Chapter 6: Behavioral Modeling

Learning Objectives

- Understand the rules and style guidelines for sequence and communication diagrams and behavioral state machines.
- Understand the processes used to create sequence and communication diagrams, behavioral state machines and CRUDE matrices.
- Be able to create sequence and communication diagrams, behavioral state machines and CRUDE matrices.
- Understand the relationship between the behavioral models and the structural and functional models.

Introduction

- Behavioral models describe the internal behavior of a system
- Behavioral model types:
  - Representations of the details of a business process identified by use-cases
    - Interaction diagrams (Sequence & Communication)
    - Shows how objects collaborate to provide the functionality defined in the use cases.
  - Representations of changes in the data
    - Behavioral state machines
- Focus (for now) is on the dynamic view of the system, not on how it is implemented

Behavioral Models

- Analysts view the problem as a set of use cases supported by a set of collaborating objects
  - Aids in organizing and defining the software
  - Behavioral models depict this view of the business processes:
    - How the objects interact and form a collaboration to support the use cases
    - An internal view of the business process described by a use case
- Creating behavioral models is an iterative process which may induce changes in other models

Interaction Diagrams

- Objects—an instantiation of a class
  - Patient is a class
  - Mary Wilson is an instantiation of the patient class (object)
- Attributes—characteristics of a class
  - Patient class: name, address, phone, etc.
- Operations—the behaviors of a class, or an action that an object can perform
- Messages—information sent to objects to tell them to execute one of their behaviors
  - A function call from one object to another
- Types
  - Sequence Diagrams—emphasize message sequence
  - Communication Diagrams—emphasize message flow

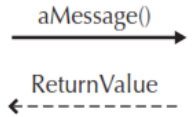
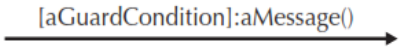
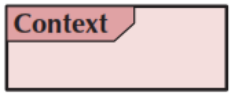
Sequence Diagrams

- Illustrate the objects that participate in a single use-case
- A dynamic model
  - Shows the sequence of messages that pass between objects
  - Aid in understanding real-time specifications and complex use-cases
- Generic diagram shows all scenarios for a use-case
- Instance diagrams show a single scenario

