OOAD

System Analysis
(Part I)

Nguyễn Anh Hào
0913609730 – nahao@ptithcm.edu.vn

System Analysis

- Part I: Outside view (system as a black box)
  - View the intend system as a tool that is used in its environment: understand its role and determine its services in the environment (its usecases and interactions)
- Part II: Inside view (system as a glass box)
  - Determine the system's components, and how they collaborate to solve each use-case.
  - Based on all drawn UML diagrams that describe system's usecases, specify required properties and methods of each system's component (= concept object).

System Analysis

System analysis is the process of decomposing the requirements for the system into specifications of each system component.

Requirements for the system
- Where do the requirements come from?
- Are the requirements determined by the users ?
-...

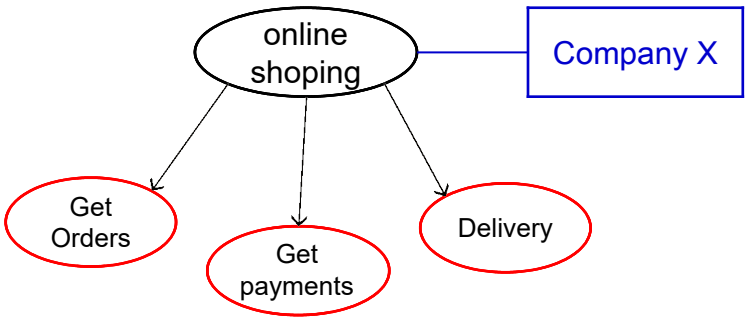
All these questions must be answered precisely.

Object-Oriented Analysis, Design and Implementation,2nd edition,Brahma Dathan&Sarnath Ramnath,Springer 2015 (chp 6, Page 134)

1. System in its working environment

- The system's environment is a larger system (e.g. a company, organisation) that has many problems that people in this environment want to solve, in order to make this working environment better.
  - Example: The company needs to sell goods: what to sell, where to sell, how to sell,.. in the best way?
- The **target system** (the system that we want to build) is a tool that people want to use, in order to solve these problems.
  - Example: The company chooses to sell goods online. So, the sales website needs to be built in order to support sales staff and their customers, such that it is in accordance with government laws, suitable for the online market, suitable for the company's capabilities => These are requirements for the website; they are not determined by the users.

Example: Online Shoping

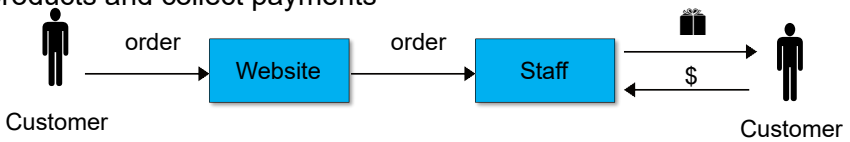


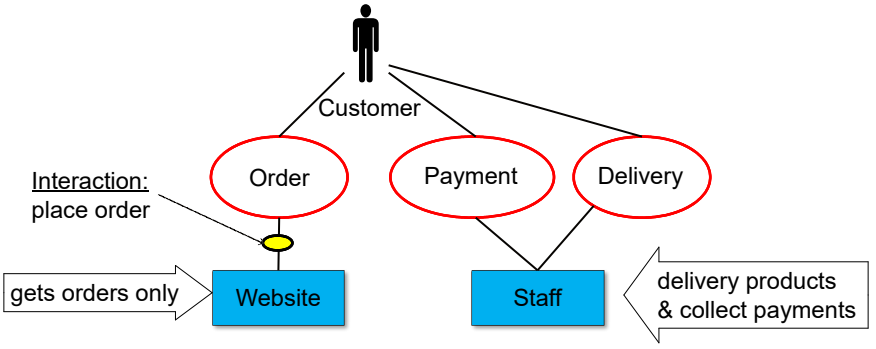
Every sales method has 3 common problems to solve: get orders, get payments, and delivery bought products.

A website can be used to help online shoping.
How will the company X use a website to sell products ?

Online Shoping: case 1

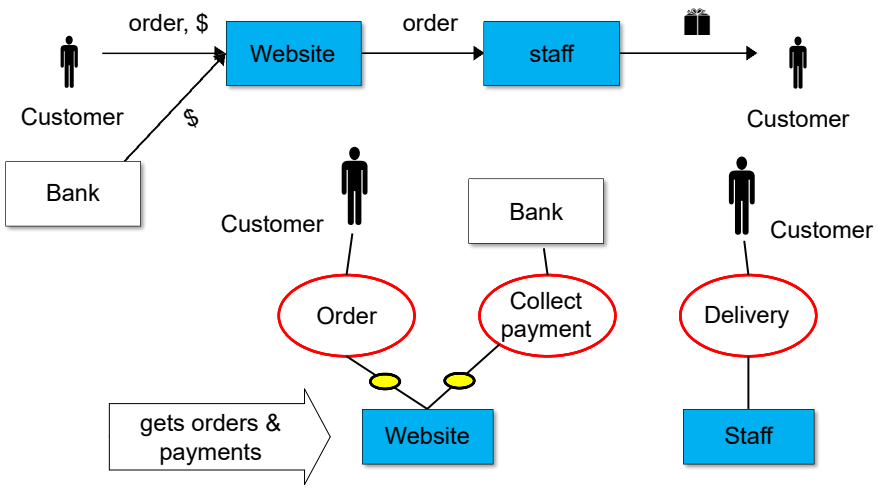
Business rule 1: Customer places orders via website.
Business rule 2: Staff receives orders from website to deliver bought products and collect payments



