

# Improved application development: Part 1, Translating requirements into an application model

Skill Level: Introductory

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

16 Aug 2004

Developing applications using the Rational Unified Process is a lot easier if you have the right tools to help you throughout the entire process. The Rational Toolset provides a range of tools that can be employed to help produce the application in different phases of the development process. On their own, these tools provide excellent support for each phase of the Rational discipline. But when used together the tools are even more powerful. By sharing the information you can track components in the application from their original requirement specification through to testing and release. In this first part of a five part series you'll learn about the integration between RequisitePro and Rational XDE.

## Section 1. Before you start

### About this tutorial

Developing applications is a complicated process that can be made easier by using tools from the Rational development suite and following the Rational Unified Process (RUP). These include tools for defining the requirements (IBM Rational RequisitePro), visually modeling the application (IBM Rational XDE) and tools for tracking defects (IBM Rational ClearQuest) and managing changes to the application (IBM Rational ClearCase). To make the process even easier, you can combine the functionality of these individual applications into a single, cohesive, process. Each product has methods for communicating with the other to obtain, update and integrate the necessary information.

In this five-part tutorial series, we're going to look at 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 where bugs and faults need to be traced and tracked through the system after the initial release.

The focus in the first part of the series, is the first stage of the process -- project inception. You're going to be using the Auction system, used as a sample development within WebSphere Studio Application Developer, as the basis of an example. See [Prerequisites](#) for more information on how to obtain the auction code and some tutorials for how to produce the system within WebSphere.

The first stage of any development effort is to create the requirements specification that defines the scope of the project. These requirements come from many different places and managing all of this information and merging it together is where RequisitePro fits into the equation. Once you've developed your unified list of requirements, you then need to be able to model your application based on those requirements.

The Rational XDE products provide the methods for application modeling and you can use Rational XDE in combination with RequisitePro to build your application model. The integration between the two products enables you to match application requirements with the components in your model and vice versa, making it easy for all the people involved in the project to understand the approach being used.

This tutorial demonstrates the basics of RequisitePro and Rational XDE and how the two can be linked together to provide an overall picture of requirements for your application, the application model they produce, and how you can monitor changes to requirements and components into the final system.

Key parts of the tutorial include:

- Introduction to the development process
- Developing the Requirements Spec using RequisitePro
- Documenting the project requirements through RequisitePro and Word
- Translating the requirements into an Application Model through XDE
- Tracing model elements back to the Requirements

## Should I take this tutorial?

This tutorial series is primarily targeted for project managers, program managers, and developers. Others in the software development business may also find it useful. In each of the tutorials in this series, you will learn about different Rational tools and the part they play in the software development process. Different team

members might be interested in some tutorials more than others, but it is worthwhile to take the complete series from beginning to end to see how all the products work together to form a cohesive process.

This first tutorial focuses on the requirements phase of software development. Anyone involved in gathering and documenting requirements and converting those requirements into models will be most interested in the content covered in this tutorial.

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

## Prerequisites

The focus of this tutorial is on two products in the Rational Suite: RequisitePro and Rational XDE Developer (for Java). Links to download trial copies of these products are below. You will also need Microsoft Word, and optionally Microsoft Project. You might also want to download the Rational Unified Process package, which includes complete documentation on the techniques, methods and systems behind the Rational Unified Process.

[RequisitePro trial downloads](#)

[Rational XDE Developer trial downloads](#)

[Rational Unified Process](#)

Note that you must have the same releases of the tools in order for the integration to work. V2003.06.12 of all the various tools is used in this series.

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

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

You can also find the code itself supplied as the example with the WebSphere Studio Application Developer, a tool you'll actually be using later in the series. See [The Auction application](#) for more details of the auction system and where to obtain the code.

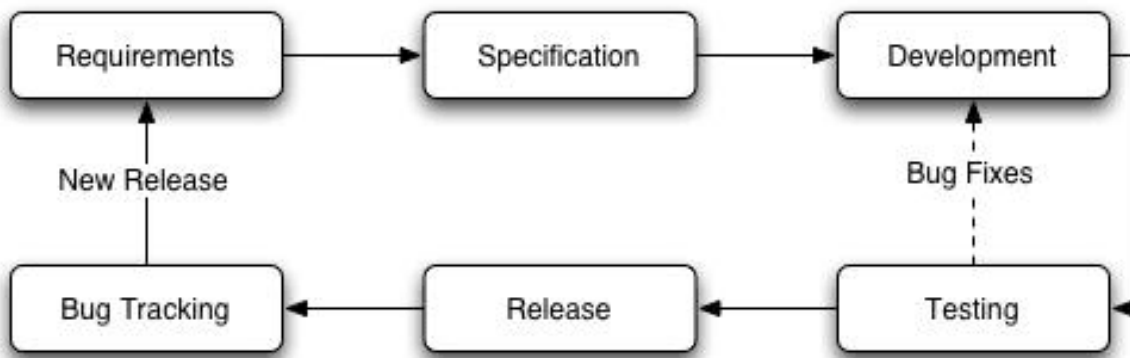
## Section 2. Develop apps using Rational integrated tools

### Application development process

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. Once you've started to test and deploy your application, change requests and defect tracking add to this complexity. You'll start to 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 the original requirements.

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

**Figure 1: A basic development model**



If you look at this diagram, the inception of a new project starts with the definition of the Requirements. Once 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 event of a bug, before finally releasing the software. Once 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 starts all over again.

The Rational Unified Process applies a similar, simple model to the development process. Let's have a look at the Rational model, and then how you apply the Rational tools to the process.

## Developing with the Rational Unified Process

Developing applications using the Rational Unified Process centers around the same basic principles outlined in the previously, but applies four 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 only developing part of the application each time.
- **Object oriented** -- 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.
- **Manage and control** -- create logs and a traceable sequence that enable you to trace features from the original requirement to the component in the application.

The process can work in a completely manual fashion. However, the process also lends itself to being automated. This is the role of the Rational Suite - providing a range of tools and applications to simplify and automate the process. Using these tools also simplifies the process for the teams and individuals working in your development department to develop applications quickly and easily, while following the previous three features.

The result is a development cycle that looks like the one shown in Figure 2.

**Figure 2: The Rational Unified Process Model**

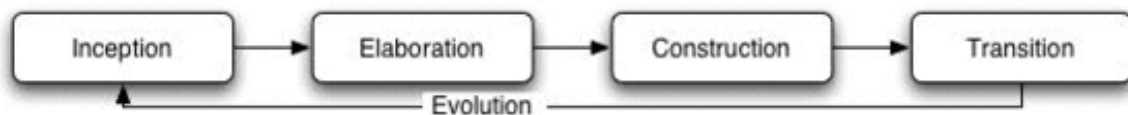


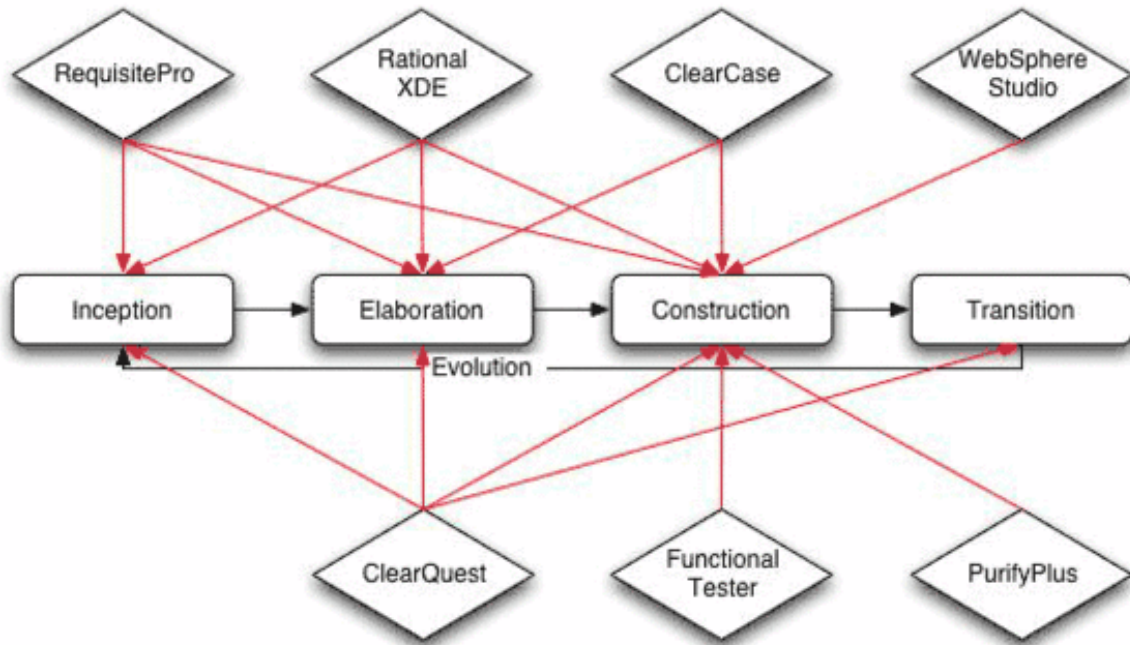
Figure 2 follows the same basic sequence as described previously. Start with the inception of the project, defining the features and requirements, the elaboration that expands the requirements and defines the model, the construction that builds the model and generates code and finally the transition where you start to track defects.

The automation of the process relies on a number of Rational applications. In the next section you'll see where in the process these tools fit into the equation.

## Developing with the Rational Toolset

Rational software includes a variety of tools that enable you to develop an application using the principles of the Rational Unified Process (RUP). You can attach specific tools to different parts of the process according to Figure 3.

**Figure 3: Attaching Rational Tools to the Rational Unified Process**



In more detail, the packages mentioned in figure 3 cover the following tasks:

- RequisitePro -- 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 XDE -- Application modeling, using UML and the model driven architecture. Using Rational XDE, it's possible to, manually, turn the requirements from RequisitePro into a model within XDE, and then build the corresponding classes and code that can be used to build the application.
- WebSphere Studio Application Developer (WSAD) -- Application development. This integrates directly with the Rational XDE to enable you to customize and extend the functionality of your application, based on the model, and ultimately the requirements, which you developed in the first phase.

- ClearCase -- Configuration Management of your code base and models. WSAD and XDE can interface with ClearCase to provide management across the whole development process for your entire team.
- ClearQuest -- 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 include code, models and other information. ClearQuest integrates with the other applications so that you can track defects and activity across components, requirements and follow the information from component to component.
- PurifyPlus -- Runtime 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 toolset. This integrates with WSAD to provide information on the elements of your application that are causing problems.
- Functional Tester -- Function-based testing of an application to ensure that the application achieves its original goal. This integrates with WSAD and other tools to identify and track problems and their resolution.
- Team Unifying Platform (TUP) -- is a collection of the core tools (RequisitePro, ClearCase, ClearQuest, TestManager) and a new tool called ProjectConsole. ProjectConsole aggregates status and metrics information from all of the tools, including Rational XDE and others, and provides a central location for the viewing and dissemination of status information within across a development team.

The primary purpose of this series is to look at the integration of these different components when building an application and how the integration can simplify the process of developing applications. We'll also take the opportunity to look at how the tools can be used by teams to develop applications across a department, entire company or enterprise.

## The Auction application

The IBM Online Auction sample is used throughout this series as an example application. The Auction system is included as the sample application with WebSphere Studio Application Developer. The best way to get the system is to install a copy of WSAD and then follow 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 [Prerequisites](#) for more information.

Because the auction application is one you might be familiar with from using WebSphere Studio Application Developer, you can concentrate less on the code for the application and more on the integration of the different tools you'll be using in this tutorial. For that reason, on occasion, you might look at an alternative example to demonstrate a particular piece of functionality, although the overall aim is to document the process of building the Online Auction system.

For those not familiar with the application, the Online Auction is a Web application that employs J2EE technology using Java beans, JDBC, HTML, JavaScript, JSP pages and other components. The result is an auction system that is functionally very similar to Ebay, Yahoo, Amazon and other auction sites and systems. Users can place items for sale and other users can bid for the items and eventually win and pay for them.

---

## Section 3. Specify requirements with RequisitePro

### Requirements and 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 your auction system such as: "Allows you to buy and sell goods through an auction-like system."

But this really wouldn't be all 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. The primary sources for requirements are 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 producing the initial list of requirements that will be used to start developing the application. This is what RequisitePro does, it provides a mechanism for recording all the various requirements from the different groups, then building a final list of the requirements and their inter-relationships.

These requirements can be specified and defined in a number of different ways, including basic statements and Use Cases. The Online Auction system is completely

freeform, and individual requirements can have an unlimited number of properties (fields) which are used to store additional information about each requirement.

## Driving development with requirements

Once the requirements have been collated they should be used to help drive the development of the application. In theory, requirements should be used to drive all software development. In reality, the initial set 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, and then using the requirements to design the model, and code application components. This is precisely what you aim to solve by using the integration features between RequisitePro and Rational XDE.

By creating a formal link between the model (and code) and the requirement that drove 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 stakeholders in the project.

Requirements form the basis of all the development and by keeping these up to date with the current model you can address issues and questions like:

- 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?

These are all addressed by mapping requirements and model components together.

Let's have a quick look at how you will be defining the requirements for your auction project.

## Use case recap

Integration between Rational XDE and RequisitePro links models with requirements. The RUP is use case driven, and we'll be using that approach in our applications. Theoretically it's possible (but not advised) to use it 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.

If you are already familiar with use case, then you can probably skip this section. If you want a recap on the structure of use cases and how they drive development, read on.

A use case captures the agreement between the stakeholders of a project or system about its exact behavior. The use case describes the behavior of the system under a number of different situations. Within each situation, the system responds to the request from a stakeholder after they have initiated a specific interaction to achieve a goal. 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 the 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. Pre-conditions
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 Event

#### 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.

3.1.4 Confirm bid -- the system confirms the bid to the buyer through an email. The

seller is also notified by email.

Alternative flows might describe what happens if the bid 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 allows for a flexible description of the sequence, but in such a fashion that it lends itself well to a structured description and recording system, like that supported by RequisitePro.

To understand how the use case statements and XDE integrate, let's have a look at the basic integration functionality.

## First-stage integration of RequisitePro and XDE

Integration between the two applications is driven from XDE. From the XDE side, connect models and model components in XDE back to Use cases within RequisitePro. In fact, a single model project within XDE is typically linked back to the RequisitePro project that is driving it.

Once the connection has been made, it is possible to create new project documents and individual requirements by using the Tools menu or by using the context sensitive menu against a model or component within Rational XDE.

From within RequisitePro you can also view the model elements and associate with Use Cases textual specifications.

The aim is to map the design elements within Rational XDE with the requirements specification in RequisitePro. Once you have the two elements, you can build a traceability matrix between the two that will be used to monitor whether the design and the requirements match. More detail on the integration and tracing process is provided later in this tutorial, once you have the two systems up and running.

