UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS
ILMU KOMPUTER

TOPIC 4

FUNCTIONAL MODELLING

ANALISIS DAN PERANCANGAN SISTEM INFORMASI
CSIM603183

Learning Objectives

1. Able to explain **the rules and component** of a **use case and use-case diagram**
2. Able to explain how to **create use case and use-case diagram**
3. Able to explain the **rules and component** of an **activity diagram**
4. Able to **create functional model** by using **activity diagrams, use cases, and use-case diagrams**

Session Outline

1. Use Case Diagram
2. Activity Diagram
3. Use Case Description
4. Functional Model (Use Cases as **core building blocks**): Creating, Verify and Validating Activity Diagram and Use Case Descriptions and Use Case Diagrams
5. (Additional) Use Case Ranking and Prioritization Matrix for Project Management (Estimation)

Introduction

- ….
- Project team gathers requirements from the users (last week)
- Using the gathered requirements, the project team then identifies the business processes and their environment using **use cases** and use-case diagrams
- Next, users work closely with the team to model the business processes in the form of activity diagrams, and the team documents the business processes described in the **use-case** and activity diagrams by creating a use-case description for each use case
- Finally, the team verifies and validates the business processes by ensuring that all three models (use-case diagram, activity diagram(s), and use case descriptions) agree with one another.
- Once all documented, the team is ready to move on to structural modeling (next week)
- ….

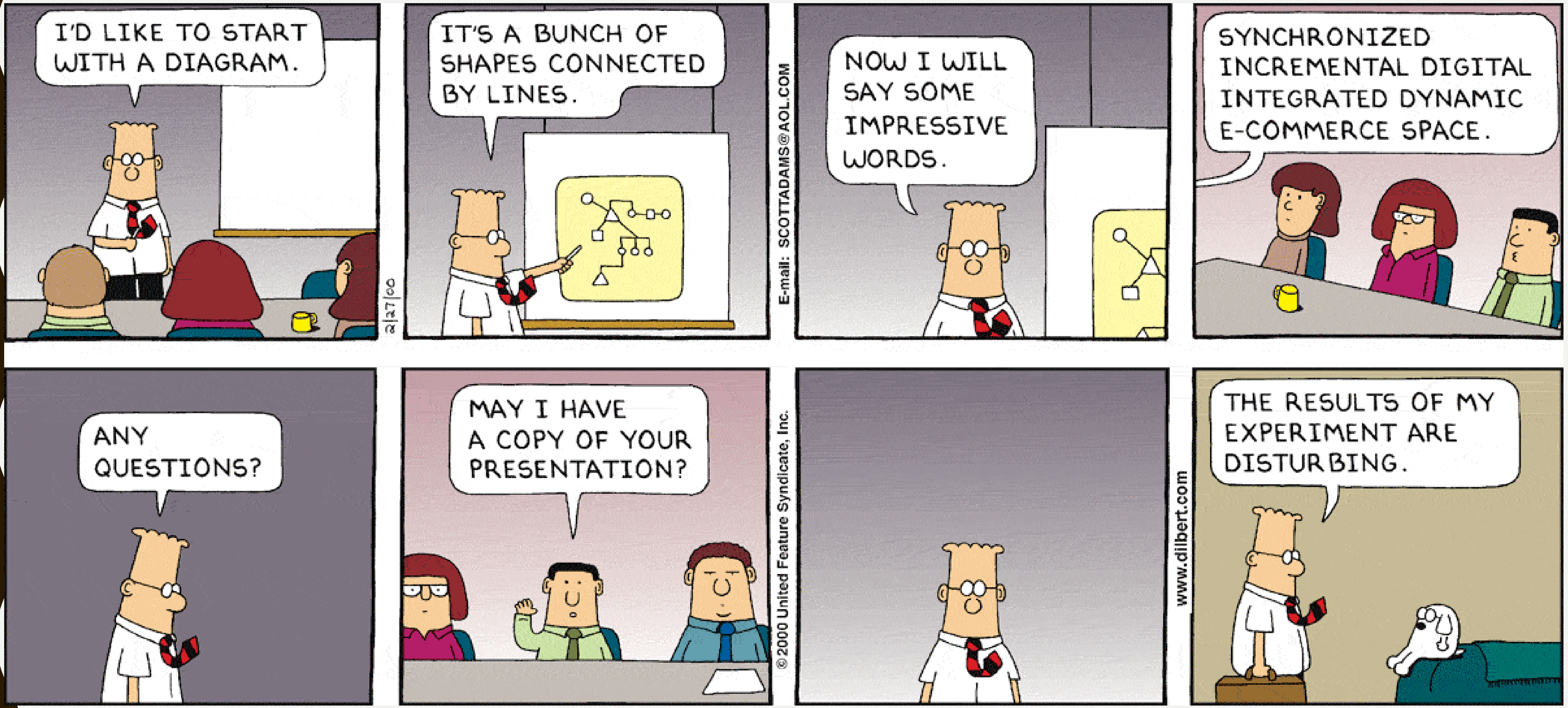
Introduction

1. **Functional Model:**
    - Use Case Diagram, Activity Diagram
2. **Structural Model (Conceptual Model):**
    - Class Diagram, Object Diagram
3. **Behavioral Model:**
    - Interaction Diagram, Behavioral State Machines

This slide is focused on: **Functional Model** (**Use Case and Activity Diagram**)

Key Ideas from the modelling perspectives

- **Use cases** (Use cases diagram and use case description) are the **logical model** used to describe to **basic functions** of an information system
  - Understand the **rules and style** guidelines for use cases and use case diagrams.
  - Understand the **process** used to create use cases and use case diagrams.
  - Become familiar with the use of use case points
- **Activity diagrams** support the **logical modeling** of business processes and workflows
  - Understand the **rules and style** guidelines for activity diagrams
  - Activity diagrams can be used to represent **Business Process Model** of the IS
- **Both** can be used to describe As-Is and To-Be of IS

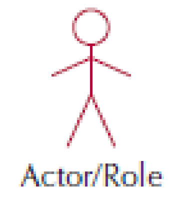
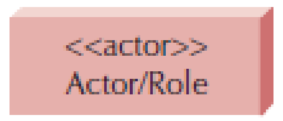
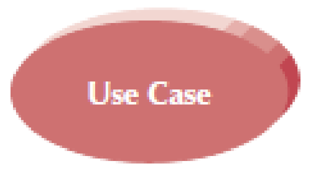
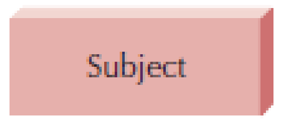

I'D LIKE TO START WITH A DIAGRAM.
IT'S A BUNCH OF SHAPES CONNECTED BY LINES.
2/27/00
E-mail: SCOTTADAMS@AOL.COM
NOW I WILL SAY SOME IMPRESSIVE WORDS.
SYNCHRONIZED INCREMENTAL DIGITAL INTEGRATED DYNAMIC E-COMMERCE SPACE.
ANY QUESTIONS?
MAY I HAVE A COPY OF YOUR PRESENTATION?
© 2000 United Feature Syndicate, Inc.
www.dilbert.com
THE RESULTS OF MY EXPERIMENT ARE DISTURBING.

1.1 USE CASES & USE-CASE DIAGRAMS

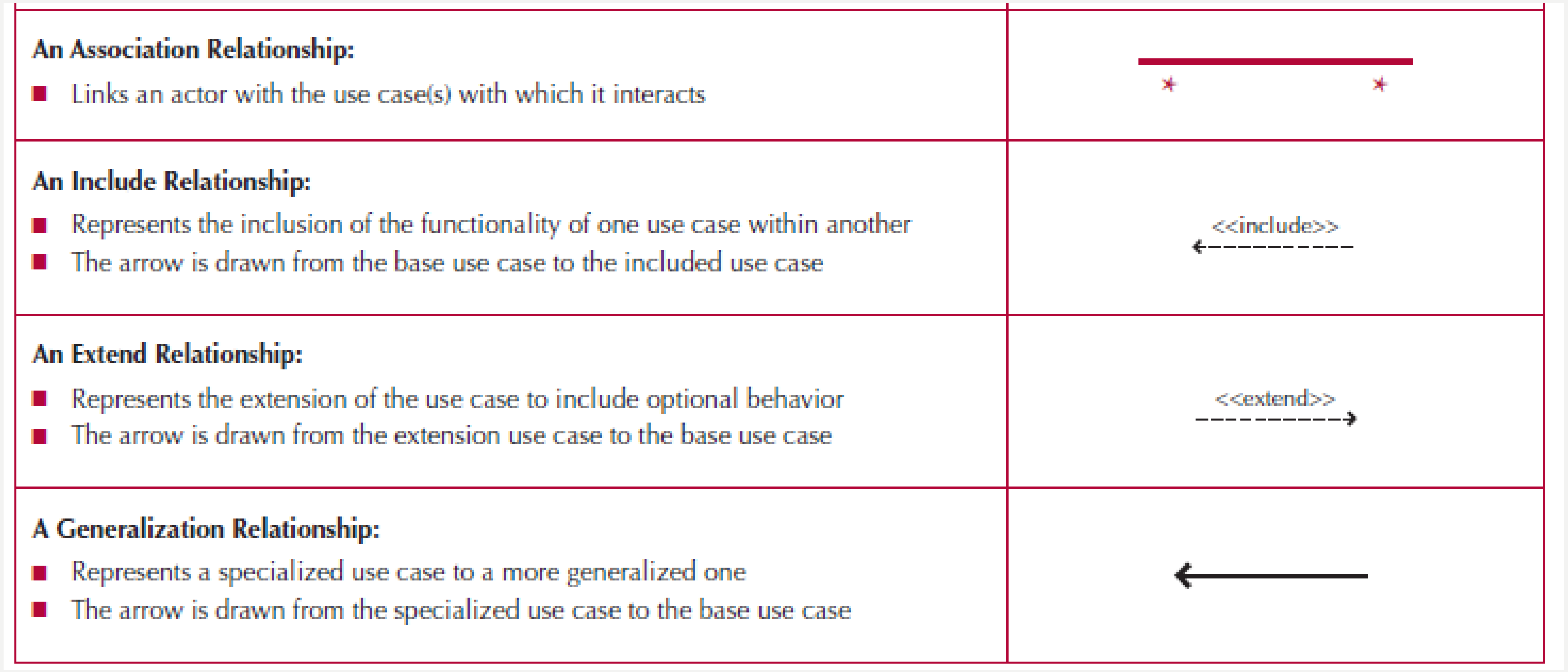
Overview

- An analyst can employ use cases and the use-case diagram to better understand the **functionality** of the system at a **very high level.**
- Typically, because a use-case diagram provides a simple, straightforward way of communicating to the users exactly what the system will do, a **use-case diagram is drawn when gathering and defining requirements** for the system.
- In this manner, the use-case diagram can encourage the users to provide additional **high-level requirements.**
- A use-case diagram illustrates in a very **simple way, the main functions of the system** and the different **kinds of users** that will interact with it