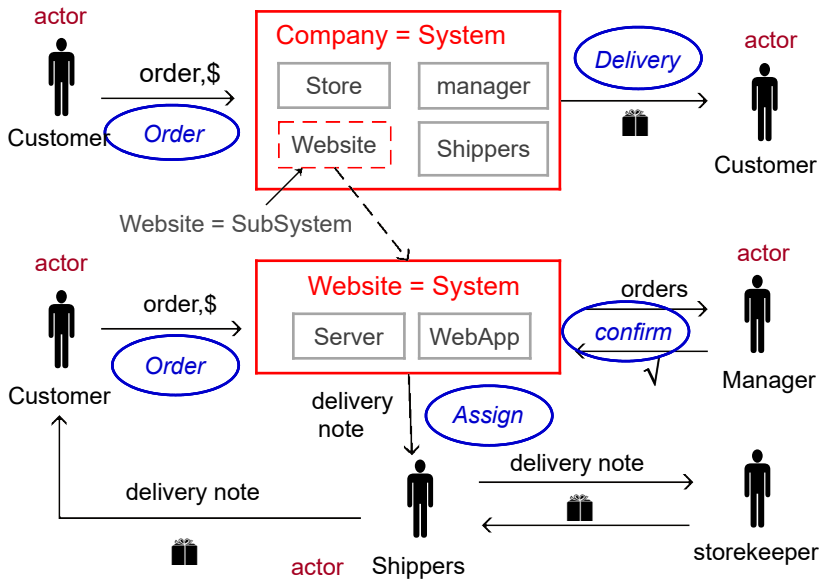


Online Shoping: case 2

Business rule 1: Customer places order via website and pays online.
Business rule 2: Staff receives order & payments from website, then deliver products



Case 2: Online Shoping System



The birth of a system

- The very first question on the intend-created system is: what are the roles and responsibilites of this system in its working environment ?
  - Example: The roles of shoping website is to receive orders in situation 1, or receive orders & payments in situation 2 (depends on shoping model).
  - So that, the system is created as a tool to solve some problems in some situations
- Each situation (has some problems) that the system participate in, is called a system's use case (a case in which the system is used to solve the problems).
- Objects that interact with the system in order to solve the use case are called actors.
  - Customers, manager, shippers are actors of the website.

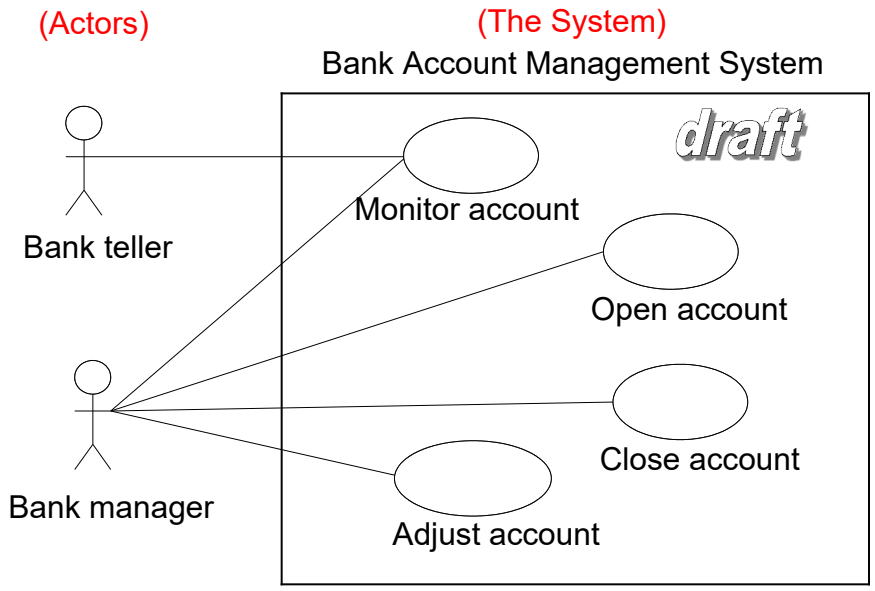
2. System modeling: UML

- UML stands for **Unified Modeling Language**
  - It is a collection of graphical notations,
  - It is a visual modeling language.
- It provides a standard way to represent complex system architecture and behavior (modeling).
  - Diagramming (usecase, class, …)
  - Grammar, context and semantics
- UML helps teams communicate and explore potential designs.

UML: Use Case

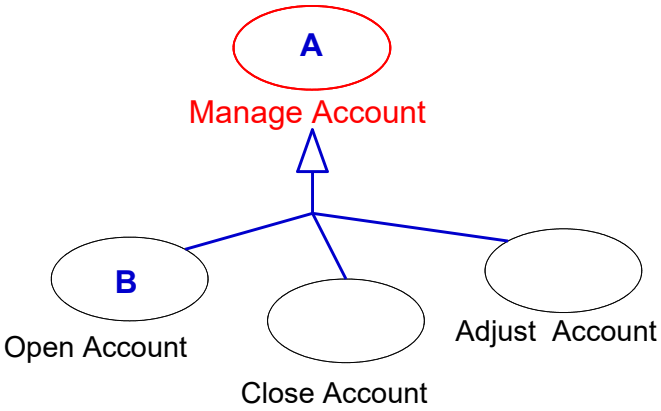
- A situation (tình huống cần giải quyết) that the system participates in, is called a system's use case (a case in which the system is used).
- Objects that use the system (as tool) in order to solve the problems in the situation are called actors.
  - Customers, manager, shippers are actors of the website.
- A use case is used to define requirements on objects (or system) that participate in the use case.
  - A required system function is not a use case.
- Modeling & documenting use case : UML use case

Use case diagram example



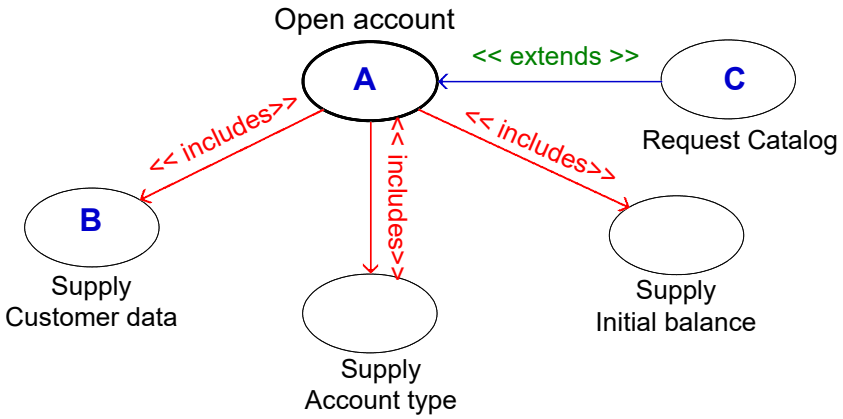
Use case: generalization

- Use case A is a generalisation of use case B if use case B inherits all the actors & behavior of A.



