

System Analysis and Design

Eighth Edition

Alan Dennis, Barbara Wixom, Roberta M. Roth

Chapter 4

Understanding Processes with Use Cases and Process Models

Objectives

- Understand the process used to identify business processes and use cases.
- Understand the process used to create use-case diagrams.
- Understand the process used to model business processes with activity diagrams.
- Understand the rules and style guidelines for activity diagrams.
- Understand the process used to create use-case descriptions.
- Understand the rules and style guidelines for use-case descriptions.
- Be able to create functional models of business processes using use-case diagrams, activity diagrams, and use-case descriptions.

Introduction

- A key aspect of determining the requirements for the new system is understanding the ***user requirements***
 - The things the users need to accomplish with the new system
- Use cases help us understand and clarify the users' required interactions with the system and can help us more fully understand the functional requirements of the new system
- A use case represents how a system interacts with its environment
- Use cases are especially valuable for business system applications and websites
- Process models have been a part of structured systems analysis and design techniques for many years

What Is a Use Case?

- A use case depicts a set of activities performed to produce some output result
- Each use case describes how an ***event triggers*** actions performed by the system and the user
 - Everything in the system can be thought of as a response to some trigger event

Basic Information

- Each use case has a *name* and *number*, and brief *description*.
- The *priority* may be assigned to indicate the relative significance.
- The *actor* refers to a person, another system, or a hardware device that interacts with the system to achieve a useful goal.
- The *trigger* for the use case – the event that causes the use case to begin.
- Events triggers can be *external* or *temporal*

Create Preliminary Custom Drone Order Use Case—Casual Format

Use Case Name: Create preliminary custom drone order	ID: UC-6	Priority: High
Actor: Customer		
Description: The customer selects and customizes a commercial drone to purchase		
Trigger: Customer wants to purchase a commercial drone		
Type: <input checked="" type="checkbox"/> External <input type="checkbox"/> Temporal		

Preconditions

- It is important to define clearly what needs to be accomplished before each use case begins
- Preconditions define the state the system must be in before the use case commences

Preconditions:

1. The customer is authenticated by logging in to his account
 2. The Sales System Order Processing application is online
-

Normal Courses

- The normal course lists the steps that are performed when everything flows smoothly in the system
- Sometimes called the *happy path*

Normal Course:

- 1.0 Order a customized drone
 1. The customer selects a base model drone from a list of models
 2. The system provides availability status for that model (in stock, out of stock)
 3. For out of stock status, system displays expected date available
 - a. Customer accepts future availability date; proceed to step 4
 - b. Customer rejects future availability date; return to step 1
 4. The system displays a list of options and upgrades for the selected model
 5. The customer selects desired model options and upgrades
 6. Preliminary order with cost estimate is created and displayed
 7. Customer may return to step 4, confirm order, save for future consideration, or exit without saving
 8. Unconfirmed orders are stored in Unconfirmed Custom Order datastore
 9. Confirmed orders are saved in Confirmed Custom Order datastore
 10. Shop manager is notified of Confirmed Order requiring approval
-

Postconditions

- In this section of the use case, we define the final products of this use case
- These postconditions also serve to define the preconditions for the next use case in the series

Postconditions:

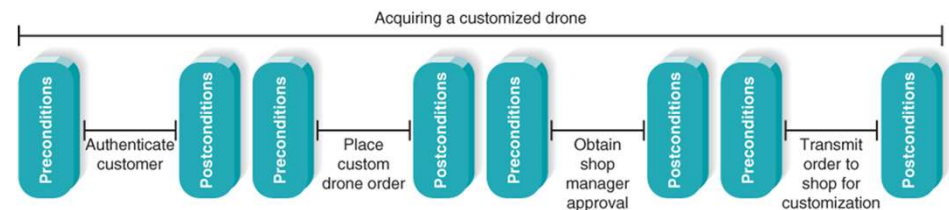
1. Unconfirmed order is stored in Unconfirmed Custom Order datastore
2. Confirmed order is stored in Confirmed Custom Order datastore
3. Shop manager sent notice of Confirmed Order requiring approval

Exceptions

- To be complete, a use case should describe any error conditions or exceptions that may occur as the use case steps are performed
- These are not normal branches in decision logic but are unusual occurrences or errors that could potentially be encountered and will lead to an unsuccessful result
- We want to be sure that the system does not fail while in use because of an error

Use Cases in Sequence

- Uses cases often performed in sequence
- No single use case should be too large
- Important to define initial and ending states



Additional Use Case Issues

- Alternative paths
- Summary of inputs and outputs
- Frequency of use
- Business rules
- Special requirements
- Assumptions
- Notes and issues

More Elaborate Use Cases are Especially Valuable When...

