

# Improved application development: Part 1, Collating requirements for an application

Skill Level: Intermediate

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

21 Jun 2005

Developing applications using the IBM Rational Unified Process is a lot easier if you have the tools to help you throughout the process. The Rational family of software offers a range of tools that on their own provide excellent support for each phase of the development process. But you can also use the different tools together to build an entire application. By sharing the information, you can track components in the application from their original requirement specification through testing and release. This first part of a five-part series shows how to use Rational RequisitePro to manage and organize the requirements specification for a new project. Then, after you've developed your unified list of requirements, the tutorial shows how to use Rational Software Modeler to model your application based on those requirements.

## Section 1. Before you start

### About this tutorial

Many people and many steps are involved in developing software, and not all of them relate to actually writing code. Defining the functionality and requirements of the project, controlling the code base, and monitoring progress are all processes that need to occur, regardless of whether you're involved in developing the code.

The IBM® Rational® family of products can help the entire application development process, from the inception of an idea through final release and bug tracking. The Rational family includes tools for defining requirements IBM Rational

RequisitePro®), visually modeling (IBM Rational Software Modeler) and coding (IBM Rational Application Developer) an application, tracking defects (IBM Rational ClearQuest®), and managing changes to the application (IBM Rational ClearCase®). All these tools can be used with or without the support of IBM Rational Unified Process® (RUP).

To make the development process even easier, you can combine the functionality of individual Rational applications into a single, cohesive environment for developing, managing, and tracking applications. Each product has methods for communicating with the others to obtain, update, and integrate the necessary information.

In this five-part tutorial series, you'll learn about the integration of these components and how to use them together to manage the application development process from its inception from a client to the point at which bugs and faults need to be traced and tracked through the system after the initial release. For the examples employed throughout this five-part series, the well-known Auction system -- a sample application used in many IBM applications -- is used.

Part 1 of this series shows how to use Rational RequisitePro to manage and organize the requirements specification for a new project. Then, after you've developed your unified list of requirements, the tutorial shows how to use Rational Software Modeler to model your application based on those requirements.

Key parts of this tutorial include:

- Introducing the development process
- Developing the requirements spec using Rational RequisitePro
- Documenting the project requirements through Rational RequisitePro and Microsoft® Word
- Accessing requirements data through Rational Software Modeler
- Translating requirements into an application model
- Tracing model elements back to the requirements

## Prerequisites

To run the examples and sample code in this tutorial, you'll need to download trial copies of Rational RequisitePro, Rational Software Modeler, and Rational Application Developer as well as the RUP package, which includes complete documentation on the techniques, methods, and systems behind RUP. You'll also need Microsoft Word (at least Word 97, although Word 2000 or later is preferred).

To complete the steps in this tutorial, you need:

- [Rational RequisitePro](#) Version 2003.06.x
- [Rational Application Developer](#) Version 6.0
- [Rational Software Modeler](#) Version 6.0
- [RUP](#)

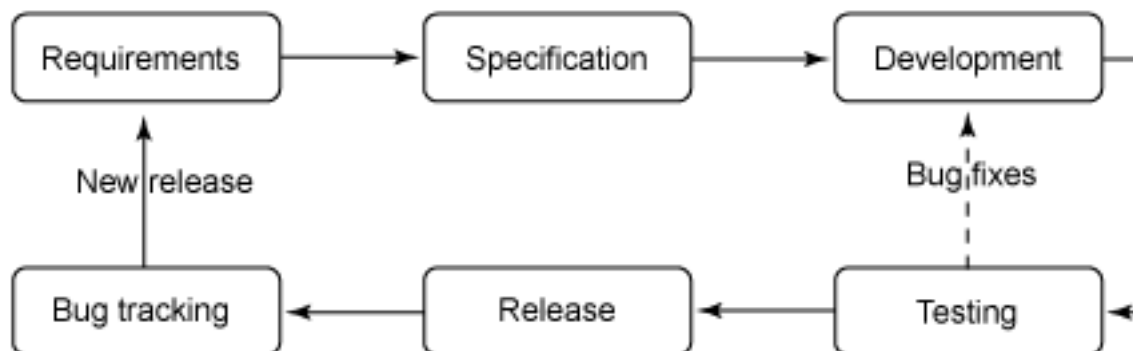
---

## Section 2. Development using Rational integrated applications

Building applications is a long and difficult process, often made even more difficult through the complexities of actually managing the development process itself. Collating requirements, building a suitable model, writing and extending the code, and testing the product are stages that all software goes through. After you've started to test and deploy your application, change requests and defect tracking add to this complexity. You need more in-depth management systems to track a defect from its origin during testing, to the source of the problem in the model, and maybe even to the original requirements.

Ignoring the Rational approach for a second, it's possible to model the development process like Figure 1.

**Figure 1. A simple development process**



If you look at this diagram, the inception of a new project starts with the definition of the requirements. After the requirements are sorted, you can develop a specification of the project followed by the development of the actual code and application. You can go through a series of testing, repeating the development cycle in the event of a bug, before finally releasing the software. When the software has been released, you need to track any bugs, which should eventually feed back into the requirements for the next release of the same project, at which point the whole process begins again.

RUP applies a similar, simple model to the development process. By looking at the Rational model, you can see how to apply the Rational tools to the process.

## Developing with RUP

Developing applications using the RUP centers around the same basic principles outlined in the previous panel but applies three main features to the process:

- **Iterative and incremental development:** Rather than designing the entire application, then writing the code, then testing, go through a series of iterations within each following the same sequence but developing only part of the application each time.
- **Object oriented:** This makes the code easier to build and reuse and promotes the creation of small, flexible components that can be bonded together to create larger applications.
- **Management and control:** Create logs and a traceable sequence that enable you to track features from the original requirement to the component in the application.

The process can work in a completely manual fashion, but it also lends itself to automation, which is where the Rational family of products comes in. Using the Rational tools also simplifies the process by which the teams and individuals working in your development department create applications quickly and easily. The result is a development cycle that looks like the one shown in Figure 2.

**Figure 2. The RUP model**



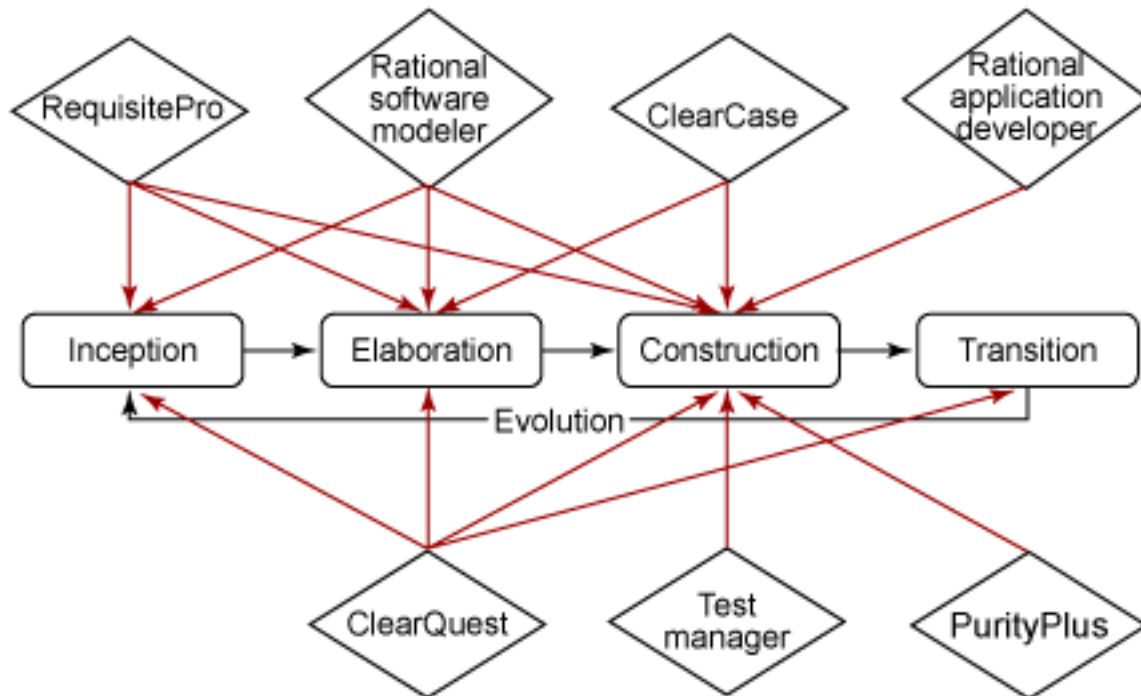
Figure 2 follows the same basic sequence described in the previous panel. It's important to note, though, that the development process is iterative rather than sequential; that is, all these stages occur during a phase of development. The process moves from the inception of the project, defining the features and requirements, the elaboration that expands the requirements and defines the model, the construction that builds the model and generates code, and finally the transition where you start to track defects. This process automation relies on several Rational applications, which are introduced in the next panel.

## Developing with the Rational tools

Rational software includes a variety of tools that enable you to develop an

application using the RUP principles. You can attach specific tools to different parts of the process according to Figure 3. Note that the tools given here are examples only; there are many different tools in the Rational Suite® , and the exact tool you use depends on the application you are developing.

**Figure 3. Rational tools and the RUP model**



In more detail, the tools mentioned in Figure 3 cover the following tasks:

- **Rational RequisitePro:** This tool handles requirements management using a design such that requirements can be updated either directly or through Microsoft Word. You can change requirements by simply editing the content of a document, and you can update, track, and merge requirements through the tool.
- **Rational Software Modeler:** This tool fills the role previously held by Rational Rose® XDE. It is based on the Eclipse platform and was successfully used within the IBM WebSphere® family of products as the basis for the WebSphere Studio Application Developer integrated development environment (IDE). Rational Software Modeler provides an interface for modeling an application before the coding starts.
- **Rational Application Developer:** This tool provides an IDE to turn models into code and the final application. Rational Application Developer also provides an integrated environment for testing a completed application.
- **Rational ClearCase:** This tool provides configuration management for

your code base and models. WebSphere Studio Application Developer and Rational Rose XDE can interface with Rational ClearCase to provide management across the whole development process for your entire team.

