COMP CO835
Object-Oriented Systems

Week 5

Professor: Dr. Stanley Ukah
Email: Stanley.ukah@mohawkcollege.ca

Outline

- Use Case Descriptions
- Activity Diagrams for Use Cases
- The System Sequence Diagram—Identifying Inputs and Outputs
- SSD Notation
- Use Cases and CRUD
- Relationship types in use case diagrams
- Integrating Requirements Models

Learning Objectives

- Write fully developed use case descriptions
- Develop activity diagrams to model the flow of activities
- Develop system sequence diagrams
- Use the CRUD technique to validate use cases
- Explain how use case descriptions and UML diagrams work together to define functional requirements
- Understand Different relationship types in use case diagrams, and their importance in software projects

Use Case Descriptions

- Use case descriptions provide more information about use cases
  - Written at two separate levels of detail: brief description and fully developed description.

- Detailed information about each use case is described with a **use case description**.

Use Case Descriptions (1 of 2)

- Brief use case descriptions.

Use case	**Brief use case description**
Create customer account	User/actor enters new customer account data, and the system assigns account number, creates a customer record, and creates an account record.
Look up customer	User/actor enters customer number, and the system retrieves and displays customer and account data.
Process account adjustment	User/actor enters order number, and the system retrieves customer and order data; actor enters adjustment amount, and the system creates a transaction record for the adjustment.

Use Case Descriptions (2 of 2)

- Fully Developed Use Case Descriptions
- The fully developed use case description is the most formal method for documenting a use case.
- It helps to understand business processes and systems requirements
- It includes a whole sequence of steps to complete a business process.
    - It indicates the beginning and the end of a use case

Fully Developed Use Case Descriptions

- Write a *fully developed use case description* for more complex use cases
- **Typical use case description templates include:**
- Use case name
- Scenario (if needed)
- Triggering event
- Brief description
- Actors
- Related use cases (<<includes>>)
- Stakeholders
- Preconditions
- Postconditions
- Flow of activities
- Exception conditions

Use Case Description Details (1 of 2)

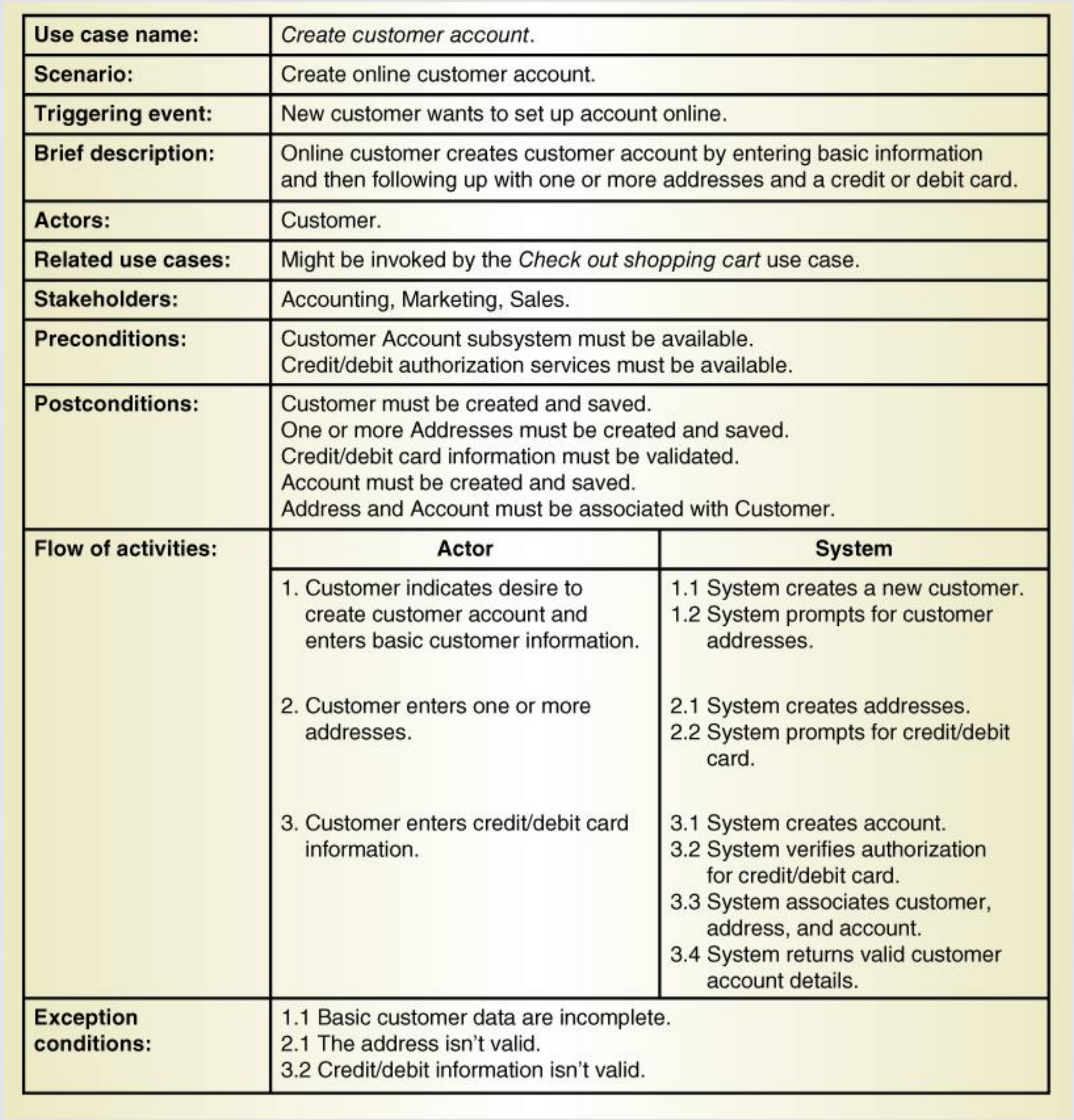
- Use case name
  - Verb-noun
- Scenario (if needed)
  - A use case can have more than one scenario (special case or more specific path)
- Triggering event
  - Based on event decomposition technique or user goal
- Brief description
  - A brief description of the use case
- Actors
  - One or more users from use case diagrams

Use Case Description Details (2 of 2)

- Related use cases <<includes>>
  - If one use case invokes or includes another
- Stakeholders
  - Anyone with an interest in the use case
- Preconditions
  - What must be true or present before the use case begins
- Post conditions
  - What must be true or present when the use case is completed
  - Use for planning test case expected results
- Flow of activities
  - The activities that go on between an actor and the system
- Exception conditions
  - Where and what can go wrong

