



# **Topic 7:**

# **Use Case Modeling**

# References

- [1] Dennis, Alan, et. al., System Analysis and Design with UML 3rd Edition, John Wiley & Sons, 2010. (Chapter 5 Use Case Diagrams)
- [2] Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd Edition, Pearson Education International, USA, 2005. (Chapter 6 Use Cases)

# Objectives

- To outline the **functionality of the system**.
- To define what will be handled **by the system** and what will be handled **outside the system**.
- To define **who** and **what** will interact with the system.
- Create diagrams of the **use-case model**.
- Create the **use-case specification**.

# Background

- In requirement elicitation:
  - ❖ Users may not know what is and is not possible for the system to do.
  - ❖ Users may have difficulty envisioning new ways to redesign business processes.
  - ❖ It is common for users to describe things they think they want from the new system, but our focus should be the real needs for the new system.

Finally, users often found it difficult to learn the process and data modelling language used by the analysts.

# Motivation

- Use cases are a good way to make describing requirements simple.
- Use cases encourage users (e.g. business person, domain experts, requirement donors) participation.
- Use cases emphasise the user goals and their perspective
  - ❖ Who is using the system? What are their typical scenarios of use? What are their goals?
- Use cases are scalable (both up & down) in terms of sophistication and formality.

# What is use case modeling?

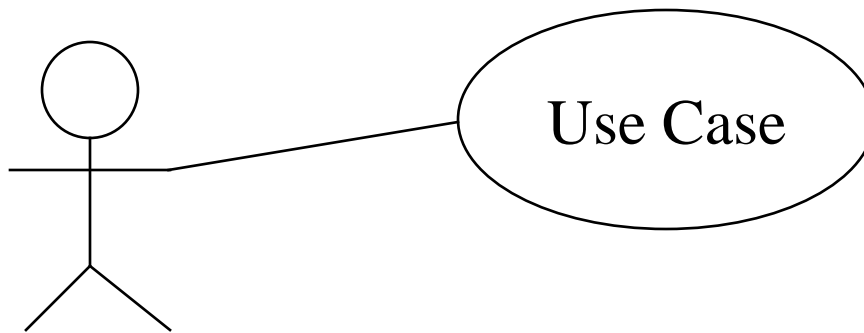
- **Use Case Model:**

- ❖ A view of a system that emphasizes the behavior as it appears to outside users. A use case model partitions system functionality into transactions ('use cases') that are meaningful to users ('actors').
- ❖ Consists of Use Case Diagrams & Use Case Descriptions

# What is Use Case?

A use case describes a sequence of actions of a system performs that yields a result of value to a particular actor.

**A series of user – system interactions that helps the user to accomplish something.**



# Use Case and Domain Process

- ❖ A use case describes a process flow.
- ❖ A process flow describes, **from start to finish**, a sequence of events, actions, and transactions required to produce or complete something of value to an organization or actor.
- ❖ Examples of use cases:
  - ☐ Withdraw cash from an ATM
  - ☐ Register for courses at school
  - ☐ Book flight tickets
  - ☐ Checkout shopping



# A Common Mistake with Use Cases


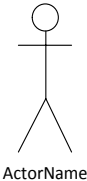
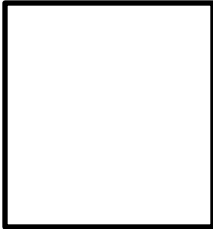
A use case is a relatively large **end-to-end** process description that typically includes many steps or transactions; it is not normally an individual step or activity in a process. (Except use case of view functions e.g. View Balance, etc.).

“printing the receipt”  
in the ATM System



**NOT A USE CASE,  
because it is only a step  
inside other use cases, we  
cannot print the receipt at  
any desired time**

# USE CASE DIAGRAMS

# Core Elements

| Construct       | Description  | Examples  | Notation  |
|-----------------|--|---|---|
| use case        | A sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system. | Log in,<br>Log out,<br>View profile,<br>Edit profile  |    |
| actor           | A coherent set of roles that users of use cases play when interacting with these use cases.                                    | Admin,<br>user,<br>receptionist,<br>guest,<br>manager |    |
| system boundary | Represents the boundary between the physical system and the actors who interact with the physical system.                      |   |  |

# Core Relationships

| Construct             | Description   | Syntax  |
|-----------------------|---|---|
| <b>association</b>    | The participation of an actor in a use case. i.e., instance of an actor and instances of a use case communicate with each other.  |  |
| <b>generalization</b> | A taxonomic relationship between a more general use case and a more specific use case.<br>It can also be a relationship between a more general actor and a more specific actor. |  |

Notes:

It is possible to represent the multiplicity of the association using asterisk (\*) see [1]

# Core Relationships

| Construct      | Description   | Syntax   |
|----------------|---|--|
| <b>extend</b>  | A relationship <b>from an <i>extension</i> use case to a <i>base</i> use case</b> , specifying how the behavior for the extension use case can be inserted into the behavior defined for the base use case. | <pre>&lt;&lt;extend&gt;&gt;<br/>-----&gt;</pre>  |
| <b>include</b> | A relationship <b>from a <i>base</i> use case to an <i>inclusion</i> use case</b> , specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.     | <pre>&lt;&lt;include&gt;&gt;<br/>-----&gt;</pre> |

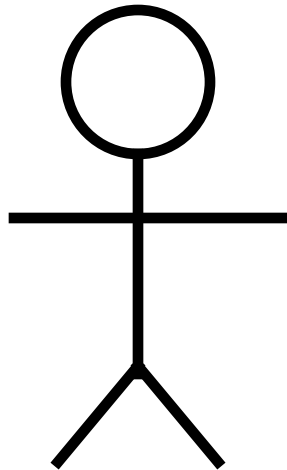
# Naming Use Cases

Name a use case starting with a verb in order to emphasize that it is a process.

- Buy Items
- Enter an Order
- Maintain Customer Profiles

# Actors

An actor is **an entity external to the system** who in some way participates in the story of the use case.



# Find Actors

To find the actors, may ask the following questions:

- Which **user groups require help from the system** to perform their tasks?
- Which **user groups are needed to execute** the system's most obvious **main functions**?
- Which **user groups** are required to **perform secondary functions**, such as system maintenance and administration?
- Will the system interact with **any external hardware or software system**?
- Who uses the system?
- Who gets information from this system?
- Who provides information to the system?
- Where in the company is the system used?
- Who supports and maintains the system?
- What other systems use this system?