- **Rational ClearQuest:** This tool facilitates defect tracking and activity management. The defect tracking allows you to trace and track errors back to their original location. Activity management enables you to track the activity on different components of your development, including code, models, and other information. Rational ClearQuest integrates with the other Rational applications so that you can track defects and activity across components and requirements and follow the information from component to component.
- **Rational PurifyPlus:** Rational PurifyPlus provides run time analysis of your application to monitor its use of memory. Memory leaks and problems are one of the most common faults in applications barring typographical errors and bad programming, which should be eliminated through the testing and methodologies applied when using the Rational tools. This tool integrates with WebSphere Studio Application Developer to provide information about the elements of your application that are causing problems.
- **Rational TestManager:** Rational TestManager provides a single interface for testing a range of parameters for your application and includes the ability to automate that testing, which is particularly useful if you're deploying on multiple platforms.
- **Team Unifying Platform:** Rational Team Unifying Platform (TUP) is a collection of the core tools (RUP, Rational RequisitePro, Rational ClearCase, Rational ClearQuest, Rational TestManager, and Rational SoDA®) and a new tool called Rational ProjectConsole. Rational ProjectConsole aggregates status and metrics information from all the tools, including Rational Rose XDE, and provides a central location for viewing and disseminating status information across a development team.

## The auction application

Throughout this tutorial series, you'll be building an auction application modeled on the auction application that has been used in many other IBM software products, most notably WebSphere Studio Application Developer. You'll build the application from scratch, using each Rational tool at each stage of the development process. The Online Auction is a Web application that employs Java 2 Enterprise Edition (J2EE) technology using Java beans, Java Database Connectivity (JDBC), HTML, JavaScript code, JavaServer Pages (JSP) files, and other components. The result is an auction system that is functionally similar to eBay, Yahoo, and Amazon.

If you want to obtain a working version of the IBM Online Auction application, you

can download it as part of WebSphere Studio Application Developer; it's one of the free samples provided. View the system by installing WebSphere Studio Application Developer, then following the built-in help and instructions to create a WebSphere project based on the Auction code. You can also download a training video for building the system from the IBM Web site. (See the Resources section for more information.)

---

## Section 3. Manage requirements with Rational RequisitePro

A *requirement*, at its most basic level, is the most fundamental capability that an application should provide. Requirements should be as specific and definitive and possible. For example, you could have a requirement for the auction system of *Allows you to buy and sell goods through an auction-like system.*

Such a requirement really wouldn't be that useful when it came to developing the application. You need specifics about how the different parts work -- for example, how bids are made and processed, how a sale proceeds, and how information about the products in the auction are made available and listed for others to view.

Requirements come from a variety of sources, with the primary source being the various stakeholders in the project. The client is usually the primary stakeholder in any development, whether that's identified as an internal department, an external company, or the general public.

The problem is obtaining and collating these different requirements from the stakeholders to produce the initial list of requirements that will be used to start developing the application. This task is what Rational RequisitePro performs: It provides a mechanism for recording all the various requirements from the different groups, then builds a final list of the requirements and their inter-relationships.

These requirements can be specified and defined in several different ways, including basic statements and use cases. The system is easy to customize, and individual requirements can have an unlimited number of properties (fields) used to store additional information about each requirement.

### Driving development with requirements

After the requirements have been collated, they should be used to help drive application development. In theory, requirements should be used to drive all

software development. In reality, the initial sets of requirements provide a framework of functionality, and the requirements are extended and expanded upon to support the final application.

Traditionally, it has always been a problem communicating the requirements to the developers, then using the requirements to design the model, code, and application components that provide the functionality. Fortunately, the integration between Rational RequisitePro and Rational Software Modeler solves many of these problems.

By creating a formal link between the model (and ultimately the code) and the requirement that drives the development of a given component, you can ensure that the application being developed meets the requirements, does not overstep the requirements, and ultimately matches the needs of the project stakeholders.

Requirements form the basis of development, and by keeping them up-to-date with the current model, you can address issues and questions such as:

- What is the latest set of requirements?
- Does the current design match the requirements?
- Are all the requirements represented within the application model?
- Does the design meet the original goals of the stakeholders?

Address these questions by mapping requirements to model components.

## Use case recap

RUP is use case-driven, and you'll be using that approach in your application. Theoretically, it's possible (but not advised) to use the RUP with other requirement types; however, it never quite works out the same way, and the flow of information and descriptions between the different components is never the same.

**Note:** If you are already familiar with use cases, you can skip this panel. If you want a recap of the structure of use cases and how they drive development, read on.

A *use case* captures the agreement between the actors (that is, the users or components of the system) and a component's behavior. When an actor uses the system, the system performs a use case. The use case also describes the behavior of the system under several different situations. The result is a definition that specifies the sequence of events, from start to finish, including any alternative sequences to achieve the result.

The format of a use case is a text description that follows this basic outline within

## RUP:

1. Brief Description
2. Actors
3. Flows of Events
4. Special Requirements
5. Preconditions
6. Post-conditions
7. Extension Points

For example, within the auction application, you might have a sequence like the abbreviated example below, which is initiated by an actor bidding on an item in the auction system:

- 3. Flow of Events
  - 3.1 Main Flow
    - 3.1.1 Bid: The use case starts when a buyer bids on the currently displayed item.
    - 3.1.2 Enter Amount: The buyer enters the bid amount. The system ensures that the bid amount is larger than the current highest bid by a multiple of the bid increment for the item.
    - 3.1.3 Buyer confirms the bid: The buyer confirms that the bid should be placed.
    - 3.1.4 Post-bid: The system adds the bid to the item.
    - 3.1.5 Confirm bid: The system confirms the bid to the buyer through an e-mail. The seller is also notified by e-mail.

Alternative flows might describe what happens if the bidding has closed before the bid was submitted, the bid amount is not valid, or there is no confirmation from the buyer to submit the bid.

Use cases allow for a flexible description of the sequence but in such a fashion that they lend themselves well to a structured description and recording system, like that which Rational RequisitePro supports.

## Rational RequisitePro integrations

Rational RequisitePro can integrate with several different tools, both Rational-based and a few external tools. Rational RequisitePro's primary role is in the first stages of development to document requirements. You can document requirements directly within Rational RequisitePro in the following ways:

- Rational ClearQuest provides change-request management. You can integrate individual changes with the original requirements specification. Part 3 of this tutorial series provides more detail about merging original requirements with changes.
- Rational TestManager integration enables you to build test systems that will provide verification for all the requirements in the original project. This level of integration will be covered in part 5 of this tutorial series.
- Microsoft Office Project can be used with Rational RequisitePro. Using the requirements as a baseline and coupled with the additional properties of individual requirements, you can build a project plan based on the requirements specification.
- This tutorial, however, is concerned with the integration of Rational RequisitePro with Rational Software Modeler, which provides modeling tools, and Rational Application Developer, which provides an IDE for coding the application. In the next section, you start building the requirements specification.

---

## Section 4. Building a requirements specification

Rational RequisitePro provides a system for recording and managing requirements. The requirements system is based on a flexible database design that allows for an unlimited number of requirement types, each type having an unlimited number of fields that can be used to track additional information about each requirement.

An individual requirement can also be linked and identified as being a relation of another requirement. For example, you might have a top-level feature request for a security system, with use cases defining an authentication and authorization system. You would trace the use cases, and therefore the requirements used to define the functionality, back to the original top-level security feature.

In Rational RequisitePro, the requirements reside in a database. However, you can

record and edit them through a connection to Microsoft Word, which enables project managers and non-technical staff to generate requirements within a familiar interface while still generating and updating the information in the database.

Rational RequisitePro acts as the aggregator of information that project managers and team leaders use to define the project. After the initial version of the requirements has been developed, the developers can then use that information to help them build the application.

In this section, you learn the basics of creating requirements that you can use and integrate with other Rational tools.

## Key advantages of using Rational RequisitePro

The main advantage of Rational RequisitePro over stand-alone documentation is the control provided by a background, structured database. Because the information is in a structured format, you can generate reports, create connections to other requirements, and record additional information about the requirements beyond what you could typically describe within an Microsoft Word document.

Another advantage of Rational RequisitePro is that changes to the individual requirements are recorded and tracked. Even in an isolated situation with just one person working on the requirements, this information is useful. But it becomes vital when the requirements are used, monitored, and updated by a team of users. You can see exactly what changes have been made and why.

## Create a new project

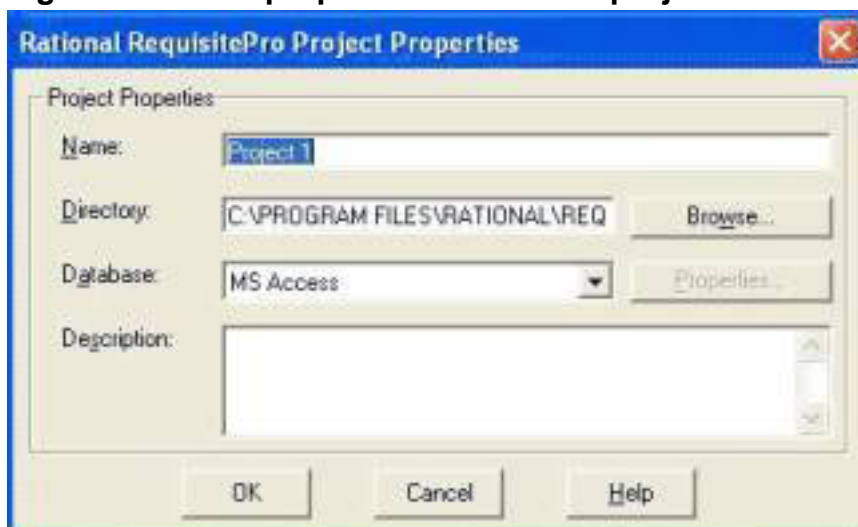
To start the process of building requirements, create a new project within Rational RequisitePro:

1. Open Rational RequisitePro, then choose **File > New**. A list of available project templates appears, as shown in Figure 4.

