IF2261 Software Engineering

OOSE – Analysis
Analysis Model

Program Studi Teknik Informatika
STEI ITB

Analysis Model

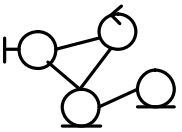
- When the requirement model has been developed, and often also signed up by the orderers, we can focus on the structuring of the system by developing the analysis model
- The aims is to create a good platform for the system and will also form the basis of the design
- Developing the analysis model really entails distributing the behavior specified in the use case description among the objects in the analysis model
  - It should be stated explicitly which object is responsible for which behavior in the use case
- Describe the system using three different types of object:
  - Entity object
  - Control object
  - Interface object

OOSE Analysis Model

Interface object
Interface objects interact directly with the environment.

Entity object
Information about an entity object is stored even after a use case is completed.

Control object
A control object illustrates functionality that is not contained in any other object in the system.



Analysis Model

Interface Object

- All functionality specified in the use case description that is directly dependent on the system environment is placed in interface objects
- Actors communicate with the system through these objects
- The task is to translate the actor's input to the system into events in the system, and to translate these events into something which is presented to the actor
- Having identified interface objects, it will be easy to modify an interface in the system
  - this is a common situation that must be manageable

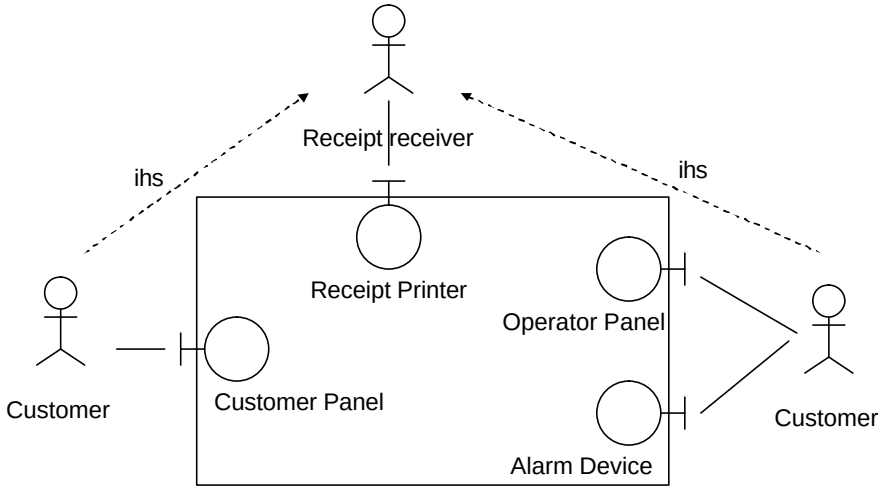
Strategies in finding interface objects

- Identified from the system interface description accompanying the requirement models
- Start from the actors
- Read the use case description and extract the functionality that is interface-specific

Example – Use Case Decription Use Case Returning Item

- When the customer returns a deposit item, it is measured by the system. The measurements are used to determine what kind of can, bottle or crate has been deposited. If accepted, the customer total is incremented, as is the daily total for that specific item type. If the item is not accepted, the lights for “NOT VALID” is highlighted on the panel.
- When the customer pressed the receipt button, the printer prints the date. The customer total is calculated and the following information printed on the receipt for each item type:
  - Name
  - Number returned
  - Deposit value
  - Total for this type
- Finally the sum that the customer should receive is printed on the receipt.

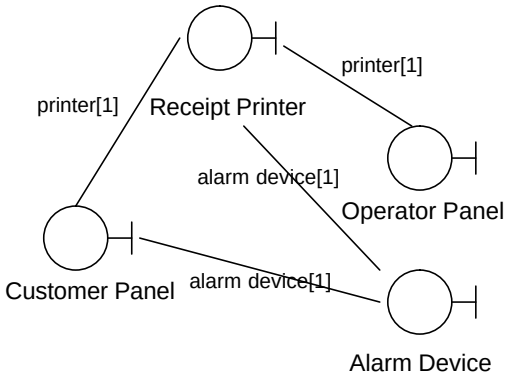
Recycling Machine System
Interface Object – Example



Association between interface objects

- Interface objects must know of each other to be able to solve certain task
- Acquaintance association
  - Solid directed line (uni directional)
  - Having name and cardinality
- Cardinality:
  - Says how many instances can be associated

