جامعة الملك سعود
King Saud University
College of Computer and Information Sciences

System Modeling

Section 5.2 (Sommerville)

Fall Semester 2021

1st Semester 1443 H

System modeling

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

- System modeling has now come to mean representing a system using a graphical notation (UML - Unified Modeling Language).

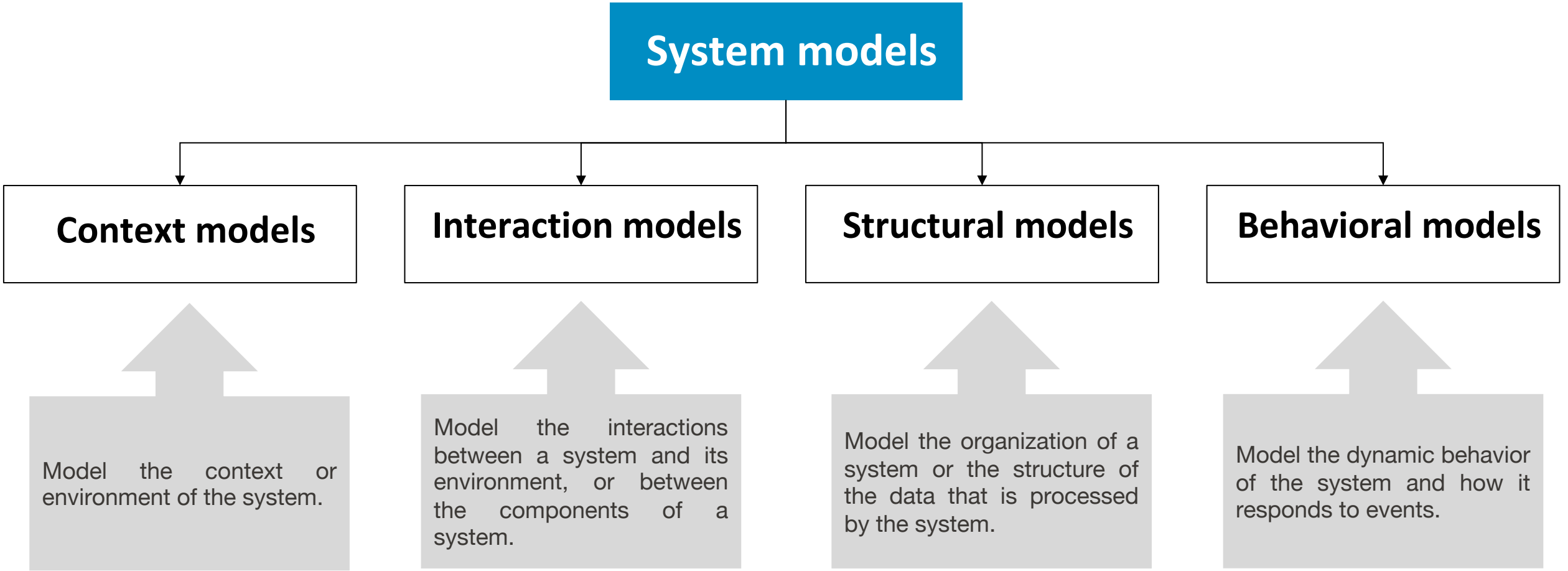
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

System modeling

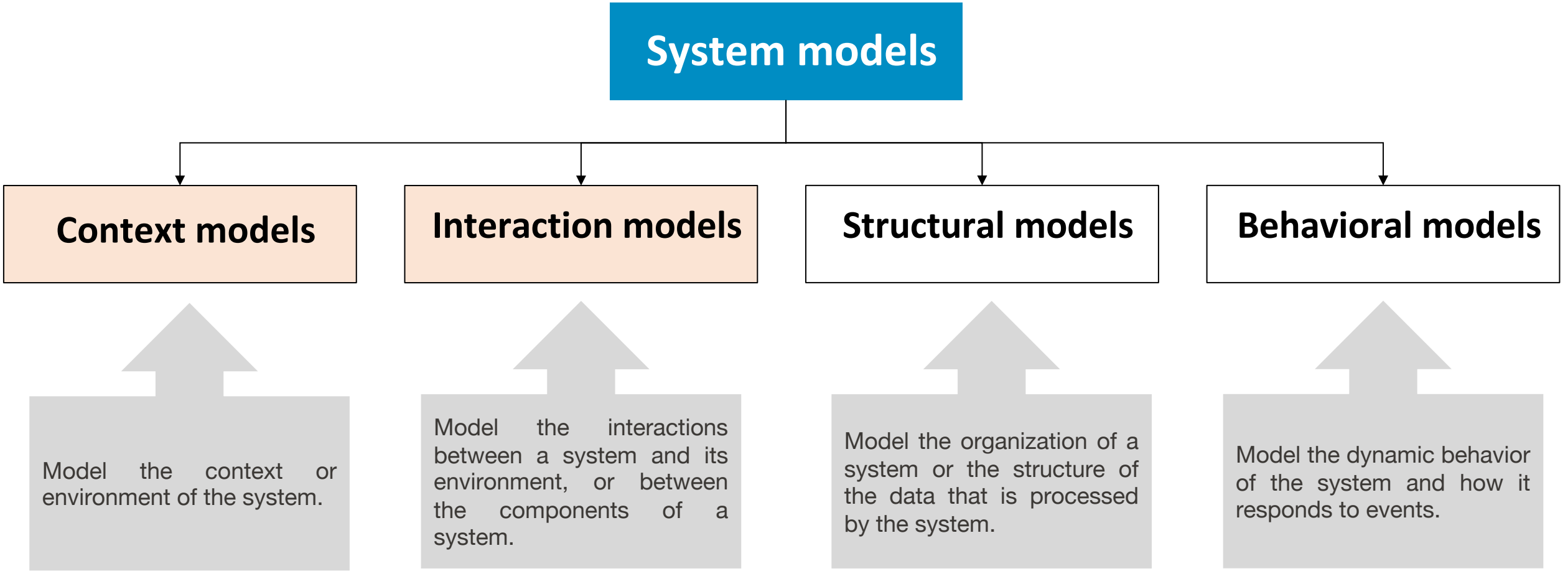
System models are used during requirements engineering to

- Clarify what the existing system does.
- Explain the proposed requirements to the stakeholders.
- Discuss design proposals.
- Document the system for implementation.