### **Figure 4. Rational RequisitePro project templates**



2. Create a document for use with RUP by selecting **Use-Case Template**. The Rational RequisitePro Project Properties window opens. **Figure 5. Set the properties for the new project**

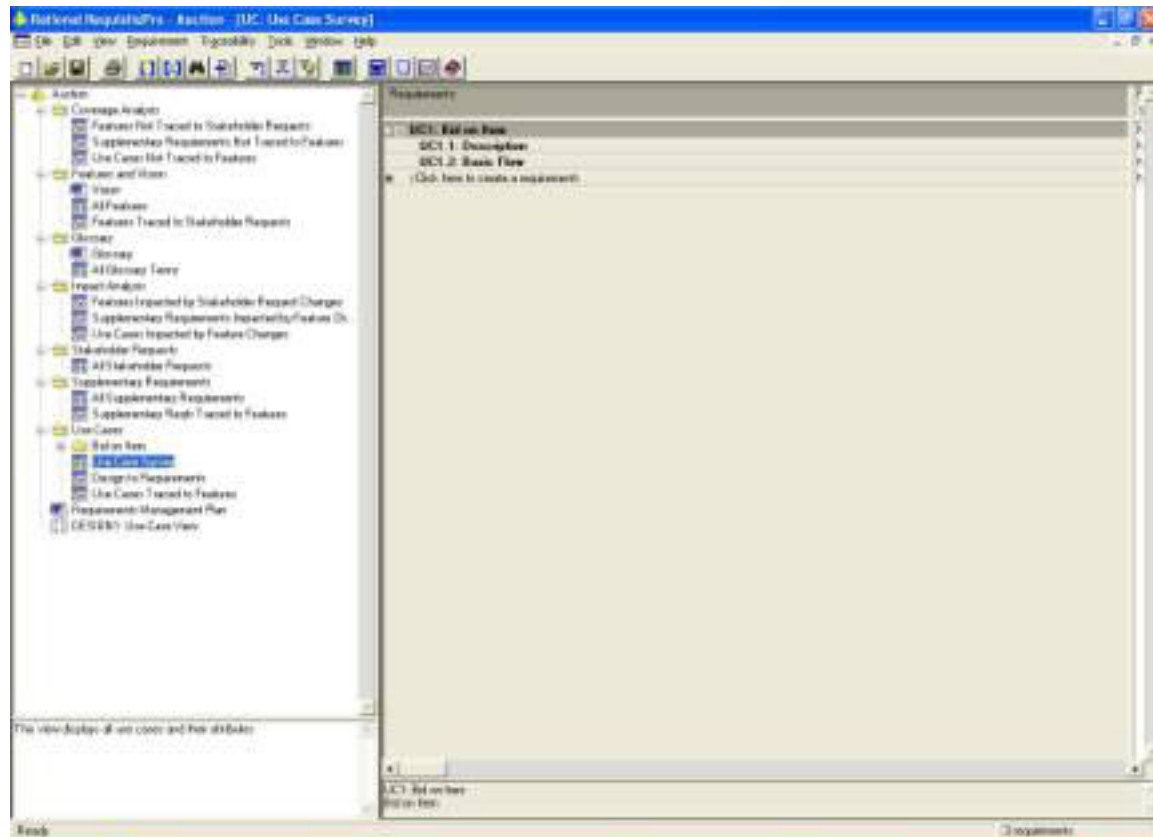


3. Enter `Auction` as the name of the project. Choose a directory and a database. (Use the built-in Microsoft Office Access database for this

example.) Then, provide a project description.

After your new project has been created, a window appears that looks something like the one shown in Figure 6. (all the items in the tree view on the left are expanded in a live RequisitePro project so you can see the layout: The view on the right will not be visible at this stage in the process.) Each folder in the tree on the left is called a *package*, and it logically groups documents and requirements to help you organize your project.

**Figure 6. The Rational RequisitePro project interface**



## Create requirement documents

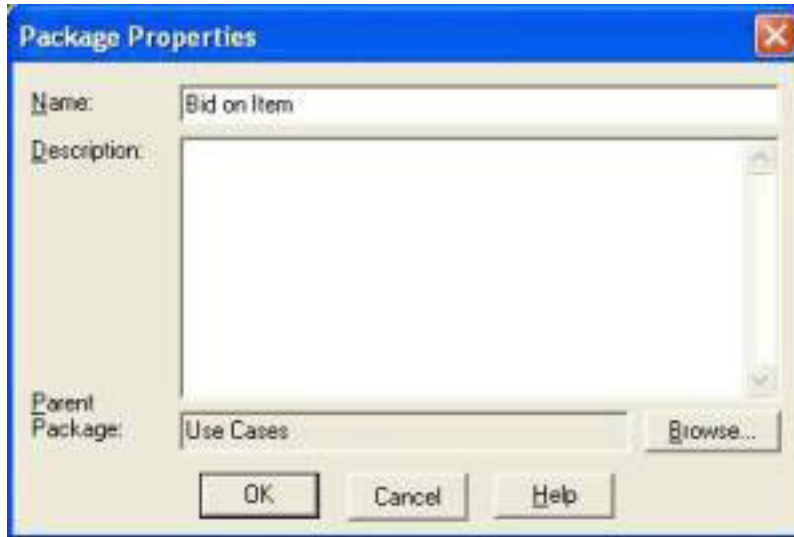
You can create requirements within Rational RequisitePro in three places: directly within Rational RequisitePro, through a Web interface, or by using a special Microsoft Word extension that allows you to create requirements directly in an Microsoft Word document and have it update the database in the background. The result is that you can create your Microsoft Word-based requirement documents, including any additional descriptive or encompassing text, while still benefiting from the database-led approach.

First, create a new package in which to hold the use case information for the Bid on

Item use case:

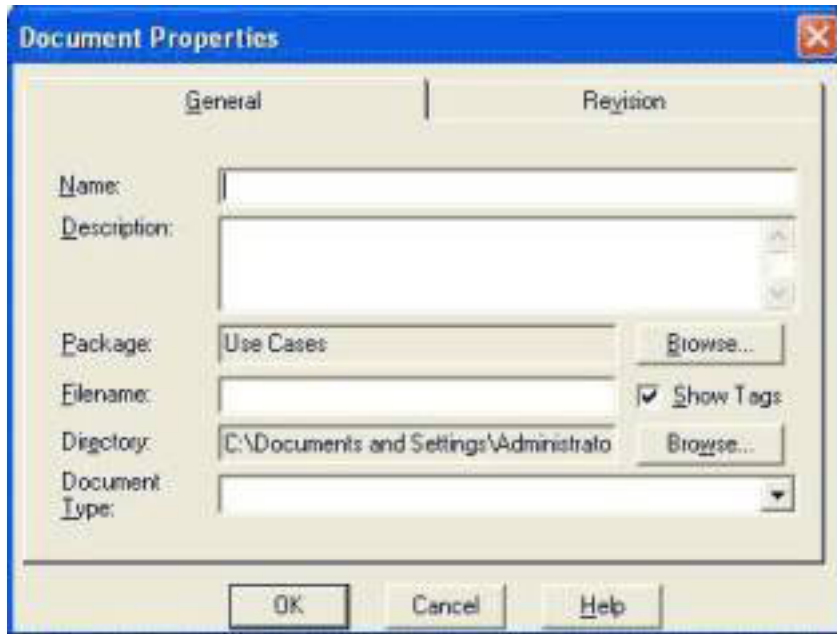
1. Right-click the Use Cases folder, then select **New Package**.
2. In the Package Properties window (see Figure 7), enter Bid on Item as the name of the package.

**Figure 7. Create a new use case package**



Now you have a package into which you can place the use case definition for the Bid on Item use case. You could import an existing document, but for this tutorial, you'll create a new document to be used to define the use cases for the Bid on Item portion of the auction system:

1. Within the Use Cases folder, right-click the Bid on Item folder, then select **New > Document**. The window shown in Figure 8 appears.  
**Figure 8. Create a new document**



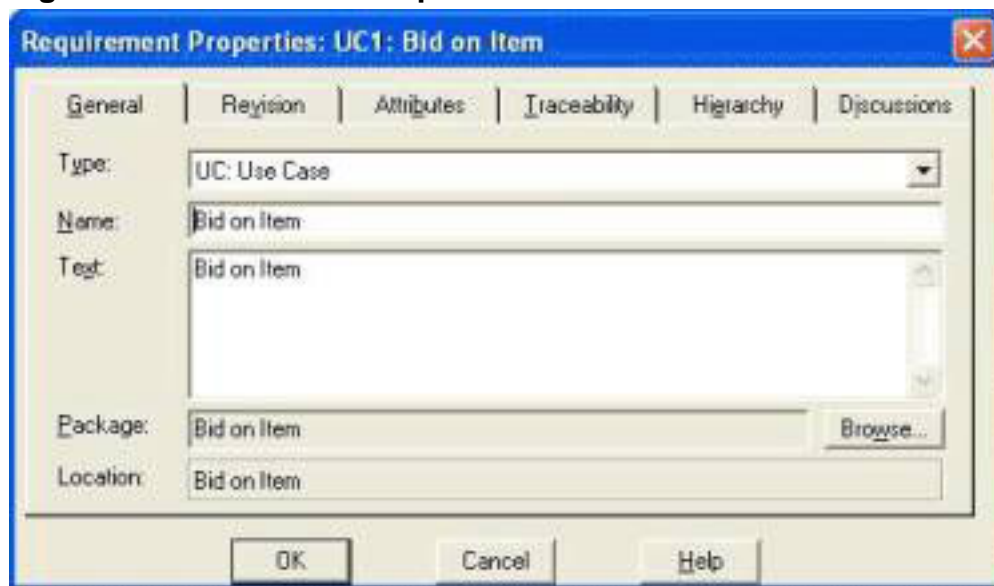
2. Enter `Bid on Item` as the name of the document as it will appear in the project, then provide a brief description of what it contains.
3. From the Document Type drop-down list, select the use case document to open a document based on a predefined template that matches the document template used in RUP. (Within Rational RequisitePro, certain Microsoft Word documents are available as Microsoft Word templates for use within the system. A window like that shown in Figure 9 appears in Microsoft Word.)