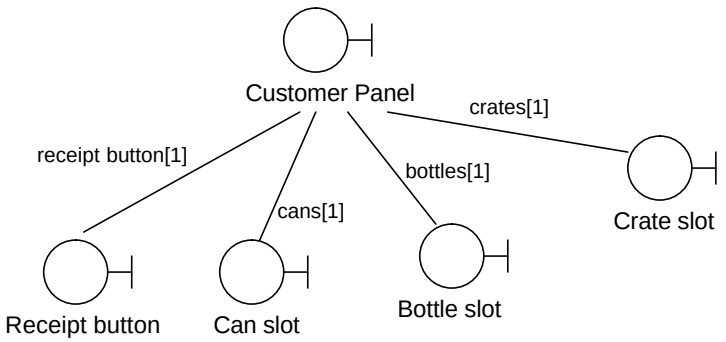
Interface Object – Example Acquaintance Association



Association between interface objects (2)

- Consist of association
  - Aggregate
  - Containment hirarchy
- Central interface

Interface Object – Example
Central Interface



Interface object type

- Interface to other system
  - Translates to a standardized protocol
  - Just sends out stimuli without any complex conversion

  Advantages:
  - If the protocol changes, these changes will be local to this object
- Interface to a human user
  - Often request GUI
  - In interface-intensive application, interfaces object could be the major part (up to 80%) of entire application
    - Use supporting tools

Strategies for allocating functionality (Hartson and Hix 1989)

- Computational dominant control or embedded control
  - Place the controlling functionality inside the system (in the control objects or entity objects
  - Interface objects do not have very much functionality
- Dialogue dominant control
  - Place the controlling functionality in the interface objects
  - Do not have many control objects
- Mixed control
  - Place the controlling functionality on both side, which offers more flexibility
- Balanced control
  - Separate control form both the dialogue and computational
  - The global control component governs sequencing among invocations of dialogue and computational functions

Which type to choose must be decided on from application to application

Entity Object

- Used to model the information that the system will handle over a longer period of time
  - The information should be kept even if the use case has been complete
- Also used to allocate the behavior that naturally belongs to this information
- Entity objects use attributes to store information

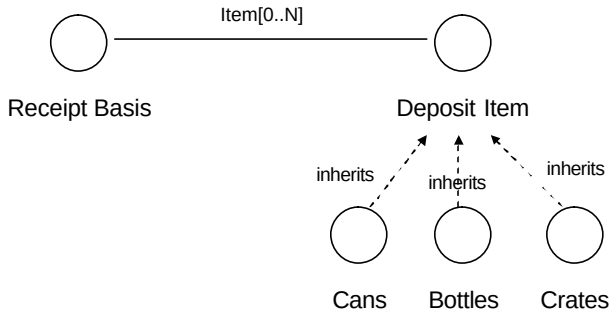
Strategies in finding entity objects

- Identified from the use case description
  - Most are found early and obvious
- It's easy to find entity object, but more difficult to identify its operations and attributes
  - In nomal case, operations are not identified in the analysis model (but later in the design model)
  - Typical operations that must be offered by an entity object:
    - creating and deleting the entity object
    - storing and fetching information
    - behavior that must be changed if the entity object is changed

Example – Use Case Decription Use Case Returning Item

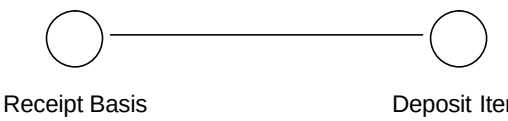
- When the customer returns a deposit item, it is measured by the system. The measurements are used to determine what kind of can, bottle or crate has been deposited. If accepted, the customer total is incremented, as is the daily total for that specific item type. If the item is not accepted, the lights for "NOT VALID" is highlighted on the panel.
- When the customer pressed the receipt button, the printer prints the date. The customer total is calculated and the following information printed on the receipt for each item type:
  - Name
  - Number returned
  - Deposit value
  - Total for this type
- Finally the sum that the customer should receive is printed on the receipt.