# Actors

- ❖ There is one **initiator actor** who generates the starting stimulus, and possibly several other **participating actors**.
- ❖ Kind of actors include:
  - ❑ **roles** that people play
  - ❑ computer **systems**
  - ❑ electrical or mechanical devices

# Identifying Use Cases

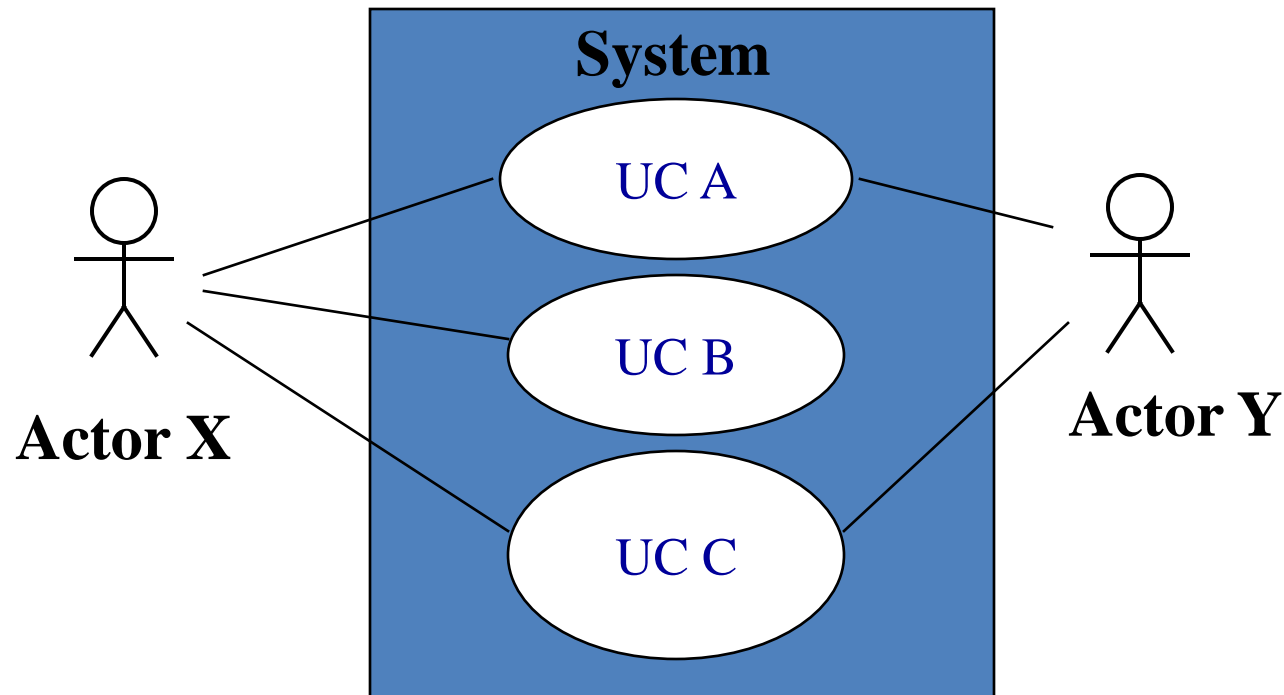
## Actor-based:

1. Identify the actors related to a system or organization.
2. For each actor, identify the process they initiate or participate in.

## Event-based:

1. Identify the external events that a system must respond to.
2. Relate the events to actors and use cases.

# Use Case Diagrams



A use case diagram illustrates a set of use cases for a system, the actors, and the relation between the actors and use cases.

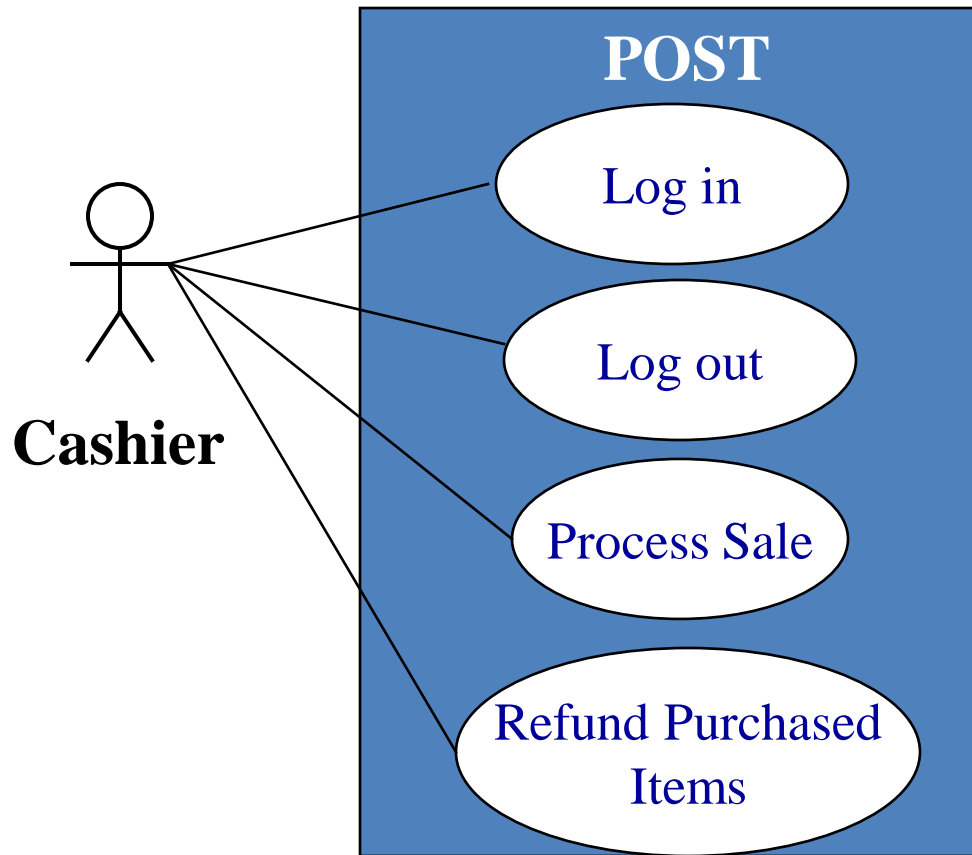
# Example System:

## Point-of-Sale Terminal

Assume that we have been requested to create the software to run a point-of-sale terminal (POST).