- User representatives are not actively engaged with the development team throughout the project
- The application is complex and has a high risk associated with system failures
- Comprehensive test cases will be based on the user requirements
- Collaborating remote teams need a detailed, shared understanding of the user requirements

Use Case Practical Tips

- Use gradual refinement
- Concentrate on describing the user's objectives with the system completely and accurately
- Keep both audiences in mind – users and developers
- Create use cases only when needed to clarify what the system must do from the user's perspective
 - Not needed for simple events

Use Cases and the Functional Requirements

- Use cases are useful tools to clarify user requirements
- Use cases convey only the user's point of view
- Transforming the user's view into the developer's view through functional requirements is one of the system analyst's key contributions
- The derived functional requirements tell the developers more about what the system must do

Steps for Writing for Use Cases

1. Identify the use cases
2. Identify the major steps within each use case
3. Identify elements within steps
4. Confirm the use case

Creating Use Cases

- Identify events the system must respond to – develop event-response list
- Create use case form for the complex events
- For each use case:
 - Identify the major steps
 - Identify elements with each major step (inputs and outputs)
 - Confirm use case with users through role-playing
- Revise functional requirements as needed

Identify the Major Steps for Each Use Case

- The analyst should ask the users what tasks need to be completed before the use case steps can begin
- Next, the user–system interactions should be outlined as a series of steps in the Normal Course section of the form
- Each step should be about the same size as the others
- Occasionally, a use case is so simple that further refinement is not needed
- Once the steps have been outlined at the proper level of detail, the postconditions can be completed

Identify Elements within Steps






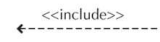
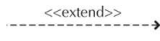

- The last column (“Information for Steps”) must be completed, and arrows may be drawn to describe inputs and outputs from the steps
- The goal at this point is to identify the major inputs and outputs for each step
- The users and analysts then return to the steps in the use case and begin tracing the flow of the steps
- It is not unusual at this point for users to discover that they forgot to list the entire steps
- The Summary area for inputs and outputs found at the end of the use case form is completed once the team is satisfied with the steps, inflows, and outflows

Confirm the Use Case

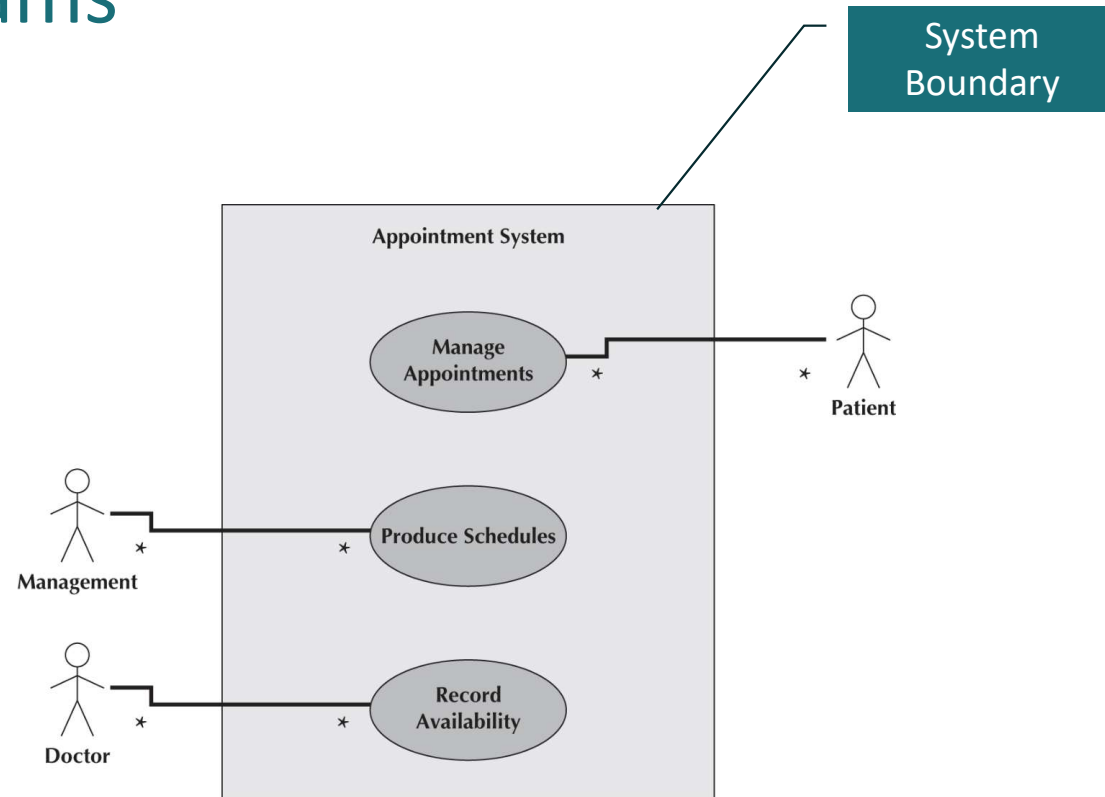
- The final step is for the users to confirm that the use case is correct as written
- Review the use case with the users to make sure that each step and each input and output are correct