System modeling



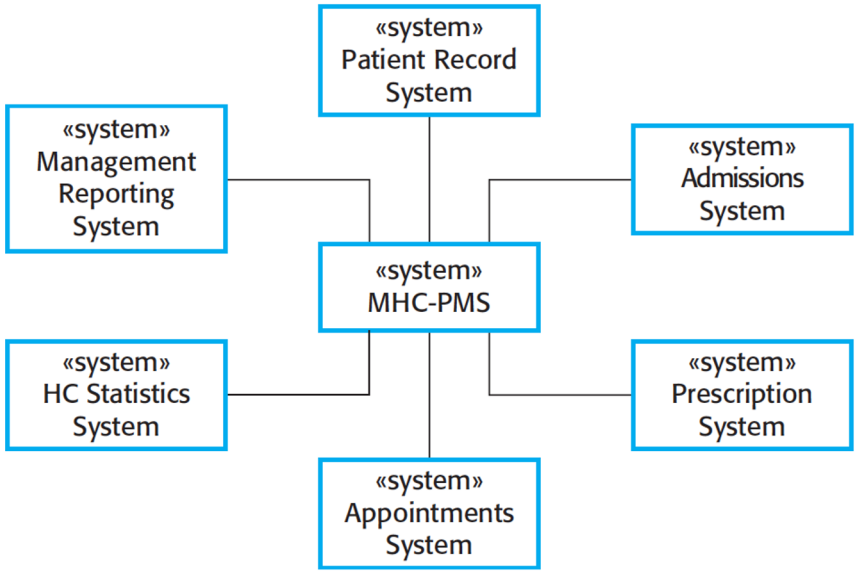
System modeling



Context models

- Illustrate the operational context of a system.
  - Show what lies outside the system boundaries.
  - Show the system and its relationship with other systems.
  - Architectural models.

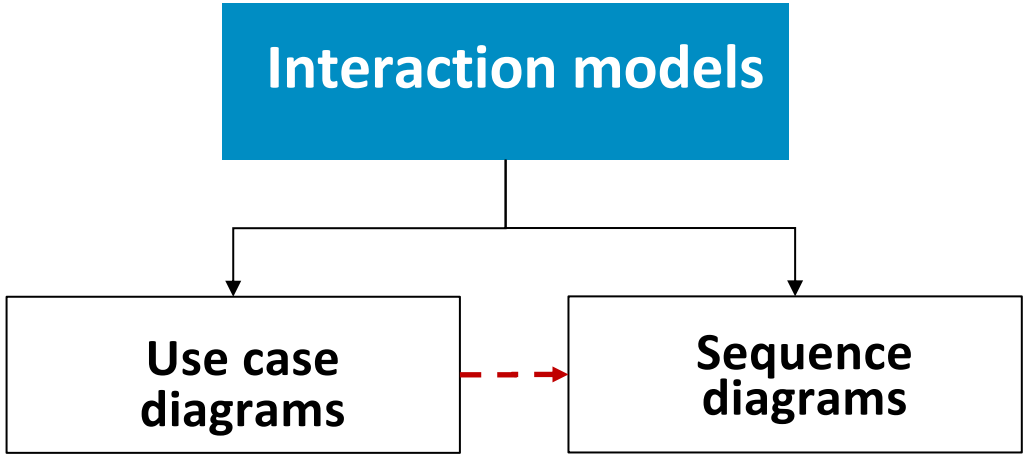
- System boundaries:
  - Define what is inside and outside the system.
  - Show other systems that are used or depend on the system.
  - Affect the system requirements.



The context of the MHC-PMS

Interaction models

- Modeling ***user*** interaction helps to identify user requirements.
- Modeling ***system-to-system*** interaction highlights the communication problems that may arise.
- Modeling ***component*** interaction helps us understand if a proposed system structure is likely to deliver the required performance and dependability.



1. Use Case Diagrams

What is a Use case?

A use case is a **description** of the possible **sequences of interactions** between the **system** and its external **actors**, related to a particular **goal**.

Key Elements of a Use Case Diagram

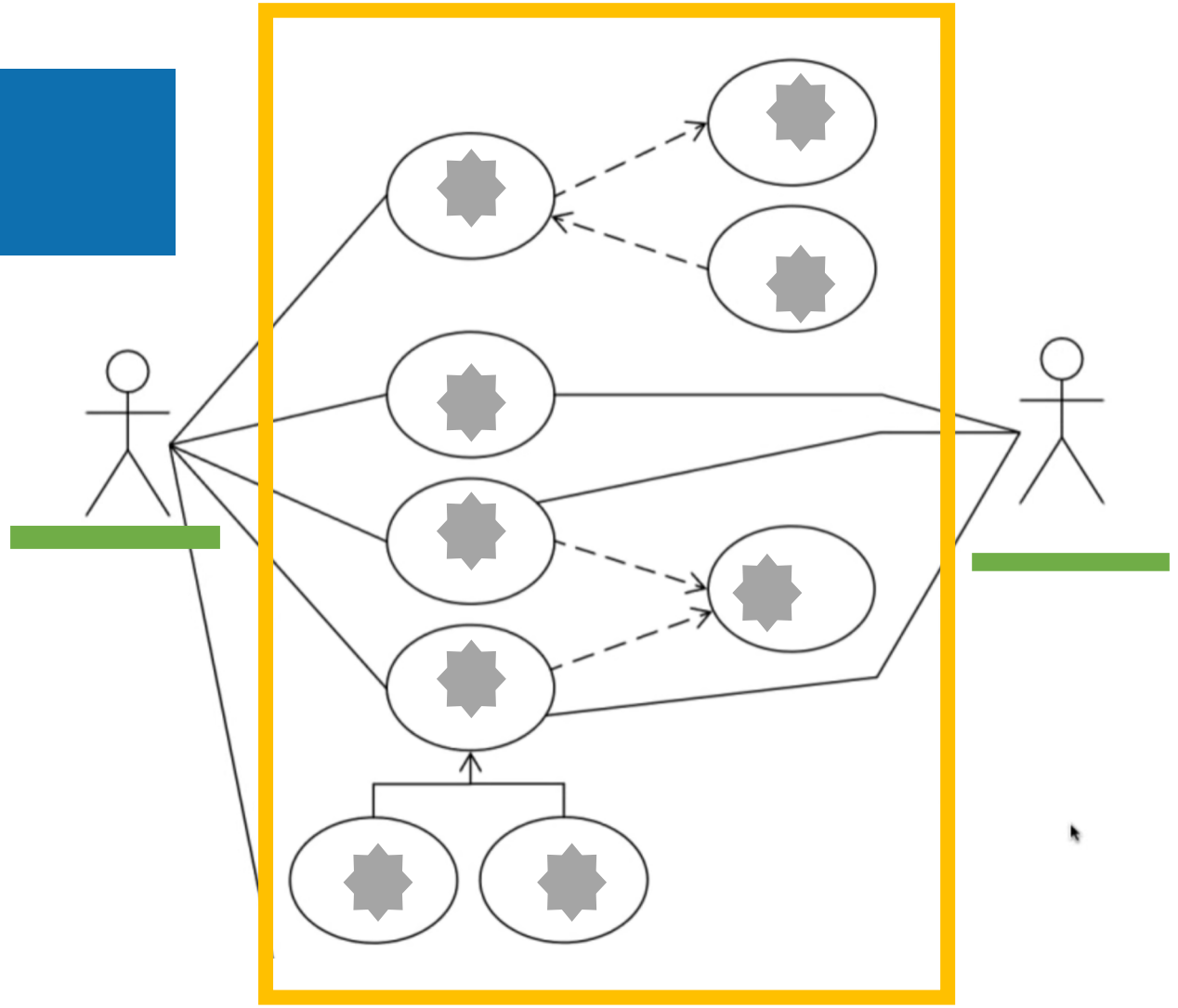
System

Actors

Use Cases

Relationships

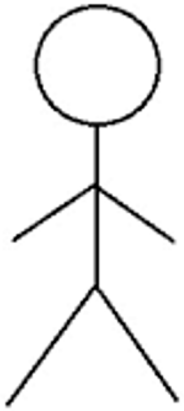
- Association
- Include
- Extend
- Generalization



Use case diagrams describe the software **functionality** from the **users point of view**

Actors

- Actors are the **users (humans)**, **devices**, and **programs** that the software system interacts with.
- **Any entity** that **acts** on the system is an actor.
- In UML (Unified Modeling Language), an actor is represented as a **stick figure or a box**.

