

# **Study of Rational Rose with UML**

A Dissertation Submitted in Partial fulfillment for  
the requirement of the award of Degree of  
**Master of Engineering**  
**in**  
**Computer Technology and Applications**

By:

**TOWFIK JEMAL**  
**30/CTA/03**

**Under the guidance of**

**Prof. ASOK DE**



**DEPARTMENT OF COMPUTER ENGINEERING**  
**DELHI COLLEGE OF ENGINEERING**  
**UNIVERSITY OF DELHI-110042**  
**2003-2005**

## **DECLARATION BY THE CANDIDATE**

I hereby certify that the work which is being presented in the dissertation entitled “**Study of Rational Rose with UML**”, in the partial fulfillment of the requirements for the award of the degree of Master of Engineering in Computer Technology and Applications, is an authentic record of my work carried out under the supervision of **Dr. Asok De (Prof.)**. I have not submitted this work anywhere else for the award of any other degree.

(Towfik Jemal)

M.E. (CTA)

Department of Computer Engineering.

DCE, Delhi-1100042

## CERTIFICATE

This is to certify that declaration made by the candidate is correct to the best of my knowledge and belief. This is to certify that the project entitled **"Study of Rational Rose with UML"** is an authentic record of candidate's own work carried out by him under my guidance and supervision .He has not submitted this work for the award of any other degree.

Dr. Asok De (Prof.)

Head  
Department of IT and project guide  
DCE, Delhi-110042

Dr. D. Roy Choudhury (Prof.)

Head  
Department of Computer Engineering  
DCE, Delhi-1100042

## **ACKNOWLEDGEMENTS**

I am delighted to express my heartily and sincere gratitude and indebtedness to Prof Asok De, Head Information Technology Department, Delhi College of Engineering, Delhi for his invaluable guidance and wholehearted cooperation. His continuous inspiration only has made me complete this project.

I am greatly thankful to Prof. D. Roy Choudhary Head of Department and Dr. Goldie Gabrani Project Coordinator, for their support in providing resource. My heartily thanks to all professors for their expertise and all rounded personality they have imparted to me.

(TOWFIK JEMAL)

# Table of Contents

Abstract

i

## 1. Introduction

1-2

## 2. Overview of Rational Rose

3-11

### 2.1. Application Window

3

## 3. Use Case Diagram and Specification

12-25

### 3.1. System behavior

12

#### 3.1.1. Actors

12

#### 3.1.2. Use Cases

12

### 3.2. Creating and documenting actors and use cases

14

#### 3.2.1. Creating Actors

14

#### 3.2.2. Actor documentation

14

#### 3.2.3. Creating Use Cases

15

#### 3.2.4. Use Case documentation

15

### 3.3. Flow of events

16

#### 3.3.1. Linking flow of Events to Use Cases

18

### 3.4. Use Case Relationship

19

### 3.5. Use Case Diagram

20

#### 3.5.1. Use Case Toolbox

20

#### 3.5.2. Creating the main Use Case diagram

21

### 3.6. Use Case Specification

5

	22
<b>4. Class Diagram and Specification</b>	<b>26-39</b>
4.1. Overview	26
4.2. Class diagram	30
4.2.1. Class diagram Toolbox	30
4.2.2. Adding and Hiding classes and Filtering class relationship	31
4.2.3. Class Specification	31
<b>5. Interaction Diagram and Specification</b>	<b>40-54</b>
5.1. Interaction diagram overview	40
5.2. Use Case realization	40
5.3. Collaboration diagram	41
5.4. Sequence diagram	42
5.5. Toolbox	43
5.5.1. Collaboration diagram toolbox	43
5.5.2. Sequence diagram toolbox	44
5.6. Specification	44
5.6.1. Object Specification	44
5.6.2. Class Instance Specification	46
5.6.3. Link Specification	47
5.6.4. Message Specification	50
5.7. Sequence diagram for JU course registration system	53
<b>6. Relationship Specification</b>	<b>55-57</b>

6.1. Association Relationship	55
6.2. Aggregation Relationship	55
6.3. Naming Relationship	56
6.4. Role Name	56
6.5. Multiplicity Indicator	56
<b>7. State Machine Diagram and Specification</b>	<b>58-70</b>
7.1. Overview	58
7.2. State machine specification	58
7.3. State chart diagram	58
7.3.1. States	59
7.3.2. States transition	59
7.3.3. Special states	60
7.4. Activity diagram	61
7.4.1. Activities	61
7.4.2. Transitions	62
7.4.3. Decision points	62
7.4.4. Synchronization bar	62
7.4.5. Swimlanes	63
7.4.6. Initial and final activities	63
7.4.7. Object flow	64
7.5. Swimlane specification	64
7.6. State and activity specification	65
7.7. Action specification	65

7.8. State transition specification	68
7.9. Decision specification	69
7.10. Synchronization specification	70
7.11. Object flow specification	70
<b>8. Component Diagram and Specification</b>	<b>71-78</b>
8.1. Component diagram overview	71
8.2. Source code component	71
8.3. Component diagram toolbox	72
8.4. Component specification	72
8.5. Package specification	76
8.6. Software component for the JU course registration system	77
<b>9. Deployment Diagram and Specification</b>	<b>79-82</b>
9.1. Deployment diagram overview	79
9.2. Deployment diagram toolbox	79
9.3. Processor specification	79
9.4. Device specification	81
9.5. Process specification	81
9.6. Deployment diagram for the JU course registration system	82
<b>10. Code Generation and reverse engineering with Visual Basic</b>	<b>83-89</b>
10.1. Code generation	83
10.2. Reverse engineering	85



### 10.3. Visual Basic code for Select Course to Teach use case

86

## **11. Rational RequisitePro**

**90-118**

### 11.1. Overview

90

### 11.2. Overview

93

#### 11.2.1. Project template

95

### 11.3. Working with view

96

#### 11.3.1. The Attribute Matrix

96

#### 11.3.2. The Traceability Matrix

98

#### 11.3.3. The Traceability Tree

100

#### 11.3.4. Creating Views

102

### 11.4. Querying and Searching

103

#### 11.4.1. Querying overview

103

#### 11.4.2. Creating and modifying queries

103

#### 11.4.3. Navigating to a Requirement using the Go To command

104

#### 11.4.4. Reviewing a project with cross-project traceability

104

#### 11.4.5. Requirement Metrics

105

### 11.5. Discussion in RequisitePro

106

#### 11.5.1. Viewing discussions

106

#### 11.5.2. Configuring E-mail to discussion

106

#### 11.5.3. Creating discussion

107

#### 11.5.4. Reading discussion

108

#### 11.5.5. Responding to discussion

108

### 11.6. Documents in RequisitePro

11.6.1. Creating RequisitePro documents	109
11.6.2. Microsoft Word	110
11.7. Requirements	111
11.7.1. Creating Requirements	112
11.7.2. Inserting Microsoft word-linked Files in requirement	112
11.7.3. Creating requirements in view	113
11.8. Hierarchy	114
11.8.1. Child requirements	114
11.8.2. Peer-requirements	114
11.8.3. Suspect relationship	114
11.9. Traceability	115
11.9.1. Traceability in view	115
11.9.2. Suspect relationship	116
11.10. Importing requirements and documents	116
11.10.1. Preparing to Import	117
<b>12. Conclusions</b>	117
<b>Bibliography</b>	119
	120

## **ABSTRACT**

Rational Rose is the visual modeling software solution that creates, analyze, design, view, modify, and manipulate components. Visual modeling is the mapping of real world processes of a system to a graphical representation.

An overview of the behavior of the system being developed can be depicted graphically with a use-case diagram. Rational Rose provides the collaboration diagram as an alternative to a use-case diagram. It shows object interactions organized around objects and their links to one another. The statechart diagram provides additional analysis techniques for classes with significant dynamic behavior. A statechart diagram shows the life history of a given class, the events that cause a transition from one state to another and the actions that result from a state change. Activity diagrams provide a way to model a class operation or the workflow of a business process.

Rational Rose provides the notation needed to specify and document the system architecture. The logical architecture is captured in class diagrams that contain the classes and relationships that represent the key abstractions of the system under development. The component architecture is captured in component diagrams that focus on the actual software module organization within the development environment. The deployment architecture is captured in deployment diagrams that map software to processing nodes, showing the configuration of run-time processing elements and their software processes. The way how code is generated is also included in this project.

RequisitePro helps projects succeed by giving teams the ability to manage all project requirements comprehensively and facilitating team collaboration and communication. RequisitePro combines both document-centric and database-centric approaches. By deeply integrating Microsoft Word with a multi-user database, RequisitePro organize, prioritize, trace relationships, and easily track changes to the requirements. The program's unique architecture and dynamic links make it possible to move easily between the requirements in the database and their presentation in Word documents. We have studied everything mentioned above and used JU registration system to show how to use rational rose and RequisitePro for implementation.

# 1. Introduction

Rational Rose is the visual modeling software solution that creates, analyze, design, view, modify, and manipulate components. The Rational Rose product family is designed to provide the software developer with a complete set of visual modeling tools for development of robust, efficient solution to real-time systems environment. Rational Rose share a common universal standard, making modeling accessible to nonprogrammers wanting to model business processes as well as to programmers modeling application logic.

Visual modeling is the mapping of real world processes of a system to a graphical representation. Models are useful for understanding problems, communicating with everyone involved with the project (customers, domain experts, analysts, designers, etc.), modeling complex systems, preparing documentation, and designing programs and databases. Modeling promotes better understanding of requirements, cleaner designs, and more maintainable systems. Increasing complexity, resulting from a highly competitive and ever changing business environment, offers unique challenges to system developers. Models help to organize, visualize, understand, and create complex things.

As software systems become more complex, cannot understand them in their entirety. To effectively build a complex system, the developer begins by looking at the big picture without getting caught up in the details. A model is an ideal way to portray the abstractions of a complex problem by filtering out nonessential details. Abstraction is a fundamental human capability that permits us to deal with complexity. The developer must abstract different views or blueprints of the system, build models using precise notations, verify that the models satisfy the requirements of the system, and gradually add detail to transform the models into an implementation.

Visual modeling has one communication standard, the Unified Modeling Language (UML). The UML provides a smooth transition between the business domain and the computer domain. Using the UML, all members of a design team can work with a common vocabulary, minimizing miscommunication and increasing efficiency.

Visual modeling captures business processes by defining the software system requirements from the user's perspective. This streamlines the design and development process. Visual modeling also defines architecture by providing the capability to capture the logical software architecture independent of the software language. This method provides flexibility to the system design since the

logical architecture can always be mapped to a different software language. Finally, with visual modeling, parts of a system or an application can be reused by creating components of the design. These components can then be shared and reused by different members of a team allowing changes to be easily incorporated into already existing development software.

Through out our study project we have used JU university registration system as an example. In JU, at the beginning of each semester, students may request a course catalog containing a list of course offerings for the semester. Information about each course, such as professor, department, and prerequisites will be included to help students. The new system allows students to select 21 credit hour course offerings for the coming semester. Once the registration process is completed for a student, the registration system sends information to the billing system so the student can be billed for the semester, i.e. the student name is registered by the billing system for loan so that they will pay after graduation. Professors must be able to access the online system to indicate which courses they are going to teach, and to see which students signed up for their course offerings. For each semester, there is a period of time that students can change their schedule. Students must be able to access the system during this time to add or drop courses.

For the purpose of comparison we have also studied Rational RequisitePro in this project. The primary objective of Rational RequisitePro is to manage requirements so that the system developers provide quality software, on time, and on budget to the users.

## 2. Overview of Rational Rose

When Rational Rose is started, some editions display a Framework dialog box. From this dialog box, a model can be loaded with predefined model elements, allow focusing the modeling efforts on the parts that are unique to the system.

Independent of Frameworks, Rational Rose's graphical user interface can be used to display, create, modify, manipulate, and document the elements in a model using these windows:

- Application window
- Browser window
- Documentation window
- Diagram window
- Overview window
- Specification window
- Log window

Rational Rose displays the diagram, specification, and documentation windows within the application window. The log window is a dockable window can be moved, docked or undocked, or closed.

### 2.1 Application Window

An application window contains a title bar, menu bar, toolbar, and a work area where the toolbox, browser, documentation window, diagram window, and specification window appear.

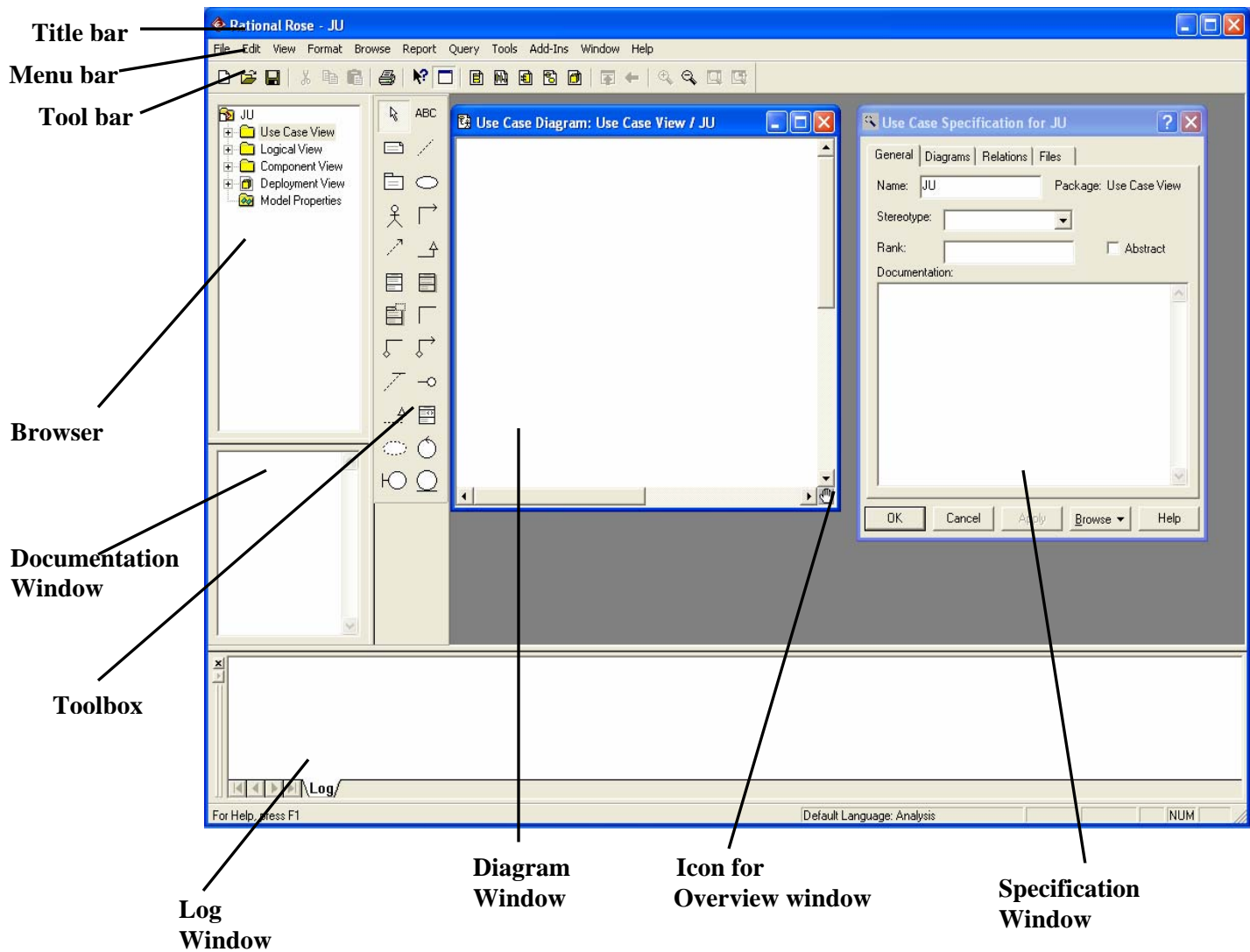


Figure 2.1 Applications Window

### Title Bar

The title bar always displays the diagram type. Additional information (like the view or diagram name) is often displayed depending on the diagram/model being viewed. The title bar includes a Control Menu box, Minimize button, Restore button, and Close button.

### Control-Menu Box

Clicking the Control-Menu box (on the application or diagram window) displays a menu with the following commands:

**Restore** Restores focus to that diagram window.

- Move**            Highlights the border of the window. Move the pointer to the Title Bar, click and drag the window to the desired location.
- Size**            Highlights the border of the window. Move the pointer to the border and resize the window as desired.
- Minimize**       Reduces the window to an icon placing it in the bottom of the application window.
- Maximize**       Enlarges the window to fit the entire screen.
- Close**            Closes the window.

**Minimize, Restore, and Close Buttons**

These buttons allows minimizing, restoring, or closing the diagram or application window.

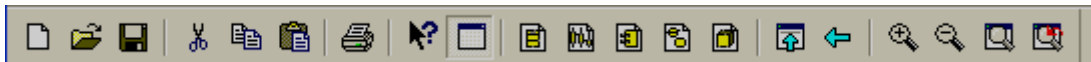
**Menu Bar**

The menu bar changes depending on which diagram is opened.

**Toolbar**

The standard toolbar is displayed directly under the menu bar, along the top of the application window. This toolbar is independent of the open diagram window.

The following icons are available for use on the standard toolbar.



*Figure 2.2. Standard toolbar*

 **New Model**

Clicking the New Model icon creates a new model.

 **Open Model**

Clicking the Open Model icon opens the Load Model dialog box. Open a model from anywhere within the design.





### **Save Model or Log**

Clicking the Save Model icon opens the Save Model to dialog box. Enter a new file name. After the model is named and saved, clicking this icon automatically saves changes to the current model without displaying the dialog box. This will also save the log if the log window is open.



### **Cut**

Clicking the Cut icon removes icons from the model. Element(s) must be selected to activate the icon. Cutting an element will also cut associated relationships. Cutting multiple selected items is possible.



### **Copy**

Clicking the Copy icon copies an element to a new location on the same model, or to a new model, without affecting the original model.



### **Paste**

Clicking the Paste icon pastes a previously cut or copied element on the Clipboard onto another location.



### **Print Diagrams**

Clicking the Print icon prints diagrams to the default printer.



### **Context Sensitive Help**

Clicking the Context Sensitive Help icon makes all topics covered in the online Help available. Click this icon and then click the item with which help is needed.



### **View Documentation**

Clicking the View Documentation icon displays the documentation window on the diagram.



### **Browse Class Diagram**

Clicking the Browse Class Diagram icon opens the Select Class Diagram dialog box.



### **Browse Interaction Diagram**

Clicking the Browse Interaction Diagram icon opens the Select Interaction Diagram dialog box.



### **Browse Component Diagram**

Clicking the Browse Component Diagram icon opens the Select Component Diagram dialog box.



### **Browse State Machine Diagram**

Clicking the Browse State Machine Diagram icon opens the Select Statechart Diagram or Activity Diagram dialog box.



### **Browse Deployment Diagram**

Clicking the Browse Deployment Diagram icon opens the Deployment Diagram dialog box.



### **Browse Parent**

Clicking the Browse Parent icon displays the “parent” of the selected diagram or specification. If a specification is selected, the specification for the parent of the “named” item is displayed.



### **Browse Previous Diagram**

Clicking the Browse Previous Diagram icon displays the last displayed diagram.



### **Zoom In**

Clicking the Zoom In icon magnifies the current diagram to view an area in detail.



### **Zoom Out**

Clicking the Zoom Out icon minimizes the current diagram allow to view more information.



### **Fit in Window**

Clicking the Fit in Window icon centers and displays a diagram within the limits of the window. This command changes the zoom factor so that the entire diagram appears.



### **Undo Fit in Window**

Clicking the Undo Fit in Window icon undoes the actions performed on the previous Fit In Window command.

## Toolbox

The diagram toolbox consists of tools that are appropriate for the current diagram. Changing diagrams automatically displays the appropriate toolbox.

When a modifiable diagram window is active, a toolbox with tools appropriate for the current diagram is displayed. If the current diagram is contained by a controlled unit or the model is write protected, the toolbox is not displayed. While each diagram has a set of tools applicable for the current diagram, all toolboxes have the Selector, Separator, and Lock icons.

The selector icon is used to select icons on the diagram. This icon cannot be removed from the toolbox.

The separator icon is used to put a small space between icons on the toolbox. There can be as many separators as wanted, but there must be at least one.

The lock icon can be set to locked or unlocked. In the locked mode, any tool icon stays in the selected state until the diagram loses focus or another tool button is selected. This option facilitates the rapid placement of several identical icons without repeatedly returning to the diagram toolbox.

To access the Customize Toolbar dialog box in order to modify the displayed toolbox:

- Right-click anywhere on the toolbox and then click Customize from the shortcut menu.
- Double-click anywhere on the toolbox not occupied by a button.
- Click **View: Toolbars: Configure**.
- Click **Tools: Options**. On the Option dialog box, click the Toolbars tab. This approach gives the ability to modify all the diagram toolboxes without first displaying a specific diagram type.

## Browser

The browser is a hierarchical navigational tool that allows to view the names and icons of interaction, class, use case, statechart, activity, and deployment diagrams as well as many other model elements. When a class or interface is assigned to a component, the browser displays the assigned component name in an extended name.

## **Documentation Window**

The documentation window is used to describe model elements or relationships. The description can include information such as the roles, keys, constraints, purpose, and essential behavior of the element. Information can be typed either here or through the documentation field of a specification.

To view the documentation window, click View: Documentation. A check mark next to documentation indicates the window is open. Only one documentation window can be open at a time, but if different items are selected, the window will be updated accordingly.

## **Log Window**

Rose uses the log window to report progress, results, and errors that occur as a result of a command or action in the model. The messages posted to the log are prefixed with a time stamp, enabling to track when an event or action occurred.

Like the documentation window, the log window can be docked or floated. This window can be dock or undock by right-clicking anywhere in the window and toggling Allow Docking. When docked, the log window is positioned along the border of the application window. If docking is not enabled or if the window dragged outside of the application frame, the window is floating. A floating window is always on top.

In addition, the log window can be hide by right clicking anywhere in the window and clicking Hide. To redisplay the window, click View: Log.


## **Diagram Window**

Diagram window allows creating and modifying graphical views of the current model. Each icon in a diagram represents an element in the model. Since diagrams are used to illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams. This means it can be possible to control which elements and properties appear on each diagram.

## **Overview Window**

The overview window is a navigational tool that helps to move to any location on all Rational Rose diagrams. When a diagram is larger than the viewable area within the diagram window, it is not possible to see the whole diagram without scrolling. The overview window provides a scaled-down view of the current diagram so the entire diagram can be seen.

To move to an exact area of the diagram, the following steps should be used:

- Move the pointer over the hand  located in the lower, right side of the diagram window. Notice that the pointer appears as a + when the pointer is located over the active hand.
- Click on the hand icon so the overview window appears.

### Specification Window

A specification enables to display and modify the properties and relationships of a model element, such as a class, a relationship, an operation, or an activity. The information in a specification is presented textually; some of this information can also be displayed inside icons representing the model element in diagrams.

Properties or relationships can be changed by editing the specification or modifying the icon on the diagram. The associated diagram or specification is automatically updated.

To display a specification:

- Right-click the icon in either the diagram or browser, and then click **Open Specification** from the shortcut menu.
- Click the icon in either the diagram or browser, and then click **Browse: Specification**.
- Double-click on the icon in either the diagram or browser.

### Printing diagrams and Specifications

The Print dialog box allows to print diagrams and specifications. Table 1 describes the tabs in the Print dialog box.

*Table 1 Print Dialog Tabs*

<b>Tab</b>	<b>Description</b>
General	Allows to specify a printer, a selection of diagrams and specifications, and the number of copies to be printed.
Diagrams	Allows selecting and viewing a list of diagrams to be printed.
Specification	Allows selecting and viewing a list of specifications to be printed.
Layout	Allows to select layout settings for printing diagrams and specifications.

### **Print Preview**

The print preview option allows to see how a diagram will appear when printed. Also, print preview displays the total number of pages the diagram will take to print on the status bar.

### **Zoom In**

To view a diagram at different magnified sizes, click either **Zoom In** or **Zoom Out**. Clicking on any part of the diagram is also used to get a magnified view.

### **Close**

Click Close to return to an active window.

### **Apply Filter Dialog Box**

The Apply Filter dialog box allows searching for diagrams and specifications within the model. The filter is especially useful when diagrams are printed from large models.

To print a specific diagram in a model, type in the name, type, or path of the diagram to be printed.

**Name**            Provides a list of all diagram names depending on search criteria.

**Type**            Provides a list of all diagram types depending on search criteria.

**Path**            Provides a list of each path for diagrams displayed.

## 3. Use Case Diagrams and Specifications

### 3.1 System behavior

The behavior of the system under development (i.e. what functionality must be provided by the system) is documented in a use case model that illustrates the systems intended functions (use case), its surroundings (actors), and relationships between the use cases and actors (use case diagram).

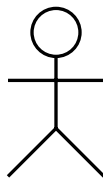
#### 3.1.1 Actors

Actors represent system users (i.e. anyone or any thing that must interact with the system). They help define the system and give a clear picture of what the system should do. An actor may:

- Only input information to the system.
- Only receive information from the system.
- Input and receive information to and from the system.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.



*Figure 3.1 UML notation for an actor*

#### 3.1.2 Use Case

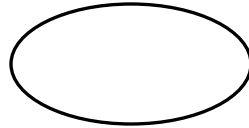
A use case is a sequence of events (transaction) performed by a system in response to a trigger initiated by an actor. A use case contains all the events that can occur between an actor-use case pair. They represent the functionality provided by the system, that is, what capabilities will be provided to an actor by the system.

In its simplest form, a use case can be described as a specific way of using the system from a user's (actor's) perspective. Use cases provide a means to:

- Capture system requirements.

- Communicate with the end users and domain experts.
- Test system.

Use cases are best discovered by examining what the actor's needs and defining what the actor is able to do with the system, this helps ensure that the system is what the user expects. Use case names often start with a verb.



*Figure 3.2 UML notations for a use case*

### **Actors in JU course registration system**

In JU:

- Students want to register for courses.
- Professors want to select courses to teach.
- The registrar must create the curriculum and generate a catalog for the semester.
- The Billing system must receive billing information from the system.

Based on this the following actors have been identified: student, professor, registrar, and the Billing system.

### **Use cases in JU course registration system**

The following needs must be addressed by the system under development:

- The student actor needs to use the system to register for course.
- The professor actor needs to use the system to select courses to teach for a semester, and must be able to receive a course roster from the system.
- After the course selection process is completed, the Billing system must be supplied with billing information.
- The registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the professors needed by the system.

Based on these needs, the following use cases have been identified:

- Register for Courses



- Select Courses to Teach
- Request Course Roster
- Maintain Course Information
- Maintain Professor Information
- Maintain Student Information
- Create Course Catalog.

## 3.2 Creating and documenting actors and use cases

### 3.2.1 Creating actors

To create an actor:

- Right-click on the use case view packages in the browser to make the shortcut menu visible.
- Select the **New: Actor** menu option. A new actor called NewClass is placed in the browser.
- With the actor called NewClass selected, enter the desired name of the actor.

### 3.2.2 Actor documentation

A brief description for each actor should be added to the model. The description should identify the role the actor plays while interacting with the system.

The actor's descriptions for the JU Course Registration system are:

- Student: - a person who is registered to take classes at University.
- Professor: - a person who is certified to teach classes at University.
- Registrar: - the person who is responsible for the maintenance of the University Course Registration system.
- Billing system: - the external system responsible for student billing.

To document an actor:

- Open the documentation window by clicking the documentation menu choice from the view menu if it is not already opened.
- Click to select the actor in the browser.
- Position the cursor in the documentation window and enter the documentation.

### 3.2.3 Creating Use Cases

To create a use case:

- Right-Click on the use case view in the browser to make the shortcut menu visible.
- Select the **New: Use Case** menu option. A new unnamed use case is placed in the browser.
- With the use case selected, enter the name of the use case.

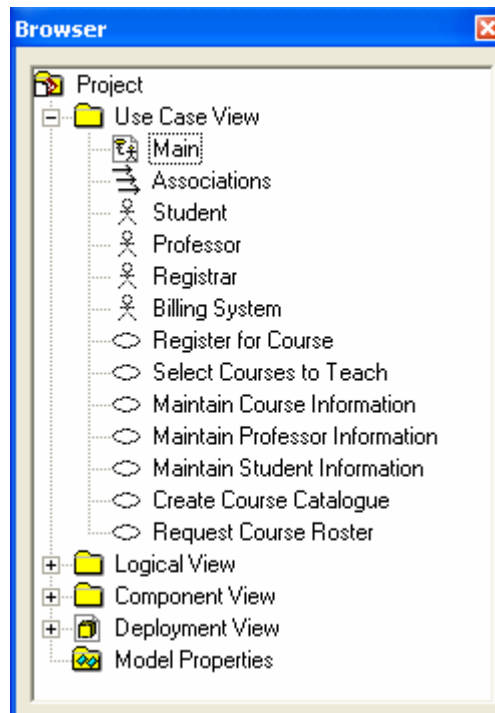


Figure 3.3 JU Actors and Use Cases

### 3.2.4 Use Case documentation

The brief description of a use case states the purpose of the use case in a few sentences, providing a high level definition of the functionality provided by the use case.