## Other integrations with RequisitePro

Before you take a closer look at the detail of building applications, starting with the requirements process, it's worth considering some of the other interactions that can be employed within RequisitePro itself.

RequisitePro can integrate with a number of different tools, both Rational based and a few external tools. Its primary role is in the first stages of development to document requirements. Requirements can be documented directly within RequisitePro in the following ways:

- ClearQuest provides change request management. You can integrate individual changes with the original requirements specification. Part 3 of this series describes this in more detail.
- ClearCase can be used with RequisitePro either to coordinate changes with the RequisitePro project, or you can simply use ClearCase to archive versions of the project for later retrieval. Part 4 of this series describes this in more detail.
- 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 series.
- Microsoft Project can be used with 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.

Our primary concern in this tutorial, however, is the integration of RequisitePro with Rational Rose tools, and in particular the Rational XDE suite of products. In the next section, you'll start by building the requirements specification.

---

## Section 4. Build requirements specification

### Overview

RequisitePro provides a system for recording and managing requirements. The requirements system is based on a very flexible database design that allows for an unlimited number of requirement types, with 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 defined the functionality back to the original top-level security feature.

Although in RequisitePro the requirements are stored in a database, they can be recorded and edited through a connection to Microsoft Word. This enables project managers and non-technical staff to generate requirements within a familiar interface, while still generating and updating the information in the database.

This section describes the basics of creating requirements that you can use and integrate with other Rational tools.

## Key advantages of using RequisitePro

The main advantage of RequisitePro over standalone documentation is the control provided by a background structured database. Because the information is in a structured format you can generate reports and record additional information about the requirements, beyond that which you could normally describe within a Word document.

The other advantage of 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, at a glance, what changes have been made and why.

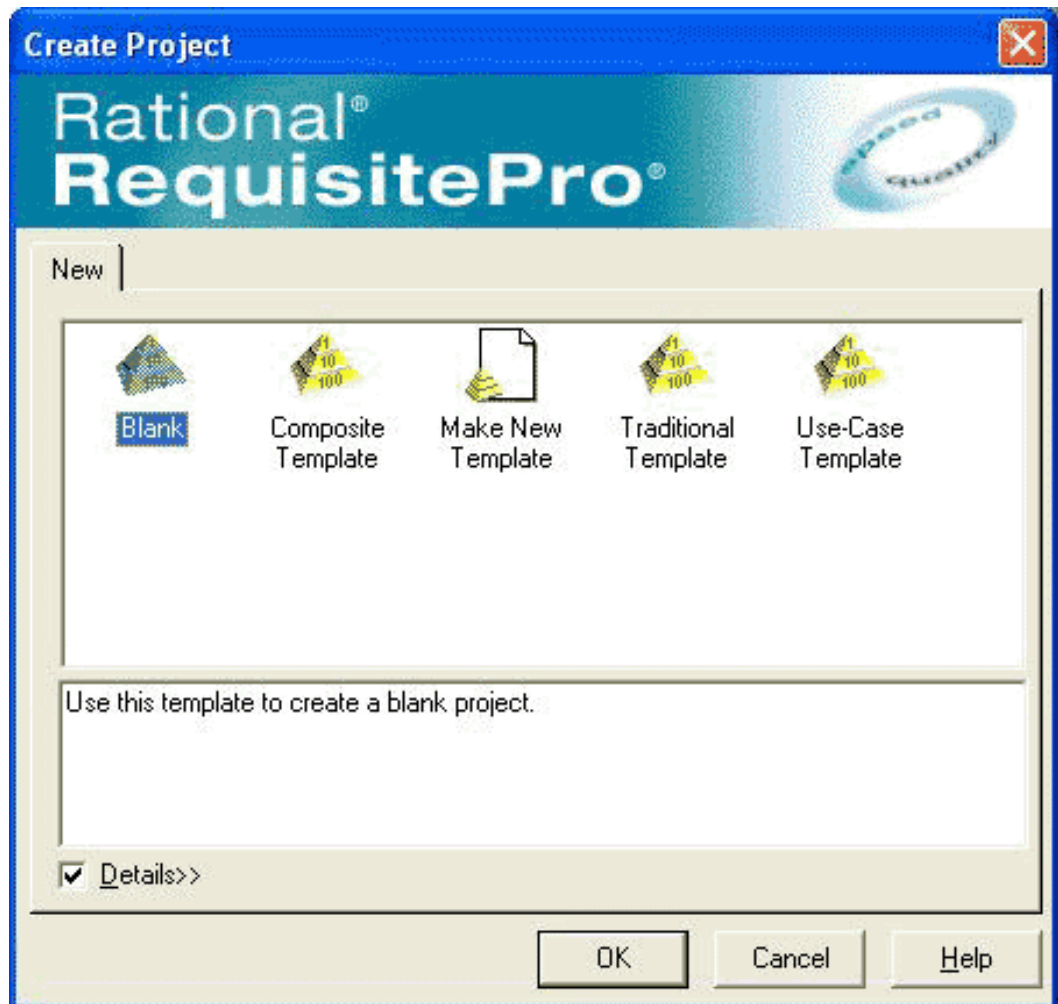
Most significant of all though is that you can use the database to integrate with the other tools in the Rational suite. This includes the model, the code, and also through ClearQuest, ClearCase and TestManager the requests for changes to the requirements and application.

## Creating a new project

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

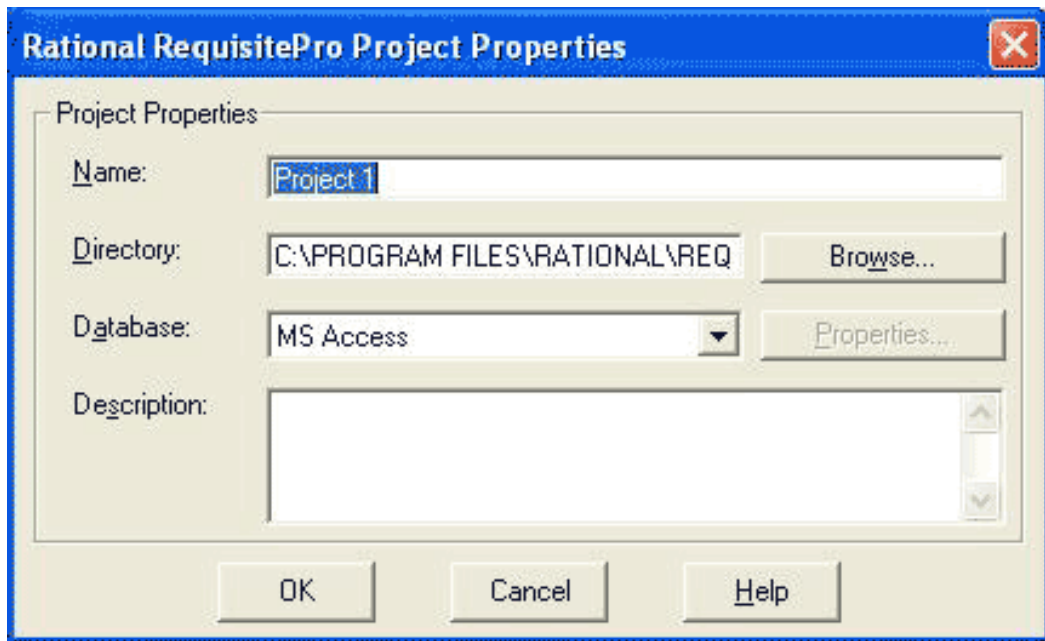
1. Open Rational RequisitePro and choose **New** from the File menu. You'll be prompted with the list of available project templates, as shown in Figure 4.

**Figure 4. Creating a new RequisitePro project.**



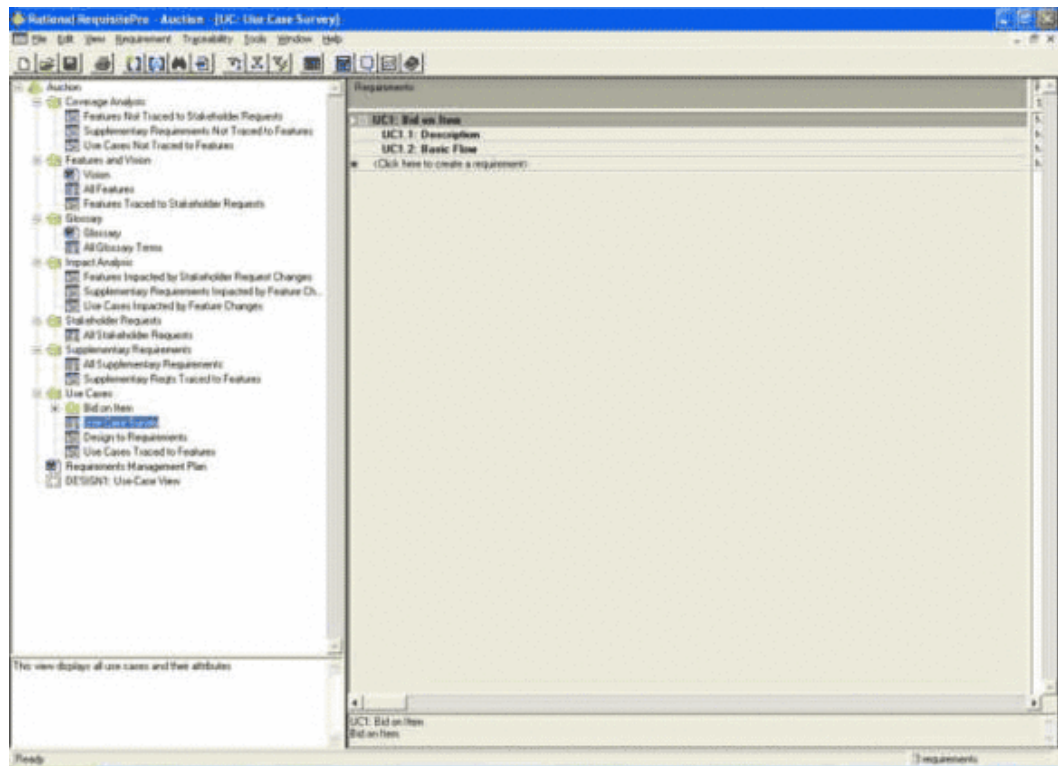
2. You're creating a document for use with the RUP, so choose the Use-Case template. You'll get the project properties window shown in Figure 5.

**Figure 5. RequisitePro Project Properties**



3. Enter the name of the project, choose a directory, database, and optionally provide a description. Use the built-in MS Access database for this example.
4. Once your new project has been created, you're presented with a window that looks something like the one shown in Figure 6 -- all of the items in the tree view on the left are expanded. Each of the folders in the tree is called a package and it logically groups documents and requirements to help organize your project.

**Figure 6. The RequisitePro Project Interface**



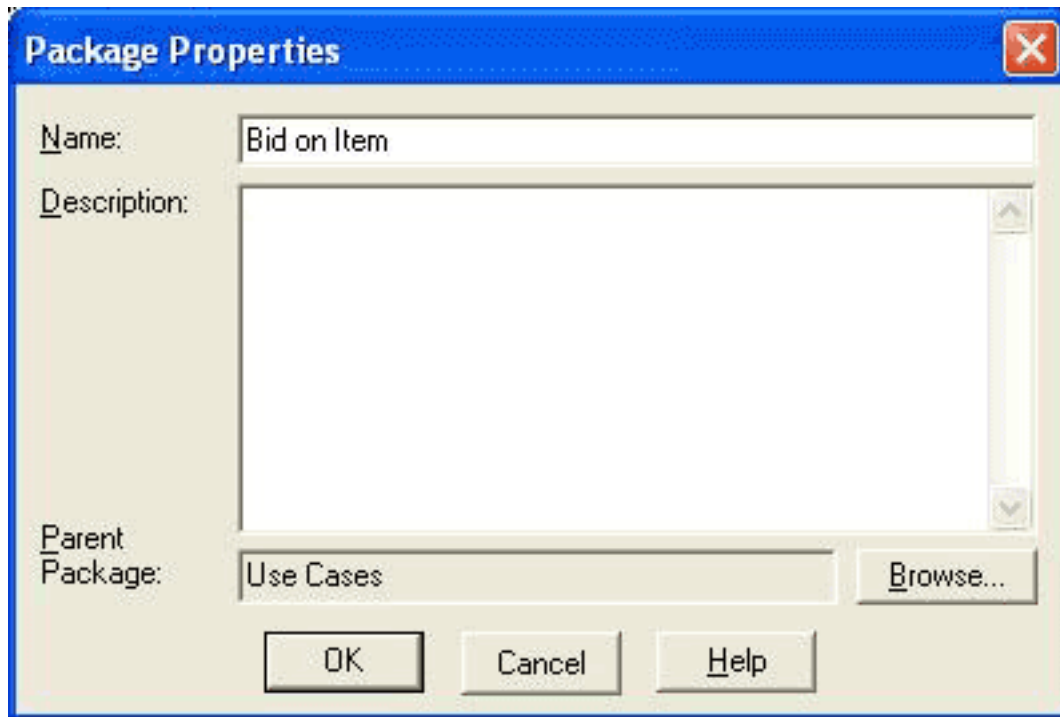
## Creating requirement documents

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

First, let's create a new package in which to hold the use case information for our Bid on Item use case.

Select the Use Cases folder. Right-click on the folder and choose **New Package**. You'll be prompted for the name of the package within the Use Cases package as shown in Figure 7.

### Figure 7. Creating a new package

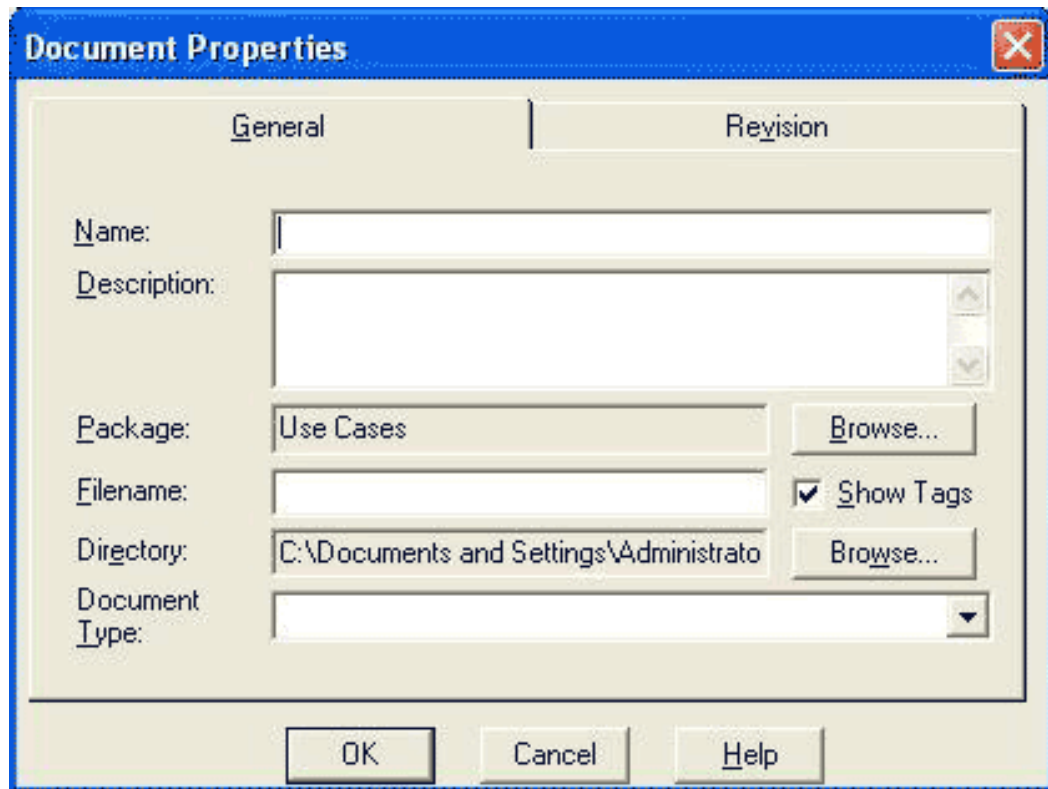


You've now got a package into which you can place your use case definition for the Bid on Item use case.