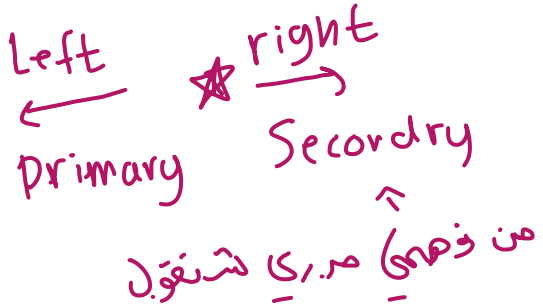


<< External DB >>

Student Data

Type of Actors

- Users
- Database systems
- Clients and servers
- Platforms/systems
- Devices



Primary and Secondary Actors

- A ***primary actor*** initiates an interaction with the system.
- The system initiates interactions with ***secondary actors***.
- **Primary** actors are located on **the left of** the system boundary**.**
- **Secondary** actors are located on **the right** of the system boundary.

Use cases

- Represented as an **ellipse shape**
- Represents **a function** that the system performs (*a goal that some actor can achieve with the system*)
- Should be a **short phrase, starting** with an active **verb that describes a goal.**

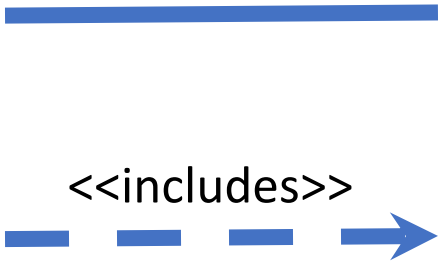
مهم

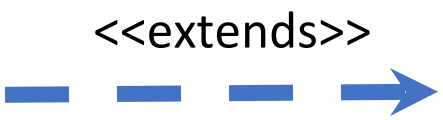
شكل بيضاوي ⇐ تبدأ بأ كتيف فيرب

Relationships: Association

Association shows a relationship, it can be: 3 type of relationship

- Between actor and use case ✓ not directed
- Between use cases → only between use cases
- Between actors → between user cases and actor

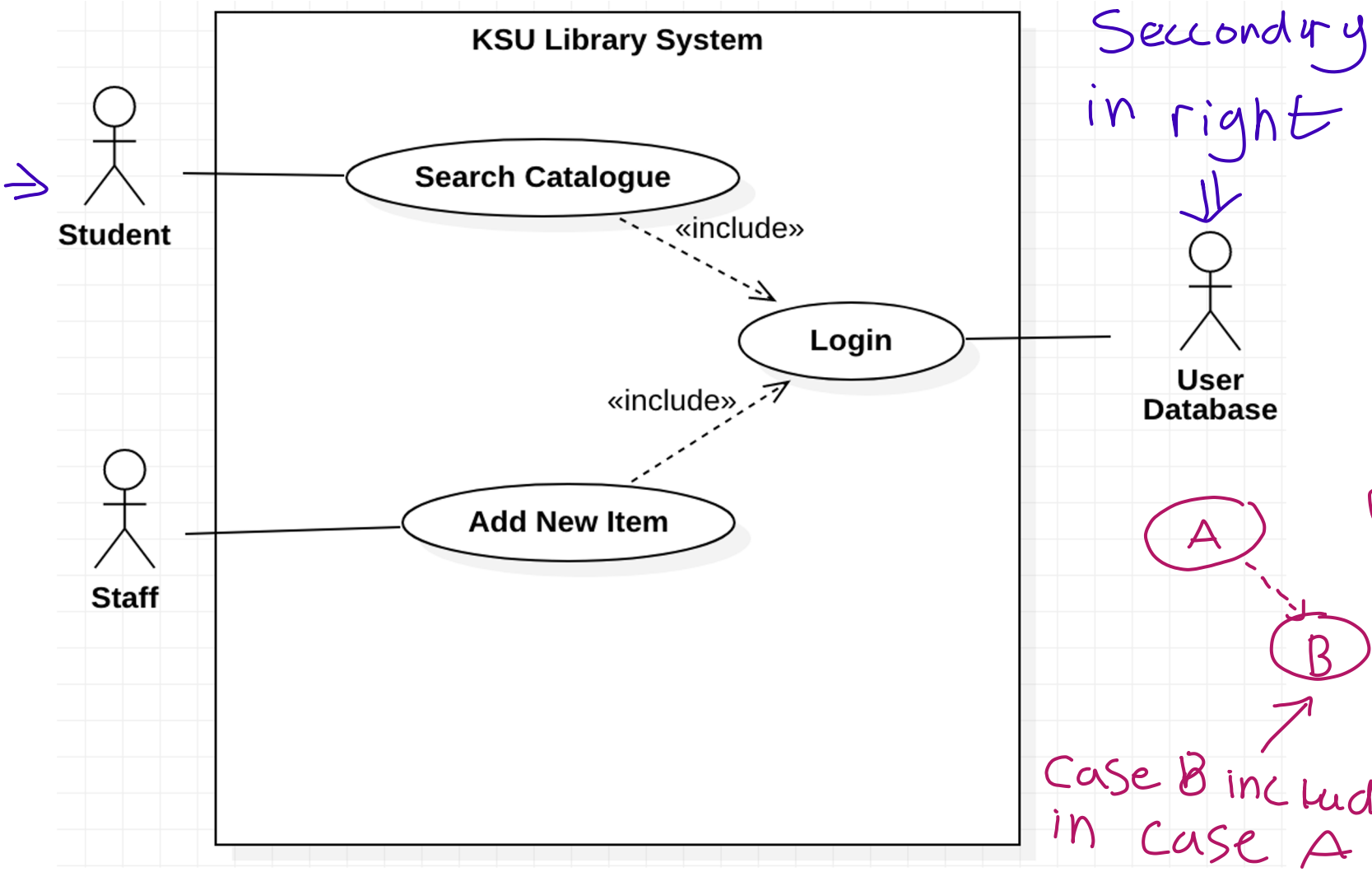




Relationships: <<include >>

- Is used when a use-case **contains** the processes of another use-case *as part* of its **normal flow** of actions.
- For **reusable parts** of behavior across two or more use cases**.** Its main goal is to **avoid repetition**; They are usually included by **multiple use cases**
- Happens **every time**
- The base use case depends on included use case indicated by **arrow direction** (towards the included use case)**.**
- An included use case is like a **helper** function that is called by other higher level functions.
- Included use cases usually **don't have actor associations**.