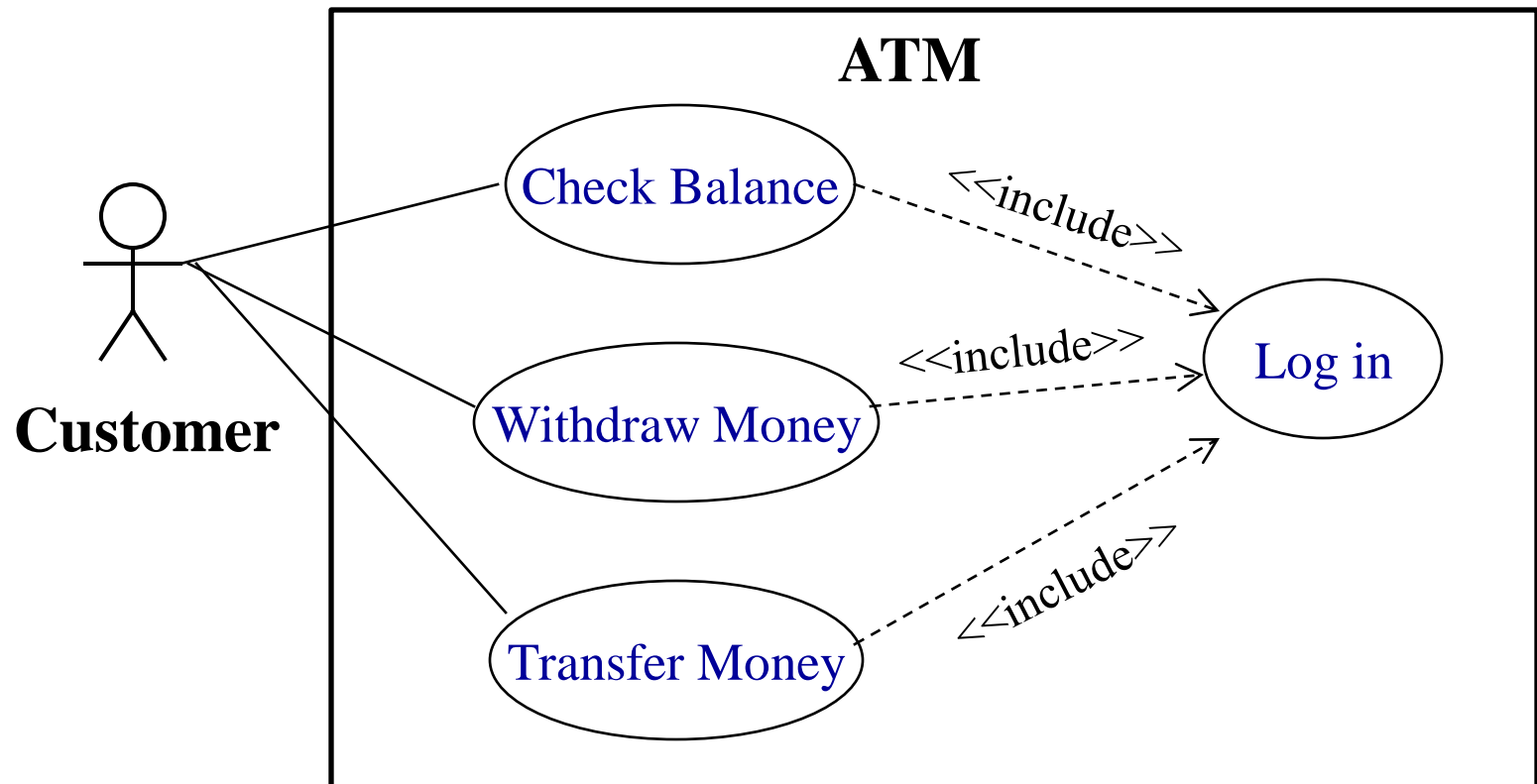
# Use Case Diagrams



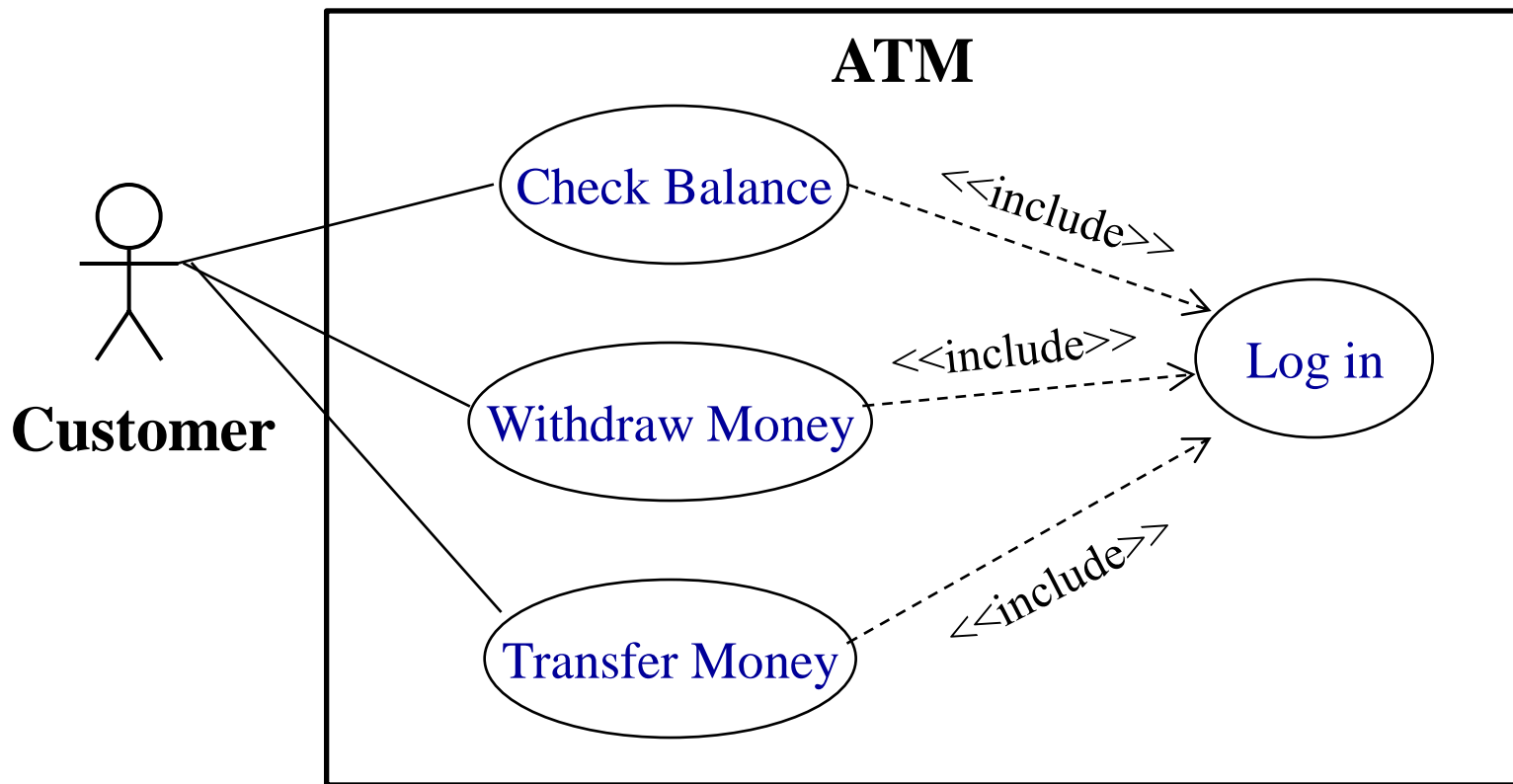
A use case diagram illustrates a set of use cases for a system, the actors, and the relation between the actors and use cases.