You can import an existing document but for this tutorial, create a new document to be used to define our use cases for the Bid on Item portion of the auction system:

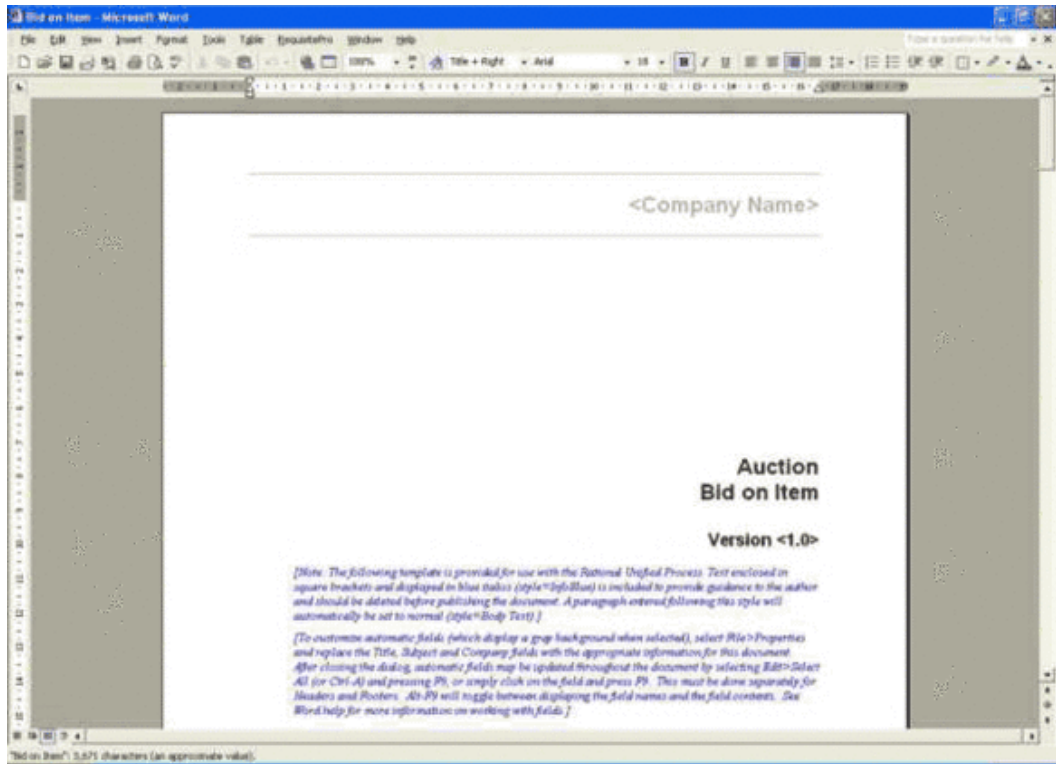
1. Right-click on the Bid on Item folder within the Use Cases folder in the browser and choose **New > Document**. You'll be prompted with the window shown in Figure 8.

**Figure 8. Creating a new document**



2. Enter a name for the document as it will appear in the project and provide a brief description of what it contains. You'll be creating a single use case document for the Bid on item, so enter Bid on item.
3. For the document type, choose the use case document. This opens a document based on a pre-defined template that matches the document template used in the Rational Unified Process. Within RequisitePro certain Word documents are available as Word templates to use within the system. You'll get a window in Word like the one in Figure 9.

**Figure 9. A blank Use Case document**



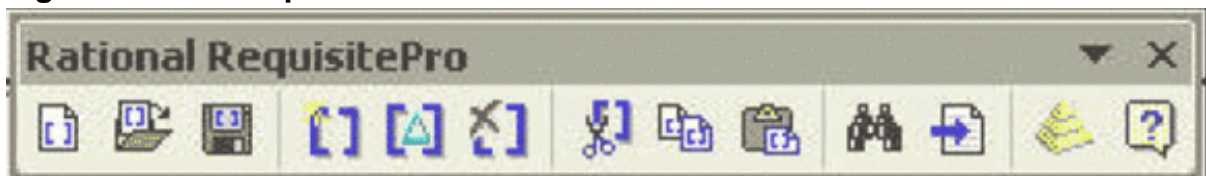
If you browse through the template, you can see that the document provides a reasonable walk through of the sort of information that would be expected in the use case.

## Creating requirements

Requirements within RequisitePro are ordered in a familiar tree structure, just like files and folders on a file system. Thus you can have a primary level of use cases that provide basic information about a given section before you go into more detail in child requirements. This matches the structure of a use-case document within the RUP.

While you are editing a RequisitePro document within Word you may have noticed an additional toolbar (shown in Figure 10) and menu. This provides a link to the RequisitePro system to create requirements in the RequisitePro project from your Word document.

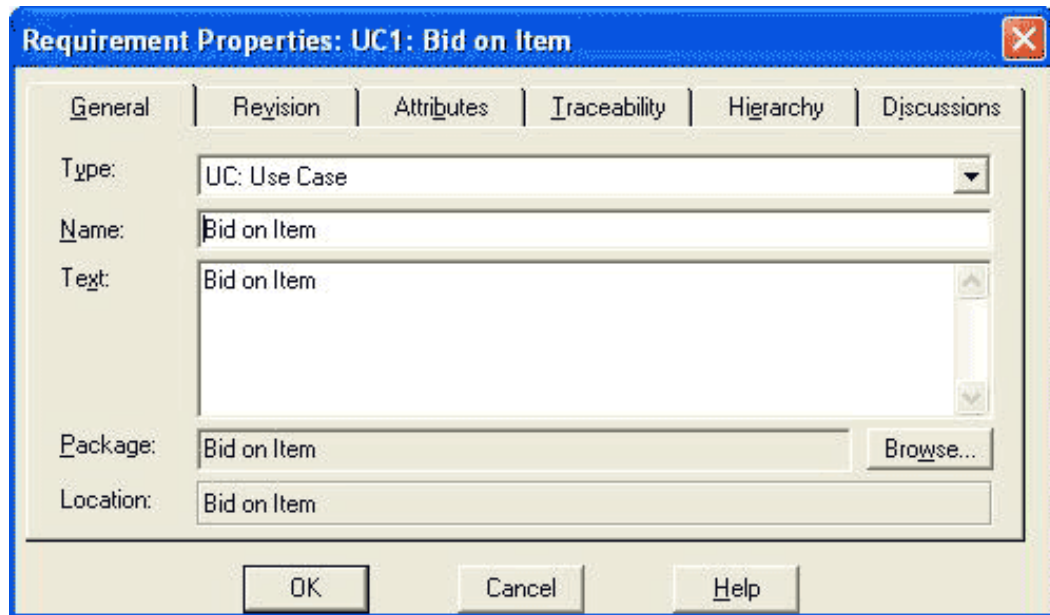
**Figure 10. The RequisitePro Word toolbar**



To create a requirement from a Word document:

1. Enter the text description for the use directly into the Word document. Use the text `Bid on Item` for this example.
2. Select the text you just typed into the document and then click on the New Requirement button (the first square bracket ([[]]) button), or choose **New Requirement** from the RequisitePro menu. See Figure 11.

**Figure 11. Creating a new Requirement**



3. Enter a name for the requirement, specify the requirement type (Use Case), 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 are editing.

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

Repeat the process, adding in additional use cases within the Bid on Item use case that define the individual steps within the Bid on Item requirement.

## Extending requirement information

In the previous section, you created requirements with some basic information about

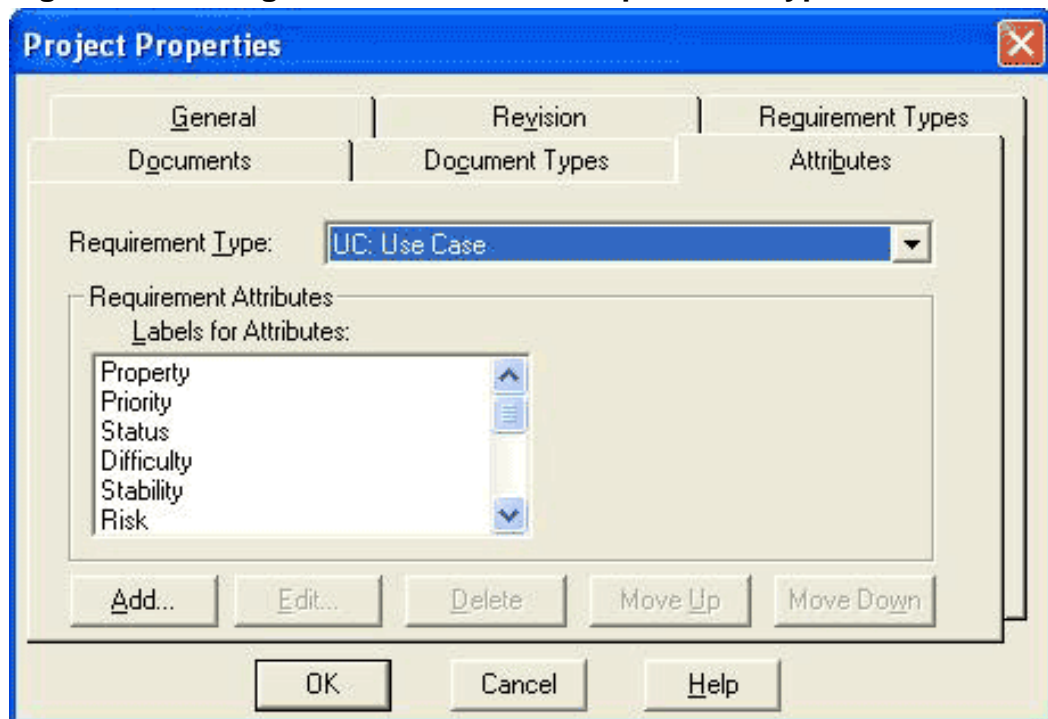
the use case for entering a bid into the system. Now that you have the basic information, it is time to start appending other information to the requirements to make them more useful.

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

To help with your project, you're going to add a source field to your 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 and then select the Use Case type from the popup. A list of the attributes currently defined for the type is shown in the section seen in Figure 12.

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

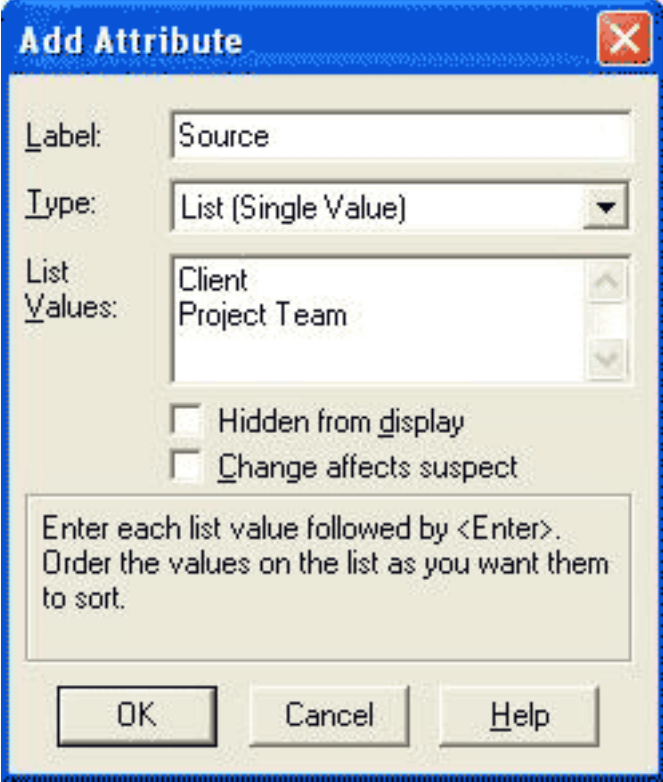


3. Click **Add**. Attributes have a label (the field name you'll populate when you create a requirement), a type, and an optional list of values that are used within a popup or list. The type is a field type just as in a database and can be an integer, floating point, text string, date, time, URL or a list,

either single or multiple selection. Choose **List (Single Value)**.

4. Populate the list (press Return to separate the values) using the information shown in Figure 13.

**Figure 13. Adding the attribute definition**



The screenshot shows a dialog box titled "Add Attribute". It has a blue title bar with a close button. The dialog contains the following fields and controls:

- Label:** A text input field containing "Source".
- Type:** A dropdown menu currently showing "List (Single Value)".
- List Values:** A list box containing "Client" and "Project Team".
- Hidden from display
- Change affects suspect
- A text box with instructions: "Enter each list value followed by <Enter>. Order the values on the list as you want them to sort."
- Buttons: "OK", "Cancel", and "Help".

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

## Views and reports

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

- **Attribute Matrix** -- a simple table of requirements and their attributes.
- **Traceability Matrix** -- a matrix table, showing the relationship between two requirements lists. For example, you might want to show the relationship between the stakeholder requests and the main software requirements.

- **Traceability Tree (Traced into)** -- a tree showing how requirements relate to the specified requirement. For example, you might want to show a tree that describes how stakeholder, developer and other requirements relate to the main software requirements.
- **Traceability Tree (Traced out of)** -- a tree showing the requirements traced from the specified requirement type. For example, you might want to show all the requirements traced from a stakeholder request.

You'll use the traceability matrix later to map your Rational XDE model designs to your requirements.

---

## Section 5. Model applications with Rational XDE

### Overview

Rational XDE is a modeling and development environment that enables you to create diagrams and UML elements that describe the design of an application. These diagrams can be used to automatically generate code for the application, such as the classes, database structures and other components.

Once you have these in place, you can also use Rational XDE to start building the additional functionality that has not been modeled (for example, a GUI or custom java class) to build the application. Rational XDE, like other parts of the development tools from IBM, uses the Eclipse platform to provide an interface for building the various elements.

In this section, you'll learn the basic steps for creating models in Rational XDE.