Syntax for Use-Case Diagram

An Actor: ■ 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 non-human actor is involved, as 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 ■ Are placed outside the subject boundary	 
A Use Case: ■ 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	
A Subject Boundary: ■ 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	

Syntax for Use-Case Diagram

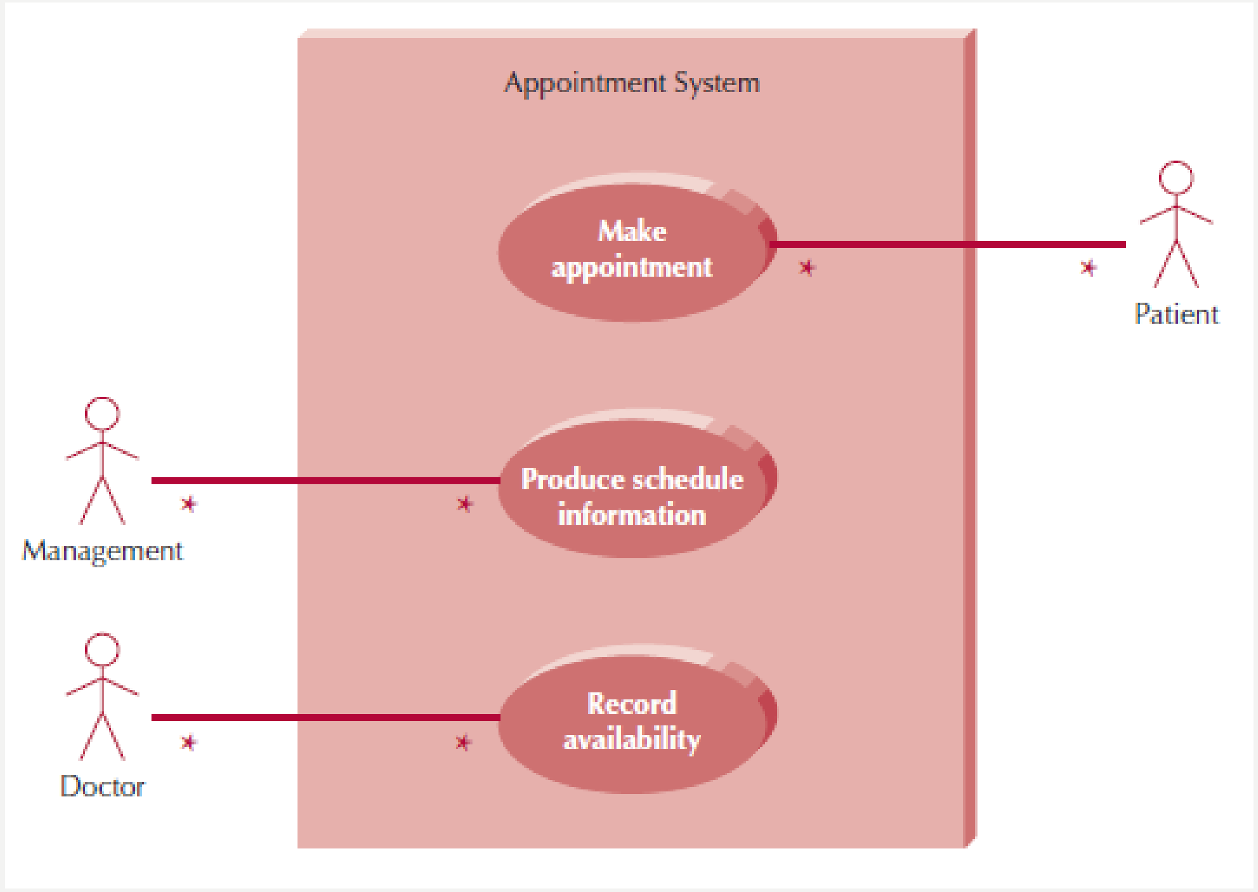


An Association Relationship: ■ Links an actor with the use case(s) with which it interacts	—————— * *
An Include Relationship: ■ Represents the inclusion of the functionality of one use case within another ■ The arrow is drawn from the base use case to the included use case	<<include>> ←- - - - - - - - - - -
An Extend Relationship: ■ Represents the extension of the use case to include optional behavior ■ The arrow is drawn from the extension use case to the base use case	<<extend>> - - - - - - - - - - - →
A Generalization Relationship: ■ Represents a specialized use case to a more generalized one ■ The arrow is drawn from the specialized use case to the base use case	←————

Actor and Association

- An ***actor*** is not a specific user, but **a role** that a user can play while interacting with the system.
- An **actor** can also represent **another system** in which the current system interacts.
- In this case, the actor optionally can be represented by a rectangle with **<<actor>>** and the name of the system.
- Basically, actors represent the principal elements in the environment in which the system operates.
- Actors can **provide input to the system, receive output from the system, or both**.

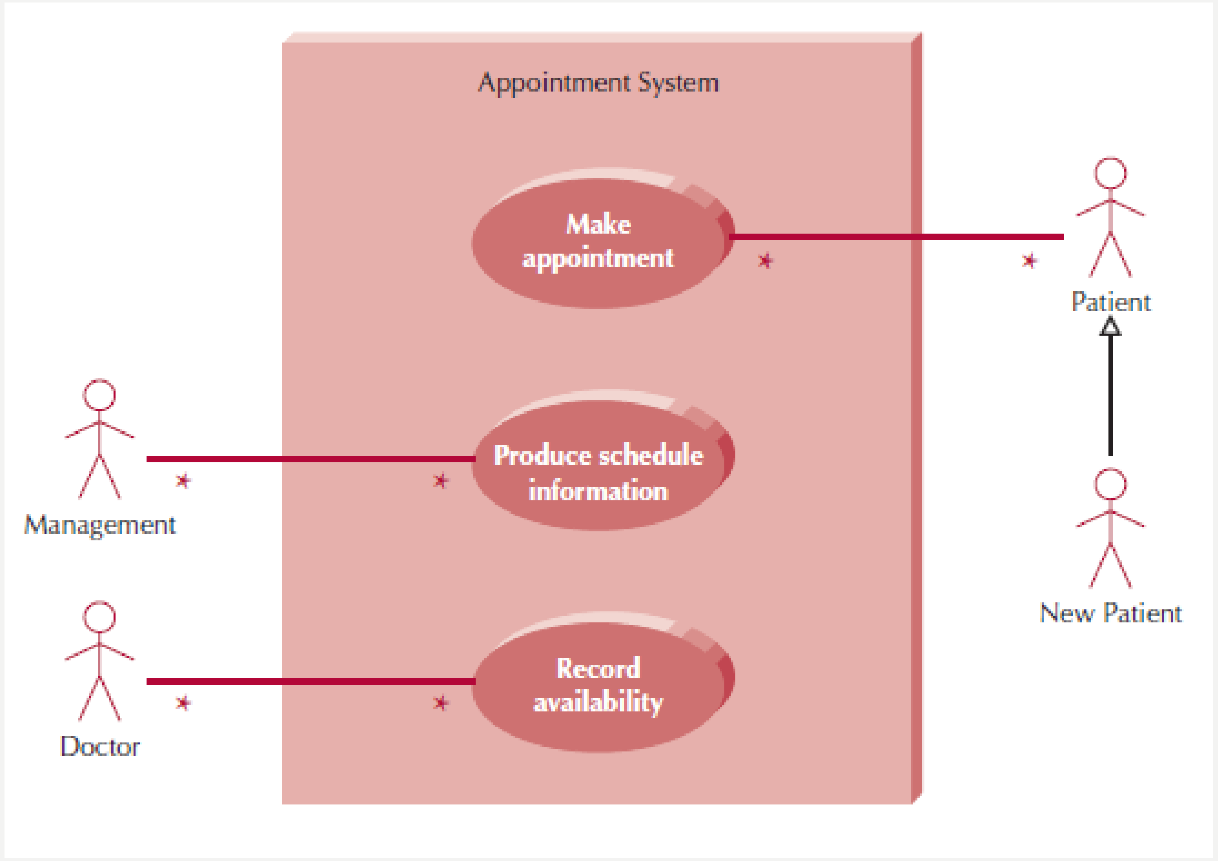
The Use-Case Diagram for Appointment System



Actor & Association

- Sometimes an actor plays a **specialized role** of a more general type of actor.
- For example, there may be times when a **new patient** interacts with the system in a way that is somewhat different than a **general patient**.
- In this case, a *specialized actor* (i.e., new patient) can be placed on the model, shown using a line with a hollow triangle at the end of the more general actor (i.e., patient).
- The **specialized actor will inherit** the behavior of the more general actor and extend it in some way.
- Actor specialization is related to use case generalization (see Use Case (generalization example) slide)

The Use-Case Diagram for Appointment System (a Specialized Actor added)