Use Case Diagrams

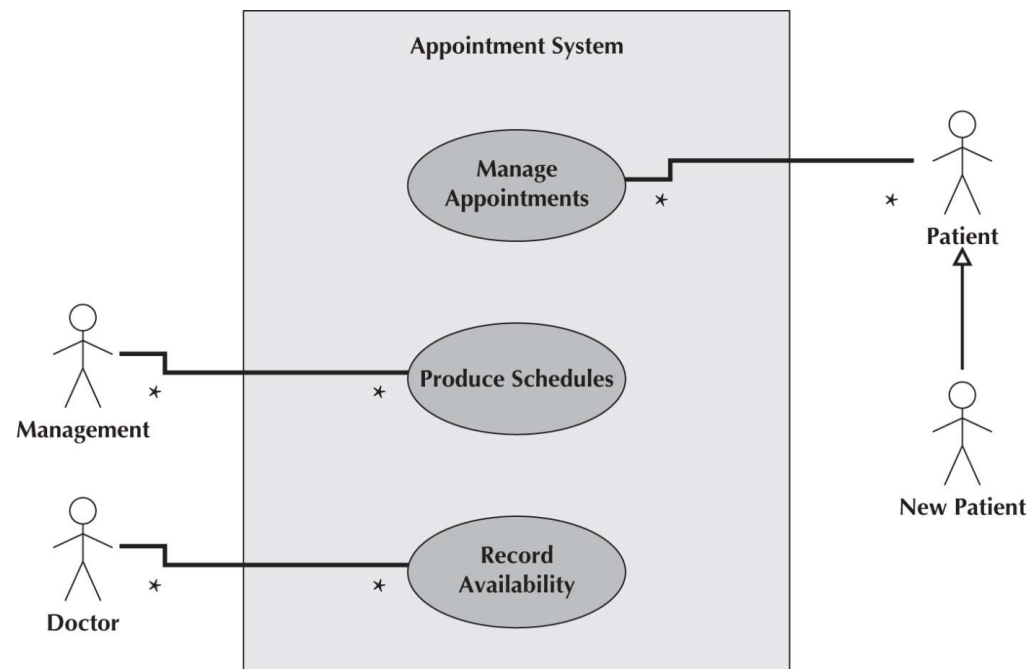
- Elements

<p>An actor:</p> <ul style="list-style-type: none"> ■ Is a person or system that derives benefit from and is external to the subject. ■ Is depicted as either a stick figure (default) or, if a nonhuman actor is involved, a rectangle with <<actor>> in it (alternative). ■ Is labeled with its role. ■ Can be associated with other actors using a specialization/superclass association, denoted by an arrow with a hollow arrowhead. ■ Is placed outside the subject boundary. 	 <p>Actor/Role</p> 
<p>A use case:</p> <ul style="list-style-type: none"> ■ Represents a major piece of system functionality. ■ Can extend another use case. ■ Can include another use case. ■ Is placed inside the system boundary. ■ Is labeled with a descriptive verb-noun phrase. 	
<p>A subject boundary:</p> <ul style="list-style-type: none"> ■ Includes the name of the subject inside or on top. ■ Represents the scope of the subject, e.g., a system or an individual business process. 	
<p>An association relationship:</p> <ul style="list-style-type: none"> ■ Links an actor with the use case(s) with which it interacts. 	
<p>An include relationship:</p> <ul style="list-style-type: none"> ■ Represents the inclusion of the functionality of one use case within another. ■ Has an arrow drawn from the base use case to the used use case. 	
<p>An extend relationship:</p> <ul style="list-style-type: none"> ■ Represents the extension of the use case to include optional behavior. ■ Has an arrow drawn from the extension use case to the base use case. 	
<p>A generalization relationship:</p> <ul style="list-style-type: none"> ■ Represents a specialized use case to a more generalized one. ■ Has an arrow drawn from the specialized use case to the base use case. 	