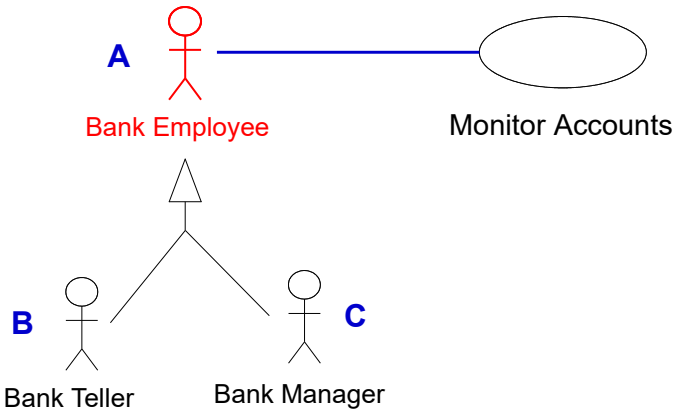
Use case: includes, extends

- Use case A 'includes' use case B when A needs to use use case B (B must be present for A).
- Use case A 'extends' use case C if C is a processing extension of A (A may need C, may not).

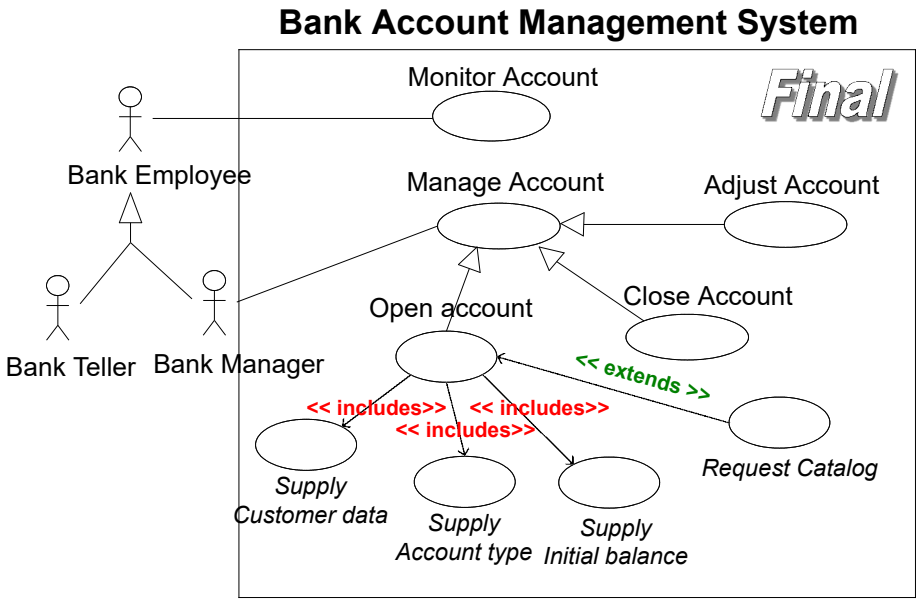


Use case: actor generalisation

If Actor A is a generalisation of actors B and C, then what A can do on the system, actors B and C can also do.

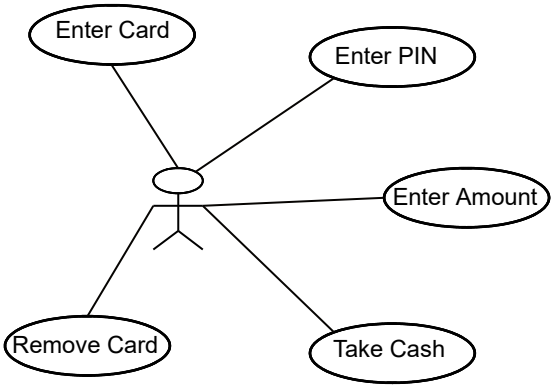


Use case diagram

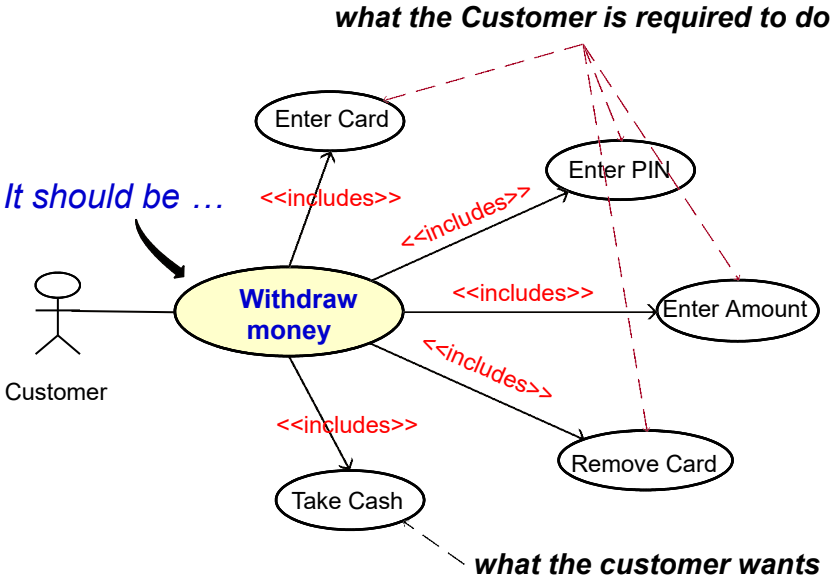


Validate use case: example 1

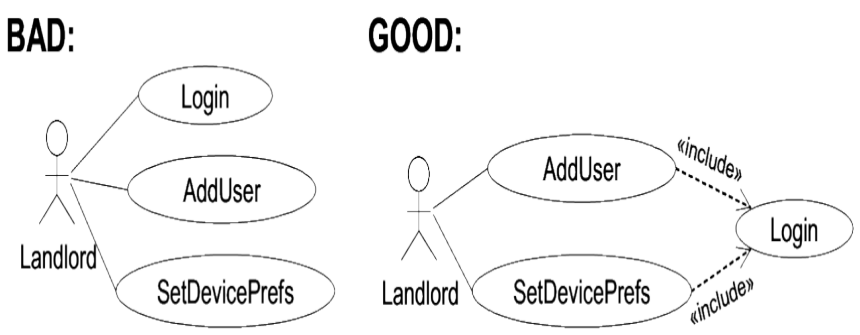
Is this a useful use case diagram for an ATM ?



