Unit 3.
Requirement Analysis

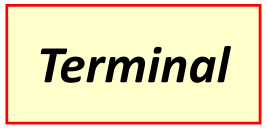
Fall 2019

Soo Dong Kim, Ph.D.
Professor, School of Software
Soongsil University
Office 02-820-0909 Mobile 010-7392-2220
sdkim777@gmail.com http://soft.ssu.ac.kr

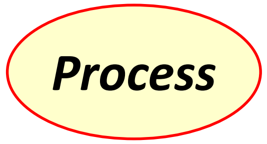
System Context with Data Flow Diagram

Data Flow Diagram (DFD)

- DFD or Bubble Chart
  - Depicts information flow and the transformation that are applied as data move from input to output.
- 4 Major Components
  - Terminal (External Entity)
    - A producer or consumer of information

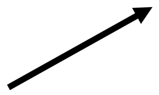


  - Process
    - A transformer of information (a function)

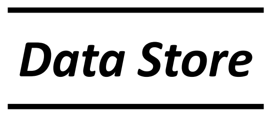


Data Flow Diagram (DFD)

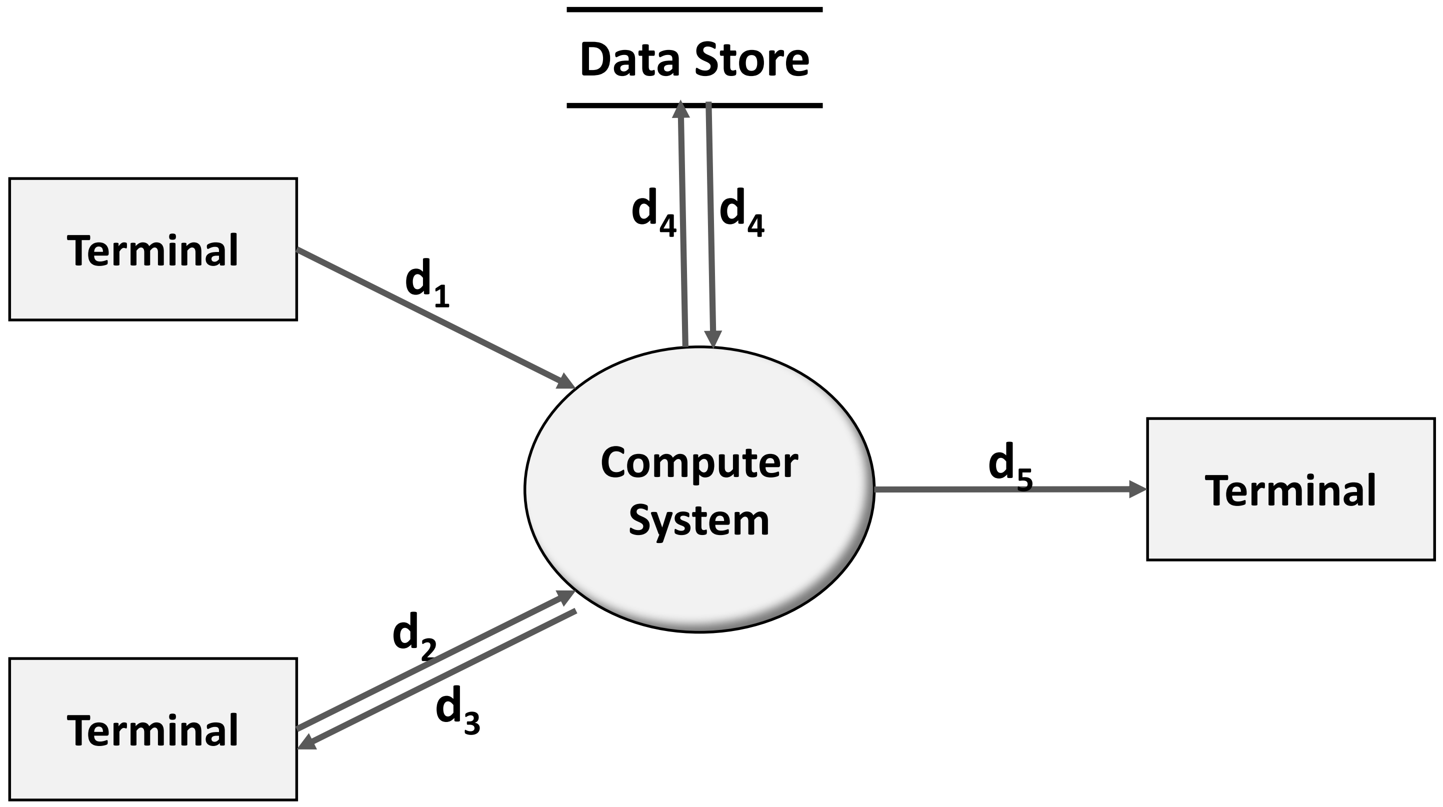
- Data Flow (Data Objects)
  - Arrowhead indicates the direction of Input and Output Data



- Data Store
  - A repository of data that is to be stored for use by processes.
  - May be as simple as a buffer or as sophisticated as a database.



