Capturing Requirements – Use Case Model

1

Objectives

- Define use cases.
- How to identify a usecase.
- Learn how to capture functional requirements with use cases.
- Learn how to develop a **use case model.**
- Learn how to write a **use case specification.**

2

What is a use case?

"**A specification of sequences of actions**, including variant sequence and error sequences, that a system, subsystem or class can perform by interacting with outside actors." - UML Reference Manual

3

What is a use case?

- A use case is something an actor needs the system to do.
- It is a "case of use" of the system by a specific actor
- Use cases are *always* started by an actor
  - The ***primary actor*** triggers the use case
  - Zero or more ***secondary actors*** interact with the use case in some way
- Use cases are *always* written from the point of view of the actors
- UML notation for use case:

4

Identifying Use Cases

- Start with the list of **actors** that interact with the system.
- When identifying use cases ask:
  - What functions will a specific actor want from the system?
  - Does the system store and retrieve information? If so, which actors trigger this behaviour?
  - What happens when the system changes state (e.g. system start and stop)? Are any actors notified?
  - Are there any external events that affect the system? What notifies the system about those events?
  - Does the system interact with any external system?

5

Use Case Modelling

- Use case modelling is a form of requirements engineering.
- Use case modelling proceeds as follows:
  - Find the system boundary
  - Find actors
  - Find use cases
    - Use case specification
    - Scenarios
  - Show relationships between:
    - Actors and use cases
    - Use cases

6

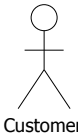
The Subject (system boundary)

- Before we can build anything, we need to know:
  - Where the boundary of the system lies
  - Who or what uses the system
  - What functions the system should offer to its users
- We create a Use Case model containing:
  - Subject: the edge of the system
    - also known as the system boundary
  - Actors: who or what uses the system
  - Use Cases: things actors do with the system
  - Relationships: between actors and use cases

7

What are actors?

- An actor is anything that interacts ***directly*** with the system.
  - Actors identify who or what uses the system.
- Actors are *external* to the system.
- An Actor specifies a *role* that some external entity adopts when interacting with the system
- UML notation for Actor

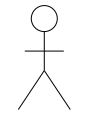


«actor»
Customer

8

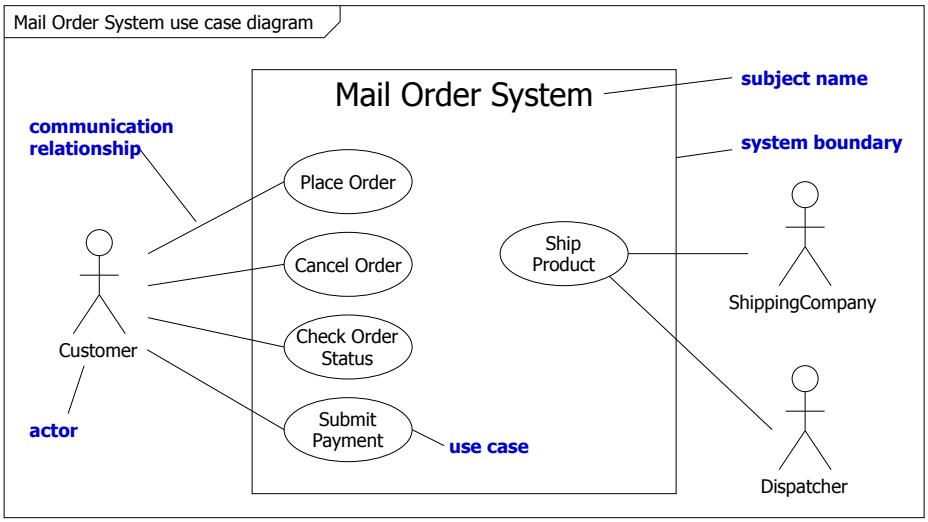
Identifying Actors

- When identifying actors ask:
  - Who or what uses the system?
  - What roles do they play in the interaction?
  - Who installs the system?
  - Who starts and shuts down the system?
  - Who maintains the system?
  - What other systems use this system?
  - Who gets and provides information to the system?
  - Does anything happen at a fixed time?



9

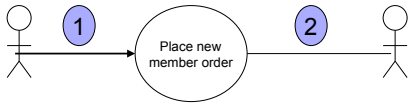
Partial Use Case Diagram



10

Use Case Relationships

Associations: describes an interaction between an actor and a user case.



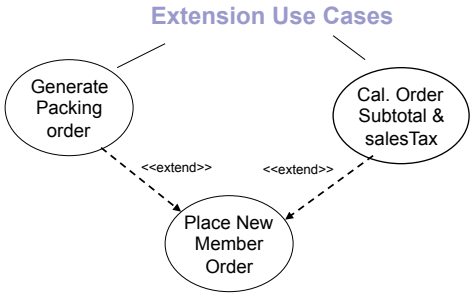
1 Indicates the use case was initiated by the primary actor.