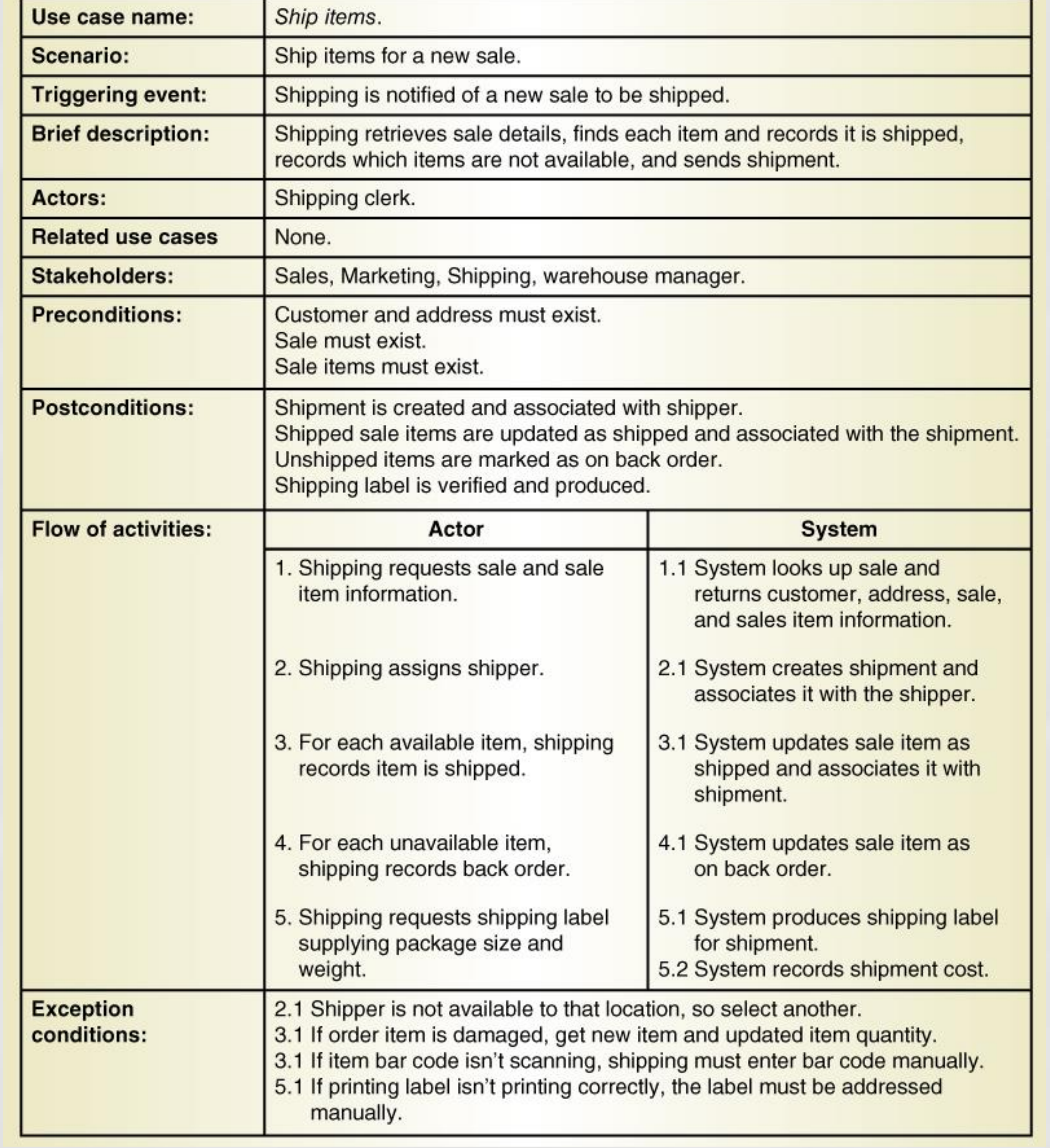
Fully Developed Use Case Description:

Use case: *Create customer account*



Use case name:	*Create customer account*.	
Scenario:	Create online customer account.	
Triggering event:	New customer wants to set up account online.	
Brief description:	Online customer creates customer account by entering basic information and then following up with one or more addresses and a credit or debit card.	
Actors:	Customer.	
Related use cases:	Might be invoked by the *Check out shopping cart* use case.	
Stakeholders:	Accounting, Marketing, Sales.	
Preconditions:	Customer Account subsystem must be available. Credit/debit authorization services must be available.	
Postconditions:	Customer must be created and saved. One or more Addresses must be created and saved. Credit/debit card information must be validated. Account must be created and saved. Address and Account must be associated with Customer.	
Flow of activities:	**Actor**	**System**
	1. Customer indicates desire to create customer account and enters basic customer information.	1.1 System creates a new customer. 1.2 System prompts for customer addresses.
	2. Customer enters one or more addresses.	2.1 System creates addresses. 2.2 System prompts for credit/debit card.
	3. Customer enters credit/debit card information.	3.1 System creates account. 3.2 System verifies authorization for credit/debit card. 3.3 System associates customer, address, and account. 3.4 System returns valid customer account details.
Exception conditions:	1.1 Basic customer data are incomplete. 2.1 The address isn't valid. 3.2 Credit/debit information isn't valid.	

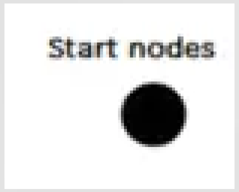
Another Fully Developed Use Case Description Example: Use case *Ship items*

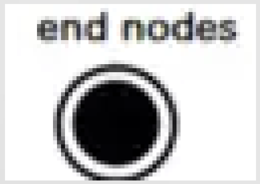


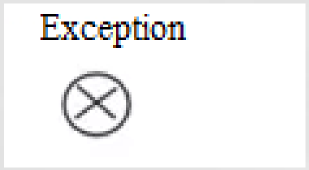
Use case name:	*Ship items.*	
Scenario:	Ship items for a new sale.	
Triggering event:	Shipping is notified of a new sale to be shipped.	
Brief description:	Shipping retrieves sale details, finds each item and records it is shipped, records which items are not available, and sends shipment.	
Actors:	Shipping clerk.	
Related use cases	None.	
Stakeholders:	Sales, Marketing, Shipping, warehouse manager.	
Preconditions:	Customer and address must exist. Sale must exist. Sale items must exist.	
Postconditions:	Shipment is created and associated with shipper. Shipped sale items are updated as shipped and associated with the shipment. Unshipped items are marked as on back order. Shipping label is verified and produced.	
Flow of activities:	**Actor**	**System**
	1. Shipping requests sale and sale item information.	1.1 System looks up sale and returns customer, address, sale, and sales item information.
	2. Shipping assigns shipper.	2.1 System creates shipment and associates it with the shipper.
	3. For each available item, shipping records item is shipped.	3.1 System updates sale item as shipped and associates it with shipment.
	4. For each unavailable item, shipping records back order.	4.1 System updates sale item as on back order.
	5. Shipping requests shipping label supplying package size and weight.	5.1 System produces shipping label for shipment. 5.2 System records shipment cost.
Exception conditions:	2.1 Shipper is not available to that location, so select another. 3.1 If order item is damaged, get new item and updated item quantity. 3.1 If item bar code isn't scanning, shipping must enter bar code manually. 5.1 If printing label isn't printing correctly, the label must be addressed manually.	

UML Activity Diagrams

- The activity diagram is used to represent the flow of activities in a system
- The dark filled-in circle at the beginning represents the start node or the start of the activity
- The dark circle with a white edge at the end represents the end node, or the final activity
- Activity diagrams are helpful when the flow of activities for a use case is complex.