Validate use case: example 1

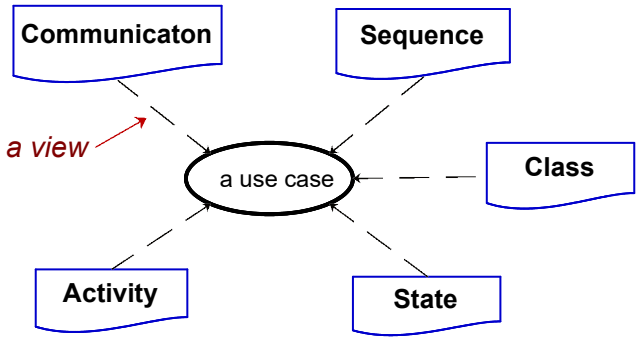


Validate use case: example 2



Use case analysis

- Use case analysis is a case-based way (a view) of describing the uses of the system with the goal of defining and documenting the system functional requirements.
- Each diagram is created for a view to the usecase



Use case scenario

USE CASE: Open Account

Actor: Bank Manager (BM)

Actor's goal: Create a new Customer's Account on the System

Basic flows:

1.BM: Request Open Account

2.SYSTEM: Ask Customer Data

3.BM: Give Customer Data

4.SYSTEM: Ask Account Type

5.BM: Give Account Type

6.SYSTEM: Ask Initial Balance

7.BM: Give Initial Balance

8.SYSTEM: Confirm to BM

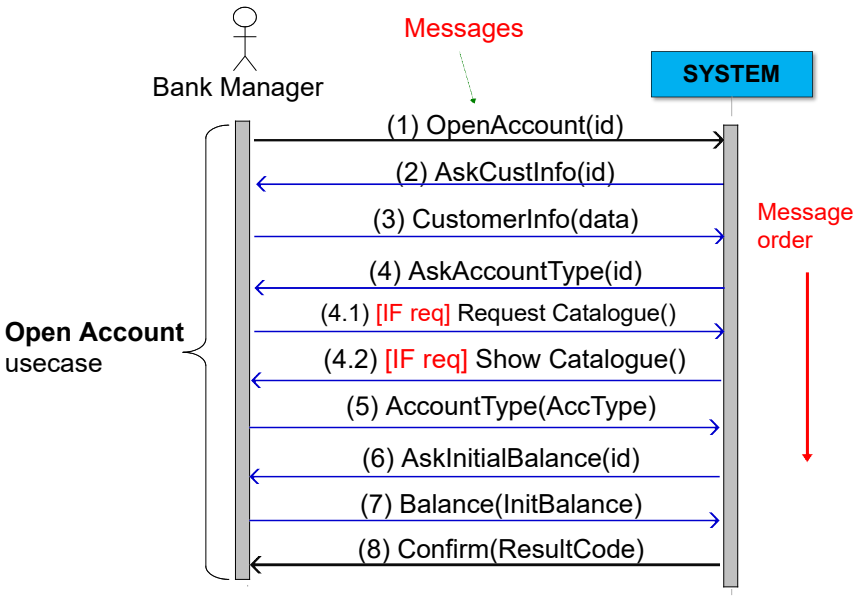
Alternative flows:

4.1.BM: Request Catalogue

(Continue step 5)

4.2.SYSTEM: Show Catalogue

UML: Sequence diagram



Message

- Message is a notification carrying a request or response sent from one object to another object (*going in one direction*).
- UML messages are expressed as:
  [condition] <message_entity_name> (parameter)
  - [condition]: for the message to appear.
  - <message_entity_name>: name of the message
  - (parameter): properties of the message group(system data)
- Example
  [acc=valid] Order (cust_id,prod_id, amount, date)

Message types

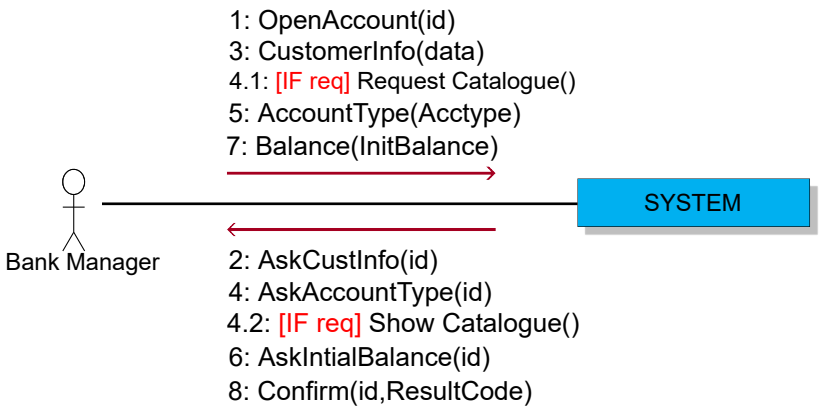
1. Messages for physical flows
   - chairs, books, food, ..
2. Messages for information
   - Orders, requests, invoices,...
3. Messages for currencies
   - Purchase money, service fee, tax amount,..
4. Messages for services
   - Laundry, car repair, warranty,...

- Messages must be appropriate to the sender/receiver

[Website] ---- book() -----> [customer] : wrong !
[Shipper] ---- book() ----> [customer]: OK