2 Indicates an interaction between the use case and a secondary actor.

11

Extension Use Case

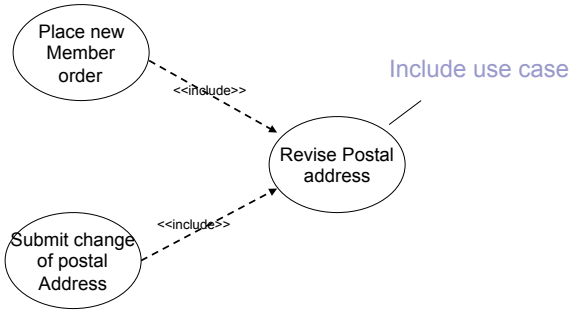
- A use case may contain **exceptions and complex functionality** consisting of several steps.
- We can extract more complex steps into their own use case – called extension UC.



12

Include Use Cases

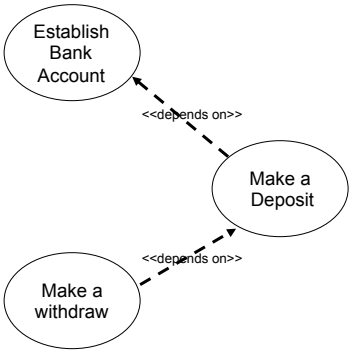
- Two or more use cases may perform steps of identical functionality.
- Best to **extract** these **common steps** into their own separate use cases.
- This helps minimize redundancy.



13

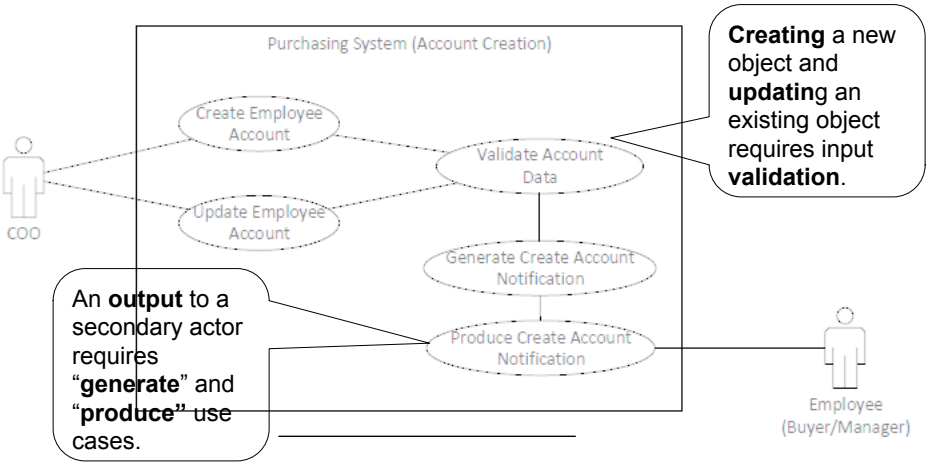
"Depends on" Relationship

- A relationship between use cases indicating that one use case cannot be performed until another use case has been performed.
- Can be used for planning and scheduling purposes.



14

Working with input and output



15

Developing a use case model

- Based on prior requirements analysis
  - Identify primary actors.
  - Identify use cases.
  - Identify secondary actors.
- If creating a new object, must include the "Validate *named-object* Data" and "Update *named-object*" use cases.

16

Developing a use case model

- If notifying secondary actors, include "Generate named-notification" and "Produced named-notification" use cases.
- Place primary actors on the LEFT-side of the system boundary.
- Place secondary actors on the RIGHT-side of the system boundary.

17

Developing a use case model

Keep in mind…

- An actor may be both a primary and a secondary actor.
- Some functional requirement may or may not have a secondary actor.
- A functional requirement must have a specific primary actor.

18

Developing use case model

- Use standard association relationship (solid line from primary actor(s) to use-case, from use-case to use-case, from use-case to secondary actor(s).
- May organize and draw each page based on each primary actor. This will produce multiple pages.
- May organize and draw the entire model in one page if possible.

19

Develop a Use Case Model (in class)

The Purchasing System must allow an authorized procurement **director** to **create Approver Accounts** for authorized buyers and managers and set their purchasing limit for purchase order approval. After that, the system must **notify the buyers and managers** so that they can use their account to log into the purchasing system for to approve purchase orders.

20

Develop a Use Case Model (in-class)

The Purchasing System must allow an authorized **buyer to submit one or more "approved" purchase orders to associated vendors.** After the submission, the system must **notify the buyers** that the orders have been submitted. In addition, the **AP and IM systems must be notified of the submitted purchase orders**.

21