Chapter 3

Creating Use Cases

- System Behavior
- Actors
- Use Cases
- Use Case Relationships
- Use Case Diagrams
- Summary

SYSTEM BEHAVIOR

THE BEHAVIOR OF the system under development, that is what functionality must be provided by the system, is documented in a use case model that illustrates the system's intended functions (use cases), its surroundings (actors), and relationships between the use cases and actors (use case diagrams). The most important role of a use case model is one of communication. It provides a vehicle used by the customers or end users and the developers to discuss the system's functionality and behavior.

The use case model starts in the Inception Phase with the identification of actors and principal use cases for the system. The model is then matured in the Elaboration Phase—more detailed information is added to the identified use cases, and additional use cases are added on an as-needed basis.

ACTORS

ACTORS ARE NOT part of the system—they represent anyone or anything that must interact with the system. An actor may

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

Typically, these actors are found in the problem statement and by conversations with customers and domain experts. The following questions may be used to help identify the actors for a system:

- Who is interested in a certain requirement?
- Where in the organization is the system used?
- Who will benefit from the use of the system?
- Who will supply the system with this information, use this information, and remove this information?
- Who will support and maintain the system?

- Does the system use an external resource?
- Does one person play several different roles?
- Do several people play the same role?
- Does the system interact with a legacy system?

In the UML, an actor is represented as a stickman, as shown in Figure 3-1.

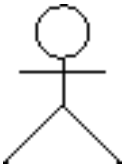

Figure 3-1 UML Notation for an Actor

What Constitutes a Good Actor?

Care must be taken when identifying the actors for a system. This identification is done in an iterative fashion—the first cut at the list of actors for a system is rarely the final list. For example, is a new student a different actor than a returning student? Suppose you initially say the answer to this question is yes. The next step is to identify how the actor interacts with the system. If the new student uses the system differently than the returning student, they are different actors. If they use the system the same way, they are the same actor.

Another example is the creation of an actor for every role a person may play. This may also be overkill. A good example is a teaching assistant in the ESU Course Registration System. The teaching assistant takes classes and teaches classes. The capabilities needed to select courses to take and to select courses to teach is already captured by the identification of functionality needed by the Student and the Professor actors. Therefore, there is no need for a Teaching Assistant actor.

By looking at the identified actors and documenting how they use the system, you will iteratively arrive at a good set of actors for the system.

Actors in the ESU Course Registration System

The previous questions were answered as follows:

- 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 Registrar must maintain all the information about courses, professors, and students
- The Billing System must receive billing information from the system

Based on the answers to the questions posed, the following actors have been identified: Student, Professor, Registrar, and the Billing System.

CREATING ACTORS IN RATIONAL ROSE

1. Right-click on the Use Case View package in the browser to make the shortcut menu visible.
2. Select the New:Actor menu option. A new actor called New Class is placed in the browser.
3. With the actor called New Class selected, enter the desired name of the actor.

The Browser view of the actors for the ESU Course Registration System is shown in Figure 3-2.

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 descriptions for the ESU Course Registration System are:

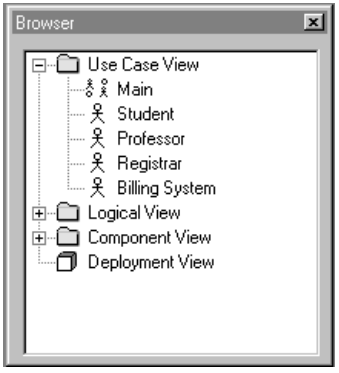

Figure 3-2 Actors

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

DOCUMENTING ACTORS IN RATIONAL ROSE

1. If the documentation window is not visible, open the documentation window by selecting the Documentation menu choice from the View menu.
2. Click to select the Actor in the browser.
3. Position the cursor in the documentation window and enter the documentation.

The documentation for the Student actor is shown in Figure 3-3.