Use Case Diagrams

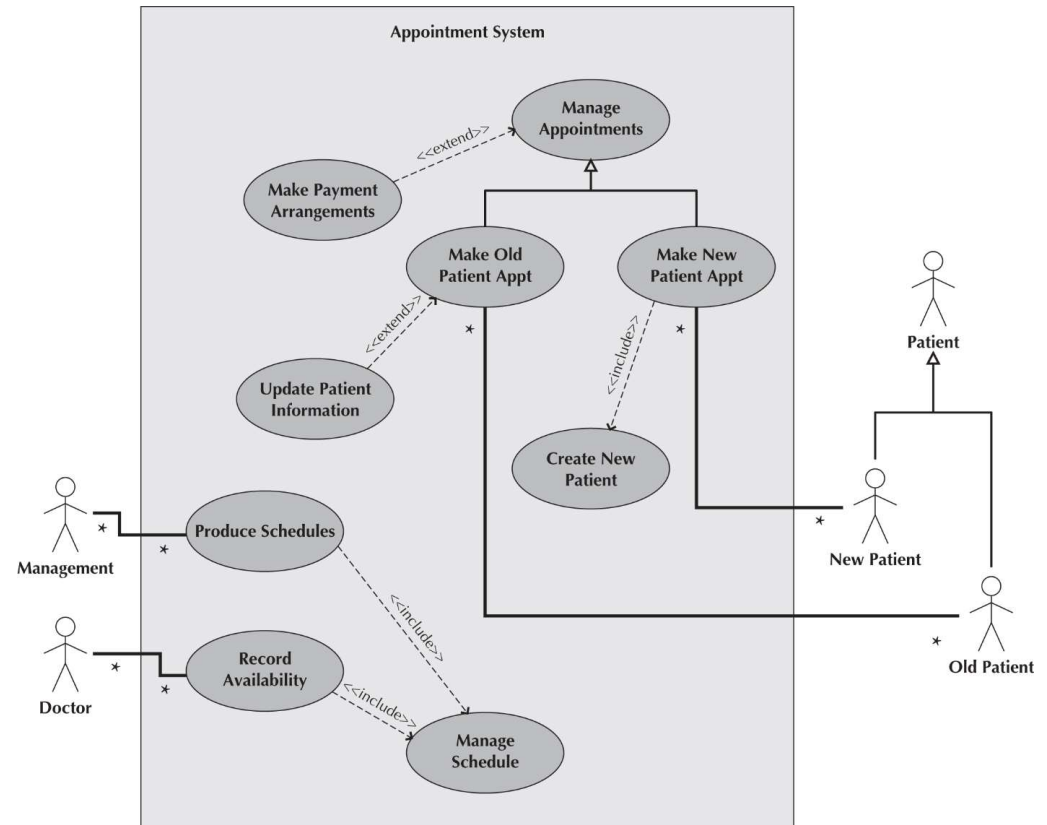


Use Case Diagram – Specialized Actor

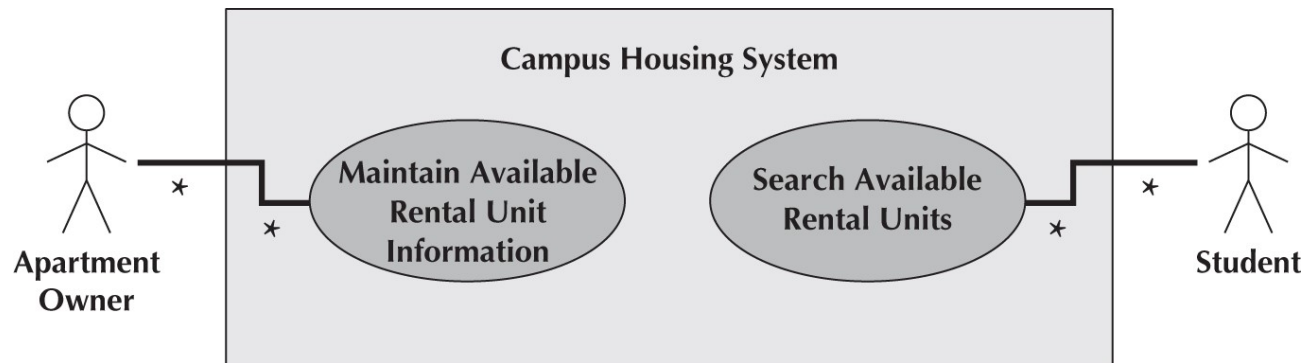


Use Case Diagrams – <<extends>> and <<includes>>

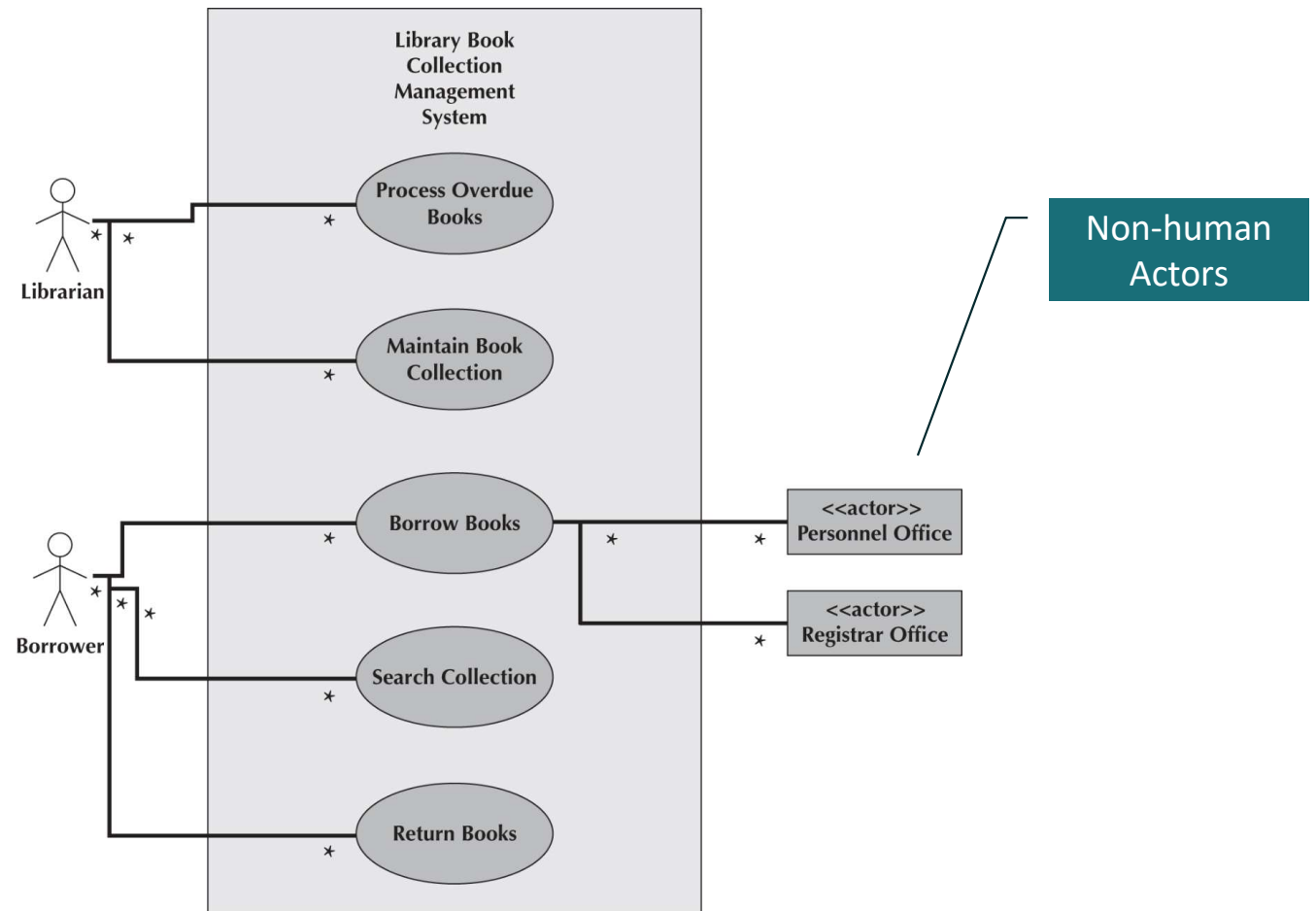
- In practice
 - <<includes>> relationships are used to show the breakout of higher level Use Cases into Lower Level Use Cases
 - <<extends>> relationships are to show Uses Cases that are added after the initial development. Much like adding a new Class/Object in an Object Oriented



Use Case Diagrams - Example











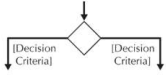
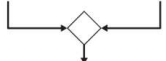