### The Rational XDE interface

Rational XDE is part of a suite of tools that provide modeling and application development tools. All are based on the Eclipse platform. Eclipse is a development environment that was funded by IBM to develop tools for developing applications just like WebSphere and Rational XDE. The Eclipse project is now open source, having been donated to the open source community by IBM.

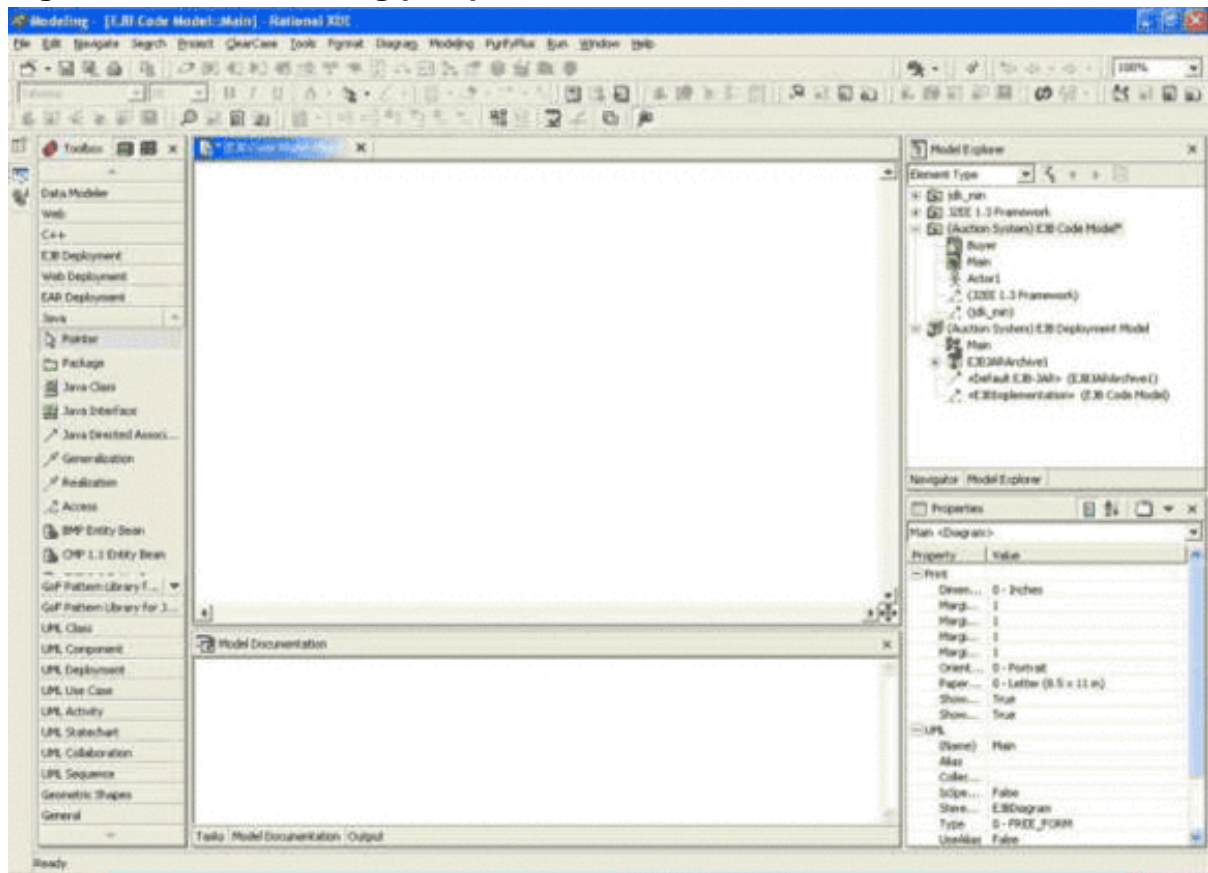
Eclipse itself is written in Java using a flexible plug-in system that allows Eclipse to be used to develop a range of different content. Its primary use is as an Integrated Development Environment (IDE) for developing applications, and through the plug-in

system it's possible to develop applications in a range of different languages -- primary Java and C++.

The same plug-in flexibility also allows it to be used for the visual modeling that you will use in this tutorial to create a use case diagram of our Bid on Item portion of the Auction system.

Eclipse interacts and displays information using different perspectives. Perspectives provide an alternative range of sub-windows used to display information about a project or project component. For example, in the modeling perspective you can browse the different models in a project. Within the Java perspective, you can view code and the class structure. Figure 14 shows the main modeling perspective.

**Figure 14. Main modeling perspective within Rational XDE**

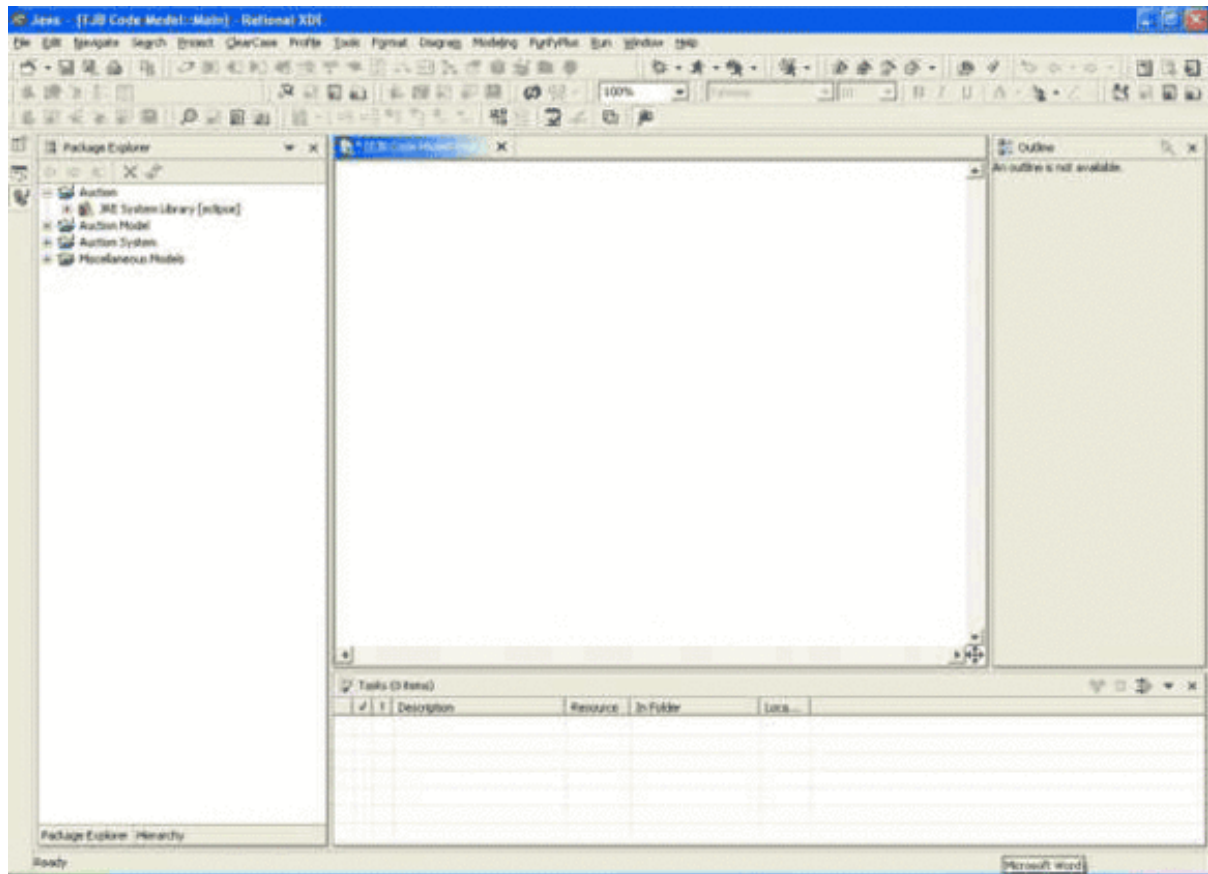


The toolbars along the top are a mixture of generic and context sensitive toolbars. The toolbar on the left changes between perspectives. The Toolbox provides a general toolbar for adding specific components to a system, for example, classes and sequence diagrams when working with a model. The central panel supports a number of different tabs in which you can view source code, models and the detail of other components. On the right, you have explorers for models and class hierarchies, and properties for a selected component. In each case the windows are

optional. The standard ones shown here are the defaults for a given perspective.

If you change the perspective to using the toolbar, you can switch to the Java perspective and get a slightly different view, as seen in Figure 15.

**Figure 15. The Java Perspective**

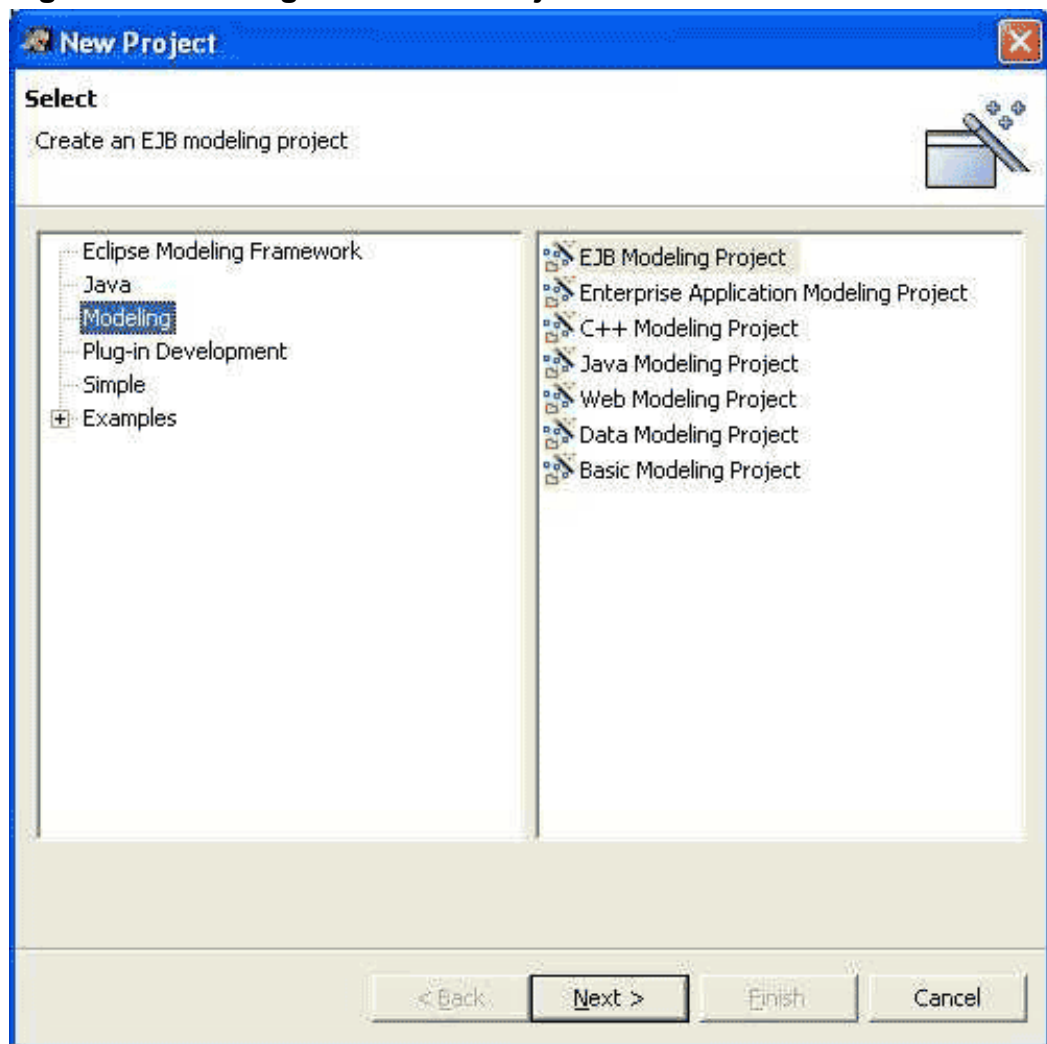


The toolbar on the left has been replaced with a package explorer, which matches the various packages that make up your Java application. A replacement tasks window at the bottom shows any outstanding tasks you've added to the project, and the outline browse on the left allows you to work through the entries in the current source file.

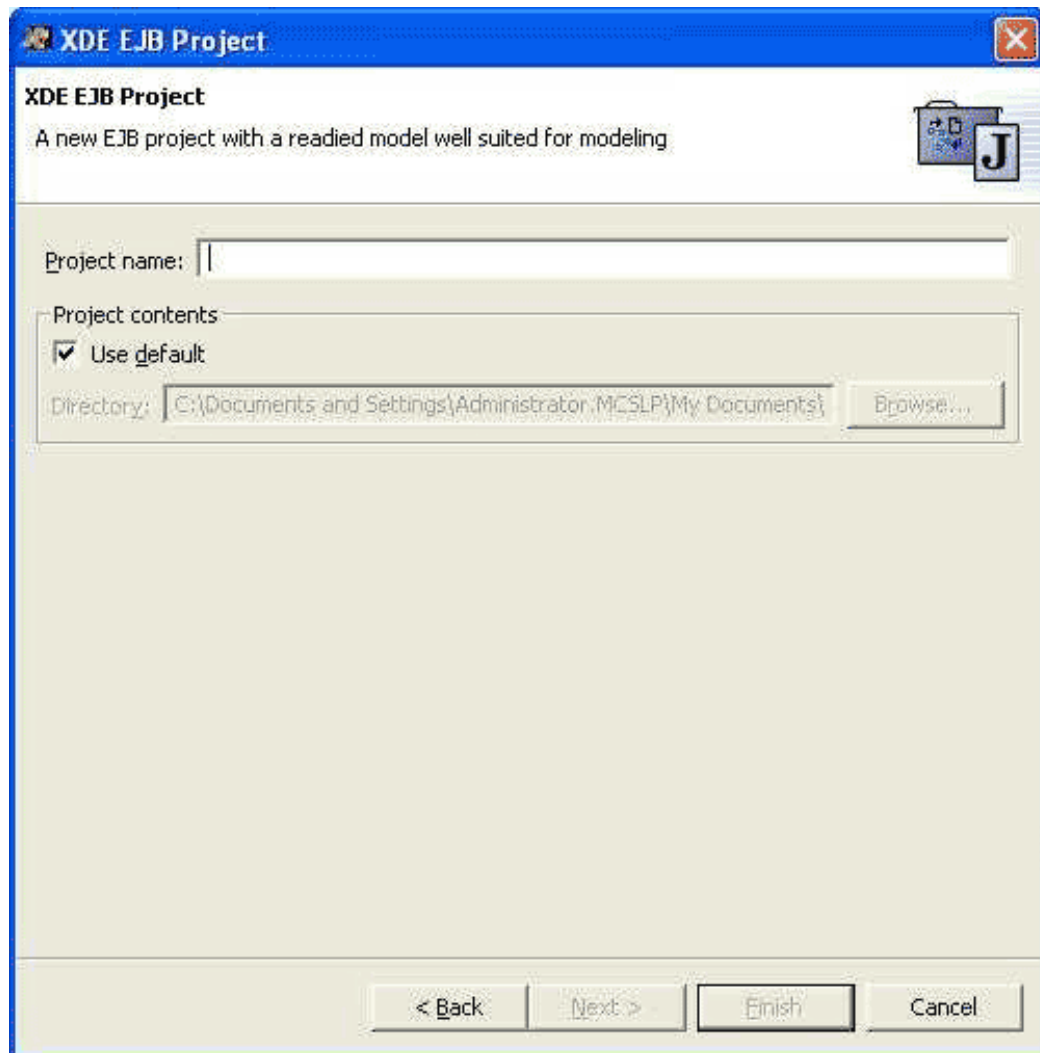
## Creating a new project

You need to build a use case model diagram that models your Bid on Item use case. To create a new project in which to hold the models, you first need to create a new Java EJB model:

1. Select **New > Project**. A window opens as shown in Figure 16.

**Figure 16. Creating a new XDE Project**

2. Choose **Modeling** from the list on the left and **EJB Modeling Project** from the list on the right.
3. Click **Next**. See Figure 17.  
**Figure 17. Setting Project Preferences**



4. Give the project a name -- let's call it `Auction System`.
5. Click **Finish** to create the project. You'll get the modeling perspective you saw in the previous section.