# Include UC Relationship

A relationship **from a base use case to an inclusion use case**, specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

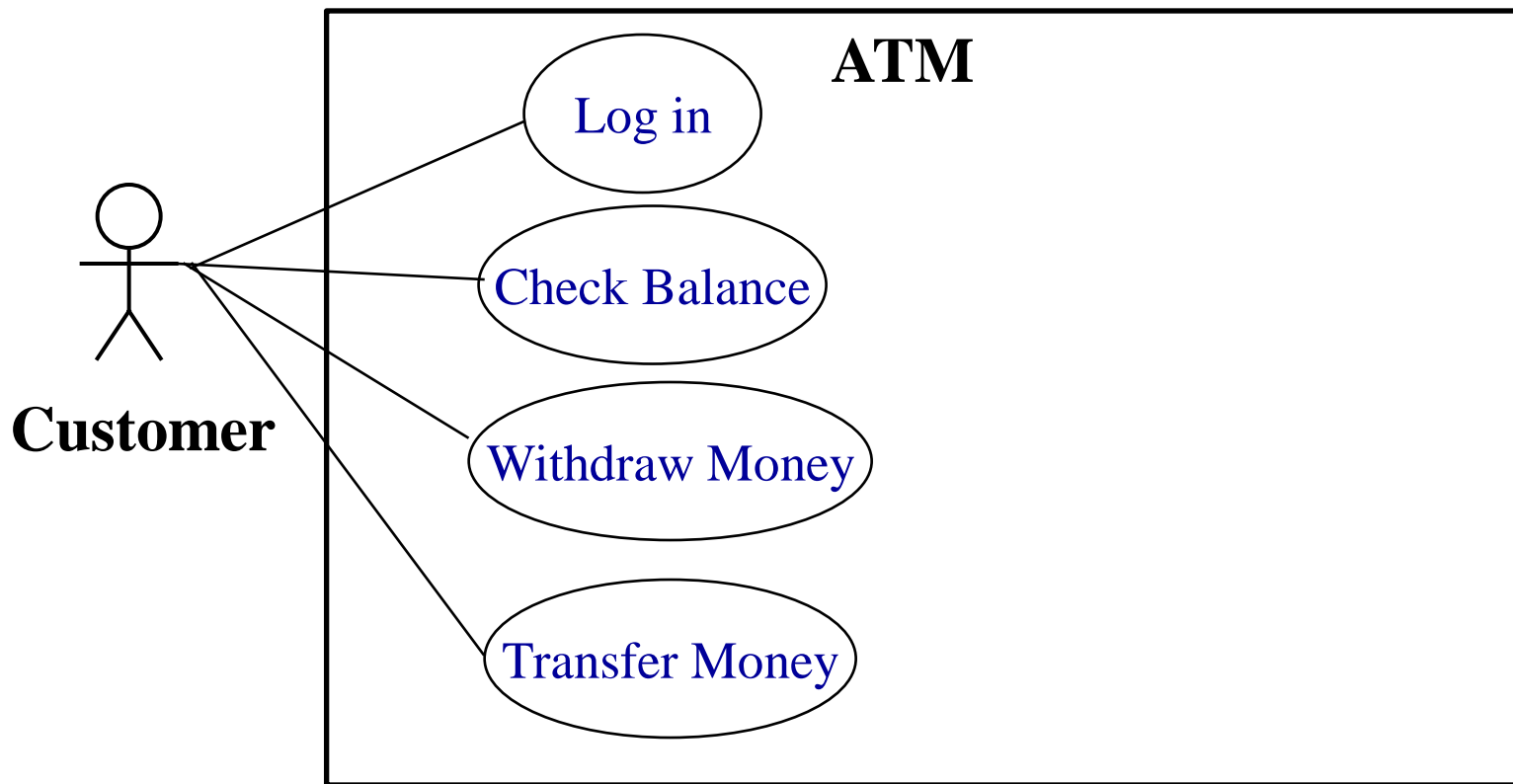


# Discussion : Login as precondition vs inclusion use case



**Q:** Does customer has the ability to invoke UC "Login" as an independent use case?

# Discussion : Login as precondition vs inclusion use case

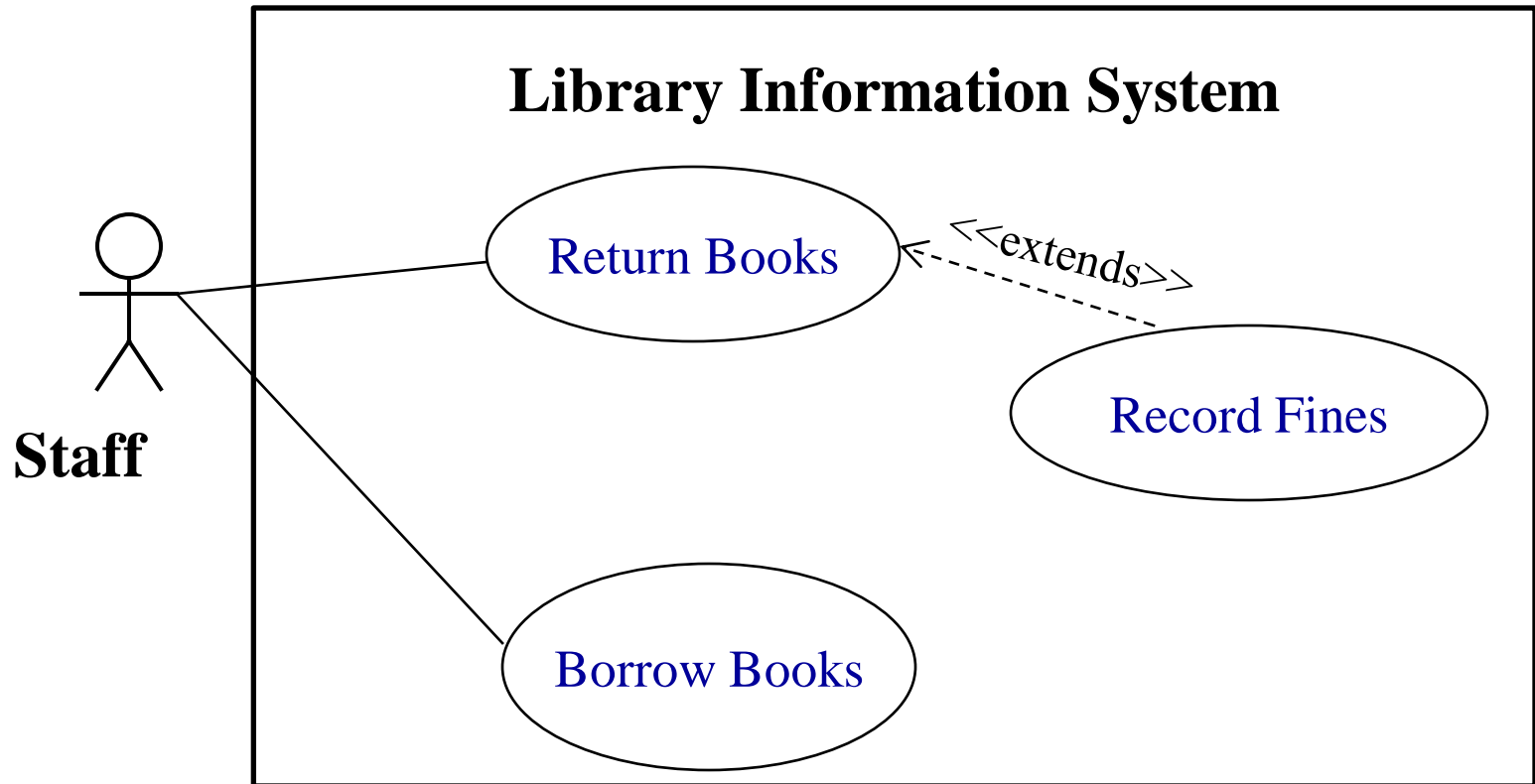


**Q:** Does customer has the ability to invoke UC “Login” as an independent use case?



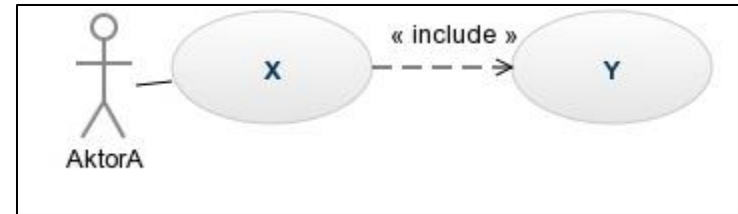
# Extend UC Relationship

A relationship **from an *extension use case* to a *base use case***, specifying how the behavior for **the extension use case can be inserted into** the behavior defined for the **base use case**.