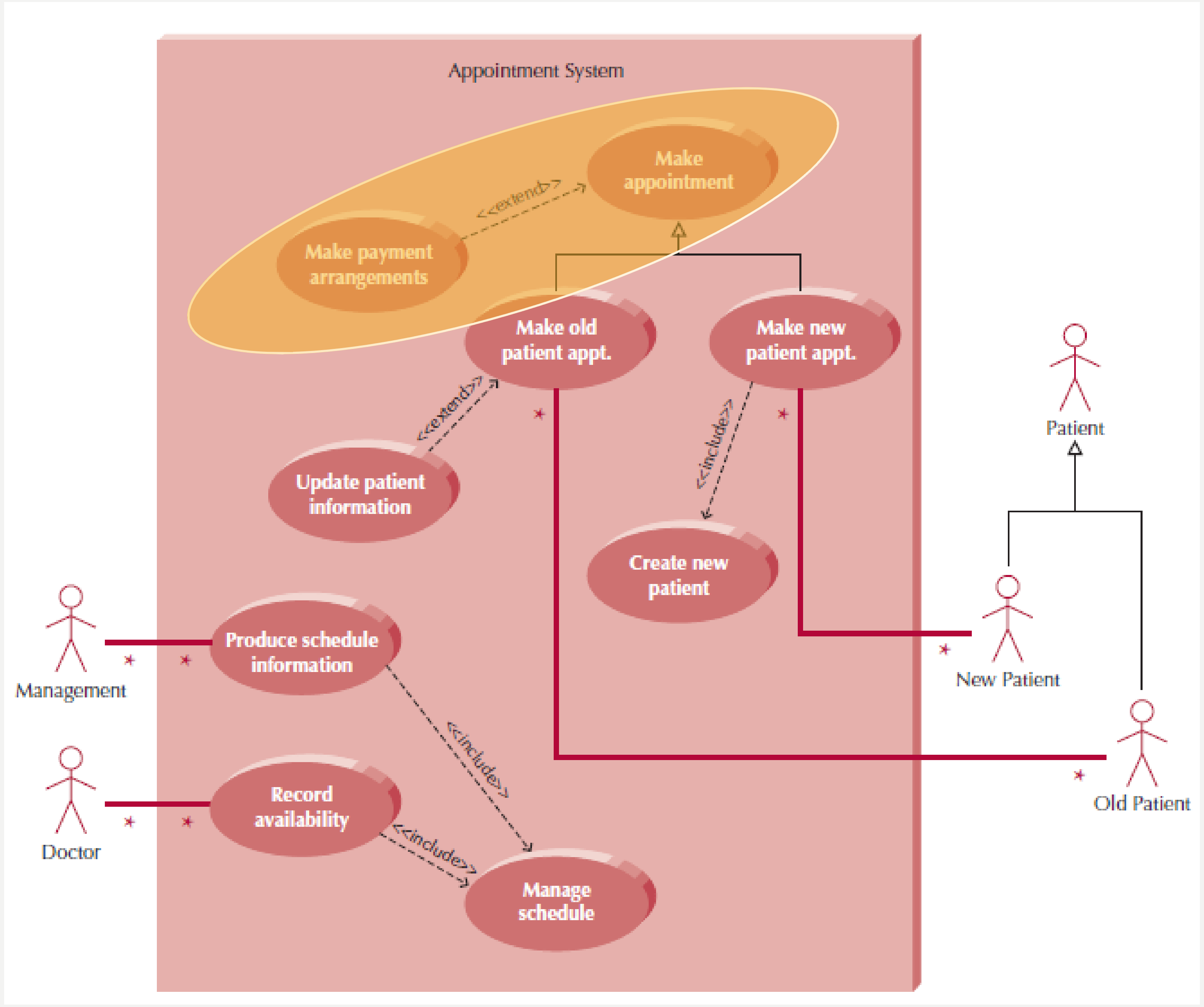
Use Case

- A use case, depicted by an oval in the UML, is a major process that the system will perform that benefits an actor(s) in some way, and it is labeled using a descriptive verb-noun phrase
- There are times when a use case includes, extends, or generalizes the functionality of another use case on the diagram.
- These are shown using include, extend, and generalization relationships.
- To increase the ease of understanding a use case diagram, “higher-level” use cases normally are drawn above the “lower-level” ones.

Use Case Extend Relationship (example)

- Let's assume that every time a patient makes an appointment, the patient is asked to verify payment arrangements.
- However, occasionally it is necessary to actually make new payment arrangements.
- Therefore, we may want to have a use case called Make Payment Arrangements that extends the Make Appointment use case to include this additional functionality.
  - ❑ The Make Payment Arrangements use case was drawn lower than the Make Appointment use case

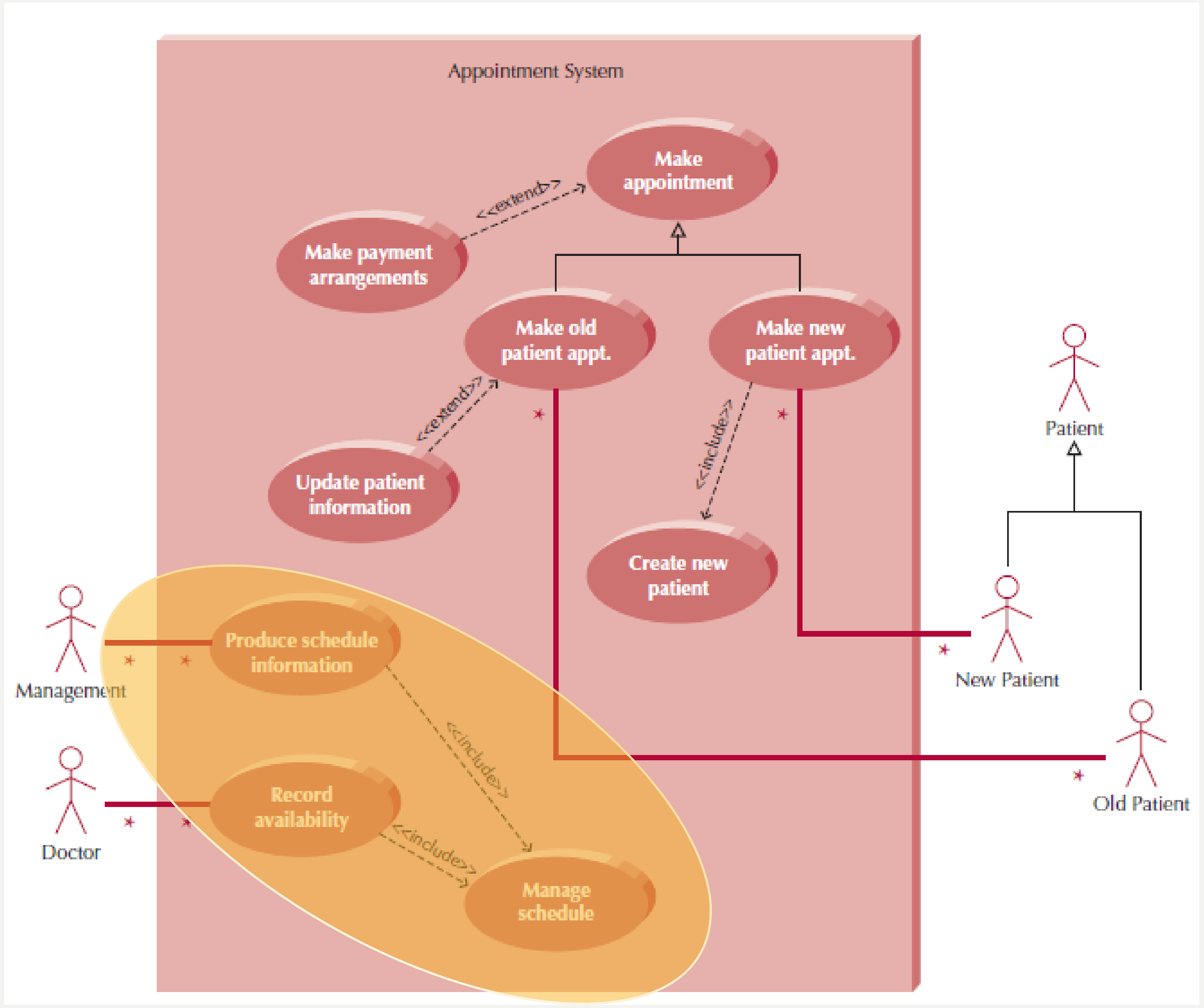
The Use-Case Diagram for Appointment System (Extend Relationships added)



Use Case Include Relationship (example)

- Similarly, there are times when a single use case contains common functions that are used by other use cases.
- For example, suppose there is a use case called Manage Schedule that performs some routine tasks needed to maintain the doctor’s office appointment schedule, and the two use cases Record Availability and Produce Schedule Information both perform the routine tasks
- We can design the system so that Manage Schedule is a shared use case that is used by others.
- An arrow labeled with **include** is used to denote the include relationship and the *included* use case is drawn below the use cases that contain it.

The Use-Case Diagram for Appointment System (Include Relationships added)



Include VS Extend Relationship

- **Misconception** about extend and include relationship
  - **includes are always and extends are sometimes**
  - **Not about always and sometimes, but inclusion and optional behavior**
- **Include Relationship**
  - A **base use case is dependent** on the included use case(s); **without it/them the base use case is incomplete** as the included use case(s) represent sub-sequences of the interaction that may happen always OR sometimes.
  - The base use case knows about (and refers to) the included use case, but the included use case shouldn't 'know' about the base use case.
  - This is why **included use cases can be: a) base use cases in their own right** and **b) shared by a number of base use cases.**

Include VS Extend Relationship

- **Extend Relationship**
  - The extending use case is dependent on the base use case;
  - It literally extends the behavior described by the base use case. **The base use case should be a fully functional use case in its own right** without the extending use case's additional functionality.
  - The base use case represents the "must have" functionality of a project while the **extending use case represents optional** (should/could/want) behavior. This is where the term optional is relevant – optional whether to build/deliver rather than optional whether it sometimes runs as part of the base use case sequence.

http://stackoverflow.com/questions/1696927/whats-is-the-difference-between-include-and-extend-in-use-case-diagram

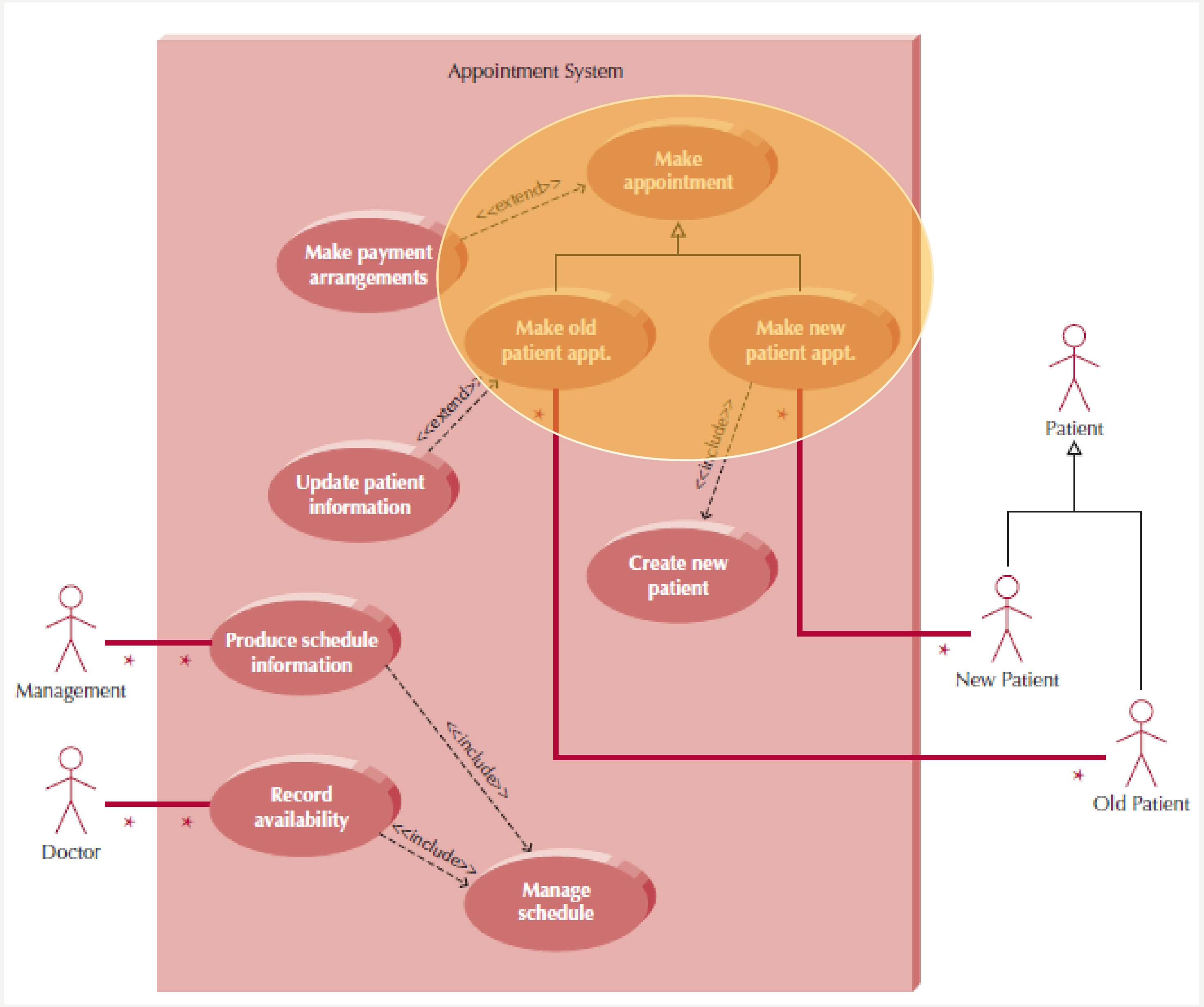
Use Case (Generalization example)

- Finally, there are times that it makes sense to use a generalization relationship to **simplify the individual use cases**.
- For example, the Make Appointment use case has been **specialized** to include a use case for an Old Patient and a New Patient.
- The Make Old Patient Appt use case inherits the functionality of the Make Appointment use case (including the Make Payment Arrangements use case extension) and extends its own functionality with the Update Patient Information use case.
- The Make New Patient Appt use case also inherits all of the functionality of the generic Make Appointment use case and calls the Create New Patient use case, which includes the functionality necessary to insert the new patient into the patient database.