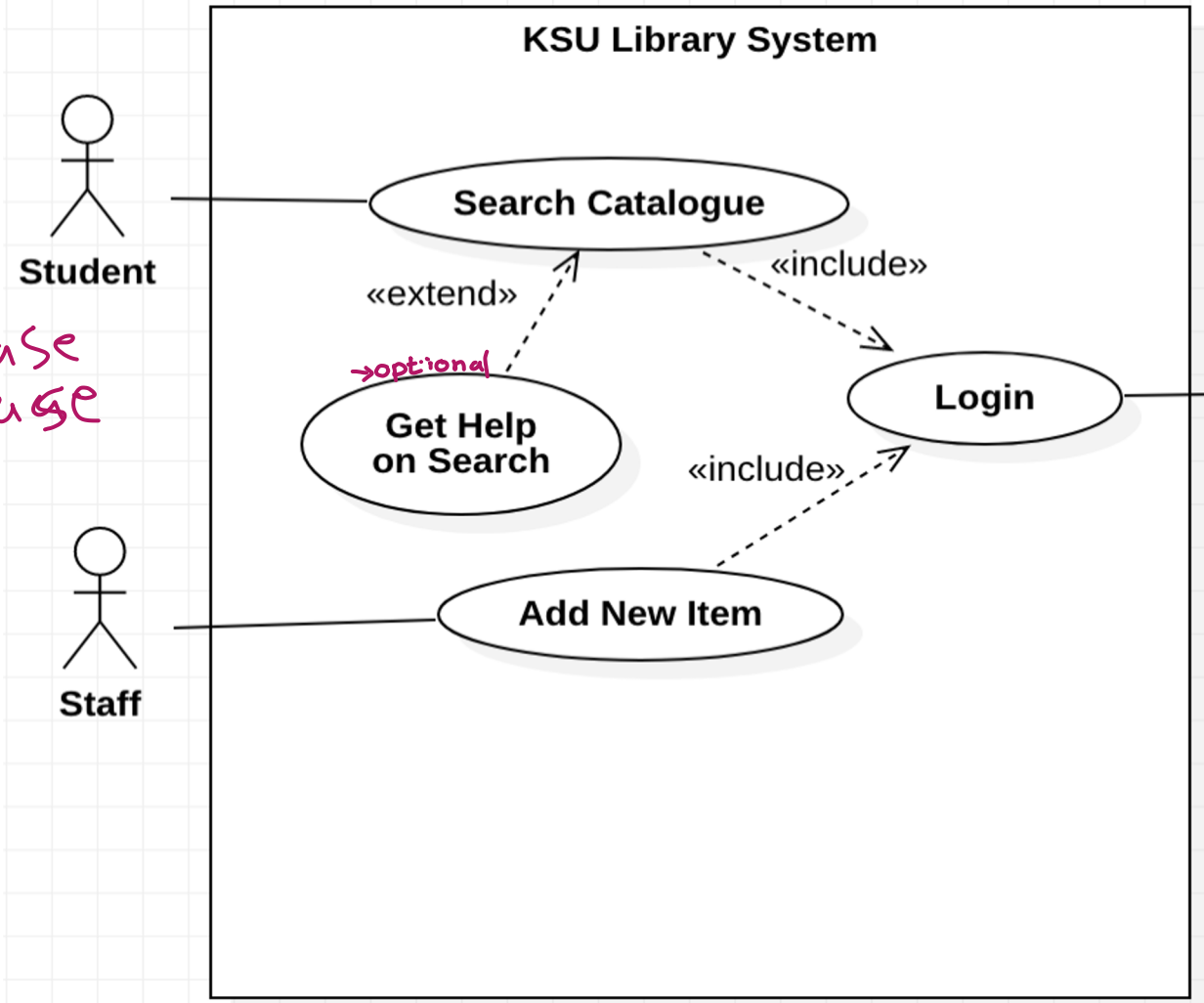
Relationship: <<include>>



Relationship: <<extends>> → Something optinal

- Is used when a use-case **augments** the behavior of another use-case.
- When the functionality of the original use-case need to be extended as a result of some exceptional circumstance (the original use-case can be executed without the extended use-case). Optional behavior added to use case.
- The main goal of using the Extend relationship is to reduce the complexity of a large use-case (by dividing it).
- Happens **some times** * like help
- Helps keep the base case unchanged while adding more specifics or conditional changes.
- The base case is independent of the extend use case. **The arrow direction** (towards the base use case)**.**
- The advantage is that use case extensions can be replaced without modifying the extended use case.