Use Case Diagram - Example



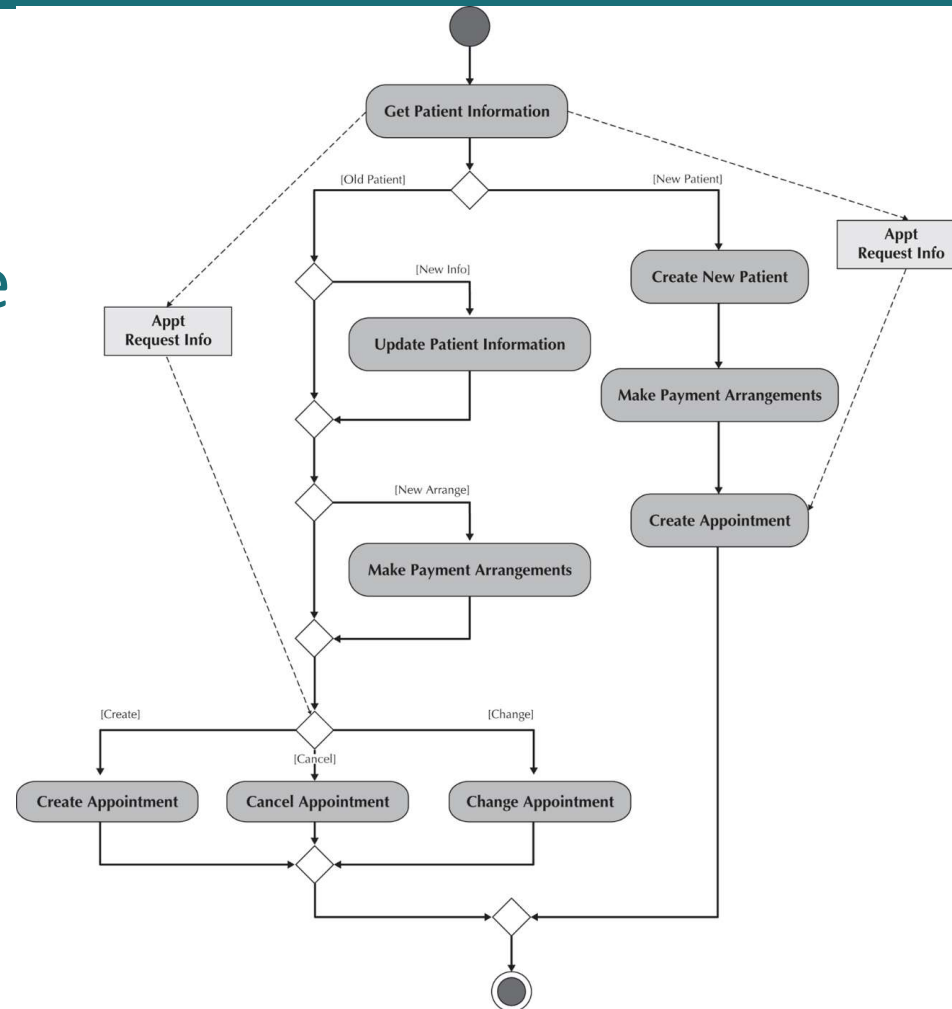
Control Flows and Object Flows

- Control flows model the paths of execution through a business process.
- A control flow is portrayed as a solid line with an arrowhead on it showing the direction of flow.
- Control flows can be attached only to actions or activities.
- Object flows model the flow of objects through a business process.
- Because activities and actions modify or transform objects, object flows are necessary to show the actual objects that flow into and out of the actions or activities.
- An object flow is depicted as a dashed line with an arrowhead on it showing the direction of flow.
- An individual object flow must be attached to an action or activity on one end and an object node on the other end.

Activity Diagram Pallet

An action: <ul style="list-style-type: none"> ■ Is a simple, nondecomposable piece of behavior. ■ Is labeled by its name. 	
An activity: <ul style="list-style-type: none"> ■ Is used to represent a set of actions. ■ Is labeled by its name. 	
An object node: <ul style="list-style-type: none"> ■ Is used to represent an object that is connected to a set of object flows. ■ Is labeled by its class name. 	
A control flow: <ul style="list-style-type: none"> ■ Shows the sequence of execution. 	
An object flow: <ul style="list-style-type: none"> ■ Shows the flow of an object from one activity (or action) to another activity (or action). 	
An initial node: <ul style="list-style-type: none"> ■ Portrays the beginning of a set of actions or activities. 	
A final-activity node: <ul style="list-style-type: none"> ■ Is used to stop all control flows and object flows in an activity (or action). 	
A final-flow node: <ul style="list-style-type: none"> ■ Is used to stop a specific control flow or object flow. 	
A decision node: <ul style="list-style-type: none"> ■ Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. ■ Is labeled with the decision criteria to continue down the specific path. 	
A merge node: <ul style="list-style-type: none"> ■ Is used to bring back together different decision paths that were created using a decision node. 	
A fork node: <ul style="list-style-type: none"> Is used to split behavior into a set of parallel or concurrent flows of activities (or actions) 	
A join node: <ul style="list-style-type: none"> Is used to bring back together a set of parallel or concurrent flows of activities (or actions) 	
A swimlane: <ul style="list-style-type: none"> Is used to break up an activity diagram into rows and columns to assign the individual activities (or actions) to the individuals or objects that are responsible for executing the activity (or action) Is labeled with the name of the individual or object responsible 	

