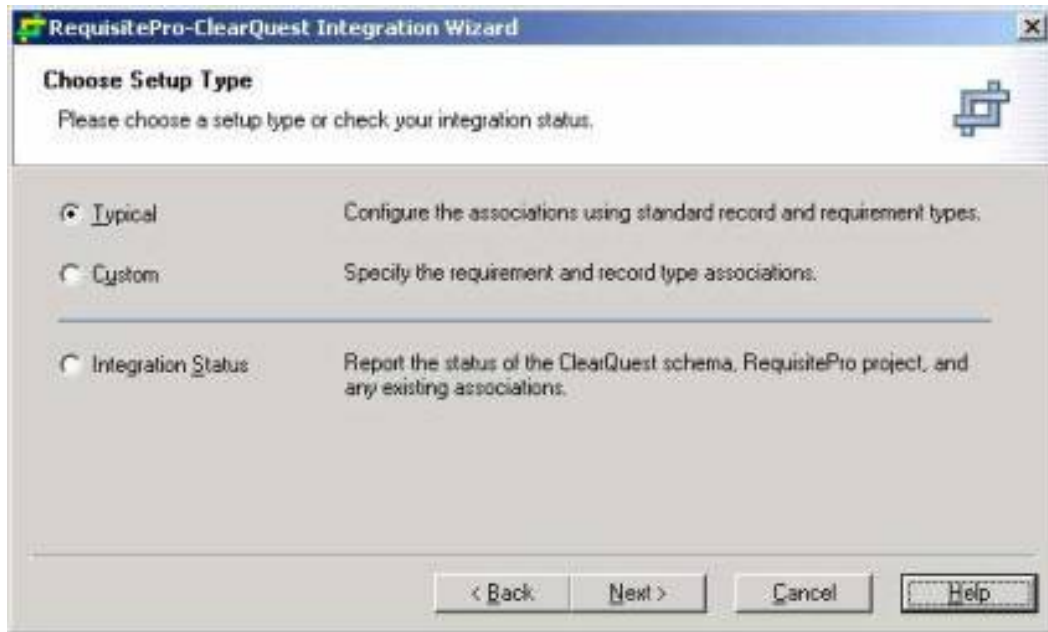
**Figure 37. Running the integration wizard**



The screenshot shows a Windows-style dialog box titled "RequisitePro-ClearQuest Integration Wizard". The main area is titled "Welcome" and contains the following text: "This wizard configures the integration between the RequisitePro project and the ClearQuest database associated in your Rational Administrator project." Below this text are five input fields: "Rational Administrator Project:" with the value "C:\Documents and Settings\Administrator\My Documents\Auction Projec"; "RequisitePro Project:" with the value "C:\Documents and Settings\Administrator\My Documents\Auction\Aucti"; "ClearQuest Database:" with the value "auc"; "Administrator User ID:" with the value "admin"; and "Password:" which is empty. At the bottom of the dialog, there is a paragraph of instructions: "Click Next to verify the presence of record types, requirement types, and attributes required for the configuration. Click Cancel to exit the wizard." and four buttons: "< Back", "Next >", "Cancel", and "Help".