Data Flow Diagram (DFD)



Terminal (External Entity)

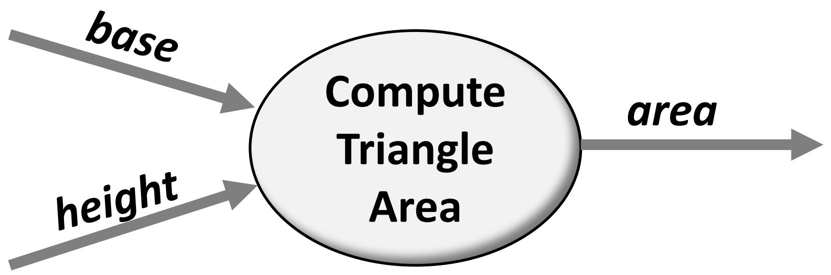
- Producer or Consumer of Data
- Terminal can be;
  - User
    - Consumer, Staff
  - External System
    - Credit Card Authorizer
  - Hardware Device
    - Sensor, Actuator

Process

- **Functionality to Manipulate Data**
  - Receive input, manipulate, and returns an output.
- **Examples**
  - Compute Tax
  - Determine Face Emotion
  - Generate Report
  - Deposit Money

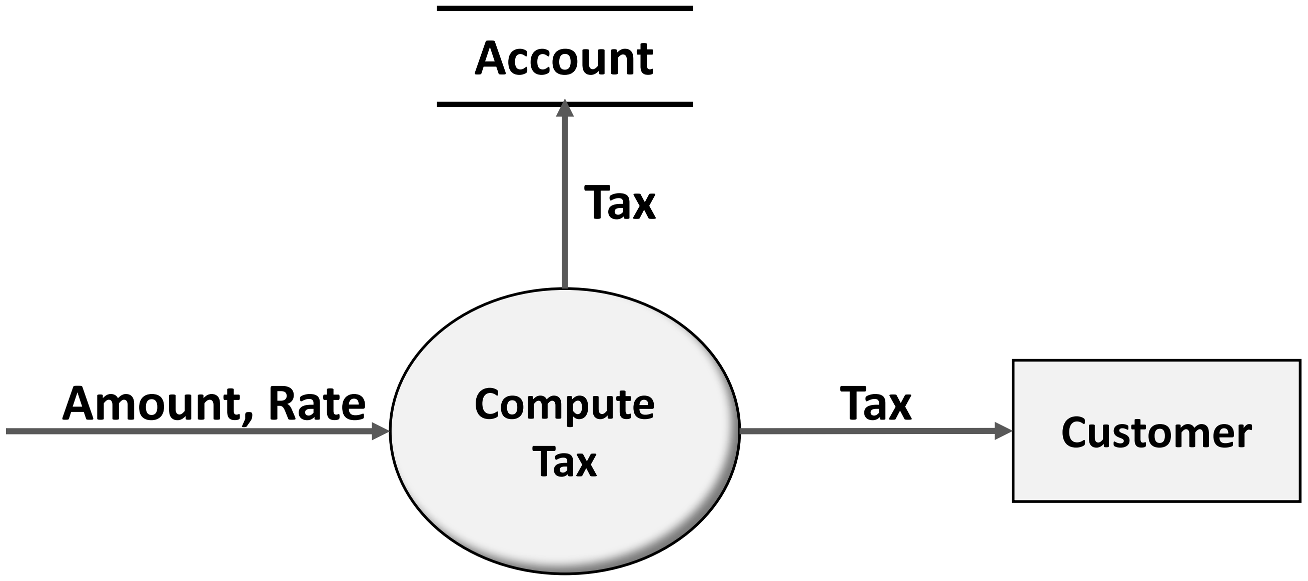
Data Flow

- Flow of Data
- Represented with Arrow and Data Name



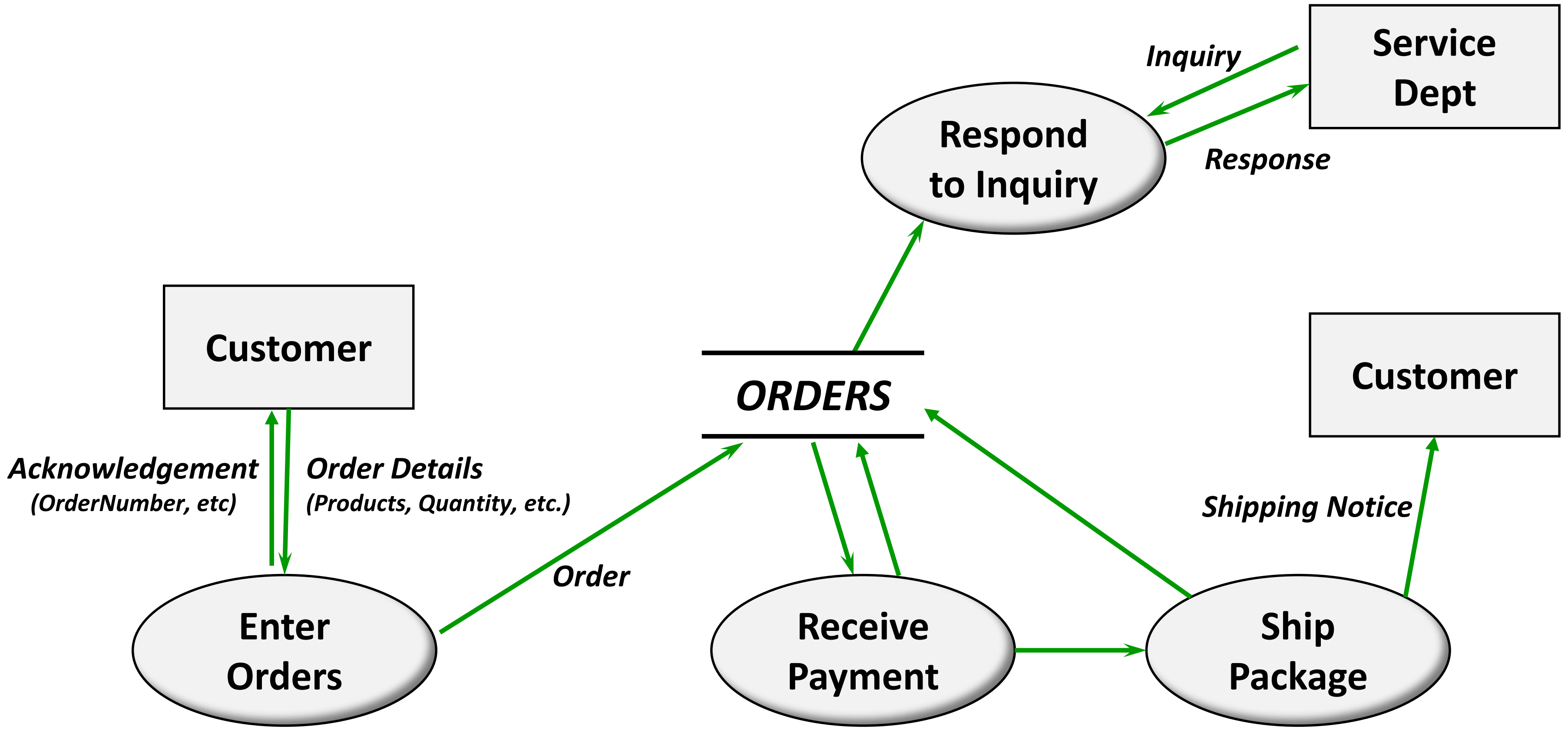
Data Stores