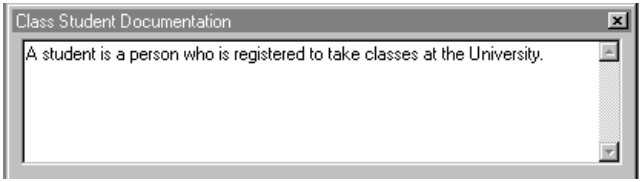

Figure 3-3 Student Actor Documentation

USE CASES

USE CASES MODEL a dialogue between an actor and the system. They represent the functionality provided by the system; that is, what capabilities will be provided to an actor by the system. The collection of use cases for a system constitute all the defined ways the system may be used. The formal definition for a use case is: A use case is a sequence of transactions performed by a system that yields a measurable result of values for a particular actor.

The following questions may be used to help identify the use cases for a system:

- What are the tasks of each actor?
- Will any actor create, store, change, remove, or read information in the system?
- What use cases will create, store, change, remove, or read this information?
- Will any actor need to inform the system about sudden, external changes?
- Does any actor need to be informed about certain occurrences in the system?
- What use cases will support and maintain the system?
- Can all functional requirements be performed by the use cases?

In the UML, a use case is represented as an oval, as shown in Figure 3-4.

Figure 3-4 UML Notation for a Use Case

What Constitutes a Good Use Case?

Over the years there has been a lot of discussion dealing with the "goodness" of a use case. One problem that I have encountered is the level of detail found in use cases. That is, how big (or how little) should they be? There is no one, right answer. The rule of thumb that I apply is:

> A use case typically represents a major piece of functionality that is complete from beginning to end. A use case must deliver something of value to an actor.

For example, in the ESU Course Registration System, the student must select the courses for a semester, the student must be added to the course offerings, and the student must be billed. Is this three use cases, or just one? I would make it one since the functionality represents what happens from beginning to end. What good would the system be if a student was not added to the courses selected (or at least notified if the addition does not occur)? Or if the student was not billed (the University would not stay in business if all courses were free!)?

Another problem is how to bundle functionality that is different but seems to belong together. For example, the Registrar must add courses, delete courses, and modify courses. Three use cases, or one use case? Here again, I would make one use case—the maintenance of the curriculum, since the functionality is started by the same actor (the Registrar) and deals with the same entities in the system (the curriculum).

Use Cases in the ESU Course Registration System

The following needs must be addressed by the system:

- The Student actor needs to use the system to register for courses
- After the course selection process is completed, the Billing System must be supplied with billing information
- The Professor actor needs to use the system to select the courses to teach for a semester, and must be able to receive a course roster from the system
- 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

CREATING USE CASES IN RATIONAL ROSE

1. Right-click on the Use Case View in the browser to make the shortcut menu visible.
2. Select the New:Use Case menu option. A new unnamed use case is placed in the browser.
3. With the use case selected, enter the desired name of the use case.

The browser view of the use cases for the ESU Course Registration System is shown in Figure 3-5.

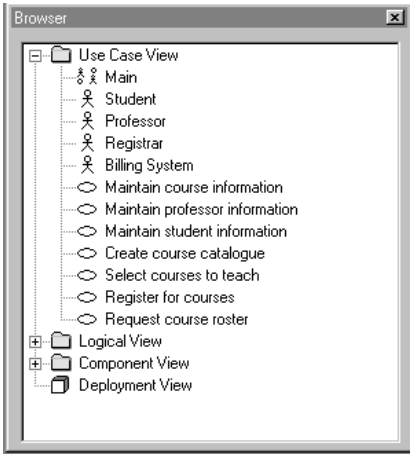

Figure 3-5 Use Cases

Brief Description of a Use Case

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. This description is typically created during the Inception Phase as the use case is identified.

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, delete, modify, and/or review a student schedule for a specified semester.

CREATING A USE CASE BRIEF DESCRIPTION IN RATIONAL ROSE

1. Click to select the use case in the browser.
2. Position the cursor in the documentation window and enter the brief description for the use case. If the documentation window is not visible, select the View:Documentation menu choice to make the window visible.

The brief description of the *Register for Courses* use case is shown in Figure 3-6.

