IF2261 Software Engineering

OOSE - Construction

Program Studi Teknik Informatika

STEI ITB

Why do we have a construction process ?

- The analysis model is not sufficiently formal
- The actual system must be adapted to the implementation environment
- We want to validate the analysis result

What is done in the construction phase ?

- Identify the implementation environment
  - Identifying and investigating the consequences that the implementation environment will have on the design
- Incorporate these conclusions and develop a first approach to a design model
- Describes how the objects interact in each specific use case

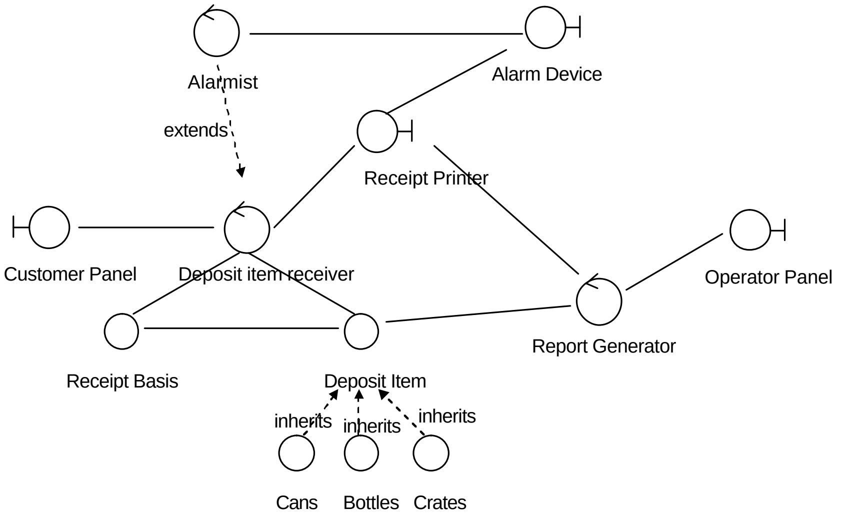
The Design Model

- Refine the analysis model in the light of the actual implementation environment
  - Define interface of the objects
  - Define semantics of the operation
- Decide how different issues such as DBMS, programming language features, and distribution will be handled
- Compose of blocks which are the design objects
  - The block will abstract the actual implementation
  - The implementation of the blocks may be:
    - one specific class in the source code
    - several different classes

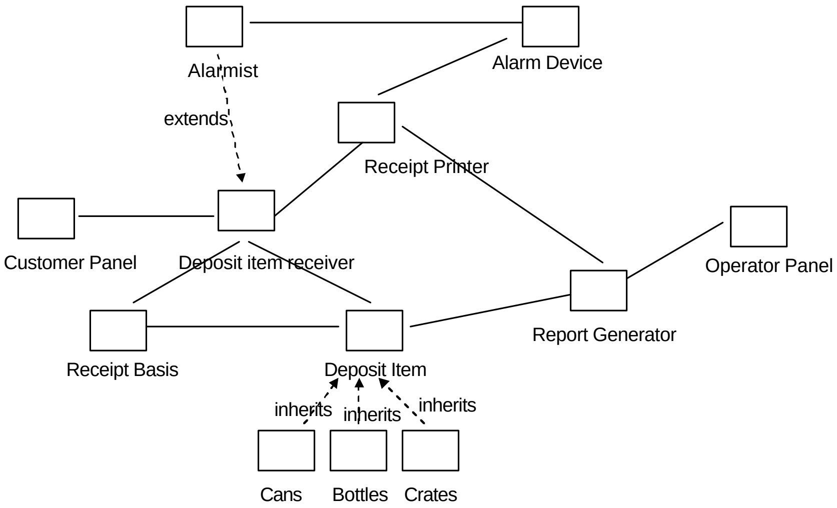
Building a design model

- First attempt can be made mechanically
  - based on the analysis model
  - obtain a clear traceability to the analysis model
- The Analysis Model vs The Design Model
  - The analysis model: developed in logical terms and is only a conceptual picture of the system to be built
  - The design model: an abstraction of how the actual system really is built
- The final structure should reflect how the implementation environment has affected construction
  - For example, if the programming language does not support inheritance, the model must reflect how the inheritance is really implemented