Activity Diagram for the Manage Appointments Use



Control Nodes

- An **initial node** is shown as a small filled-in circle.
- A **final-activity node** is used to stop the process being modeled.
- A **final-flow node** is similar to a final-activity node, except that it stops a specific path of execution through the business process but allows the other concurrent or parallel paths to continue
- The **decision node** is used to represent the actual test condition that determines which of the paths exiting the decision node is to be traversed.
 - A **guard condition** represents the value of the test for that particular path to be executed.
- The **merge node** is used to bring back together multiple mutually exclusive paths that have been split based on an earlier decision

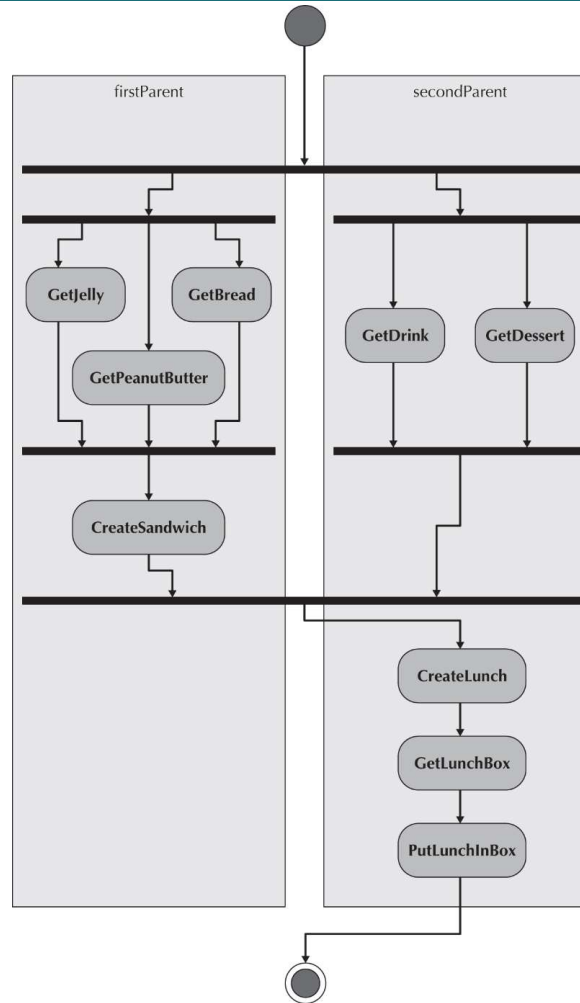
Control Nodes (Cont)

- The **fork node** is used to split the behavior of the business process into multiple parallel or concurrent flows.
- The **join node** simply brings back together the separate parallel or concurrent flows in the business process into a single flow.

Swimlane

- Identifies activities that are dependent upon the object in the system that will be responsible for implementing it.
- Can be drawn vertically or horizontally.

Swimlane (Cont)