Use Case (Generalization example)

- The generalization relationship is represented as **an unlabeled hollow arrow** with the more general use case being higher than the lower-use cases.
- Also notice that we have consequently added a second specialized actor, Old Patient, and that the Patient actor is now simply a generalization of the Old and New Patient actors.

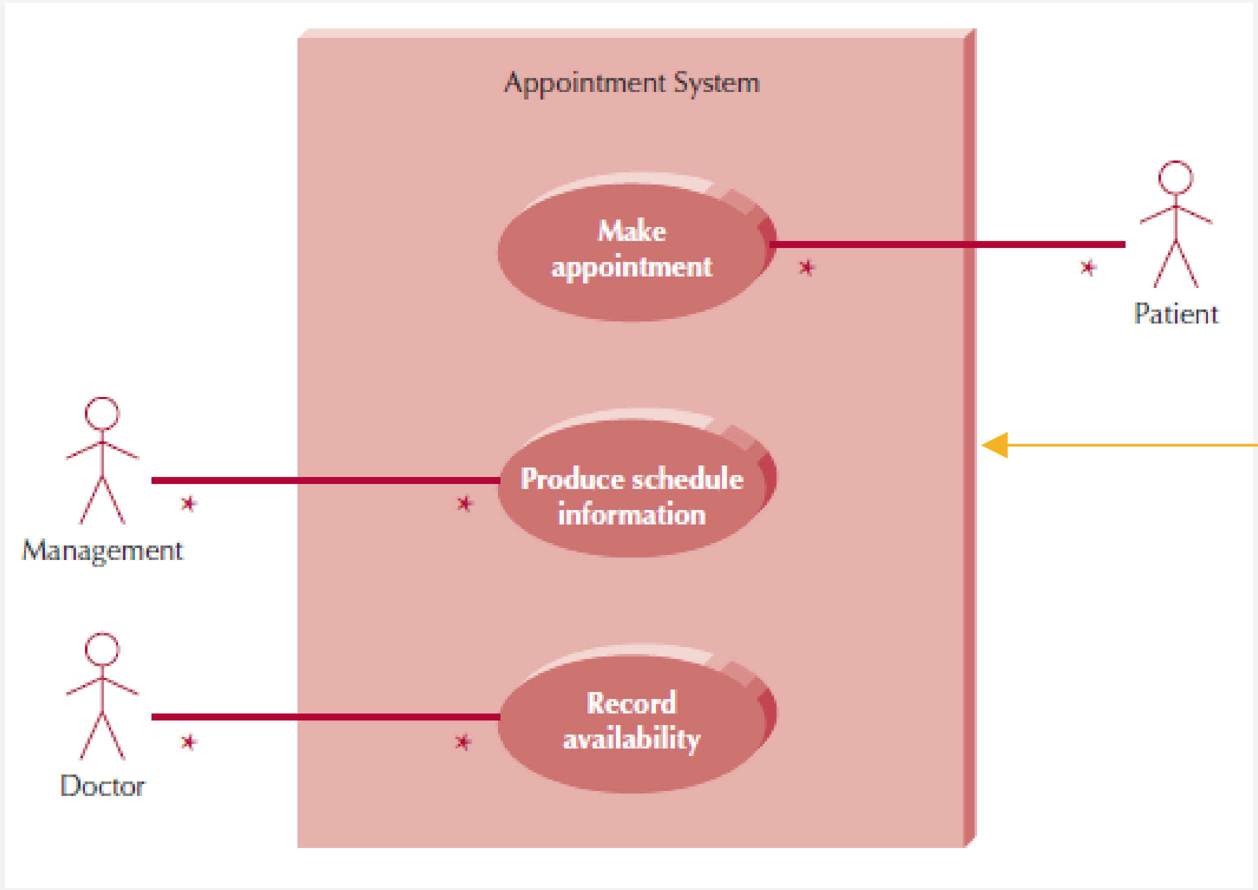
The Use-Case Diagram for Appointment System (Generalization Relationships added)



Subject (System) Boundary

- The use cases are enclosed within a *subject boundary,* which is a box that defines the scope of the system and clearly delineates what parts of the diagram are external or internal to it.
- One of the more difficult decisions to make is where to draw the subject boundary.
  - A subject boundary can be used to separate a software system from its environment, a subsystem from other subsystems within the software system, or an individual process in a software system.
  - They also can be used to separate an information system, including both software and internal actors, from its environment. As such, care should be taken to carefully decide on what the scope of the information system is to include.
- The name of the subject can appear either inside or on top of the box.
- The subject boundary is drawn based on the scope of the system

The Use-Case Diagram for Appointment System



System Boundary

Major Steps in Writing Use Cases & Use-case Diagram

1. **Identifying the Major Use Cases**
   a. Review Requirements Definition
   b. Identify Subject's Boundaries
   c. Identify Primary Actors & Goals
   d. Identify Business Processes & Major Use Cases
   e. Review Current set of Use Cases
2. **Creating a Use-case Diagram**
   a. Place & Draw Use Cases
   b. Place & Draw Actors
   c. Draw Subject Boundary
   d. Add Associations

1.2 BUSINESS PROCESS MODELING WITH ACTIVITY DIAGRAMS (FOR USE CASES)

BPM With Activity Diagrams (1)

- Activity diagram typically used to augment our understanding of the **business processes** and our **use cases** model.
- Technically, an activity diagram can be used for **any type of process-modeling activity.**
  - Can be used to model everything from a **high-level business** workflow that involves many different use cases, **to the details of an individual use case**, all the way down to the specific details of an individual method.
- In this course, we restrict our coverage of activity diagrams to documenting and modeling high-level business processes

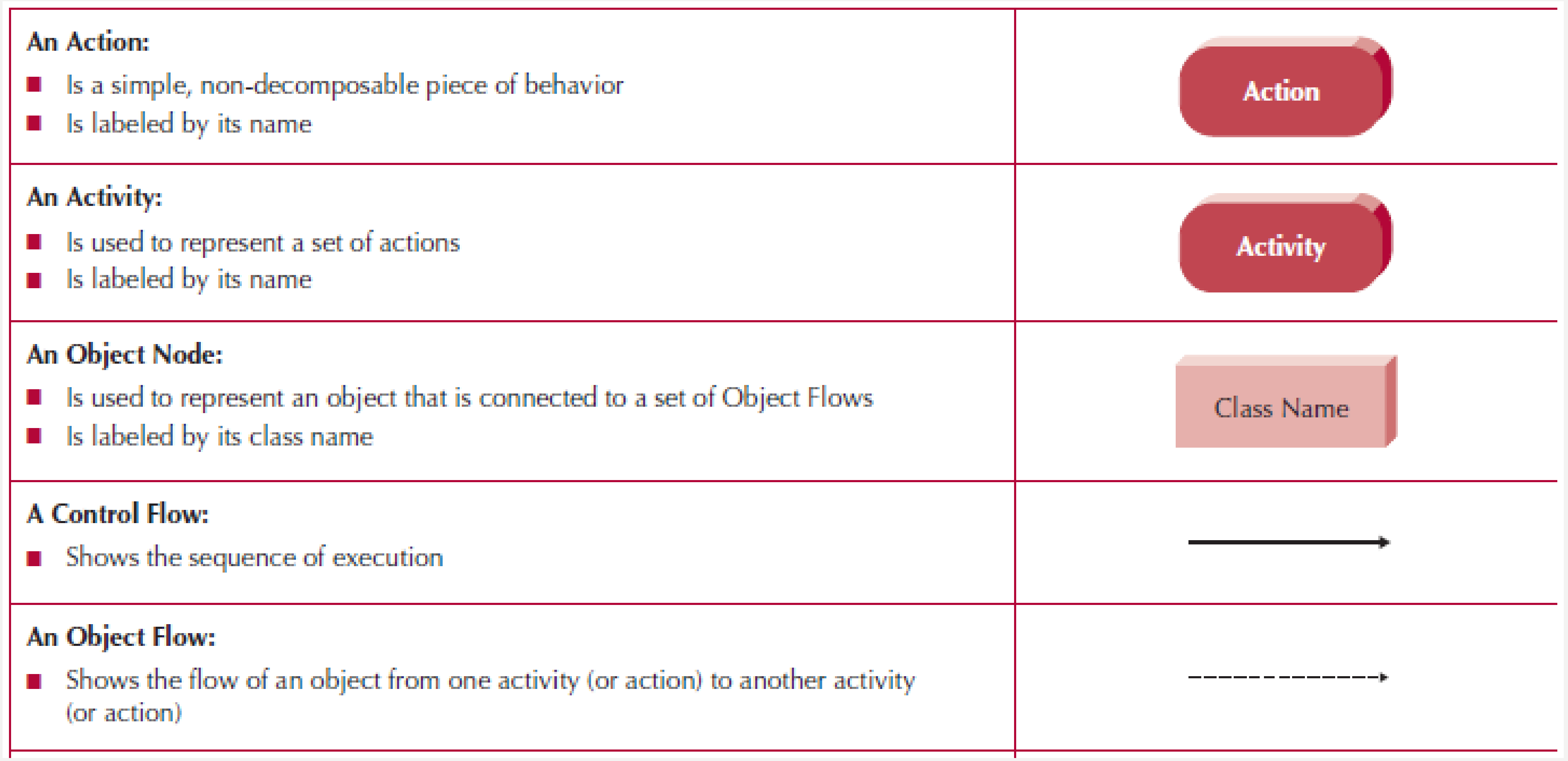
BPM With Activity Diagrams (1)

- Process models depict **how a business system works**
- Process models illustrate the **processes or activities that are performed** and **how objects (data) move among them.**
- Business process models typically **cut across functional departments**
  - For example, the creation of a new product involves many different activities that combine the efforts of many employees in many departments)

Best practices when modeling business processes from Martin Schedlbauer

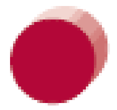
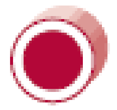
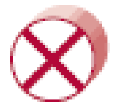
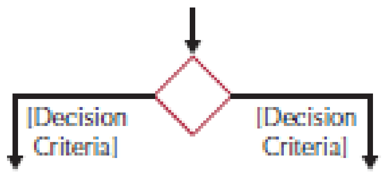
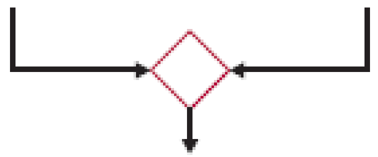
- Be realistic → imposible to identify everything, everything is not equally important
- Be agile
- All modeling is a **collaborative**/social activity
- Do not use a CASE tool, use **whiteboards** instead
- Should be done in an **iterative** manner
- Stay focused on a specific process, if tasks associated with other business processes are identified, simply record them on a to-do list thing and get back to the business process you are currently modeling
- A business process model is an **abstraction of reality**, you should not include every minor task.