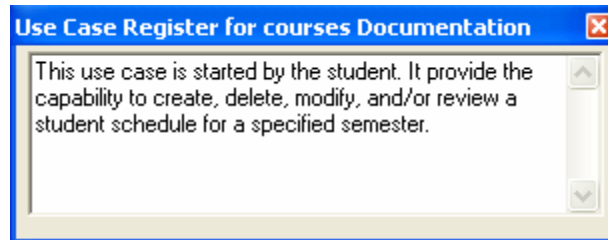
The brief description of the Register for Courses use case is:

*This use case is started by the student. It provides the capability to create, modify, and/or review a student schedule for a specified semester.*

To document a use cases:

- Click to select the use case in the browser.

- Position the cursor in the documentation window and enter the brief description for the use case.



*Figure 3.4. Register for Course use case description*

### 3.3 Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as a separate file of documents in any text editor and then attached or linked to a use case. The flow of events for a use case is a description of the events needed to accomplish the required behavior of the use case.

A flow of event should include:

- When and how the use case starts and ends.
- What interaction the use case has with the actors.
- Data needed by the use case.
- Normal sequence of events for the use case.
- An alternate or exceptional flow.

The flow of events for a use case is contained in a document called use case specification. Each project should use a standard template for the creation of the use case specification. We used the template from the rational unified process.

- 1.0 Use case Name
- 1.1 Brief Description
- 2.0 Flow of Events
  - 2.1 Basic Flow
  - 2.2 Alternate Flows
    - 2.2. X < Alternate Flow X >
- 3.0 Special Requirements

- 3. X <Special Requirement X >
- 4.0 Preconditions
- 4. X < Precondition X >
- 5.0 Post condition
- 5. X < Post condition X >
- 6.0 Extension Points
- 6. X < Extension Point X >

Use case specification for the **Select Courses to Teach** use case

### **1.0 Use case name**

Select Courses to Teach

### **1.1 Brief Description**

This use case is started by professor. It provides the capability for the professor to select up to three courses to teach for a selected semester.

### **2.0 Flow of Events**

#### **2.1 Basic Flow**

This use case begins when the professor logs onto the Registration system and enters his/her password. The system verifies that the pass word is valid (if the password is invalid, Alternate Flow 2.2.1 is executed) and prompts the professor to select for the future semester (if an invalid semester is entered, Alternate Flow 2.2.2 is executed).The professor enters the desired semester. The system prompts the professor to select the desired activity: ADD and DELETE.

If the activity selected is ADD, the systems display the course screen containing a field of course name and number. The professor enters the name and number of courses (If an invalid name/number combination is entered, Alternate Flow 2.2.3 is executed).

If the activity selected is DELETE, the system displays the course offering screen containing a field for a course offering name and number. The professor selects the name and number of course offering (if an invalid name/number combination is entered, Alternate Flow 2.2.3 is executed). The use case then begins again.

#### **2.2 Alternate Flows**

##### **2.2.1 Invalid Password**

An invalid password is entered. The user can re-enter a password or terminate the use case.

##### **2.2.2 Invalid Semester**

The system informs the user that the semester is invalid. The user can re-enter the semester or terminate the use case.

### **3.3 Special Requirements.**

There is no special requirement for this use case.

#### **4.0 Pre-condition**

There is no precondition.

#### **5.0 Post-condition**

There is no post-condition.

#### **6.0 Extension point**

There is no extension point.

### **3.3.1 Linking flow of Events Document to use cases**

To link flow of events document to use case:

- Right-click on the use case in the browser to make the shortcut menu visible.
- Select the open specification menu option.
- Select the Files tab.
- Right-click to make the shortcut menu visible.
- Select the Inset file menu option.
- Browse to the appropriate directory and select the desired file
- Click the open button.
- Click the OK button to close the specification.

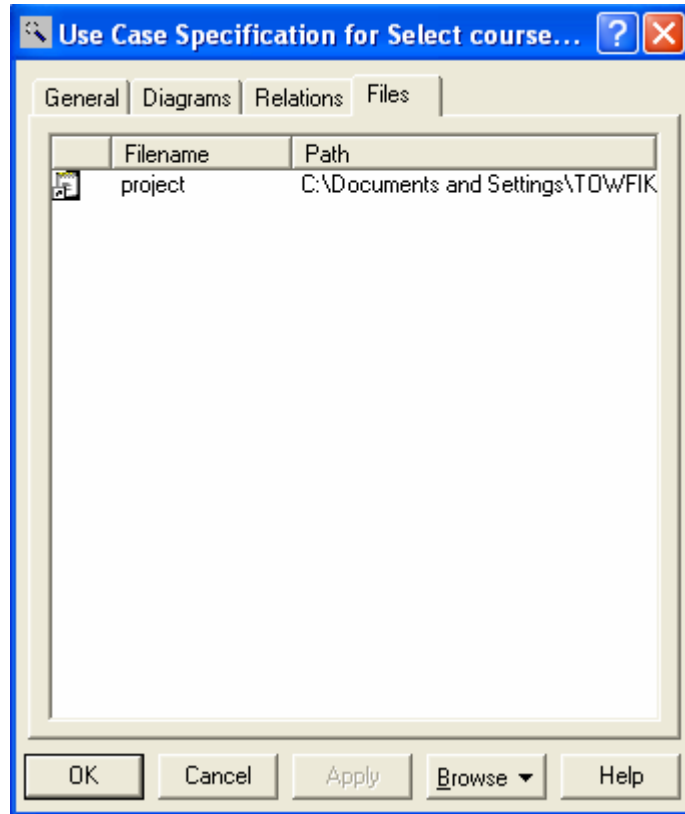


Figure 3.5 Linked Flow of events Document.

### 3.4 Use case Relationships

An association relationship exists between an actor and a use case. This type of association is often referred to as a communicate association since it represents communication between an actor and a use case. An association can be navigable in both direction (actor to use case and use case to actor) or it can be navigable in only one direction. The navigation direction of an association represents who is initiating the communication. An association is represented as a line connecting the related elements. There are two types of relationships that exist between use cases: **include** and **extend**. Multiple use case may share pieces of the same functionality. This functionality is placed in a separate use case rather than documenting it in every use case that needs it. Include relationships are created between the new use case and any other use case that “uses” its functionality. An extend relationship is used to show optional behavior, behavior that may be run based on actor.

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning.

A generalization relationship is a relationship between a more general class or use case and a more specific class or use case. A generalization is shown as a solid line path from the more specific element to a more general element. The tip of a generalization is a large hollow triangle pointing to the more general element.

### 3.5 Use case diagram

Use case diagrams presents a high-level view of how a system is used as seen from an outsider's (or actor's) perspective. These diagrams graphically depict system behavior. A use case diagram may depict all or some of the use cases of a system.

A use case diagram can contain:

- Actors (things outside the system).
- Use case (system boundary identifying what the system should do).
- Interactions or relationships between actors and use cases in the system.

Use case diagrams can be used during analysis to capture the system requirements and understand how the system should work. During the design phase, use case diagrams can be used to specify the behavior of the system as implemented.

#### 3.5.1 Use Case Tool Box

The graphic below shows all the tools that can be placed on the use case diagram toolbox. The application window displays the following toolbox when the current window contains a use case diagram and As Unified is selected from the View menu.

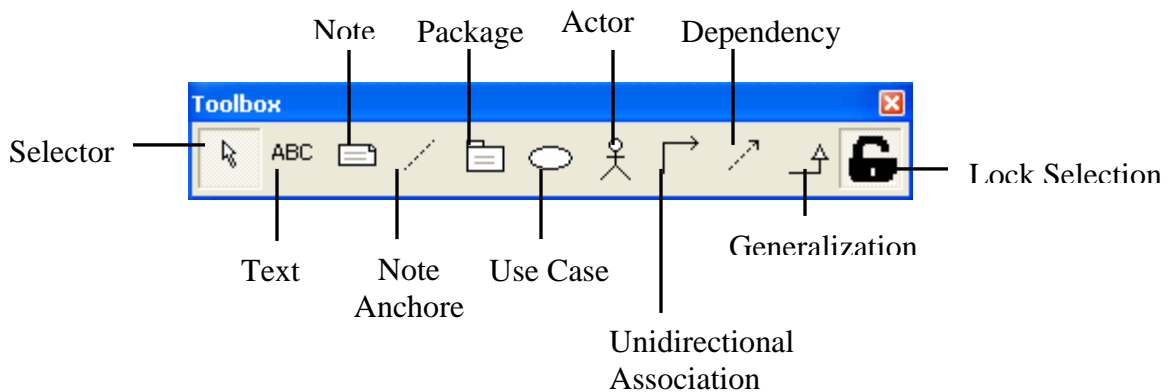


Figure 3.6 Use Case diagram tool Box

### 3.5.2 Creating the main use case diagram

To create main use case diagram:

- Double-click on the main diagram in the use case view in the browser to open the diagram.
- Click to select an actor in the browser and drag the actor on to the diagram.
- Repeat the above step for each additional actor needed in the diagram.
- Click to select a use case in the browser and drag the use case onto the diagram. Repeat for each additional use case needed in the diagram.

To Create Include Relationships:

- Click to select the dependency icon from the toolbar.
- Click on the base use case and drag the Dependency icon the used use case.
- Double-click on the dependency arrow to make the specification visible.
- Click the arrow in the stereotype field to make the drop-down menu visible, and select include.
- Click the OK button to close the specification.

To Create extend relationships:

- Click to select the Dependency icon from the toolbar.
- Click on the use case containing the extended functionality and drag the dependency icon to the base use case.
- Double-click on the dependency arrow to make the specification visible.
- Click the arrow in the stereotype field to make dropdown menu visible and select extend.
- Click the OK button to close specification



The main use case diagram for the JU course Registration system is:

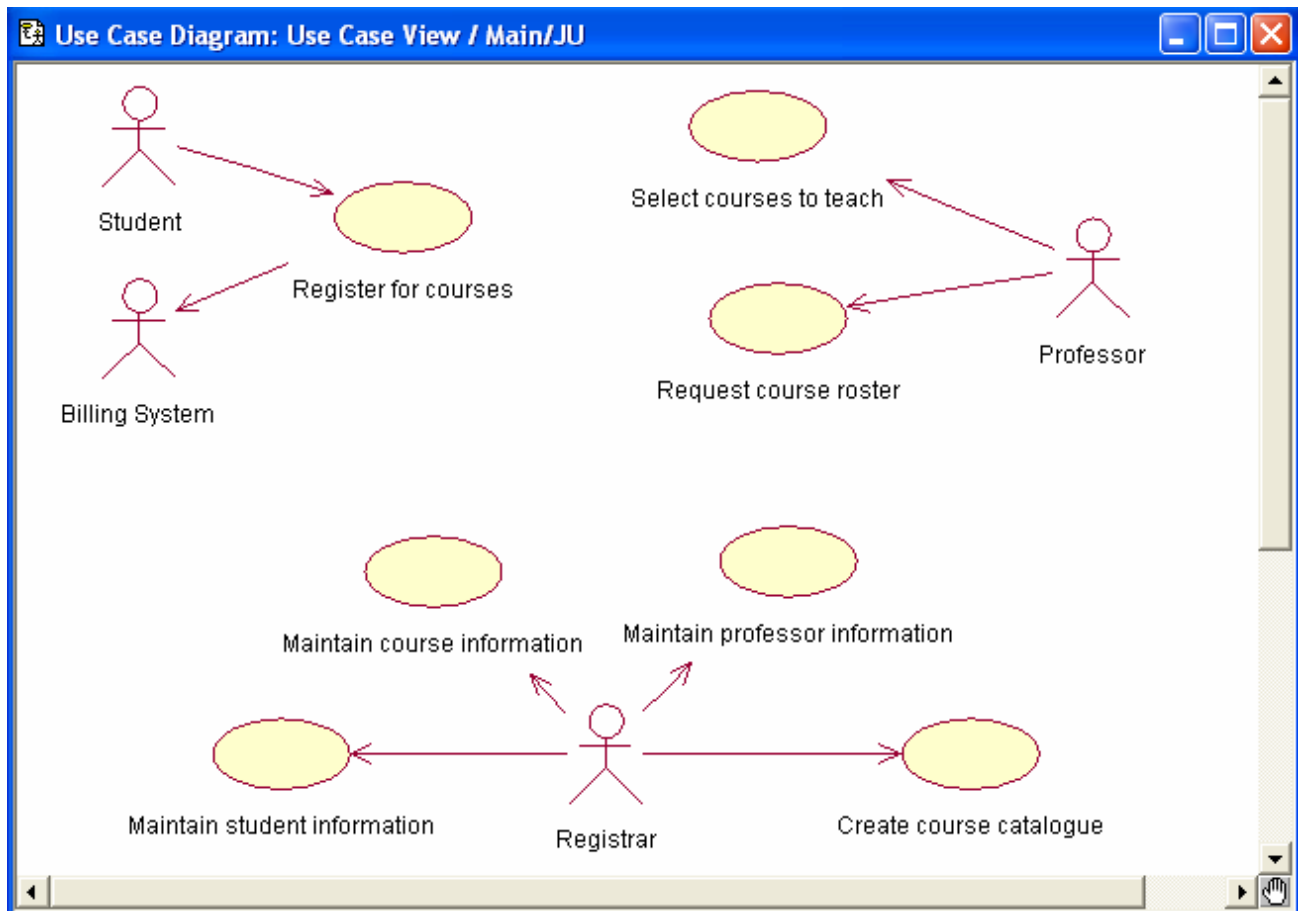


Figure 3.7 Main JU course Registration use case diagram

### 3.6 Use Case Specification

A Use Case Specification allows to display and modify the properties and relationships of a use case in the current model. The Use Case Specification contains the following tabs: General, Diagram, Relations, and Files.

## Use Case Specification-General Tab

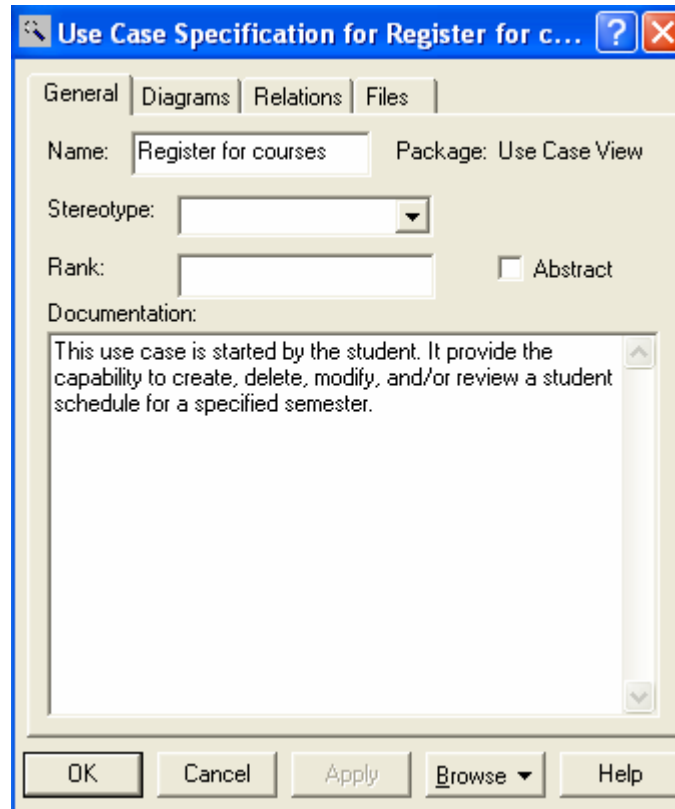


Figure 3.8 Use Case Specification-General Tab

### **Name**

A use case name is often written as an informal text description of the external actors and the sequences of events between elements that make up the transaction. Use case names often start with a verb. The name can be entered or changed on the specification or directly on the diagram.

### **Package**

This static field identifies the package to which the components belong.

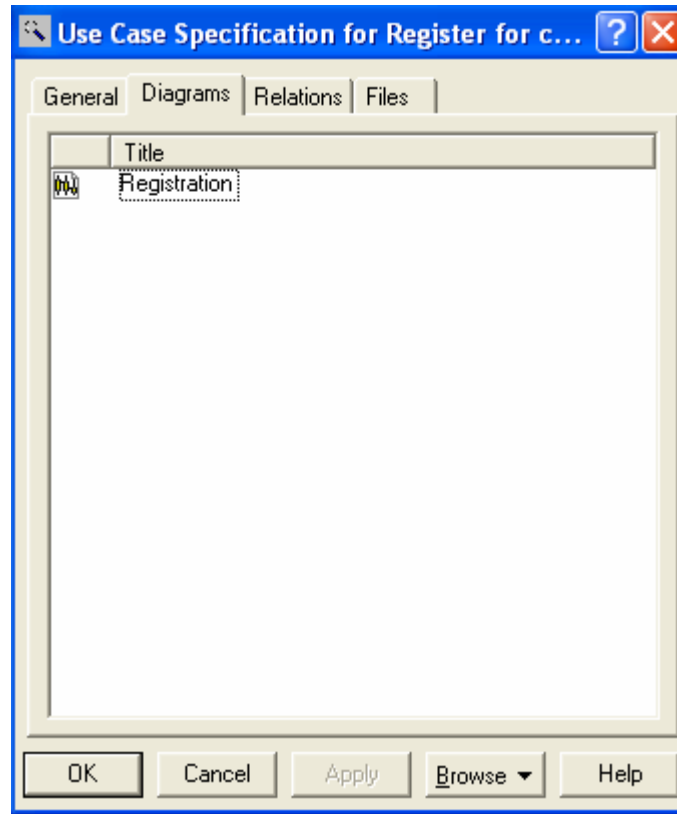
### **Rank**

The Rank field prioritizes use cases. For example, the rank field can be used to plan the iteration in the development cycle at which a use case should be implemented.

### **Abstract**

An abstract notation indicates a use case that exists to capture common functionality between use cases (uses) and to describe extensions to a use case (extends).

## Use Case Specification-Diagram Tab



*Figure 3.9 Use Case Specification-Diagram Tab*

### **Diagram List**

The diagram list contains all the diagrams owned by the use case. The diagram list consists of two columns. The first (unlabeled) column displays the diagram icon type for the diagram. The second column displays the diagram name. To insert a new diagram in the list, click one of the Insert choices in the shortcut menu that corresponds to the diagram type.

### **Use Case Specification-Relation Tab**

The Relations tab lists all the association relationships that correspond to the selected use case. The client and supplier names and type icons are displayed to the right of the relation name. Double-clicking on any column in a row displays the element's specification.

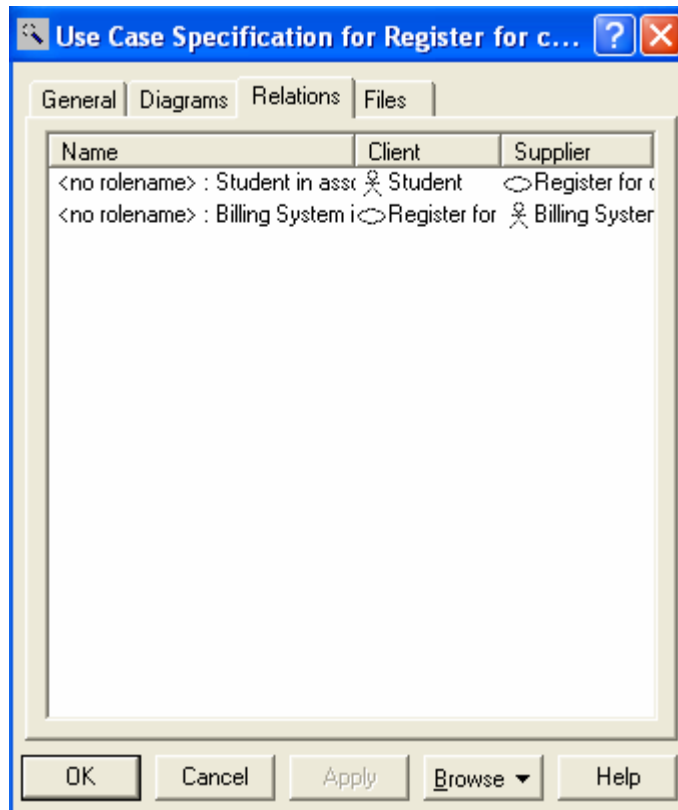


Figure 3.10 Use Case Specification-Relation Tab

## 4. Class diagram and Specification

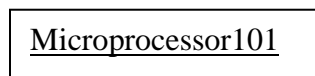
### 4.1. Overview

A class is a description of a group of objects with common properties (attributes), common behavior (operation's), common relationships to other objects. Each object is an instance of some class and objects can not be instances of more than one class.

An object is a representation of an entity, either real word or conceptual. An object is a concept, abstraction, or thing with well defined boundaries and meaning for application. Each object in a system has three characteristics: state, behavior, and identity.

The state of an object is one of the possible conditions in which it may exist. The state of an object typically change over time, and is defined by a set of properties (called attributes), with the values of the properties, plus the relationships the object have with other object. Behavior determines how an object responds to requests from other objects and typifies everything the object can do. Behavior is implemented by the set of operations for the object. Identity means that each object is unique, even if its state is identical to that of another object.

In the UML, objects are represented as rectangles and the name of the object is underlined and classes are represented as a compartmentalized rectangles as shown in figure 4.1



*Figure 4.1 UML Notation for an object*

To create classes:

- Right-click to select the Logical view in the browser.
- Select the **New: class** menu choice, a class called Newclass is placed in the browser.
- While the new class is still selected, enter the name of the class.

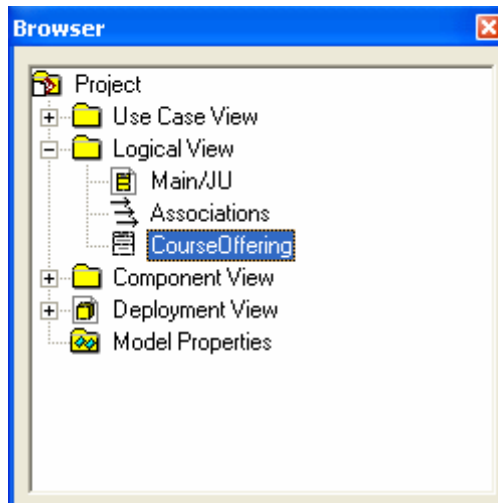


Figure 4.2 Class created in the Browser

A stereotype provides the capability to create a new kind of modeling element. The uses of stereotype are:

- Allow for customization of the development process.
- Provides mnemonic help and visualization aids.
- Allows making presentation with greater detail.

Classes have stereotypes. That is, new kind of classes can be created; some common stereotypes for a class are entity, boundary, control, and exception. These stereotypes are shown in figure 4.3.

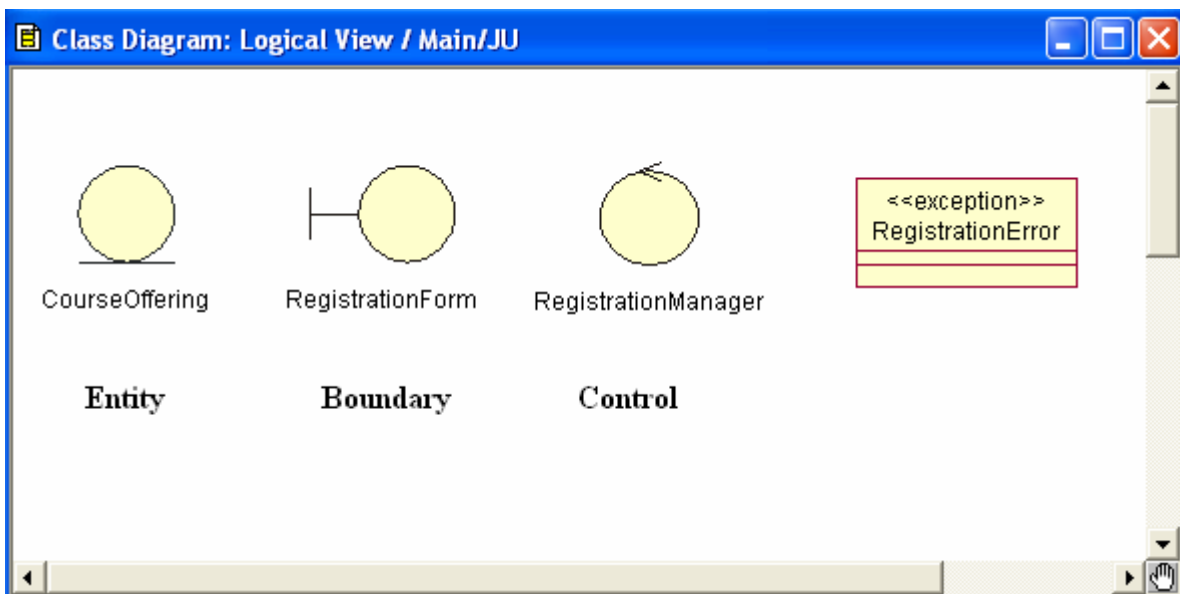


Figure 4.3 Classes with stereotypes

An entity class models information and associated behavior that is generally long lived. This type of class reflects a real-world entity or needed to perform tasks internal to the system. They are typically independent of their surroundings, that is, they are not sensitive to how the surrounding communicates with the system.

Boundary classes handle the communication between the system surroundings and the inside of the system. They provide the interface to a user or another system (i.e. the interface to an actor). They constitute the surroundings dependent part of the system. Boundary classes are used to model the system interfaces.

Control classes model sequencing behavior specific to one or more use cases. Control classes coordinate the events needed to realize the behavior specified in the use case. Control classes typically are application dependent classes.

To create stereotypes for classes:

- Right-click to select the class in the browser.
- Select the open specification menu choice.
- Select the General tab.
- Click the arrow in the stereotype field to make the drop down menu visible and select the desired stereotype.
- Click the OK button to close specification.

As classes are created, they should also be documented. The documentation should state the purpose of the class and not the structure of the class. For example, a student class could be documented as follows:

*Information needed to register and bill students. A student is some one  
Currently registered to take classes at the university.*

To document a class:

- Click to select the class in the browser.
- Position the cursor in the documentation window and enter the documentation for the class.

Most systems are composed of many classes, and thus there should be a mechanism to group them together for ease of use, maintainability, and reusability. This is where the concept of package is useful. A package in the logical view of the model is a collection of related packages and /or classes. By grouping classes into packages, looking at the “higher” level view of the model is possible. Each

package contains an interface that is realizable by its set of public classes, those classes to which classes in other packages talk. The rest of the classes in a package are implementation classes, classes do not communicate with classes in other packages.

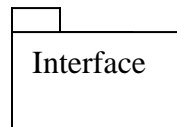


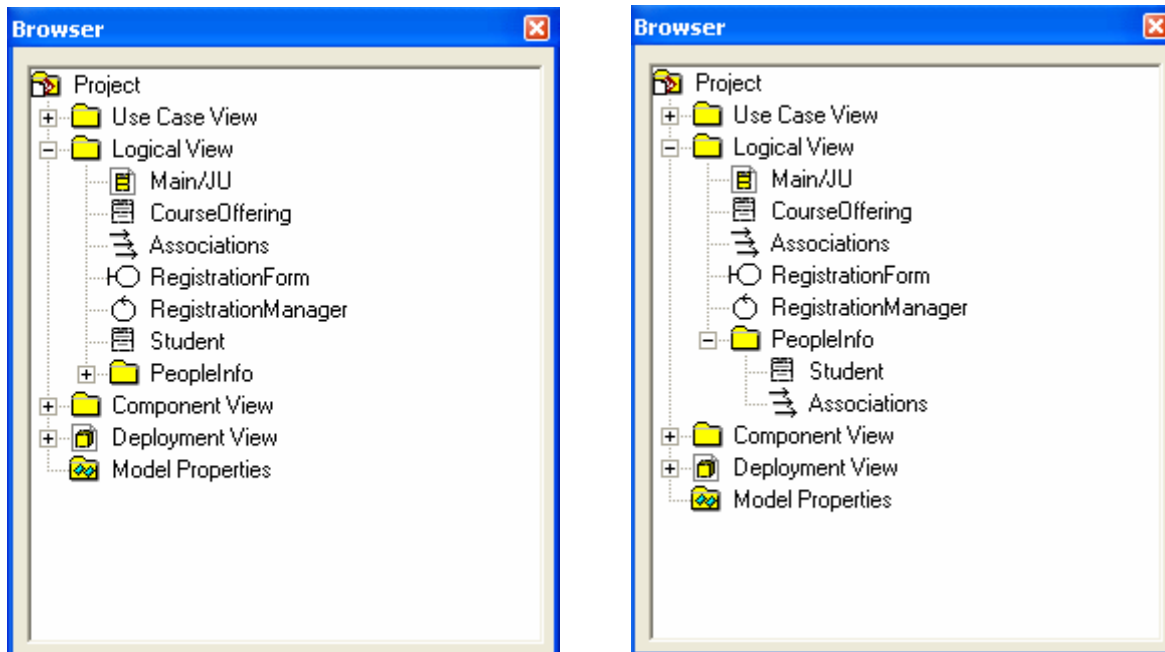
Figure 4.4 UML Notation for a package

To create packages:

- Right-click to select the Logical view in the browser.
- Select the New package menu choice.
- While the package is still selected, enter the name of the package.

To relocate classes:

- Click to select the class in the browser.
- Drag the class to the desired package.
- Repeat the steps for each class that is to be relocated.



(a)

(b)

Figure 4.5 a) Creating package b) relocated classes



## 4.2. Class Diagrams

A class diagram is a picture for describing generic description of possible systems. Class diagrams contain classes and object diagrams contain objects, but it is possible to mix classes and objects when dealing with various kinds of metadata, so the separation is not rigid.

Class diagrams contain icons representing classes, interfaces, and their relationships. It is possible to create one or more class diagrams to depict the classes at the top level of the current model. It is also possible to create one or more class diagrams to depict class contained by each package in the model. Properties and relationship of a class can be changed by editing the specification or modifying the icon in the diagram. The associated diagrams or specification are automatically updated.