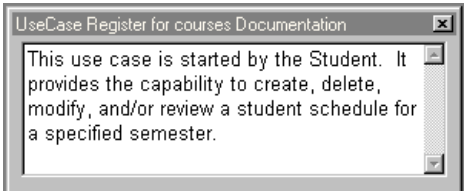

Figure 3-6 Use Case Brief Description

The Flow of Events for a Use Case

Each use case also is documented with a flow of events. The flow of events for a use case is a description of the events needed to accomplish the required behavior of the use case. The flow of events is written in terms of what the system should do, not how the system does it. That is, it is written in the language of the domain, not in terms of implementation. The flow of events should include:

- When and how the use case starts and ends
- What interaction the use case has with the actors
- What data is needed by the use case
- The normal sequence of events for the use case
- The description of any alternate or exceptional flows

The flow of events documentation typically is created in the Elaboration Phase in an iterative manner. At first, only a brief description of the steps needed to carry out the normal flow of the use case (i.e., what functionality is provided by the use case) is written. As analysis progresses, the steps are fleshed out to add more detail. Finally, the exceptional flows are added to the use case (the what happens if . . . part of the flow of events).

Each project should use a standard template for the creation of the flow of events document. I have found the following template to be useful.

X Flow of Events for the <name> Use Case

X.1 Preconditions

X.2 Main Flow

X.3 Subflows (if applicable)

X.4 Alternative Flows

where X is a number from 1 to the number of use cases.

A sample completed Flow of Events document for the *Select Courses to Teach* use case follows.

1.0 Flow of Events for the* Select Courses to Teach *Use Case

1.1 Preconditions

The Create Course Offerings *subflow of the* Maintain Courses *use case must execute before this use case begins.*

1.2 Main Flow

This use case begins when the professor logs onto the Registration System and enters his/her password. The system verifies that the password is valid (E-1) and prompts the professor to select the current semester or a future semester (E-2). The professor enters the desired semester. The system prompts the professor to select the desired activity: ADD, DELETE, REVIEW, PRINT, or QUIT.

If the activity selected is ADD, the S-1: Add a Course Offering *subflow is performed.*

If the activity selected is DELETE, the S-2: Delete a Course Offering *subflow is performed.*

If the activity selected is REVIEW, the S-3: Review Schedule *subflow is performed.*

If the activity selected is PRINT, the S-4: Print a Schedule *subflow is performed.*

If the activity selected is QUIT, the use case ends.

1.3 Subflows

S-1: Add a Course Offering
The system displays the course screen containing a field for a course name and number. The professor enters the name and number of a course (E-3). The system displays the course offerings for the entered course (E-4). The professor selects a course offering. The system links the professor to the selected course offering (E-5). The use case then begins again.

S-2: Delete a Course Offering
The system displays the course offering screen containing a field for a course offering name and number. The professor enters the name and number of a course offering (E-6). The system removes the link to the professor (E-7). The use case then begins again.

S-3: Review a Schedule
The system retrieves (E-8) and displays the following information for all course offerings for which the professor is assigned: course name, course number, course offering number, days of the week, time, and location. When the professor indicates that he/she is through reviewing, the use case begins again.

S-4: Print a Schedule
The system prints the professor schedule (E-9). The use case begins again.

1.4 Alternative Flows

E-1: An invalid professor ID number is entered. The user can re-enter a professor id number or terminate the use case.

E-2: An invalid semester is entered. The user can re-enter the semester or terminate the use case.

E-3: An invalid course name/number is entered. The user can re-enter a valid name/number combination or terminate the use case.

E-4: Course offerings cannot be displayed. The user is informed that this option is not available at the current time. The use case begins again.

E-5: A link between the professor and the course offering cannot be created. The information is saved and the system will create the link at a later time. The use case continues.

E-6: An invalid course offering name/number is entered. The user can re-enter a valid course offering name/number combination or terminate the use case.

E-7: A link between the professor and the course offering cannot be removed. The information is saved and the system will remove the link at a later time. The use case continues.

E-8: The system cannot retrieve schedule information. The use case then begins again.

E-9: The schedule cannot be printed. The user is informed that this option is not available at the current time. The use case begins again.