- **Represents a long-lasting data.**
  - **Buffer in Main Memory**
  - **Database Table**



DFD Example

- Order Processing

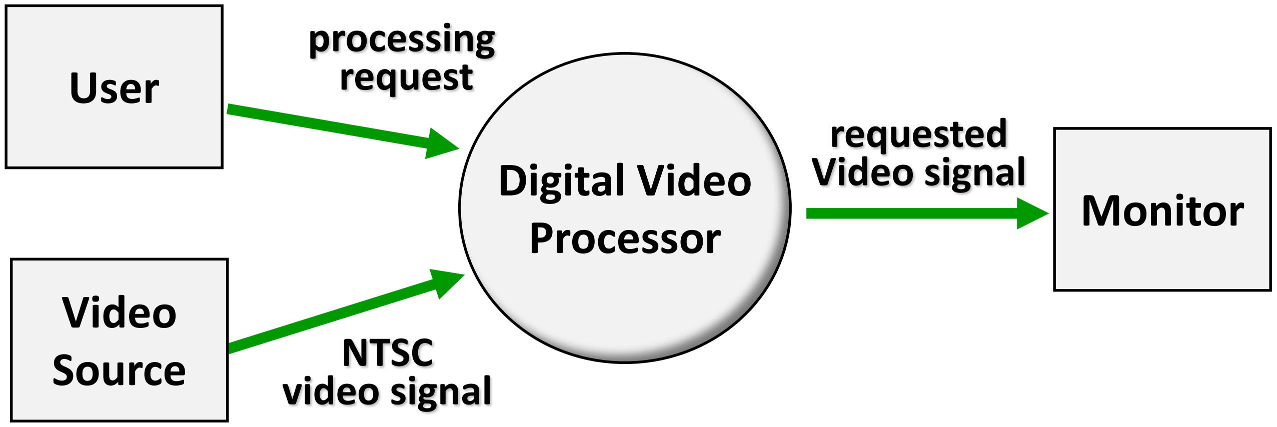


DFD Leveling

- **DFD Levels**
  - A DFD may be partitioned (extended) into levels that represent increasing information flow and functional detail.
- **Level 0 DFD**
  - Fundamental System Model, Context Model/Diagram
  - Represents the retire software as a single bubble with input and output data indicated by incoming and outgoing arrows.
- **Information Flow Continuity (Balancing)**
  - Input and output to each refinement must remain the same.
  - Produce consistent models.