### 4.2.1. Class Diagram Tool box

The graphic below shows all the tools that can be placed on the class diagram toolbox. The application window displays the following toolbox when the current window contains a class diagram. It is possible to customize the toolbox to display all the tool options.

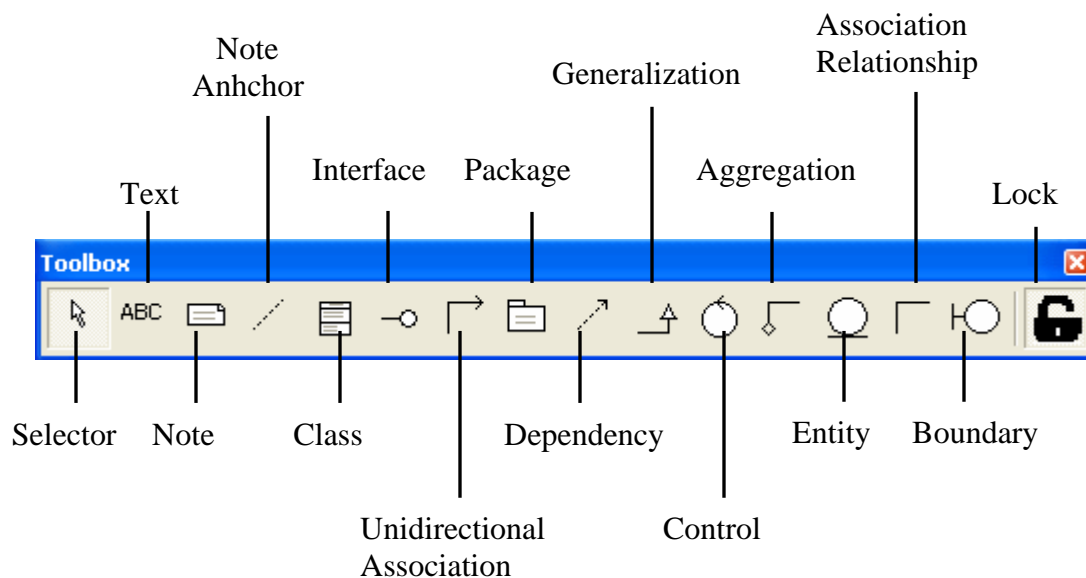


Figure 4.6 Class diagram toolbox

To create and display class diagram:

- Click **Browse: Class Diagram**.
- On the toolbar, click the class diagram icon.
- On the browser, double-click the class diagram icon.

To add packages to class diagram:

- Double-click on the main diagram in the browser to open the diagram.
- Click to select the package in the browser.
- Drag the package onto the diagram.
- Repeat the proceeding steps for each package that is to be added to the diagram.

#### 4.2.2. Adding and Hiding Classes and Filtering Class Relationships

The commands on the Query menu allow to control which model elements are represented by icons in the current diagram.

On the Query menu, clicking:

- **Add Classes** adds classes to the diagram by name.
- **Add Use Cases** adds use cases to the diagram by name.
- **Expand Selected Elements** adds classes to the diagram based on their relationships to selected classes.
- **Hide Selected Elements** removes selected classes from the diagram and optionally removes their clients or suppliers from the diagram.
- **Filter Relationships** controls which kinds of relationships appear in the diagram.

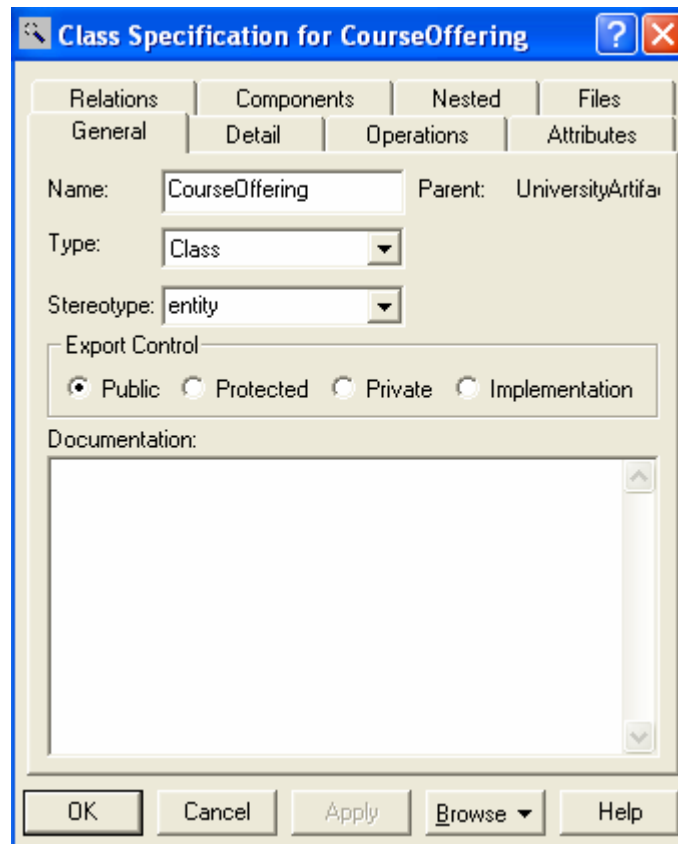
### 4.3. Class Specification

A Class Specification displays and modifies class properties and relationships. Some of the information in the specification can also be displayed inside class icons. If a field does not apply to a particular class type, the field is unavailable and cannot add or change information in the field.

To display a Class Specification, click an icon representing the class in a class diagram and click **Browse: Specification**.

The Class Specification consists of the following tabs: General, Detail, Operations, Attributes, Relations, Component, Nested, and Files.

## Class Specification - General Tab



*Figure 4.7 Class Specifications, General Tab*

### **Type**

Type choices include: Class, Parameterized Class, Instantiated Class, Class Utility, Parameterized Class Utility, Instantiated Class Utility, and Metaclass.

### **Parent**

The parent to which the class belongs (its package) is displayed in this static field.

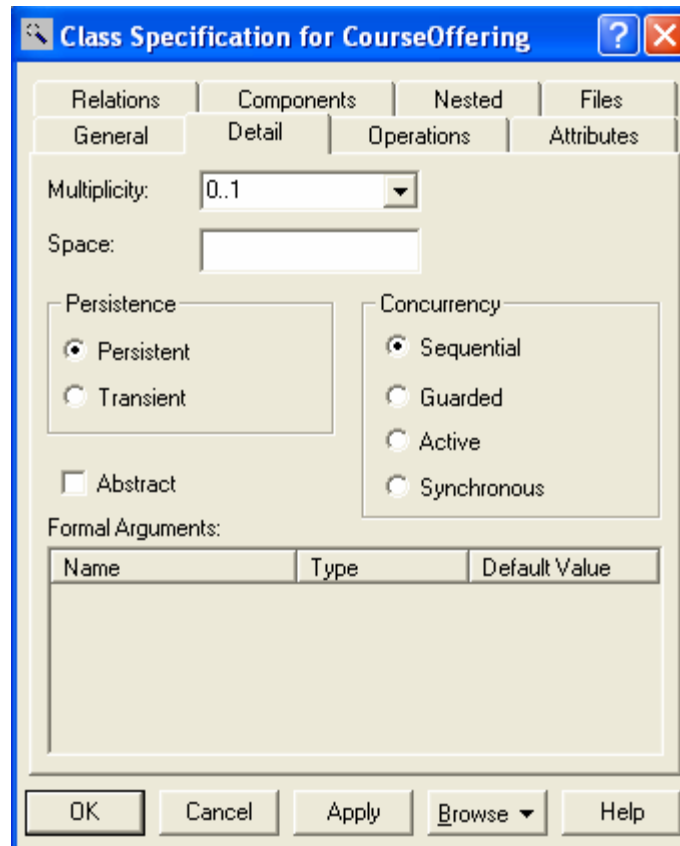
### **Stereotype**

A stereotype represents the sub classification of an element. It represents a class within the UML metamodel itself; that is, a type of modeling element. Some stereotypes are already predefined. Stereotypes can be shown in the browser and on diagrams. The name of the stereotype may appear in angle brackets <<>>, depending on the settings found in either the Diagram or Browser tabs of the Options dialog box.

## Export Control

The Export Control field specifies how a class and its elements are viewed outside of the defined package.

## Class Specification - Detail Tab



The screenshot shows a dialog box titled "Class Specification for CourseOffering" with a blue title bar. It has several tabs: "Relations", "Components", "Nested", "Files", "General", "Detail", "Operations", and "Attributes". The "Detail" tab is selected. The dialog contains the following fields and controls:

- Multiplicity: A dropdown menu showing "0..1".
- Space: An empty text input field.
- Persistence: A group box containing two radio buttons: "Persistent" (selected) and "Transient".
- Concurrency: A group box containing four radio buttons: "Sequential" (selected), "Guarded", "Active", and "Synchronous".
- Abstract: A checkbox that is currently unchecked.
- Formal Arguments: A table with three columns: "Name", "Type", and "Default Value". The table is currently empty.
- Buttons: "OK", "Cancel", "Apply", "Browse" (with a dropdown arrow), and "Help".

*Figure 4.8 Class Specifications, Detail Tab*

## Multiplicity

The Cardinality field specifies the number of expected instances of the class. In the case of relationships, this field indicates the number of links between each instance of the client class and the instance of the supplier. A specific cardinality value can be set for the client class, supplier class, or both.

## Space

The Space field is used to document the amount of storage required by objects of the class during execution.

**Persistence**

Persistence defines the lifetime of the instances of a class. A persistent element is expected to have a life span beyond that of the program or one that is shared with other threads of control or other processes. This field is used to identify the persistence for elements of this class.

**Concurrency**

A class concurrency defines its semantics in the presence of multiple threads of control.

**Abstract**

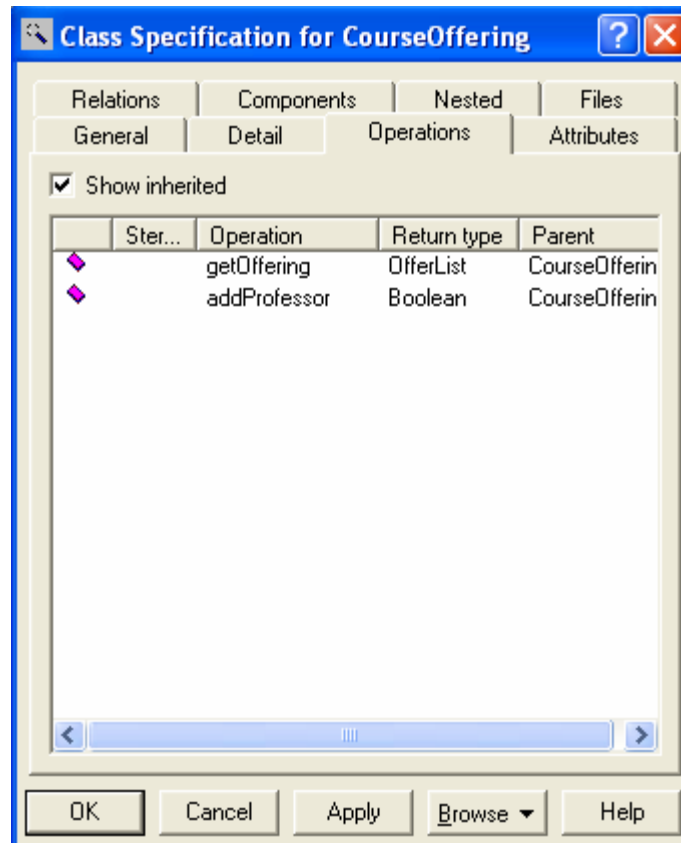
The Abstract check box identifies a class that serves as a base class. An abstract class defines operations and states that will be inherited by subclasses. This field corresponds to the abstract class adornment displayed inside the class icon.

**Formal Arguments**

In the Parameterized Class or Parameterized Class Utility Specification, the formal, generic parameters declared by the class or class utility are listed. In the Instantiated Class or Instantiated Class Utility Specification, the actual arguments that match the generic parameters of the class being instantiated are listed.

**Class Specification - Operations Tab**

Operations denote services provided by the class. Operations are methods for accessing and modifying Class fields or methods that implement characteristic behaviors of a class. The Operations tab lists the operations that are members of this class. Rational Rose stores operation information in an Operation Specification. Operation Specifications can be accessed from the Class Specification or from the Browser.



*Figure 4.9 Class Specifications, Operation Tab*

### **Show Inherited**

Selecting the Show Inherited check box will show the operations inherited from other classes. If there is no check mark in this field, operations only associated with the selected class are viewed.

## Class Specification - Attribute Tab

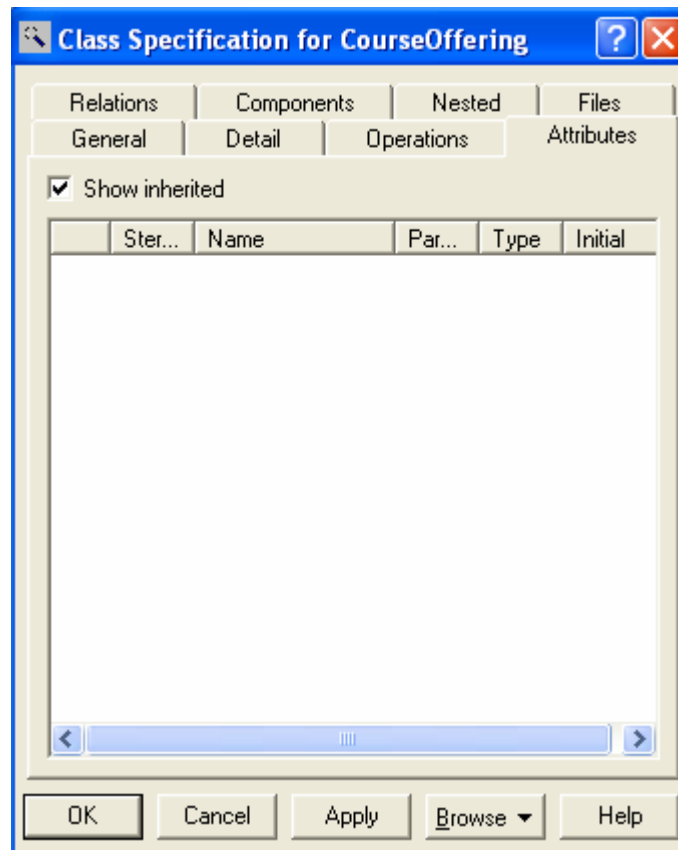


Figure 4.10 Class Specifications, Attribute tab

The Rational Unified Process asserts that attributes are data values (string or integer) held by objects in a class. Thus, the Attributes tab lists attributes defined for the class through the Class Attribute Specification.

## Class Specification - Relations Tab

Classes collaborate with other classes in a variety of ways. The Relations tab identifies the relationships in which this class is the client (class) and the corresponding supplier (end) class. If the relationship is labeled, Rational Rose displays its name after the kind of relationship.

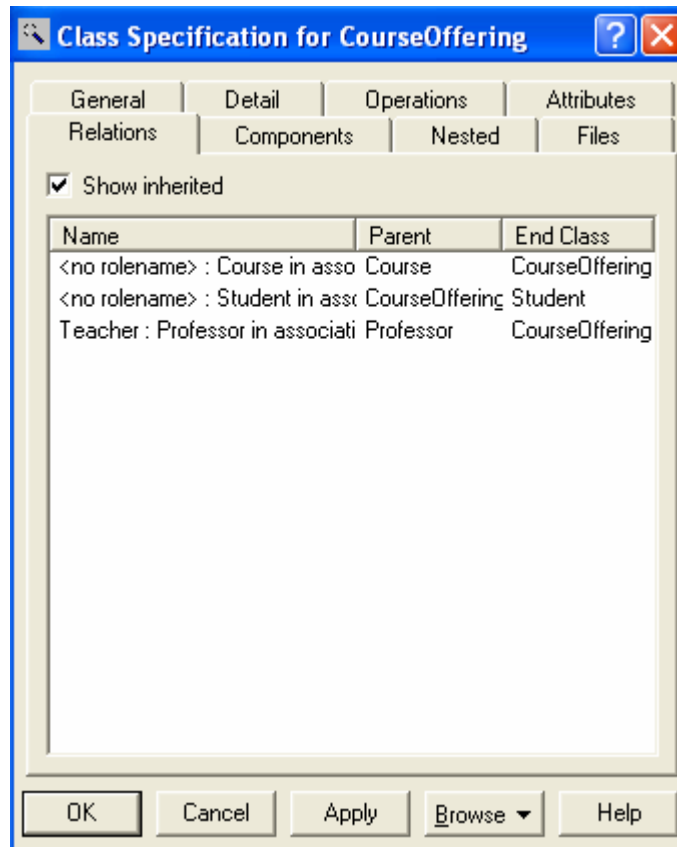


Figure 4.11 Class Specifications, Relation tab



## Class Specification - Component Tab

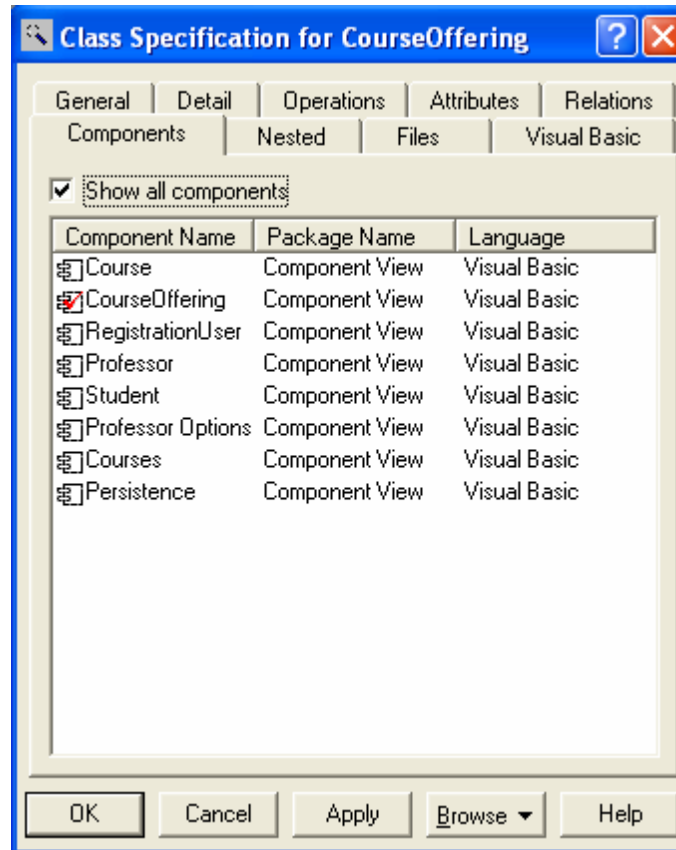


Figure 4.12 Class Specifications, component tab

### Show All Components

To get a list of all components in a model, this option should be selected. If this option is not selected, the component to which only this class is assigned is listed.

### Component Name

The component list identifies the components to which this class is assigned (with a check mark). A class can be assigned to a note or to several components with the same implementation language assigned. Class can be assigned to a component through Assign on the shortcut menu or by dragging a component from the browser and dropping it in the list.

### Package Name

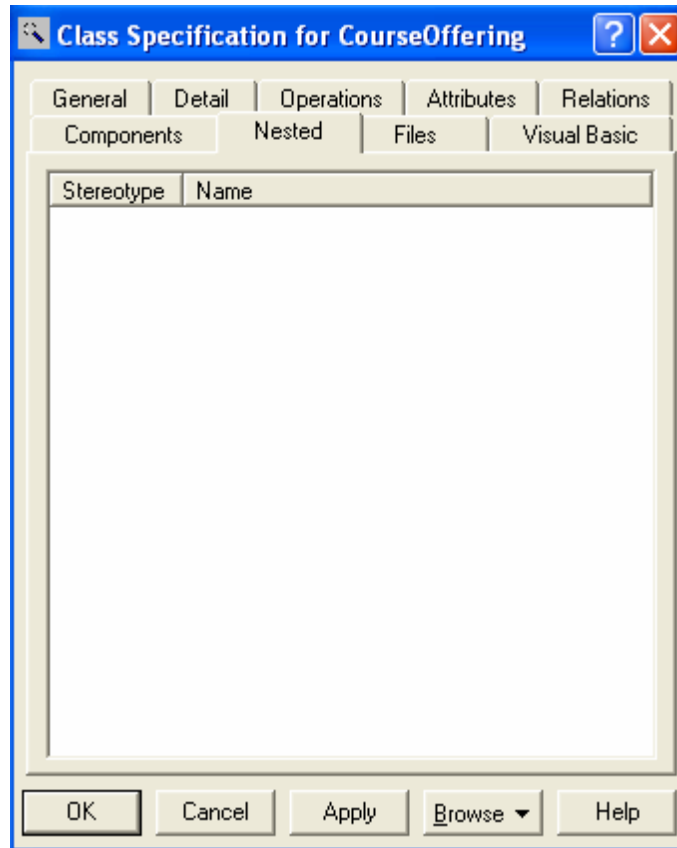
This field displays the package that the component belongs to.

### Language

The Language field identifies the implementation language assigned to this element.

### Class Specification - Nested Tab

A nested class is a class that is enclosed within another class. Classes may contain instances of, inherit from, or use a nested class. Enclosing classes are referred to as parent classes, and a class that lies underneath the parent class is called a nested class.



*Figure 4.13 Class Specifications, Nested tab*

## 5. Interaction Diagram and Specifications

### 5.1. Interaction Diagram Overview

An interaction is an important sequence of interactions between objects. Rational Rose provides two alternate views or representations of each interaction, a collaboration and sequence diagram. These are collectively referred to as interaction diagrams. The main difference between sequence and collaboration diagrams is that sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

It is possible to specify and modify an interaction with either kind of diagram, or with both. Rational Rose automatically reflects all changes made either to a sequence or collaboration diagram in the corresponding collaboration or sequence diagram, if one has been created.

### 5.2. Use case realization

The use case diagram presents an outside view of the system. The functionality of the use case is captured in the flow of events. Scenarios are used to describe how use cases are realized as interactions among societies of objects. A scenario is an instance of a use case; it is one path through the flow of events for the use case. Scenarios are developed to help identify the objects, the classes, and the object interactions needed to carry out a piece of the functionality specified by the use case. Scenarios document decisions about how the responsibilities specified in the use cases are distributed among the objects and classes in the system. They also provide an excellent communication medium to be used in the discussion of the system requirements with customers.

In the Rational Unified Process, use case realizations are captured in the Logical View of the model. The concept of a stereotype is used to show that the use cases that are created in the Logical View of the model are the realization of the use cases contained in the Use Case View along with a stereotype realization. In UML, use case realizations are drawn as dashed ovals.



*Figure 5.1. UML Notation for Use Case Realization*

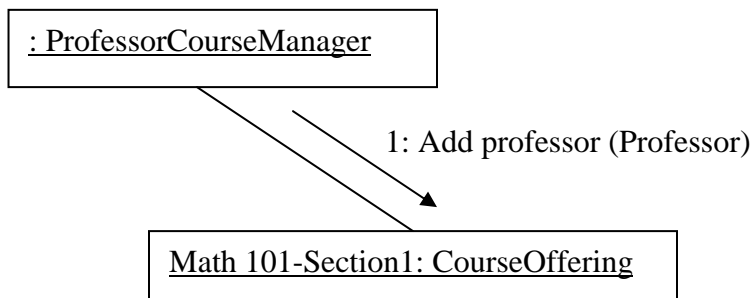
To create use case realizations:

- Double-click on the realizations use case diagram in the browser.
- Click to select the use case icon from the toolbar.
- Click the use case diagram window to place the use case.
- Double-click on the use case to open the Use Case Specification.
- Enter the name of the use case (same name as the use case in the use case view) in the name field.
- Click the arrow in the stereotype field to make the dropdown menu visible.
- Select use-case realization.
- Click the OK button to close the use case specification.

### 5.3. Collaboration Diagram

A collaboration diagram is an interaction diagram which shows object interactions organized around the objects and their links to each other. These diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

Collaboration diagrams are used as the primary vehicle to describe interactions that express decisions about the behavior of the system. They can also be used to trace the execution of a scenario by capturing the sequential and parallel interaction of a cooperating set of objects. Collaboration diagrams may also depict interactions that illustrate system behavior.



*Figure 5.2 UML notations for objects, Links, and Message in collaboration diagram*

To create and display collaboration diagram:

- Click **Browse: Interaction Diagram**.  
The Select Interaction Diagram dialog box is displayed.
- Select a package to “own” the diagram.
- On the right side of the dialog box, click the diagram name, and then click OK.
- From the New Interaction Diagram dialog box, enter the diagram title and click the diagram type. The choice is Collaboration.

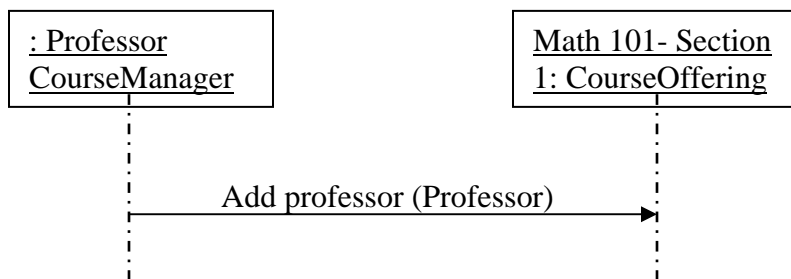
## 5.4. Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence, what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. Sequence diagrams are normally associated with use cases.

This type of diagram is best used during early analysis phases because they are simple and easy to comprehend. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Sequence diagrams are closely related to collaboration diagrams and each are alternate representations of an interaction.

A sequence diagram traces the execution of a scenario in time.



*Figure 5.3. UML Notations for Objects and Messages in a Sequence diagram*

To create and display sequence diagram:

- Click **Browse: Interaction Diagram**.  
The Select Interaction Diagram dialog box is displayed.
- Select a package to “own” the diagram.

- On the right side of the dialog box, click the diagram name, and then click OK.
- From the New Interaction Diagram dialog box, enter the diagram title and click the diagram type. The choice is Sequence.

To Create Objects and message in sequence or collaboration diagram:

- Double-click on the sequence or collation diagram in the browser to open the diagram.
- Click to select the actor in the browser.
- Drag the actor onto the sequence or collaboration diagram.
- Click to select the object icon from the toolbar.
- Click to the sequence or collaboration window to place the object.
- While the object is still selected, enter the name of the object.
- Repeat the preceding steps for each object and actor in the scenario.
- Click to select the object message icon from the toolbar.
- Click on the actor or object sending the message and drag the message line to the actor or object receiving the message.
- While the message line is still selected, enter the name of the message.
- Repeat the above steps for each message in the scenario.

## **5.5. Toolboxes**

Each diagram type has its own unique toolbox. The collaboration and sequence diagram toolboxes are illustrated below.

### **5.5.1. Collaboration Diagram Toolbox**

The graphic below shows some of the tools that can be placed on the collaboration diagram toolbox. The application window displays the following toolbox when the current window contains a collaboration diagram. View: As Unified should be selected.

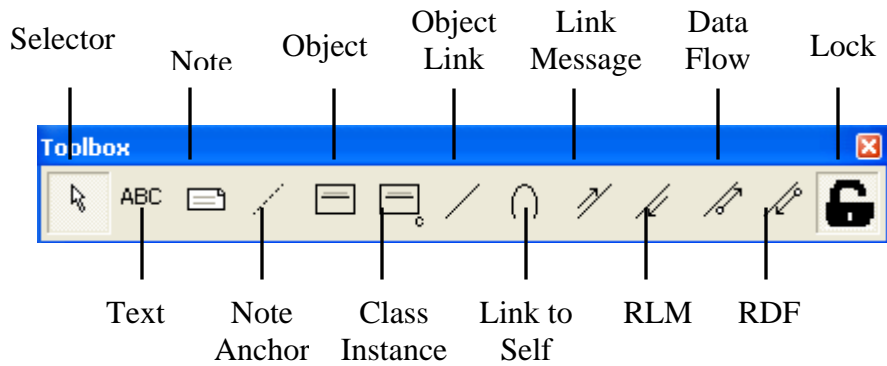


Figure 5.4 Collaboration Diagram Toolbox

### 5.5.2. Sequence Diagram Toolbox

The graphic below shows all the tools that can be placed on the sequence diagram toolbox. The application window displays the following toolbox when the current window contains a sequence diagram. View: As Unified should be selected.

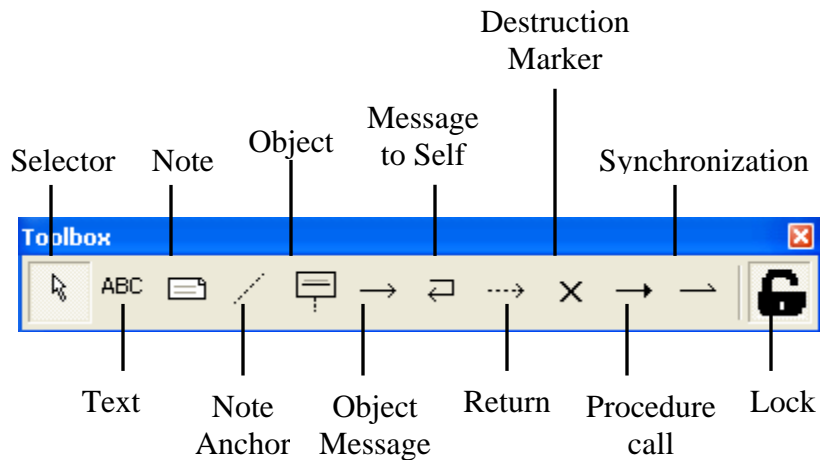


Figure 5.5 Sequence Diagram Toolbox

## 5.6. Specifications

### 5.6.1. Object Specification

An object specification allows to display and modify the properties and relationships of an object in the current model.