Use case flow of events documents are entered and maintained in documents external to Rational Rose. The documents are linked to the use case.

LINKING FLOW OF EVENTS DOCUMENTS TO USE CASES IN RATIONAL ROSE

1. Right-click on the use case in the browser to make the shortcut menu visible.
2. Select the Specification menu option.
3. Select the Files tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert File menu option.
6. Browse to the appropriate directory and select the desired file.
7. Click the Open button.
8. Click the OK button to close the specification.

A linked use case flow of events document is shown in Figure 3-7.

USE CASE RELATIONSHIPS

AN ASSOCIATION RELATIONSHIP may exist between an actor and a use case. This type of association is often referred to as a *communicates association* since it represents communication between an actor and

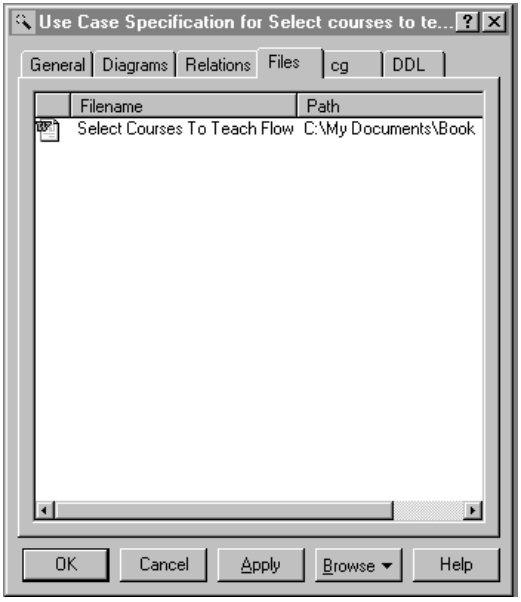

Figure 3-7 Linked Flow of Events Document

a use case. An association may be navigable in both directions (actor to use case and use case to actor), or it may be navigable in only one direction (actor to use case or use case to actor). The navigation direction of an association represents who is initiating the communication (i.e., the actor is initiating the communication with the use case, the use case is initiating the communication with the actor). An association is represented as a line connecting the related elements. Navigation in only one direction is depicted by adding an arrowhead to the association line that denotes the direction.

There are two types of relationships that may exist between use cases: uses and extends. Multiple use cases 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. Uses relationships are created between the new use case and any other use case that uses its functionality. For example, each use case in the ESU Course Registration System starts with the verification of the user. This functionality can be captured in a User Verification use case, which is then used by other use cases as needed. A uses

relationship is drawn as an arrow with a large hollow triangle (generalization arrow) at the end closest to the used use case.

An extends relationship is used to show:

- Optional behavior
- Behavior that is only run under certain conditions, such as triggering an alarm
- Several different flows which may be run based on actor selection

For example, a use case that monitors the flow of packages on a conveyer belt can be extended by a Trigger Alarm use case if the packages jam. At this time, no extensions have been identified for the ESU Course Registration System. An extends relationship is drawn as an arrow with a large hollow triangle (generalization arrow) at the end closest to the base use case.

The UML has a concept called a *stereotype,* which provides the capability of extending the basic modeling elements to create new elements. Thus, the concept of a stereotype allows the UML to have a minimal set of symbols that may be extended where needed to provide the communication artifacts that have meaning for the system under development. Stereotype names are included within guillemets (< < > >) and placed along the relationship line. Stereotypes are used to create the needed use case relationships. The stereotype < < communicates > > may be added to an association to show that the association is a communicates association. This is optional since an association is the only type of relationship allowed between an actor and a use case. Uses and extends relationships must use stereotypes since they are both represented by a generalization arrow.

Use case relationships are shown in Figure 3-8.