**Figure 9. A blank use case document in Microsoft Word**



1. Enter the text description for the use case directly into the Microsoft Word document. (Use the text *Bid on Item* for this example.)
2. Select the text you just typed into the document, then click **New Requirement** (the first square bracket ([]) button on the Rational RequisitePro toolbar).
3. In the Requirement Properties window (see Figure 11), specify the requirement type (Use Case), enter a name for the requirement, and ensure that the Package and Location information within the requirement properties are correct. (These last two properties should automatically be populated based on the location within the document you're editing.)

**Figure 11. Create a new requirement for the Bid on Item use case**



The document text you selected in Microsoft Word will be replaced with a field containing the text information just submitted into the Rational database. Note that the item will be marked as pending until you save and close the document in Microsoft Word. Part of the save process imports all the information into the Rational RequisitePro database.

Now, repeat this process to add more information within the Bid on Item use case that define the individual steps within the Bid on Item requirement.

## Extend the requirement information

In the previous panel, you created requirements with some basic information about the use case for entering a bid into the system. Now that you have the basic information, you can start appending other information to the requirements to make

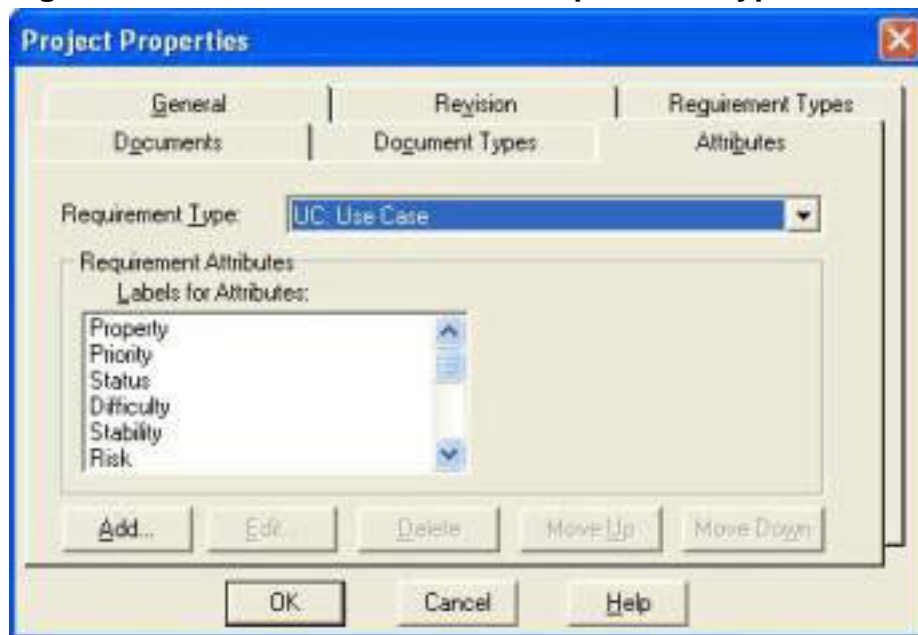
them more useful.

Requirement types within Rational RequisitePro are fully customizable. Standard fields are applied to the requirement types available in the predefined project types, such as the Use-Case template you're using here, but the actual fields are flexible and can be used to store all sorts of information. Additional fields can be free form or fixed type and can even use custom pop-up lists.

To help with this project, you're going to add a source field to the use case type so that you can identify who was originally responsible for a given requirement. To add a field to a requirement type:

1. Open the project properties.
2. Click the Attributes tab (see Figure 12). From the Requirement Type drop-down list, select **Use Case**. A list of the attributes currently defined for the type appears.

**Figure 12. Add new attributes to a requirement type**



3. Click **Add**. Attributes have a label (the field name you populate when you create a requirement), a type, and an optional list of values that's used within a pop-up menu or list. The type is a field type just as in a database; it can be an integer, a floating point, a text string, a date, a time, a URL, or a list (either single or multiple selection). Select **List (Single Value)**.
4. Populate the list (pressing **Return** to separate the values) with the information shown in Figure 13.

**Figure 13. Add attribute definitions**

You now have a property, or field, defined against the main Use Case requirement type. You can populate this property with information about who supplied the requirement in the first place.

## Views and reports

Views in Rational RequisitePro use standard query techniques on the database of requirements to show information and relationships. Each view consists of the query that generates the information to be displayed and the view type -- one of four predetermined formats for viewing the information. These view types are:

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

You'll use the traceability matrix later in this tutorial to map the Rational Software

Modeler model designs to your requirements.

---

## Section 5. Model applications with Rational Software Modeler

Rational Software Modeler provides a coherent tool for modeling your application before you start building the components and writing the code. Rational Software Modeler replaces Rational XDE and provides several significant enhancements over the interface and functionality that Rational XDE offered.

One key difference with Rational Software Modeler is that it has enhanced the functionality and integration with other Rational products by making more extensive use of the Eclipse 3.0 platform. Eclipse is the same application base that WebSphere Studio Application Developer uses (Rational XDE used a previous edition of Eclipse) and is becoming the standard environment for all IBM coding and development applications. Rational Software Modeler also works in a similar fashion to the new Rational Application Developer (also based on Eclipse) that you'll use in other parts of this tutorial series. Through Eclipse, Rational Software Modeler has access to many of the advanced toolsets and environments that make working with different types of development data, such as models and class structures, so easy.

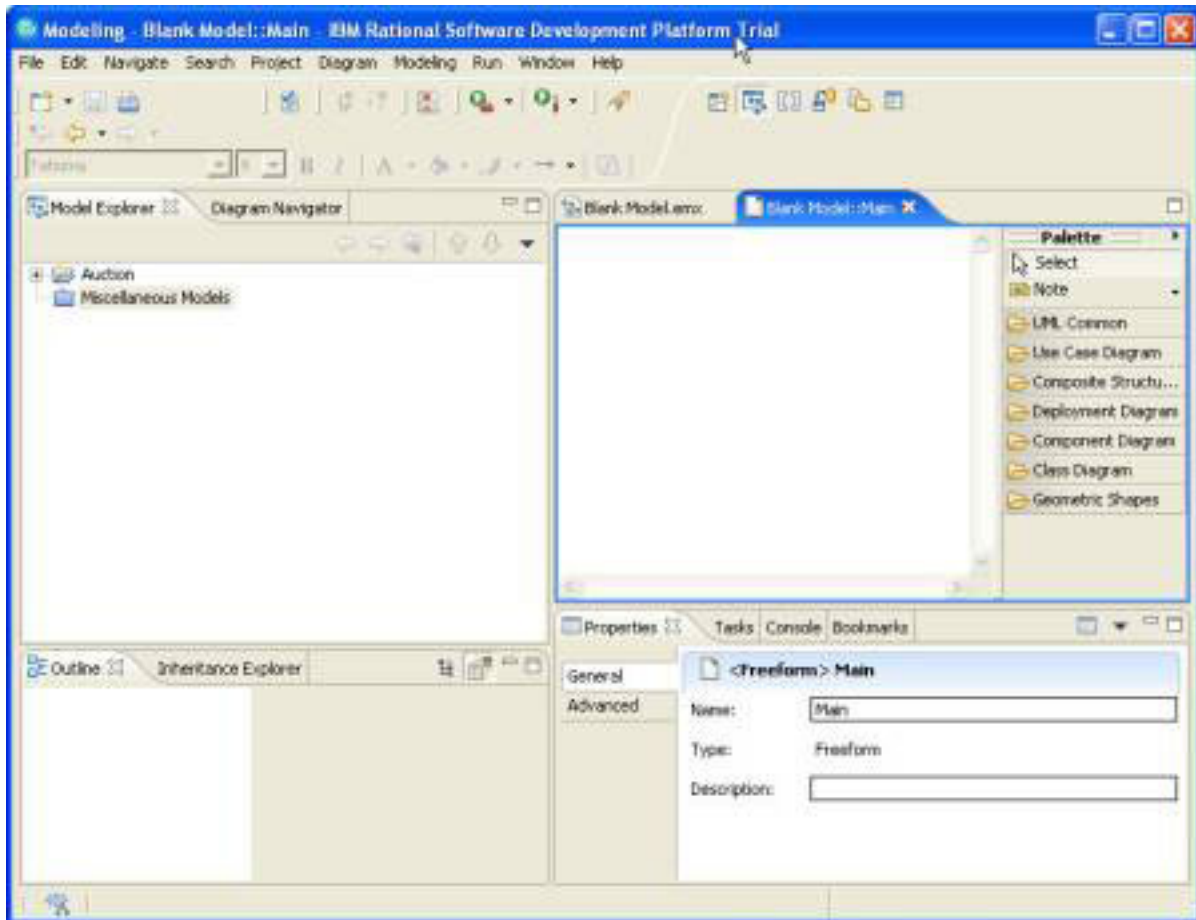
For example, Rational Software Modeler (and Rational Application Developer) uses the perspectives functionality in Eclipse to provide integration with other components and tools in the Rational Suite. This tutorial focuses on the integration with Rational RequisitePro, and a Requirements perspective is available for that purpose.

Let's start with an overview of Rational Software Modeler.

### Rational Software Modeler overview

Figure 14 shows Rational Software Modeler, here with a simple empty model project already open.

#### **Figure 14. Rational Software Modeler interface**



The panel on the left is the Model Explorer, which enables you to find and select models that exist in your project. Models, their components, documentation, and other information are organized in a combination of folders and documents in this tree. The larger panel on the top right is used as the workspace for creating and manipulating models; you can see a palette on the far right that contains the various types of Unified Modeling Language (UML) diagrams that you can add to your model.

