IF2261 Software Engineering

OOSE – Analysis
Requirement Model

Program Studi Teknik Informatika
STEI ITB

Analysis

- The aim is to analyze, specify, and define the system to be built
- The models developed
  - will describe what the system is to do
  - will make it easier for us to understand the system
  - are fully application-oriented (no consideration is paid to the real implementation environment)
  - can be used without changing even if the implementation environment changed
- Two models are yielded:
  - the requirements model
  - the analysis model.

Requirement Model

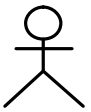
- Requirement model:
  - describes all the functional requirements from user perspective
  - describes the way the system is to be used by end users
  - structured from a logical perspective into a robust and maintainable system.
  - based on the requirement specification and discussions with the prospective users

Requirement Model (2)

- Consists of:
  - Use case model
  - Interface description
  - Problem domain object model

Use Case Model

- Uses actors and use cases
- Actor
  - are not part of the system, they represent roles a user of the system can play.
  - may actively interchange information with the system.
  - may be a passive recipient of information.
  - can represent a human, a machine or another system.
- Use case
  - models a dialogue between actors and the system.
  - is initiated by an actor to invoke a certain functionality in the system.
  - models a dialogue between one or more actors and the system that returns a result of measurable value to at least one actor.
  - is a complete and meaningful flow of events.
  - represents a major system usage goal for one or more of the actors that interact with the use case.



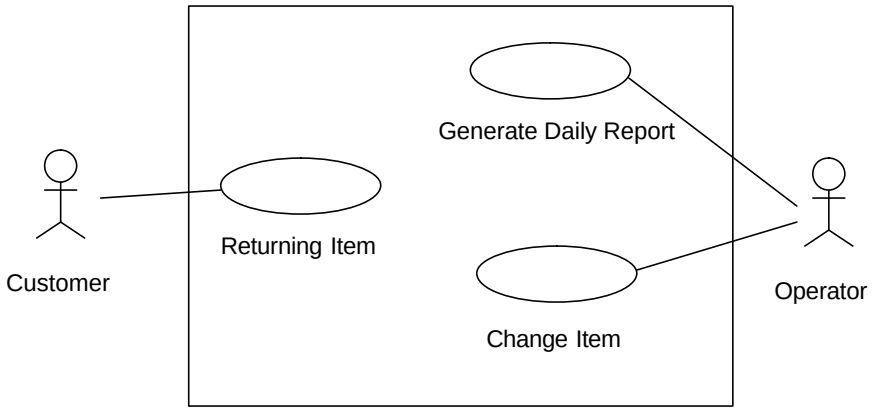
Actor

Use-Case

Use Case Model (2)

- Activity:
  - Identify actors (by identifying the users of the system)
  - Identify use cases through actors by asking a number of questions:
    - What are the main tasks of each actor?
    - Will the actor hate to read/write/change any of the system information?
    - Will the actor have to inform the system about outside changes?
    - Does the actor wish to be informed about unexpected changes?
  - Draw the use case diagram to show relationships between actors and use cases
    - Use extension association between use case to specify how one use case description may be inserted into another use case description
  - Describe the use cases
    - Describe the basic flow
    - Describe alternative flows

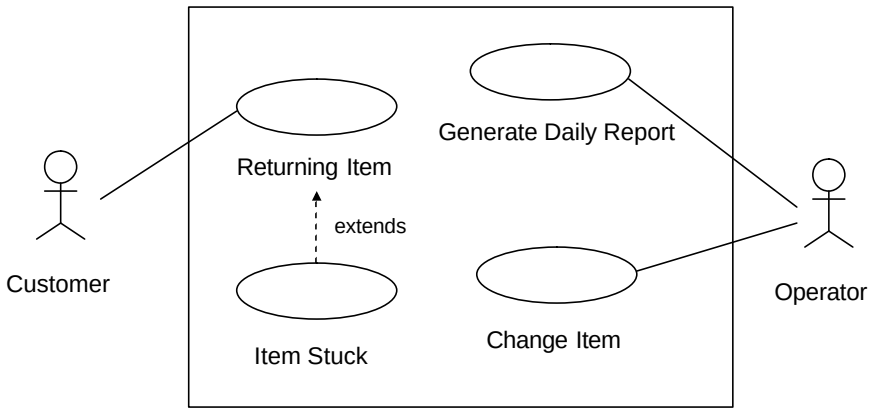
Use Case Diagram – Example
Recycling Machine System



Use Case Diagram - Extension

- *Extends*; defines alternative course of events:
  - optional parts of use cases
  - complex and alternative courses which seldom occur
  - separate sub-courses which are executed only in certain cases
  - situation where several different use case can be inserted into a special use case