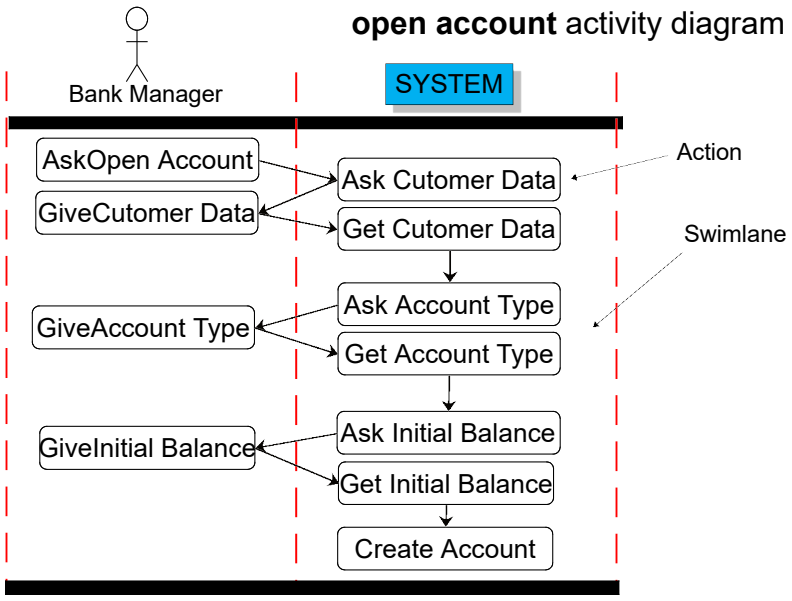
UML: Communication diagram

open account communication diagram



UML: Activity diagram

open account activity diagram

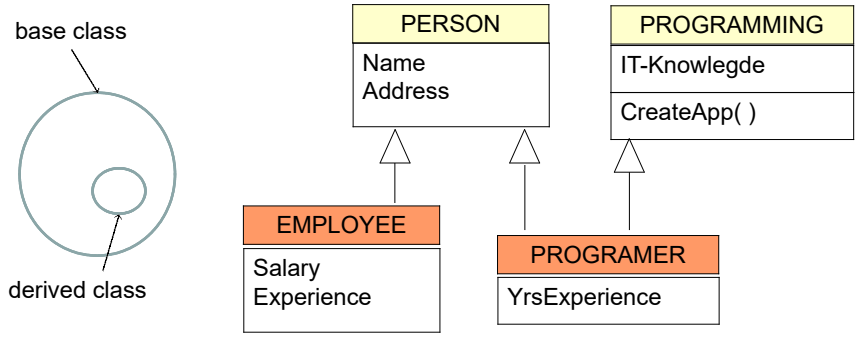


UML: Class diagram

- A class diagram is a diagram that links classes within a system, to describe this system static structure.
- A class diagram has three basic relationships:
  - Generalization
  - Aggregation/composition
  - Association
- Identifying relationships between object classes in a system is to find out the ability of objects in the system
  - Generalization: Know more details about the object
  - Association: Know requirements for the object in the association
  - Aggregation: Know the constraints on the components

Class: generalization

- Generalization : all child classes share the same attributes & behaviors of the parent class. The unique characteristics of the parent class are present in all child classes.
- Concretization : in addition to the unique attributes & behaviors of each child class, all child classes inherit all the attributes & behaviors of the parent class.



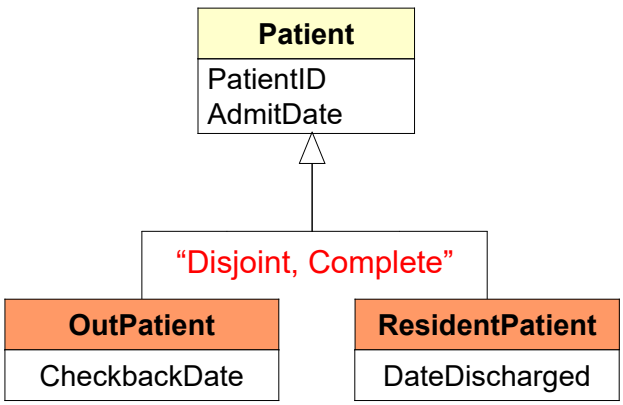
Inheritance

1. A child class inherits all the attributes, relationships, and services of the parent class.
2. A child class can inherit entire behavior of the parent class, and can also modify this behavior (polymorphism).

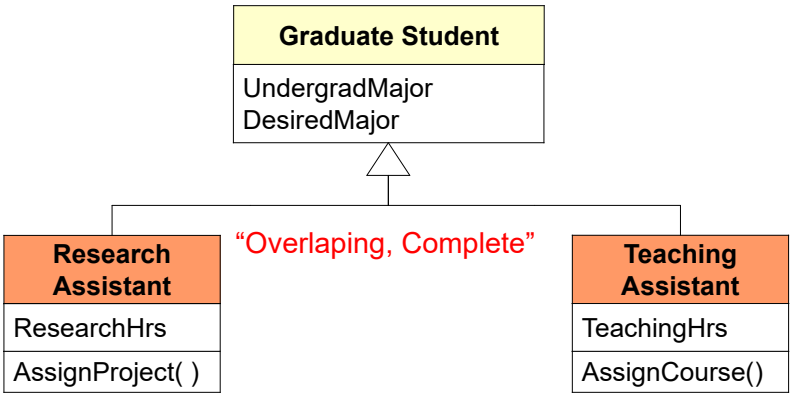
3. In UML diarams:

- Disjoint: No multiple inheritance, otherwise Overlapping.
- Complete: child classes are fully drawn, otherwise Incomplete (still some classes not drawn).