The two panels on the bottom of the window are generally used for additional information or related entities. In this case, the panel on the left holds an outline of the model; the panel on the right shows details of the properties for the current diagram.

Each panel is capable of displaying the type of information relevant to the view you have selected; you can see additional tabs within each panel that take you to different areas and information details. The selection of panels shown here is a collective known as the *perspective* -- that is, the specific view of information, data, and trees used to describe a project. Many different perspectives are available, and you can create and customize your own. Perspectives are an easy way to change the selection of panels and information displayed while remaining in the context of

the same project.

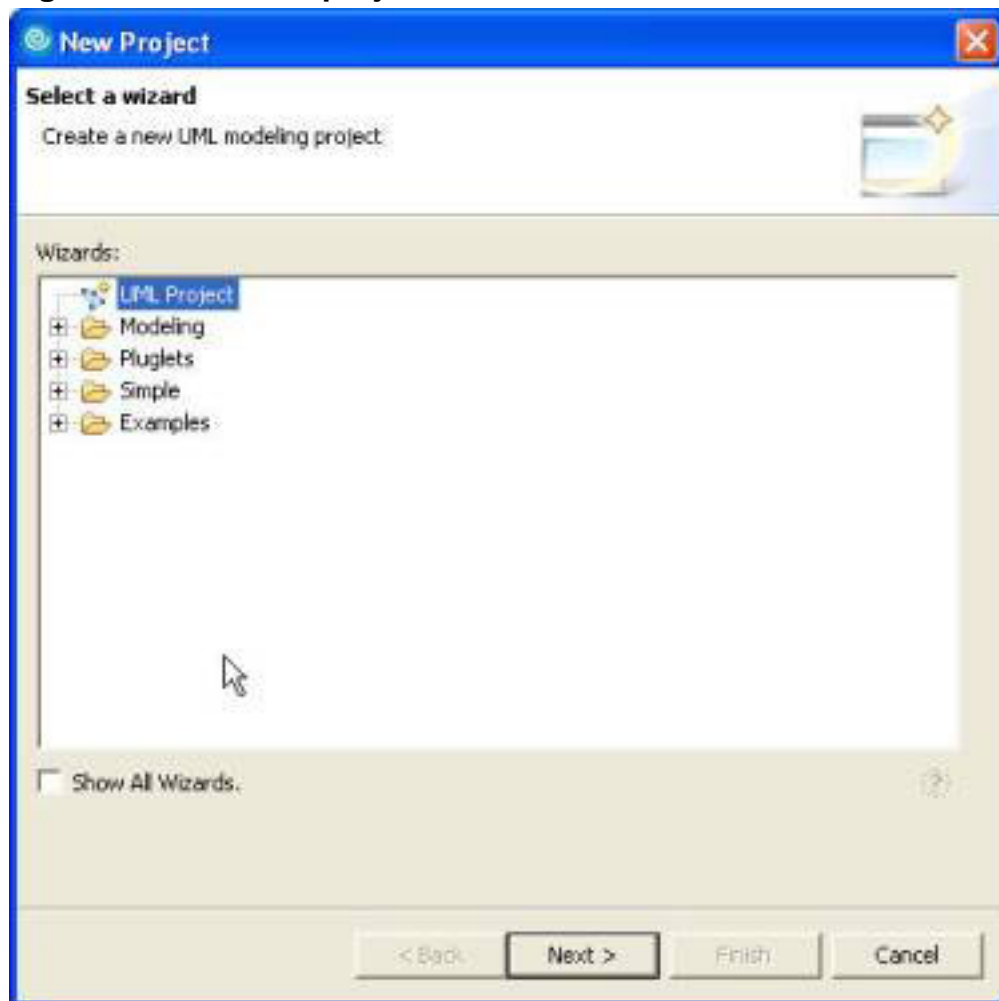
Now you can create a new project that will contain your models.

## Create a new UML project

Create a new project to contain your models:

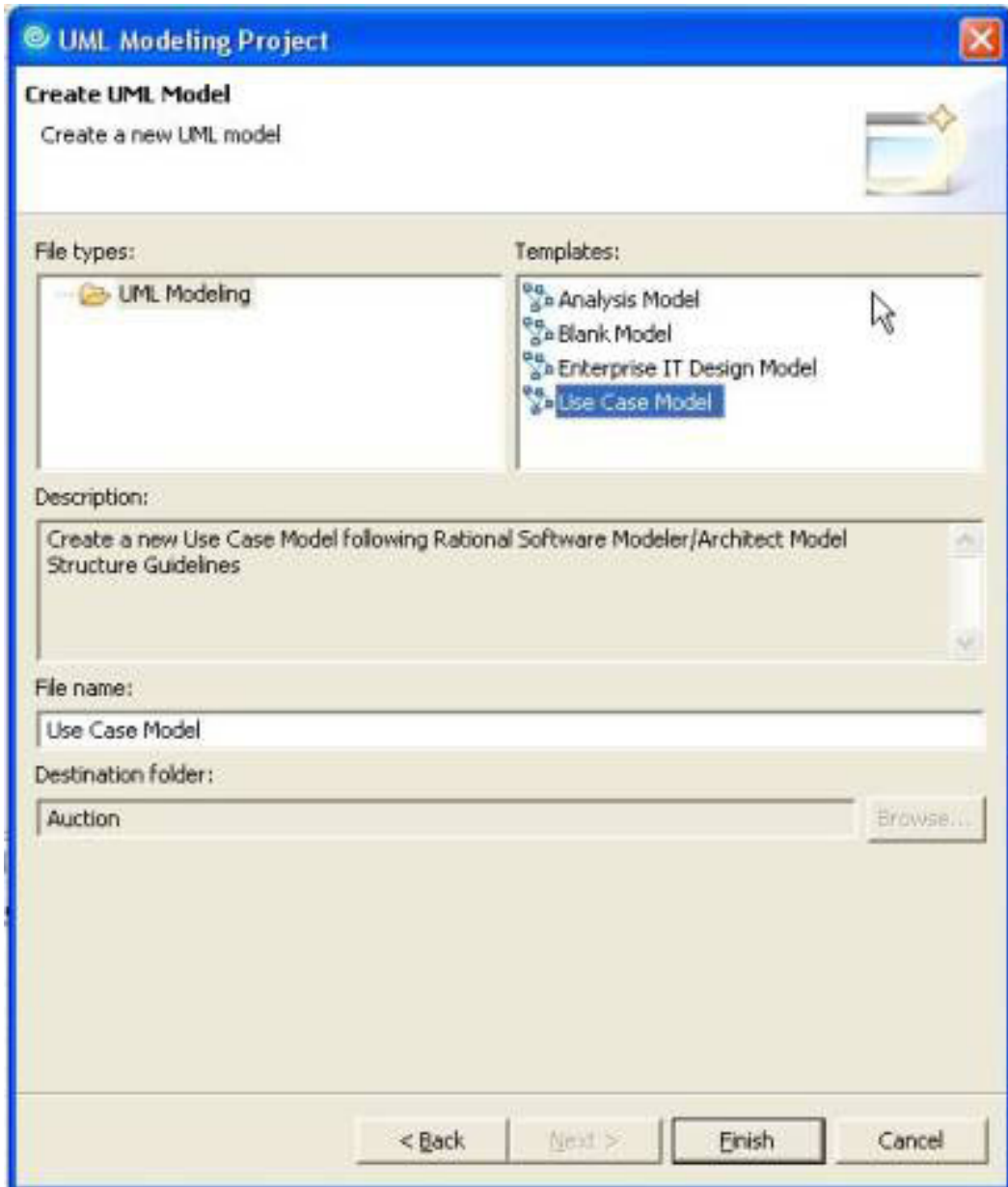
1. Open Rational Software Modeler.
2. Select **File > New > Project**.
3. In the New Project window (see Figure 15), select **UML Project**. Click **Next**.

**Figure 15. Choose a project wizard**



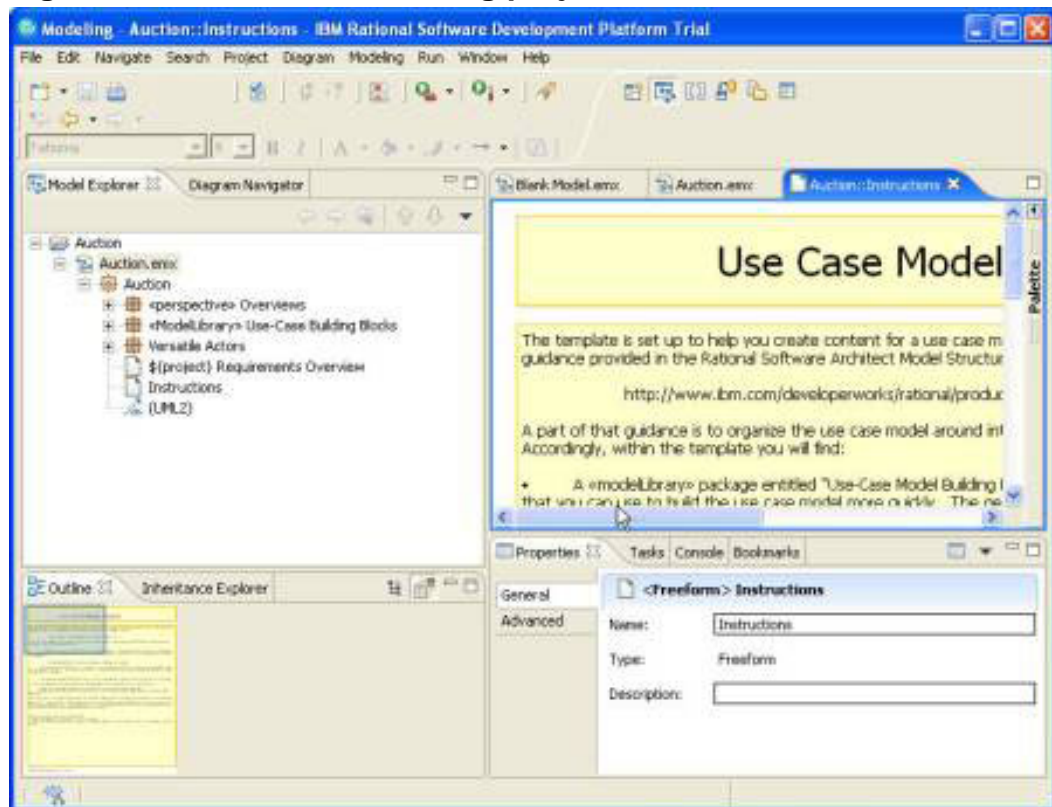
4. In the UML Modeling Project window, enter `Auction` as the name. Click **Next**.
5. On the final window in the wizard (see Figure 16), select **Use Case Model** from the Templates panel to automatically create a single UML model in the project.