Elements of an activity diagram

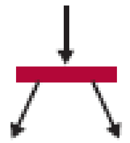
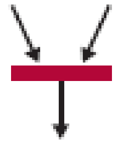
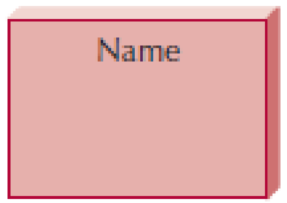


An Action: ▪ Is a simple, non-decomposable piece of behavior ▪ Is labeled by its name	Action
An Activity: ▪ Is used to represent a set of actions ▪ Is labeled by its name	Activity
An Object Node: ▪ Is used to represent an object that is connected to a set of Object Flows ▪ Is labeled by its class name	Class Name
A Control Flow: ▪ Shows the sequence of execution	⟶
An Object Flow: ▪ Shows the flow of an object from one activity (or action) to another activity (or action)	⇢

Elements of an activity diagram

An Initial Node: ■ Portrays the beginning of a set of actions or activities	
A Final-Activity Node: ■ Is used to stop all control flows and object flows in an activity (or action)	
A Final-Flow Node: ■ Is used to stop a specific control flow or object flow	
A Decision Node: ■ 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: ■ Is used to bring back together different decision paths that were created using a decision-node	

Elements of an activity diagram

A Fork Node: ■ Is used to split behavior into a set of parallel or concurrent flows of activities (or actions)	
A Join Node: ■ Is used to bring back together a set of parallel or concurrent flows of activities (or actions)	
A Swimlane: ■ 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	

Elements of an activity diagram

- **Actions & Activities**
  - Actions & activities are performed for some specific business reason
  - Can represent manual or computerized behavior
  - Should have a name that begins with **a verb and ends with a noun**
  - An activity **can be further decomposed** further into a set of activities and/or actions
  - An action represents a **simple non-decomposable** piece of the overall behavior being modeled

Elements of an activity diagram

- **Object Nodes**
  - Activity and actions typically modify or transform objects; object nodes model these objects.
  - Essentially, object nodes represent the flow of information from one activity to another activity
- **Control Flows & Object Flows**
  - Control flows model the **paths of execution** through a business process
    - Control flows can be attached only to actions or activities
  - Object Flows model the **flows of objects** through a business process
    - Object flows are necessary to show the actual objects that flow into and out of the actions or activities

Elements of an activity diagram

- **Control Nodes (7 types):**
  1. Initial nodes: portray the beginning of a set of actions or activities
  2. Final-activity nodes: used to stop the process being modeled (regardless of whether they are completed)
  3. Final-flow nodes: 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
  4. Decision nodes: used to represent the actual test condition that determines which of the paths exiting the decision node is to be traversed; each exiting path must be labeled with a guard condition

Elements of an activity diagram

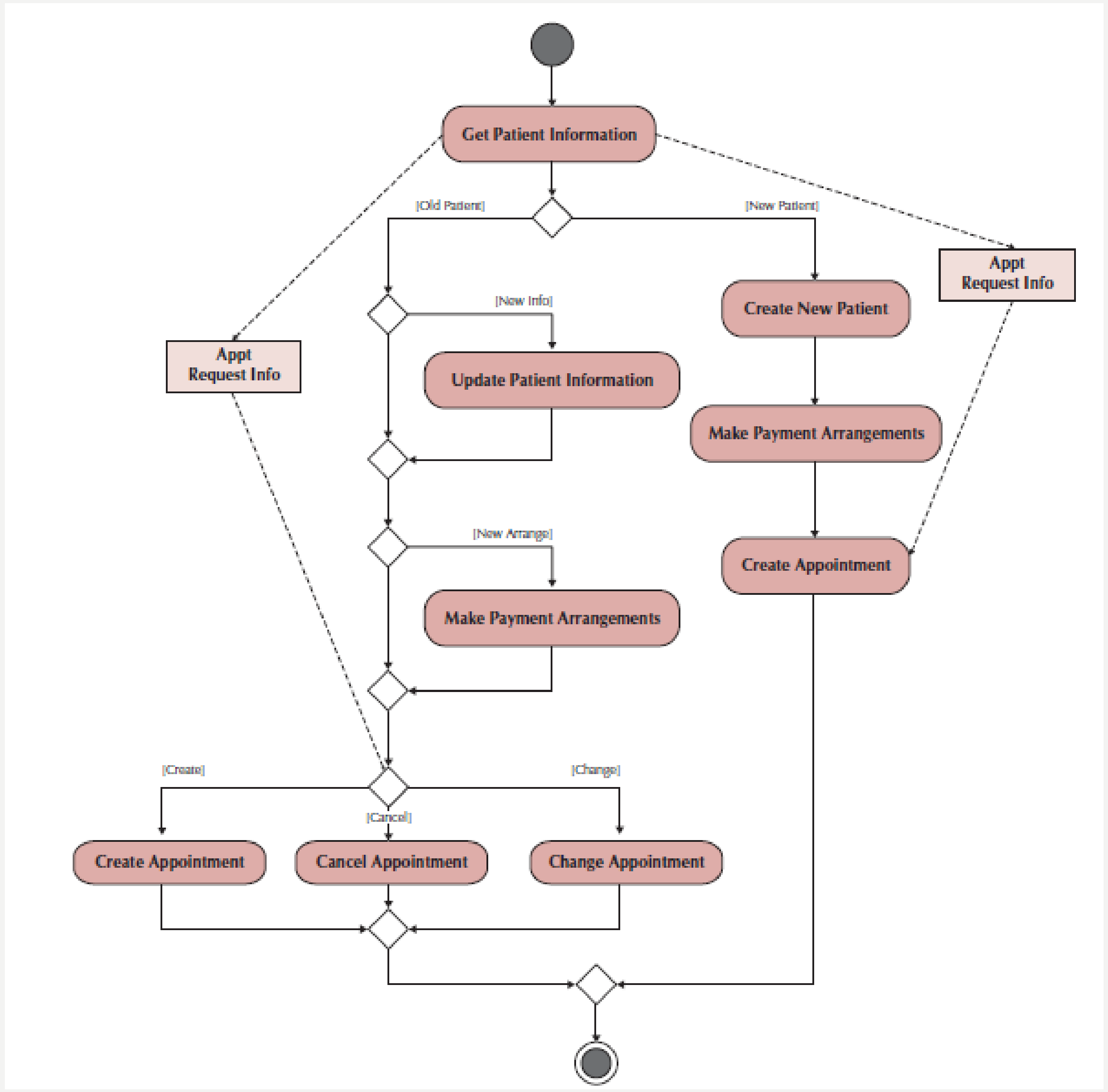
- **Control Nodes (7 types):**

5. Merge nodes: used to bring back together multiple mutually exclusive paths that have been split based on an earlier decision
6. Fork nodes: used to split the behavior of the business process into multiple parallel or concurrent flows
7. Join nodes: simply brings back together the separate parallel or concurrent flows in the business process into a single flow

Elements of an activity diagram

- **Swim lanes**
  - When modeling a business workflow, it is especially beneficial to break up an activity diagram in a manner that is useful in **assigning responsibility to objects or individuals that would actually perform the activity**
  - You could also draw the activity diagram using more of a **left-to-right orientation** instead of a top-down orientation
  - In an actual business workflow, we would see that we had activities that should be associated with roles of individuals that are involved in the business workflow (e.g., employees or customers) and the activities that were to be accomplished by the information system that was being created.
    - This association of activities with external roles, internal roles, and the system is very useful when creating the use case descriptions and diagram described later

Activity Diagram Example



Guidelines for Creating Activity Diagrams from Scott Ambler

1. Since 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.
2. You must identify **the activities, control flows, and object flows** that occur between the activities.
3. You should identify **any decisions** that are part of the process being modeled.
4. You should attempt to identify any **prospects for parallelism** in the process.
5. You should draw the activity diagram.

Major Steps in Creating an Activity Diagram

1. Choose a business process
2. Identify activities
3. Identify control flows & nodes
4. Identify object flows & nodes
5. Lay out & draw diagram

Your turn: Create an Activity Diagram for Borrow Books (Use Case) ☺

The borrowing activities are built around checking books out and returning books by borrowers. There are three types of borrowers: students, faculty or staff, and guests. Regardless of the type of borrower, the borrower must have a valid ID card. If the borrower is a student, having the system check with the registrar's student database validates the ID card. If the borrower is a faculty or staff members, having the system check with the personnel office's employee database validates the ID card. If the borrower is a guest, the ID card is checked against the library's own borrower database. If the ID card is valid, the system must also check to determine whether the borrower has any overdue books or unpaid fines. If the ID card is invalid, the borrower has overdue books, or the borrower has unpaid fines, the system must reject the borrower's request to check out a book, otherwise the borrower's request should be honored.

1.3
USE CASES &
USE-CASE
DESCRIPTIONS

Introduction

- Use cases are the primary drivers for all the UML diagramming techniques. A use case communicates at a high level what the (Computer-based Information) system needs to do, and all the UML diagramming techniques build on this by presenting the use –case functionality in a different way for a different purpose.
- Use cases are the building blocks by which the system is designed and built.

Introduction

- Use-case diagrams provide a **bird’s-eye view of the basic functionality** of the business processes contained in the evolving system.
- Activity diagrams open up the **black box of each business process** by providing a more-detailed graphical view of the underlying activities that support each business process.

Use-Case Descriptions

- provide a means to more fully document the different aspects of each individual use case.
- are based on the identified requirements, use-case diagram and the activity diagram.
- contain all the information needed to document the functionality of the business processes.

As to which should come first—use case descriptions or use case diagram—technically speaking, it really does not matter. Both should be done to fully describe the requirements that the information system must meet.