You now have a project into which you can start building models for the application.

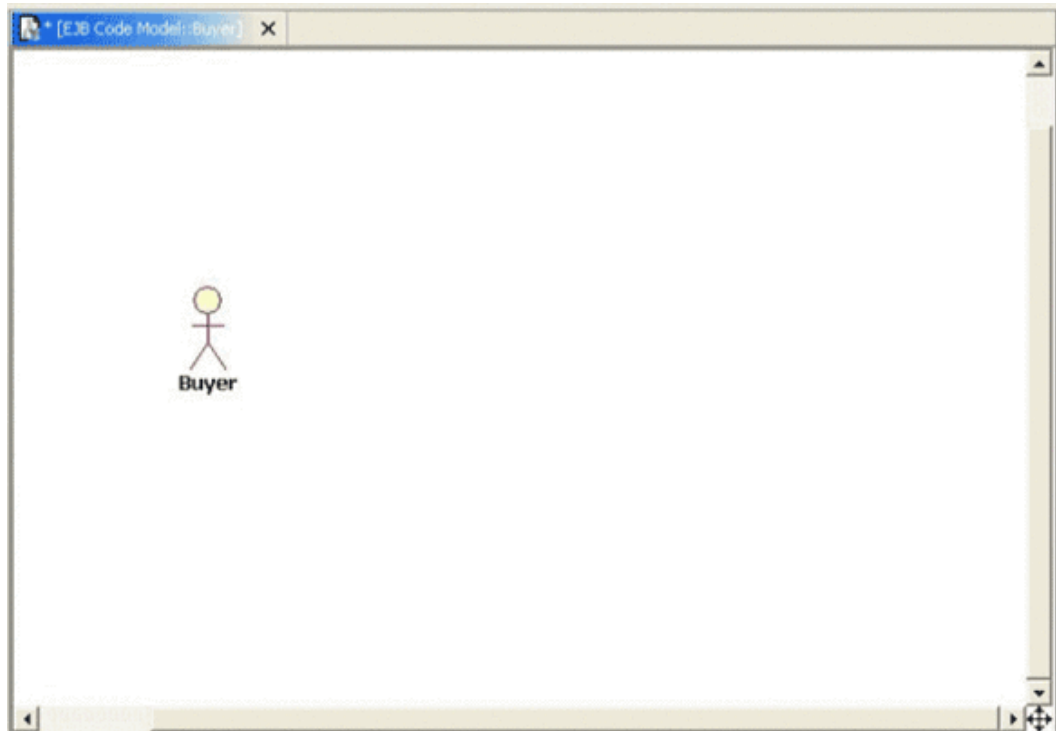
## Creating a new use-case diagram

With your project open, it's time to start building a use case diagram for your system that documents the functional requirements. You'll be connecting the elements within this diagram back to the use cases you defined in RequisitePro.

To start creating a use case diagram for your system:

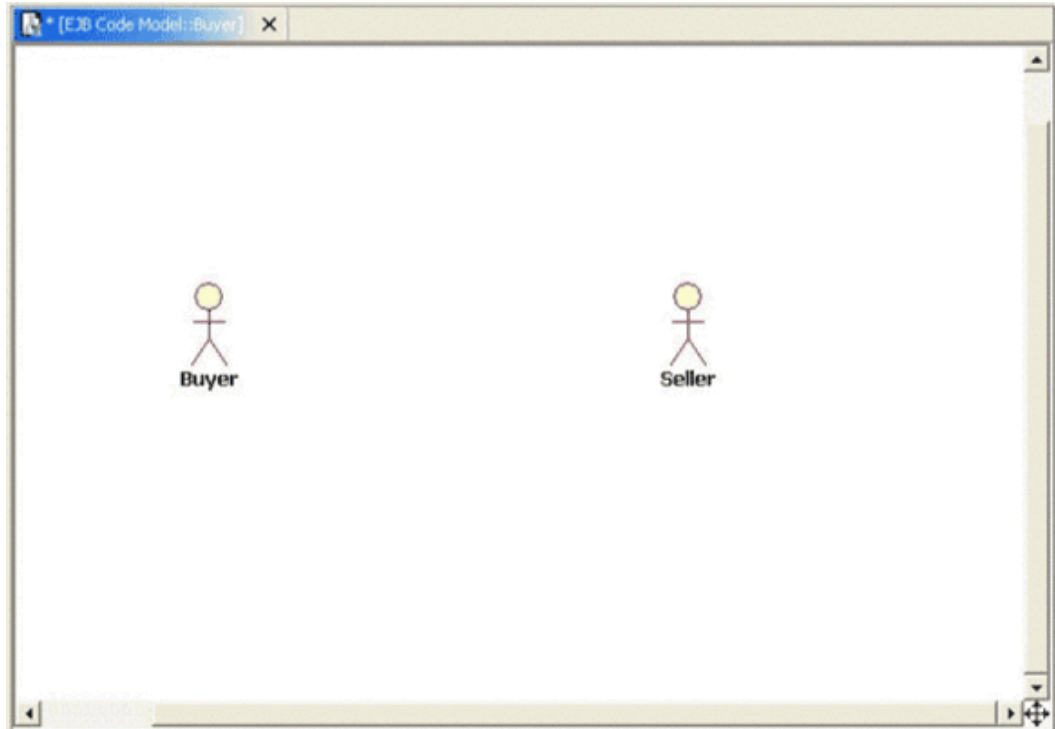
1. Select the EJB Code Model package for your Auction System. Right-click and choose **Add Diagram > Use Case**. A new entry appears in the tree.
2. Enter `Buyer` for the name of the new diagram, which you'll use to model use cases for the buyer portion of the auction system. A blank canvass appears. Use this to diagram your use case for the buyer.
3. Using the toolbox on the left, change to the UML Use Case toolset. This provides a range of tools that can be used to define the different elements of our use case model.
4. Click on the Actor tool.
5. Move the pointer to the blank diagram and click to create a new actor. Name the actor `Buyer`. See Figure 18.

**Figure 18. The first actor in your model**



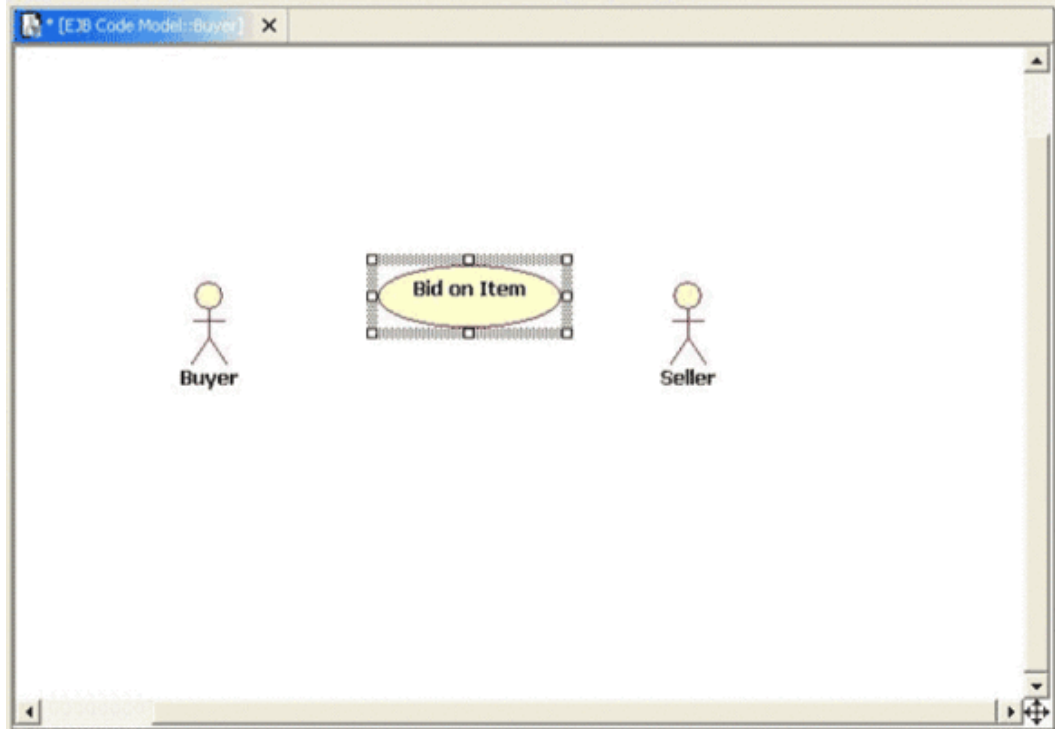
6. Repeat the process, this time creating an Actor with the name `Seller`. See Figure 19.

**Figure 19. The second actor in your model**



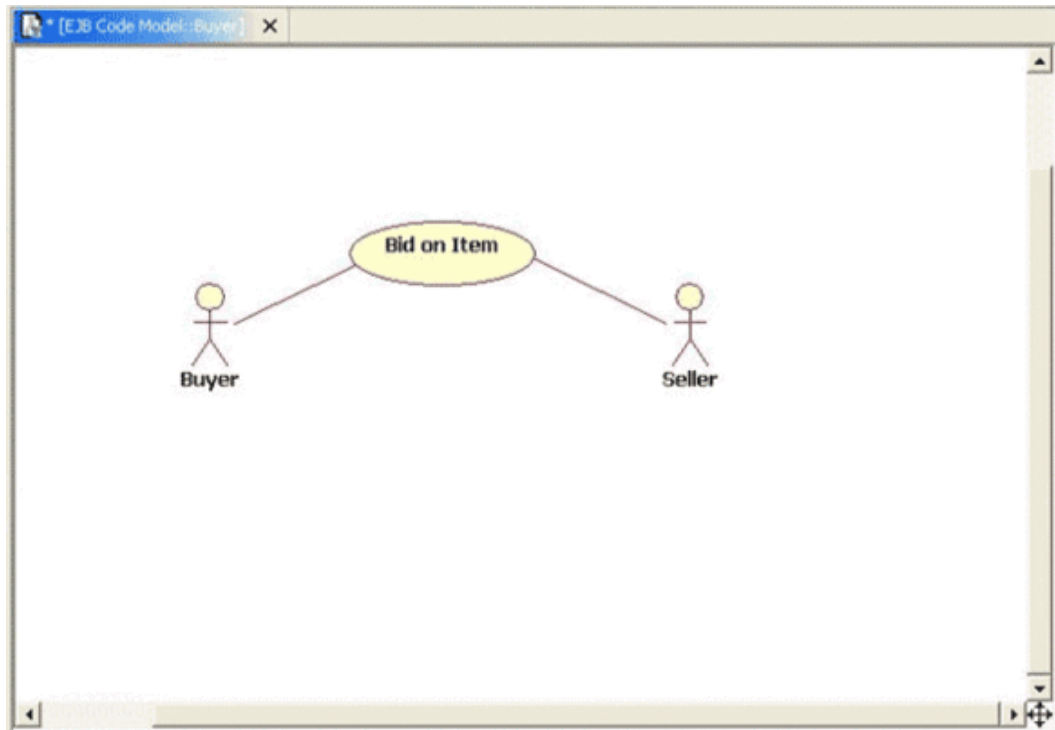
7. Create a use case that defines the functional requirement and its affects on these two actors. Click on the Use Case tool in the Toolbox, and then click in the diagram to create a use case between the two actors. Name the use case `Bid on Item`. See Figure 20.

**Figure 20. Use Case model**



8. To create a relationship, scroll through the list of tools until you get to the Association. This defines the relationship between an actor and a use case. In your main diagram, the Buyer has a relationship to the Bid on Item use case, which in turn has a relationship to the Seller. To create the relationship, click the first element and then the second element to create the association. You should end up with a diagram like the one in Figure 21.

**Figure 21. Your first stage model**



You created a very basic use case diagram that describes the basic interaction between the buyer, the seller, and the Bid on Item use case. With this basic structure in place, you need to associate the Bid on Item use case in the diagram with the use case requirement in RequisitePro -- that'll be the focus of the next section.