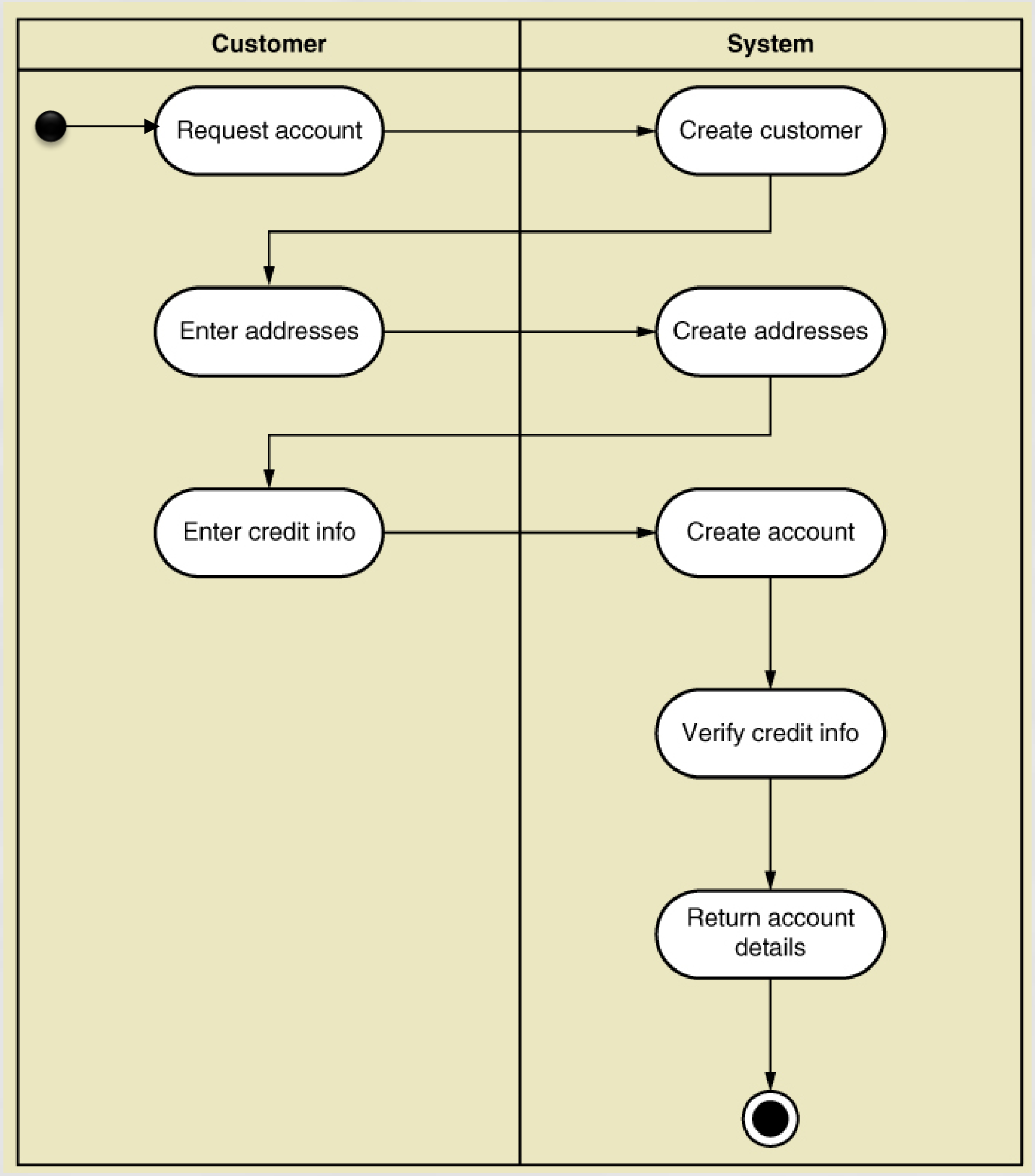






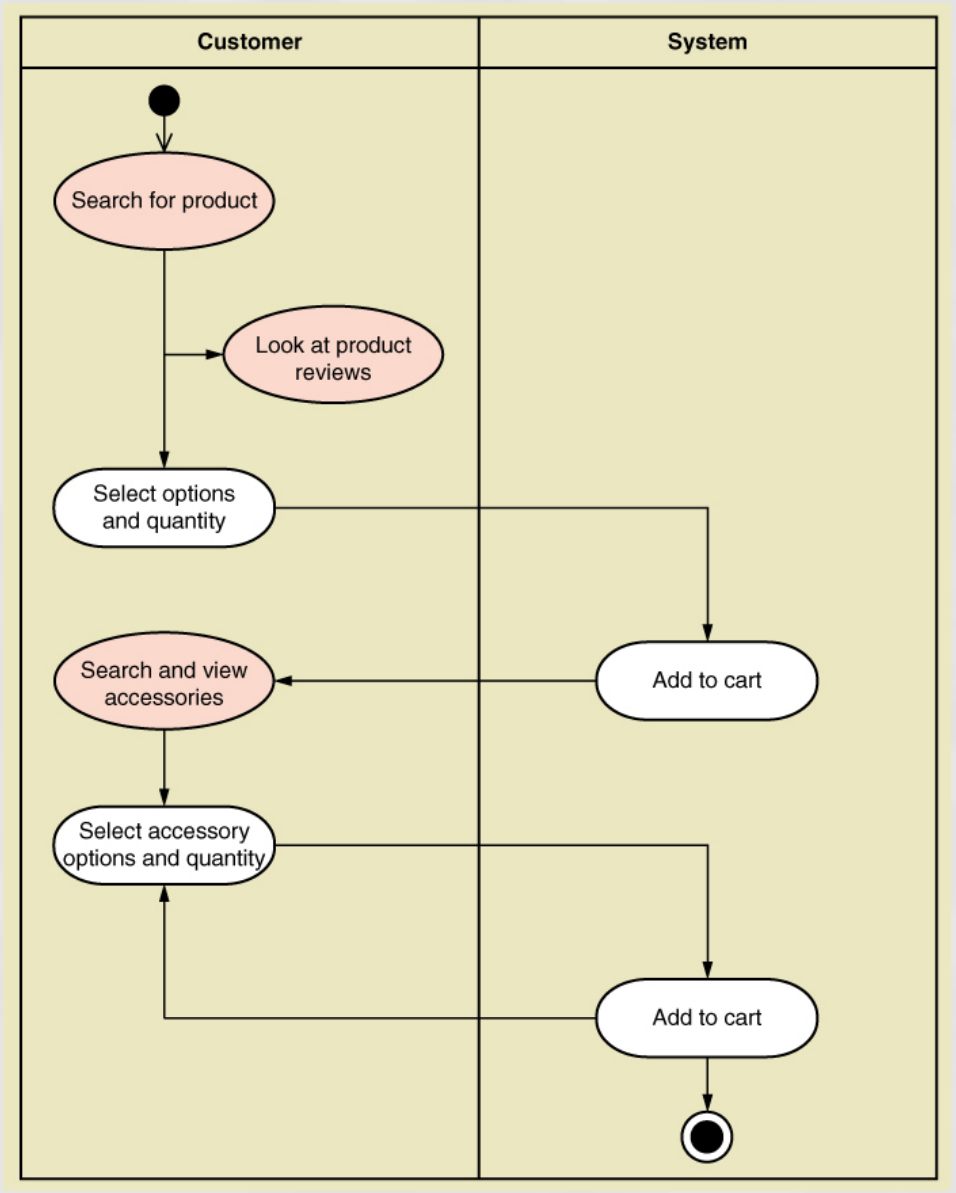
UML Activity Diagram for Use Case: *Create Customer Account*

- Note: this shows flow of activities only



UML Activity Diagram for Use Case: *Fill shopping cart*

- Note: this shows use case with <<includes>> relationship


Customer
System
Search for product
Look at product reviews
Select options and quantity
Add to cart
Search and view accessories
Select accessory options and quantity
Add to cart

System Sequence Diagram (SSD)

- In the object-oriented approach, the flow of information is achieved through sending messages either to and from actors or back and forth between internal objects. A **system sequence diagram (SSD)** is used to describe this flow of information into and out of the automated portion of the system.
- An SSD documents the inputs and the outputs and identifies the interaction between actors and the system.
- It is an effective tool to help in the initial design of the user interface by identifying the specific information that flows from the user into the system and the information that flows out of the system back to the user.