Use Cases

- Use cases capture the typical interaction of the system with the system’s users (end users and other systems).
- These interactions represent the **external or functional view** of the system from the perspective of the user.
- Each use case describes **one and only one function in which users interact with the system**, although a use case may contain several “paths” that a user can take while interacting with the system (e.g., when searching for a book in Web bookstore, the user might search by subject, by author, or by title).
- Each path through the use case is referred to as a *scenario*.

How Are Use Cases and Use-Case Description Created?

- Project team must work closely with the users to fully document the functional requirements.
- Organizing the functional requirements and documenting them in a use-case description are a relatively simple process, but it takes considerable practice to ensure that the description are complete enough to use in the next structural and behavioral modeling.
- The best place to begin is to review the use-case and activity diagrams.
- The key thing to remember is that each use case is associated with one and only one role that users have in the system

Types of Use Cases

1. Overview versus detail
2. Essential versus real

Overview versus Detail Use Case

- Used to enable the analyst and user to agree on a high-level overview of the requirements.
- Typically, they are created very early in the process of understanding the system requirements, and they only document basic information about the use case such as its **name, ID number, primary actor, type, and a brief description.**
- Once the user and the analyst agree upon a high-level overview of the requirements, the overview use cases can be converted to **detail use cases**.
- A *detail use case* typically **documents, as far as possible, all of the information needed for the use case.**

Essential versus Real Use Case

- An esssential use case only describes the **minimum essential issues necessary to understand the required functionality.**
- A *real use case* will **go further and describe a specific set of steps.**
- For example, an **essential use case** in a dentist office might say that the **receptionist should attempt to "Match the Patient's desired appointment times with the available times,"** while a **real use case** might say that the receptionist should "**Look up the available dates on the calendar using MS Exchange to determine if the requested appointment times were available.**"

Essential versus Real Use Case

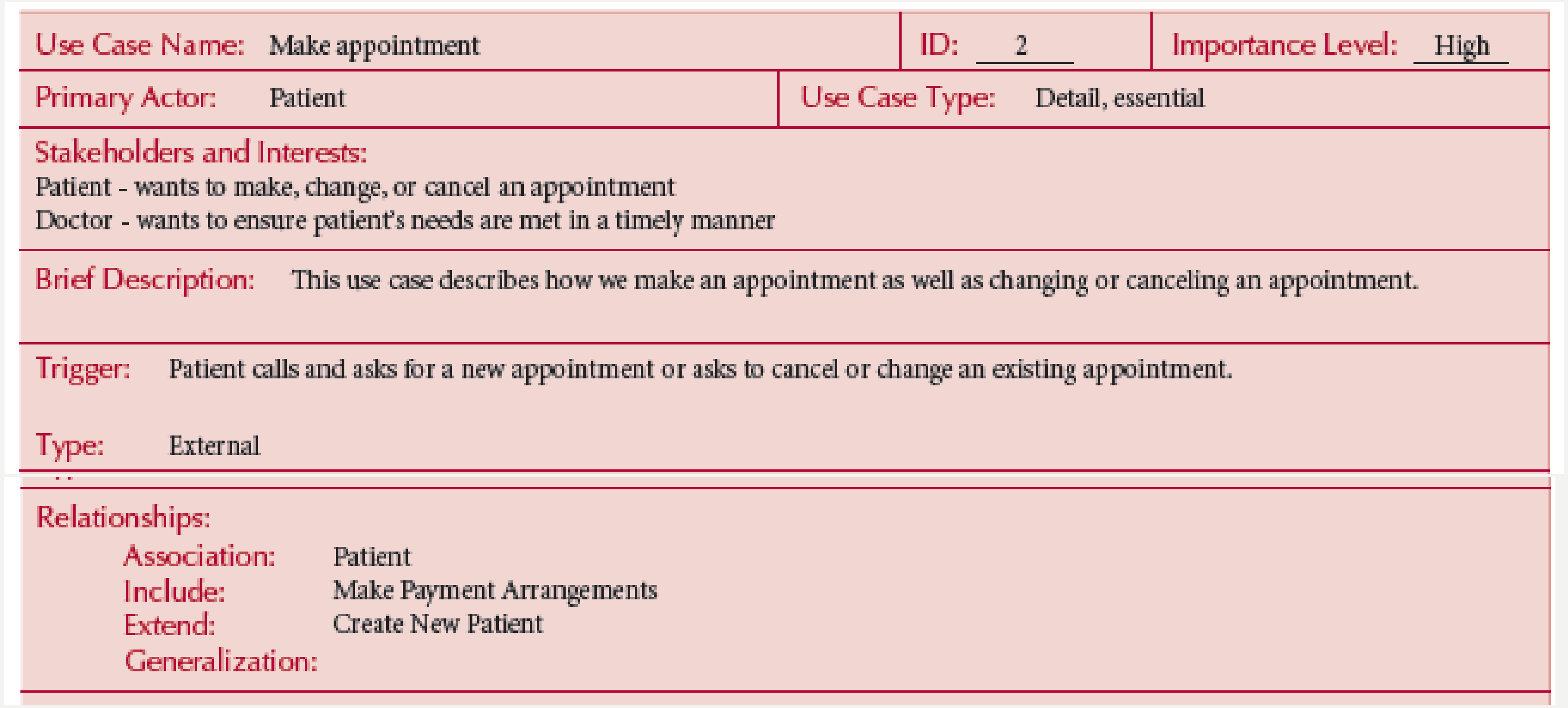
- The primary difference is that essential use cases are implementation independent, while real use cases are detailed descriptions of how to use the system once it is implemented.
- As such, **real use cases tend to be used only in detailed design, implementation, and testing.**

Elements of a Use-case Description

There are three basic parts to a use case description:

1. overview information,
2. relationships, and
3. the flow of events

Overview & Relationships (example)



Use Case Name: Make appointment	ID: 2	Importance Level: High
Primary Actor: Patient	Use Case Type: Detail, essential	

Stakeholders and Interests:
Patient - wants to make, change, or cancel an appointment
Doctor - wants to ensure patient's needs are met in a timely manner

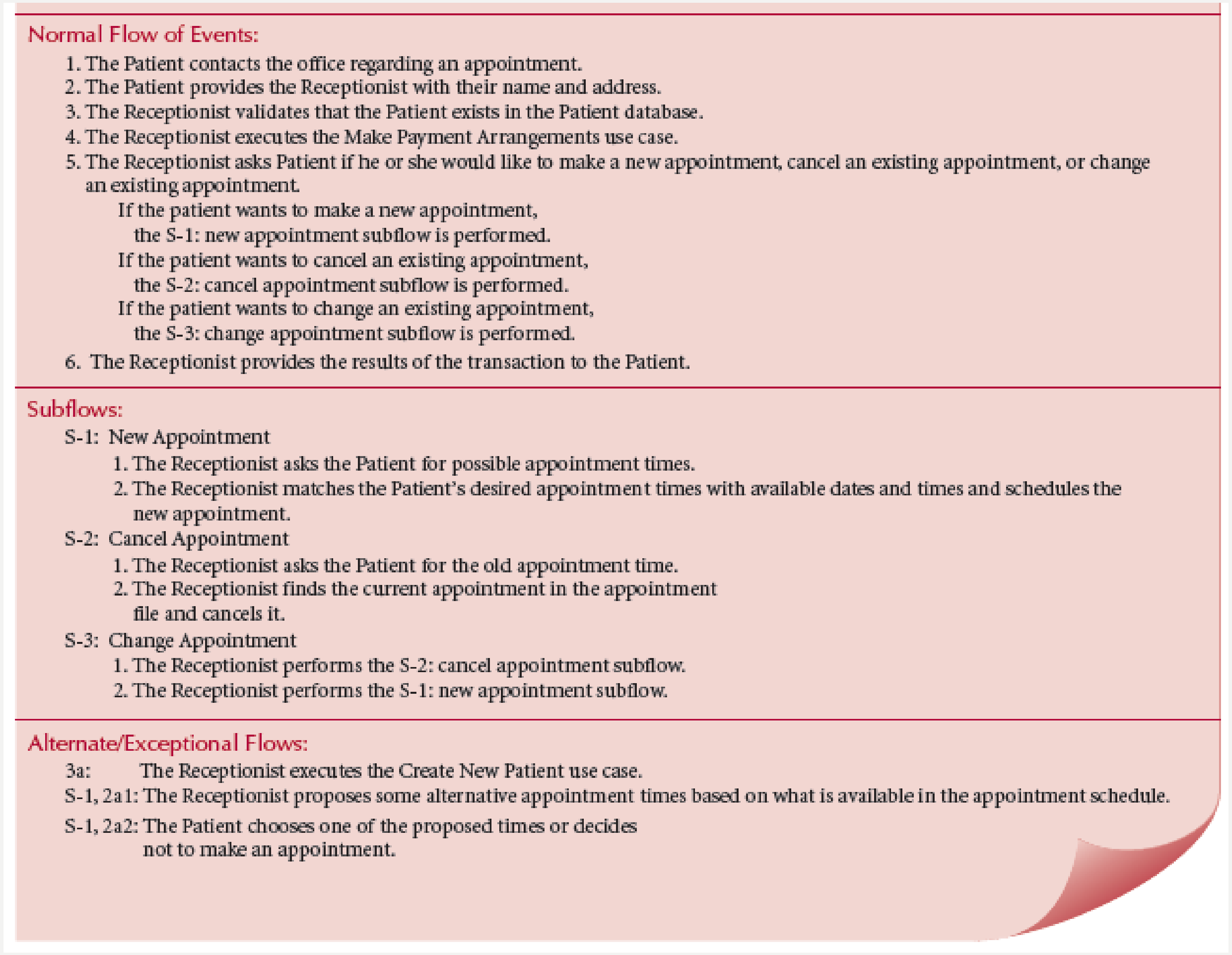
Brief Description: This use case describes how we make an appointment as well as changing or canceling an appointment.

Trigger: Patient calls and asks for a new appointment or asks to cancel or change an existing appointment.

Type: External

Relationships:

Association:	Patient
Include:	Make Payment Arrangements
Extend:	Create New Patient
Generalization:	



Normal Flow of Events:

1. The Patient contacts the office regarding an appointment.
2. The Patient provides the Receptionist with their name and address.
3. The Receptionist validates that the Patient exists in the Patient database.
4. The Receptionist executes the Make Payment Arrangements use case.
5. The Receptionist asks Patient if he or she would like to make a new appointment, cancel an existing appointment, or change an existing appointment.
   - If the patient wants to make a new appointment,
     the S-1: new appointment subflow is performed.
   - If the patient wants to cancel an existing appointment,
     the S-2: cancel appointment subflow is performed.
   - If the patient wants to change an existing appointment,
     the S-3: change appointment subflow is performed.
6. The Receptionist provides the results of the transaction to the Patient.

Subflows:

S-1: New Appointment
1. The Receptionist asks the Patient for possible appointment times.
2. The Receptionist matches the Patient's desired appointment times with available dates and times and schedules the new appointment.

S-2: Cancel Appointment
1. The Receptionist asks the Patient for the old appointment time.
2. The Receptionist finds the current appointment in the appointment file and cancels it.