---

## Section 6. Build a class diagram in XDE

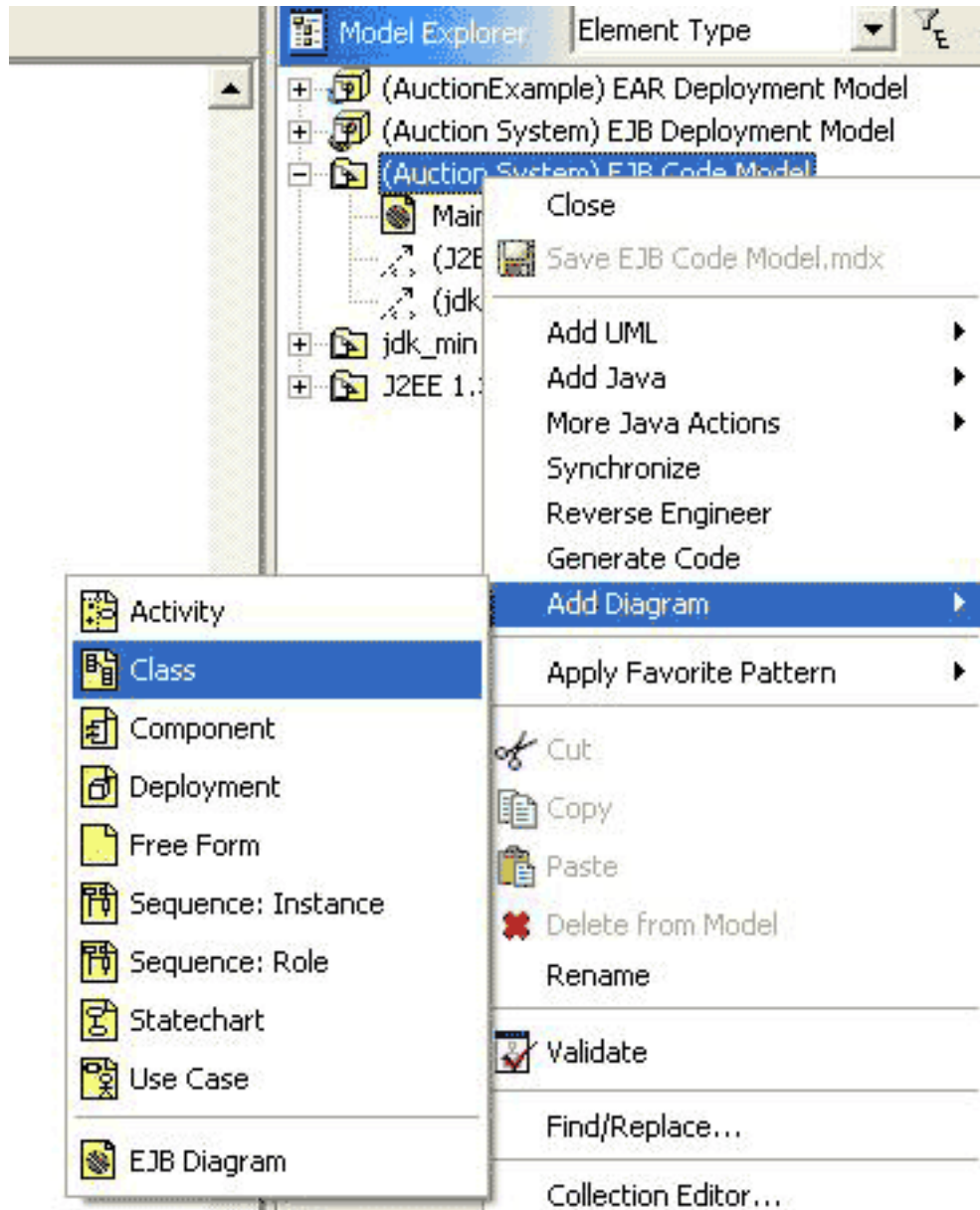
### Creating a new class diagram

Having built a use case diagram in XDE, and linked this back to the original Requisite Pro requirements, you can move forward with development. The next stage is -- in all likelihood -- to build a static class diagram based on the information in your use case.

XDE remains the appropriate tool to use for class diagram building. Create the diagram in the same Auction System model that is already housing your use case.

1. Open the Auction System model (if it's not already open).
2. From the Model Explorer, right-click on the Auction System EJB Code Model and select **Add Diagram > Class** (see Figure 22).

**Figure 22. Add class diagram**

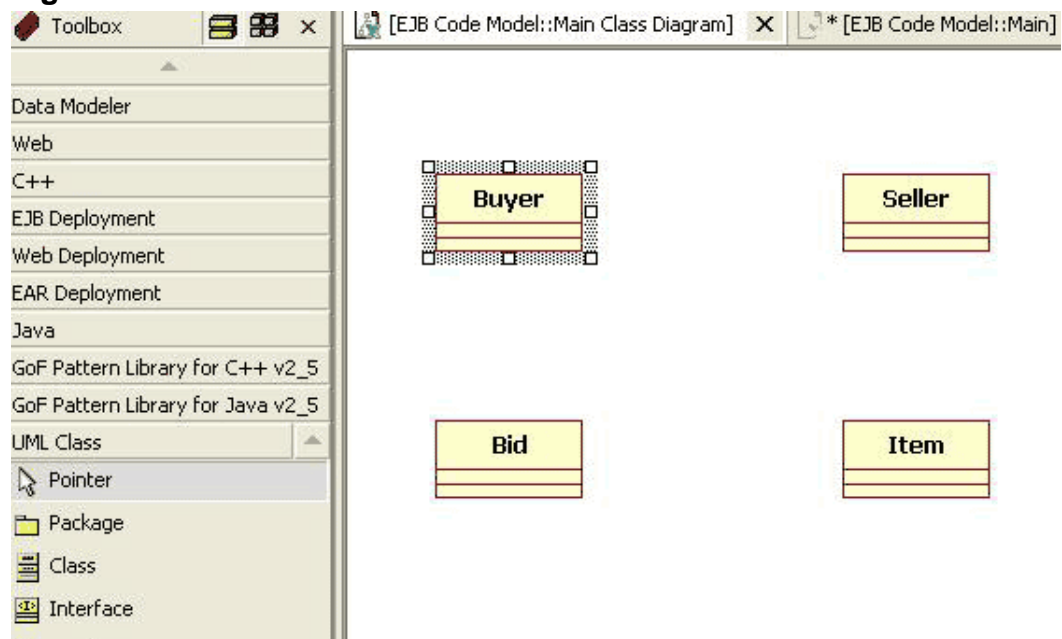


3. Rename the diagram Main Class Diagram. Although you are starting this static class model from a single use case, more use cases will probably follow. Those use cases should contribute to this same static class model rather than spawning fresh class diagrams of their own; this is likely to be a class diagram for the whole system rather than any single

use case.

4. A diagram editor window for the use case opens (if a window did not open, double-click on the diagram name in Model Explorer).
5. Do the following:
  1. Select the editor window for the EJB Code Model::Main Class Diagram if it's not already selected.
  2. Note that the UML Class palette of the Toolbox view (left-hand side) expands uppermost automatically.
  3. Click on **Class** in the UML Class palette, and then click on your diagram. After a moment or two, a class box with a default name `Class1` appears on the diagram. Change the name to `Buyer`.
  4. Use the same technique to add three more classes to your diagram -- "Seller", "Bid" and "Item".
6. Save the diagram by clicking **File > Save** or use Ctrl-S. The end result should look something like Figure 23:

**Figure 23. All classes added**



## Adding attributes to classes

Before proceeding, think back to your requirement for a moment or two. "Buyer" and "Seller" are classes derived directly from the actors in your use case diagram for the

Auction System. Some actors stay external to the system, but chances are that you want to model attributes of the buyer and seller within the system. They're both likely to be registered users of the Auction System, so you'll need some internal system record of who they are, which items they are bidding for or selling, and contact details -- and that's just to start with.

"Bid" and "Item" are classes that leap out of the use case text. Again, there are a number of obvious attributes belonging to these classes that you will want to capture. A particular bid will be time-stamped in some way, and have an associated amount. Items usually have a reserve price, a title and a description.

Having thought about the requirement, look at the mechanics of adding some of these attributes to the classes you have placed on the XDE class diagram.

1. Right-click on the **Buyer** class in the diagram.
2. From the context menu, select **Add UML > Attribute**.
3. Type over the default attribute text with `- name` and press Enter. You'll be on the next line ready to add another attribute.
4. Type `- address` and press Enter.
5. Add a third attribute by typing `- rating` and press Enter.
6. Press the Esc key to cancel the addition of a fourth attribute.

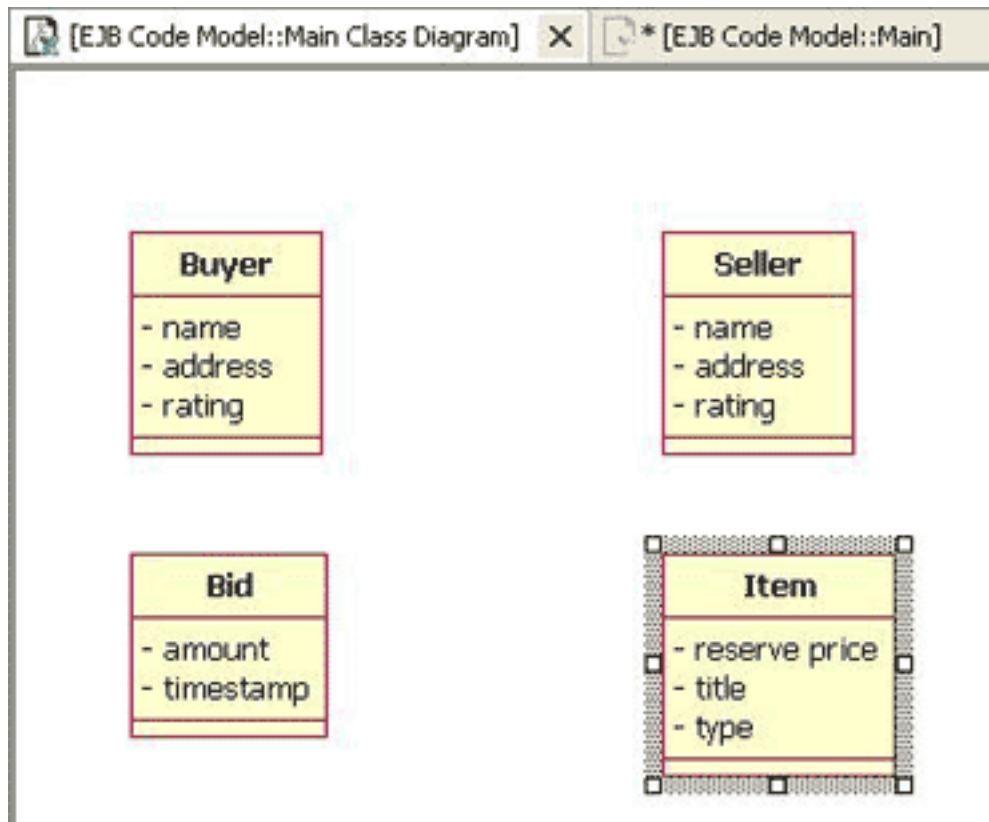
The "-" sign before the attribute name has to do with the UML standard for expressing access rights -- as this is a well-behaved class, keep the data private and graft in accessor methods later. You can associate a type for the attribute at the same time by typing (for example) `- name: String`. At this stage though, keep your options open.

Using the same technique as described above, add attributes to other classes as follows:

- Seller: name, address, rating
- Bid: amount, timestamp
- Item: reserve price, title, type

Don't forget to save your work. The net result should look something like the diagram in Figure 24:

### Figure 24. All classes attributes



## Adding behavior to classes

At this point, you have classes and attributes, but nothing in our Auction System does anything -- there's no behavior as yet! You need to add some operations to your classes to make them meaningful.

For some classes, there might be little to provide other than accessor operations to get at the private data. Currently, the buyer and seller don't have much behavior beyond this. Sure, they participate in the use case, but all the action happens around bids, items and auctions. So, leave the Buyer and Seller classes alone. You can add operations such as "getName", but those just clutter your logical view of the world, which is all it is at this stage. Later, XDE provides accessor methods practically free of charge.

For a bid, though, you can imagine particular behaviors. You might want a bid to be able to tell you if it is (currently) the highest bid for an item. At the end of the auction, you'll want a bid to know if it's successful or not. Items need to keep track of the bids kept against them and provide information such as the starting bid, the current bid, and the maximum bid.

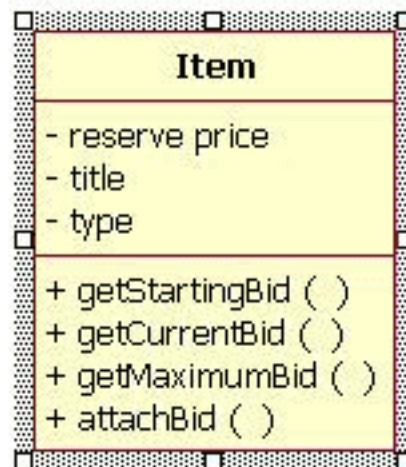
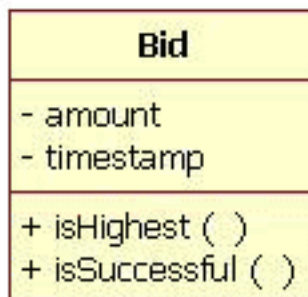
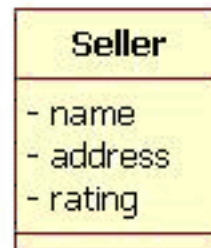
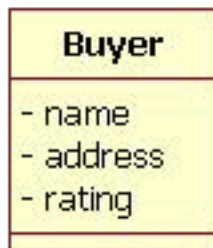
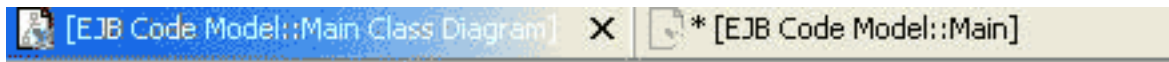
Let's add some operations to the Bid and Item classes. Adding operations is a very

similar business to adding attributes:

1. Right-click on the **Bid** class in the diagram.
2. From the context menu, select **Add UML > Operation**.
3. Replace the default operation text with `isHighest`.
4. Press Enter. You'll be on the next line ready to add another operation.
5. Type `isSuccessful` and press press Enter.
6. Press the Esc key to cancel the addition of a third operation.
7. Repeat this series of steps on the "Item" class. Name the class operations `getStartingBid`, `getCurrentBid`, `getMaximumBid` and `attachBid`.
8. Don't forget to save your work.

The net result should look something like the diagram in Figure 25:

**Figure 25. All classes operations added**



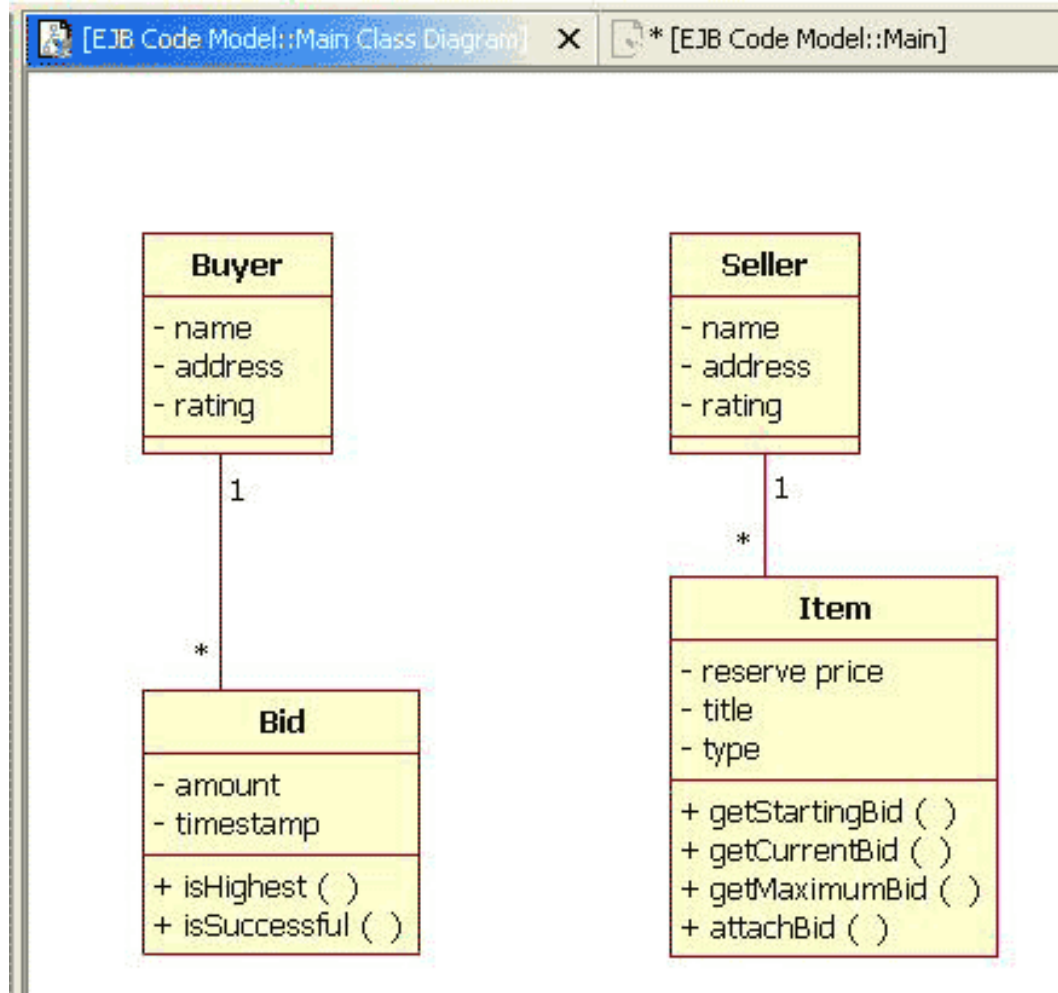
## Making associations between classes

Currently, you have classes with some operations and attributes, but they aren't linked in any way. Next, you'll learn how to make those links.