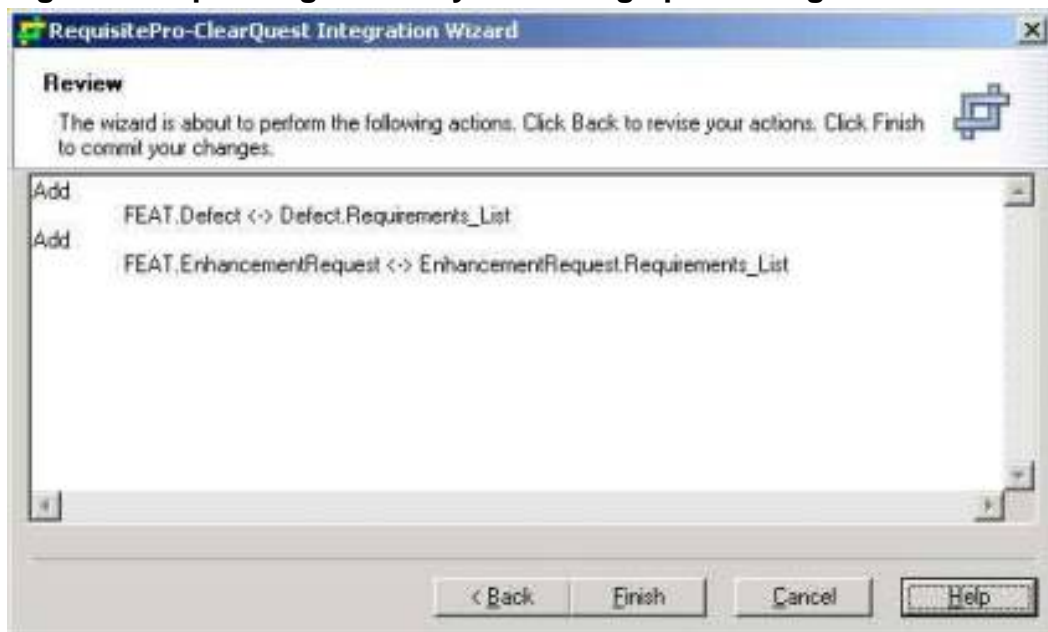
2. The Choose Setup Type window controls how information is associated between the two tools. The **Typical** option links Rational ClearQuest Defects and Enhancement Requests record types to either Feature or Use Case requirement types. If you use other record or requirement types or want to establish other associations, select the **Custom** option to select which Rational ClearQuest record types to associate with specific Rational RequisitePro requirement types. You can also select **Integration Status** to report on the state (and any issues) of the integration. For now, though, use the **Typical** setting, as shown in Figure 38.

**Figure 38. Choosing the setup type**



3. The final window, illustrated in Figure 39, summarizes what the integration wizard is going to do. 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 39. Operating summary for setting up the integration**



When the wizard is finished, it returns you to the main Project Configuration window.

You're now ready to start associating the feature definitions in your original requirements specification with requests in Rational ClearQuest. In the next section, you'll start by associating an enhancement request with one of the original requirements.

---

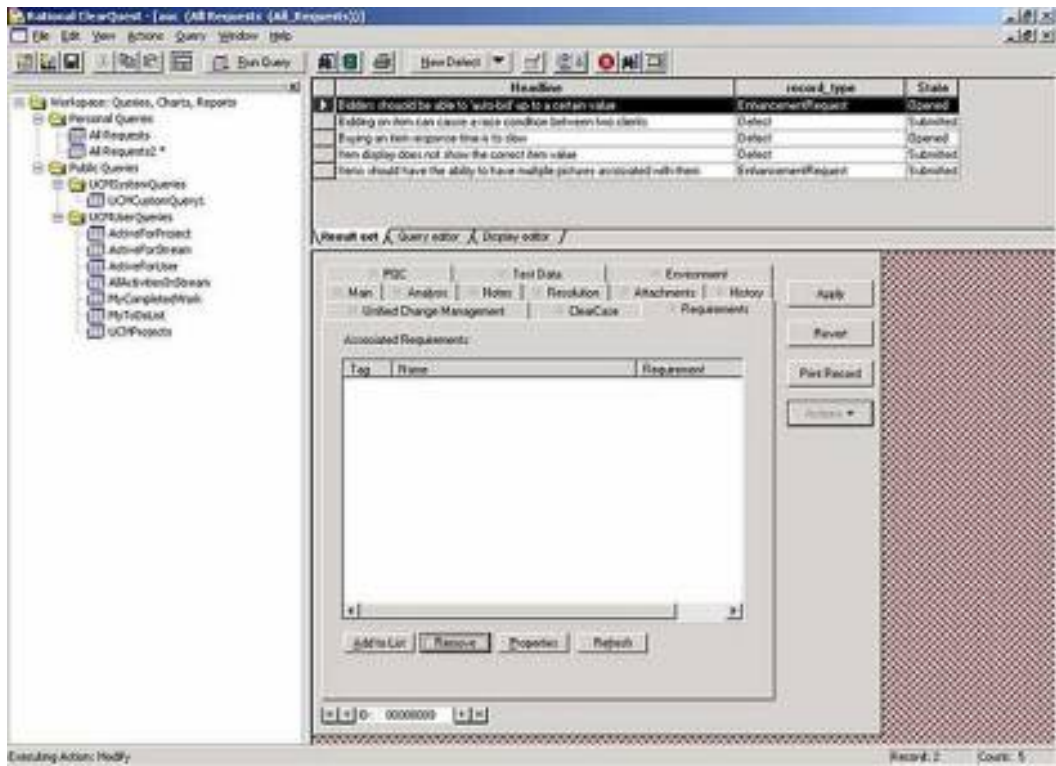
## Section 6. Building a new requirements specification