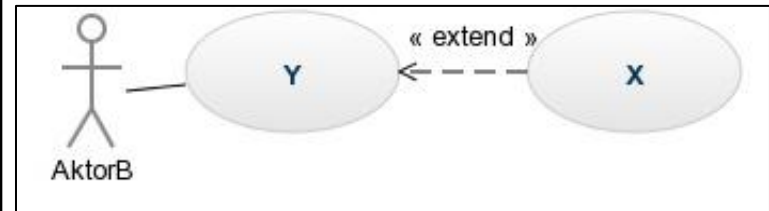


# What is the difference between *include* and *extends*?

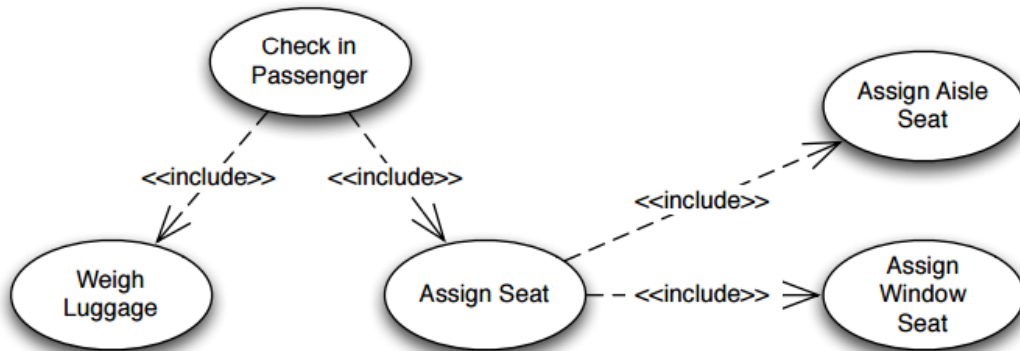
"X *include* Y" indicates that the task "X" has a subtask "Y"; that is, in the process of completing task "X", task "Y" will be completed at least once.



"X *extends* Y" indicates that "X" is a task for the same type as "Y", but "X" is a special, more specific case of doing "Y". That is, doing X is a lot like doing Y, but X has a few extra processes to it that go above and beyond the things that must be done in order to complete Y.



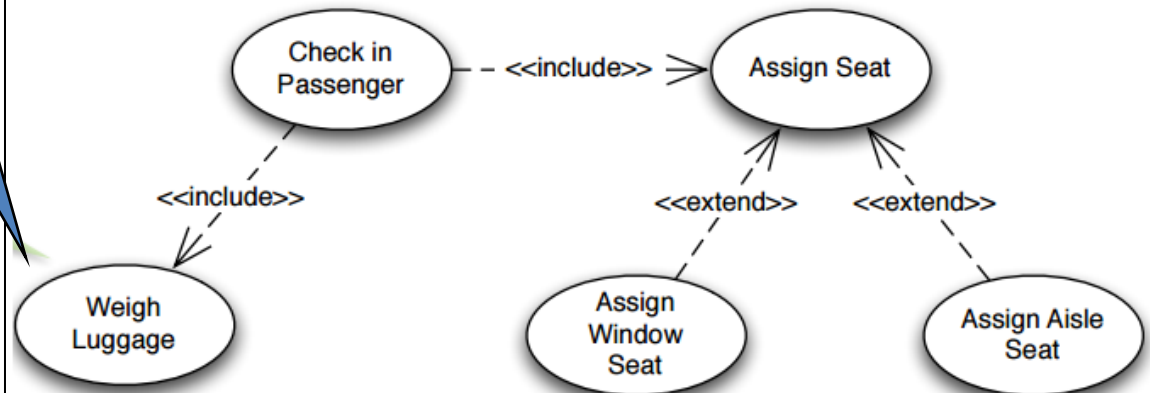
Trying to add detail (INCORRECT):