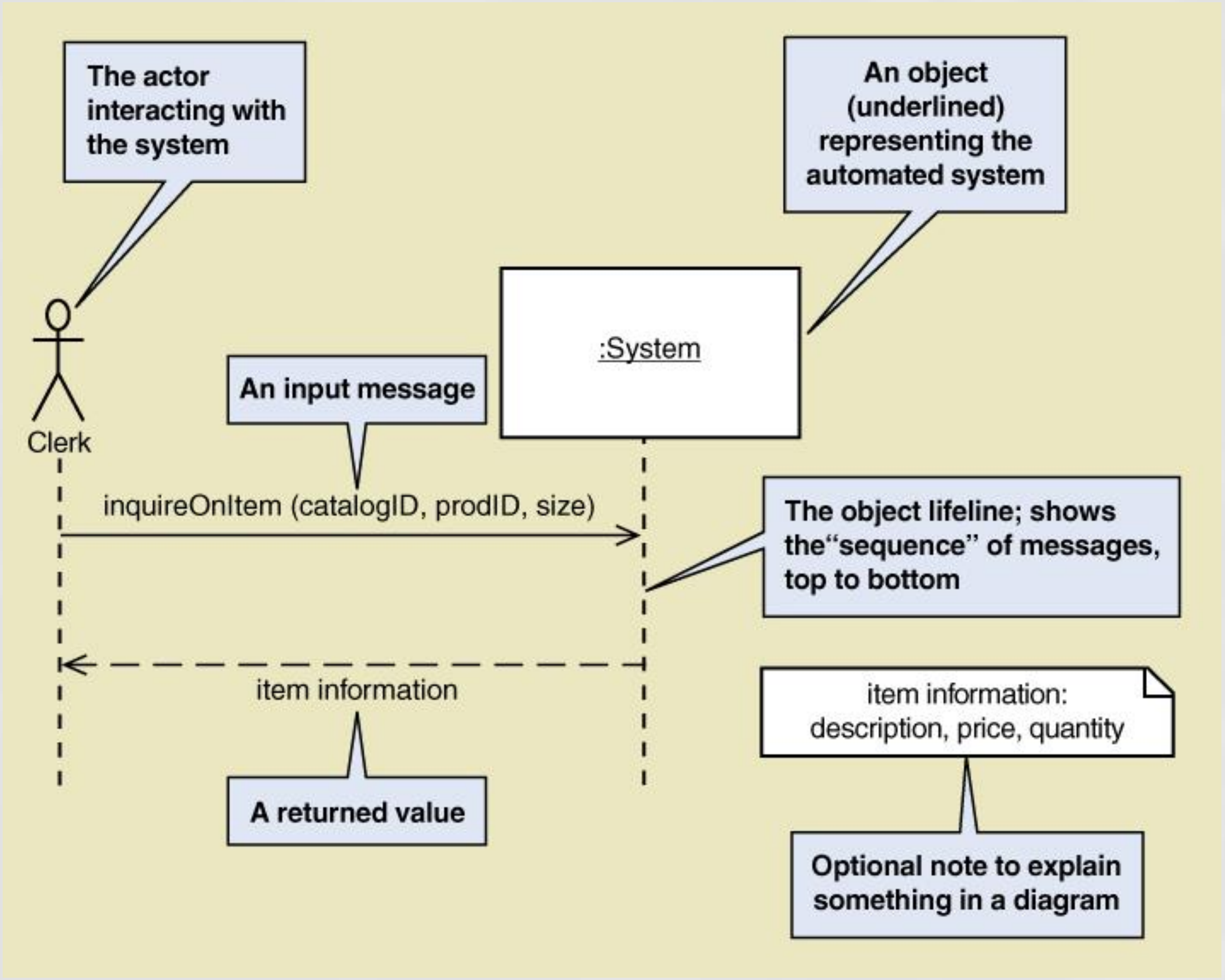
System Sequence Diagram (SSD)

- Special case for a sequence diagram
  - Only shows the actor and one object
  - The one object represents the complete system
  - Shows input and output messaging requirements for a use case
- Actor, : <u>System</u> , object lifeline
- Messages
  - In a sequence diagram, a message is an action that is invoked on the destination object, much like a command. The requested service
- The box labeled **:System** is an object that represents the entire automated system.

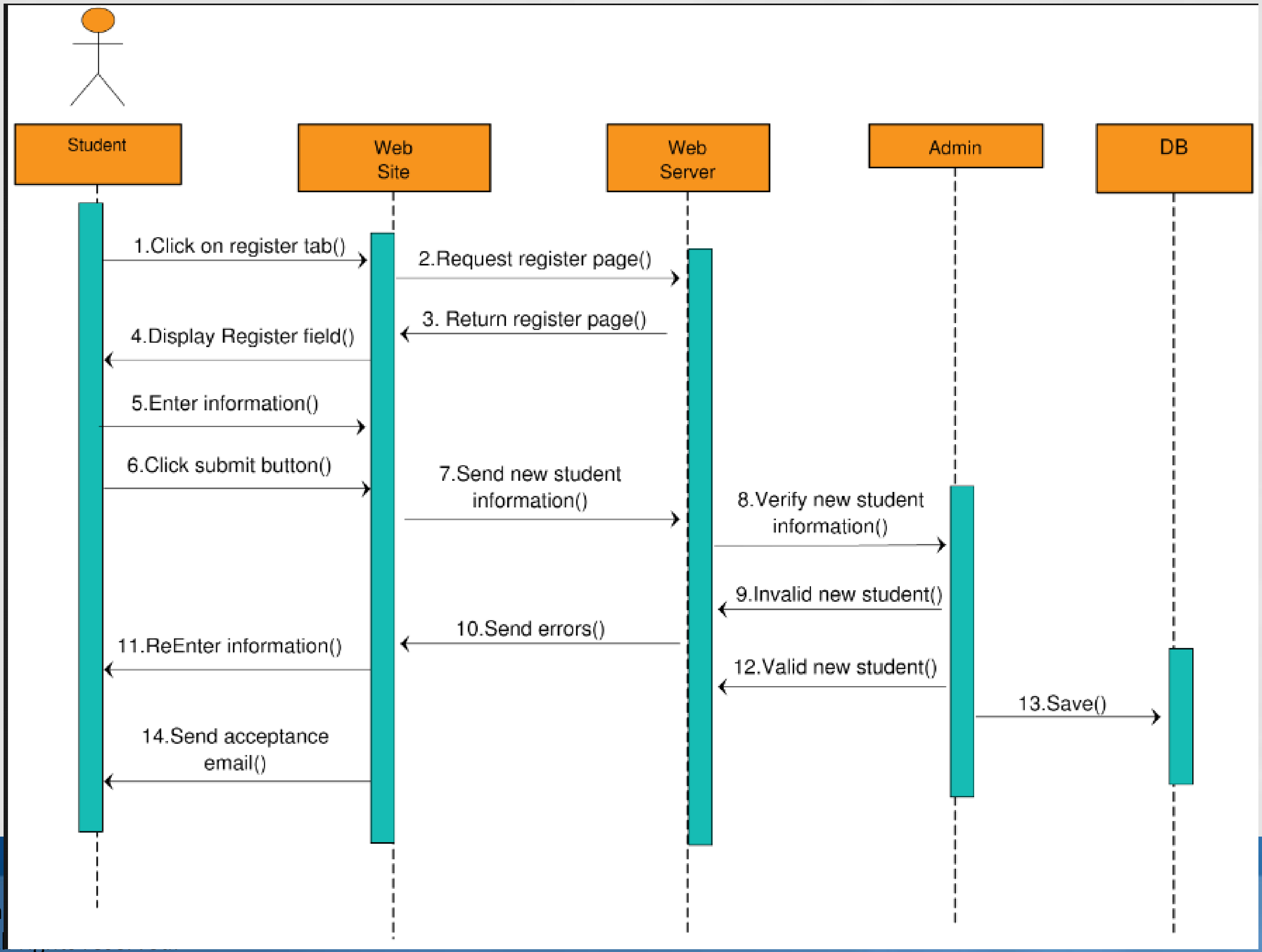
System Sequence Diagram (SSD)

- A lifeline, or object lifeline, is the extension or lifespan of that object—either actor or object—during the use case.
- The arrows between the lifelines represent the messages that are sent by the actor. Each arrow has an origin and a destination.
- The origin of the message is the actor or object that sends it, as indicated by the lifeline at the arrow’s tail.
  - Similarly, the destination actor or object of a message is indicated by the lifeline that is touched by the arrowhead.
- The purpose of lifelines is to indicate the sequence of the messages sent and received by the actor and object.