To display an Object Specification, double-click any icon representing an object, or click Browse: Specifications. The Object Specification consists of the General tab.

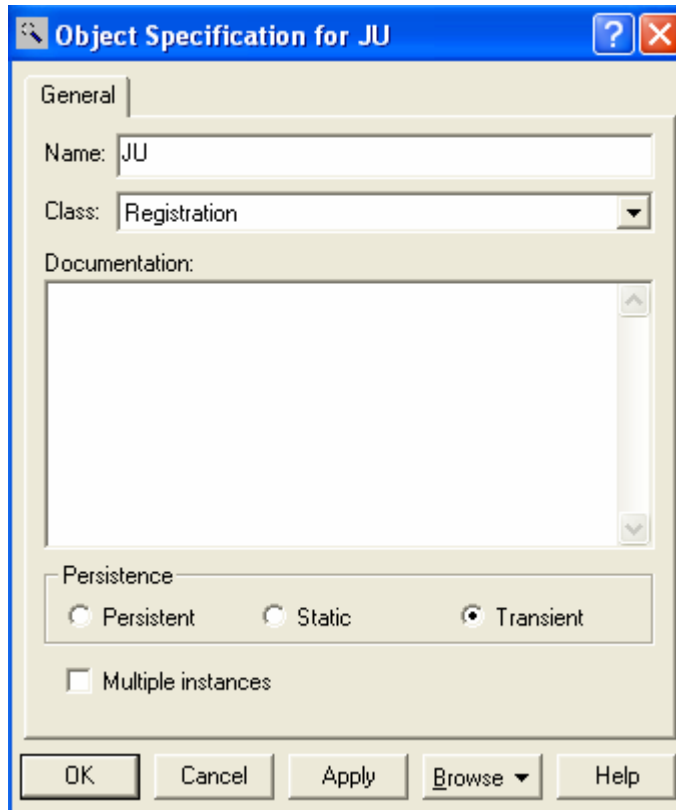


Figure 5.6 Object specifications, General tab

### **Name**

If the name of the object's class specified in the Object Specification, the name must identify a class defined in the model.

### **Class**

The Class field displays the name of the object's parent class. The default class for a newly created object is Unspecified. The object will accept messages conveying the operations of its parent class, and the operations of the superclasses of its parent class.

### **Persistence Field**

These options specify the object's persistence.

**Persistent:** - This option will make the object exists after the termination of the program in which it was created.

**Static:** - This option will make the object exist during the entire execution of a program.



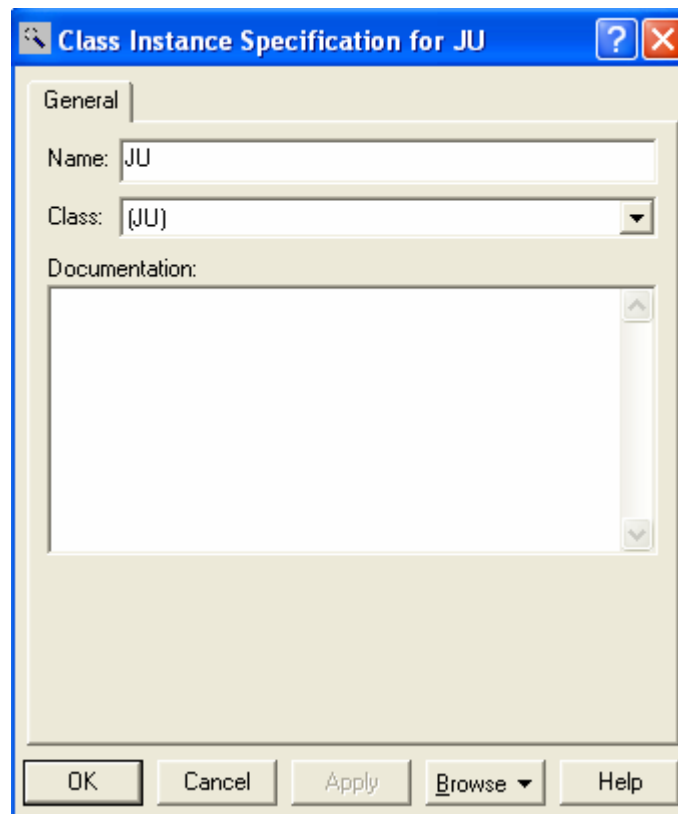
**Transient:** - This option will make the object is created and destroyed dynamically during the execution of a program.

### **Multiple Instances Check Box**

Multiple Instances check box is selected to indicate that this object represents multiple instances of the same class. When this field is selected, the icon changes from one object to three staggered objects. The object group is considered as one entity, but this icon indicates that several objects are involved.

## **5.6.2. Class Instance Specification**

A class instance places a representation of a class on a collaboration diagram. To display a Class Instance Specification, double-click any icon representing a class instance, or click Browse: Specifications. The Class Instance Specification consists of the General tab.



*Figure 5.7 Class instance specifications, General tab*

## Class

The class the element belongs to is displayed here. The default class for a newly created element is (Unspecified). If an object's class in the Object Specification is specified, the class name must identify a class defined in the model, or create a new class.

To create a new class through the Object Specification, click the scroll arrow to the right of the Class field. A list box will display all the possible class selections, including New. Double-click New display a Class Specification dialog box. Enter the information regarding the new class.

### 5.6.3. Link Specification

A link is the path of communication between two objects. A link can exist between two objects, between an object and a class instance, or between an object and itself.

To display a Link Specification, double-click any icon representing a link, or click Browse: Specifications. The Link Specification consists of the following tabs: General and Messages.

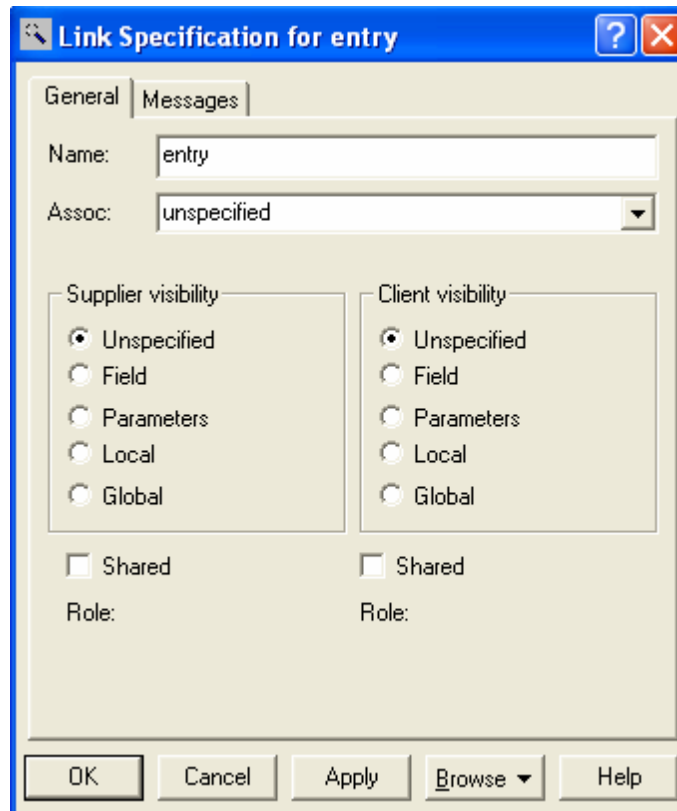


Figure 5.8 Link specifications, General tab

## **Assoc**

The Assoc field lists any valid role(s) or association(s) tied to the classes belonging to the two objects. Association is selected from the drop-down list. The name of the role tied to the association is displayed beside the link on the diagram.

## **Supplier and Client Visibility**

Visibility is the ability of one object to see another object.

**Unspecified:** - indicate that object visibility has not been specified.

**Field:** - indicate that the supplier object is visible because it is a field of client.

**Parameters:** - indicate that the supplier object is visible to the client because it is a parameter for one of the client's operation.

**Local:** - indicate that the supplier is local to an operation of the client object.

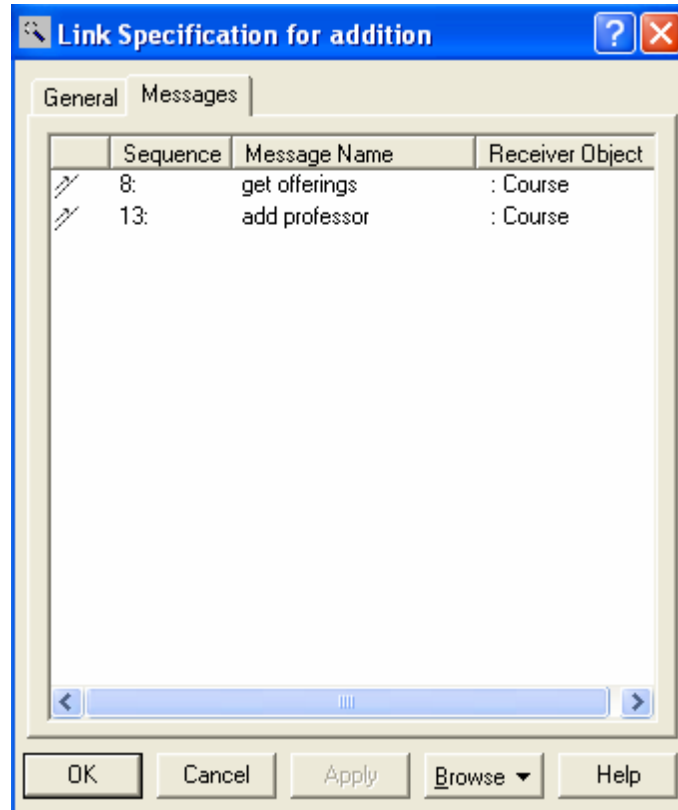
**Global:** -indicate that the supplier object is global to the client.

## **Shared**

Shared visibility indicates structural sharing of the given object; that is, the shared object's state can be altered through more than one path. Unshared visibility represents unique access given to the client object. When a link is created, unshared visibility is the default.

## **Role**

This field lists the role names tied to the selected associations. This is especially useful since many associations are not named. This field cannot be edited.



*Figure 5.9 Link specifications, Message tab*

**Icon**

This left-most unlabeled field contains a small version of the link message icon indicating the direction of the message.

**Sequence**

This is a system assigned, sequential message number.

**Message Name**

The Pick list box showing all available operations on the class can be seen by clicking the item. This is the only editable column on this tab.

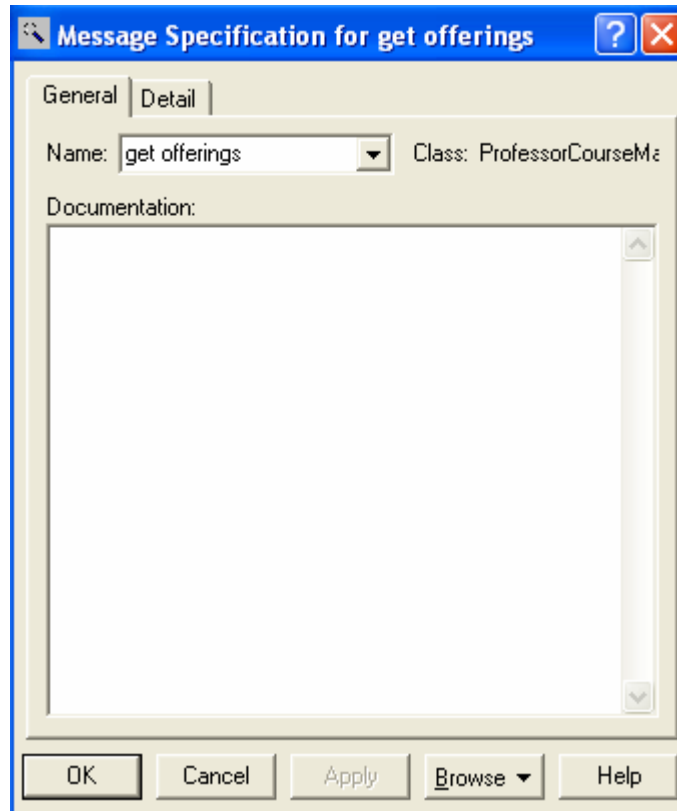
**Receiver**

This is the object receiving the message.

### 5.6.4. Message Specification

A message conveys an operation through a link between objects. A message's specification identifies the operation it conveys, its synchronization, its frequency, and its associated documentation.

To display a Message Specification, double-click any icon representing a message, or click Browse: Specifications. The Message Specification consists of the following tabs: General and Detail.



*Figure 5.10 Message specifications, General tab*

#### **Class**

The Class field displays the name of the class to which the element belongs.

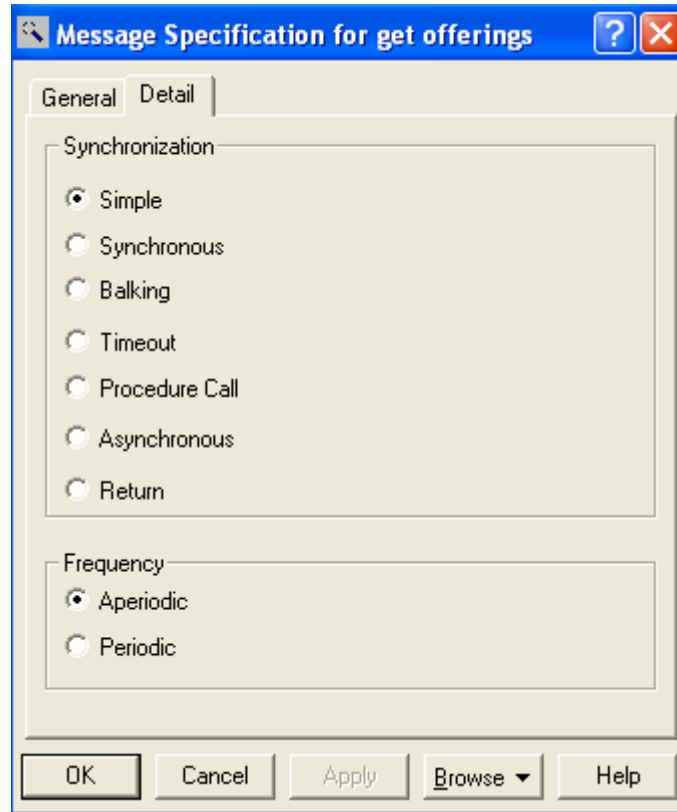


Figure 5.11 Message specifications, Detail tab

## Synchronization

These options are used to specify concurrency semantics for the operation named in the Synchronization field.

**Simple:** - indicate that the message has a single thread of control.

**Synchronous:** - indicate that the operation proceeds only when the client sends a message to the supplier and the supplier accepts the message.

**Balking:** - indicate that the client passes a message only if the supplier is immediately ready to accept the message; the client abandons the message if the supplier is not ready.

**Timeout:** - indicate that the client abandons a message if the supplier cannot handle the message within a specified amount of time.

**Asynchronous:** - indicate that the client sends a message to the supplier for processing and continues to execute its code without waiting for or relying on the supplier's receipt of the message.

**Procedure Call:** - indicate that the entire nested sequence is completed before the outer level sequence resumes. This can be used with ordinary procedure calls as well as with concurrently active objects when one of them sends a signal and waits for a nested sequence of behavior to complete.

**Return:** - Return from a procedure call. The return arrow may be suppressed since it is implicit at the end of activation.

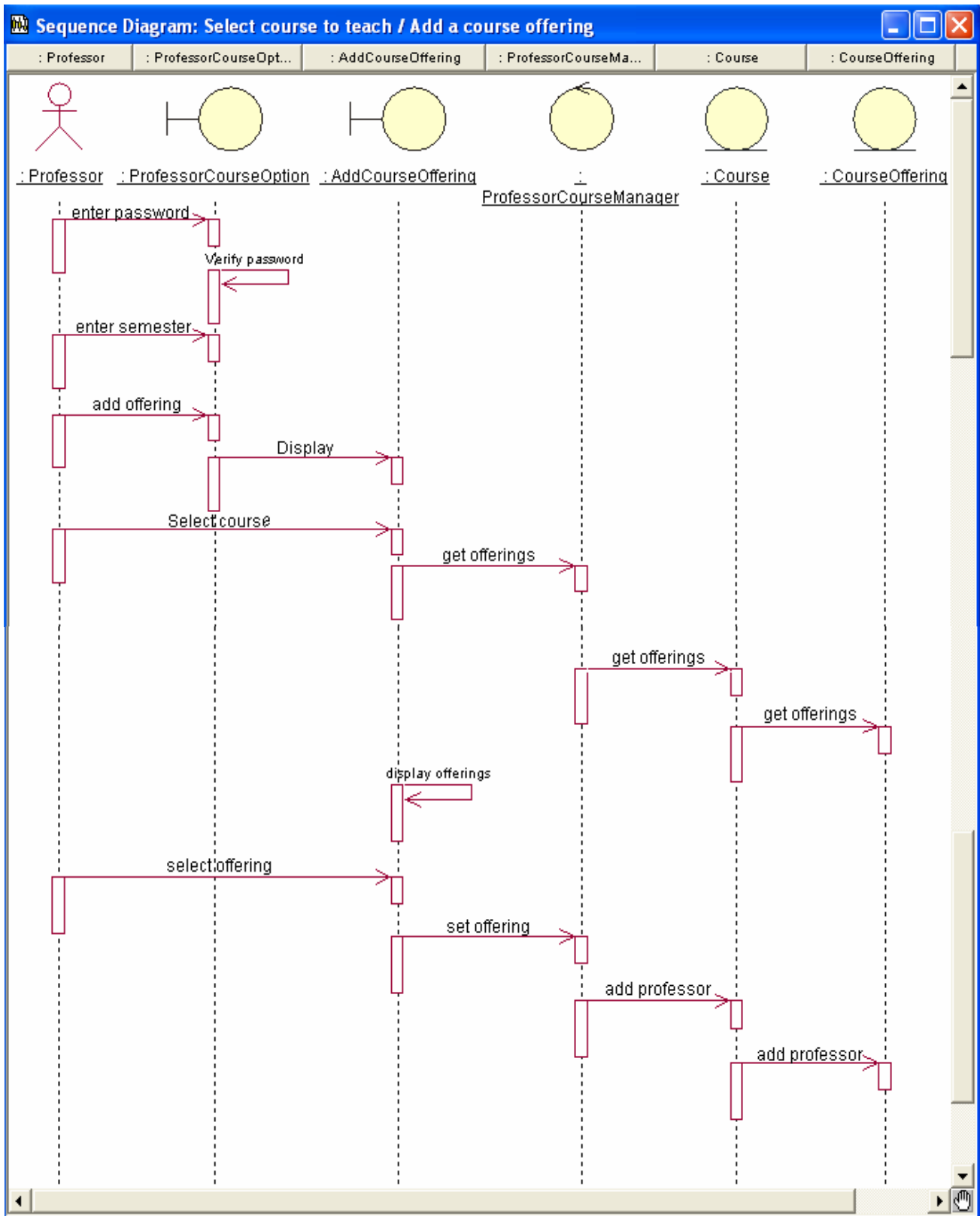
### **Frequency**

These options are used to indicate whether the message is sent periodically or aperiodically.

**Aperiodic:** - indicate that the message is sent at irregular intervals, or does not have a regular interval.

**Periodic:** - indicate that the message is sent at regular intervals.

## 5.7. Sequence Diagram for JU course registration system





*Figure 5.12 Sequence Diagram for the Add a course offering scenario*

To create collaboration diagram from Sequence diagram or sequence diagram from collaboration diagram:

- Double-click on the sequence diagram in the browser to open the diagram.
- Choose the **Browse: Create collaboration** diagram menu choice or press the F5 key.
- Rearrange the objects and messages on the diagram as needed.

## 6. Relationship Specification

All systems are made up of many classes and objects. System behavior is achieved through the collaborations of the objects in the system. Relationships provide the conduit for object interaction; two types of relationships discovered are associations and aggregations.

### 6.1. Association Relationships

An association is a bidirectional semantic connection between classes. An association between classes means that there is a link between objects in the association classes. The number of objects connected depends upon the multiplicity of the association.

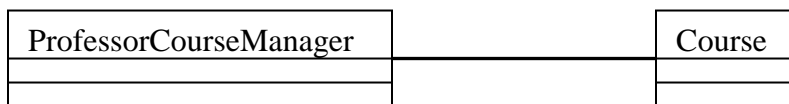


Figure 6.1. UML Notation for association relationship

To create an association relationship:

- Click to select the association icon from the toolbar.
- Click on one of the associated classes in a class diagram.
- Drag the association line to the other associated class.

### 6.2. Aggregation Relationship

An aggregation Relationship is a specialized form of association in which a whole is related to its part (s). Aggregation is known as a “part-of” or containment relationship. The UML notation for an aggregation relationship is an association with a diamond next to the class denoting the aggregate (whole) as shown in figure 6.2.

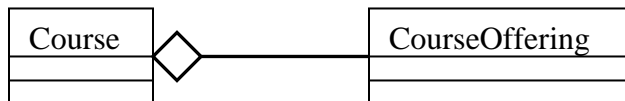


Figure 6.2. UML notion for an aggregation relationship

To create an aggregation relationship:

- Select the aggregation icon from the toolbar.

- Click on the class playing the role of the “whole” in a class diagram and drag the aggregation line to the class playing the role of the “part”.

### **6.3. Naming Relationships**

An association may be named; usually the name is an active verb or verb phrase that communicates the meaning of the relationship. Since the verb phrase typically implies a reading direction, it is desirable to name the association so it reads correctly from left to right or top to bottom. It is important to note that the name of the association is optional. Aggregation relationships typically are not named since they are read using the words “has” or “contains”.

To name relationships:

- Click to select the relationship line on a class diagram.
- Enter the name of the relationship.

### **6.4. Role Name**

The end of an association where it connects to a class is called an association role. Role names can be used instead of association names. A role name is a noun that denotes the purpose or capacity where in one class associates with another. The role name is placed on the association near the class that it modifies, and may placed on one or both ends of an association line. It is not necessary to have both a role name and an association name.

To create Role Name:

- Right-click on the relationship line near the class that it modifies to make the shortcut menu visible.
- Select the Role Name menu choice.
- Enter the Name of the Role.

### **6.5. Multiplicity Indicators**

Multiplicity is specified for classes, it defines the number of objects that participate in a relationship. Multiplicity defines the number of objects that are linked to one another. There are two multiplicity indicators for each association or aggregation, one of each end of the line. Some common multiplicity indicators are:

1	Exactly one
0..*	Zero or more
1..*	One or more
0..1	Zero or one
5..8	Specific range (5, 6, 7, or 8)
4..7, 9	Combination (4, 5, 6, 7, or 9)

## 7. State Machine Diagrams and Specifications

### 7.1. Overview

The state/activity model icon that appears in the browser can be thought of as a “container” for statechart and activity diagrams and all of their model elements. A state/activity model owns statecharts and activity diagrams and is represented semantically with a state machine. A state machine can be defined as a behavior that specifies the valid sequences of activities that an object or interaction goes through during its life in response to events, together with its responses and actions. Rational Rose automatically creates one state/activity model when a statechart or activity diagram is created. A state/activity model can be relocated to a new owner, such as a class operation or a use case, by dragging it to a new location in the browser. Rational Rose limits to only one state/activity model per owner.

To Create and display a State Machine Diagram:

- Click **Browse: State Machine Diagram**.
- Double-click **New**.
- Name the diagram.
- Specify the type of diagram that is wanted to be created: Activity or Statechart.
- Click OK.

### 7.2. State Machine Specification

A State Machine Specification allows to display and modify the properties and relationships of a state/activity model. A state/activity model contains statechart and activity diagrams.

To view the State Machine Specification, double click the state/activity model in the browser. Changes made either through the specification or directly on the icon are automatically updated throughout the model.

### 7.3. Statechart Diagram

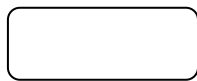
Statechart diagrams model the dynamic behavior of individual classes or any other kind of object. They show the sequences of states that an object goes through, the events that cause a transition from one state or activity to another and the actions that result from a state or activity change.

Statechart diagrams are closely related to activity diagrams. The main difference between the two diagrams is statechart diagrams are state centric, while activity diagrams are activity centric. A statechart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event. A statechart diagram typically contains one start state and multiple end states. Transitions connect the various states on the diagram.

### 7.3.1. States

A state is a condition during the life cycle of an object during which it satisfies some conditions, performs some action, or waits for an event. The state of an object can be characterized by the value of one or more of the attributes of the class.



*Figure 7.1 UML Notations for a State*

To create states:

- Click to select the State icon from the toolbar.
- Click to place the state on the statechart diagram.
- With the state still selected, enter the name of the state.

### 7.3.2. States Transitions

A State Transition represents a change from an originating state to a successor state. An action can accompany a state transition.

To create a State Transition:

- Click to select the State Transition icon from the toolbar.
- Click to select the originating state on the statechart diagram.
- Drag the state transition to the successor state.
- If the state transition is named transition, enter the name while the state transition arrow is still selected.

### 7.3.3. Special States

There are two special states that are added to the statechart diagram. The first is a start state. Each diagram must have one and only one start state since the object must be in a consistent state when it is created. The second special state is a stop state. An object may have multiple stop states.



Start State



Stop State

Figure 7.2 UML Notation for start and stop states

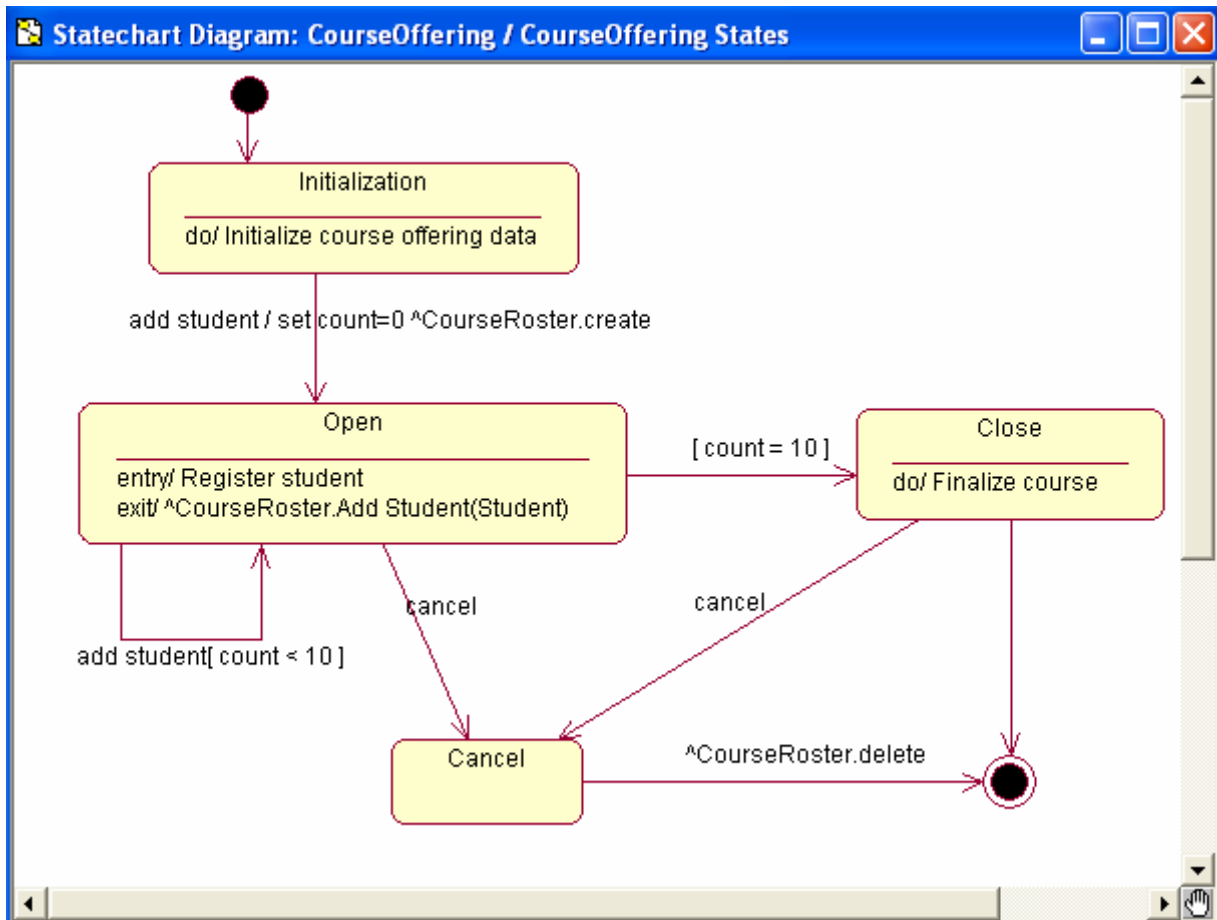
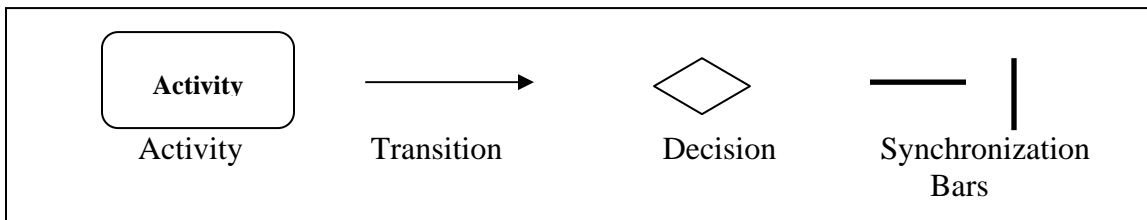


Figure 7.3 State chart diagram for courseOffering class

## 7.4. Activity Diagram

Activity diagrams provide a way to model the workflow of a business process. Activity diagrams can also be used to model code-specific information, such as a class operation. Activity diagrams are very similar to a flowchart because of modeling a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

Each activity represents the performance of a group of actions in a workflow. Once the activity is complete, the flow of control moves to the next activity or state through a transition. If an outgoing transition is not clearly triggered by an event, then it is triggered by the completion of the contained actions inside the activity. A unique activity diagram feature is a swimlane that defines who or what is responsible for carrying out the activity or state. It is also possible to place objects on activity diagrams. The workflow stops when a transition reaches an end state. It is possible to attach activity diagrams to most model elements in the use case or logical views. Activity diagrams cannot reside within the component view. The following tools on the activity diagram toolbox can be used to model activity diagrams:



*Figure 7.4 UML Notations for Activity Diagram Elements*

### 7.4.1. Activities

An activity represents the performance of some behavior in the workflow.