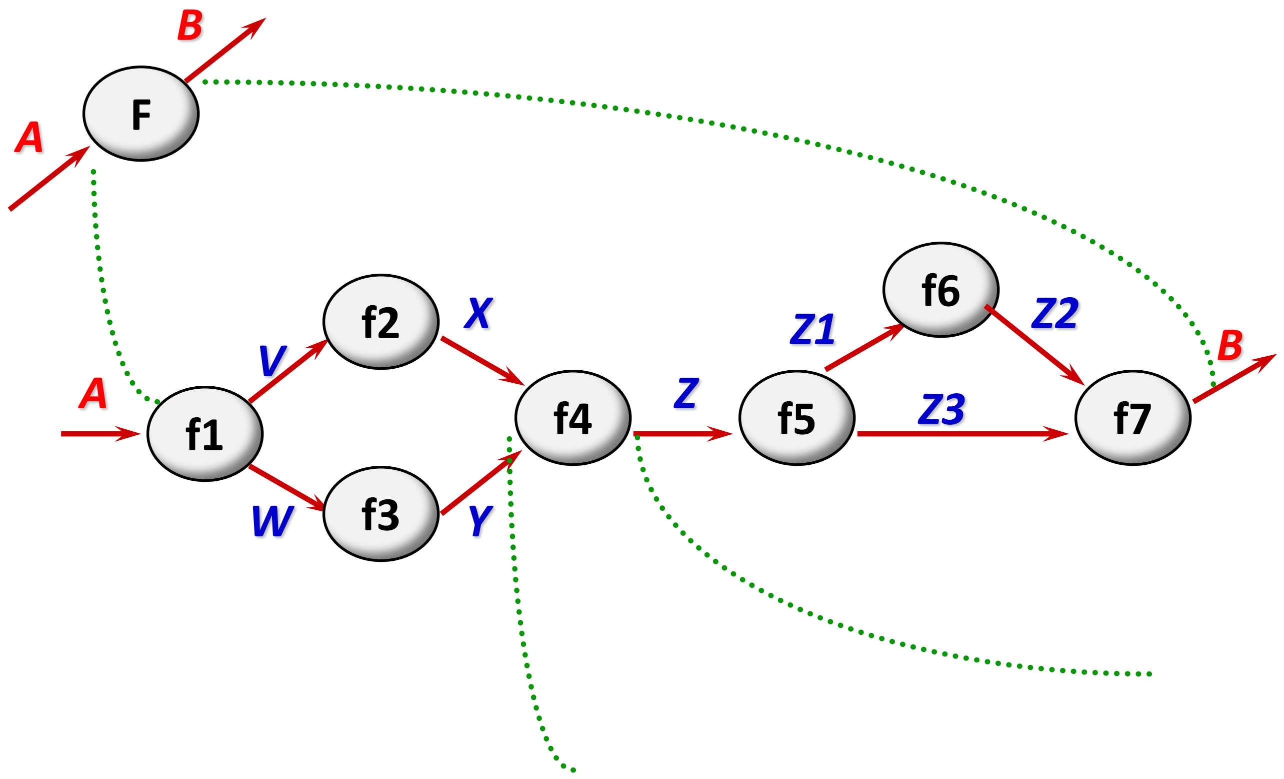
DFD Leveling

- **Level 0, Context Diagram**



DFD Leveling

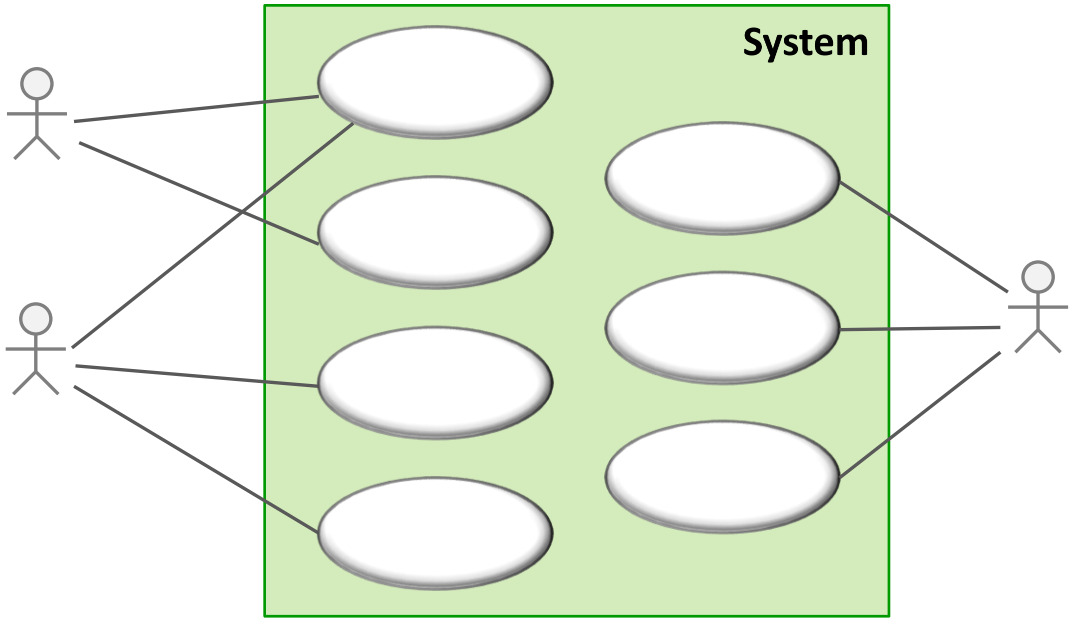
- **Balancing**



Functionality Analysis with Use Case Diagram

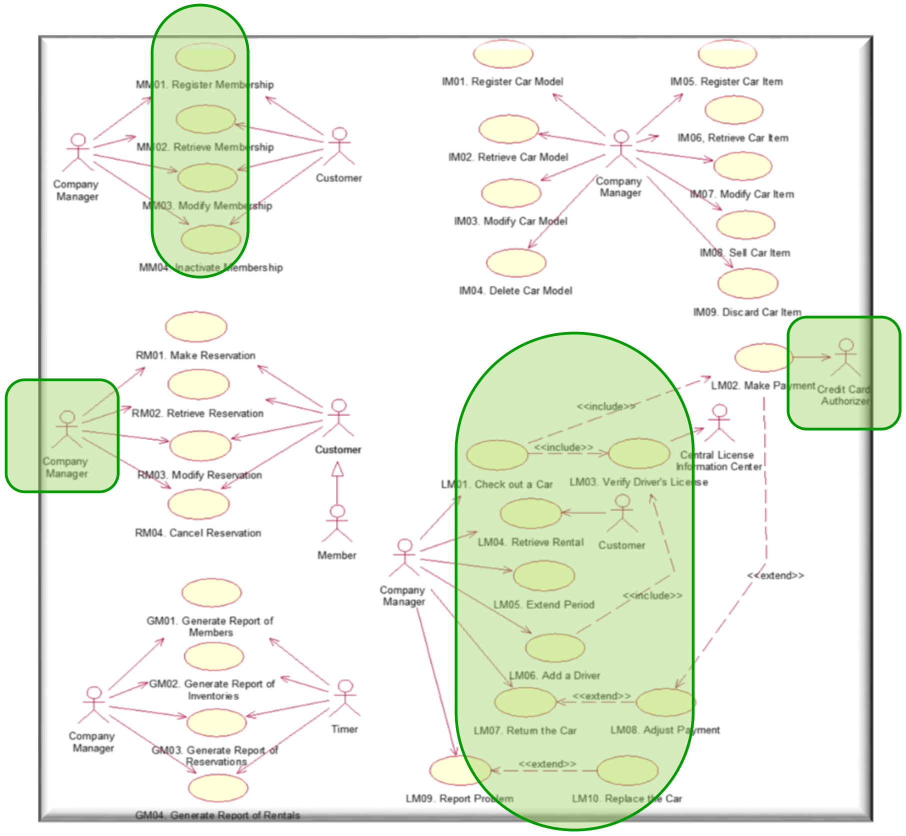
Overview of Use Case Diagram

- To describe the externally observable behavior of a system
  - Exposed Functionality
- To describe the main interactions between the system and external entities including users and other systems
  - Interaction between Actors and System