### Create a request-requirement association

Associating a request to a requirement gives you a look into the origin of requirement and requirement changes. You can also use the information produced to provide metric information, such as the requirement or feature with the highest number of faults or enhancements, that managers can then use to target resources and prevent feature creep in a product.

To create an association, perform the following steps:

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 **Actions > Modify** .
3. On 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. Click the Requirements tab, as shown in Figure 40.  
**Figure 40. Linking to a requirements project**



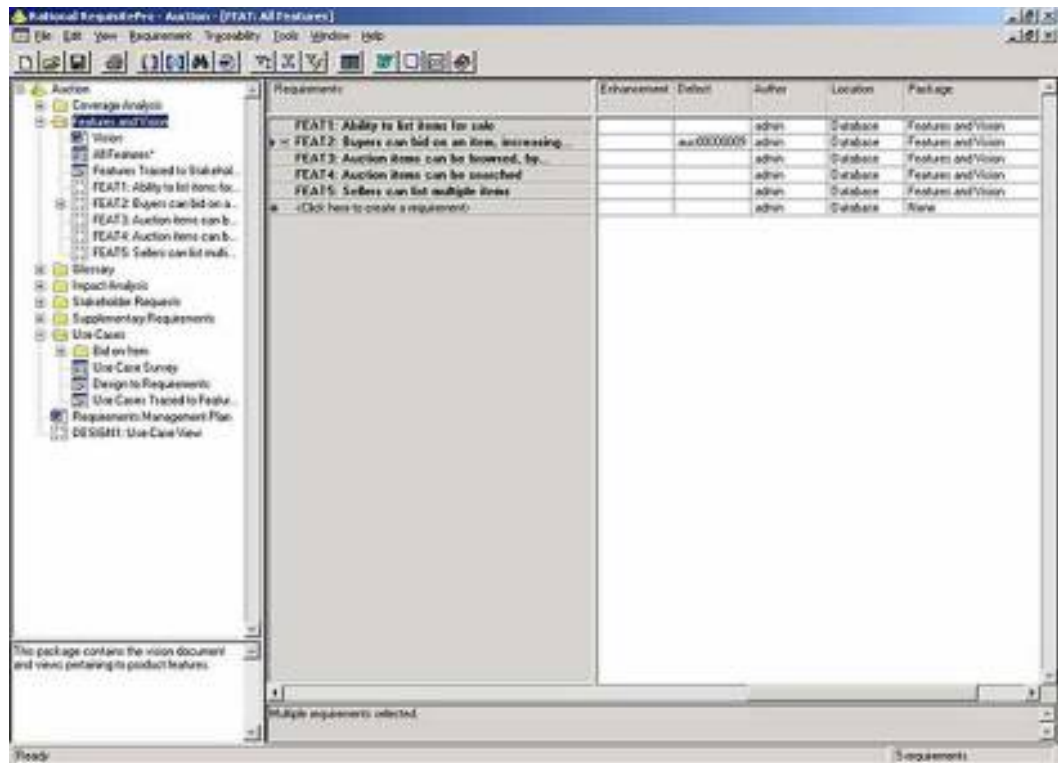
5. Click **Add to List** . A list of all the requirements in the Rational RequisitePro project associated to this Rational ClearQuest database is displayed, as shown in Figure 41.
6. Select **FEAT: Feature** as the requirement type, then select **Buyers can bid on an item** as the feature requirement.