**Figure 16. Create a UML model from the Use Case Model template**



6. Enter a name in the File name text box. Click **Finish**. You should end up with a window like the one shown in Figure 17.

**Figure 17. A bare UML modeling project**



As you can see, you have a basic structure for a use case project. Close the Instructions by clicking the **X** on the tab so that you have the main Auction use case.

Now you can begin populating the model using the information generated in the previous sections on Rational RequisitePro.

## Create a new use case diagram

To model a use case, you need to create a new diagram that will hold the details of the model. To create a new diagram:

1. Right-click the Auction Model, then select **Add Diagram > Use Case Diagram**. A new diagram appears in the panel on the right.
2. In the Model Explorer, change the name of the diagram to *Bid On Item*.

You're now ready to start adding items to the diagram. Remember that the main reason you created the requirements and use case statements in Rational RequisitePro was that you wanted to capture the features and functionality of the application before you started building it. The secondary but more long-term aim was

to use these requirements as a method of tracking and tracing the development of the application.

So, you need to create a diagram of your use case. To do that, you need to integrate Rational RequisitePro and Rational Software Modeler.

---

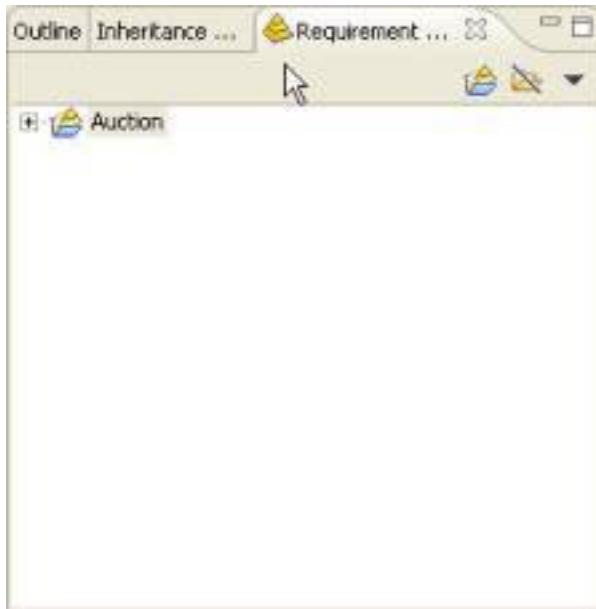
## Section 6. Accessing your requirements

### Connect to a requirements project

To create an entity linked to your requirements within the UML model of Rational Software Modeler, you need to open the Rational RequisitePro project from within Rational Software Modeler. You could do so in several ways, including changing to a different perspective. The default Requirements perspective, for example, provides a complete environment for viewing and working with requirements.

However, because you want to continue generating your use case diagram, all you want to do is open the Requirements project and access the use case requirement that will be related to this diagram. To do so, add the Requirement Explorer view to the current perspective in Rational Software Modeler by selecting **Window > Show View > Requirement Explorer** to create a Requirement Explorer view (see Figure 18). You can move this view to a different panel or create a new panel for it by dragging the view title to its new location.

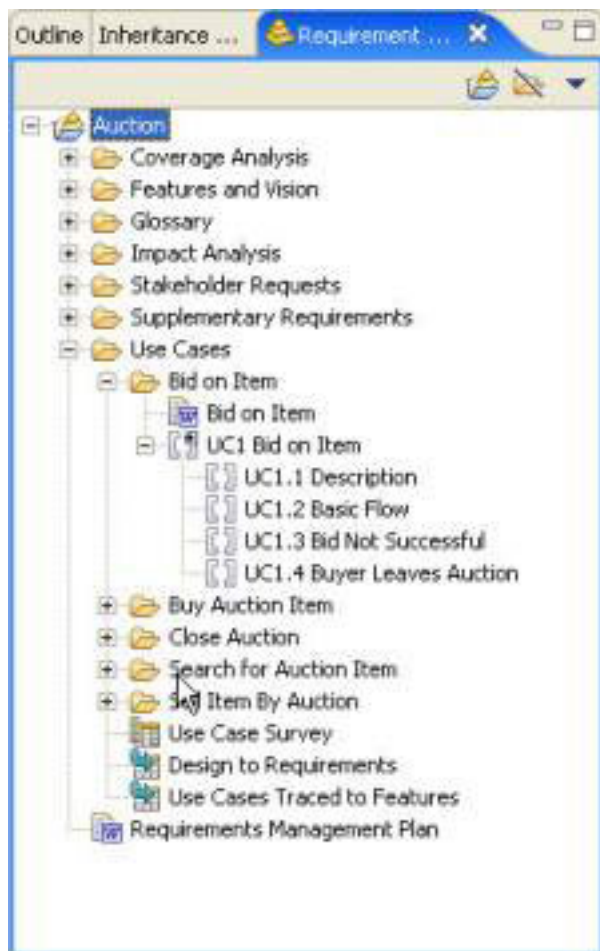
#### **Figure 18. Add the Requirement Explorer in Rational Software Modeler**



You now have a view through which you can browse requirements as generated through a Rational RequisitePro project. If you've already opened a Rational RequisitePro project, as I have, it will appear in this view. To open a project that isn't listed or that you haven't already used, click the **Open Project** icon; in the window that appears, select the Rational RequisitePro project file to open. You might be prompted to supply a user name and password to open the document.

You should now have a tree display of your requirements from the Rational RequisitePro project, as shown in Figure 19. Note that this is a slightly expanded view of the of the requirements you created earlier in this tutorial.

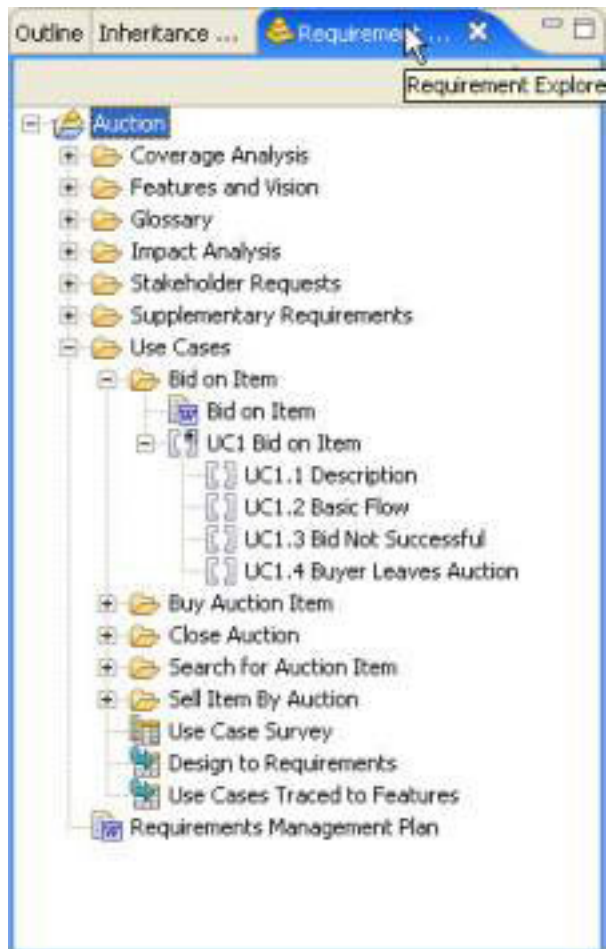
**Figure 19. An open requirements project**



## Browsing requirements

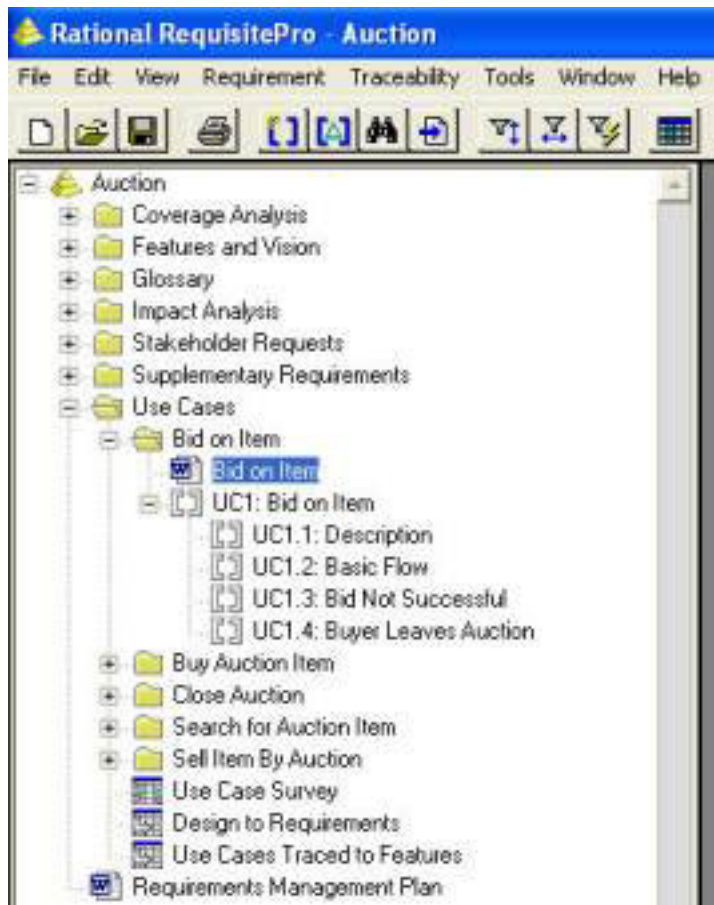
Look at the Requirement Explorer again, shown here in Figure 20.