USE CASE DIAGRAMS

A USE CASE diagram is a graphical view of some or all of the actors, use cases, and their interactions identified for a system. Each system typically has a Main Use Case diagram, which is a picture of the

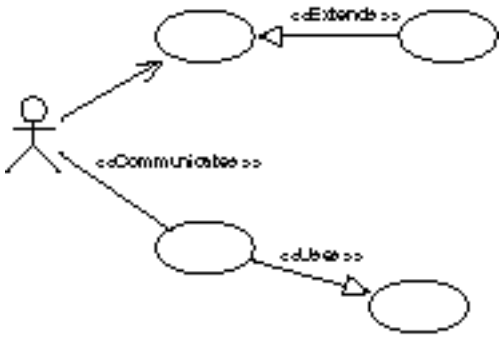

Figure 3-8 Use Case Relationships

system boundary (actors) and the major functionality provided by the system (use cases). Other use case diagrams may be created as needed. Some examples are:

- A diagram showing all the use cases for a selected actor
- A diagram showing all the use cases being implemented in an iteration
- A diagram showing a use case and all its relationships

CREATING THE MAIN USE CASE DIAGRAM IN RATIONAL ROSE

1. Double-click on the Main diagram in the Use Case View in the browser to open the diagram.
2. Click to select an actor in the browser and drag the actor onto the diagram.
3. Repeat step 2 for each additional actor needed in the diagram.
4. Click to select a use case in the browser and drag the use case onto the diagram.
5. Repeat step 4 for each additional use case needed in the diagram.

Note: Actors and use cases may also be created directly on a use case diagram by using the toolbar.

CREATING COMMUNICATES ASSOCIATIONS IN RATIONAL ROSE

1. Click to select the Association icon or the Unidirectional Association icon from the diagram toolbar. Note: If the Association icon is not present on the toolbar, it may be added by right-clicking on the toolbar, selecting the Customize menu choice from the shortcut menu, and adding the icon to the toolbar.
2. Click on an actor initiating a communication and drag the association line to the desired use case.

To add the Communicates stereotype (optional):

1. Double-click on the association line to make the Specification visible.
2. If this is the first time the stereotype is being used, enter Communicates in the Stereotype field. If a Communicates stereotype has already been created, click the arrow in the Stereotype field to make the drop-down menu visible and select Communicates.
3. Click the OK button to close the Specification.
4. Right-click on the association line to make the shortcut menu visible.
5. Select the Show Stereotype menu choice.
6. Repeat the preceding steps for each additional association relationship.

CREATING USES RELATIONSHIPS IN RATIONAL ROSE

1. Click to select the Generalization icon from the toolbar.
2. Click on the using use case and drag the Generalization icon to the used use case.
3. Double-click on the generalization arrow to make the Specification visible.
4. If this is the first time a uses relationship is being created, enter Uses in the Stereotype field. If a Uses stereotype has already been created, click the arrow in the Stereotype field to make the drop-down menu visible and select Uses.

5. Click the OK button to close the Specification.
6. Right-click on the generalization arrow to make the shortcut menu visible.
7. Select the Show Stereotype menu choice.

CREATING EXTENDS RELATIONSHIPS IN RATIONAL ROSE

1. Click to select the Generalization icon from the toolbar.
2. Click on the use case containing the extended functionality and drag the Generalization icon to the base use case.
3. Double-click on the generalization arrow to make the Specification visible.
4. If this is the first time an extends relationship is being created, enter Extends in the Stereotype field. If an Extends stereotype has already been created, click the arrow in the Stereotype field to make the drop-down menu visible and select Extends.
5. Click the OK button to close the Specification.
6. Right-click on the generalization arrow to make the shortcut menu visible.
7. Select the Show Stereotype menu choice.

The Main use case diagram for the ESU Course Registration System is shown in Figure 3-9.

CREATING ADDITIONAL USE CASE DIAGRAMS IN RATIONAL ROSE

1. Right-click on the Use Case View in the browser to make the shortcut menu visible.
2. Select the New:Use Case Diagram menu option.
3. While the use case diagram is selected, enter the name of the use case diagram.
4. Open the diagram and add actors, use cases, and interactions to the diagram as needed.