System Sequence Diagram (SSD) Notation


The actor interacting with the system
An object (underlined) representing the automated system
:System
Clerk
An input message
inquireOnItem (catalogID, prodID, size)
The object lifeline; shows the“sequence” of messages, top to bottom
item information
item information: description, price, quantity
A returned value
Optional note to explain something in a diagram

System Sequence Diagram (SSD)


Student
Web Site
Web Server
Admin
DB
1.Click on register tab()
2.Request register page()
3. Return register page()
4.Display Register field()
5.Enter information()
6.Click submit button()
7.Send new student information()
8.Verify new student information()
9.Invalid new student()
10.Send errors()
11.ReEnter information()
12.Valid new student()
13.Save()
14.Send acceptance email()

Use Cases and CRUD

- CRUD is an acronym that stands for Create, Read, Update, and Delete. It represents the four basic operations that can be performed on data in most database systems or software applications

- CRUD technique –
  - Create
  - Read/Report
  - Update
  - Delete

Verifying use cases for Customer

Data entity/domain class	**CRUD**	**Verified use case**
Customer	Create	Create customer account
BLANK	Read/report	Look up customer Produce customer usage report
BLANK	Update	Process account adjustment Update customer account
BLANK	Delete	Update customer account (to archive)

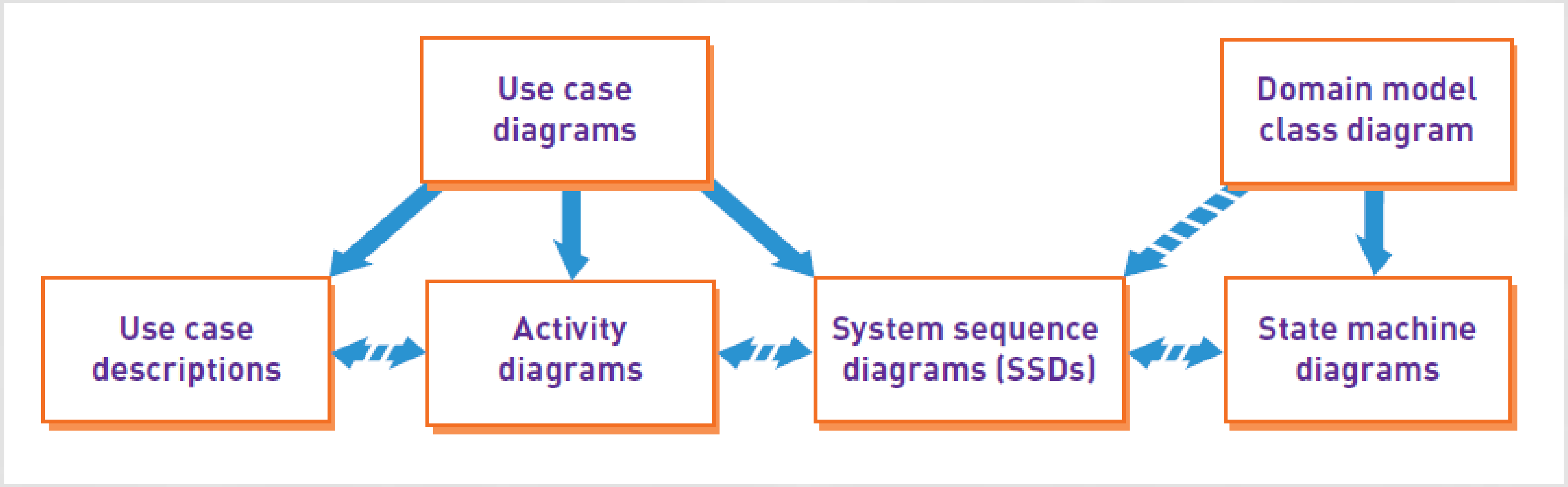
- **Example of use cases and CRUD:** In a web application, a user might use CRUD operations to create a new profile (Create), view their profile information (Read), update their profile picture (Update), or delete their account (Delete).

- CRUD operations specify how data is managed within the system to support use cases.

Integrating Requirements Models

- The use case diagram and the domain model class diagram are the primary models from which others draw information.
- You should develop those two diagrams as completely as possible.
  - The detailed descriptions—either in narrative format or in activity diagrams—provide important internal documentation of the use cases and must completely support the use case diagram.
- The internal descriptions as preconditions and postconditions use information from the domain model class diagram.
- These detailed descriptions are also important for the development of system sequence diagrams. Thus, the detailed descriptions, activity diagrams, and system sequence diagrams must all be consistent with regard to the steps of a particular use case.

Integrating Requirements Models



- The above illustrates the primary relationships among the requirements models for object-oriented development. The use case diagram and other diagrams on the left are used to capture the processes of the new system.
- The class diagram and its dependent diagrams capture information about the classes for the new system. The solid arrows represent major dependencies, and the dashed arrows show minor dependencies.
- The dependencies generally flow from top to bottom, but some arrows have two heads to illustrate that influence goes in both directions.

More on Use Case Specifications

Systems Use Case Specifications

- **Use case name:** Withdraw Cash
- **Brief Description:** This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account.
- **Actors:** Bank Customer
- **Triggers:** A customer would like to withdraw some cash
- **Results:** Cash Withdrawn
- **Preconditions:** There is an active network connection to the Bank. Customer must have an active account. The ATM has cash available.
- **Post conditions:** Customer account balance reduced by amount withdrawn, cash dispensed by ATM

Basic Flow of Events (Systems Use Case specification)

1. The use case begins when Bank Customer inserts their Bank Card.
2. User validation is performed. Information is sent to Bank as a transaction
3. The ATM displays the different alternatives that are available. In this case the Bank Customer always selects "Withdraw Cash".
4. The ATM prompts for an account (Checking, savings)
5. The Bank Customer selects an account.
6. The ATM prompts for an amount.
7. The Bank Customer enters an amount.
8.The Bank Consortium replies with a go/no go reply telling if the
9. Money is dispensed.
10. The Bank Card is returned.
11. The receipt is printed.
12. The use case ends successfully.

Alternative Flows

- Invalid User
  - If the use case ‘’Validate User’’ does not complete successfully, then
  - The use case ends with a failure condition.

- Wrong account
  - If the account selected by the Bank Customer is not associated with the bank card in use, then
  - 1. The ATM shall display the message "Invalid Account – please try again".
  - 2. The use case resumes at ‘’select account’’.

- Wrong amount
  - If the Bank Customer enters an amount that can't be withdrawn from the machine, or not available in the account, then
  - 1. The ATM shall display a the message indicating that the amount must be a multiple of specific bills, and/or ask the Bank Customer to reenter the amount.
  - 2. The use case resumes at select/enter amount.

Alternative Flows (Contd.)

- Amount Exceeds Withdrawal Limit
  - If the Bank Customer enters an amount that exceeds the withdrawal limit, then
  - 1. the ATM shall display a warning message, and ask the Bank Customer to reenter the amount
  - 2. The use case resumes at select/enter amount

- Amount Exceeds Daily Withdrawal Limit
  - If the Bank response indicates the daily withdrawal limit has been exceeded, then
  - 1. The ATM shall display a warning message, and ask the Bank Customer to reenter the amount.
  - 2. The use case resumes at select/enter amount.

Alternative Flows (Contd.)