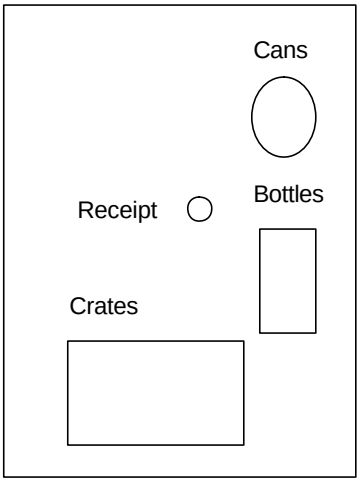
Use Case Diagram – Example - Extends



Interface Description

- The interface has to **capture the user's logical view of the system**
  - the main interest is the consistency of this logical view and the actual behavior of the system.
- Techniques:
  - Use sketches of what the user will see on the screen when performing the use case
  - Simulations using a User Interface Management System (UIMS)
- Such a technique will eliminate several possibilities for misunderstandings
- It is important that users are involved in making detailed interface descriptions.

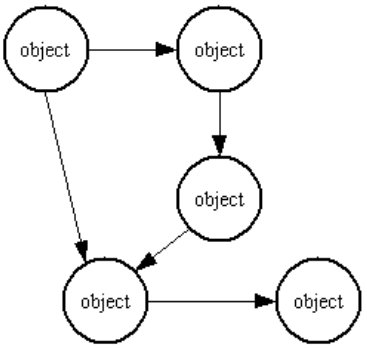
Example - Recycling Machine System



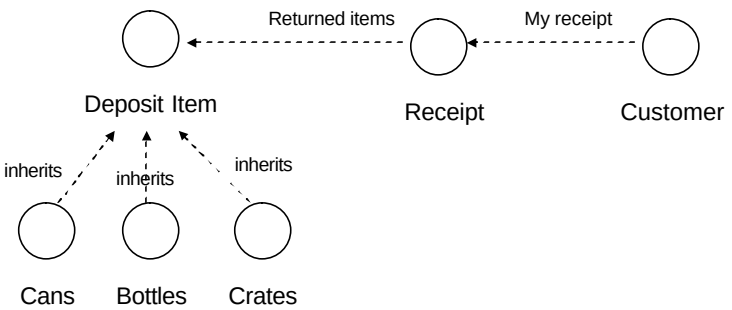
Problem Domain Object Model

- The model is the **identification of the objects** that have a direct counterpart in the application environment and that the system must know about.
- The refinement of the problem domain objects can be obtained in different possible degrees:
  - *object name*
  - *logical attributes*
  - *static instances associations*
  - *inheritance*
  - *dynamic instance associations*
  - *operations*

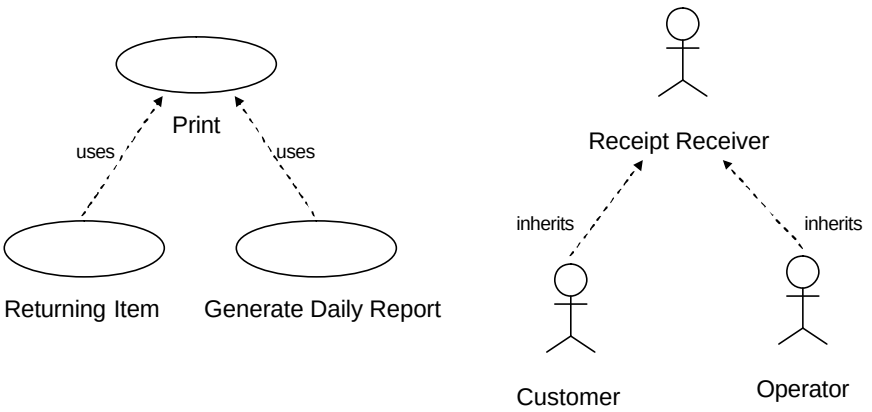
Problem Domain Object Model (2)



Example - Recycling Machine System



Refinement of the Requirement Model



Discussions

- How complete the use cases should be?
- Generally, it is better to have longer and more extensive use cases than several smaller ones.
- The arguments in favor of having the sequence as one complete use case are:
  - When specifying the use case, we may follow a complete flow through the entire system
  - From the orderer's point of view, it is a logical cohesive flow of events in the system
  - It may be more effective when testing the use case since it covers more logical cohesive events in the system
  - It is easier to synchronize the use case since it is one sequence that starts diffeent events in chronological order
- The arguments in favor of separating the use case into several different use cases are:
  - It may be troublesome to find the right instance of a use case
  - From a potential actor's view, it is more logical to have use cases that the actor starts
  - It may be easier to test the use case since every use case starts from external events and not from internal system events