To create Activity:

- Click to select the Activity icon from the toolbar.
- Click on the activity diagram window to place the activity.
- While the activity is still selected, enter the name of the activity.



### 7.4.2. Transitions

Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity or by events.

To create Transitions:

- Click to select the State transition icon from the toolbar.
- Click on the originating activity and drag the transition arrow to the successor activity.

To create guarded transition:

- Click to select the State transition icon from the toolbar.
- Click on the decision and drag the transition to the successor activity.
- Double-Click on the transition arrow to make the specification visible.
- Select the Detail tab.
- Enter the guard condition in the guard condition field.
- Click OK button to close the specification.

### 7.4.3. Decision Points

When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point. The transitions from a decision point contain a guard condition, which is used to determine which path from the decision point is taken. Decisions along with their guard conditions allow to show alternate paths through the workflow.

To create decision point:

- Click to select the Decision icon from the toolbar.
- Click on the activity diagram window to place the decision.
- While the decision is still selected, enter the name of decision.
- Click to select the Transition icon on the toolbar.
- Click on the originating activity and drag the transition to the decision icon.

### 7.4.4. Synchronization Bars

In a workflow there are typically some activities that may be done in parallel. A synchronization bar allows to specify what activities may be done concurrently. Synchronization bars also used to show joins in the workflow, that is, what activities must complete before processing may continue.

To create synchronization bars:

- Click to select the Horizontal synchronization or Vertical synchronization icon from the toolbar.
- Click on the activity diagram window to place the synchronization bar.
- Click to select the State transition icon on the toolbar and add any needed incoming and out going transitions to the synchronization bar.

#### **7.4.5. Swimlanes**

Swimlanes are used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swimlane.

To create Swimlanes:

- Click to select the Swimlane icon from the toolbar.
- Click on the activity diagram window to place swimlane.
- Double-click on the new swimlane to open the specification.
- Enter the name of the swimlane in the Name field.
- Click the OK button to close the specification.
- To resize the swimlane, click on the swimlane border and drag the swimlane to the desired location.

#### **7.4.6. Initial and final Activities**

There are special symbols that are used to show the starting and final activities in the workflow. The starting activity is shown using a solid filled circle and the final activities are shown using a bull's eye. Typically there is one starting activity for the workflow and there may be more than one ending activity.

To create starting and ending activities:

- Click to select the start state or the end state icon from the toolbar.
- Click on the activity diagram window to place the start or end state.
- If a start state is added, click on the state transition icon, click on the start state, and drag the transition to the first activity in the workflow.
- If an end state is added, click on the state transition icon, click on the successor activity, and drag the transition to end state

### 7.4.7. Object Flow

An object flow on an activity diagram represents the relationship between an activity and the object that creates it (as an output) or uses it (as an input). Rational Rose draws object flows as dashed arrows rather than solid arrows to distinguish them from ordinary transitions. Object flows look identical to dependencies that appear on other diagram types.

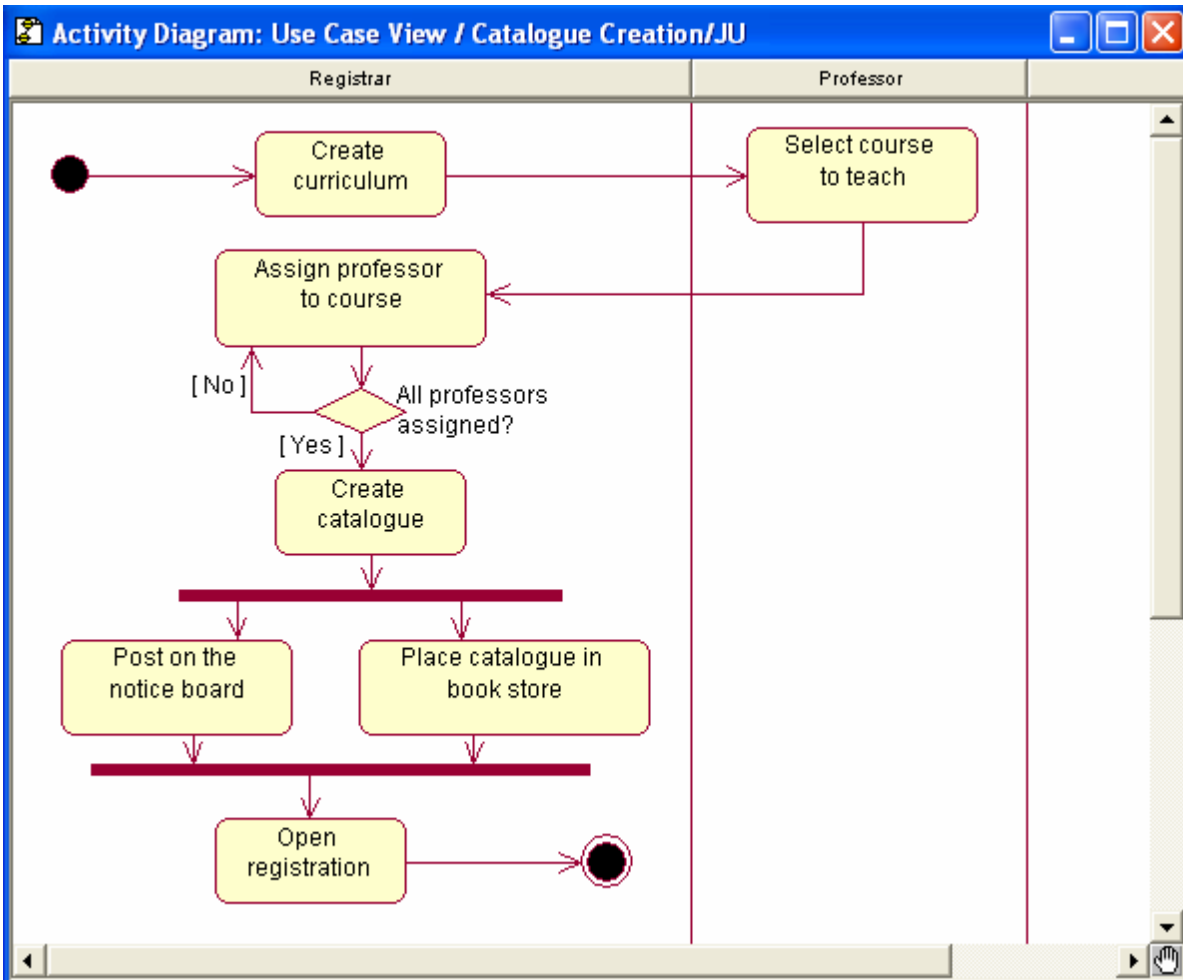


Figure 7.5 Activity diagram for catalogue creation use case

### 7.5. Swimlane Specification

A Swimlane Specification allows to display and modify the properties and relationships of a swimlane on an activity diagram.

To display a Swimlane Specification, select the swimlane header on an activity diagram and double-click. Double-click on the swimlane icon in the browser. The Swimlane Specification consists of the General tab.

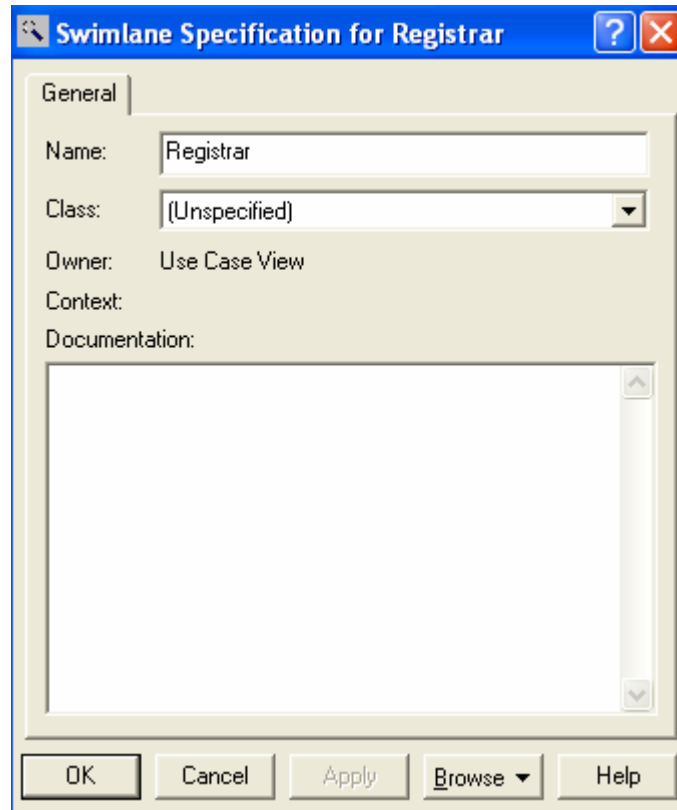


Figure 7.6 Swimlane specifications, General tab

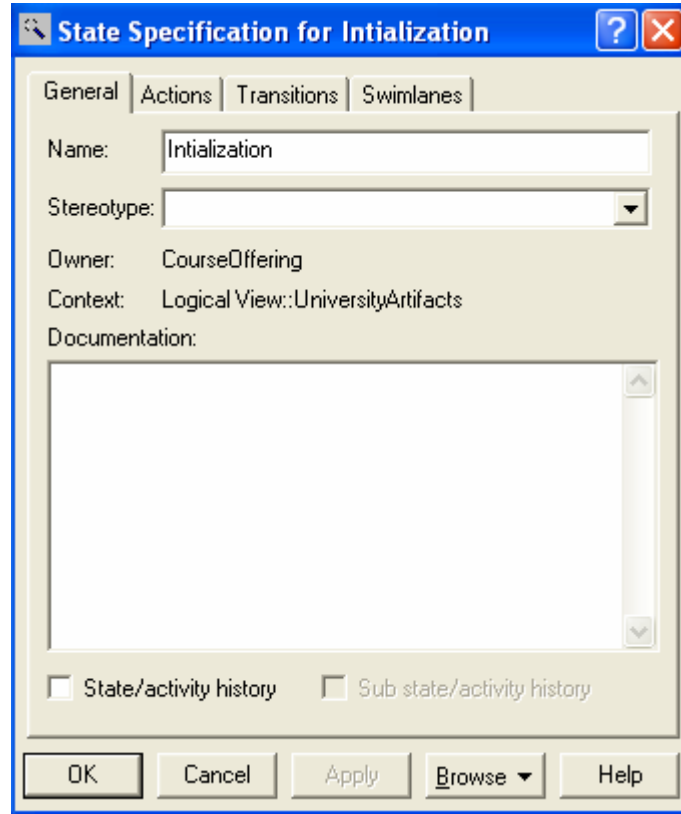
## 7.6. State and Activity Specification

A State and Activity Specification allows to display and modify the properties and relationships of a state or activity on a statechart diagram or activity diagram. Although a state and activity have almost identical features, they are used for different purposes. Start states and end states use the same specifications as states because they are a type of state. However, they appear as circles on statechart and activity diagrams.

The State, Activity, Start State, and End State Specifications consist of the following tabs: General, Action, Transitions, and Swimlanes.

### State and Activity specification-General Tab

Information about the name, stereotype, owner, context, documentation, state/activity history, and sub state/activity history is entered or displayed on this tab.



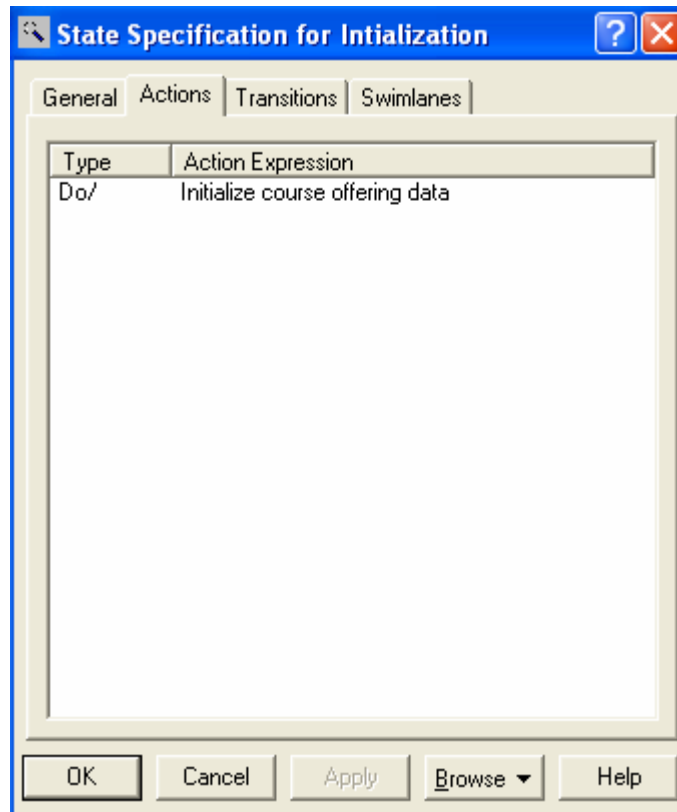
*Figure 7.7 State and activity specifications, General tab*

### State/Activity History

History provides a mechanism to return to the most recently visited state when transitioning directly to a state with substates. History applies to the level in which it appears. It may also be applied to the lowest depth of nested states.

### State and Activity specification-Action Tab

Information about the type and action expression is entered or displayed on this tab.



*Figure 7.8 State and activity specifications, Action tab*

### **Type**

The Type field identifier bar lists the kind of action specified in the Action Specification.

### **Action Expression**

The Action Expression field identifier bar lists the four possible timing options that specify when to carry out an action, and it specifies the types of actions that are carried out.

## State and Activity specification-Transition Tab

Information about the icon, event, and end is displayed on this tab.

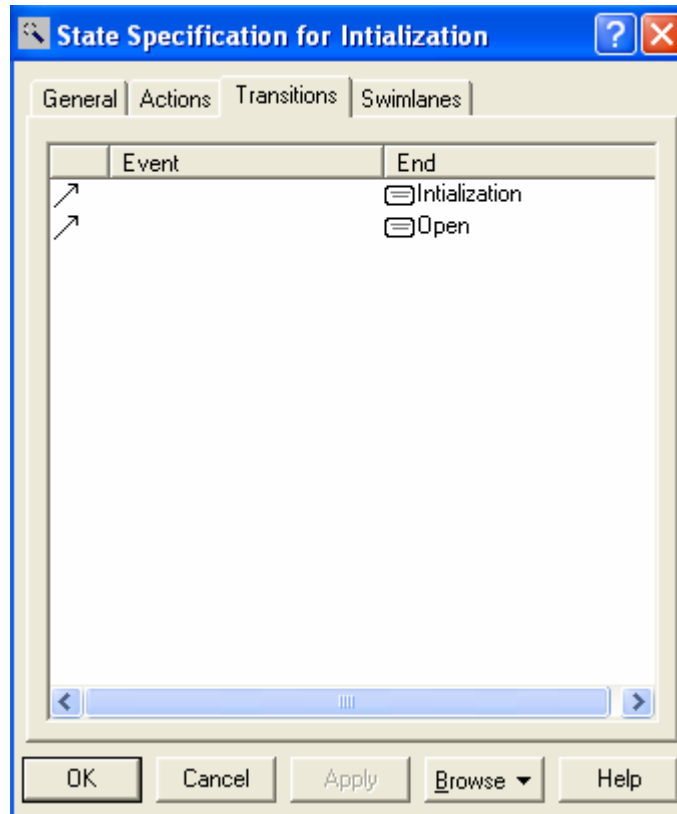


Figure 7.9 State and activity specifications, Transition tab

## 7.7. Action Specification

An Action Specification allows to display and modify the action properties in a statechart diagram or activity diagram.

### State and Activity Actions

Each state and activity on a statechart or activity diagram may contain any number of internal actions. An action is best described as a “task” that takes place while inside a state or activity.

There are four possible actions within a state or activity:

- On Entry
- On Exit
- Do
- On Event

## 7.8. State Transition Specification

A State Transition Specification allows to display and modify the properties and relationships of a transition on a statechart diagram or activity diagram. The State Transition Specification lists the events and actions that are comprised by the transition. The State Transition Specification consists of General and Detail tabs.

### State Transition Specification-Detail tab

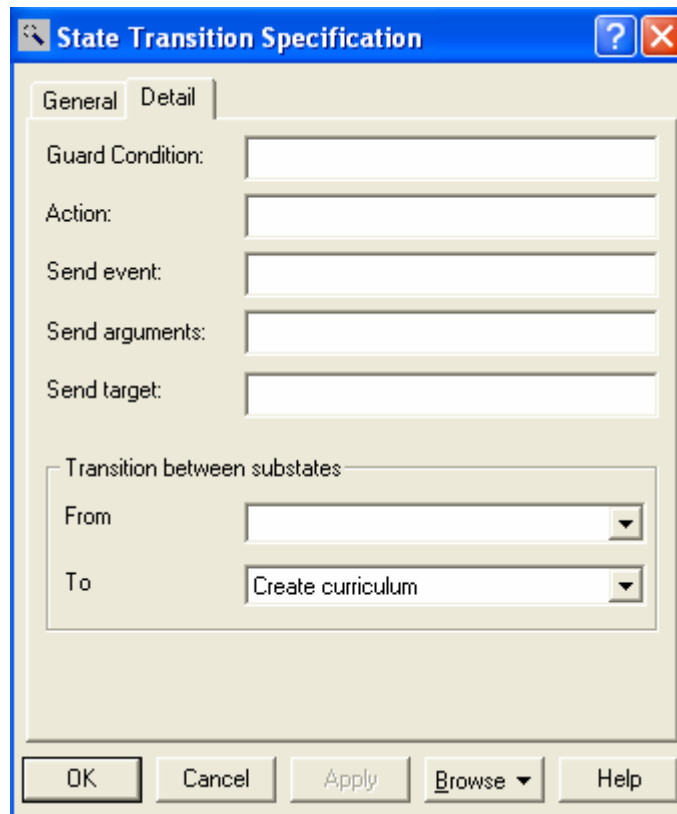


Figure 7.10 State Transition Specification, Detail tab

#### Guard Condition

Conditional state transitions are triggered only when the conditional expression evaluates to true. Conditions are denoted by surrounding brackets:

**Event (args) [condition] / Action ^target.someEvent (args)**

#### Transition Between Substates

Transition between sub-states is useful when a transition is placed to or from a sub-state that has been hidden from view. The From field displays the state name from which the transition



is initiated. The To field displays the state name to which the transition is pointing. Both fields are active at all times.

## **7.9. Decision Specification**

A Decision Specification allows to display and modify the properties and relationships of a decision on a statechart diagram or activity diagram. The Decision Specification consists of the following tabs: General, Transitions, and Swimlanes.

## **7.10. Synchronization Specification**

A Synchronization Specification display and modify the properties and relationships of synchronization on a statechart diagram or activity diagram. The Synchronization Specification consists of the following tabs: General and Transitions.

## **7.11. Object Flow Specification**

An Object Flow Specification display and modify the properties and relationships of an object flow on an activity diagram. The Object Flow Specification consists of the General tab.

## 8. Component Diagram and Specifications

### 8.1. Component Diagram Overview

A component diagram shows the physical dependency relationships (mapping to a file system) between components such as main programs, subprograms, packages, and tasks, and the arrangement of components into component packages. The modeling elements in the component view of architecture are packages and components along with their connections. Component diagrams are contained (owned) either at the top level of the model or by a package. This means the diagram will depict the components and packages in which the diagram is contained.

To create and display component diagram:

- Click **Browse: Component Diagram**.
- On the toolbar, click the component diagram icon.
- On the browser, double-click the component diagram icon.

Every component is assigned to a package. When a component is created using a creation tool from the component diagram toolbox, the component is assigned to the package containing the component diagram.

To reassign a component from one package to another:

- Select a component icon in a diagram directly contained by the package to which the component should be assigned.
- Click **Edit: Relocate**.

### 8.2. Source Code Component

In the component view of the model, a source code component represents a software file that is contained by the package. The type of file is language dependent (e.g. in C++, software components represent .h and .cpp files, in Java they represent .java files, and Power Builder a software component is a .pbl). Each component is assigned a language. Classes in the logical view are mapped to components in the component view. In C++, the mapping is typically one-to-one; that is, one class maps to one component. However, there are times that more than one class will be mapped to a component. This is usually done when there is a very tight coupling between the classes.

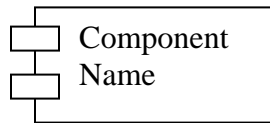


Figure 8.1. UML notation for a Component

### 8.3. Component Diagram Toolbox

The application window displays the following toolbox when the current window contains a component diagram and View: As Unified is selected.

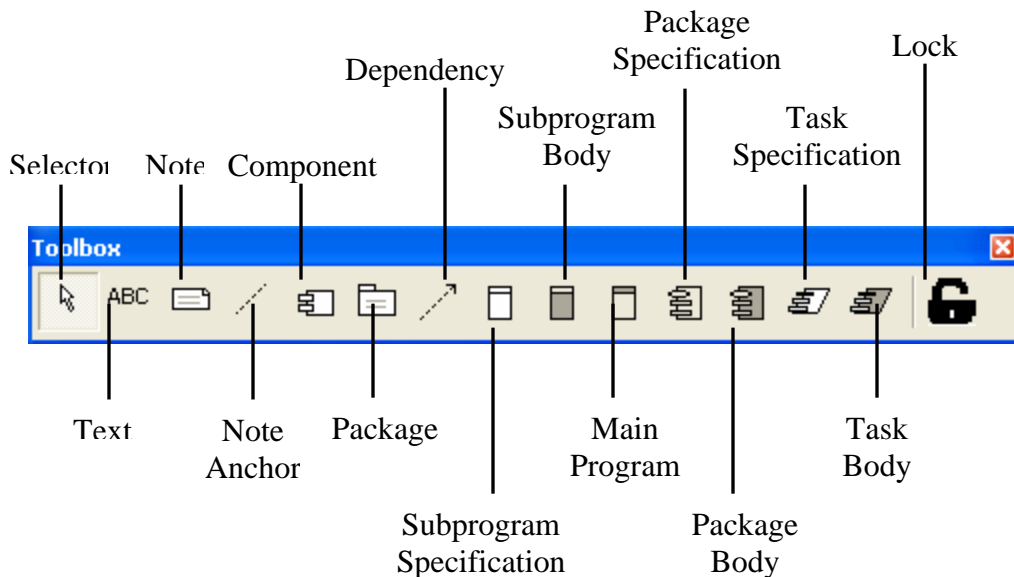


Figure 8.2. Component diagram toolbox

### 8.4. Component Specification

A Component Specification displays and modifies the properties and relationships of each component in the current model. The same specification is used for all kinds of components. Some of the information on this specification can also be displayed inside icons representing the component in a component diagram.

To display a Component Specification, double-click any icon representing the component, or click Browse: Specifications. The Component Specification consists of General, Detail, Realizes, and Files tabs.

## Component Specification-General tab

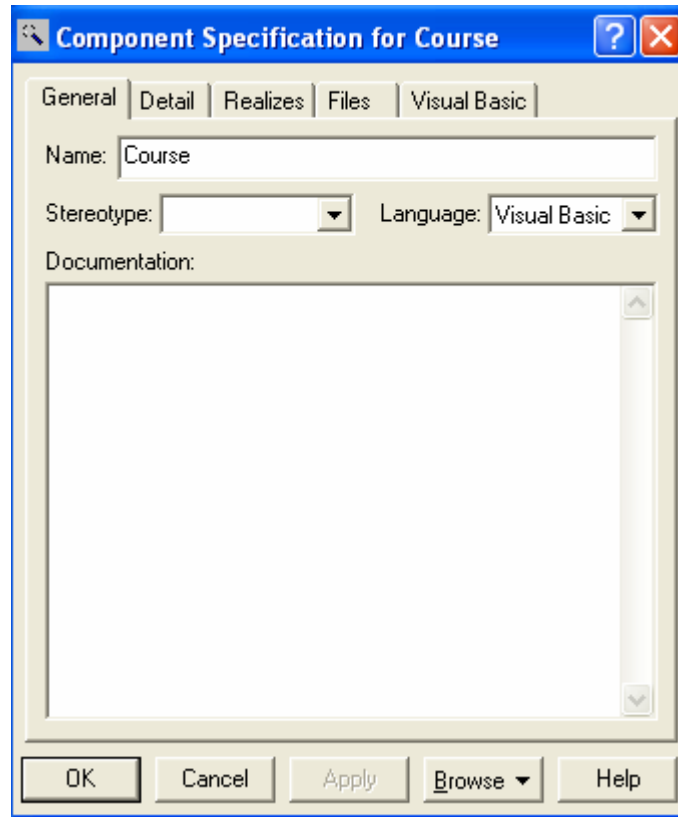


Figure 8.3. Component specification, General tab

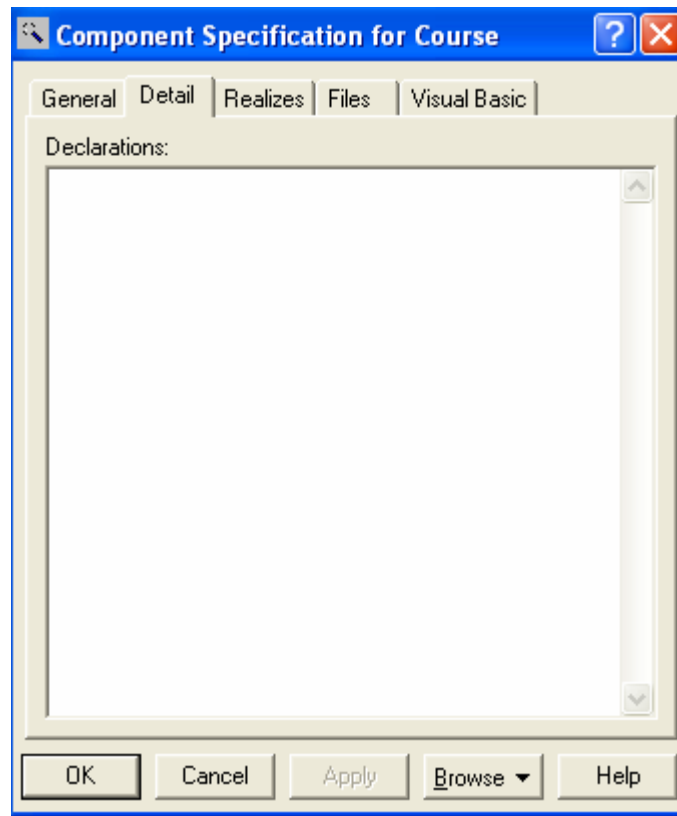
### **Stereotype (Component)**

A component stereotype represents the sub classification of an element. The most common type of components are already predefined as stereotypes, including Main Program, Package Body, Package Specification, Subprogram Body, Subprogram Specification, Task Body, and Task Specification.

### **Language**

This field identifies the implementation language that is assigned to this component. Note that when changing the implementation language of a component, the data types that are used in the specification of operations and attributes of the assigned classes are not automatically converted to data types in the new implementation language. Also, if the implementation language for a component with classes that are assigned to other components is changed, a dialog box that specifies how to handle those classes appears.

## Component Specification-Detail tab



*Figure 8.4. Component specification, Detail tab*

### **Declarations**

The Declarations field contains a list of declarations, such as class names, variables, and other language-specific features (such as #includes or similar constructs). Declarations can include classes, objects, and any other language-specific declarations.

## Component Specification-Realize tab

### **Show All Classes**

A list of all classes in the model will be seen, if this check box is selected. If this check box is cleared, only classes that are assigned to this component will be seen.

## Classes

The list identifies the classes and interfaces that are assigned to this component (indicated with check marks). The Logical Package column shows to which package a class belongs, and the Language column shows the programming language that is assigned to a specific class. A class or interface can be assigned to a component through Assign on the shortcut menu in the list, or by dragging a class or interface from the browser and dropping it in this list. Classes that are unassigned or classes that are assigned to components with the same implementation language as this component can be assigned.