Generalization: Example



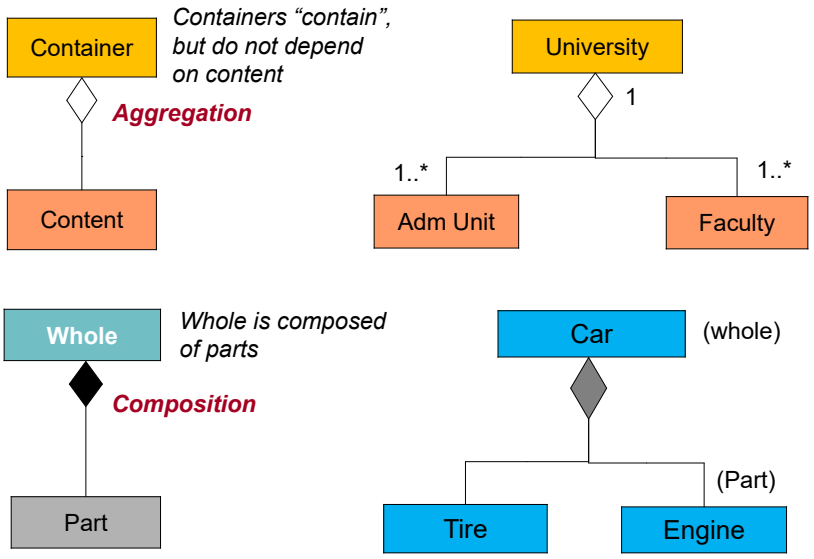
Generalization: Example



Class relationship: aggregation/composition

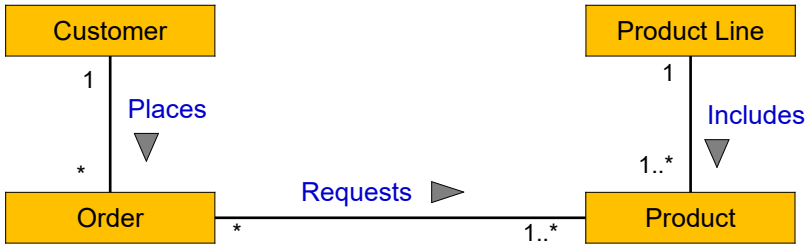
- Is a relationship that describes the connection of many objects with a common purpose (function) to "form" a single object, based on some properties:
  - a purchase order (must) consists of customer information (cust-info) and requested items (items)
  - Integration of components: wheels, engines, frames ... of a car.
  - Materials to make objects: iron, wood, plastic, ... to make chairs.
  - Container content: items, purchase date, place, ... in the order request.

Aggregation/composition: example



Class relationship: association

Is a "logical" relationship between object classes; the classes in the association exist independently of each other. Association name is required.

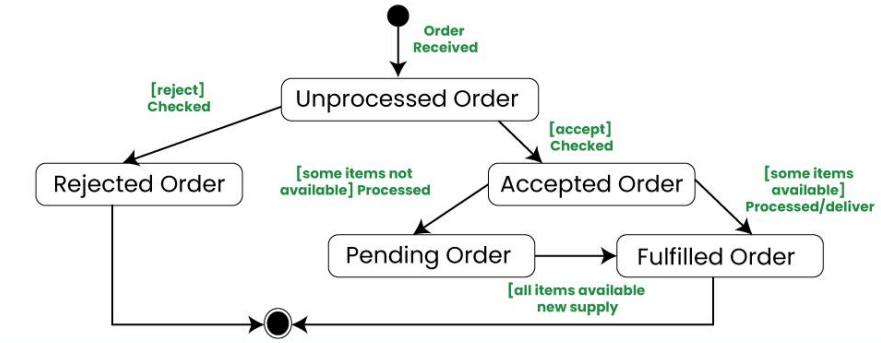


State model

- A state model describes objects's states and the relationship between them.
  - A state: is a temporary set of values of object's attributes at a certain moment (before or after the object responds to an event).
  - An event: a triggering event that changes the state.
  - An action: is an internal response activity of the object that causes a change in its state.
  - State transition: is a change from an old state to a new state due to the event.
- UML state model described as UML state diagram

UML State diagram example

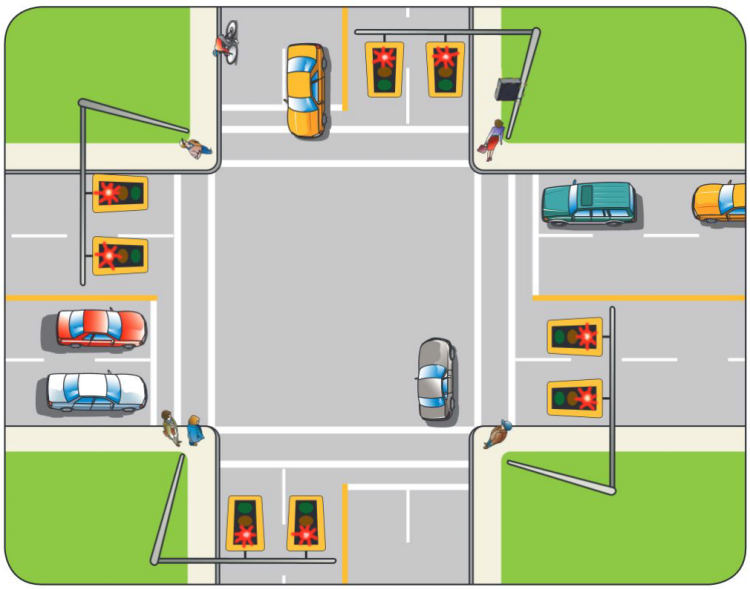
State machine diagram for an online order



State Machine Diagrams | Unified Modeling Language (UML)

State diagram has 3 components:
1. States 2.state transition (arrow) 3. Event on transition

State transition: Traffic light example

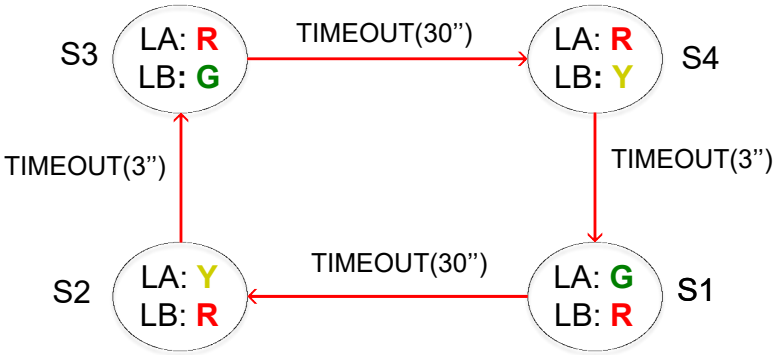


Traffic light- state

- Each light pole has 3 lights: Red (R), yellow (Y), Green (G) and only one is on, so its traffic light only has 3 states: R, Y and G.
- Each road has 2 opposite directions, each direction needs 1 traffic light, so a road needs a pair of lights.
- Traffic lights in a pair operate identically, so for simplification, the intersection has 2 lights: LA for road A, LB for road B.
- If one road is going (LA = G or Y) then the other must stop (LB = R), so the traffic light system at an intersection has 4 states:
  S1 = (LA=G, LB=R),
  S2 = (LA=Y, LB=R),
  S3 = (LA=R, LB=G),
  S4 = (LA=R, LB=Y)

Traffic light- state transition

- The duration(seconds) of each light is determined by a timer.
- Timeout (t) is state transition event, which changes the state of a traffic light after t seconds.
- Suppose the Green light will be on for 30'' & the Yellow light will be on for 3", the state transition diagram of the traffic light system at intersection A & B is



3. System analysis - Outside view

The major goal is to address this basic question:
what should the system do in its working environment ?