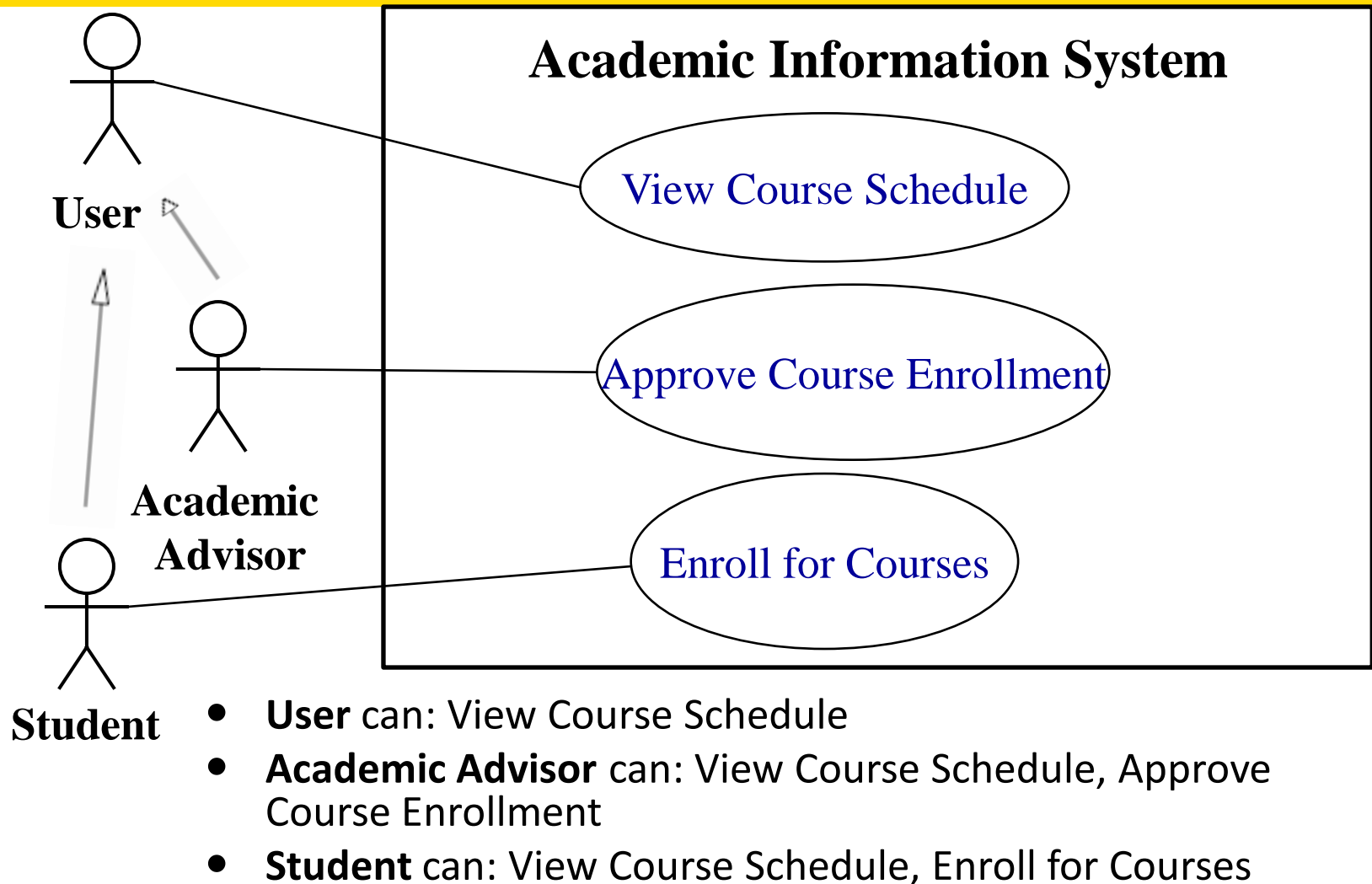
There are two ways to assign a seat: assigning a window seat and assigning an aisle seat, but only one need be completed in the process of assigning the passenger a seat.

This diagram says that in order to assign a seat you must assign both a window seat AND an aisle seat to the passenger.

Trying to add detail (CORRECT):



# Actor Generalization Relationship



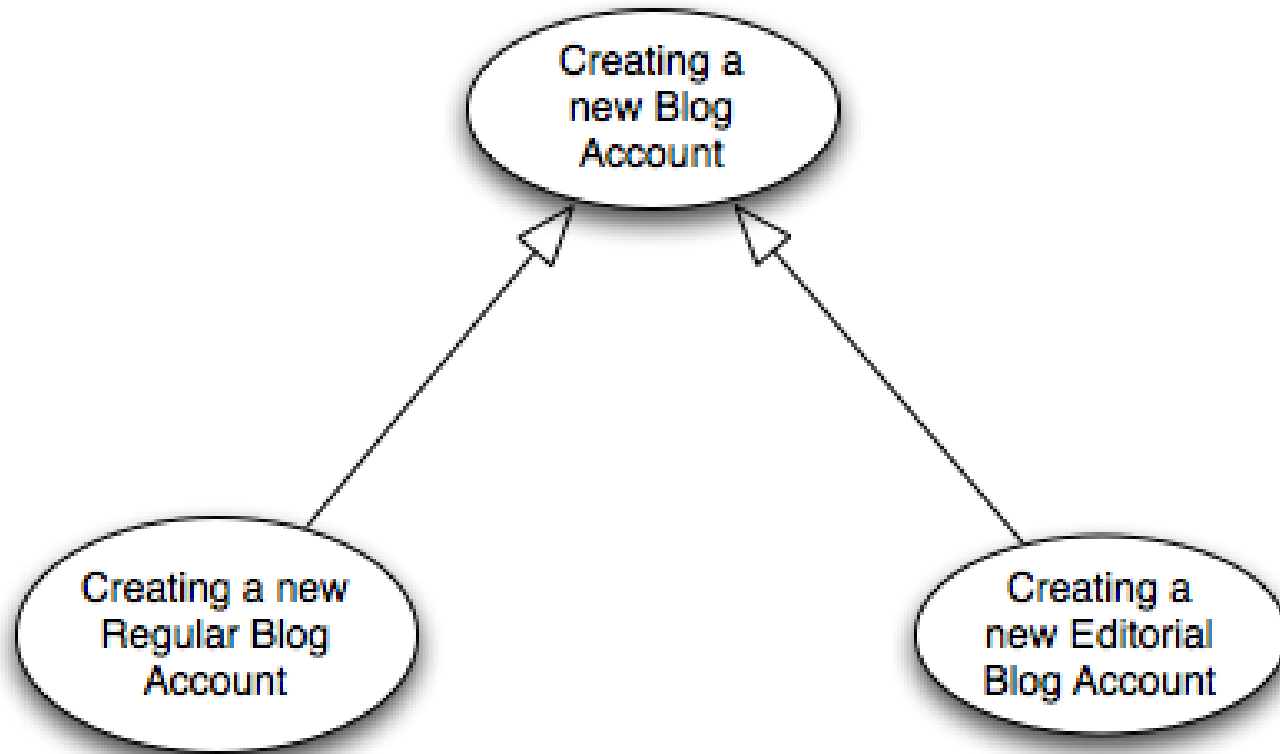
# Use Case Generalization Relationship

- Use case generalization is useful in showing a condition where one use case is a special type of another use case.
  - ❖ Applying a use case with small changes for a collection of specific situations.
  - ❖ Powerful way of reusing a use case so that **you only have to specify the extra steps that are needed in the more specific use cases.**

# Use Case Generalization Relationship

- Things that you need to be aware when using inheritance relationship:
  - ❖ **Every** step in the general use case **must** occur in the specialized use cases.
  - ❖ **Every** relationship of the general use case with the actor **must** also make sense in the more specialized use cases
- If you don't want your specialized use case to do everything that the general use case describe, the **don't use generalization.**

# Use Case Generalization Relationship



# **USE CASE SPECIFICATIONS**



# Writing Use Case

- Three formats

- ❖ Terse

- One-paragraph summary, usually the main success scenario

- ❖ Casual

- Informal paragraph format

- ❖ Fully-dressed

- All steps and variations are written in detail
    - There are supporting sections, such as pre-/post-conditions.

# Example : Terse (Brief) Use Case

**Process Sale:** A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

# Example : Casual Use Case

## Handle Returns

*Main Success Scenario:* A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item ...