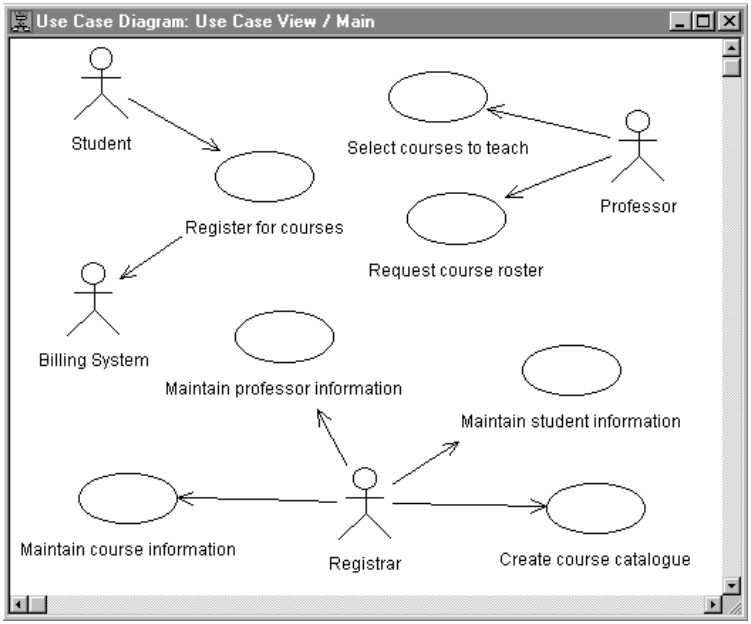

Figure 3-9 Main Use Case Diagram

An additional use case diagram is shown in Figure 3-10.

SUMMARY

SYSTEM BEHAVIOR IS documented in a use case model that illustrates the system's intended functions (use cases), its surroundings (actors), and the relationships between the use cases and actors (use case diagrams). The most important role of a use case model is to communicate the system's functionality and behavior to the customer or end user.

The use case model starts in the Inception Phase with the identification of the actors and the principal use cases for the system. The model is then matured in the Elaboration Phase.

Actors are not part of the system—they represent anyone or anything that must interact with the system under development. Use cases represent the functionality provided by the system. They model a dialogue between an actor and the system.

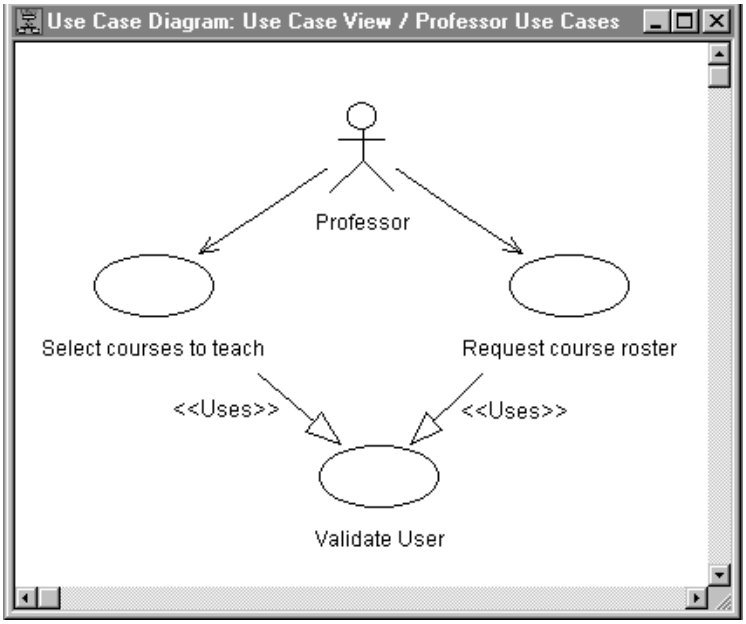

Figure 3-10 An Additional Use Case Diagram

Each use case contains a flow of events, which is a description of the events needed to accomplish the use case functionality. The flow of events is written in terms of what the system should do, not how the system does it. A use case diagram is a graphical representation of some or all of the actors, use cases, and their interactions for a system.

Two popular types of use case relationships are uses and extends. A uses relationship is drawn to show functionality that is shared by several use cases; an extends relationship depicts optional behavior of a use case.