- Insufficient Cash
  - If the Bank Customer enters an amount that exceeds the amount of cash available in the ATM or possible daily account withdrawal limit, then
  - 1. The ATM will display a warning message, and ask the Bank Customer to reenter the amount.
  - 2. The use case resumes at select/enter amount

Relationship Types in Use Cases

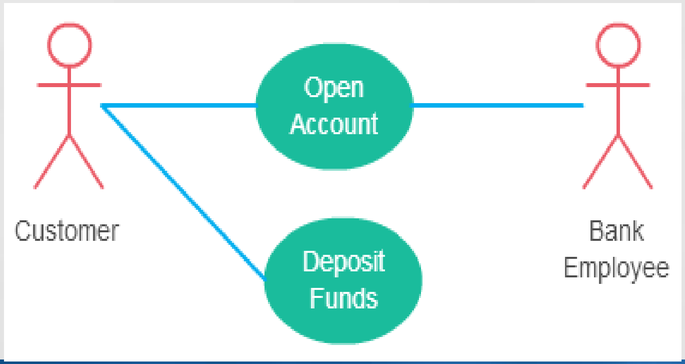
Relationship types in a use case diagrams

There are usually 5 relationship types in a use case diagram.

- Association between actor and use case
- Generalization of an actor
- Extend between two use cases
- Include between two use cases
- Generalization of a use case

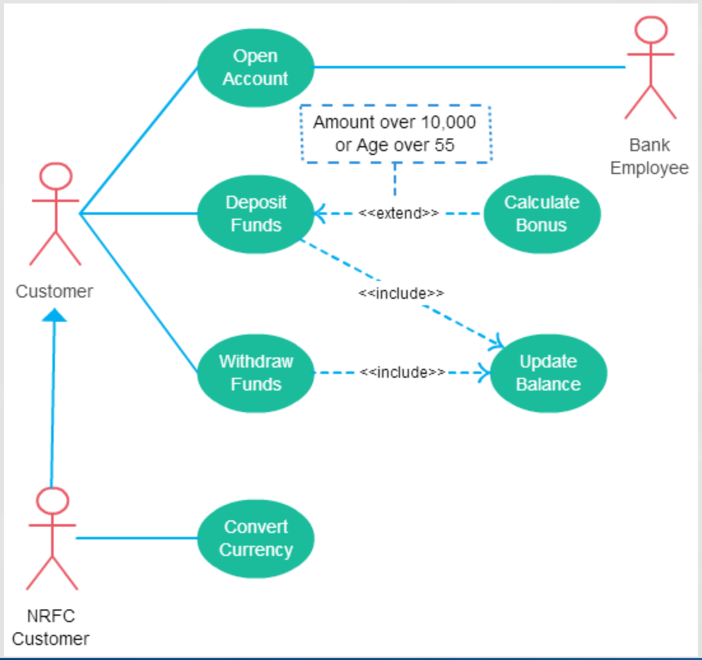
Association Between Actor and Use Case

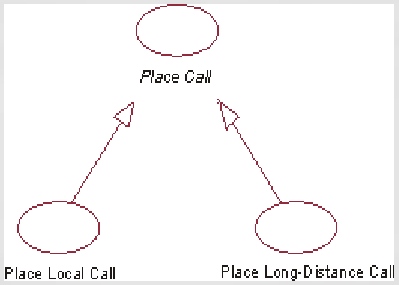
- This one is straightforward and present in every use case diagram.
- Few things to note.
  - An actor must be associated with at least one use case.
  - An actor can be associated with multiple use cases.
  - Multiple actors can be associated with a single use case.

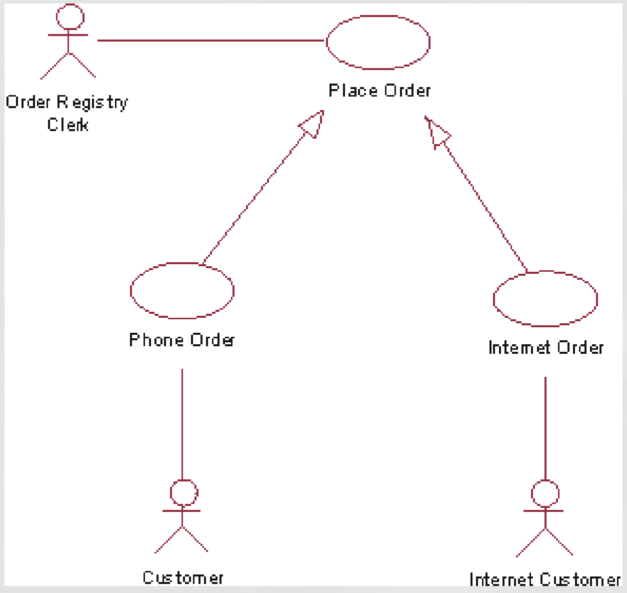


Use Case Generalization

•A use-case-generalization is a relationship from a child use case to a parent use case, specifying how a child can specialize all behavior and characteristics described for the parent.
•Generalization of an actor means that one actor can inherit the role of the other actor. The descendant inherits all the use cases of the ancestor
•A base use case is the use case that includes the functionality of another use case to accomplish a goal



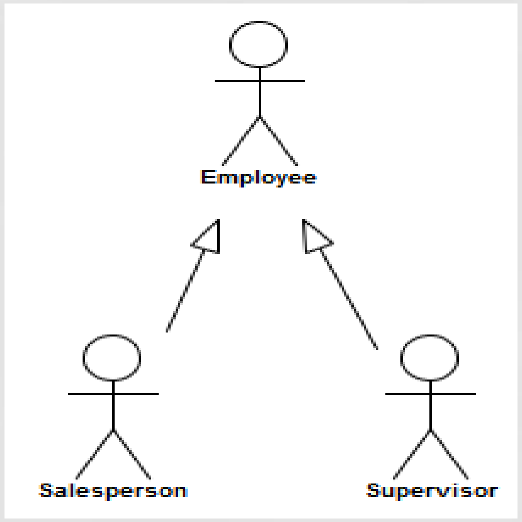


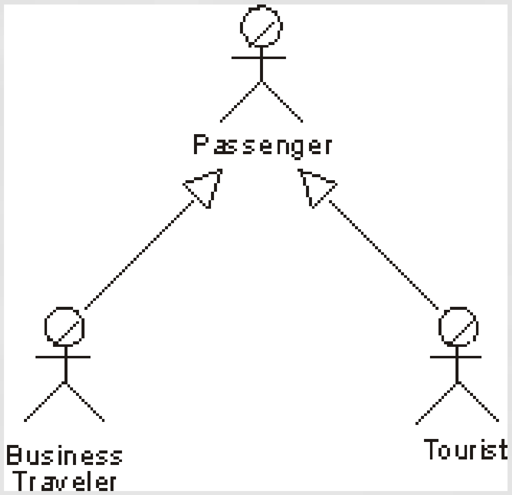


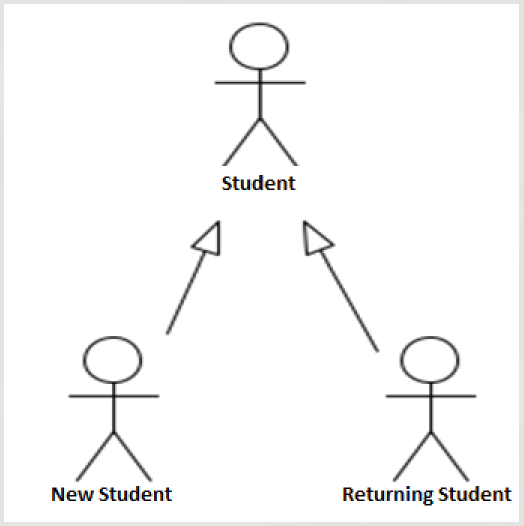
Non-resident foreign currency customer

Actor Generalization

•An Actor Generalization is the relationship which can exist between two actors where one actor (The descendant/child) inherits the role and properties of another actor (The Ancestor/parent)