Use Case Diagram - Example

- **Car Rental System**



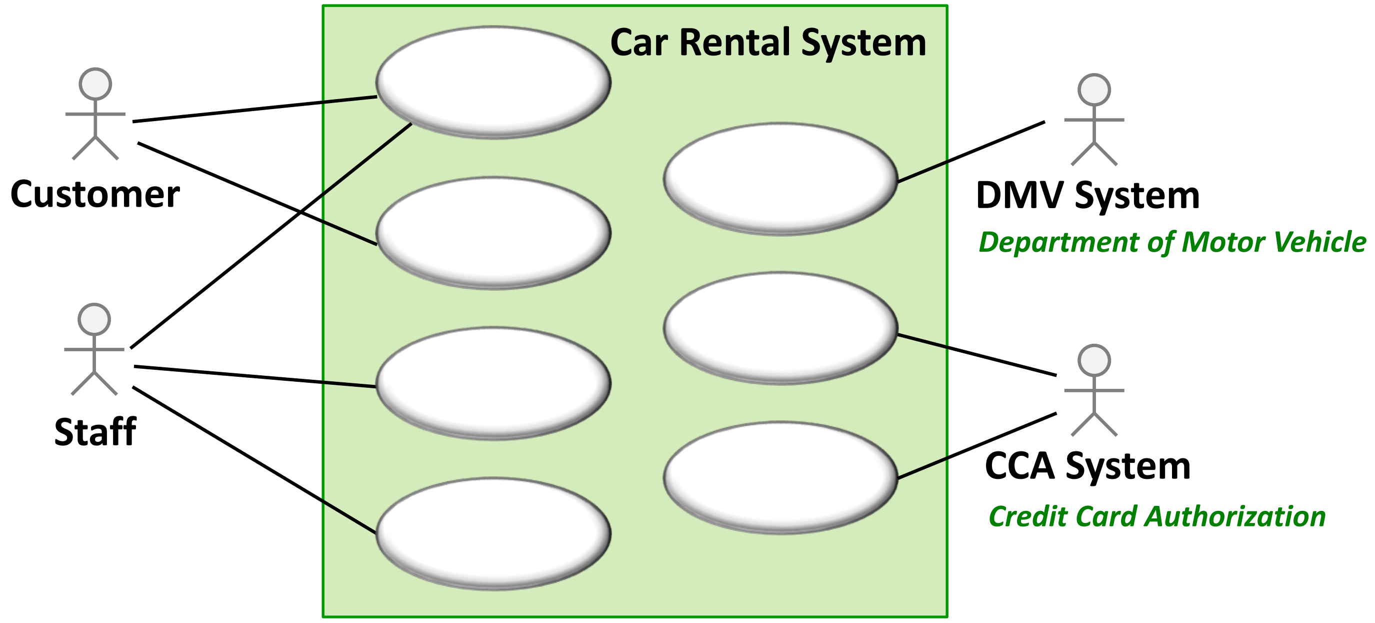
Actor (1)

- A type of role played by an entity that interacts with the subject
- To represent roles played by human users, external hardware, or other subject
- Actor can be;
  - Human User
  - Hardware Device
  - Another System interacting with the system

Customer
Staff

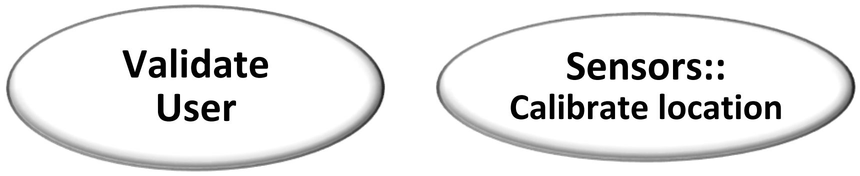
Actor (2)

- **Active Actor**
  - Actor can be active as Trigger
- **Passive Actor**
  - External Systems