Relationships: <<extends >>



optional —extends→ base case

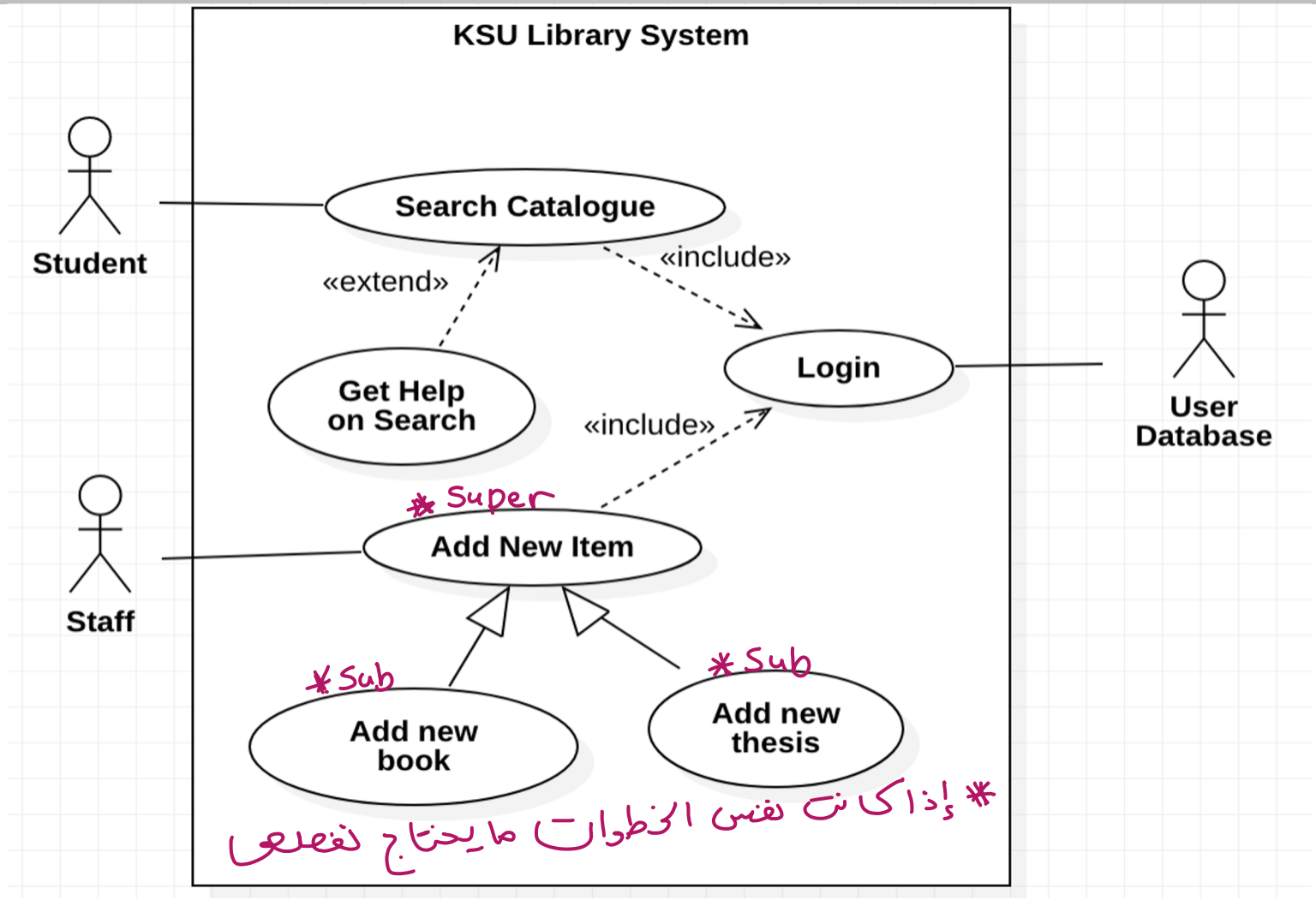
* Somthing always happend => include

* optinal => extend

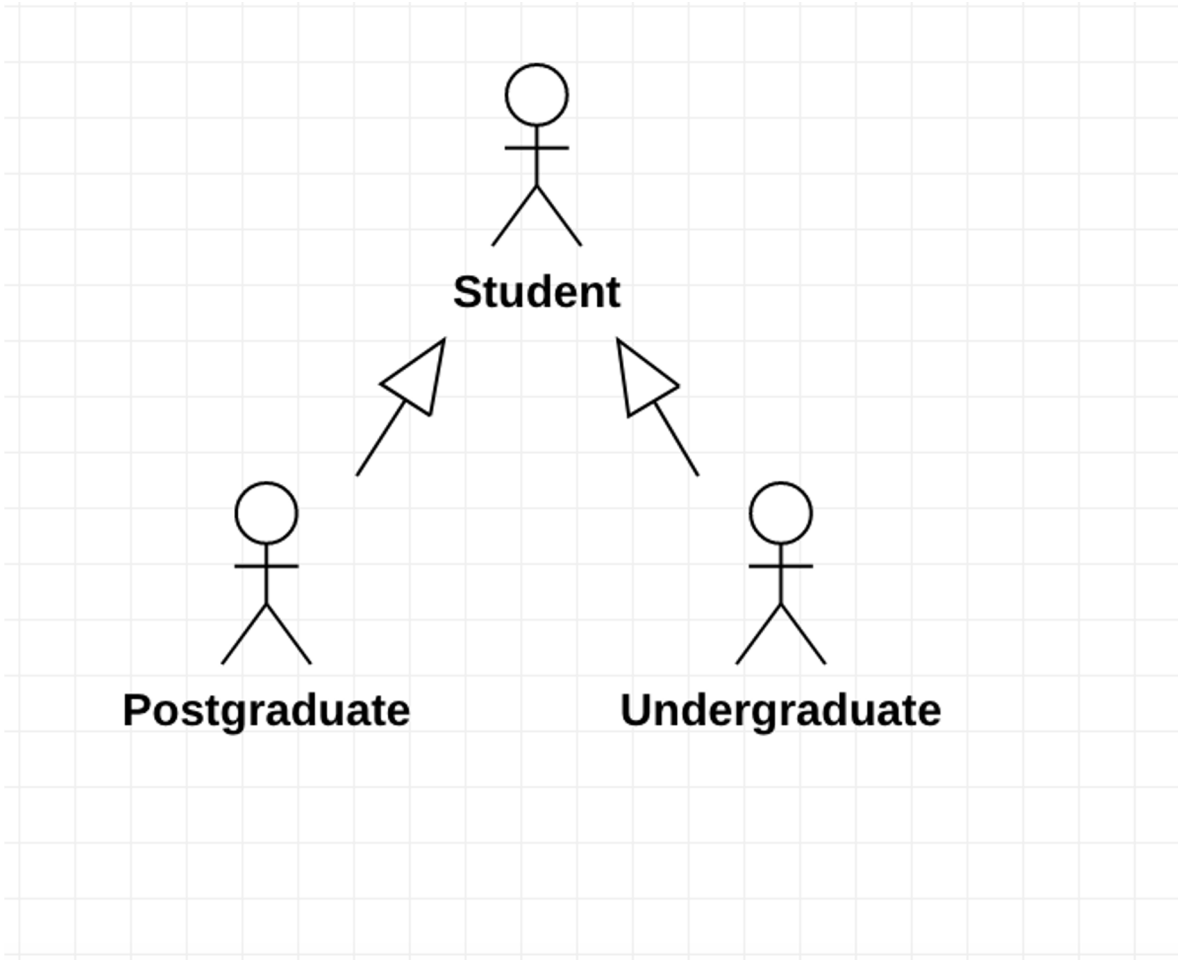
Relationships: Generalization (inheritance)

- The sub use-cases are expected to redefine the behavior of the super use-case.
- The behaviors of the super and the sub use-cases can be very different.
- When one use case is a specialized form of another use case.