### *Alternate Scenarios:*

If the customer paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.

If the item identifier is not found in the system, notify the Cashier and suggest manual entry of the identifier code (perhaps it is corrupted).

If the system detects failure to communicate with the external accounting system, ...

# Example : Fully Dressed Use Case

## **Use Case UC1: Process Sale**

**Scope:** NextGen POS Application

**Level:** User goal

**Primary Actor:** Cashier

### **Stakeholders and Interests:**

- Cashier: wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.

- Salesperson: wants sales commission updated.

...

**Preconditions:** Cashier is identified and authenticated

**Success Guarantee (or Postconditions):** Sale is saved. Tax is correctly calculated. ...

### **Main Success Scenario (or Basic Flow):**

1. Customer arrives at POS checkout with goods and/or services to purchase.

2. Cashier starts a new sale.

...

### **Extensions (or Alternative Flows):**

...

- 1a. Customer or manager indicate to resume a suspended sale.

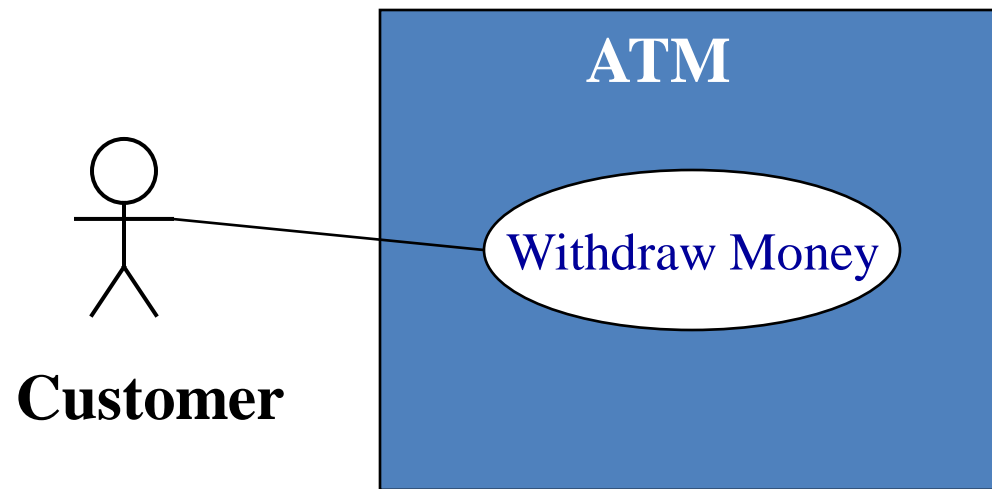
1. Cashier performs resume operation, and enters the ID ...

...

# Use Case Writing Guidelines

- Essential style writing
  - ❖ “Keep the user interface out; focus on intent”
- Write terse use case
  - ❖ Avoid adding unnecessary words
  - ❖ KISS (Keep It Simple and Short)
- Write black-box use case
  - ❖ Emphasise on “what the system must do” rather than “how the system will do it”
  - ❖ E.g. “The system records the sale” vs “The system writes the sale to a database” or “The system generates a SQL INSERT statement for the sale”

# Example Use Case: Withdraw Money



# Withdraw Money

1. The use case **begins when** a client inserts a card into the ATM. The system reads and validates information on the card.
2. The **system** prompts for a personal identification number (PIN). **Client** enters the PIN. The **system** validates the PIN.
3. The **system** asks which operation the client wishes to perform. **Client** selects “Withdraw Money”.
4. The **system** requests the amount of withdrawal. **Client** enters amount.
5. The **system** request the account type. **Client** selects the account type (checking, saving, credit).

# Withdraw Money

6. The **system** communicates with the ATM network to validate the account ID, PIN, and availability of the amount requested.
7. The **system** asks the Client whether a receipt is desired. This step is performed only if there is paper available to print the receipt.
8. The **system** asks the Client to remove the card. **Client** removes the card.
9. The **system** dispenses the requested amount of cash.
10. The **client** takes the cash.
11. The **system** prints a receipt, if required, which **ends the use case**.



# Examples

|              |   |
|--------------|---|
| Use case:    | Process Sale  |
| Actors:      | Customer (initiator)  |
| Description: | A Customer arrives at a checkout with items to purchase. The Cashier records the purchased items and collects payment. On completion, the Customer leaves with the items. |

# Flow of Events

---

## **Actor Action**

Numbered actions  
of the actors

## **System Response**

Numbered description of  
system responses.

# Use Case Specification: “Process Sale”

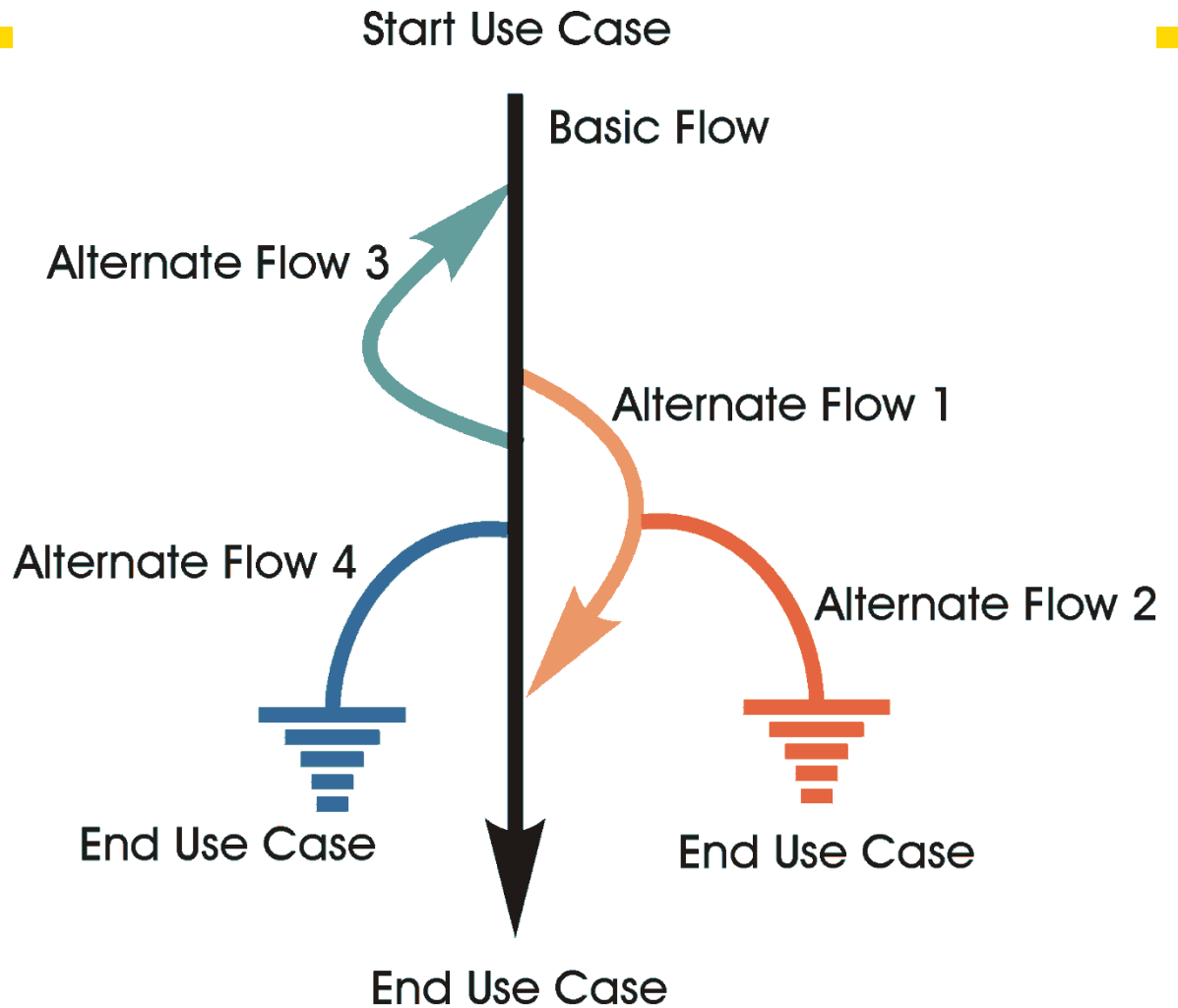
Ref: section 6.10 [2]