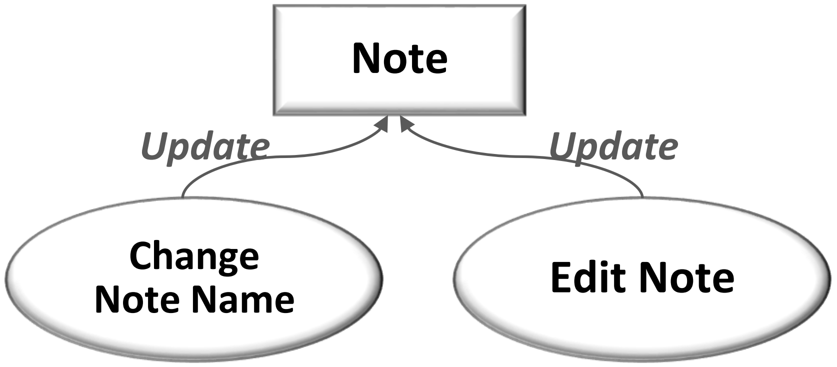
Use Case

- A use case specifies a cohesive functionality of the system.
  - Unit of Functionality
- Use cases are generally larger-grained than
  - A Function in Procedural Program
  - A Method in Object-Oriented Program
- Notation
- Verb Form

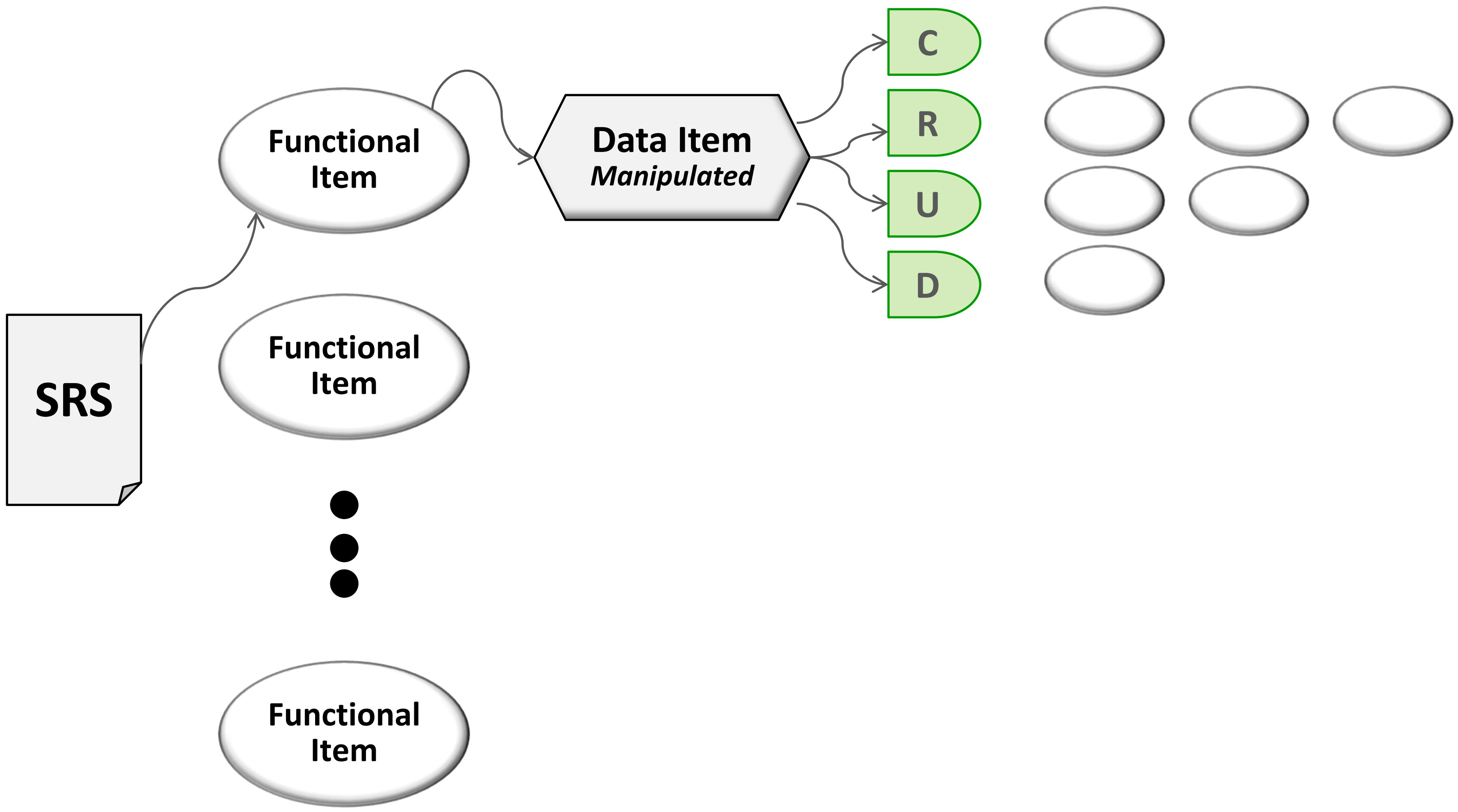


Finding Use Cases (1)

- A use case is a functional unit.
- A function is to manipulate data elements, i.e. objects.
- Consider CRUD-based functions for a target object.
  - Given an object, consider use cases corresponding to CRUD operations on the object.
    - C for Create, R for Retrieve, U for Update, and D for Delete
  - There can be more than one use cases for each of CRUD.
    - For 'U' operation on 'Note' object, two use cases are derived.
      - Change Note Name
      - Edit Note

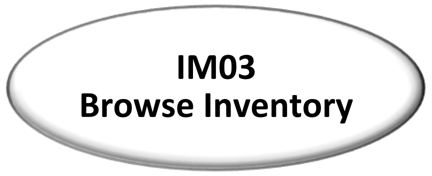


Finding Use Cases (2)



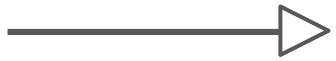
Use Cases for Complex Systems

- Decompose into sub-systems, i.e. packages.
- Apply a numbering scheme.
- Example)
  - ‘IM’ for Inventory Management



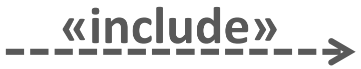
Relationships in Use Case Diagram

- **Generalization**



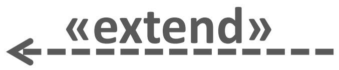
  - A *Base* use case represents the functionality of several *Derived use cases*.