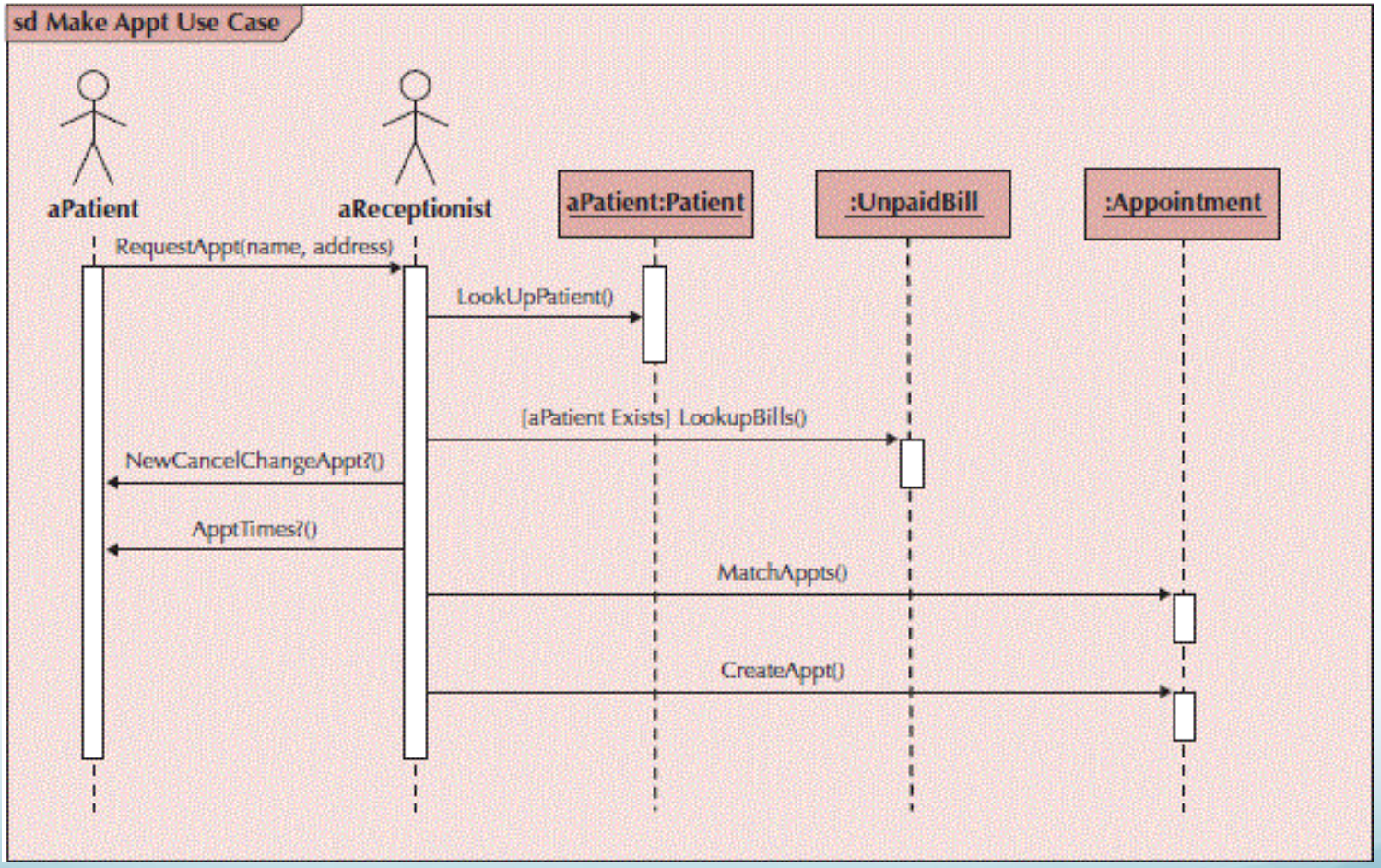
Sequence Diagram Syntax

Term and Definition	Symbol
An actor: ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative).	 anActor <<actor>> anActor
An object: ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram.	anObject : aClass
A lifeline: ■ Denotes the life of an object during a sequence. ■ Contains an X at the point at which the class no longer interacts.	

More Sequence Diagram Syntax

An execution occurrence: ■ Is a long narrow rectangle placed atop a lifeline. ■ Denotes when an object is sending or receiving messages.	
A message: ■ Conveys information from one object to another one. ■ A operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.	
A guard condition: ■ Represents a test that must be met for the message to be sent.	
For object destruction: ■ An X is placed at the end of an object's lifeline to show that it is going out of existence.	X
A frame: ■ Indicates the context of the sequence diagram.	

Sample Sequence Diagram



Building Sequence Diagrams

- Set the context
- Identify actors and objects that interact in the use-case scenario
- Set the lifeline for each object
- Add messages by drawing arrows
  - Shows how they are passed from one object to another
  - Include any parameters in parentheses
  - Obvious return values are excluded
- Add execution occurrence to each object's lifeline
- Validate the sequence diagram
  - Ensures that it depicts all of the steps in the process