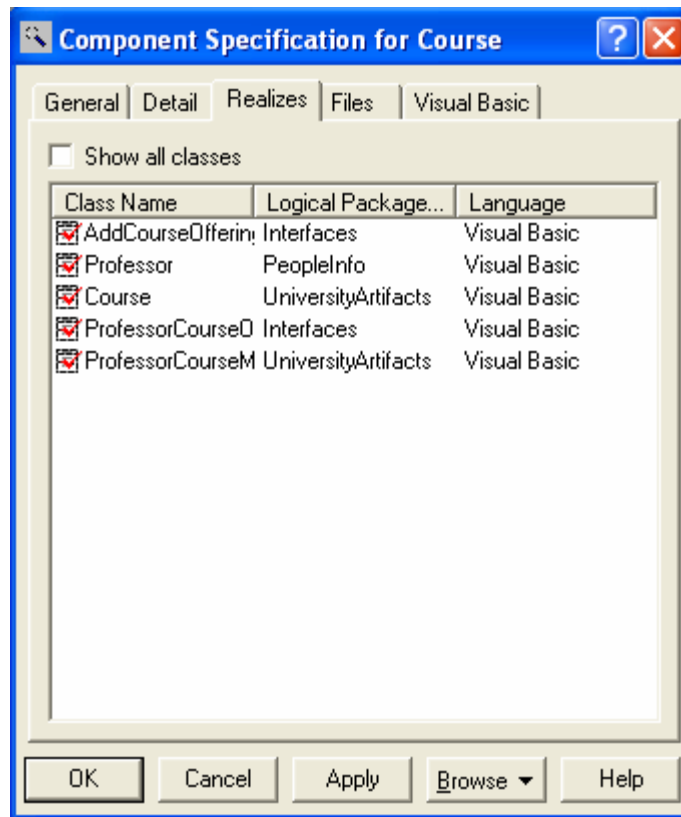


Figure 8.5. Component specification, Realize tab

## Language

This field identifies the implementation language that is assigned to this component. When changing the implementation language of a component, the data types that are used in the specification of operations and attributes of the assigned classes are not automatically converted to data types in the new implementation language.

## 8.5. Package Specification

A Package Specification displays and modifies the properties and relationships of a package in the current model. To display a Package Specification, double-click any icon representing the package, or click Browse: Specifications. The Package Specification consists of General, Detail, and Files tabs.

### Package Specification-General tab

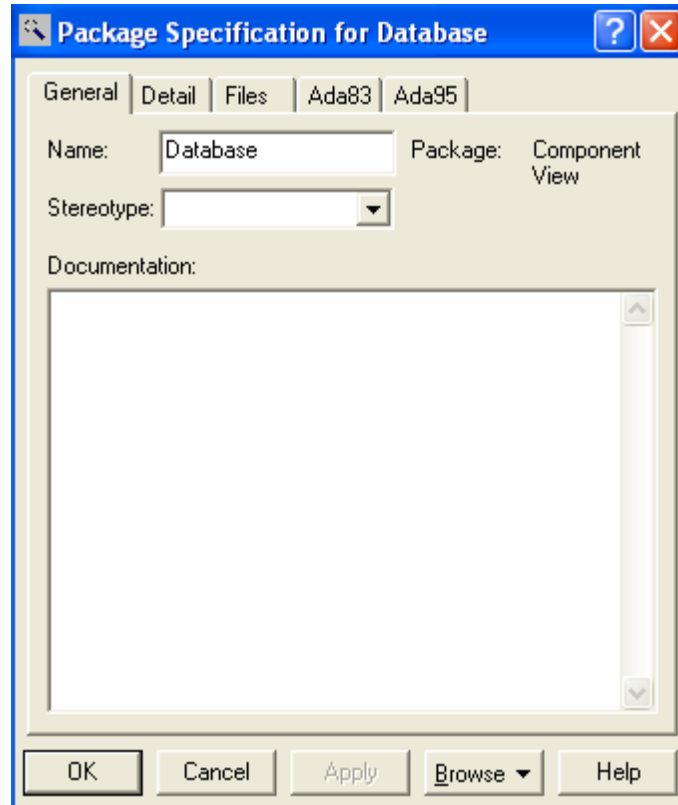


Figure 8.6. Package specification, General tab

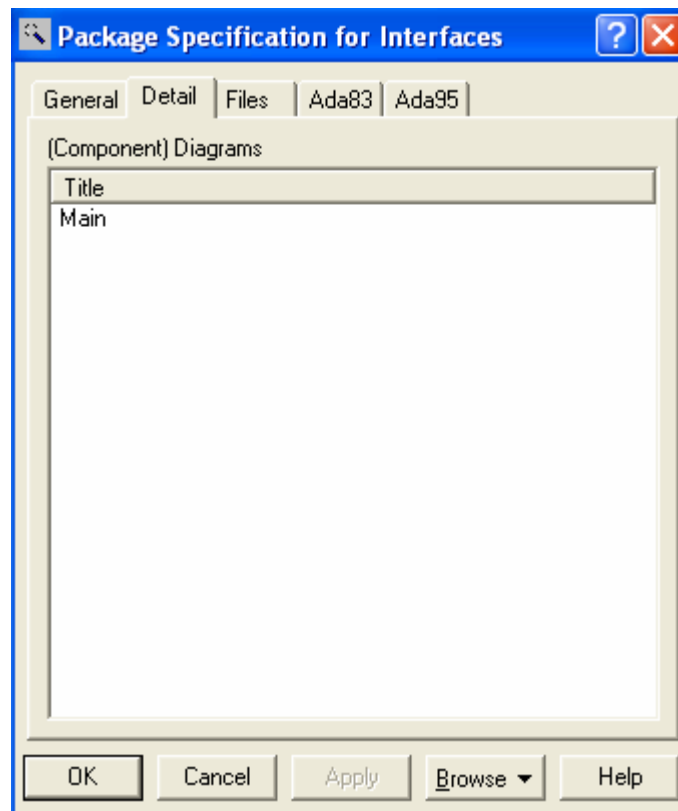
### Package

The package the component belongs to is displayed in this static field.

## Package Specification-Detail tab

### Component Diagrams

This field lists the component diagrams contained in the package. A new component diagram in the package can be created through Insert on the shortcut menu, or click Browse: Component Diagram. Component diagrams can be renamed or deleted from this field. To display a specific component diagram listed in this field, double-click its entry.



*Figure 8.7. Package specification, Detail tab*



## 8.6. Software components in the JU Course Registration Problem

This is a relatively simple system and the decision was made to provide a many-to-one mapping between classes and component.

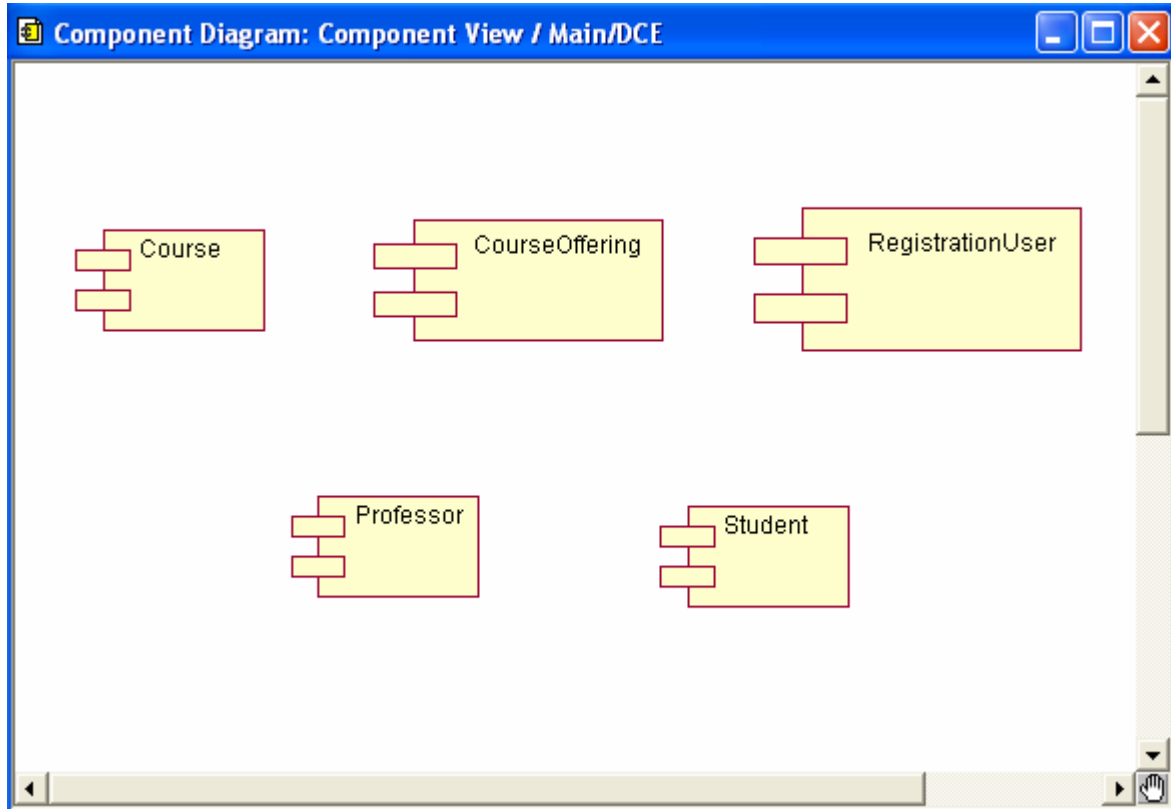


Figure 8.8. Component (Software) diagram

## 9. Deployment Diagram and Specifications

### 9.1. Deployment Diagram Overview

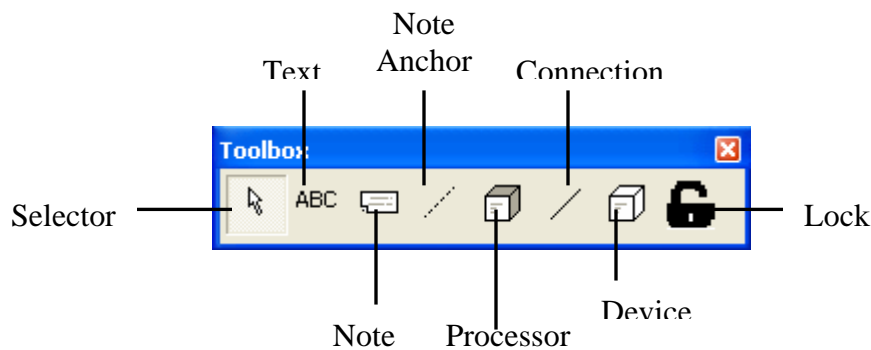
A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram that shows the connections between processors and devices, and the allocation of its processes to processors.

To create and display deployment diagram:

- Click **Browse: Deployment Diagram**.
- On the toolbar, click the deployment diagram icon.
- In the browser, double-click the deployment diagram icon.

### 9.2. Deployment Diagram Toolbox

The application window displays the following toolbox when the current window contains a deployment diagram and View: As Unified is selected.



*Figure 9.1. Deployment diagram toolbox*

### 9.3. Processor Specification

A Processor Specification displays and modifies the properties and relationships of a processor in the current model. Some of the information on the specification can also be displayed inside icons representing the processor in a model's deployment diagram. To display a Processor Specification, double-click any icon representing a processor, or click Browse: Specifications. The Processor Specification consists of General and Detail.

## Processor Specification-Detail Tab

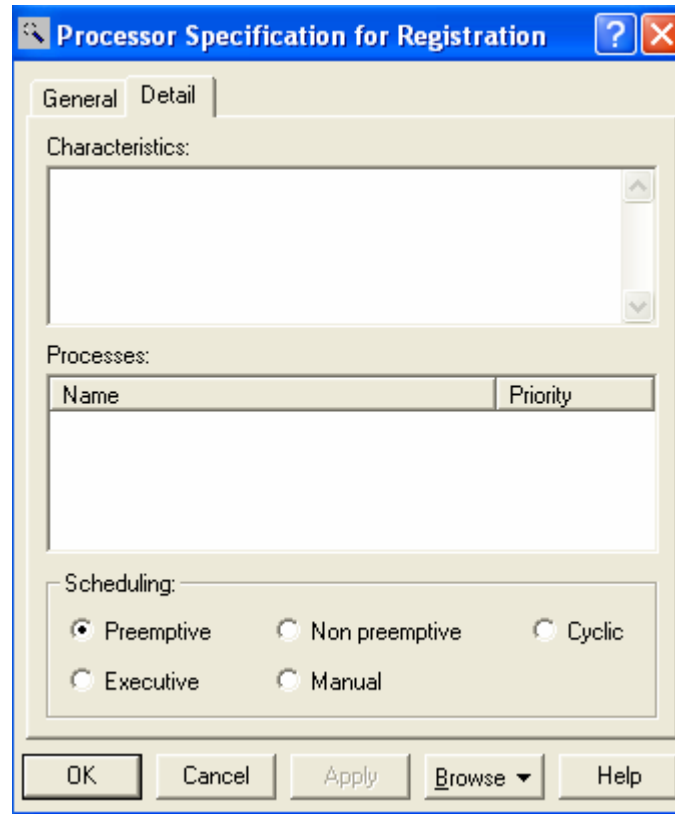


Figure 9.2. Processor Specification, Detail tab

### Characteristics

Characteristics field is used to specify a physical description of an element. For example, it can describe the kind and bandwidth of a connection; the manufacturer, model, memory, and disks of a machine; or the kind and size of a device. This field can be set only through the specification. This information is not displayed in the deployment diagram. To update this field, click the Characteristics field and enter the information.

### Processes

This field is used to identify the processes assigned to this processor. Processes denote either the root of a main program from a component diagram or the name of an active object from a collaboration diagram. To create a process, right-click in the processes area and click Insert from the shortcut menu. A new process entry is created. To change the name or priority, click the item and type the changes. A list of the processes can be displayed by selecting the processor icon and clicking Show Processes from the shortcut menu.

## **Scheduling**

The Scheduling field specifies the type of process scheduling used by the processor. These options are used to specify the appropriate scheduling.

### **Preemptive (Default)**

Higher-priority processes that are ready to execute can preempt lower-priority processes that are currently executing. Processes with equal priority are given a time slice in which to execute, allowing computation resources to be fairly distributed.

### **Non preemptive**

The current process continues to execute until it relinquishes control.

### **Cyclic**

Control passes from one process to another; each process is given a fixed amount of processing time.

### **Executive**

An algorithm controls process scheduling.

### **Manual**

Processes are scheduled by a user outside of the system.

## **9.4. Device Specification**

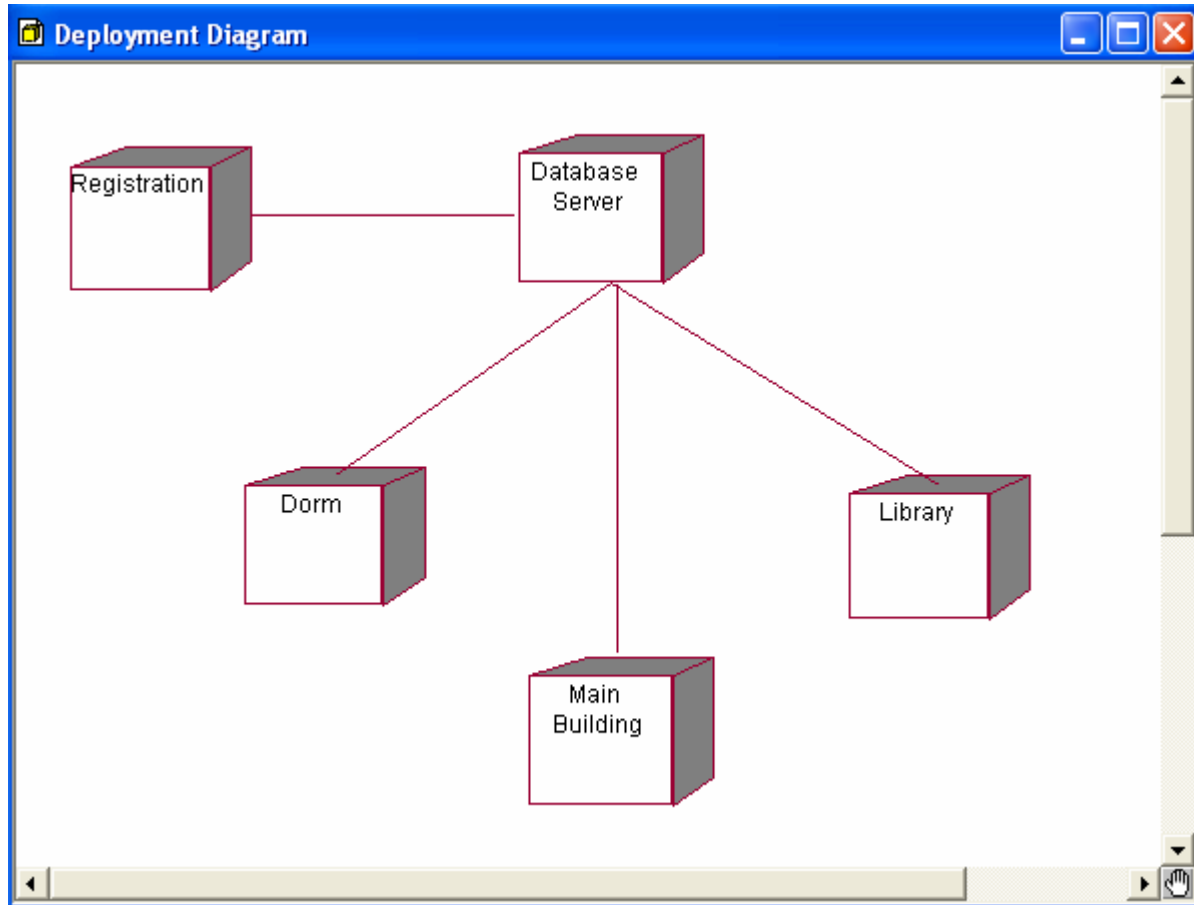
A Device Specification displays and modifies the properties and relationships of a device in the current model. Some of the information on this specification can also be displayed inside icons representing the device in a deployment diagram. To display a Device Specification, double-click any icon representing a device, or click **Browse: Specifications**. The Device Specification consists of General and Detail tab.

## **9.5. Process Specification**

Processes are threads of control that execute on a processor. One process specification documents one thread of control. The Process Specification can be accessed through the Processes field of a Processor Specification. None of the information contained in the Process Specification is displayed in a diagram; thus, process properties can only be viewed and modified through a Process Specification. The Process Specification consists of the General tab.

## 9.6. Deployment Diagram for the JU Course Registration System

After studying the component packages defined for the problem, examining existing hardware, and estimating the load on the system during the course registration period, it is required to have five processors for the system, one to handle the professor executable, one for the database, and three for student registration.



*Figure 9.3 Deployment diagram for the JU course registration problem*

## 10. Code Generation and Reverse Engineering with Visual Basic

### 10.1 Code Generation

This portion contains a step-by-step guide to Visual Basic code generation and reverse engineering. The following steps should be followed to generate code:

#### **Step 1: Assign the Visual Basic Language to the components.**

Components must be assigned a language. The language of a component is set for all classes assigned to the component.

#### **Step 2: Assign classes to components.**

Once components have been created, classes are assigned to the components. The components represent a Visual Basic project.

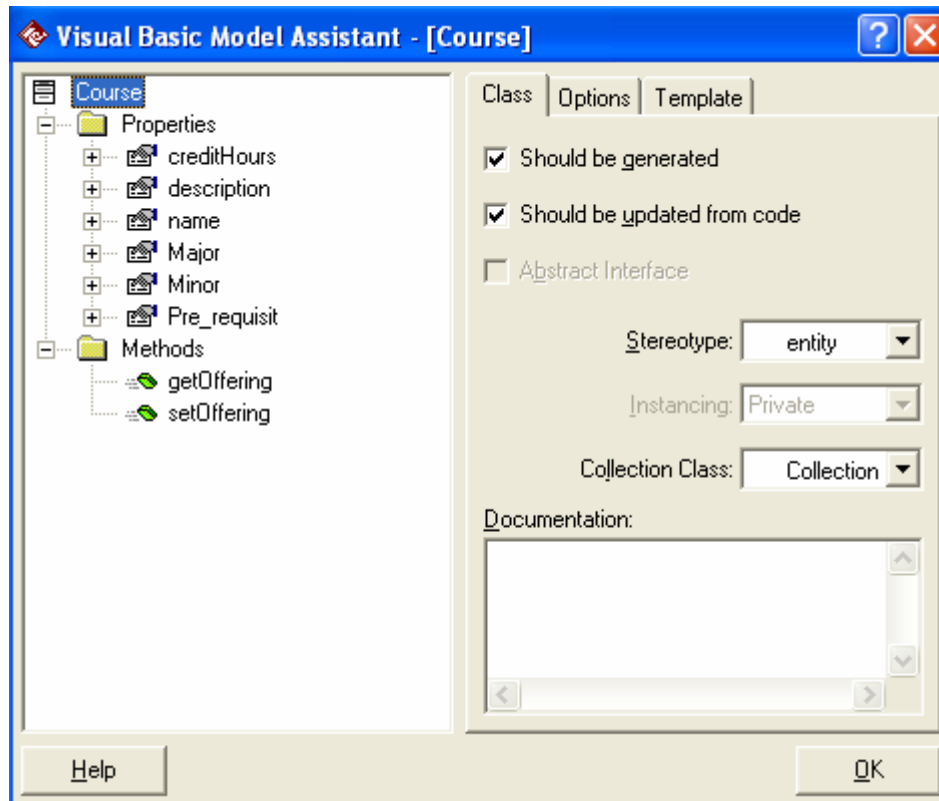
#### **Step 3: Use the Model Assistant Tool to set code generation properties.**

The Model Assistant Tool maps modeling elements in Rational Rose to visual Basic constructs. In Visual Basic, the Model Assistant Tool may be used to create and specify constants, declare statements, event statements, enum and type declarations, properties, methods, and method parameters. It also allows to set procedures for class properties and association roles, and to define and create a user-defined collection class for the class.

To start the Model Assistant tool:

- Right-Click on the class in the Browser or on the class diagram.
- Select the Model Assistant menu choice.

The Model Assistant Tool for a class assigned the Visual Basic language is shown in figure 10.1.



*Figure 10.1 Visual Basic Model Assistant*

**Step 4: Select the components and use the Code Update Tool to generate the code.**

The code Update Tool is used to generate the Visual Basic code. Code may be generated for all components in a package, a single component, or a set of components.

To start the Code Update tool:

- Right-Click on the component in the Browser or on a component diagram.
- Select the Update Code menu choice.

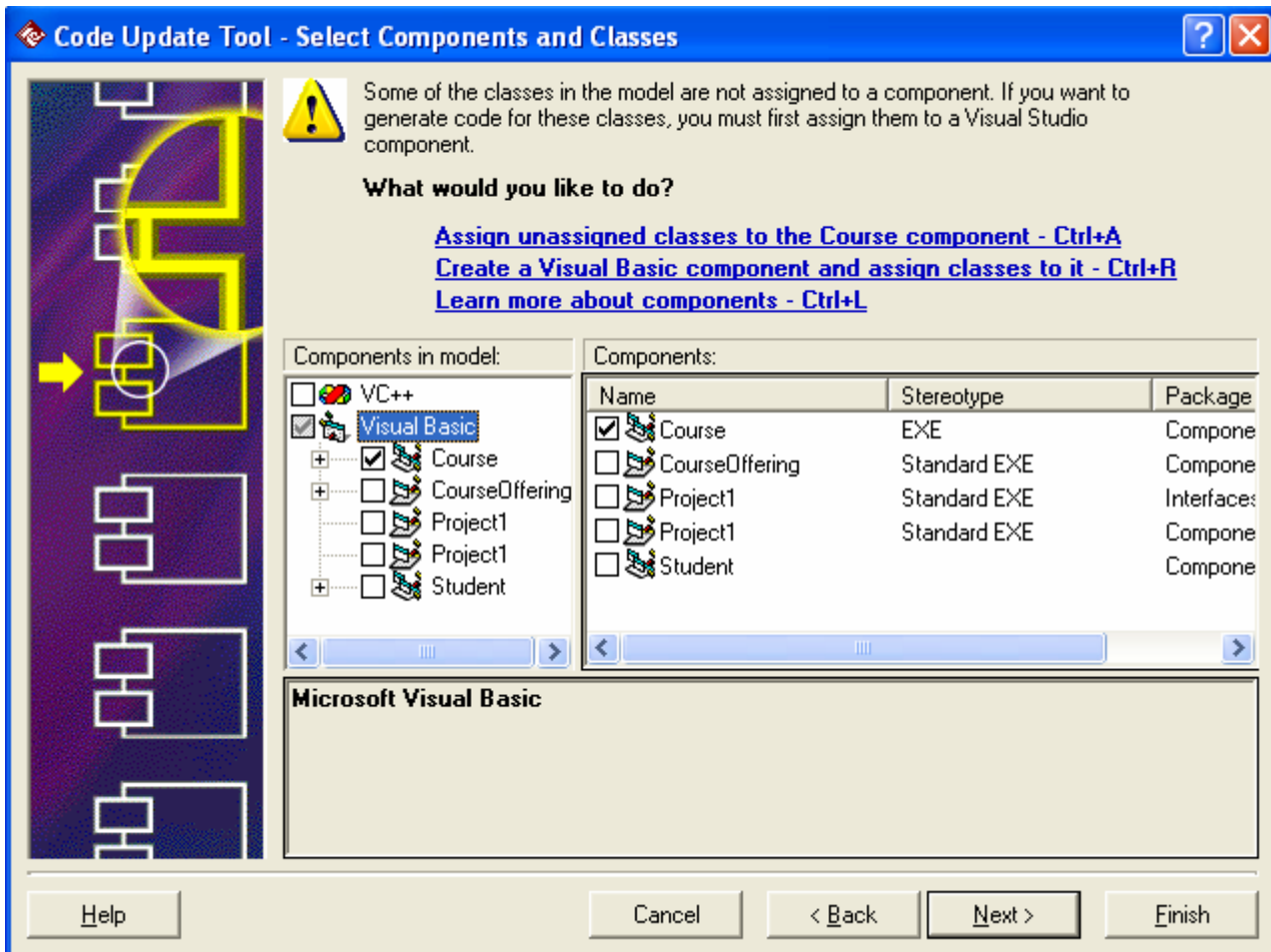


Figure 10.2. Code Update Tool

### Step 5: Evaluate the code generation errors.

When the code generation process is completed, the summary window is displayed in the Code Update tool. The summary tab contains information about the generated code and all code generation errors are written to the Log, which can be viewed by selecting the Log tab of the summary window.

## 10.2 Reverse Engineering

Once the coding in visual basic is completed it is needed to update the model to reflect any changes. This can be accomplished by using the Model Update tool. This tool can also be used to create an initial model for existing code.

To update (create) a model from code using the model update tool:

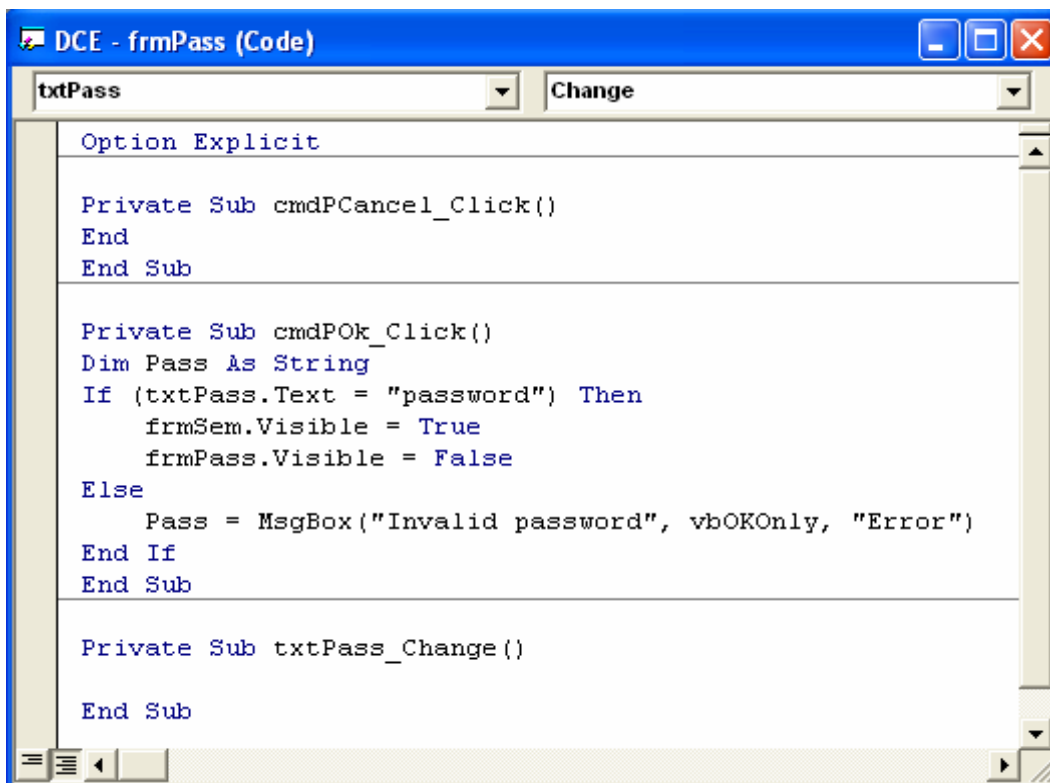


- Select the **Tools: Visual Basic: Update Model** from code menu command.
- Follow the steps of the wizard.

When the reverse engineering process is completed, the summary window is displayed in the model Update tool.