Guidelines for Creating Activity Diagrams

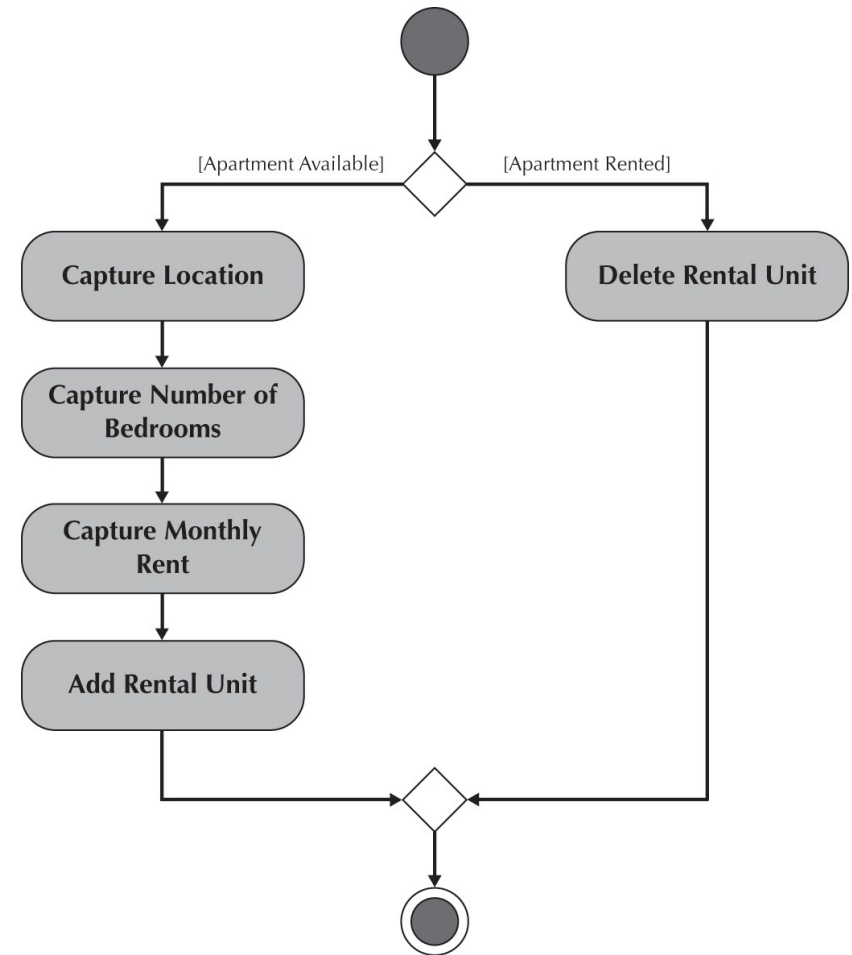
- Because an activity diagram can be used to model any kind of process, you should set the context or scope of the activity being modeled. Once you have determined the scope, you should give the diagram an appropriate title.
- You must identify the activities, control flows, and object flows that occur between the activities.
- You should identify any decisions that are part of the process being modeled.
- You should attempt to identify any prospects for parallelism in the process.
- You should draw the activity diagram.

Creating Activity Diagrams

- Choose a business process that was previously identified to model. To do this, you should review the requirements definition and the use-case diagram created to represent the requirements, as well as other relevant documentation
- Identify the set of activities necessary to support the business process. These could/should come from the Use Cases themselves.
- Identify the control flows and nodes necessary to document the logic of the business process.
- Identify the object flows and nodes necessary to support the logic of the business process.
- Lay out and draw the activity diagram to document the business process. For esthetic and understandability reasons, just as when drawing a use-case diagram, you should attempt to minimize potential line crossings. Based on the previous steps and carefully laying out the diagram, the activity diagram in Figure 4-8 was created to document the Manage Appointments business process.

Campus Housing Activity Diagram

- Maintain Available Rental Unit Information Use Case



Chapter Review (1 of 3)

- Explain the purpose of a use case in the analysis phase of the SDLC.
- Explain why use cases are commonly used in the analysis phase.
- Discuss the various sections found in a use case form and the purpose and content of each section.
- Explain how use cases help the systems analyst create a more in-depth understanding of the system's functional requirements.
- Describe how use cases can contribute to the development of test plans for the new system.

Chapter Review (2 of 3)

- Discuss the four steps of the process used to create use cases.
- Define the meaning and purpose of the four basic symbols found on a data flow diagram.
- Explain the meaning and purpose of a process model's context diagram.
- Explain the meaning and purpose of a process model's level 0 diagram.
- Explain the meaning and purpose of a process model's level 1 diagrams.

Chapter Review (3 of 3)

- Explain the concept of decomposition and why process models are created as a hierarchy of DFDs.
- Describe several common syntax and semantic errors found on DFDs.
- Discuss the process used to create a process model.
- Discuss how the process model contributes to the development of the new information system.

Key Terms

- Actor
- Balancing
- Bundle
- Children
- Context diagram
- Data flow
- Data flow diagram
- (DFD)
- Data store
- Decision tables
- Decision trees
- Decomposition
- DFD fragment
- Essential use cases
- Event-driven modeling
- Event triggers
- External entity
- External trigger
- Happy path
- IF statements
- Inputs
- Iteration
- Layout
- Level 0 diagram or level 0 DFD
- Level 1 diagram, or level 1 DFD
- Level 2 DFD
- Logical process model
- Outputs

Key Terms Continued

- Parent
- Physical model
- Postconditions
- Preconditions
- Primary actor
- Priority
- Process
- Processes
- Process model
- Role-play
- Semantics error
- Step
- Structured English
- Syntax error
- Temporal trigger
- Trigger
- Use case
- Use case packages
- User requirements
- User role
- Viewpoint
- Visualization