Relationships: Generalization

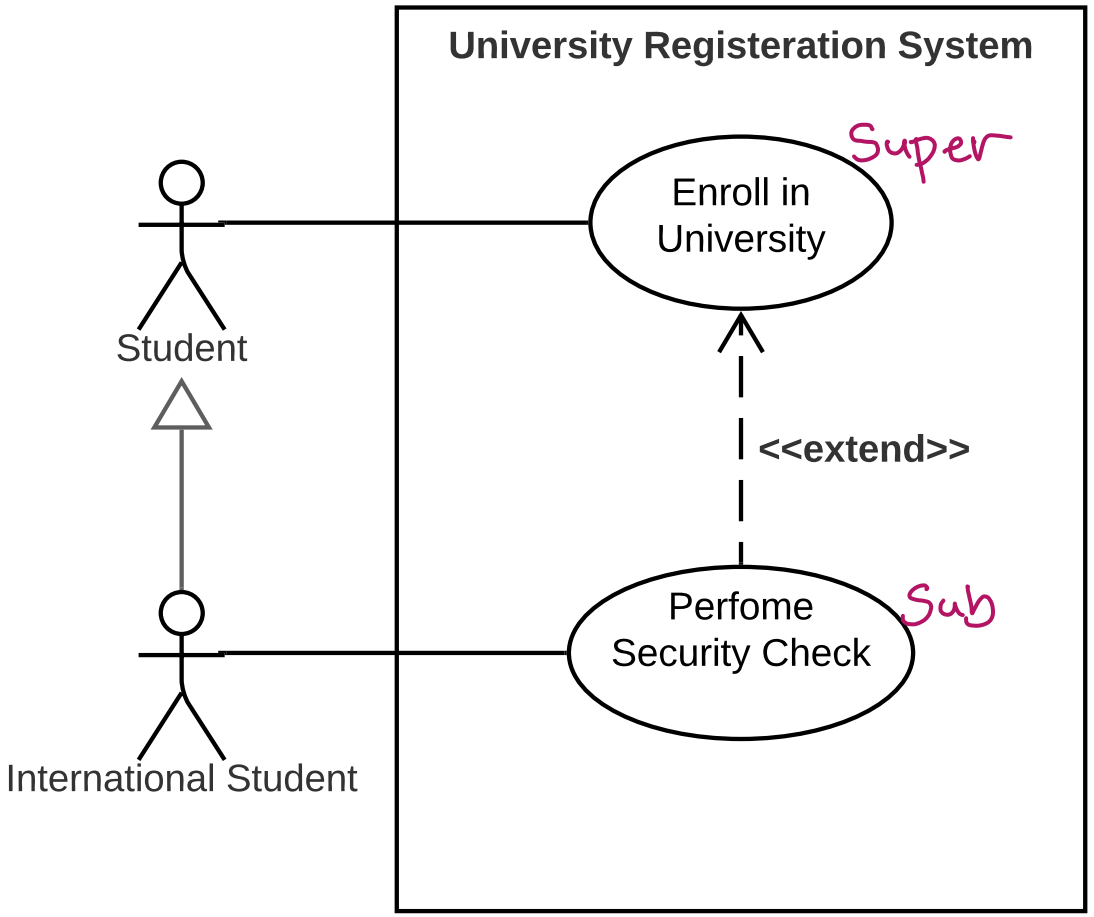


Relationships: Between Actors



Relationships: Between Actors

* why do we need inheritance



2. Use Case Descriptions

Use case description

- The details for accomplishing the use case.
- A textual description.

See details and example in system analysis and design guide

- Use Case name
- Actor
- Purpose
- Overview
- Cross references
- Pre-conditions
- Post-conditions
- Type
- Typical course of events

Use case description (cont.)

1. **Use Case name:** the use-case name must be meaningful and unique. For example, Log In, Rate Product, and Buy Items.

2. **Actor:** the type of users who can initiate this use-case. A use-case may have a single actor or multiple actors. If a use-case has multiple actors, it is a good idea to decide who is the initiator of the use-case. For example, the Buy Items use-case has two actors: customer (initiator) and cashier.

3. **Purpose:** the objective of the use-case. Technical and implementation details should not be mentioned as a part of the use-case purpose.

Use case description (cont.)

- **Overview:** what type of actions take place during this use-case? It is a good idea to start the use-case overview with this sentence: “This use-case begins when …..” and ends it with “This use-case ends when …..” or “On completion, …..”.

- **Cross references:** the number of the functional requirement that can be linked to this use-case.

- **Pre-conditions:** the conditions that need to be met before the use-case can begin.
- **Post-conditions:** the changes that occur after the execution of the use-case such as saving information or generating output.

Use case description (cont.)

- **Type:** there are two ways to describe the type of a use-case:
  (1) According to usage and
  (2) According to abstraction level.

According to **usage**, use-case type can be:

Primary: when the use-case describes a major and common function.

Secondary: when the use-case describes an exceptional function (a function that is rare or unusual).

Optional: when the use-case describes a function that may or may not be used (may or may not be implemented).

Use case description (cont.)

According to **abstraction** level, a use-case type can be:

1. ***Essential:*** when the description of the use-case is written using a high-level language (free of technology and implementation details).

2. ***Real:*** when the description of the use-case involves a detailed design description.

Essential use-case	Real use-case
The account holder identifies herself to the ATM.	The account holder inserts the card into the ATM card reader. The account holder is prompted to enter her PIN which she input using a numeric keypad.

Use case description (cont.)

- **Typical course of events:** is the formal description of the flow of events that occur during the execution of the use-case.

Actor action	System response
1. This use-case begins when a customer arrives at the checkout point.	
2. The cashier scans each item.	3. Determines the item price and adds its information to the current transaction.
4. The cashier indicates that the item entry is complete.	5. Calculates and displays the total.
6. The cashier tells the customer the total.	
7. The customer gives a cash payment. The cashier records the cash received.	8. Shows the balance due back to the customer.
9. The cashier deposits the cash and extracts the balance owing. 11. The cashier gives the balance owing to the customer.	10. Logs the completed sale. Print the receipt.
12. The cashier gives the receipt to the customer. 13. The customer leaves with the items.	
Alternatives: Line 9: insufficient cash in drawer to pay balance. Ask for cash from supervisor.	