S-3: Change Appointment
1. The Receptionist performs the S-2: cancel appointment subflow.
2. The Receptionist performs the S-1: new appointment subflow.

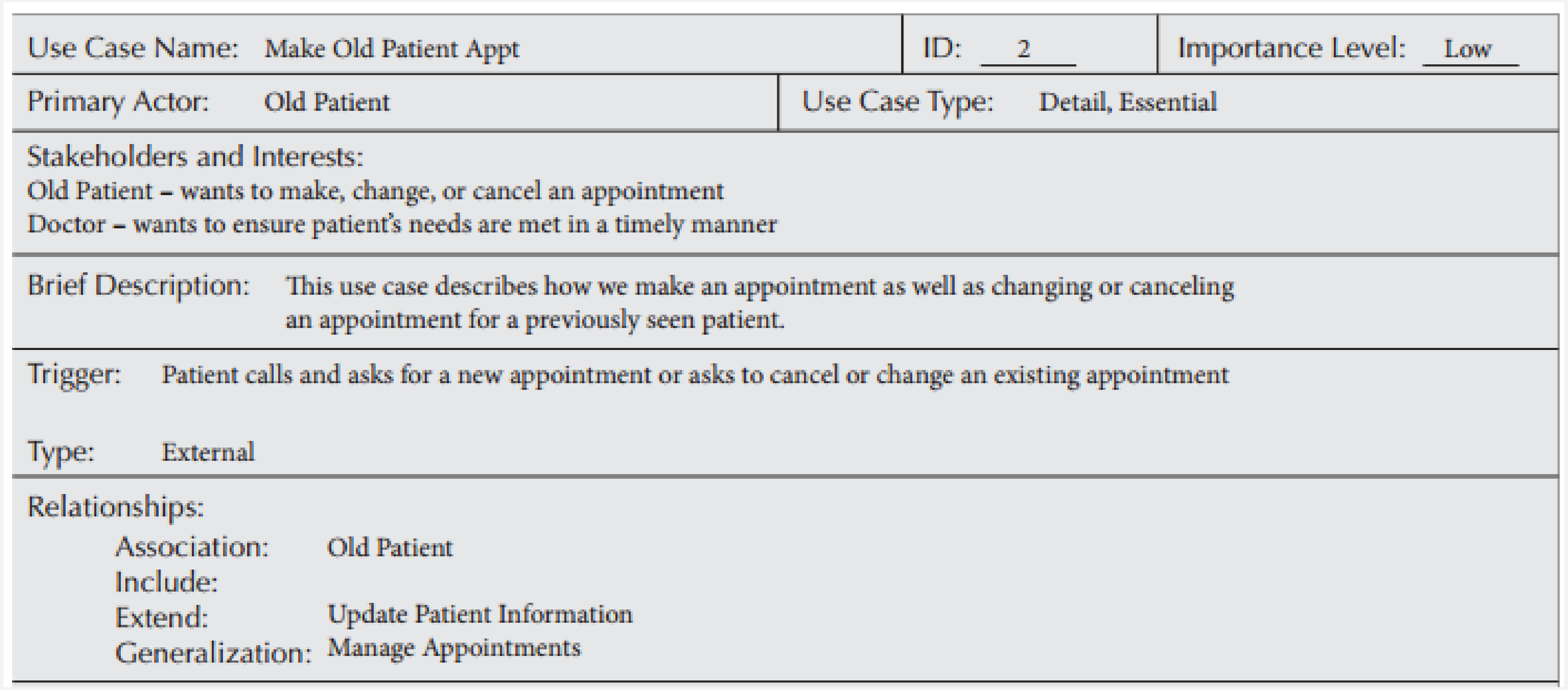
Alternate/Exceptional Flows:

3a: The Receptionist executes the Create New Patient use case.

S-1, 2a1: The Receptionist proposes some alternative appointment times based on what is available in the appointment schedule.

S-1, 2a2: The Patient chooses one of the proposed times or decides not to make an appointment.

Overview & Relationships (example)



Use Case Name: Make Old Patient Appt	ID: 2	Importance Level: Low
Primary Actor: Old Patient	Use Case Type: Detail, Essential	

Stakeholders and Interests:
Old Patient – wants to make, change, or cancel an appointment
Doctor – wants to ensure patient's needs are met in a timely manner

Brief Description: This use case describes how we make an appointment as well as changing or canceling an appointment for a previously seen patient.

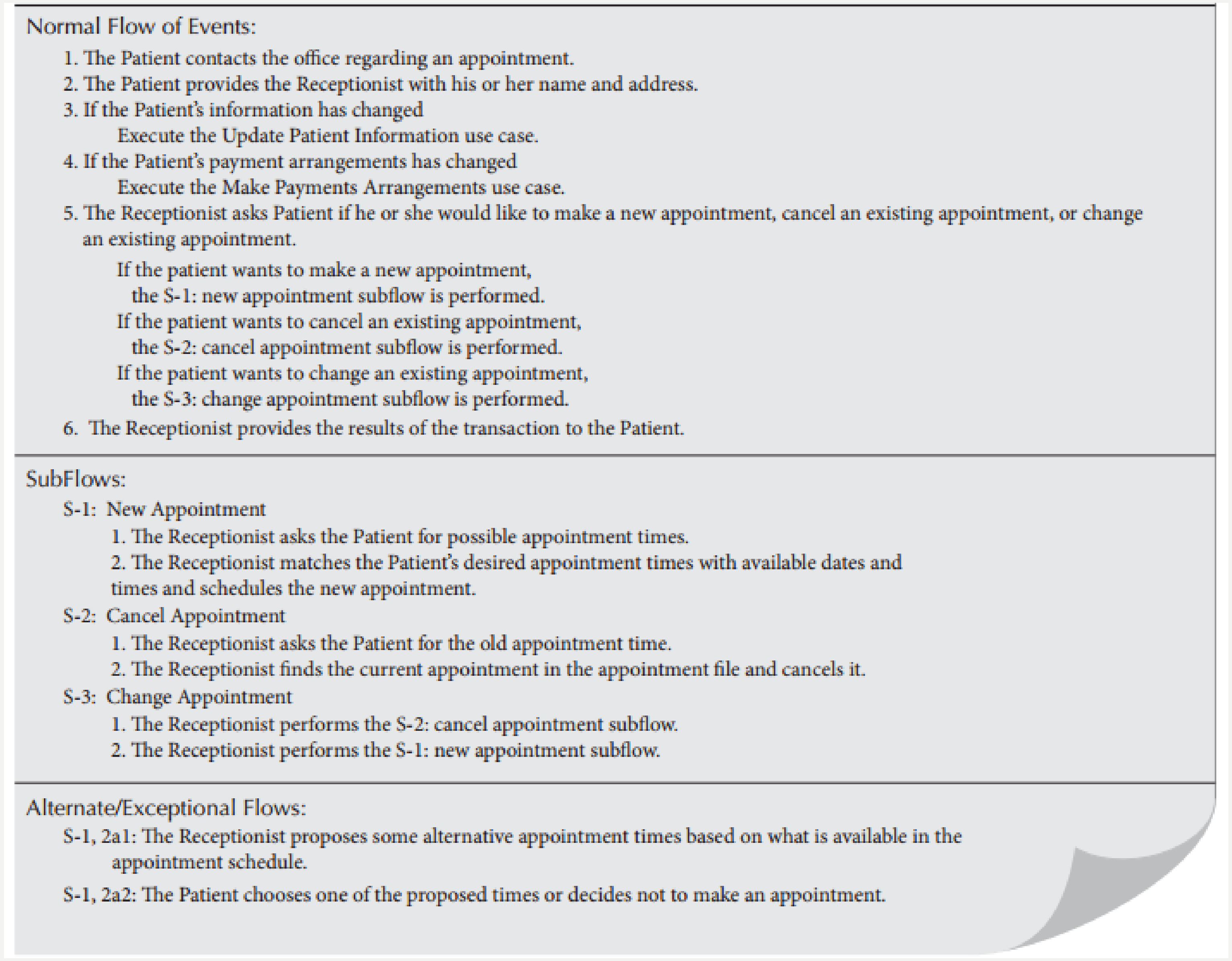
Trigger: Patient calls and asks for a new appointment or asks to cancel or change an existing appointment

Type: External

Relationships:

Association:	Old Patient
Include:	
Extend:	Update Patient Information
Generalization:	Manage Appointments

Overview & Relationships (example)



Normal Flow of Events:

1. The Patient contacts the office regarding an appointment.
2. The Patient provides the Receptionist with his or her name and address.
3. If the Patient's information has changed
   Execute the Update Patient Information use case.
4. If the Patient's payment arrangements has changed
   Execute the Make Payments Arrangements use case.
5. The Receptionist asks Patient if he or she would like to make a new appointment, cancel an existing appointment, or change an existing appointment.
   If the patient wants to make a new appointment,
   the S-1: new appointment subflow is performed.
   If the patient wants to cancel an existing appointment,
   the S-2: cancel appointment subflow is performed.
   If the patient wants to change an existing appointment,
   the S-3: change appointment subflow is performed.
6. The Receptionist provides the results of the transaction to the Patient.

SubFlows:

S-1: New Appointment
1. The Receptionist asks the Patient for possible appointment times.
2. The Receptionist matches the Patient's desired appointment times with available dates and times and schedules the new appointment.

S-2: Cancel Appointment
1. The Receptionist asks the Patient for the old appointment time.
2. The Receptionist finds the current appointment in the appointment file and cancels it.

S-3: Change Appointment
1. The Receptionist performs the S-2: cancel appointment subflow.
2. The Receptionist performs the S-1: new appointment subflow.

Alternate/Exceptional Flows:

S-1, 2a1: The Receptionist proposes some alternative appointment times based on what is available in the appointment schedule.

S-1, 2a2: The Patient chooses one of the proposed times or decides not to make an appointment.

Guideline for creating Use-case Descriptions

1. Choose a Use Case
2. Create Overview Description
3. Describe to Normal Flow of Events
4. Check the Normal Flow of Events
5. Identify Alternative or Exceptional Flows
6. Review the Use-Case Description
7. Repeat Until Done

Guideline for creating Use-case Descriptions

1. Write each step in "SVDPI" form (subject-verb-direct object, and optionally, preposition-indirect object)
2. Clarify initiator and receivers of action
3. Write from independent observer perspective
4. Write at same level of abstraction
5. Ensure a sensible set of steps
6. Apply KISS principle liberally
7. Write repeating instructions after the set of steps to be repeated

[1] These guidelines are based on Cockburn, *Writing Effective Use Cases* and Graham, *Migrating to Object Technology*

Your Turn ☺

How would you make requirements gathering (interviews, questionnaires, observation, and document analysis) more effective by knowing that eventually you will be creating use-case descriptions and diagrams?