### 10.3 Visual Basic code for Select Course To Teach Use Case

The task here is to build an application that allow the professor to access the system and select course to teach. The professor is also able to delete any course from the system. The user of the application is the professor or instructor. After strictly following the above procedure, visual basic application provide the necessary component specified by Rational Rose during analysis and design phase. The following figures shows the code and the out put for the Select to Teach Course use case.



```
Option Explicit

Private Sub cmdPCancel_Click()
End
End Sub

Private Sub cmdPOk_Click()
Dim Pass As String
If (txtPass.Text = "password") Then
    frmSem.Visible = True
    frmPass.Visible = False
Else
    Pass = MsgBox("Invalid password", vbOKOnly, "Error")
End If
End Sub

Private Sub txtPass_Change()
End Sub
```

(a)



(b)

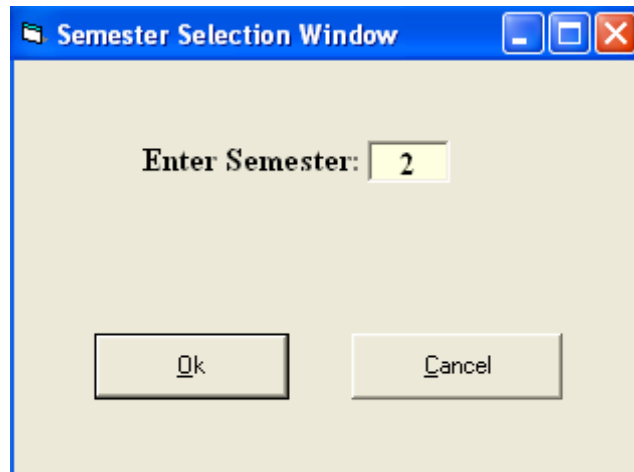
Figure10.3 a) Code for password entry b) Out put for the code

```
cmdSOk Click
Option Explicit

Private Sub cmdSCancel_Click()
End
End Sub

Private Sub cmdSOk_Click()
Dim Sem As String
If (txtSem.Text = "2") Then
    frmOption.Visible = True
    frmSem.Visible = False
Else
    Sem = MsgBox("Invalid Semester", vbOKOnly, "Error")
End If
End Sub
```

(a)



(b)

Figure 10.4 a) Code for semester entry b) Out put for the code

```

Private Sub cmdCAdd_Click()
Data1.Recordset.AddNew
End Sub

Private Sub cmdCCancel_Click()
End
End Sub

Private Sub cmdCDelete_Click()
On Error Resume Next
Dim f As String
Data1.Recordset.Delete
If Not (Data1.Recordset.EOF) Then
Data1.Recordset.MoveNext
ElseIf Not (Data1.Recordset.BOF) Then
Data1.Recordset.MovePrevious
Else
f = MsgBox("This was the last Course in the table")
End If
End Sub

Private Sub cmdCOK_Click()
On Error GoTo CancelUpdate
Data1.Recordset.Update
Exit Sub
CancelUpdate:
MsgBox Err.Description
Data1.Recordset.CancelUpdate
End Sub

```

(a)

The image shows a graphical user interface window titled "Course Addition and Deletion Window". The window has a blue title bar with standard minimize, maximize, and close buttons. The main area is light beige and contains several input fields and buttons. The fields are: "Course Name" with the text "Artificial Intelegency", "Prerequisite" with a hyphen "-", "Course Number" with "CC-243", "Credit Hour" with "3", "Professor Name" with "Dr. John", and "Professor ID" with "1234". Below the fields are four buttons: "Ok", "Add Course", "Delete Course", and "Cancel". At the bottom of the window is a navigation bar with four arrows (left, right, first, last) and the text "Click on the arrows to navigate through courses".

(b)

Figure 10.5 a) Code for course Addition and Deletion b) Out put of the code

## **11. Rational RequisitePro**

### **11.1 Overview**

Studies have shown that managing requirements is the most significant factor in delivering projects on time, on budget, and on target. RequisitePro helps projects succeed by giving teams the ability to manage all project requirements comprehensively and facilitating team collaboration and communication. Moving beyond conventional requirements management, RequisitePro combines both document-centric and database-centric approaches. By deeply integrating Microsoft Word with a multi-user database, RequisitePro allows to organize, prioritize, trace relationships, and easily track changes to requirements. The program's unique architecture and dynamic links make it possible to move easily between the requirements in the database and their presentation in Word documents. Requirements drive the entire project. RequisitePro's integration with other industry-leading tools optimizes the flow of requirements data throughout the project, promoting consistency and ensuring that what is designed, tested, documented, and delivered meets the users' needs. RequisitePro's deep integration with other lifecycle tools promotes artifact reusability and eases the sharing of information, further enhancing team collaboration.

#### **Requirements**

RequisitePro organizes requirements and provides traceability and change management throughout the project lifecycle. A requirement describes a condition or capability that a system must provide. Requirements contain a name and text, and they can be qualified with attributes to provide specific details. Requirements can be created in a document or in a view. All requirements information is stored in the database.

#### **Requirements Type**

A requirement type is an outline for requirements. Requirement types are used to classify similar requirements so they can be efficiently managed. When a requirement type is defined, common set of attributes, display style, and tag numbering are also defined.

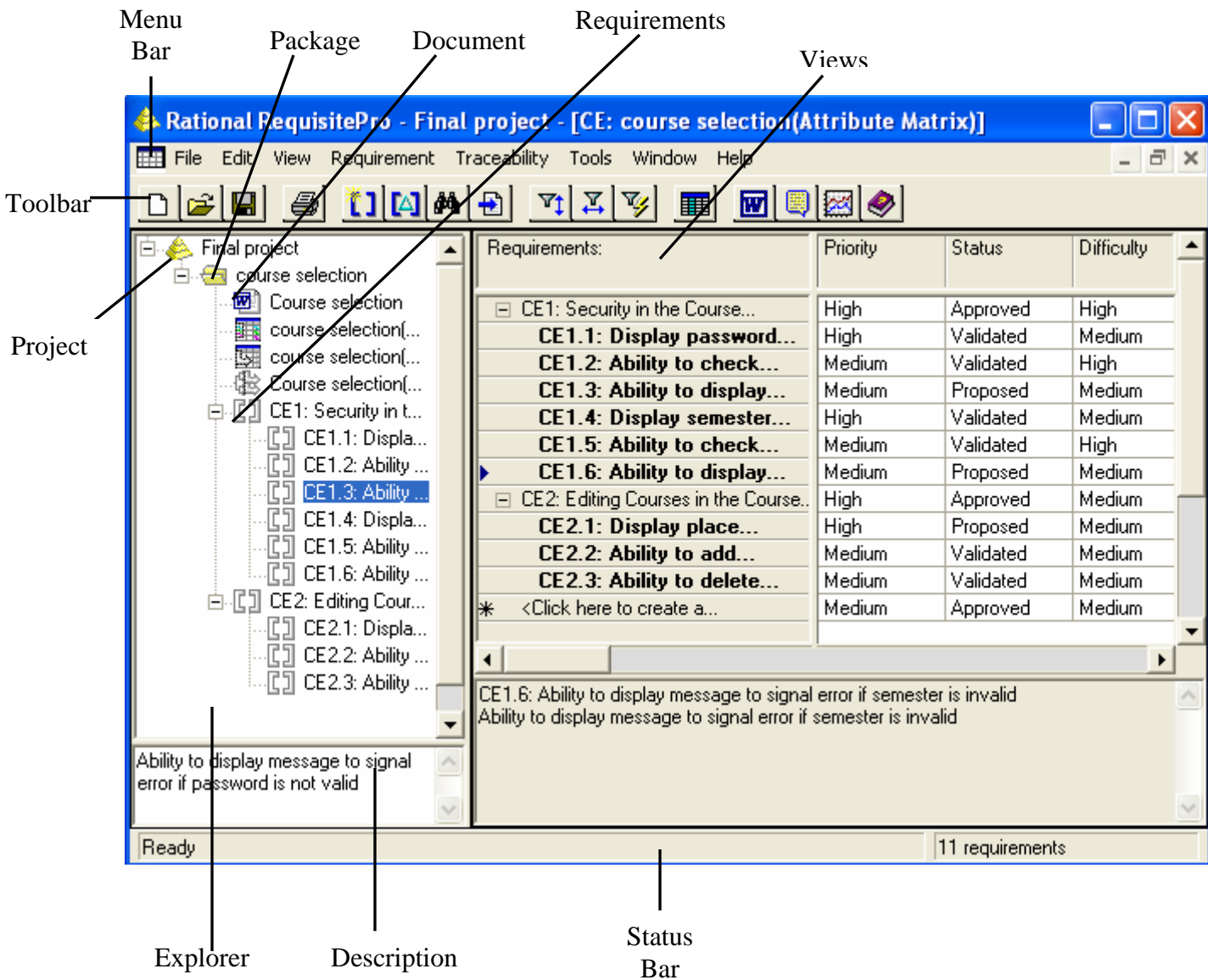


Figure 11.1 RequisitePro application window.

### Requirements Attributes

In RequisitePro, requirements are classified by their type and their attributes. An attribute provides information to manage a requirement. Attributes can provide crucial information to help a team plan, communicate, and monitor project activities from the analysis and design phases through the release phase.

Attribute information may include the following:

- The relative benefit of the requirement.
- The cost of implementing the requirement.
- The priority of the requirement.

- The difficulty or risk associated with the requirement.
- The relationship of the requirement to another requirement.

RequisitePro provides several default requirement attributes, such as Priority (high, medium, low), Status (proposed, approved, incorporated, validated), Cost, and Difficulty.

## **Project**

A RequisitePro project includes a requirements database and its related documents. A project is usually created by a project administrator, who determines the project structure and sets up security permissions for the project's users. Although all project users are encouraged to view and query requirements and to participate in discussions, only a limited group of users create and manage requirements within a project.

## **Project Database**

The project database is the requirements database managed by RequisitePro. In RequisitePro, use one of three physical databases to store requirements: Microsoft Access, Oracle, or Microsoft SQL Server. Each RequisitePro project has its own database, where all the requirements for a project are stored (With the exception of Microsoft Access, all of these databases may contain more than one project.). In the project database, requirements can be added, modified, or deleted. When requirements are changed in a document, the changes are updated in the database.

## **Project Version Control**

RequisitePro's version control allows to trace change by archiving projects. It can be possible to manage multiple versions of the projects, retrieving, modifying, and returning revisions to the archive in an organized and consistent manner. From RequisitePro, use RequisitePro's Archive command.

## **Project List**

A RequisitePro project list is a personal library of accessible RequisitePro projects. Each user's list is unique. For example, a project administrator who monitors the progress of all the projects scheduled for completion this quarter could have an extensive list of projects, whereas some users might have just one project in their list at a time. Project administrators store new projects in their file systems typically in the RequisitePro Project directory.

## Explorer

The Explorer is RequisitePro's primary navigation window. In this window, project artifacts (documents, requirements, views, and packages) are displayed hierarchically in a tree browser. Project information is organized in packages, which are units of related artifacts. The project's root package is displayed as the project node, and the contents of each root package are displayed beneath it.

## 11.2 RequisitePro Project

A Rational RequisitePro project provides the framework within which project artifacts are organized and managed. Each project includes the following: a database, documents, packages, document types, requirements, requirement types, attributes, attribute values, discussions, traceability relationships, saved personal and project wide views, revision histories, and security information. Each project resides in a separate directory. This storage method simplifies the process of organizing, archiving, and managing project files.

When a project is created in RequisitePro, there are choices of basing the project on a blank template, on one of three templates included with RequisitePro, or on a template created from an existing project. When a project is created based on a template, the document and requirement types, attributes, security information, package structure, and data of the selected template are copied to the new project. If a project template created with the Include project data option selected is used, the template also copies the following project data from the existing project: packages, requirements, documents, views, and history. If there is a baseline of a RequisitePro project, the baseline can be used to create new projects in RequisitePro.

Three project templates are shipped with RequisitePro;

- **Use-Case Template:** - The Use-Case Template is ideal for implementing the Rational Unified Process. This template is designed for RequisitePro projects that use the RequisitePro integration with Rational Rose use cases and ClearQuest enhancement requests. Use cases are particularly applicable to object-oriented software design using the Unified Modeling Language and for applications that are user intensive.
- **Traditional Template:** - The Traditional Template is best suited for projects that use declarative requirements specifications. This template includes a traditional Software Requirements Specification outline rather than use cases.



- **Composite Template:** - The Composite Template allows to combine the best qualities of both use-case modeling and traditional requirements specification techniques. This template provides an outline for a modern software requirements specification package applying both traditional document-based techniques and use-case modeling.

Before creating a RequisitePro project, decide which database to use to store the requirements information. The currently supported databases are Microsoft Access and the enterprise databases Oracle and Microsoft SQL Server.

To create RequisitePro project:

- Open the Create Project dialog box by doing one of the following:
  - From RequisitePro, click **File: New: Project**.
  - From Rational Administrator, in the Configure Project dialog box, click **Create** in the **Requirement Asset** area.
- One of the following project templates should be selected:
  - Blank template, and set each project parameter.
  - One of the default templates provided with RequisitePro.
  - A template from an existing project, using the Project Template Wizard.
- The Rational RequisitePro Project Properties dialog box opens when OK button is clicked.
- Type general information about the new project.
  - Type a project name (up to 64 characters).
  - Type the path for the project directory, or click Browse and select the directory. After the project is created, this field cannot be changed.
  - A database in which to store the project should be selected. Projects can be created in Microsoft Access, SQL Server, or Oracle. If a database other than Microsoft Access is selected, click Properties to configure the database.
  - Type a description of the project.
- Click OK.
- Click Yes to create the project directory, and then click Close when the directory has been successfully created.

### 11.2.1 Project Template

Creating a template based on an existing RequisitePro project copies the existing project's structure and security, including document types, requirement types, attributes, and user and group permissions. There is also an option to include project data (packages, requirements, documents, views, and history). An icon for the new template appears in the Create Project dialog box, and can be selected when creating new projects. Note that after creating a project template, it cannot be modified.

- Open the Create Project dialog box by doing one of the following:
  - From RequisitePro, click **File: Project: New**.
  - From Rational Administrator, in the Configure Project dialog box, click the **Create** button in the **Requirement Asset** area.
- Double-click the **Make New Template** icon. The Project Template Wizard opens.
- At the Enter Template Information screen, the following information should be typed:
  - **Template Name**: A name for the new template.
  - **Template Location**: A location for the new template.
  - **RequisitePro Project**: RequisitePro project on which to base the template structure.
  - **Include Project Data**: This check box is selected if the template needed to include data from the selected project in addition to the project structure.
  - **Documentation File (optional)**: It selects an existing .rtf file that contains details about the template.
  - **Icon File**: Selects an icon for the template that will appear in the Create Project dialog box.
- The Next button is clicked to continue.
- The information is checked at the Confirm screen and Finish button is clicked to create the template. The new template appears in the Create Project dialog box.

## 11.3 Working in view

Rational RequisitePro views use tables or outline trees to display requirements and their attributes or the traceability relationships between different requirement types. RequisitePro includes powerful query functions for filtering and sorting the requirements and their attributes in views. A view is an environment for analyzing and printing requirements. Multiple views can be opened at one time, and scroll can be used to view all requirements and attributes in the table or tree. The number of requirements in the current view appears in the lower right corner of the views window.

It can be possible to create three kinds of views:

- The Attribute Matrix view displays all requirements and their attributes within a specified type.
- The Traceability Matrix view displays the relationships between requirements of two types.
- The Traceability Tree view displays the chain of traceability through the project requirements. A Traceability Tree can be set up in one of two directions: **traced out of** requirements of a specified type or **traced into** requirements of a specified type.

All views commands are located in the toolbar. To navigate in a view quickly, use the arrow keys, the PAGE UP or PAGE DOWN keys, and the HOME and END keys. Use the right button on the mouse as a shortcut to commands that are specific to requirements and attributes. Move the mouse over the requirements and attribute labels and click the right button to access these shortcut menus.

While a view is open, other users may add, delete, or modify requirements that are displayed in the view. These changes are not automatically reflected in the opened view or in the Explorer. To update the data displayed in the opened view and in the Explorer for requirements that were modified by other users, click in the view or in the Explorer and then click **View: Refresh** (or the Refresh the view button). The Refresh command updates the displayed content.

### 11.3.1 The Attribute Matrix

The Attribute Matrix is a spreadsheet-like display that lists requirements of a specific requirement type and their attributes. Requirements are arranged in rows, listed by tag number and followed by requirement name (or requirement text, if the requirement has not been assigned a name). Attributes are arranged in columns. The Attribute Matrix displays all requirements, and requirements can be created in the database from this view.

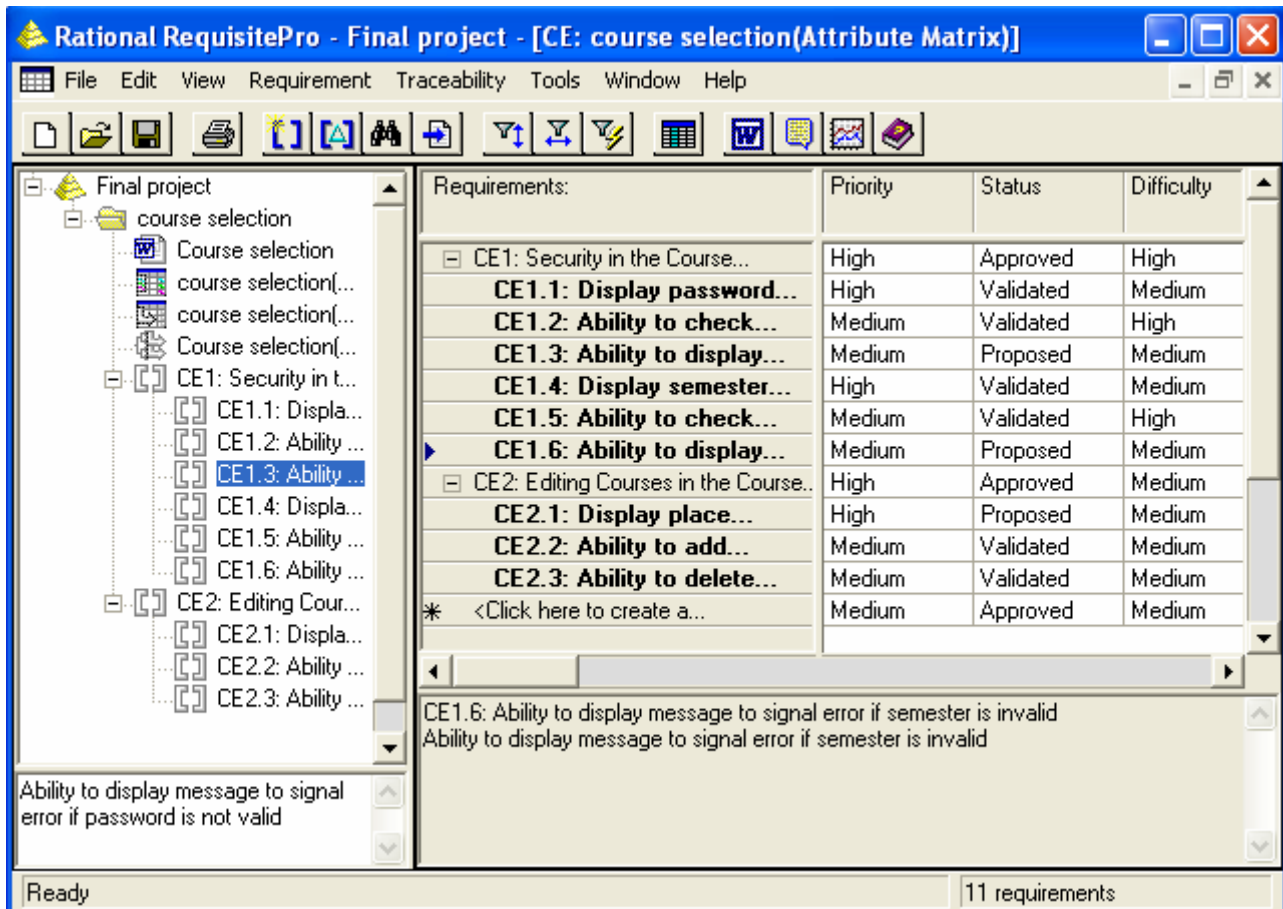


Figure 11.2 Attribute Matrix for JU Registration system

## Requirements

In an Attribute Matrix, a requirement is displayed in a single row, showing the requirement tag followed by the requirement name (or requirement text if the requirement has not been assigned a name). An \* marks the beginning of the empty row at the bottom of the matrix, where a new requirement is inserted. A pencil icon at the beginning of a row indicates that the entered requirement information has not yet been saved.

## Attributes

Attribute labels are listed at the top of each column. Corresponding attribute values are listed beneath the attribute labels. Query information is displayed here as well. Clicking **View: Displayed Attributes** selects attributes that are displayed in the view. The Attribute Matrix displays all internal and external traceability relationships in the **Traced-to** and **Traced-from** columns; suspect traceability relationships are denoted with an (s) in the relevant column.

## Text Pane

The text pane, located at the bottom of the Attribute Matrix, displays a requirement's tag, name, and text, and it reads “Multiple requirements selected” if more than one requirement have been selected. This field is read-only. Graphics and OLE objects included in requirement text are displayed as small rectangular symbols. For Word-linked files included in the requirement text, the path and time stamp of the linked file are presented. The default size of the text pane is two lines, but the size can be increased by dragging the border.

### 11.3.2 The Traceability Matrix

The Traceability Matrix displays and manipulates the relationships between two requirement types. The requirements can be of the same type or of different types, and they include all internally and externally mapped requirements. This view is used to create, modify, and delete traceability relationships and to view indirect relationships. The Traceability Matrix also shows traceability relationships that are marked as suspect. If a Traceability Matrix of requirements of the same type is displayed, it shows hierarchical relationships that are marked as suspect.

In this view, a requirement is traced to or traced from another requirement. For example, Requirement B is traced from Requirement A if it was directly or indirectly derived from Requirement A. If Requirement A is the basis of several other requirements, Requirement A is traced to these requirements. An arrow pointing from one requirement to another indicates that a direct traceability relationship exists between the requirements. A dotted line arrow indicates an indirect relationship. Rows and columns can be resized. The intersection of a row and column is called a cell.

To change the number of lines of requirement name and/or requirement text displayed, click **View: Row/Column Sizing** and then select a different value for the row or column size.

#### Cell

- ❖ If the cell is blank, no relationship exists.
- ❖ If an arrow points upward a row requirement, the row requirement is traced to the column requirement.
- ❖ If an arrow points down toward a row requirement, the row



**Traced to arrow**



**Traced from arrow**

requirement is traced from the column requirement.

- ❖ If a dotted line arrow is displayed, an indirect relationship exist.
- ❖ If an arrow with a red diagonal line through it is displayed, the traceability relationship is suspect.
- ❖ If a triangle with a red diagonal line through it is displayed, the hierarchical relationship is suspect.



**Indirect relationship**



**Suspect relationship**



**Suspect relationship**

### **Text Panes**

The Traceability Matrix displays two text panes at the bottom of the window. These panes are used to review the requirements in each area, so that traceability relationships can easily be created, modified, and deleted. The default size of each pane is set to one line and can be resized. The top text pane displays the name (or text) of the currently selected row requirement, and the bottom pane displays the name (or text) of the currently selected column requirement. These fields are read-only. If the selected requirement has been assigned a name, that name appears in the text pane. Otherwise, the requirement text appears in the text pane. When more than one row or column is selected, the corresponding text pane displays the message “Multiple requirements selected.” Graphics and OLE objects included in requirement text appear as small rectangular symbols. For Word-linked files included in requirement text, the path and time stamp of the linked file are presented.

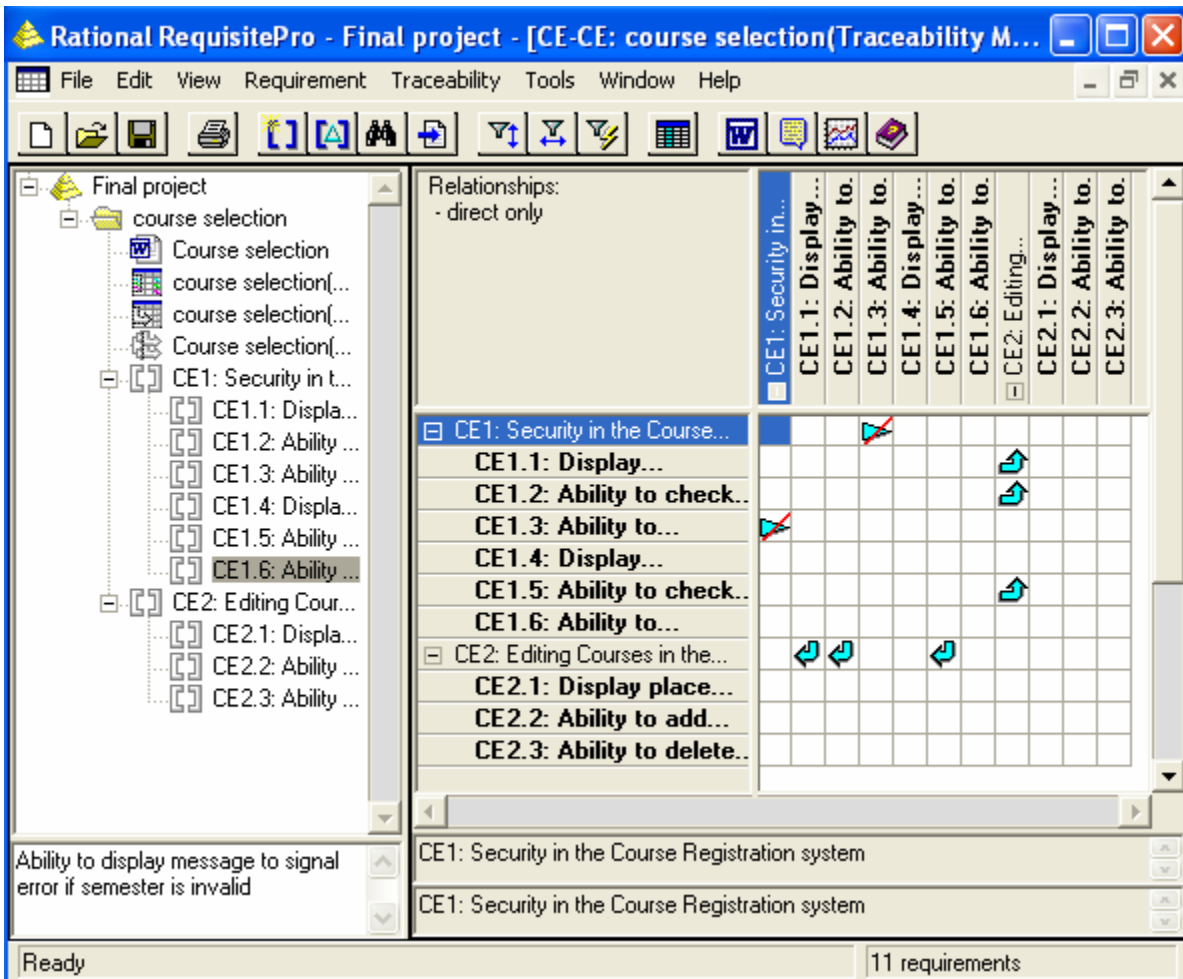


Figure 11.3 Traceability Matrix for JU Registration system

### 11.3.3 The Traceability Tree

The Traceability Tree provides a graphical view of relationships to or from (internal and external) requirements of one specific requirement type, including direct, indirect, and suspect traceability relationships. Although direct and suspect relationships are modified in this view, RequisitePro permits read-only access to indirect relationships.

In addition, the Traceability Tree displays hierarchical relationships and shows parent-child relationships that have been marked as suspect. The tag, name, and attributes of the selected requirement are displayed in the attribute pane.