It's obvious that a potential buyer is going to make bids, while a seller sells items in the Auction System. You have a couple of associations to set-up -- straight off the bat. Furthermore, a potential buyer might make many bids in the system, or none at all. Similarly, the potential seller might not register any items to sell, or might have a whole list of selling and sold items. Let's see how you can use XDE to model these two similar relationships.

1. Click **Association** in the UML Class palette (not Directed, Aggregation or Composite Association, just plain Association).
2. As you hover over the class diagram, the cursor changes to a + sign with an oblique stroke next to it. Click and *hold* in the Buyer class, and drag the cursor (trailing a dotted line) into the Bid class. Release the mouse button.
3. There should be a solid line joining Buyer and Bid. Click on it to select it (the ends of the line show solid black boxes -- the join points to the classes).
4. Go to the Properties view (bottom right of the Modeling Perspective), and find the End1Multiplicity property. Using the ... button on the right of the property, select \* as the multiplicity in the window that appears.
5. Click OK. You should see a \* on the association line, closest to the Bid class. This indicates that anything between zero and many bids can exist for a given buyer.
6. Find the End2Multiplicity property. Repeat the previous step, but this time select 1 as the multiplicity. This indicates that for a given bid, there must always be one associated buyer.
7. Set up a similar association between Seller and Item so that a seller can have many items but an item can have only one seller.
8. Save your work and compare it to Figure 26:

**Figure 26. Associations**



## More complex associations

You've set up some simple associations between the Buyer and Bid classes, and also between the Seller and Item classes. Let's look at the more complex relationship between a bid and an item.

It seems pretty clear that a bid can't exist in isolation: it's always a bid for a particular item. However, an item can perfectly well exist without any bids. But also, and more subtly, if you get rid of an item, all its associated bids should disappear.

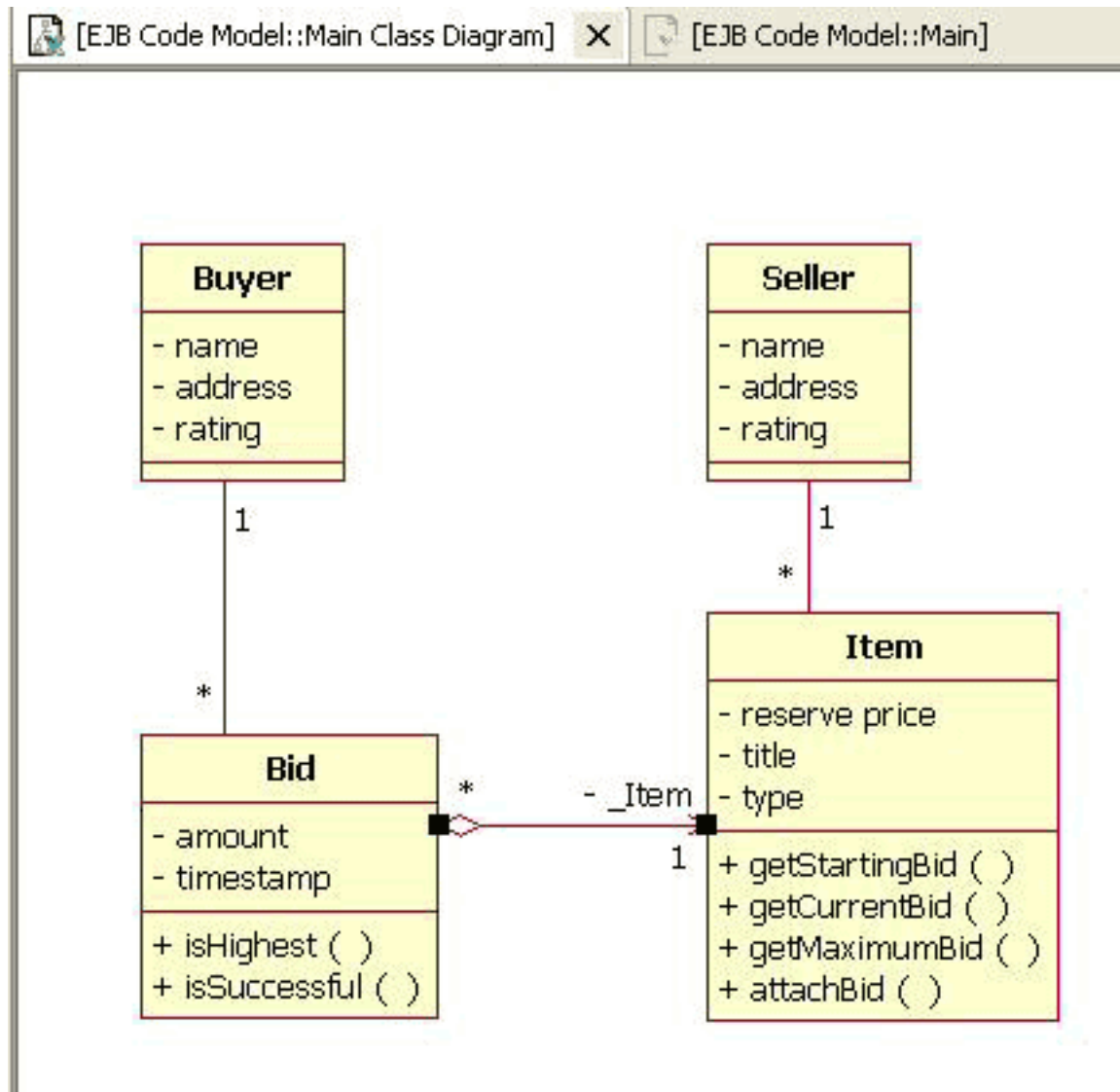
First, let's explore the relationship from Bid to Item. It's what UML calls an aggregation association -- part of the make-up of the Bid is an associated Item. To model that in XDE:

1. Click **Aggregation Association** in the Toolbox UML Class palette.

2. Hovering over the class diagram, the cursor changes to a + sign with the aggregation's hollow diamond/arrow icon adjacent.
3. Click and hold on the Bid class and drag the cursor to the Item class. Let go of the mouse button.
4. The End1Multiplicity property for the association defaults to zero to many (\*), which isn't appropriate. A bid is associated with only one item. Change End1Multiplicity to **1**.
5. Change End2Multiplicity to \* (the \* shows up at the hollow diamond, Bid end of the association on the class diagram). Anywhere from zero to many bids can be associated with an item.

Save your work. The result should look something like Figure 27:

**Figure 27. Aggregation associations**



## More complex associations (continued)

Having made the association from Bid to Item, let's finally add an association from Item to Bid. Use a composition association this time, to express the idea that there should be a cascading delete from Item to Bids (if the item goes, then the bids should too).

In XDE, perform the following steps:

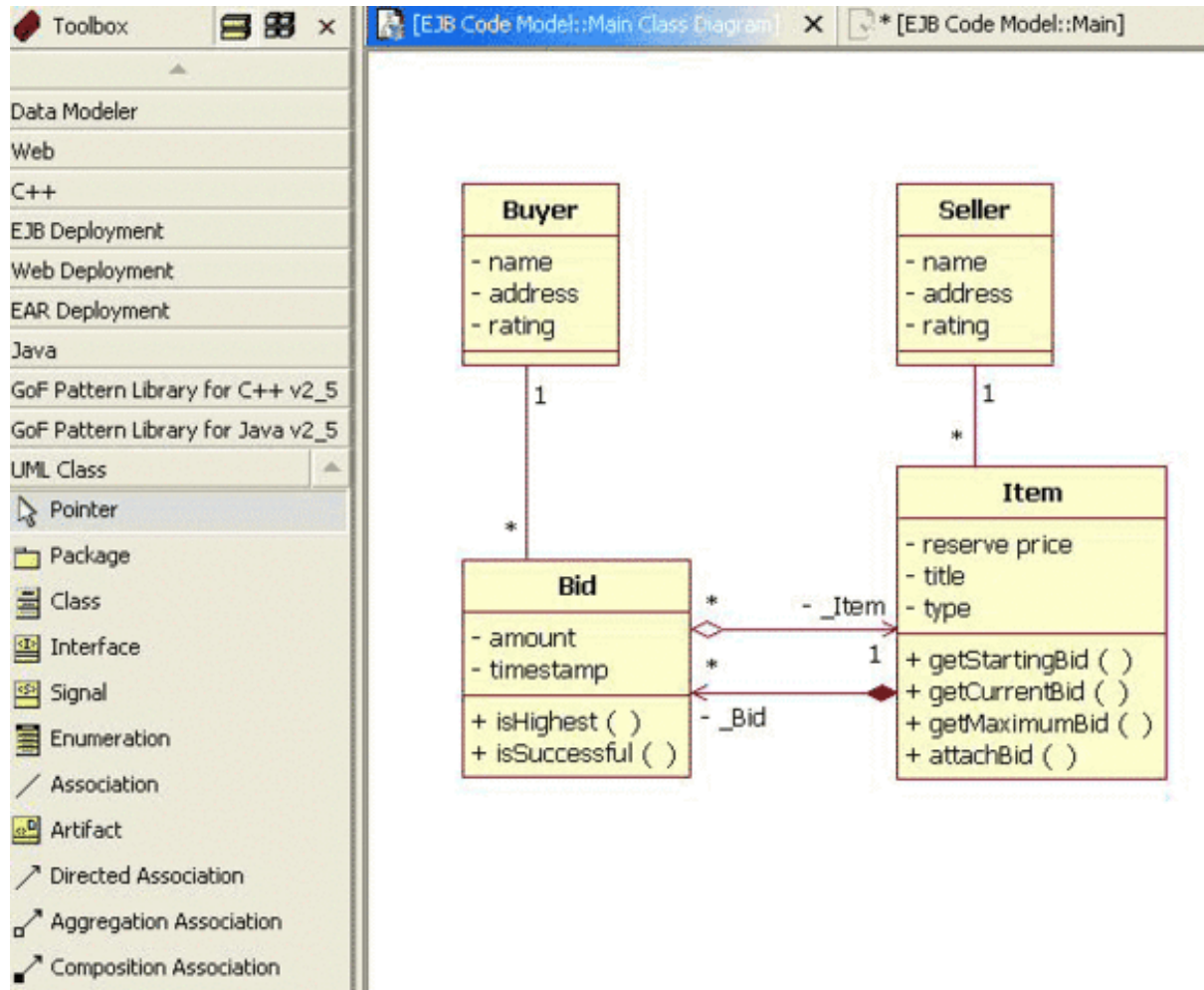
1. Click **Composition Association** in the Toolbox UML Class palette.
2. Hovering over the class diagram, the cursor changes to a + sign with the

composition's solid diamond/arrow icon adjacent.

3. Click and hold on the Item class, and drag the cursor to the Bid class. Let go of the mouse button.
4. This time, the multiplicities should be correct.

Save your work. The diagram should look like Figure 28:

**Figure 28. Composition association**



That's your logical class diagram complete, or at least as complete as you need it for now. Move on to the next phase in this mini-project lifecycle supported with the Rational toolset.

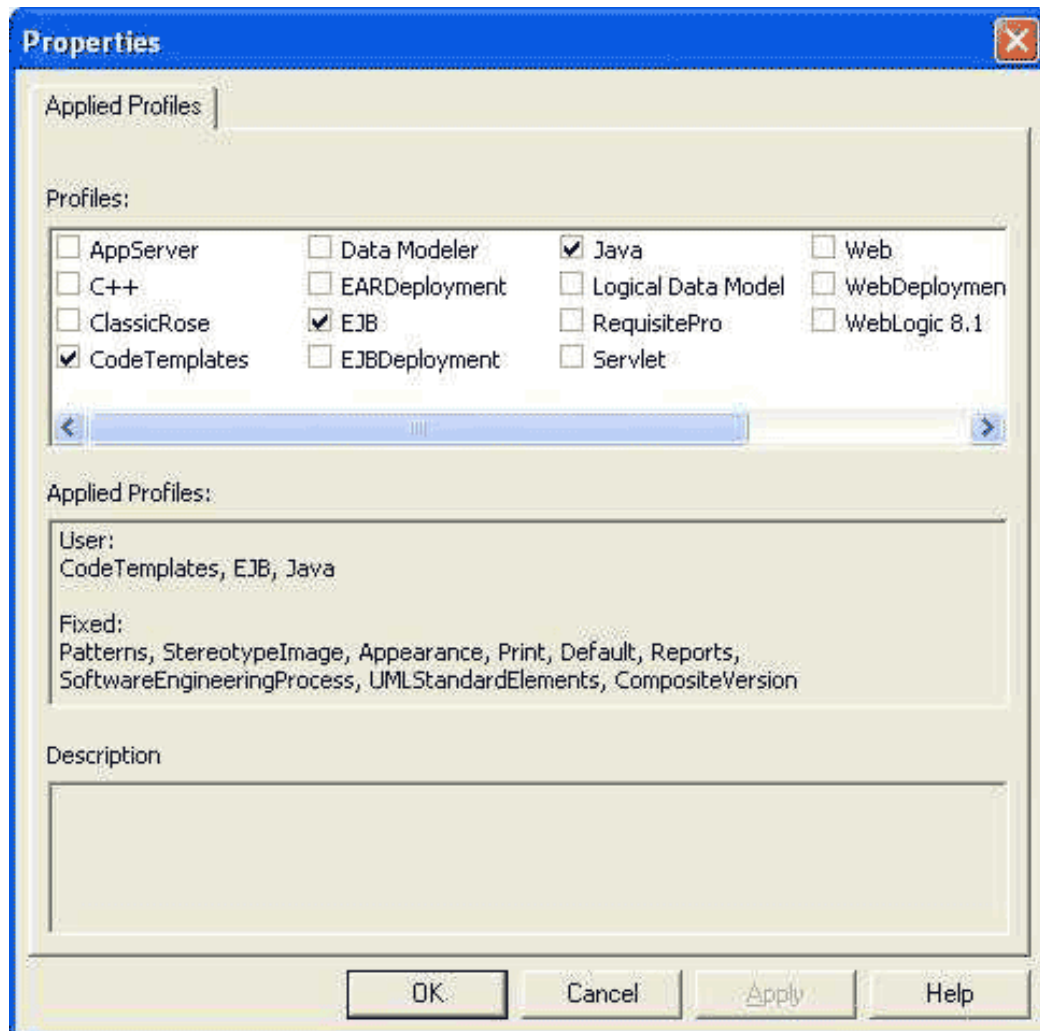
## Section 7. Track components back to requirements

### Setting up the integration

The first stage to setting up the integration is to make sure the correct profiles have been applied to the Rational XDE model. By default, Rational XDE models are not enabled for integration with RequisitePro.