1.4 VERIFYING & VALIDATING THE BUSINESS PROCESSES AND FUNCTIONAL MODELS

Introduction

- Before move on to the next modeling, i.e., structural and behavioral modeling, we need to verify and validate the current set of functional models to ensure that they faithfully represent business processes under consideration.
- This includes testing the fidelity of each model; i.e., we must ensure that the **activity diagram(s), use-case descriptions, and use case diagram, all describe the same functional requirements.**
- Before we describe the specific tests, we describe walkthroughs, a manual approach that supports verifying and validating the evolving models.

Walkthroughs

- Essentially a peer review of a product.
- Typically **completed by a team** whose members come from the development team and the client.
- The purpose is to thoroughly test the fidelity and to ensure that **the models are consistent** → uncover errors or faults in the evolving specification
- However, a walkthrough does not correct errors-it simply identifies them.
- Error correction is to be accomplished by the team after the walkthrough is completed

Walkthroughs

- Specified roles of the walkthrough team:
    1. Presenter role
    2. Recorder/scribe
    3. Maintenance oracle
    4. Organizer (calling, setting up, and running the walkthrough meetings)

Potential Danger of Walkthroughs

- When management decides **the results of uncovering errors** in the representation **are a reflection of an analyst’s capability.**
- This must be avoided at all costs.
- Depending on the organization, it may be necessary to omit management from the walkthrough process.

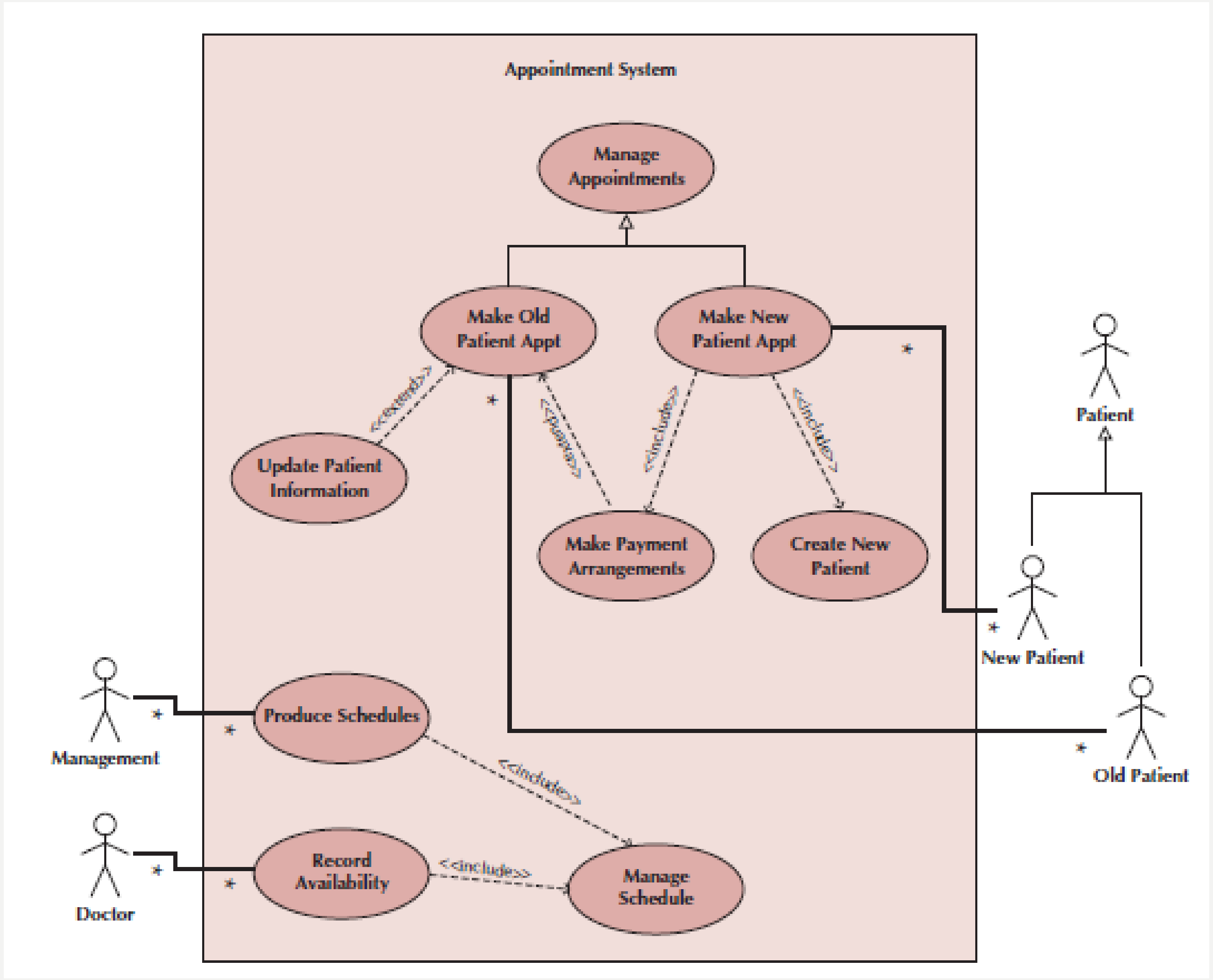
Functional Model Verification & Validation (step 1-7)

1. When comparing an **activity diagram** to a **use case description**, there should be at least one event recorded in the normal flow of events, sub-flows, or alternative/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action
2. All objects portrayed as an object node in an activity must be mentioned in an event in the normal flow of events, sub-flows, or alternative/exceptional flows of the use-case description.
3. **Sequential order** of the events in a use-case description should occur in **the same sequential order** of the activities contained **in an activity diagram**

Functional Model Verification & Validation (step 1-7)

4. When comparing a use-case description to a use-case diagram, there must be **one and only one use-case description for each use case**, and vice versa
5. All **actors** listed in a use-case description must be portrayed on the use-case diagram
6. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.
7. All other relationships listed in a use-case description **(include, extend, and generalization)** must be portrayed on a use-case diagram.

Revised Use Case Diagram



1.4
USE CASES & PROJECT MANAGEMENT (ADDITIONAL)

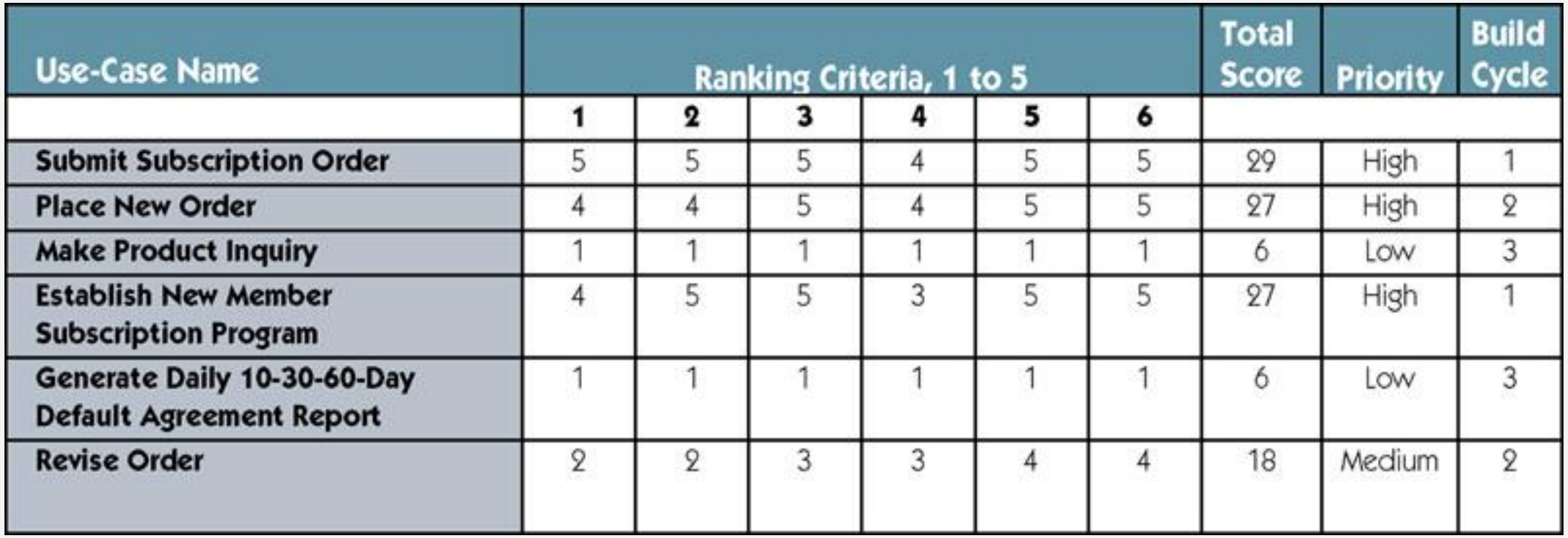
Use Cases and Project Management

- Use-case model can drive entire development effort.
- Project manager or system analyst uses business requirements **use cases to estimate and schedule the build cycles of the project.**
  - Build cycles are scoped on the basis of the importance of the use case and the time it takes to implement the use case.
- To determine importance of use cases, system analyst can create:
  - **Use-case ranking and evaluation matrix**
  - **Use-case dependency diagram**

Use-Case Ranking and Priority Matrix

- In most projects, **the most important use cases are developed first.**
- **Use-case ranking and priority matrix** – a tool used to evaluate use cases and determine their priority.
- **Evaluates use cases on 1-5 scale against six criteria.**
  1. Significant impact on the architectural design.
  2. Easy to implement but contains significant functionality.
  3. Includes risky, time-critical, or complex functions.
  4. Involves significant research or new or risky technology.
  5. Includes primary business functions.
  6. Will increase revenue or decrease costs.

Sample Use-Case Ranking and Priority Matrix



Use-Case Name	Ranking Criteria, 1 to 5						Total Score	Priority	Build Cycle
	1	2	3	4	5	6			
Submit Subscription Order	5	5	5	4	5	5	29	High	1
Place New Order	4	4	5	4	5	5	27	High	2
Make Product Inquiry	1	1	1	1	1	1	6	Low	3
Establish New Member Subscription Program	4	5	5	3	5	5	27	High	1
Generate Daily 10-30-60-Day Default Agreement Report	1	1	1	1	1	1	6	Low	3
Revise Order	2	2	3	3	4	4	18	Medium	2

Summary

- **Activity Diagrams**
  - Understand the rules and style guidelines for activity diagrams
  - Activity diagrams can be used to represent (To-Be) Business Process Model of the IS
- **Use cases (Use-case diagram and use-case descriptions)**
  - Understand the rules and style guidelines for use cases (use-case diagram and use-case descriptions)
  - Understand the process used to create use cases (use-case diagram and use-case descriptions)

Summary

- Create, Verify & Validate the **functional models of (To-Be) IS** using Use Cases via activity diagrams, use-case diagram and use-case descriptions
- (Additional) Use Case Ranking and Prioritization Matrix for Project Management (Estimate)

References

- Systems Analysis and Design: An Object Oriented Approach with UML 5th ed. Alan Dennis, Barbara Haley Wixom, and Roberta M. Roth © 2015
- http://dilbert.com/strip/2000-02-27