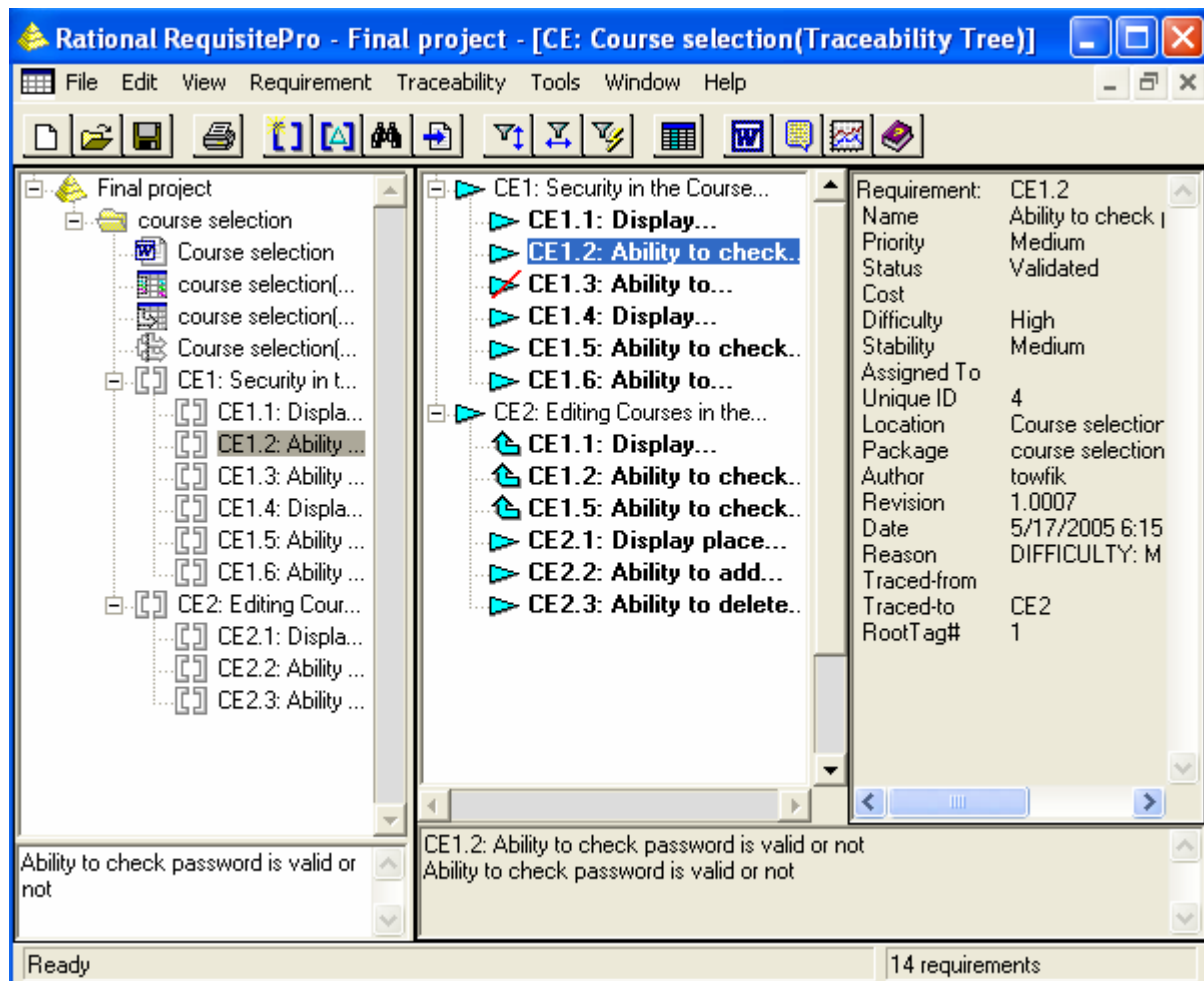


Figure 11.4 Traceability Tree for JU Registration system

### Tree Pane



Traceability relationships for root-level requirements are indented and preceded by an arrow. Child requirements are indented below their parents and preceded by a triangle.

The text pane below the tree provides the name and text of the selected requirement in the tree; the pane on the right side of the view displays requirement attributes of the highlighted requirement.



If the arrow leading from a root requirement points toward a branch requirement, the branch requirement is traced from the root requirement.



If the arrow leading from a branch requirement points toward a root requirement, the branch requirement is traced to the root requirement.





If an arrow with a red diagonal line is displayed, the traceability relationship is suspect.



If a triangle with a red diagonal line is displayed, the hierarchical relationship is suspect.

### Attribute Pane

The attribute pane, located on the right side of the Traceability Tree, displays the tag, name, and attributes associated with the currently selected requirement. To display the attributes of a particular requirement, click the requirement in the tree pane. This is a read-only text box and cannot be edited. If more than one requirement is selected, the pane displays the attributes for the first requirement selected.

### 11.3.4 Creating Views

- Select the package in which the new view appear and click **File: New: View**. The View Properties dialog box opens.
- Do the following:
  - In the Name box, type the name under which the view is to be listed in the Explorer. Filling in the Description box is optional.
  - Select the view and requirement types. Note that user-defined requirement types are also displayed. The Traceability Matrix displays two requirement types (one type for the row and another for the column), the Attribute Matrix displays a single requirement type, and the Traceability Tree displays a single requirement type at the root level. Only requirements associated with the selected requirement types appear in the view.
  - Selecting the Private check box makes the view to be opened by the creator only.
- Click OK. The view opens and appears in the Explorer, in the package selected, listed alphabetically by name.

## 11.4 Querying and Searching

### 11.4.1 Querying overview

Filtering restricts the information being displayed, and sorting determines the order in which the information is displayed. For example, in an Attribute Matrix, ordering requirements information from highest to lowest priority (sort criteria) and view only those requirements assigned (filter criteria). Filter and sort requirements by applying query criteria to the attributes. These criteria limit the values of the attributes and the traceability relationships. It can be possible to create an attribute-by-attribute query, so that the results can be seen in each query criterion, or create a query that filters and sorts several attributes at the same time. Saving a view, save the query criteria. Sort requirements temporarily by attribute value in an Attribute Matrix by right-clicking the attribute label and clicking Sort Ascending or Sort Descending on the shortcut menu.

### 11.4.2 Creating and modifying Queries

Requirements can be filtered in views by limiting the values of one or more attributes or by limiting traceability. It can be possible to query row, column, root requirements, and external requirement types.

- Select a view in the Explorer and click **File: Properties**. The View Properties dialog box opens.
- Click the Query button.  
The Query [Row/Column/Root] Requirements dialog box opens. If any query criteria is not added, the Select Attribute dialog box opens.
- Select the attribute value to query, and click OK.  
The Query Requirements dialog box opens. The dialog box displayed depends on whether the attribute is a list-type or an entry-type attribute.
- For list-type attributes, select one or more attribute values to filter an attribute; the logical operator OR is assumed for list-type attributes. For entry-type attributes, select an operator (that is, includes, equal to) and specify the value.
- Select a sort order (None, Ascending, or Descending). If querying is on the Location attribute, select the Sort by document position check box to sort the requirements in the view in the same order as in the document that contains them.

- Type When Traced criteria, if applicable. The When Traced option is available only when trace to or trace from attributes are queried. For hierarchical requirements, traceability queries automatically include the parent if the child participates in a traceability relationship. Cannot query for suspect hierarchical relationships, and cannot create a traceability query between two external requirements.
- Click OK. The Query [Row/Column/Root] Requirements dialog box opens again.
- To add other query criteria, click Add and repeat the above steps.
- Select the Retain Hierarchical Display check box to include the parents of filtered child requirements in the view, even if the parent requirements do not match the filter criteria specified in the query. And finally click OK.

### 11.4.3 Navigating to a Requirement Using the Go To Command

The Go to command is used to find a requirement location in a project. Using that command in an open document allows jumping quickly to a requirement in the document.

- Select a requirement in a view or in the Explorer.
- Click **Requirement: Go To**.

If the requirement is located in a document, the document containing the requirement opens, with the requirement selected. If the requirement is located in a view, the Requirement Properties dialog box opens, showing the name, text, attribute values, relationships, and discussions for the selected requirement.

### 11.4.4 Reviewing a Project with Cross-Project Traceability

When a project is opened, RequisitePro may connect to an external project and display requirements from both projects. In this case, the projects have been connected to one another, and cross-project traceability relationships may exist. It can be possible to review the requirements in both projects and their change-managed relationships in a view. Cross-project traceability creates traceability relationships between requirements that reside in different projects. This feature establishes connectivity among projects that were divided into subprojects or projects that relate to each other or share a set of requirements, thereby making it easy for team members to reuse common requirements across projects.

When cross-project traceability relationships are established, the following can occur:

- External requirement types can be queried for display in views.
- If a requirement that is traced to requirements in other projects is deleted, RequisitePro updates the other projects to reflect the change. If the traceability relationship in the external project cannot be deleted, the relationship is updated when connectivity is established between the projects.
- If a requirement modification causes the relationship to become suspect, and the requirement is traced to or traced from requirements in other projects, RequisitePro updates the projects to reflect the suspect link. If the traceability relationship in the external project cannot be modified, the relationship is updated when connectivity is established between the projects.
- If a link between an internal and external requirement is marked as suspect, cleared as suspect, or created or removed by the user, RequisitePro updates both the internal and external projects to reflect the change.

#### 11.4.5 Requirement Metrics

Requirement Metrics provides RequisitePro project administrators and product analysts with the capability of reporting statistics on requirement names, text, attributes, relationships, and revisions. These report results are displayed in Microsoft Excel and can be manipulated using Excel's charting capabilities. It begins by creating one or more filters. A filter creates criteria for retrieving requirement information. For example, use an Attribute Count Filter to determine how many requirements in the project have a priority with the value "High." then combine one or more filters to produce a query. A query combines the criteria from multiple filters to analyze requirements. The filters that compose a query are joined with the AND statement. Finally, combine one or more queries to produce a report.

Two types of reports are available in Requirement Metrics:

- A **static** report, which uses static filters and shows results about the project at the present time.
- A **trend analysis** report, which uses time-sensitive filters that analyze changes in requirement text, attributes, traceability, and hierarchical relationships. Trend analysis reports require specifying an increment for displaying revisions.

The Requirement Metrics main window is the primary user interface for the application. This window allows selecting a requirement type for the report, choosing filters and filtering criteria,

building queries, and adding them to the report. As filters are added to queries, limit the requirements in the report to those requirements that match filter criteria.

To access Requirement Metrics, click **Tools: Metrics** or the Metrics button on the toolbar.

## 11.5 Discussions in RequisitePro

Discussions address comments, issues, and questions to a group of participants that are defined. Discussions can be associated with one or more specific requirements, or they can refer to the project in general. A discussion item is either the initial discussion topic or a response. A participant can respond to either the initial discussion text or to another response.

### 11.5.1 Viewing Discussions

All users of a Rational RequisitePro project can read discussion items, whether or not they are discussion participants. Participants in discussion groups can create and reply to discussions. Those who have an e-mail address specified in their user information can receive discussion items by e-mail. When a user opens a project that has unread discussions or responses associated with it, the discussions icon on the toolbar appears highlighted, and the ToolTip on the Show all discussion button informs the user. In views, an icon is placed next to requirements that are associated with discussions.

### 11.5.2 Configuring E-mail for Discussions

RequisitePro offers the capability of communicating with a group of discussion participants regarding comments, issues, and questions related to one or more requirements or to the project in general. When e-mail is enabled, this feature automatically generates an e-mail copy of any discussion item entered (a new discussion or a reply) and sends it to all discussion participants with valid e-mail addresses. Otherwise, users are notified through RequisitePro only.

Discussion e-mail is enabled and configured in several ways:

- Type an e-mail address in the Project Security dialog box (click **File: Project Administration: Security**). This entry supplies the user information for each user in RequisitePro.
- Configure a notification e-mail service (click **Tools: E-mail Setup**).

- Configure participation and notification mail for all users with e-mail addresses using the Rational E-mail Reader application.

All RequisitePro users can add or edit their own e-mail addresses within RequisitePro; however, only project administrators (who have administrator permissions) can add or edit other user's information.

### 11.5.3 Creating Discussions

To create a discussion at any time:

- Do one of the following:
  - Click **Tools: Discussions** (or the Show all discussions button).
  - In the Explorer or in a view, select one or more requirements and click **Requirement: Discussion**.
  - Select a requirement in a document and click **RequisitePro: Requirement: Discussions**. The Discussions dialog box opens.
- Click Create. The Discussion Properties dialog box opens.
- Click the **General** tab, and type a subject (required) and text. The subject should be brief, but it should also be descriptive enough to inform other users of the discussion's contents. The text typed in the Text box becomes the first item in the discussion. Open the discussion by raising an issue, making a comment, or asking a question.
- Click the Participants tab. Add users and groups.
- Click the Requirements tab and add requirements.
- Click OK to close the Select Requirements dialog box, and click OK to close the Discussion Properties dialog box.
- Click Close to close the Discussions dialog box.

If e-mail is configured for discussions, the discussion is opened and the message is sent to all participants who have an e-mail address specified in their user information. An icon on the toolbar also notifies users that there is a new discussion associated with that project and its requirements. The highlighted icon appears for all users who open the project (whether or not they are participants of the discussion).

As with e-mail, discussion messages cannot be modified after they have been sent. This restriction prevents conflicts when the original text is included in a discussion reply. To add an explanation regarding to initial message, do so by creating a reply to the discussion.

#### 11.5.4 Reading Discussions

The Discussions dialog box helps to keep track of which discussion items have been read and which have not been. Unread discussion items are shown as bold text. Read discussion items are shown as regular text.

- Click **Tools: Discussions** (or the Show all discussions button). The Discussions dialog box opens.
- In the discussions list, click a discussion.  
An expand indicator is displayed to the left of discussions with responses.
- Click an item to read it. The item's text is displayed in the lower portion of the dialog box.
- Click Close to close the Discussions dialog box.

#### 11.5.5 Responding to Discussions

Discussion can be responded either in RequisitePro or in e-mail application. For those who are not discussion participant, they can respond to the discussion only if the Restrict To Participants check box is cleared in the Discussion Properties dialog box at the Participants tab. Only the discussion author and members of the Administrators group can modify this option.

To respond to a discussion in RequisitePro using the Discussions dialog box:

- Click **Tools: Discussions** (or the Show all discussions button). The Discussions dialog box opens.
- In the discussions list, click a discussion or response to a discussion. The discussion text appears in the lower portion of the Discussions dialog box.
- Click Reply. The Discussion Response dialog box opens.
- Type the response.

The text automatically wraps at the end of each line. Use CTRL-C to copy selected text from another application or document and CTRL-V to paste it into the response text box.

Text formatting, such as bold and underline, is not available in the Discussion Response dialog box. Attaching files to the response is not possible.

- Click OK. The Discussions dialog box opens again. RequisitePro updates the associated project and discussion and then sends the response to all participants who have an e-mail address specified in their user information.

It can be possible to open the Discussions dialog box to display only the discussions associated with a specific requirement, or open the Discussions dialog box to display all open discussions. Specify which discussions to view in the Discussions dialog box, based on text, requirement, users, priority, and status. In the Discussions dialog box, the sort order in which discussions appear can be changed. Discussion items can be printed. The printout shows the discussion item's subject, text, author, and creation date and time. The printout also shows information about the associated discussion, including the discussion's priority, status, requirements, and participants.

The Attributes tab in the Discussion Properties dialog box shows the discussion's author, the date and time the discussion was created, and its current priority and status. These properties can be modified only by the discussion author and project administrators.

The Discussion Properties dialog box, Participants tab determines which users and groups are included in the discussion and whether the discussion is restricted to participants. Any user can view the Participants tab, but only the discussion author and project administrators can modify its options, with the following two exceptions:

- Any user can add herself/himself to the Users list if the Restrict To Participants check box is cleared.
- Any user can always remove herself/himself from the Users list.

It can be possible to use the Discussion Properties dialog box, Requirements tab to associate specific requirements with a discussion. Note that a discussion does not need to address any requirement specifically. However, if it is wanted to address a specific requirement, use this tab to associate it with the discussion.

## **11.6 Documents in RequisitePro**

A Rational RequisitePro requirements document is a Microsoft Word file created in RequisitePro and integrated with the project database. Requirements created outside of a project document can be imported or copied into the document. If a new project document is created, RequisitePro associates



the document with the project that is open. This association is used to update the database and synchronize the revision numbering of the project and document. The document's name, location, document type, and revision information are stored in the project database. The document requirements, their attribute values, and their traceability relationships are also stored in the project database.

### 11.6.1 Creating RequisitePro Document

To create a document in a RequisitePro project:

- Open the project with which to associate the new document, and in the Explorer select the package in which to store the new document.
- Do one of the following:
  - Click **File: New: Document**.
  - In Word, click **RequisitePro: Document: New**. The Document Properties dialog box opens.
- In the General tab, type the name of the document in the Name box (64 characters maximum) and a brief description (up to 255 characters) of the document's purpose or contents in the Description box.
- In the Filename box, type, modify, or view the name of the document. The file name defaults to the name typed in the Name box. The file name is limited to eight characters unless long file names are supported by operating system.
- Click the Show Tags check box to display requirement tags in the new document. Clear this box to hide the tag text. Selecting this check box sets options in Microsoft Word to ensure that the requirement tags are visible both on the display and in the printed document.
- Type a location for the file in the Directory box, or click Browse to select a directory.
- Select a document type to assign to the new document.
- Click OK. The new document is open and ready to be edited.

When a document is saved, RequisitePro saves to the project database all document modifications, such as the document's name, revision number, label, and change description. RequisitePro automatically creates backup files for each document file in the document directory and updates them when the document is saved. In addition, RequisitePro commits all new and modified requirements in

the document to the database, and it deletes from the database all requirements that are deleted from the document. Requirements with pending tag numbers are assigned requirement tags. Save the active requirements document by clicking RequisitePro: Document: Save.

Remove command is used to remove requirements formatting in the document and remove the requirements and the document itself from the project database. To use this command, exclusive access to the project is required. This command does not delete the document or any text that are highlighted as requirements. RequisitePro saves the document as a Microsoft Word document in the project directory.

New documents are stored in the project directory by default. Documents can be created in or moved to any directory on the file system. All documents associated with the project are included in the project list regardless of where they are stored.

### **11.6.2 Microsoft Word**

RequisitePro includes an option to save a project document as a Microsoft Word document that is independent of the project. This allows attaching the document to e-mail or share the document with someone outside of working group. When this option is chosen, RequisitePro makes a copy of the active requirements document and saves it as a Word file. The resulting Word document retains the look and feel of the original requirements document and even includes bookmarks and tags used by RequisitePro to identify requirements.

While working in RequisitePro, create, open, modify, save, and close Word documents that are independent from RequisitePro. Use the standard File menu commands to manage these Word documents. (The Save As and Exit commands are unavailable in order to prevent conflicts in handling RequisitePro documents.)

The presentation style of one or more aspects of the document can be changed. RequisitePro uses formatting features in Word to highlight requirements within a document. If different formatting to a requirement is applied, either by directly formatting the requirement or by changing the template style on which it is based, requirement text may be modified or entirely lost. Correct this problem by using the Refresh Requirements command.

## 11.7 Requirements

A requirement describes a condition or capability that a system must provide; it is either derived directly from user needs or is stated in a contract, standard, specification, or other formally imposed document. Examples of requirements include inputs to the system, outputs from the system, and functions and attributes of the system and the system environment.

In Rational RequisitePro, requirements contain a name and text, and they can be qualified with attributes. Attributes describe a requirement in terms of user defined characteristics or properties, such as cost, priority, and status.

Requirements are created in a view or in a requirements document. All requirements created in RequisitePro are stored in the project database. After a requirement is created, do the following:

- Move or copy the requirement to a document or a view.
- Qualify the requirement by assigning attributes.
- Trace the requirement to and from other requirements.

Use hierarchical relationships to subdivide a general requirement into more explicit requirements. Child requirements provide additional detail about their parent requirement.

A requirement's name, text, attributes, and relationships can be modified in a view or a document. It can be possible to read information in context and add information that supports and justifies the requirements.

### 11.7.1 Creating Requirements

All requirements that are created in RequisitePro are stored in the project's database. After a requirement has been created, it can be modified, moved, and copied within the project and traced to and from other requirements in the same project or across projects.

When a requirement is created, RequisitePro stores the following information in the database:

- *The requirement name.* A user defined title for a requirement; it must not exceed 128 characters. Like tag and text attributes, the name can be used to reference requirements. The name is displayed by default in all RequisitePro views. All requirements must have either a name or text (or both), but requirements located in documents must have text. A name is not required to be unique, and change it at any time if the requirement is created or belong to a group that has update permissions.

- *The requirement text.* Requirement text is the full textual content of a requirement; it must not exceed 16,000 characters. If the requirement is located in a document, the requirement text may include embedded and linked objects, such as graphics, tables, and Microsoft Word files.
- *Attribute values.*

To create requirement:

- In the document, select the information (text, graphics, and OLE objects) that are become part of the requirement. (If text is not selected, prompted to enter text for the requirement.)
- Click **RequisitePro: Requirement: New** or the **Create Requirement** button. The Requirement Properties dialog box opens.
- Click the General tab.
  - In the Type box, select a requirement type with which to associate the requirement.
  - In the Name box, type the requirement name (up to 128 characters).
  - Click the Browse button next to the Package box to select a different package in which to place the new requirement.
- Click OK.
- To save the changes, click **RequisitePro: Document: Save**.

### 11.7.2 Inserting Microsoft Word-linked Files in Requirements

Microsoft Word-linked files can be included as part of requirement text. If there is a change in the linked file, any traceability or hierarchical relationships, the requirements are marked as suspect. Microsoft PowerPoint files, Word documents, Excel spreadsheets, bitmap files, and other types of files can be linked.

Microsoft Word's Insert commands are used to link files to RequisitePro requirement text; Word's Paste Special and Field commands (available through the Edit and Insert menus respectively) are also used. If Insert: Field is clicked, the following types of Microsoft Word Links and References are supported:

- Link
- Include Text
- Include Picture

### **11.7.3 Creating requirements in View**

Requirements are created in any view. In a Traceability Matrix or a Traceability Tree, enter data in the Requirement Properties dialog box. In an Attribute Matrix, insert a new requirement directly into the matrix and set requirement name, text, and attributes before saving it.

## **11.8 Hierarchy**

Hierarchical requirement relationships are parent-child relationships that reflect direct dependencies between requirements of the same type. In addition, these associations provide powerful tools for change management. With hierarchical requirements, a requirement with a number of sub-requirements is supported by establishing a parent requirement and creating child requirements of that parent. Like traceability relationships, hierarchical relationships are a type of change managed relationship in Rational RequisitePro. If a requirement's name, text, requirement type, or attribute is changed, the relationships with its children become suspect. Suspect relationships can be viewed and managed using a Traceability Matrix or a Traceability Tree view.

### **11.8.1 Child Requirements**

A child requirement is any requirement that has a parent. Each child requirement can have only one parent, but a requirement can be both a parent and a child. Hierarchical relationships are created in a requirements document or in a view. If the parent requirement resides in a document, the child requirement must reside in the same document. The parent requirement and all of its children must be of the same requirement type. These relationships are created in one of the following ways: use the Hierarchy tab in the Requirement Properties dialog box; create them directly in a requirements document by clicking RequisitePro: Requirement: New; or click Requirement: New to insert them in an Attribute Matrix.

### **11.8.2 Peer Requirements**

Requirements have a peer relationship when they are at the same level in the requirement hierarchy. For example, two requirements are peer requirements when they are children of the same parent. All requirements at the root level are peer requirements of one another. When a requirement is created in a document, it is automatically placed at the same level as the requirement above it.

### 11.8.3 Suspect Relationship

A relationship between requirements becomes questionable or suspect if RequisitePro detects a requirement's name, text, requirement type, or attributes have been modified. In a view, manually hierarchical relationships are marked as suspect or cleared from suspect relationship.

- If a parent requirement is modified, RequisitePro marks the relationship between the parent and all its immediate children as suspect.
- If a child requirement is modified, RequisitePro does not mark the relationship between the child and its parent as suspect, although the child's relationship with other requirements is marked as suspect.
- When children are reassigned to another parent, all hierarchical relationships between the new parent and its immediate children are automatically marked as suspect. The relationships between the children and their children are not marked as suspect.

Suspect hierarchical relationships are displayed using the Traceability Matrix and the Traceability Tree views. In a view, the suspect relationship is identified by a red diagonal line through the requirement icon. The Auto Suspect command monitors a requirement's change history and displays a suspect signal when requirements are changed. Click Tools: Auto Suspect to enable or disable automatic checking for changes that affect the traceability or hierarchical relationships between requirements in the project.

Parent-child relationship that has been marked suspect can be reset by manually clearing it in the Traceability Matrix or Traceability Tree. In a Traceability Matrix, select one or more intersection points in the matrix where a traceability link has been created. Then either click Traceability: Clear Suspect, or click Edit: Set Value, and select Clear Suspect from the list. In the Traceability Tree, select one or more requirements to modify. Then click Traceability: Clear Suspect.

## 11.9 Traceability

In Rational RequisitePro, traceability is a dependency relationship between two requirements. Traceability is a methodical approach to managing change by linking requirements that are related to each other.

Like hierarchical relationships, traceability relationships are change managed relationships in RequisitePro. If either end-point of the connection is changed, the relationship becomes suspect. RequisitePro makes it easy to track changes to a requirement throughout the development cycle, so it

is not necessary to review all documents individually to determine which elements need updating. Suspect relationships can be viewed and managed using a Traceability Matrix or a Traceability Tree view.

### 11.9.1 Traceability in View

Traceability relationships can be created and deleted in an Attribute Matrix, a Traceability Matrix, and a Traceability Tree.

To create a traceability relationship:

- Open an Attribute Matrix and select a requirement. Then click **Requirement: Properties**. The Requirement Properties dialog box opens.
- Click the Traceability tab.
- Click the Add button adjacent to the To or From box. The Trace To Requirement or the Trace From Requirement(s) dialog box opens.
- From the Requirements of type list, select the requirement type with which the requirement is associated.
- Select the location (document or database) of the requirement from the Located in list. If the location is not known, select **All locations**.
- Select a requirement to *trace to* or *trace from* in the list and click OK. RequisitePro adds this requirement to the To or From box on the Traceability tab.
- Click OK.

### 11.9.2 Suspect Relationship

A relationship between requirements becomes questionable or suspect if RequisitePro detects that a requirement's name, text, requirement type, or attribute has been modified.

The suspect state is reflected in the Requirement Properties dialog box as well as in views. In the Traceability Tree and Traceability Matrix views, suspect traceability relationships are marked as lines through the arrows. In an Attribute Matrix, an **(s)** appears after the requirement tag in the Traced-to or Traced-from column. On the Traceability tab of the Requirement Properties dialog box, an **(s)** displayed after the requirement tag in the To and From boxes represents a suspect relationship. In a view, the suspect relationship is identified by a red diagonal line through the traceability arrow.

## 11.10 Importing Requirements and Documents

Requirements are imported from a Microsoft Word document file into a Rational RequisitePro project use the import wizard.

The import process is recommended for:

- Importing nonproject documents into the active project.
- Importing new requirements into an existing project.
- Updating existing requirement attributes with new information.

The Import Wizard can import requirements and attributes into project from several sources:

- Requirement documents created in other RequisitePro projects.
- Microsoft Word documents created outside of RequisitePro.
- Any database that supports export of data in CSV format, such as Microsoft SQL Server, Oracle, Microsoft Excel, or Microsoft Access.

To use the Import Wizard to import Word documents:

- Open the project into which Word document is imported.
- Select the project in the Explorer and click **File: Import**. The Import Wizard opens, and the Select a Source screen opens.
- Select the Microsoft Word Document option.
- Type the path and name of the Word document to be imported, or click Browse to select a file.
- Click Next to continue. The Select Import Content screen opens.
- Designate the import content by selecting the appropriate option (Requirements and document, Requirements only, or Document only).

### 11.10.1 Preparing to Import

An external document or a CSV file can be imported into a RequisitePro project and placed in the package of choice. If the project administrator has not already set up a project, it should be create before proceeding with the import.

RequisitePro creates a log file each time a file is imported. The log file is a written record of the import process. The log file records these operations: requirement types matching, conflict resolutions, attribute mappings, the success or failure of search criteria, canceled operations, value errors, and the success or failure of the import. The log file is saved in the directory where the project



is stored, and it is named import#.log, where # is a number representing sequential order of creation. If a log file is deleted, the next one created receives the old number to fill in the gap.

If all the documents are wanted to import from one project into another, they can all be converted to Microsoft Word format at once.

To convert all documents to Word at once:

- Open the project.
- In the Explorer, select the project and click **File: Properties**. The Project Properties dialog box opens.
- On the Documents tab, clear the Save documents in RequisitePro Format check box.
- Click OK. When the project is closed, all documents are converted to Word format but remain in the RequisitePro system. The documents retain their RequisitePro extensions.

Requirements are exported from RequisitePro in views.

- Attribute Matrixes and Traceability Trees can be exported as CSV files or as Microsoft Word documents.
- Traceability Matrixes can be exported as CSV files only.

Setting view properties for requirement text and name affects what is exported. If the requirement text and name are not displayed in the view, they are not exported. If a view is queried, only the filtered requirements are exported.

## 12. Conclusions

Rational Rose is the visual modeling tool that is part of a comprehensive set of tools embodies software practices and spans the entire software development life cycle. We have studied that Rational Rose helps improve communication both within teams and across team boundaries, reduce development time and improving software quality. Rational Rose provides a way to describe a system being developed thoroughly using different diagrams. It also helps to figure out system requirements. So that we have concluded that any project done with rational rose is delivered on time and quality.

The add-in feature allows to quickly and accurately customize the Rational Rose environment depending on the development needs. Using the add-in tool, language (such as Visual Basic, Visual Java) tools can be installed in Rational Rose.

We have seen that notation plays an important part in any application development activity; it is glue that holds the process together. UML is a language used to specify, visualize, and document the artifacts of an object-oriented system under development. It provides a very robust notation, which grows from analysis into design.

RequisitePro is a requirement management tool that integrates a powerful multi-user requirements database utility with the familiar environment of Microsoft word for windows. We have studied that the program allows working simultaneously with a requirement database and requirements documents. Requirement management is the most significant factor in delivering projects on time, on budget, and on target. RequisitePro helps project succeed by giving teams the ability to manage all project requirements comprehensively and facilitating team collaboration and communication. It combines both document centric and database-centric approaches. By deeply integrating Microsoft word with multiuser database, RequisitePro organize, prioritize, trace relationships, and easily track changes to requirements. The integration of RequisitePro and Rational Rose improve system development process.

## REFERENCES

1. Rational the e-development company, "RATIONAL ROSE MANUAL", Version:2001A.04.00
2. Rational Software corporation, "RATIONAL REQUISITEPRO USER'S GUIDE", Version: 2003.06.00
3. Terry Quatrani, "VISUAL MODELING WITH RATIONAL ROSE AND UML".
4. <http://www.rational.com>
5. Evangelos Petroutsos, "MASTERING VISUAL BASIC 6"
6. Roger S. Pressman, "SOFTWARE ENGINEERING", Fifth edition.
7. K.K. Aggarwal & Yogesh Singh, "SOFTWARE ENGINEERING (programs, documentation and operating procedures)".