To enable the integration:

1. Click the model that you want to integrate to a RequisitePro project.
2. On the properties window, click the **AppliedProfiles** property and then click the box to open the list of available profiles. See Figure 29.  
**Figure 29. Adding a profile to your project**



### 3. Select the **RequisitePro** profile.

Once the model has been enabled for RequisitePro integration, there are a number of locations where you can invoke, apply, and control the integration between the two products. The main Tools > Rational RequisitePro menu is always available, while context sensitive menus are applied to individual elements as appropriate.

The contents of the menu in both cases depend on what is currently selected:

For use cases:

- **Open/New Use Case Document:** create or associate the use case with an existing Rational use case document.
- **View Requirement Properties:** view the properties for a given use case, and view and edit the attributes and traceability.

For design elements (classes, actors, etc.)

- **Add/View/Remove Traceability:** create, view or remove links to requirements for the given element.

Menu options for package:

- **Associate/Disassociate to RequisitePro:** associate or remove the association to a selected RequisitePro project. Once associated, use cases and traceability information is updated in the corresponding RequisitePro project.

Let's look at these options in more detail.

## Associating a model to a project

The second step, once the RequisitePro profile has been applied to a model, is to associate it with a given RequisitePro project. This tells Rational XDE where to look for requirements and where to create the Design requirements to add to your RequisitePro project. These are Design requirements you'll use to track the designed components and then relate to the use cases on which they are based.

To do this:

1. Select the package you want to associate. You Auction package that you defined in the Rational XDE section.
2. Right-click the package and then select **Associate to RequisitePro**. You'll be prompted to choose a RequisitePro project.
3. Find the project you created in the earlier section. Specify the default document type and the default requirement type. Use the Use Case Specification document type and Use Case requirement type.

## Mapping the design back to required functionality

Once the association is in place, associate individual elements within the design model to the original requirements within RequisitePro. To do that, create traceability from a model in XDE to the use case in RequisitePro. When you add a trace, a new requirement is created in the requirements database called the Design requirement, which contains information about the model component in the design.

Use the relation functionality in individual requirements to map the design

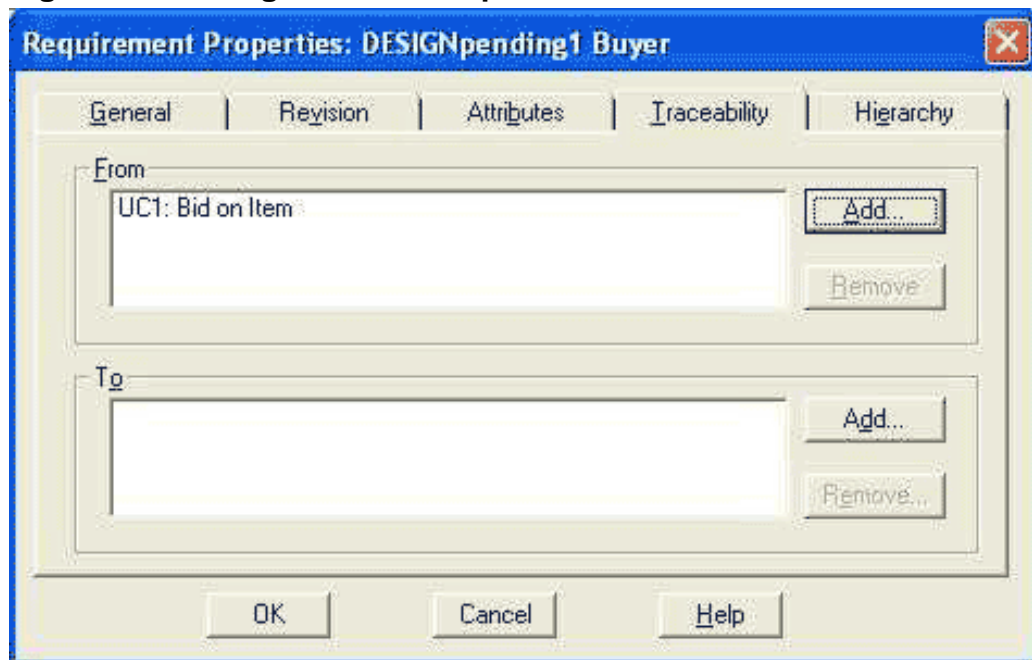
requirement you created from the Rational XDE model to the original requirement. Relationships between requirements work simply, traceability works either to or from a requirement. Thus, Requirement A can be traced *from* Requirement B. In doing so, Requirement B is traced *to* Requirement A.

In the model, Rational XDE design elements are traced *from* original requirements.

To create a trace between a model in XDE and the requirement in RequisitePro:

1. Select the class, diagram or other model component and then right-click and choose **RequisitePro > Add Traceability**. In this case, choose the Class model for the Bid on Item system. This adds a representation for this design element in RequisitePro. By doing this, you create a link between the class model that was generated because of the original requirement to the requirement itself. You can trace back that implementation to its source.
2. You'll be prompted with the properties window for a new requirement type in the RequisitePro database. This is the requirement entry that holds the reference to this model element and traces the original requirement. The Traceability tab opens, as shown in Figure 30.

**Figure 30. Adding a trace to requirements**



3. Click **Add** in the From panel to add an existing requirement from your Use Case list to the traceability for this design element.

4. Because you selected the Bid on Item Use Case diagram, select **Bid on Item Use Case**.
5. If you update a design element name or description, select **RequisitePro > Update Traceability** for the component. Alternatively, you can update all of the changes to RequisitePro by selecting **Tools > Rational RequisitePro > Update All Traceability**.

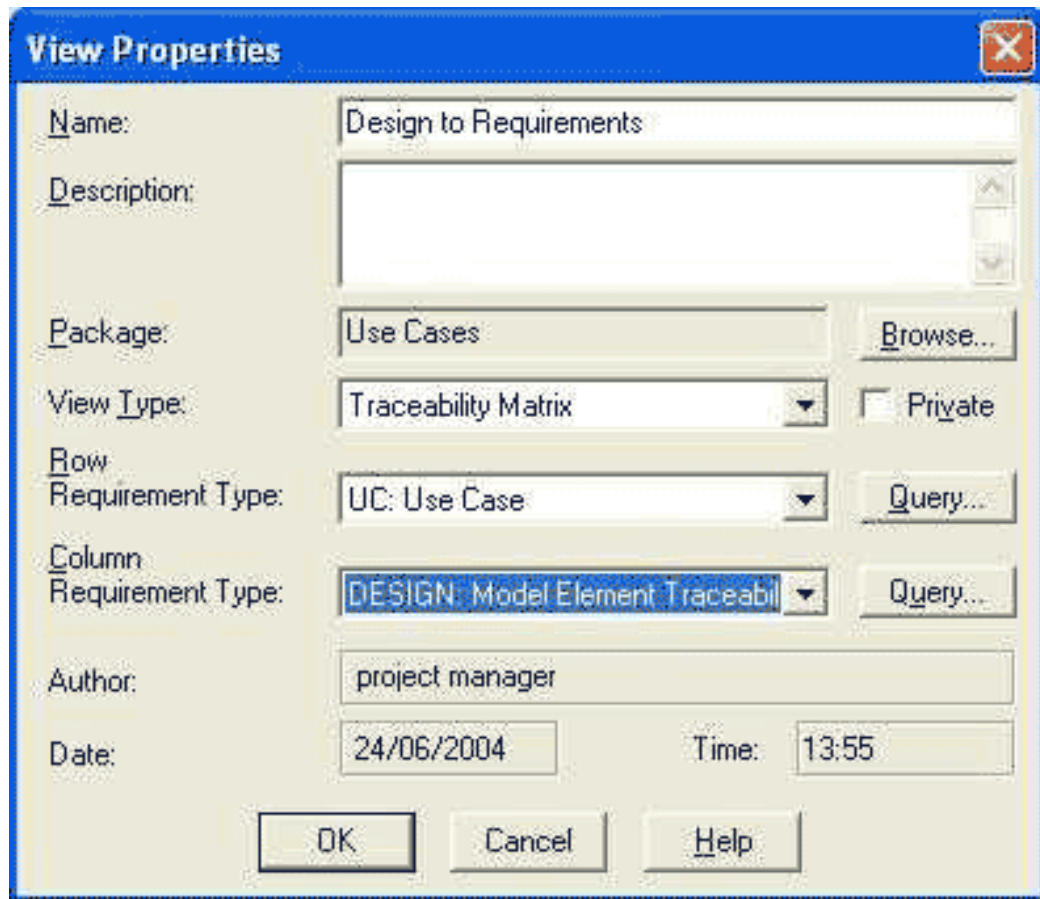
With the links in place, it's time to check that your design and requirements match. To do that, create a traceability matrix.

## 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. To do that, create a Traceability Matrix. This process creates a view that maps between two requirement types and shows us which design component relates to which requirement.

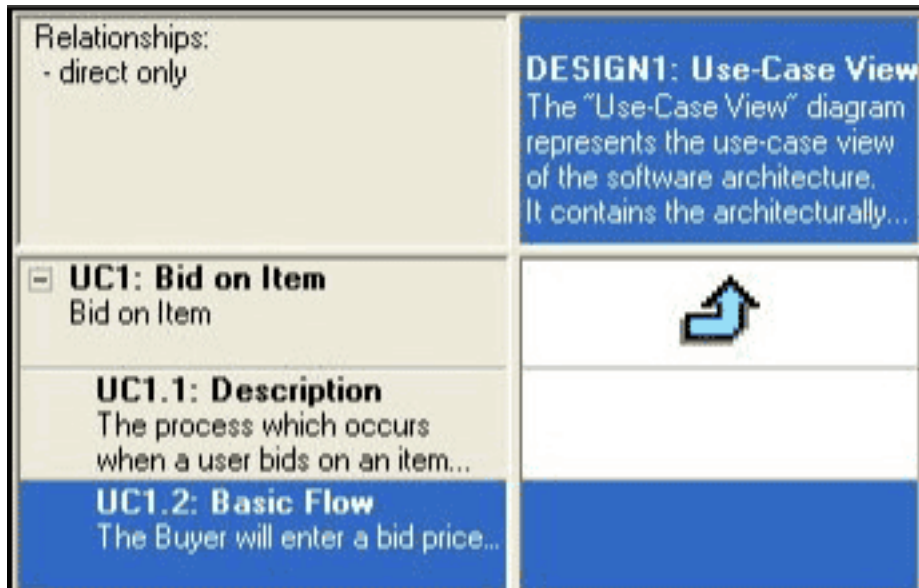
To create a traceability matrix:

1. Right-click on the Use Cases package and select **New > View**.
2. In the View Properties window, name the view and select **Traceability Matrix** as the View Type. See Figure 31.  
**Figure 31. Adding a new view**



3. Choose **Use Case** as the Row Requirement Type and **Design** as the Column Requirement Type. In a very large project you can limit the view to specific use cases by any of its properties by using the Query button to specify which requirements to include in the view.
4. Click **OK** to complete the view. You should have a diagram similar to the one in Figure 32.

**Figure 32. Traceability matrix**



The arrows show the direction of the relationship, from the requirement to the design model. You can see at a glance which components complete the functionality of your requirements. Hopefully, all the original requirements map to an element within your Use Case diagram in Rational XDE. If not, then the designer knows that more work is needed to complete the design phase of the system.

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

## Section 8. Wrap up

### Summary

RequisitePro is an excellent way to record requirements for an application. Rational XDE is an excellent method of 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 was requested by the client.

Once the integration is up and running, through the traceability matrix, you can see at a glance 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 is 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 models in RequisitePro and Rational XDE. As you continue to work through building the Auction application in future parts of this series, you'll see more detail at how to turn your Rational XDE model into a working design and code, and how you can integrate the model and coding process into ClearCase to track changes and revisions.

# Resources

## Learn

- The other tutorials in this series:
  - [Part 2, Integrating IBM Rational XDE Professional into WebSphere.](#)
  - [Part 3, Merging original requirements with changes](#)
  - [Part 4, Tracking changes during the application lifecycle](#)
- [WebSphere Studio Application Developer Library](#) provides a wealth of information, samples, and tutorials for using WebSphere.
- For tutorials and information on Rational XDE:
  - [Modeling with Rational XDE and WebSphere Studio](#)
  - [Rational XDE Model Structure Guidelines for J2EE](#)
  - [Customizing and Automating Rational XDE Developer](#)
  - [Building e-business Applications with IBM Rational XDE and WebSphere](#)
- Find more information on the entire suite of [Rational products](#).
- The [Rational developerWorks site](#) is a portal for more information, tutorials, and articles on the Rational environment.
- New to Rational? [Rational developerWorks](#) can help you get started with Rational products.

## Get products and technologies

- A trial download of RequisitePro is available from <http://www.ibm.com/developerworks/downloads/r/rp/b>.

## Discuss

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

## About the author

Martin C. Brown

Martin C. Brown, a [Studio B](#) author, is a former IT Director with experience in cross-platform integration. A keen developer he has produced dynamic sites for blue-chip customers, including HP and Oracle and is the Technical Director of Foodware.net. Now a freelance

writer and consultant, MC, as he is better known, works closely with Microsoft as an SME, is the LAMP Technologies Editor for LinuxWorld magazine, a core member of the AnswerSquad.com team and has written a number of books on topics as diverse as Microsoft Certification, iMacs and open source programming. Despite his best attempts, he remains a regular and voracious programmer on many platforms and numerous environments. MC can be contacted at [questions@mcslp.com](mailto:questions@mcslp.com), or through this Web site at <http://www.mcslp.com>.