- **Include**



  - A *Base* use case always invoke the *Included* use case.

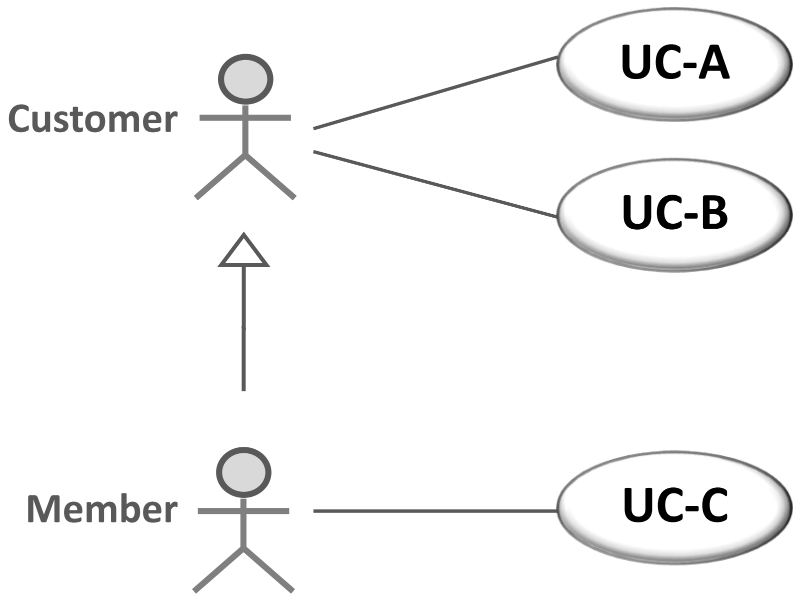
- **Extend**



  - A *Base* use case invokes the *Extended* use case when the given condition is met.

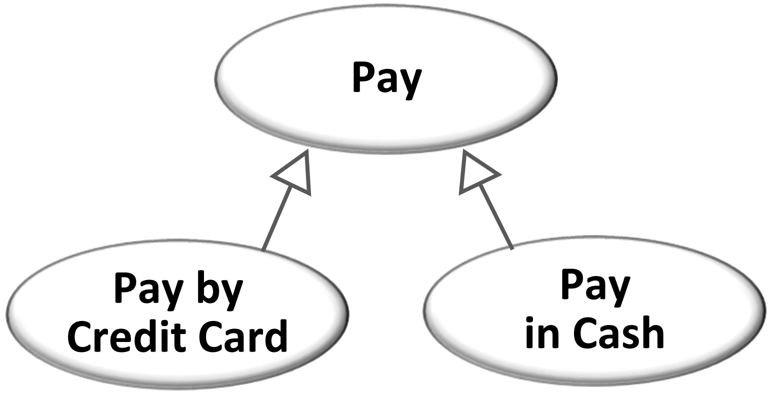
Relationship – Generalization (1)

- Generalization between Actors
  - A child actor inherits the behavior of its parent actor.
  - Example)



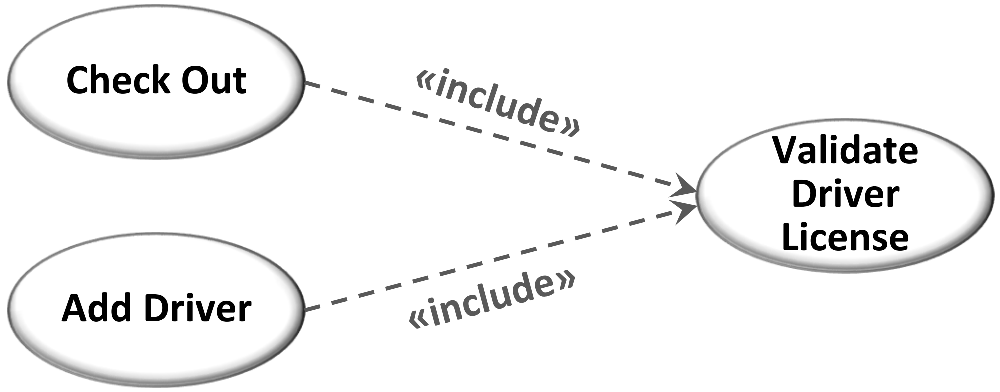
Relationship – Generalization (2)

- **Generalization between Use Cases**
  - **A taxonomic relationship between a base use case and derived use case.**
  - **The base use case captures the common functionality.**
  - **The base use case is *not* implemented.**
- **Substitutability**
  - **At runtime, the base use case is substituted by one of its derived use cases.**
- **Example**