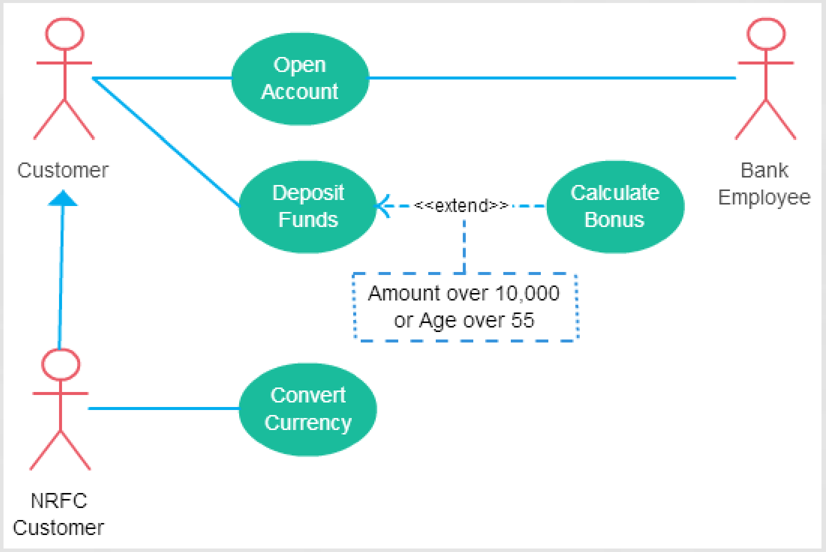






Extend Relationship Between Two Use Cases

•The extending use case is usually optional and can be triggered conditionally.

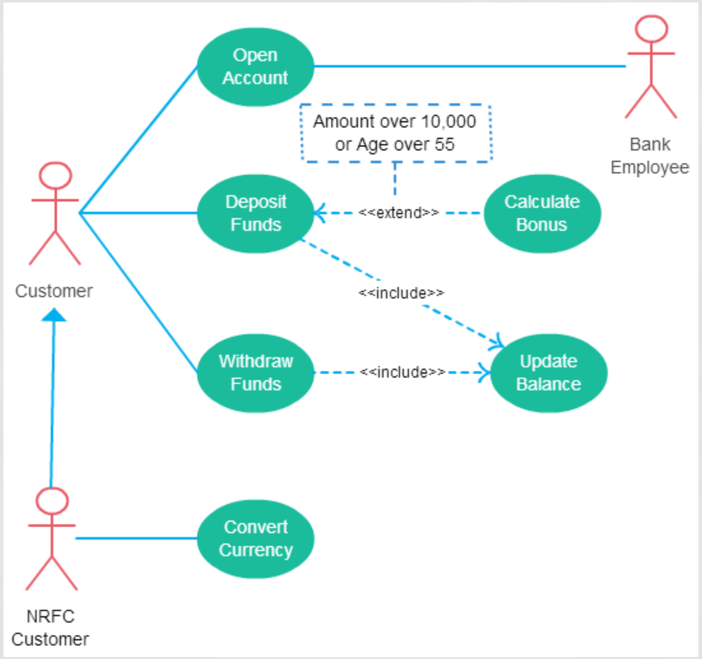


Include Relationship Between Two Use Cases

•Include relationship show that the behavior of the included use case is part of the including (base) use case.

•Few things to consider when using the <<include>> relationship.

- •The base use case is incomplete without the included use case.
- •The included use case is mandatory and not optional.



Non Resident Foreign Currency Customer

More on Include & Extend in Use-Case Diagrams

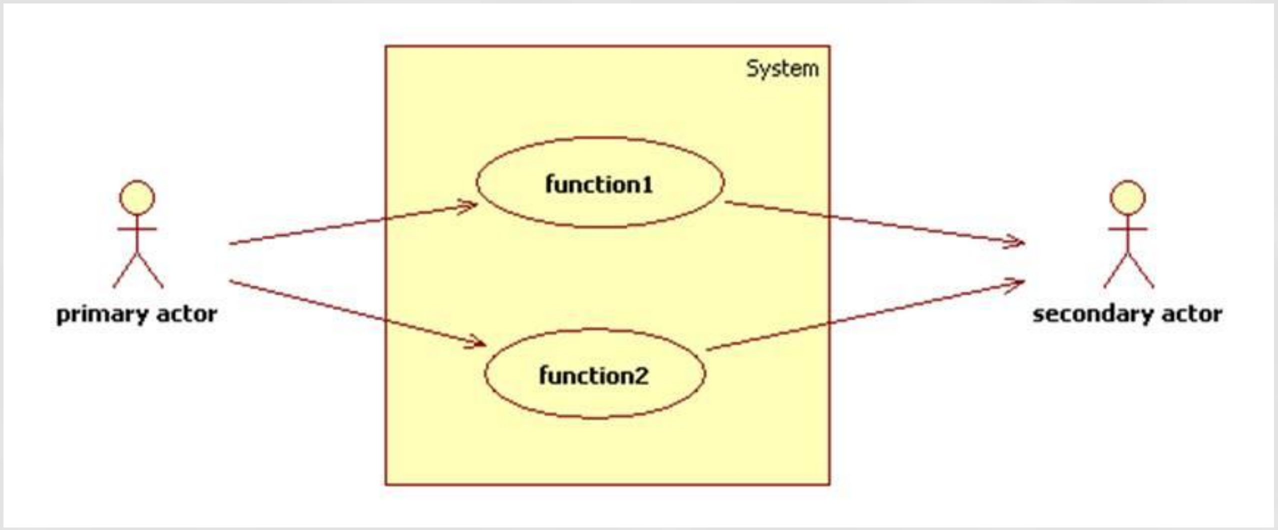
An ‘’include’’ or ‘’extend’’ example using withdrawing cash at the ATM machine

- Include ‘’enter pin’’ to withdraw cash
- Extend ‘’error message’’ if wrong pin is entered

Primary and Secondary Actors

- Actors are classified into primary actors (also called active actors) and secondary actors (also called passive or reactive actors).
- Primary actors initiate a use case and hence are somewhat independent
- They can be people, or other systems that interact with the system
- A secondary actor is one from which the system requires assistance to complete the use case.
- A secondary actor never initiates the use case. It is invoked by the system's use cases in order to obtain information or a result
- Secondary actors are usually reactive, and they can as well be people, or other systems that interact with the system.