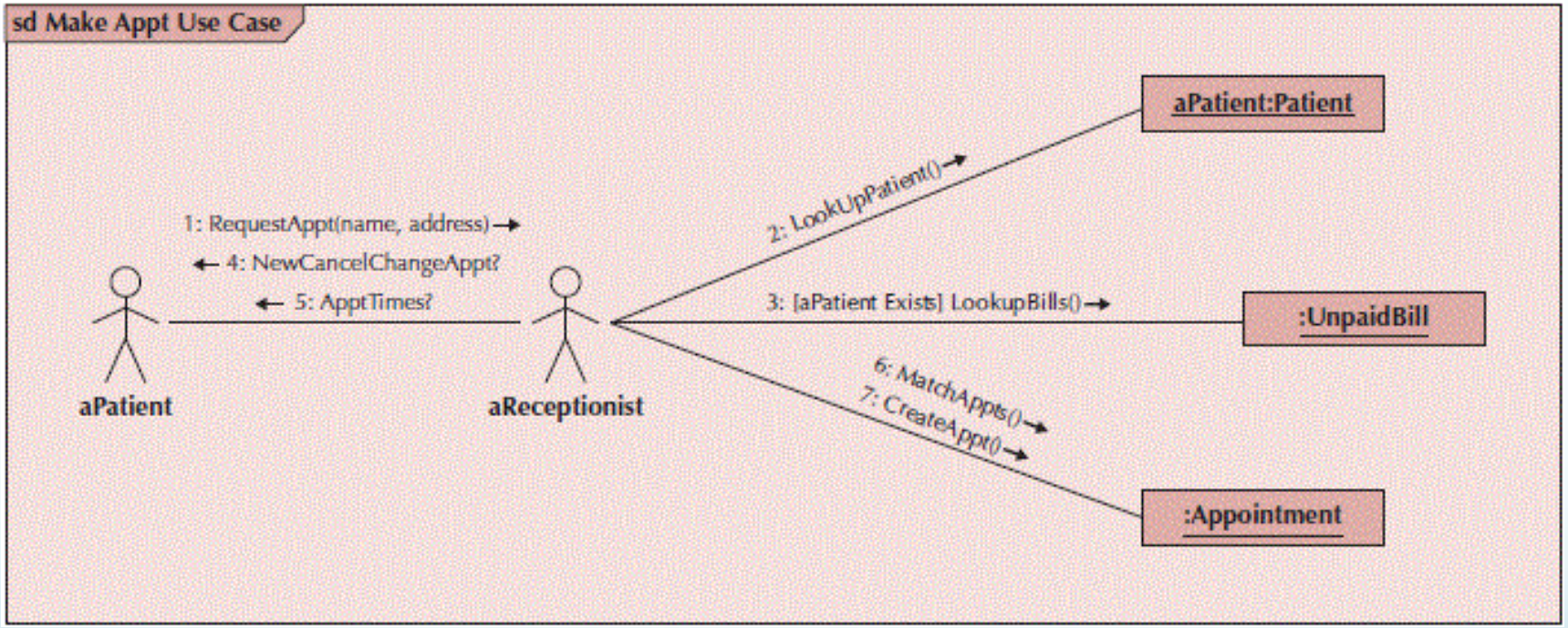
Communication Diagrams

- Depict the dependencies among the objects
- An object diagram that shows message passing relationships
- Emphasize the flow through a set of objects

Communication Diagram Syntax

Term and Definition	Symbol
An actor: ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a collaboration by sending and/or receiving messages. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative).	 anActor <<actor>> anActor
An object: ■ Participates in a collaboration by sending and/or receiving messages.	anObject : aClass
An association: ■ Shows an association between actors and/or objects. ■ Is used to send messages.	
A message: ■ Conveys information from one object to another one. ■ Has direction shown using an arrowhead. ■ Has sequence shown by a sequence number.	SeqNumber: aMessage →
A guard condition: ■ Represents a test that must be met for the message to be sent.	SeqNumber: [aGuardCondition]: aMessage →
A frame: ■ Indicates the context of the communication diagram.	Context

Sample Communication Diagram



Building Communication Diagrams

- Set the context
- Identify objects, actors and associations between them
- Lay out the diagram
- Add the messages
- Validate the model

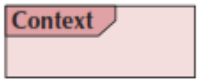
Behavioral State Machines

- Objects may change state in response to an event
- Different states are captured in this model
  - Shows the different states through which a single object passes during its life
  - May include the object's responses and actions
- Example: patient states
  - New patient—has not yet been seen
  - Current patient—is now receiving treatment
  - Former patient—no longer being seen or treated
- Typically used only for complex objects