**Figure 41. Associating the requirement with your change request**



7. Close this window, then click **Apply** .
8. In Rational RequisitePro, select a Rational RequisitePro view that contains the list of features in the Defect or Enhancement Request attribute. You'll see that the Rational ClearQuest change request number has been entered, as shown in Figure 42.

**Figure 42. List of features in Rational RequisitePro**



- Click the cell that lists the Rational ClearQuest CR. Click the ... button that is displayed. 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 offered here is that you can 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 in 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. For most feature requests though, the requirement will be proposed -- in other words, "nice to have" -- rather than a specific problem (which would of course be reported as a defect).

Rational ClearQuest and Rational RequisitePro are typically connected through the Feature Requirement type, allowing you to create FEAT requirements directly 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, perform these steps:

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 **Actions > Modify** .
3. On 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. Click the Requirements tab, then 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, as shown in Figure 43.

**Figure 43. Set properties of the feature change**



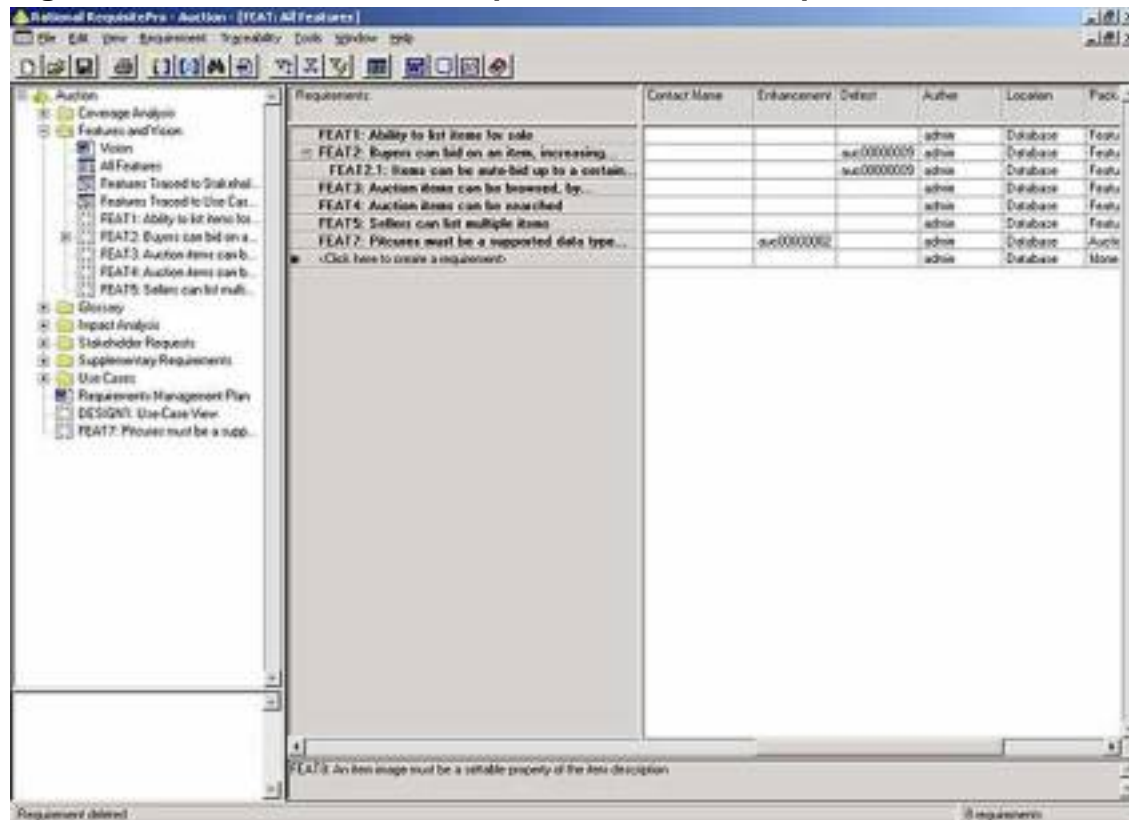
6. Set the properties of the new requirement. Take some care in writing the text. A typical pitfall is simply to copy the text of the enhancement request into the text of the requirement. Requirements must be testable, unambiguous, and compliant to the team glossary. You run the risk of

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, then click **OK**.
8. 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, as shown in Figure 44.