**Figure 20. The Requirements Explorer**



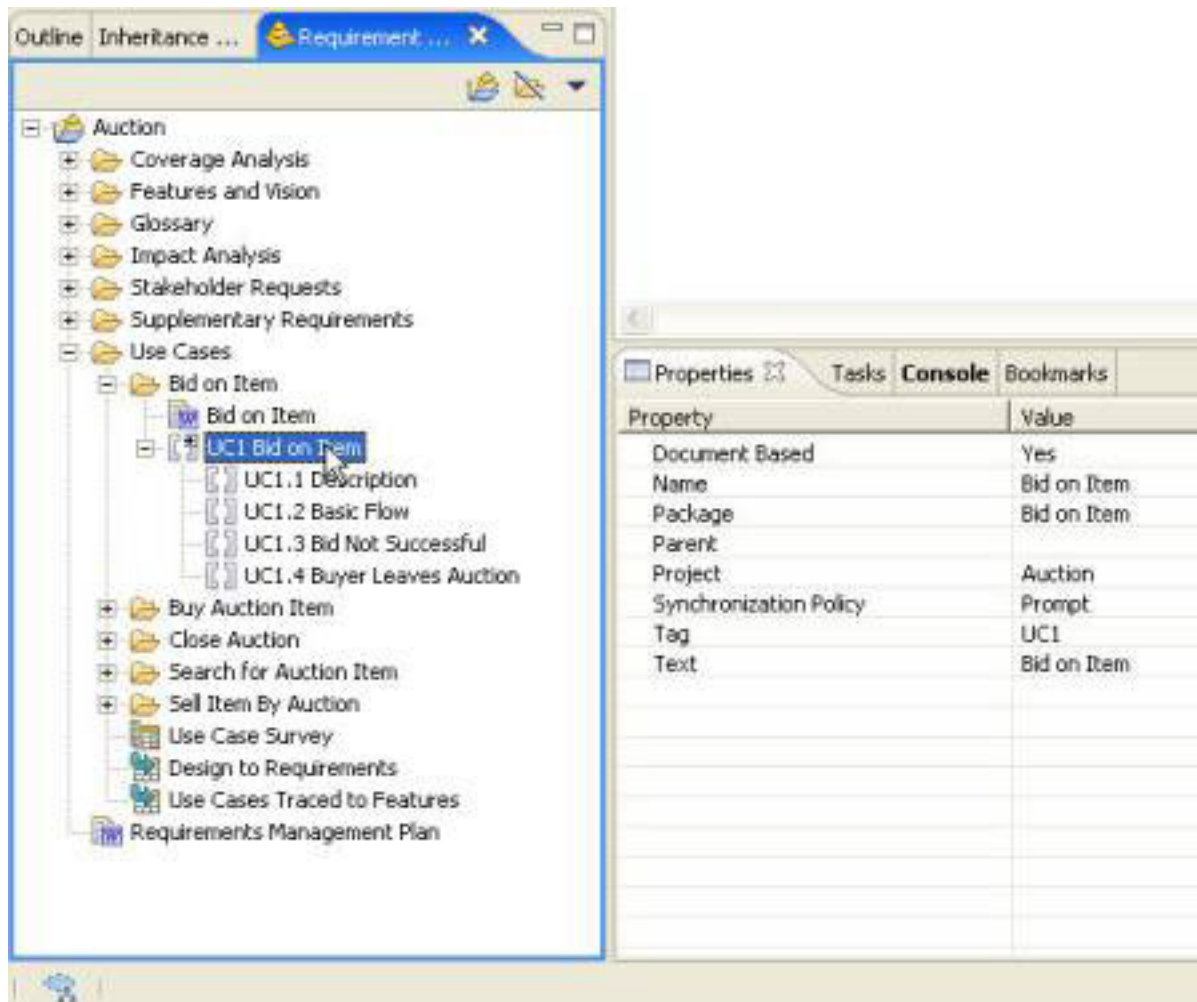
The basic layout replicates the same tree in Rational RequisitePro. The integration is extensive in that not only does it contain the requirements details, but also the views and organization of requirements in Rational RequisitePro are replicated here. You can compare this integration by looking at Figure 21, which shows the same tree in Rational RequisitePro.

**Figure 21. Viewing requirements in Rational RequisitePro**



If you right-click a requirement from the tree in the Requirement Explorer in Rational Software Architect, then select **Properties**, you can view requirement properties such as parent, identity tag, and the text used to populate the requirement (see Figure 22).

**Figure 22. View requirement details**



## Create a use case diagram based on a requirement

In the previous section, you learned how to manually create a use case diagram for use with your UML model. You now need to add the Bid On Item requirement as an element of that use case diagram.

To add a requirement to the diagram:

1. Select the **Bid On Item** diagram so that the blank diagram is open.
2. Select the requirement you want to add to this model. For this example, you're creating the use case for the Bid On Item within your diagram.
3. Drag the Bid On Item use case from the Requirements Explorer to the diagram. Doing so generates a new use case within the diagram based on the Bid on Item use case, which is in turn within the use case model

for the Auction project. You should see a basic diagram as shown in Figure 23.

**Figure 23. Create a new use case diagram in the Auction model**



To complete the diagram, add the two actors that relate to this use case -- the Buyer and the Seller -- then create the relationships between them and the use case. To add the actors:

1. On the Palette, click **Actor**.
2. Click once on one side of the diagram to create a new actor, then rename it Buyer.
3. Repeat step 2 to create a new actor named Seller.

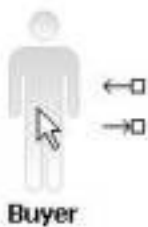
You should now have a diagram like the one shown in Figure 24.

**Figure 24. Use case diagram with actors**



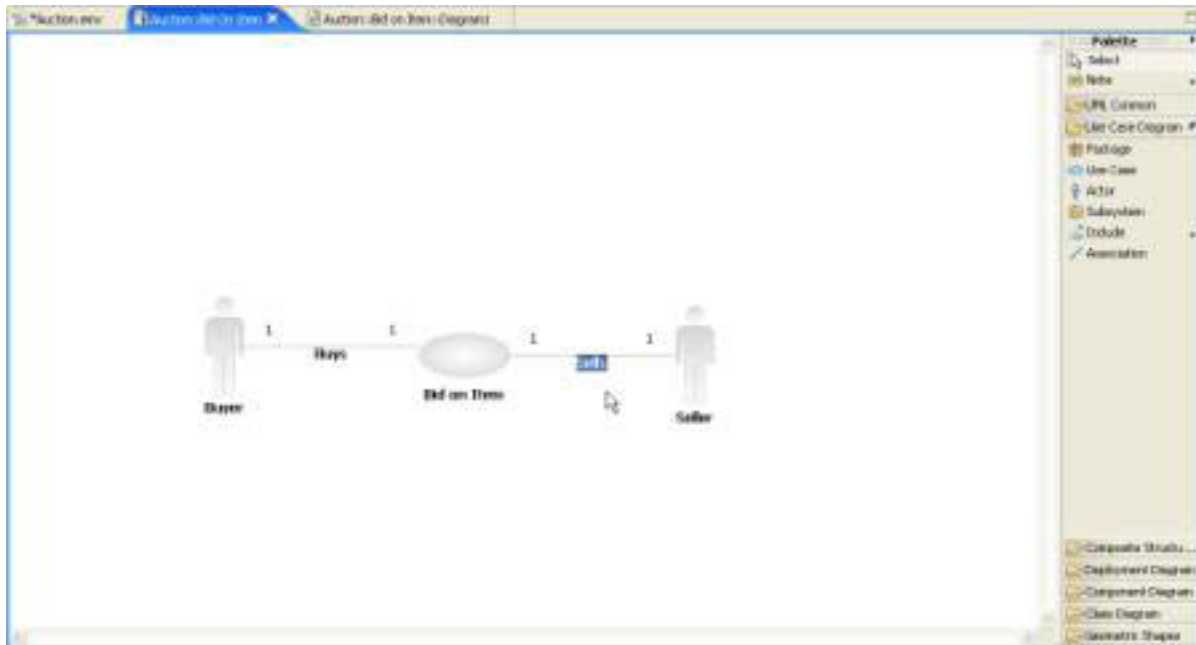
You now need to define the relationship between the actors and the use case. In the main diagram, the Buyer has a relationship to the Bid on Item use case, which in turn has a relationship to the Seller. When you hover over an entity in the diagram, two icons are displayed with the entity (see Figure 25): These are automatic connections for connecting to or from the entity. Drag the From handle from the Buyer to the Bid On Item use case, then again from the Bid On Item use case to the Seller.

**Figure 25. Entity connectors**



The result should be the diagram shown in Figure 26 -- a final use case diagram for the top-level Bid On Item use case from your requirements. (I've added the descriptions to the relationship in this example).

**Figure 26. The final use case diagram**



You'll probably want to track other elements within Rational RequisitePro -- for example, the class diagrams you'll creating in part 2 of this tutorial series to help define the application.

## Tracking the relationships

The resulting diagram is simplistic, but you now have a relationship between your requirements and the use case model that your developers will use to start building classes and coding the application.