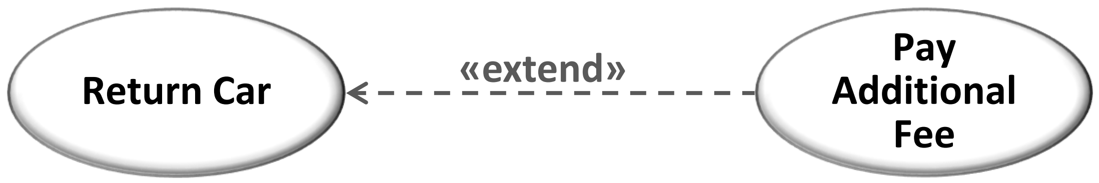
Relationship – Include

- **Include Relationship**
  - **To define a use case that contains common behavior among multiple *base* use cases.**
  - **A base use case always invokes its included use cases.**
- **Example**



Relationship - Extend

- **Extend Relationship**
  - **To define a use case that contains an <u>optional and extended</u> behavior of the *base* use case.**
  - **Extended use cases are invoked only when certain condition is met.**
- **Example**



Use Case Description

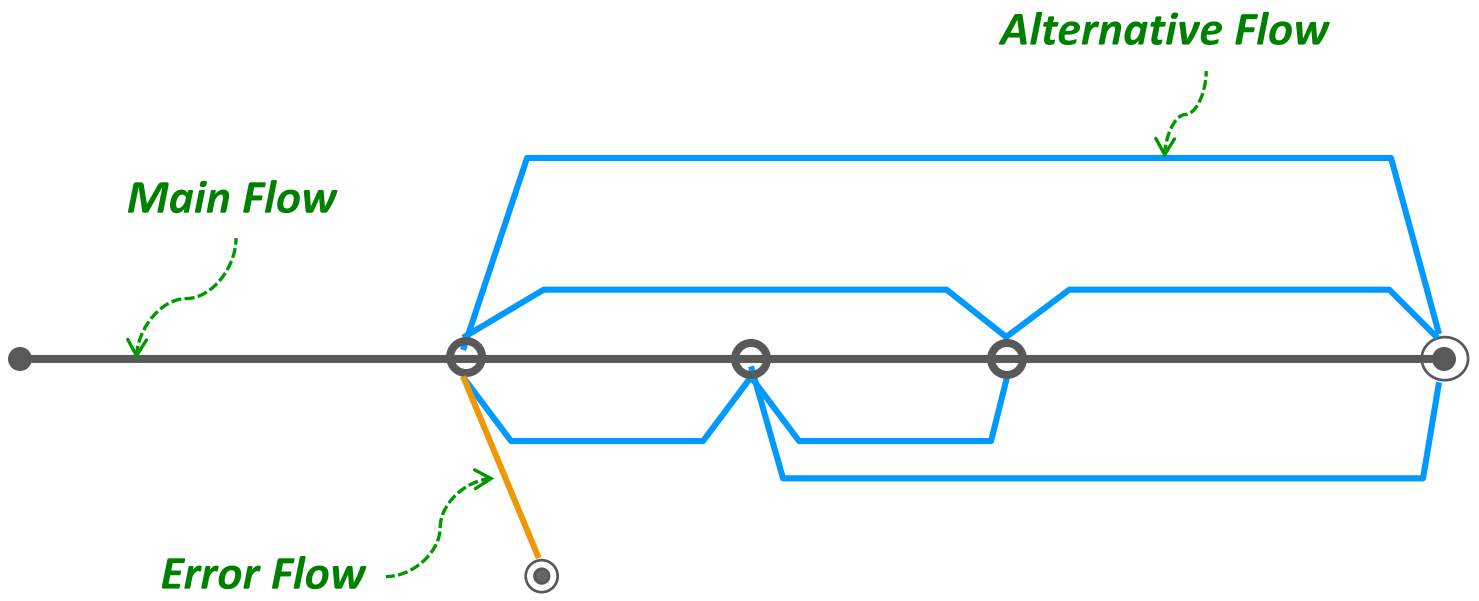
- **Contents**
  - Use Case Name
  - Overview
    - Goal and Scope
  - Pre-condition
    - State what must always be true before a scenario is begun in the use case.
  - Post-condition
    - State what must be true on successful completion of the use case.
  - Flow of Events
    - Main Flow, Alternative Flows, Error Flows
  - Scenarios
    - An Instance of a Use Case
    - Hence, many possible scenarios for a use case may occur.

Internal Flows of a Use Case

- **Main Flow**
  - The most *common*, *general*, and *normal* sequence of tasks to deliver the required functionality
  - The target use case is fulfilled.
- **Alternative Flow**
  - Not the most common, but an *alternative/exceptional* sequence of tasks to deliver the required functionality
  - The functionality of target use case is fulfilled.
- **Error Flow**
  - A sequence of tasks which results in an abnormal termination.
  - The functionality of target use case is <u>not</u> fulfilled.

Internal Flows of a Use Case

- **3 Types of Flows**



Use Case Description

- **Main Flow**

Actor	System
Enters the information of the customer, the driver license, the car model, and the rental period.	
	Checks the validity of the driver license. If it is invalid, invokes E-1.
	Searches for matching cars. It displays available cars, and asks the user to choose a car. If there is no car matching, invokes A-1.
Chooses a car from the list.	
	Asks user to choose an insurance option; "Comprehensive", "Liability Only", "Collision Damage Waiver Only", or none.
Chooses an insurance option.	
	Calculates and displays the fee for rental and insurance. Asks the user to enter a credit card information.
…	

Use Case Description

- **Alternative Flows, A-1**

Actor	System
	Displays "No Matching Car.". Asks the user to choose a different car model.
Enters a different model or 'abort'.	
	If 'abort', terminates the use case. Otherwise, resumes with the different model.

- **Error Flows, E-1**

Actor	System
	Displays "Invalided License".
	Terminates the use case.

Q & A