Primary and Secondary Actors

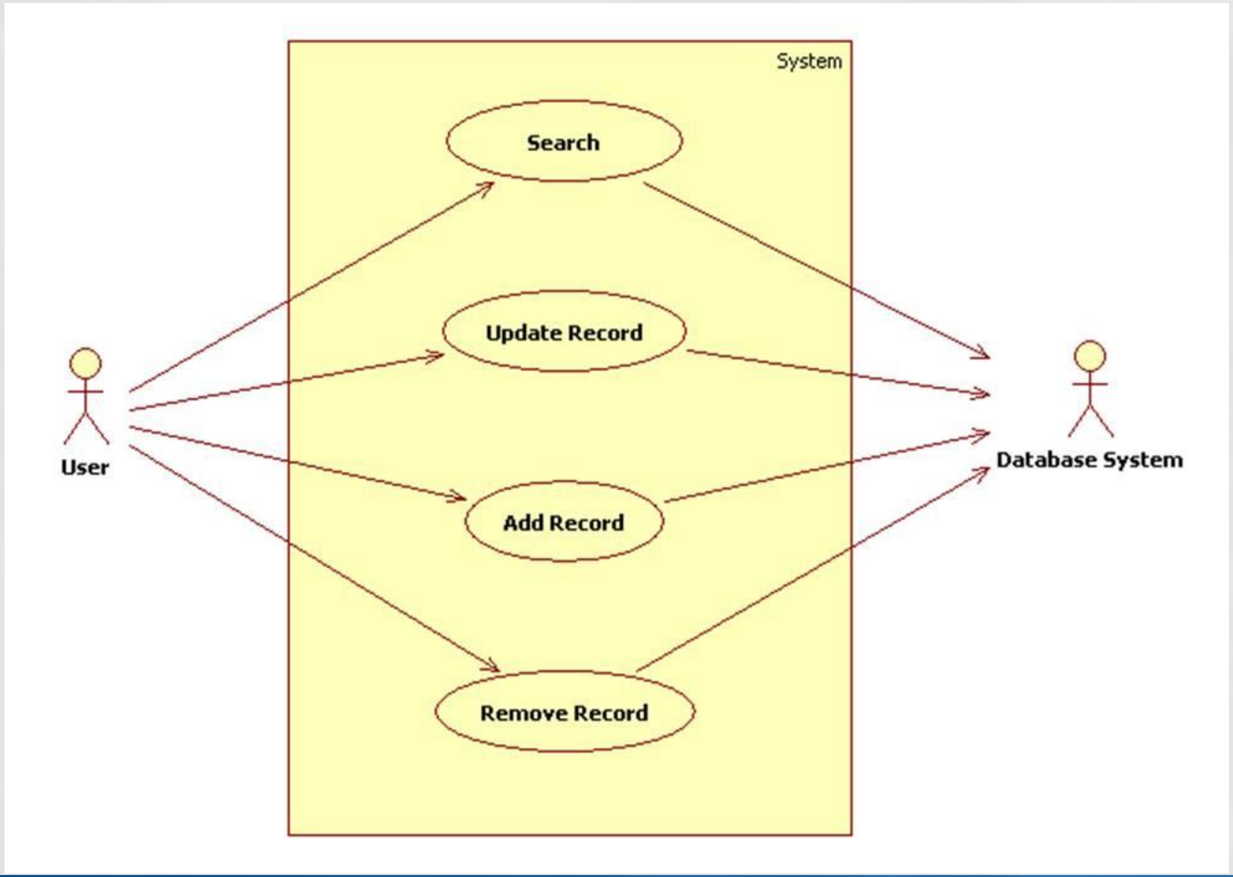


- A conversation can be viewed as an exchange of messages. For example, the system might send a message to the user by displaying a dialog box. The user might reply to this message by clicking a button on the dialog box. Conversations can be long or short. The initiator of the conversation is indicated by the direction of the arrow, which points from the initiator to the responder.

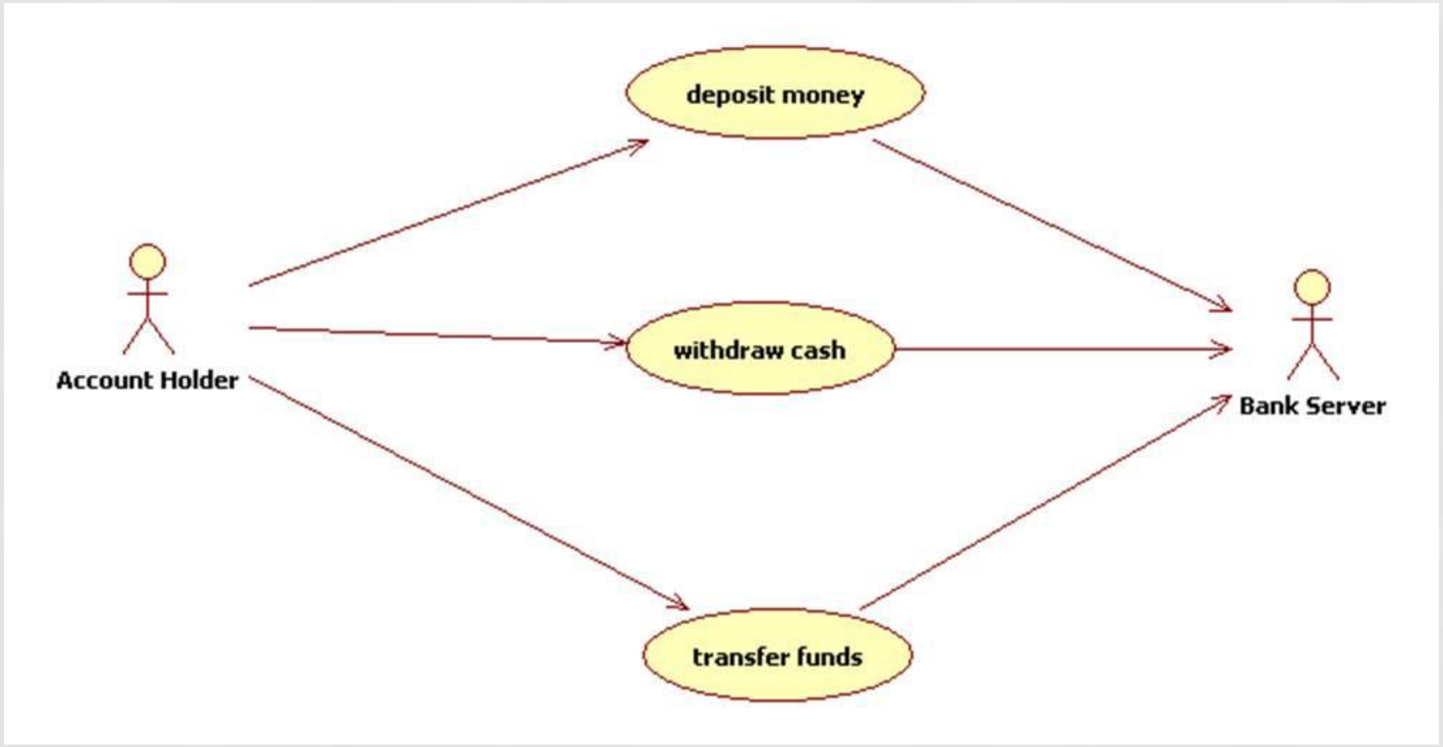
- Primary actors, such as users, always initiate conversations. Secondary actors, such as servers, always respond to an initiating use case.

- The system boundary helps to remind us that use cases are internal to the system and actors are external.

Primary and Secondary Actors Example (A Database Browser)



Primary and Secondary Actors Example (ATM Machine)



Summary (1 of 2)

- In Weeks 3 and 4, we identified and modeled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This week covered the models to provide details of use cases
- Fully *developed use case descriptions* provide information about each use case, including actors, stakeholders, preconditions, postconditions, the flow of activities, and exception conditions
- *Activity diagrams* can also be used to show the flow of activities for a use case

Summary (2 of 2)

- *System sequence diagrams* (SSDs) show the inputs and outputs for each use case as messages
- *CRUD* analysis serves to verify that all domain classes are fully supported by the new system, i.e. have use cases to fully process all required actions
- Not all use cases and domain classes are modelled at a detailed level. Only model when there is complexity and a need to communicate details.
- All of the models must be consistent and integrate together to provide a complete picture of the requirements and specification.

Summary

- There are typically 5 relationship types in a typical use case diagram.
    1. Association between actor and use case
    2. Generalization of an actor
    3. Extend between two use cases
    4. Include between two use cases
    5. Generalization of a use case
- Primary actors initiate a use case, while secondary actors respond to an initiating use case.
- Secondary actors are invoked by the system's use cases in order to obtain information or a result.

Questions?