Entity Object – Example
Recycling Machine System



Association between entity object

- Communication association
  - The entity object contact another entity object and asks for information about something



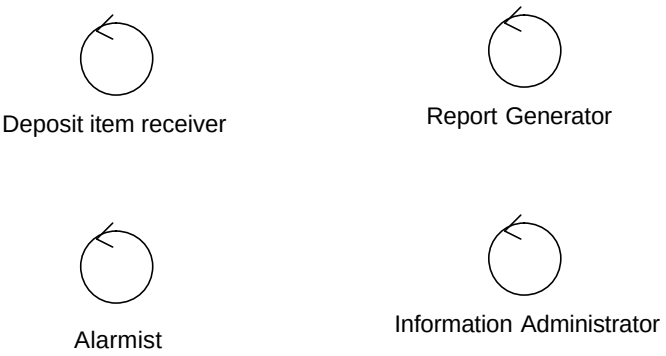
Control Object

- Used to model behavior that is not naturally placed in either of another two objects (interface and entity)
  - Usually needed in more complex use cases
- Typically acts as glue which unites the other objects
- Normally found directly from the use cases
  - In the preliminary draft, assign one control object for each use case
- Deviation from the initial approach can be made for several reason:
  - In the case where there is no behavior in the use case left to model, then the control object is not needed.
  - In the case of a very complicated behavior, the functionality may be divided into several control objects
- Typical type of functionality placed in the control objects are
  - transaction-related behavior, or
  - control sequences specific to one or a few use cases, or
  - functionality thas separates the entity objects from the interface objects.

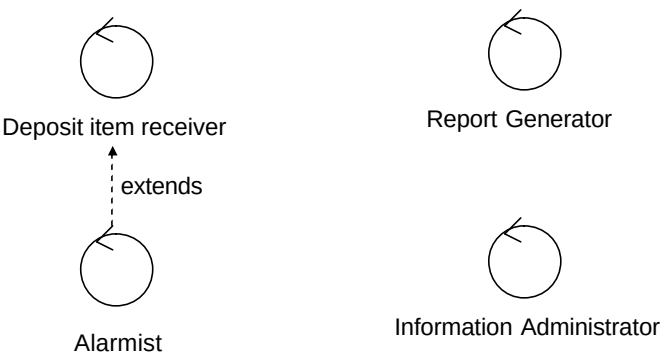
Example – Use Case Decription Use Case Returning Item

- When the customer returns a deposit item, it is measured by the system. The measurements are used to determine what kind of can, bottle or crate has been deposited. If accepted, the customer total is incremented, as is the daily total for that specific item type. If the item is not accepted, the lights for “NOT VALID” is highlighted on the panel.
- When the customer pressed the receipt button, the printer prints the date. The customer total is calculated and the following information printed on the receipt for each item type:
  - Name
  - Number returned
  - Deposit value
  - Total for this type
- Finally the sum that the customer should receive is printed on the receipt.

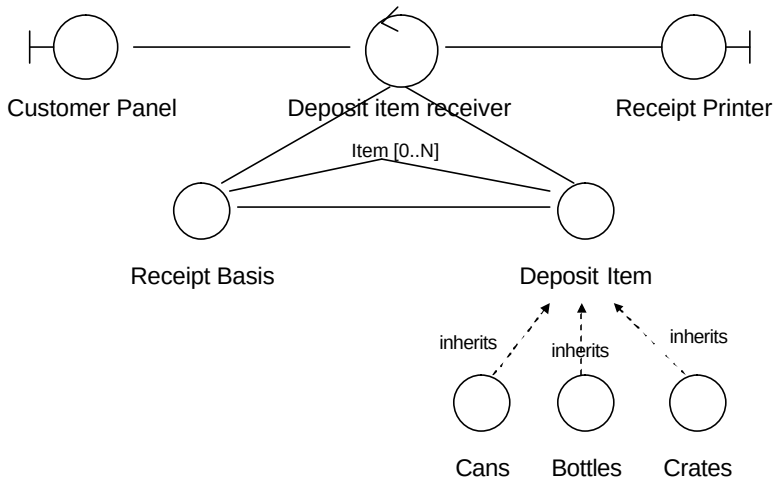
Control Object – Example
Recycling Machine System



Working with the analysis objects



Analysis Model – Example
Recycling Machine System – Use Case Returning Item



Subsystem

- To get a clear overview of the system, the objects identified need to be placed in groups called subsystem
- The taks is to package the objects so that complexity is reduced
- The lowest level of subsystem called service packages

Subsystem – Example - Recycling Machine System