Activity 1: Use Case Diagram

Draw a use case diagram that represents an *Airline Reservation System.*

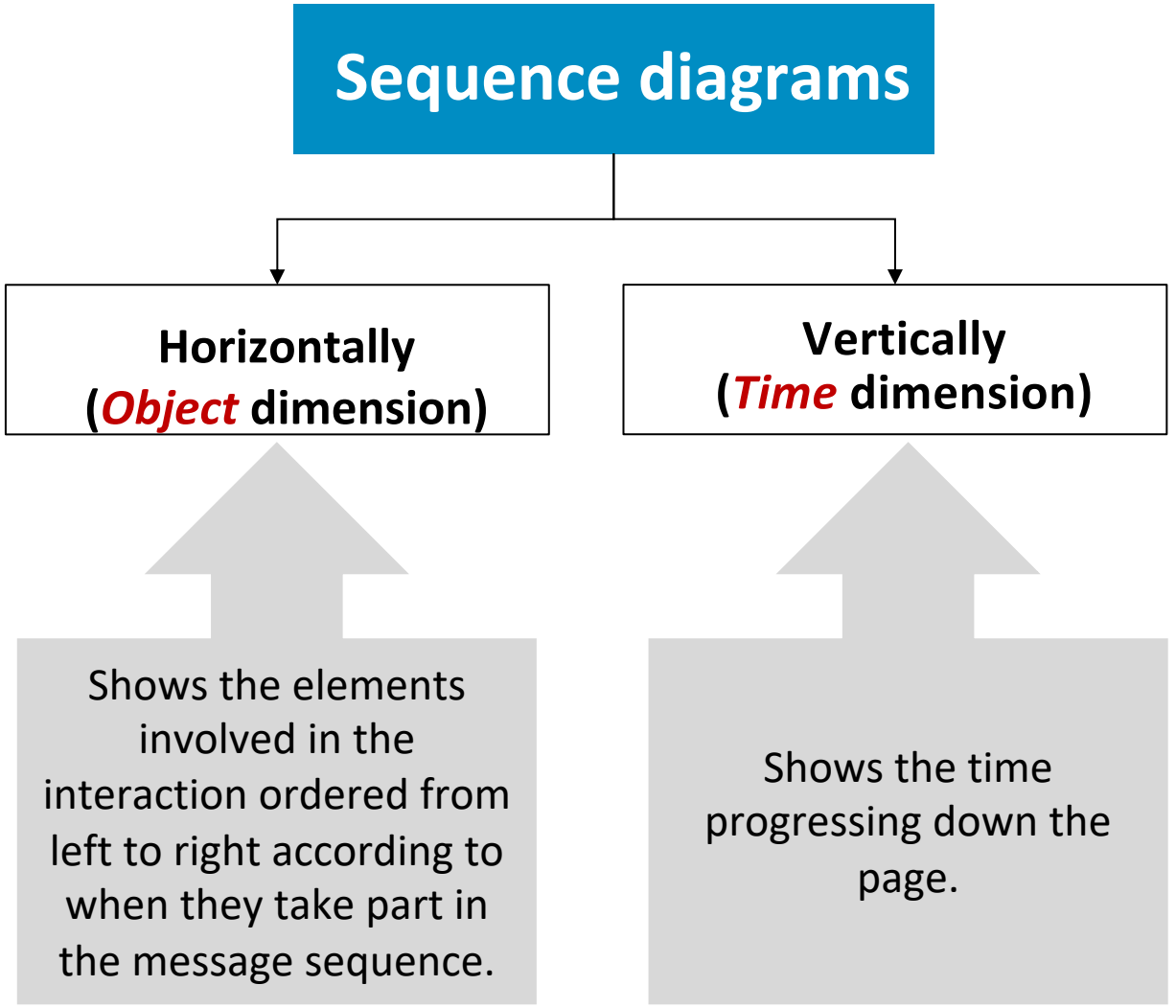
In Airline Reservation System, passengers can search available flights either by destination or travel date. They can also book a ticket and pay for it using credit card. The credit card information is verified by a third-party payment system. While booking a ticket, passengers have the option to choose a seat if they like to. Finally, the passengers can cancel their booking.

The airline administrator can use the Airline Reservation System to update the flight schedule.

2. Sequence Diagrams

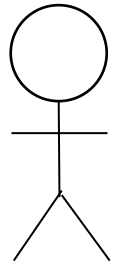
Sequence diagrams

- Used to model the interactions between the actors and the objects.
- Shows the sequence of interactions within a particular use case.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.



Sequence diagram notations

- **Actor**:
  - An external entity role that interacts with the system by exchanging signals and data
  - Can be human users or external hardware.



- **Class**:
  - Describes the way an object will behave in context.

- **Comment**:
  - Gives the ability to add various remarks to elements.

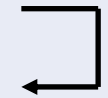
Sequence diagram notations

- **Activation or execution occurrence:**
  - Represents the time an object needs to complete a task.
  - For example, when an object is busy executing a process or waiting for a reply message.

- **Lifeline**:
  - Vertical dashed line that indicate the object's presence over time.

Sequence diagram notations

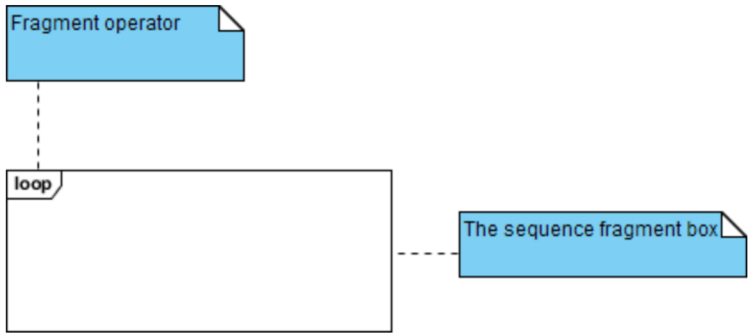
- **Messages:**
  - Messages are arrows that represent communication between objects.

Message	Description	Notation
Synchronous message	Message requires a response before the interaction can continue.	
Asynchronous message	Message doesn't need a reply for interaction to continue.	
Reply or return message	Drawn pointing back to the original lifeline.	
Self message	A message an object sends to itself.	

Sequence diagram notations

College of Computer and Information Sciences

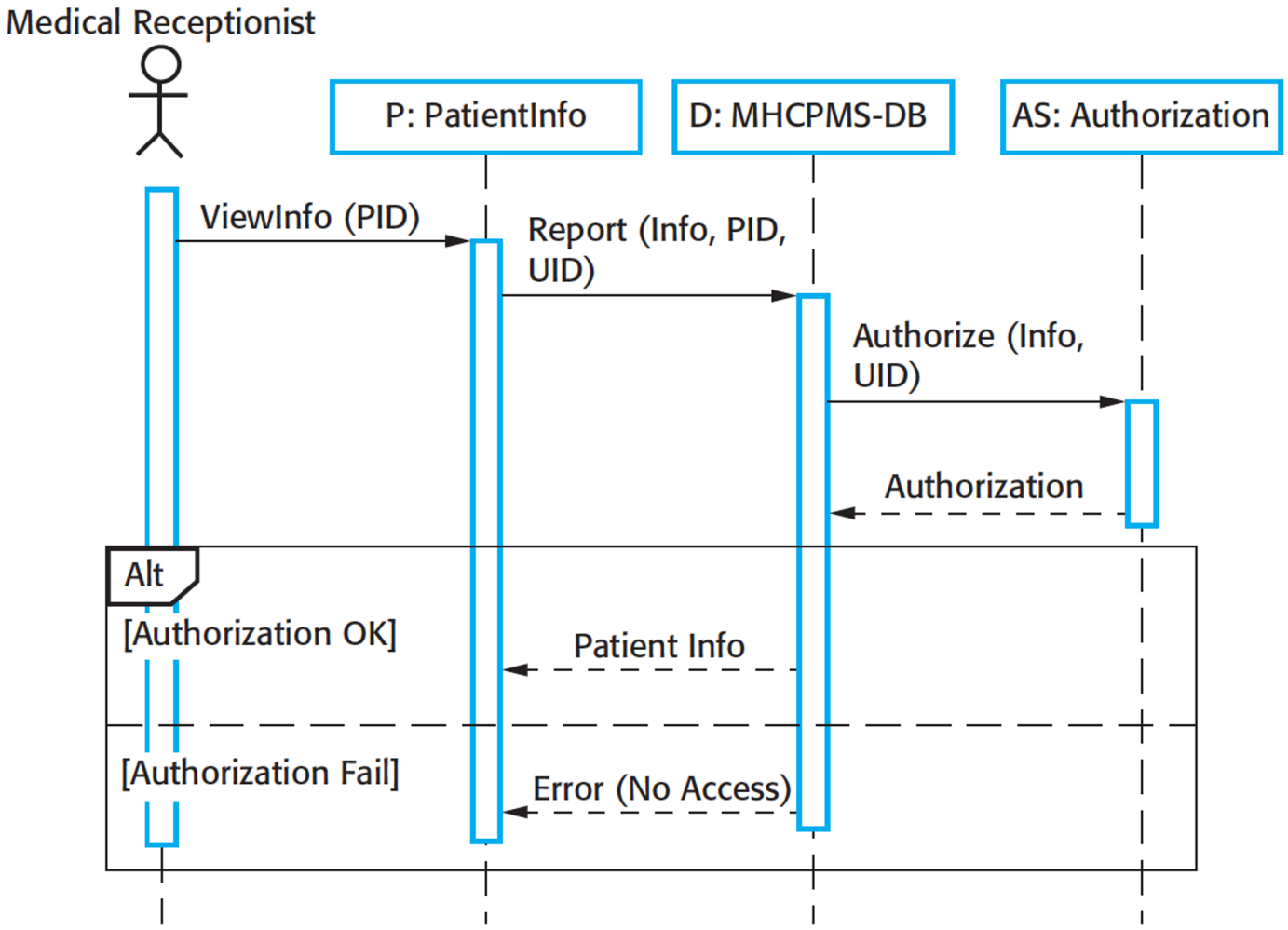
- **Sequence fragments:**
  - Represented as a box that encloses a portion of the interactions within a sequence diagram.
  - Multiple types.
  - A fragment operator is used to show the fragment type (top left cornet).
  - Multiple fragments can be combined.