1. Identify the system's responsibilities
2. Survey and document the interactions between the system and external objects
   - Draw collaboration diagram in which the system is a tool
3. Identify the situations in which the system participates
   - Draw usecase diagram for the system
4. In each use case, identify the interactions between the system and its actors (usecase scenario)
   - Draw sequence diagram

Example 1: Coffee cup

What do people need from a cup of coffee? (a cup of coffee is a system, people are actors)

Coffee cup in its environment

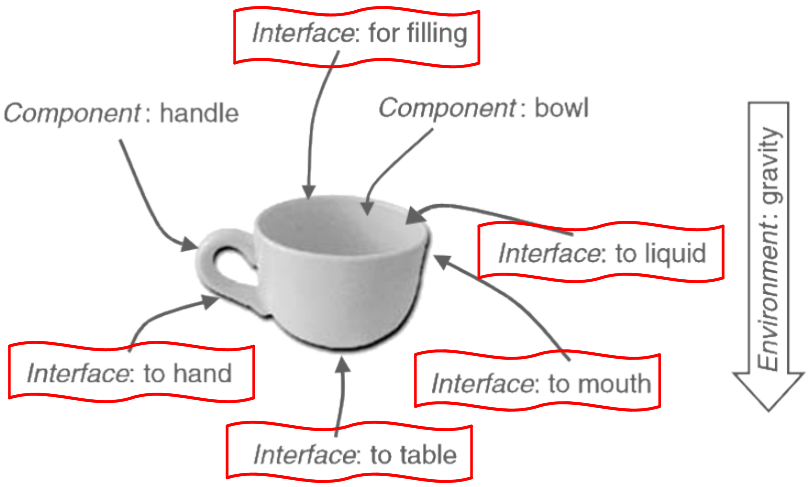


Figure 1.1 A cup as a very simple system.

Coffee cup use cases

(actor)

Filling

Holding

(actor)

Pouring

Holding

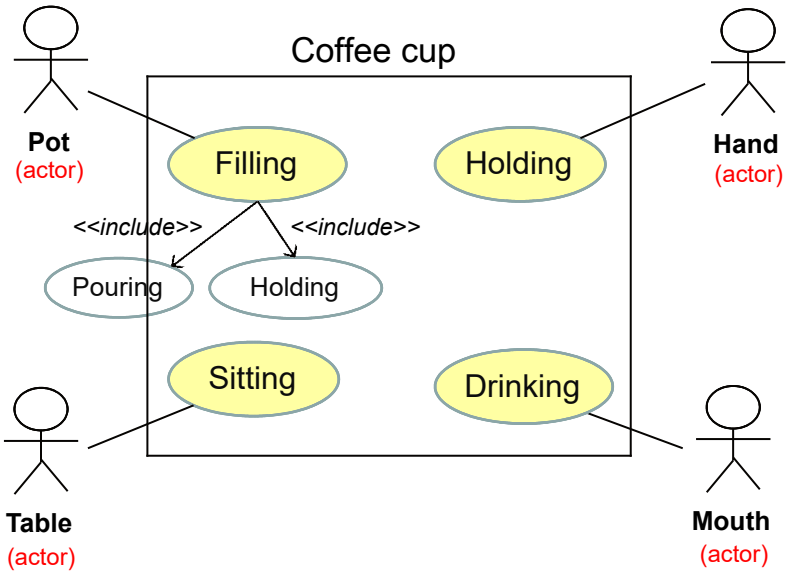
Sitting

Drinking

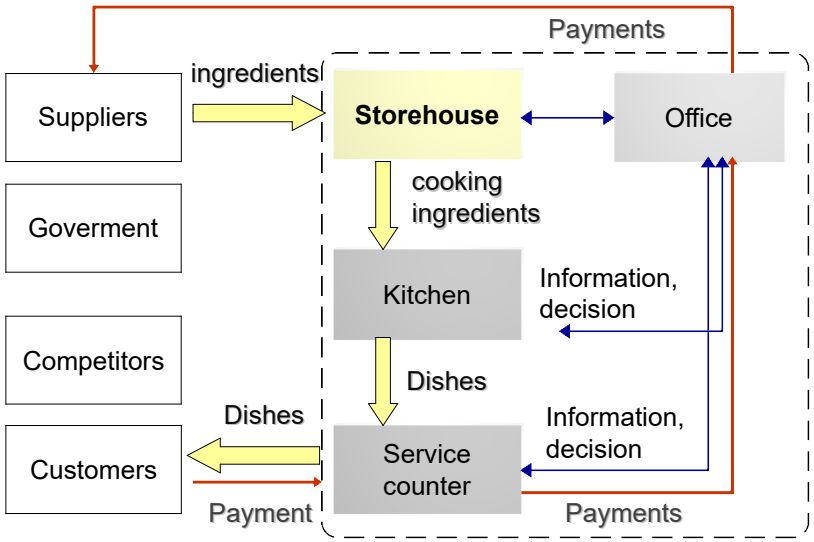
(actor)

(actor)