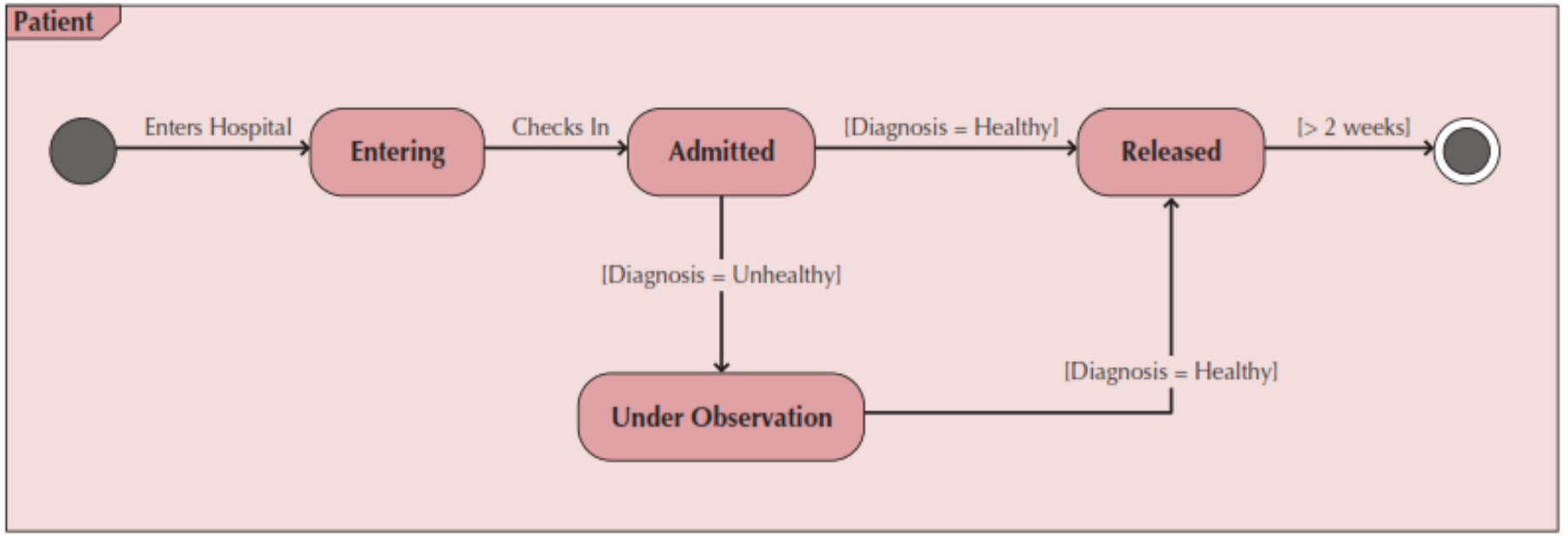
Components of State Machines

- States—values of an object's attributes at a point in time
- Events—the cause of the change in values of the object's attributes
- Transitions—movement of an object from one state to another
  - May include a guard condition to flag that a condition is true and allow the transition

State Machine Syntax

Term and Definition	Symbol
A state: ■ Is shown as a rectangle with rounded corners. ■ Has a name that represents the state of an object.	
An initial state: ■ Is shown as a small, filled-in circle. ■ Represents the point at which an object begins to exist.	
A final state: ■ Is shown as a circle surrounding a small, filled-in circle (bull's-eye). ■ Represents the completion of activity.	
An event: ■ Is a noteworthy occurrence that triggers a change in state. ■ Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time. ■ Is used to label a transition.	anEvent
A transition: ■ Indicates that an object in the first state will enter the second state. ■ Is triggered by the occurrence of the event labeling the transition. ■ Is shown as a solid arrow from one state to another, labeled by the event name.	
A frame: ■ Indicates the context of the behavioral state machine.	

Sample State Machine



Guidelines for Creating Behavioral State Machines

- Use only for complex objects
- Draw the initial state in the upper left corner
- Draw the final state in the bottom right corner
- Use simple, but descriptive names for states
- Look out for “black holes” and “miracles”
- Ensure guard conditions are mutually exclusive
- Ensure transitions are associated with messages and operations

Building a Behavioral State Machine

- Set the context
- Identify the states of the object
  - Initial
  - Final
  - Stable states during its lifetime
- Lay out the diagram—use a left to right sequence
- Add the transitions
  - Identify the triggers (events that cause the transition)
  - Identify the actions which execute
  - Identify the guard conditions
- Validate the model—ensure all states are reachable

CRUDE Analysis

- Helps to identify object collaborations
- Labels object interaction in 5 possible ways:
  - Create—can one object create another?
  - Read—can one object read the attributes of another?
  - Update—can one object change values in another?
  - Delete—can one object delete another object?
  - Execute—can one object execute the operations of another?
- Utilizes a matrix to represent objects and their interactions
- Most useful as a system-wide representation

Sample CRUDE Matrix

	Student Actor	Faculty/ Staff Actor	Guest Actor	Librarian Actor	Personnel Office Actor	Registrar's Office Actor	Book	Book Collection	Student Class	Faculty/ Staff Class	Guest Class	Interlibrary Loan System	Library	Storage
Student Actor				E			R,E	R				E		
Faculty/Staff Actor				E			R,E	R				E		
Guest Actor				E			R,E	R				E		
Librarian Actor	E	E	E		R,E	R,E	C,R,U,D,E	R,U,E	R,U	R,U	C,R,U,D,E	R,E		
Personnel Office Actor														
Registrar's Office Actor														
Book														
Book Collection														
Student Class														
Faculty/Staff Class														
Guest Class														
Interlibrary Loan System														
Library														
Storage														

Verifying & Validating Behavioral Models

- Actors must be consistent between models
- Messages on sequence diagrams must match associations on communication diagrams
- Every message on a sequence diagram must appear on an association in a communication diagram
- Guard conditions on a sequence diagram must appear on a communication diagram
- Sequence of messages must correspond to the top down ordering of messages being sent
- State transitions must be associated with a message on a sequence diagram
- Entries in a CRUDE matrix imply messages being sent

Summary

- Behavioral Models—provide a detailed view of how object collaborations support use-cases
- Interaction Diagrams
  - Sequence diagrams
  - Communication diagrams
- Behavioral State Machines—depicts the states of complex objects during its lifetime
- CRUDE Analysis—helps to identify potential collaborations
- Verifying & Validating behavioral models—ensures the completeness and consistency of the models