Fragment type	Description
alt	Multiple alternatives. Only the one whose condition is true will execute *("if .. then .. else"* statement).
opt	Optional interaction that execute if the condition is true *("if .. then"* statement).
par	Each fragment is run in parallel.
Loop	The fragment may execute multiple times.
neg	The fragment shows an invalid interaction.

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

Sequence diagrams



Sequence diagram for View patient information

Sequence diagrams - *View patient information*

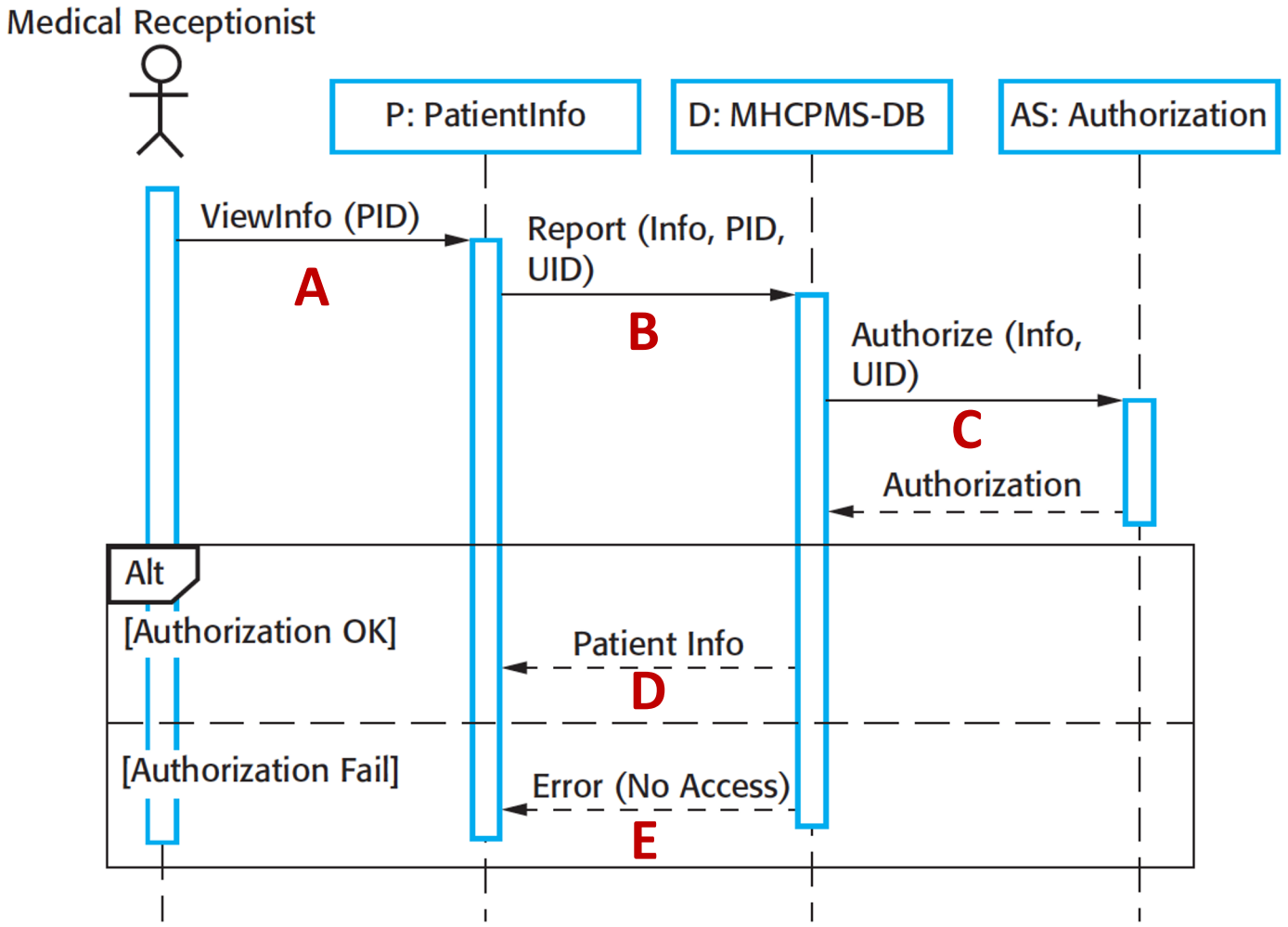
A The medical receptionist triggers the ***ViewInfo*** method in an instance ***P*** of the ***PatientInfo*** object class, supplying the patient's identifier, ***PID***. ***P*** is a user interface object, which is displayed as a form showing patient information.

B The instance ***P*** calls the database to return the information required, supplying the receptionist's identifier to allow security checking.

C The database checks with an authorization system that the user is authorized for this action.

D If authorized, the patient information is returned and a form on the user's screen is filled in.

E If authorization fails, then an error message is returned.



Sequence diagrams - *Transfer data*

1. The receptionist logs on to the PRS (patient record system).
2. There are two options available. These allow the direct transfer of updated patient information to the PRS and the transfer of summary health data from the MHC-PMS to the PRS.
3. In each case, the receptionist's permissions are checked using the authorization system.
4. Personal information may be transferred directly from the user interface object to the PRS. Alternatively, a summary record may be created from the database and that record is then transferred.
5. On completion of the transfer, the PRS issues a status message and the user logs off.