Coffee cup UML usecase

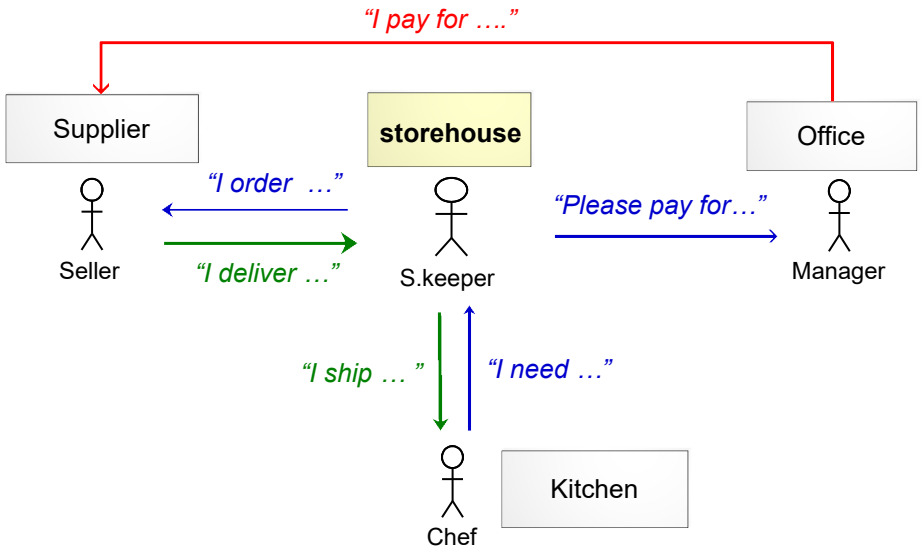


Example 2: Storehouse management

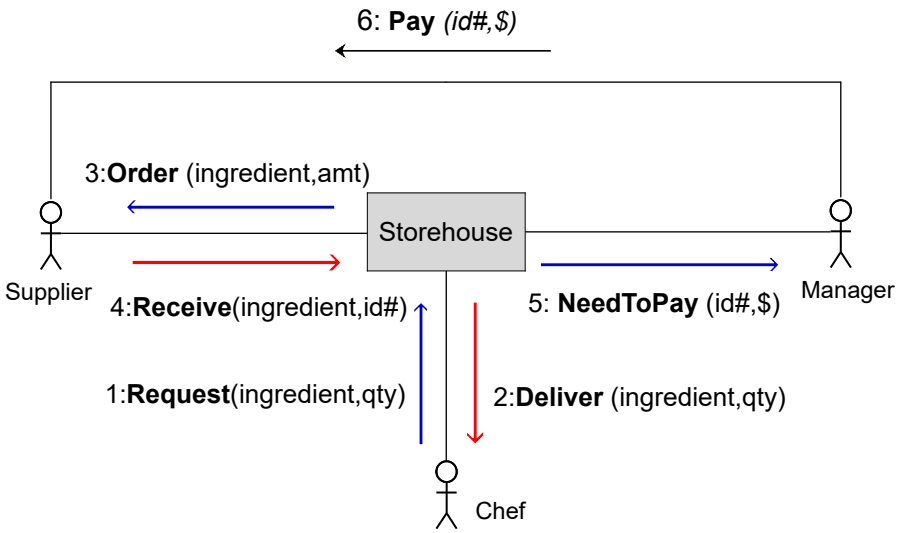


Storehouse interactions with the outside

Storehouse interactions are storekeeper interactions

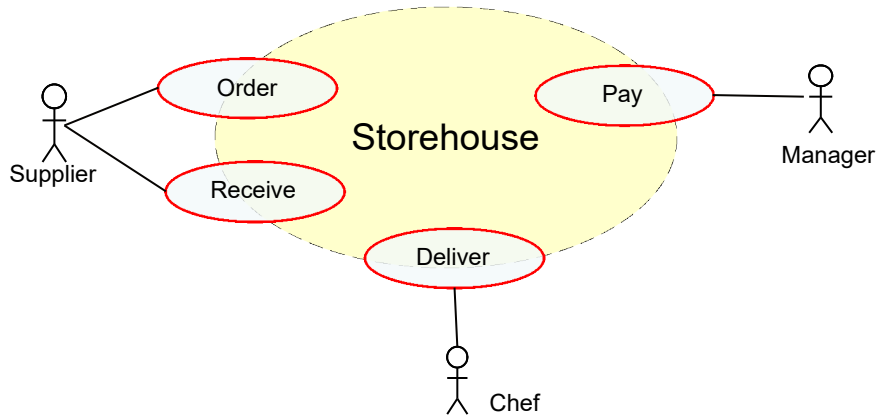


Storehouse communication diagram



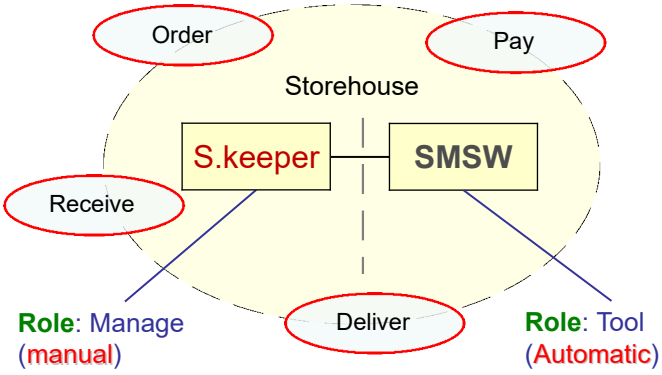
Storehouse use case diagram

The interaction messages in the collaboration diagram above introduce the situations that the storehouse handles: order ingredient, receive ingredient, deliver ingredient, and pay for supplier's bill, which are use-cases drawn in the diagram:



Storehouse management software

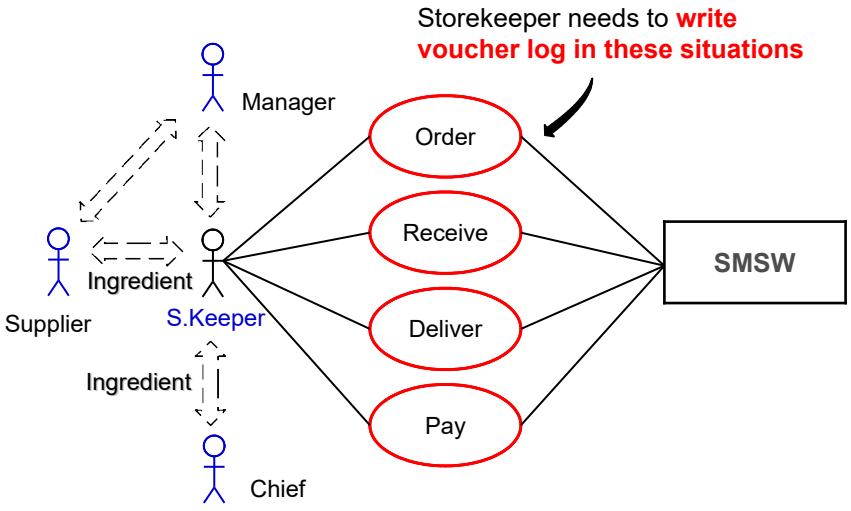
Storehouse is a system: including storekeeper (to handle material flow, money and decision) and storehouse management software (SMSW)



how do the storekeeper and the SW collaborate with each other ?

SMSW: use-cases (a)

SMSW (a): storekeeper is an only one actor



SMSW: use-cases (b)

SMSW (b): actors are storekeeper, supplier, manager, chief