| Actor Action   | System Response   |
|--|---|
| 1. Customer arrives at a POST checkout with items to purchase  |   |
| 2. Cashier starts a new sale   |   |
| 3. Cashier records UPC   |   |
|  | 4. Records each sale line item and represents item description and running total. Presents total.   |
| 5. Cashier repeats steps 3-4 until indicates done. Cashier tells Customer the total, and asks for payment. |   |
| 6. Customer pays.  |   |
|  | 7. Handles payment.<br>8. Logs the completed sale and sends information to the external accounting and inventory systems.<br>9. Presents receipt. |

# Notes on UC Specifications

1. A UC description **must include how and when the use case begins and ends**. Also, **consider the possibility of any looping** behavior within the use case.
2. **Be specific** when defining the activities of an actor. Avoid using adverbs (e.g. very, more, rather).
3. UC **should not include technology considerations**. It's not solely created for user interface specification. A UC should not contain any user interface information (e.g. radio button, check box, drop-down menu).
4. **Verify that all the functional requirements have been addressed** in the use case model. Scrutinize the specification carefully to ensure that all requirements have been met by the various use cases.

# UC Scenario



# Basic flow:

- What event starts the use case?
- How does the use case end?
- How does the use case repeat some behavior?

# Alternate flow:

- Are there optional situations in the use case?
- What odd cases might happen?
- What variants might happen?
- What may go wrong?
- What may not happen?
- What kind of resources can be blocked?

# Core Elements of Use Case Specification

- **Use Case Name**
- **Use Case Description**
- **Primary Actors**
- **Secondary Actors** (if applicable)
  - ❖ Actors that participate but are not the main players in a use case's execution.
- **Preconditions**
  - ❖ What needs to happen before the use case can be executed
- **Post-conditions**
  - ❖ What needs to happen after the use case is executed
- **Base Use Cases** (if this use case is a specialization of a certain use case)
- **Flow of Events**
  - ❖ **Basic Flow**

The place to describe each of the important steps in a use case's normal execution.
  - ❖ **Alternative Flow**

A description of any alternative steps from the ones described in the Basic Flow
- **Relationship** (includes or extends) (if applicable)



# Examples:

## Book flight ticket

Use case: Book flight tickets

Actors: Customer (initiator)

Description: This use case enables the customer to book flight tickets

Precondition: The actor access the ticket booking webpage

Postcondition: The system view the booking code

Base Use Case: N/A

Relationship (Includes/Extends): N/A

Flow of events:

# Use Case Specification: “Book flight tickets”

Ref: section 6.10 [2]

Basic Flow:

| Actor Action  | System Response   |
|---|---|
| 1. Select the Ticket Reservation menu                       |   |
|   | 2. View fields: from, to, departure date, return date, adults, children, infants, class, and Search Flights menu. |
| 3. Fill in the fields, then select the Search Flights menu. |   |
|   | 4. Validate the input then show the available flights with price information.                                     |
| 5. Select flight.   |   |
|   | 6. Ask for passenger information and billing details.   |
| 7. Enter passenger information and billing details.         |   |
|   | 8. Validate transaction then show confirmation of the reserved tickets.   |

## Alternative Flow:

### Alternative for basic flow step 4:

If the user input is not valid, then show notification and return to step 2.

### Alternative for basic flow step 8:

If the passenger information or billing details is not valid, then show notification and return to step 6.

# Example : Book flight ticket

- Use Case: ...
- Description: ...
- Primary Actors: ...
- Secondary Actors: ...
- Pre-conditions: ...
- Post-conditions: ...
- Relationships: ...
- Base Use Case: ...
- Main Flow: ...
- Alternate Flow(s): ...

# Example : Book flight ticket

- Use Case: UC1 Book flight ticket
- Description: This use case describes how a Customer book a flight ticket.
- Primary Actors: Customer
- Secondary Actors: Seat Reservation System. Payment System.
- Pre-conditions: Customer is authenticated and authorised.
- Post-conditions: Ticket(s) booked. Seat(s) reserved. Ticket sales updated. Payment received.
- Relationships: N/A
- Base Use Case: N/A
- Main Flow: ...
- Alternate Flow(s): ...

# Example of main flow

1. Customer searches for a flight.
2. System returns search results.
3. Customer chooses a flight.
4. System asks for confirmation.
5. Customer confirms the chosen flight and provides the number of tickets he/she wishes to book
6. System displays seats arrangement on chosen flight.
7. Customer picks a seat.
8. System asks for confirmation.
9. Customer confirms the chosen seat.
- Customer repeats steps 6 – 9 until indicates done.*
10. System displays total fee and asks for payment.
11. Customer pays.
12. System handles payment via Payment System.
13. Systems logs ticket booking and sends information to Seat Reservation System.
14. System presents proof of sale.

# Example of alternate flows

6a. System detects not enough seats available on chosen flight:

1. System asks Customer to input smaller number than previously provided.
2. Customer inputs smaller number.
3. System displays seats arrangement.

8a. System detects dirty read on a chosen seat:

1. System signals error to Customer
2. System asks Customer to choose another seat

12a. System detects failure to communicate with Payment System:

1. System signals error to Customer.
2. System reverts chosen seat(s) to unreserved.

# Exercises

- Draw a possible use case diagram of SIAK-NG.
  - ❖ Provide at least 2 actors and 4 use cases (excluding login, logout use case)
- Based on resulting use case diagram, pick 2 use cases and create its fully-dressed use case specifications.



Q & A