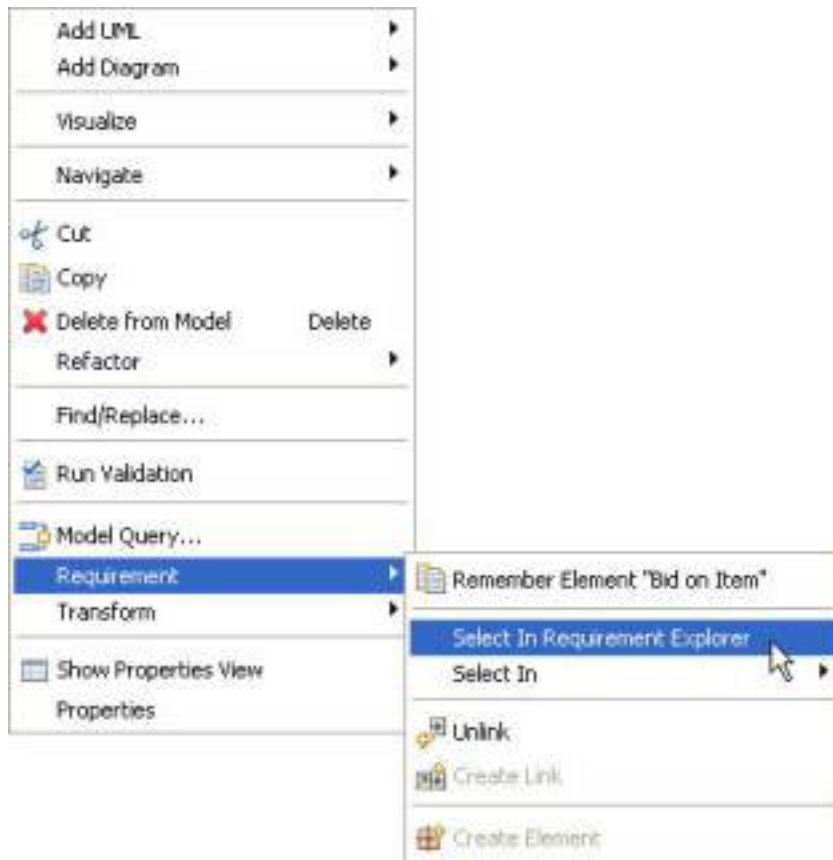
You can always tell an item that has a related element in Rational Software Modeler by looking at the icon for the element within the Explorer views. Entities with a relationship have a little arrow embedded with their icon, as shown in Figure 27.

### Figure 27. Traced entities



You can also right-click an entity and then, through the menus, select the related item and display its properties. For example, if you right-click the Bid On Item entity, you can choose to view the requirement in the Requirement Explorer, which will also open the requirement's properties (see Figure 28).

### Figure 28. Trace a requirement link



By tracing the relationship, it becomes easy to find related items and, longer term, it becomes easier to identify requirements and the change requests and bugs related to them.

## Tracking other model elements

Up to now, you've been marrying use case models in Rational Software Modeler with use cases in Rational RequisitePro. As you work through the development process, you generate other elements that you need to relate to the requirements and the Rational RequisitePro project driving the development effort.

These entities are normally tracked within Rational RequisitePro as design elements -- a different requirement type, but one that you can ultimately track back to the original application requirements. After these elements are both within Rational RequisitePro, you can use a traceability matrix to ensure that the requirements have an associated implementation element.

To learn how to create a link between items in Rational Software Modeler and Rational RequisitePro, create a simple class diagram:

1. Right-click the Bid On Item use case in the Explorer, then select **Add Diagram > Class Diagram**.
2. In the newly created class diagram, click **Class** from the Palette, then click anywhere in the diagram to create a new class.
3. Enter `Item` as the name of the class.

With this entity in place, you can create a new requirement in Rational RequisitePro that is linked to this class. To create the link and the requirement:

1. Right-click the Item class from the class diagram or in Model Explorer, then select **Requirement > Remember Element "Item"**.
2. In Requirement Explorer, right-click a package for the new requirement, then select **Create Requirement from "Item"**.

As an alternative, you can simply drag and drop the entity from the Model Explorer to the Requirements Explorer. The type of requirement created is determined by the properties of the project within the Requirement Creation Policy section of the RSM preferences. The default is to create a CLASS requirement for Class entities. (Note that this step does not create a relationship between the new requirement and the original use case.) To associate a given entity with its original use case, drag the entity from the Model Explorer over the use case requirement.

The design elements and the original requirements are in place. Now you can use a traceability matrix to determine which parts of the requirements have actually been designed.

## The traceability matrix for design and requirements

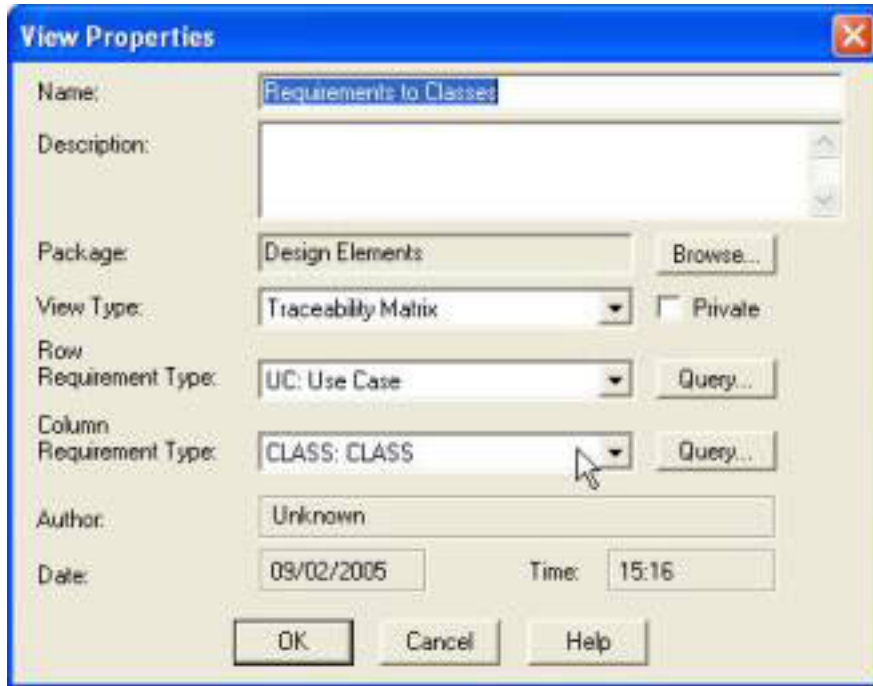
The final part of the integration is to monitor the traceability of your design components with the original requirements, which you do by creating a traceability matrix within Rational RequisitePro. This process creates a view that maps between two requirement types and shows which design component relates to which requirement.

To create a traceability matrix in Rational RequisitePro:

1. Right-click the Use Cases package, then select **New > View**.
2. In the View Properties window, enter `Requirements to Classes` as the view name.

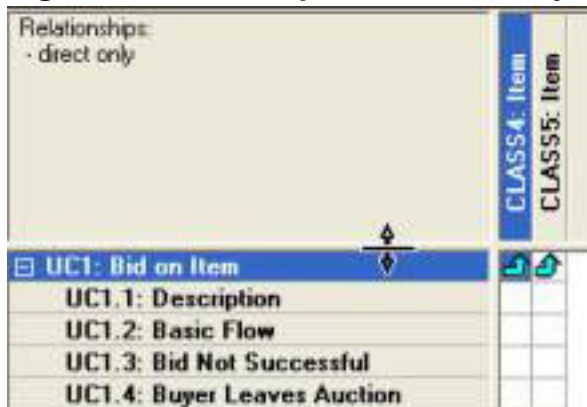
- From the View Type drop-down list, select **Traceability Matrix** (see figure 29).

**Figure 29. Add a new view**



- From the Row Requirement type drop-down list, select **Use Case**.
- From the Column Requirement Type drop-down list, select **CLASS**. (In a very large project, you could limit the view to specific use cases by any of its properties by clicking **Query** to specify which requirements to include in the view.)
- Click **OK** to complete the view. You should see a diagram like the one shown in Figure 30.

**Figure 30. The completed traceability matrix**



The arrows show the direction of the relationship from the requirement to the design model, and you can see at a glance which components complete the functionality of your requirements.

You've now created a basic link between the original requirements and the model you're building to describe the application in preparation for development. Continue the process, adding more requirements and attaching other models and elements to those requirements, until you get a full picture of the system.

---

## Section 7. Summary

Rational RequisitePro is an excellent tool for recording requirements for an application. Rational Software Modeler is an excellent tool for modeling these requirements into use case diagrams. On their own, each product fulfills its part of the development process. Together, however, they provide an excellent way for developers and designers to communicate with project leads and ultimately customers to identify and build the application that the client requested.

When the integration is up and running, you can see through the traceability matrix which components are modeled and complete and which parts of the requirements need additional work to build and model the application. Without this level of integration, the development of the application runs the risk of straying away from the original specification for the application.

This tutorial is only the first part of the series. You still have work to do as you continue to build the Auction system. For the moment, you have a list of requirements and the associated UML models in Rational RequisitePro and Rational Software Modeler. As you continue to work through building the Auction application in future parts of this series, you'll see in more detail how to turn your model into a working design and code through Rational Application Developer. You'll also learn how to track defects and trace the defects and other changes requests back into IBM Rational RequisitePro. Lastly, you will learn about the Rational testing products and how they integrate with other components in the Rational suite.

# Resources

## Learn

- [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.
- [Rational TUP home page](#). This resource contains more information about the Rational TUP.
- [Eclipse project](#). Eclipse is an open source IDE that forms the basis of many IBM products, including Rational Software Modeler and Rational Application Developer.
- [developerWorks Rational zone](#). Learn more about Rational. You'll find technical documentation, how-to articles, downloads, product information, and more.
- [Visit the Developer Bookstore](#) for a comprehensive listing of technical books.
- [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.x
- [Rational Application Developer](#) Version 6.0
- [Rational Software Modeler](#) Version 6.0
- [RUP](#)

## Discuss

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

## About the author

Martin 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. 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, is a core member of the AnswerSquad.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>.