**Figure 44. The enhancement request in Rational RequisitePro**



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

1. Right-click the new requirement in the view, then select Cut.
2. Open the Microsoft Word document containing the requirements, then position your cursor where you want to see that new feature in the document.
3. Select **RequisitePro > Requirement > Paste** (be careful not to use the Word Paste function, but rather the Rational RequisitePro Paste function).

The Cut function does not really cut the requirement from the database but rather allows you to display a requirement in a Word document.

After a new feature is added to the original requirement specification, you have to complete two final steps: Creating detailed requirements (use cases) to ensure that the new feature is implemented, and updating the model and design to reflect the new use cases. This step is normally carried out using the business analyst (who probably generated the original requirements specification) in discussion with the customer. You'll tackle these in the next panel.

## 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 of this series to create any requirement. Once you have modified and created the use cases, you should establish a traceability link between the newly added feature and any use cases it affects, so that if the feature changes later, you can quickly pinpoint the use cases that 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 change request (CR) and the use case (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 affecting the use case, so that you can retrofit the accepted request into the use case specification document in Rational RequisitePro.

---

## Section 7. 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 determine the changes that 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.

The integration between Rational Application Developer and Rational ClearQuest allows developers to use the Rational ClearQuest information to help drive their development and concentrate their efforts. Because the information is available to

them from within the application they are using for development, they have all the information they need without having to swap applications or view a Web site. And the easier it is for them to use the information, the more likely they are to use it. They are also more likely to use and update the Rational ClearQuest database and record the progress through the system.

In Part 4 of this series, you will use IBM Rational Application Developer to develop session beans and create pages that access your session beans using JavaServer Faces technology. In Part 5, you'll look at the role of testing packages in the development of an application.

# Resources

## Learn

- ["Collating requirements for an application, Part 1"](#), the first part of this tutorial series, introduces the process of defining requirements.
- ["Developing solutions with Rational Application Developer, Part 2"](#) converts the requirements into a working application model.
- [developerWorks Rational](#) is a place to learn more about Rational. You'll find technical documentation, how-to articles, downloads, product information, and more.
- [Purchase Rational books at discounted prices](#) in the Rational section of the Developer Bookstore.
- [RUP support page](#). RUP is a structure development model that affects everything from the documentation to the sequence of development.
- [IBM Software Development Platform home page](#). Learn more about the IBM Software Development Platform.
- ["Reduce complexity with model-driven development, Part 1: Use the IBM Software Development Platform to develop end-to-end solutions"](#). Read this article to see how to use the tools in the IBM Software Development Platform to create an end-to-end development cycle.
- [Eclipse project](#). Eclipse is an open source IDE that forms the basis of many IBM products, including Rational Software Modeler and Rational Application Developer.
- [IBM Software Development Platform webcast](#) . Join a live webcast and participate in the Q&A session or replay the recorded webcasts at your convenience.

## Get products and technologies

- [Rational RequisitePro](#) Version 2003.06.13
- [Rational Application Developer](#) Version 6.0
- [Rational Software Modeler](#) Version 6.0

## Discuss

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

## About the author

## Martin C. Brown



Martin C. Brown 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](http://www.foodware.net). Now a freelance writer and consultant, MC (as he is better known) works closely with Microsoft® as an subject matter expert, is the LAMP Technologies Editor for *LinuxWorld* magazine, is a core member of the [AnswerSquad.com](http://www.answer-squad.com) team, and has written 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 through this Web site at <http://www.mcslp.com>.