The Analysis Model – Example
Recycling Machine System



First attempt of the Design Model – Example Recycling Machine System



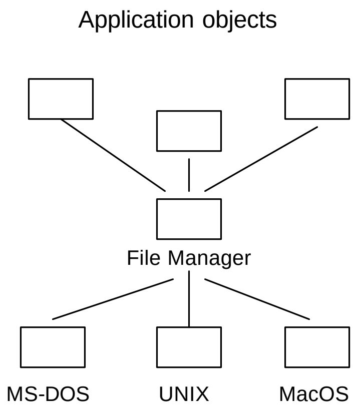
The Design Model - Implementation Environment

- Identify the actual technical constraints under which the system should be built
- Including:
  - The target environment
  - Programming language
  - Existing products that should be used (DBMSs, etc)
- Strategies:
  - As few objects as possible should be aware of the constraints of the actual implementation environment

The Design Model - Implementation Environment

- Target environment
  - Create a new blocks that represent occurrences of the changed parts in the target environment
    - Strategies:
      - Specified an abstract class
        - polymorphism
      - The object can check the platform at run-time
        - CASE statement in the source code
      - Decide this when the system us delivered
        - Provide several different modul which will be choosed later
  - Investigate whether the target environment will execute in a distributed way
    - on a different processors or different processes

Example – Adapting the target environment



The Design Model - Implementation Environment

- Programming language
  - Affect the design in translating the concepts used
  - The basic properties of the language and its environment are fundamental for the design
    - Inheritance and Multiple inheritance
    - Typing
    - Standard
    - Portability
    - Strategies for handling errors during run-time
      - Exception (Ada)
      - Assertions (Eiffel)
      - None (C++ ver 2)
    - Memory management
      - Automatic garbage collection
  - The use of component
    - Component library, such as interface objects

The Design Model - Implementation Environment

- Using existing products
  - DBMS
  - UIMS (User Interface Management System)
  - Network facilities
  - Internally or externally developed applications that should be incorporated
  - Products used during development
    - Compilers
    - Debuggers
    - Preprocessor
- Other considerations
  - Requirement for performance
  - Limitations of memory

The Design Model - Implementation Environment

- Other considerations
  - Strategies:
    - To postpone optimizations until they are needed or you are absolutly sure that they will be needed
      - the real bottlenecks are often missed and then new optimizations are necessary
    - Use simulation or prototyping to investigate potential optimization problem early
      - Extensive experiences may help to jugde at an early stage
    - If you’re not sure of the correctnessof a performance optimizations, you should not make it untill you’re sure of how it should be done

The Design Model - Implementation Environment

- The people and organization involved in the development could also afect the design
  - The principal strategy:
    - such factors should not affect the system structure.
    - The reason: the circumtances (organizations, staffing, competence areas) that are in effect today will probably change during the system’s life cycle

Working with the design model

- Changes can and should occur, but all changes should be justified and documented (for robustness reason)

- We may have to change the design model in various way:
  - To introduce new blocks which don't have any representation in the analysis model
  - To delete blocks from the design model
  - To change blocks in the design model (splitting and joining existing blocks)
  - To change the associations between the blocks in the design model

- Use the component

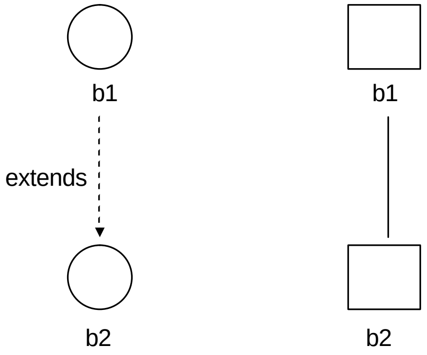
Working with the design model (2)

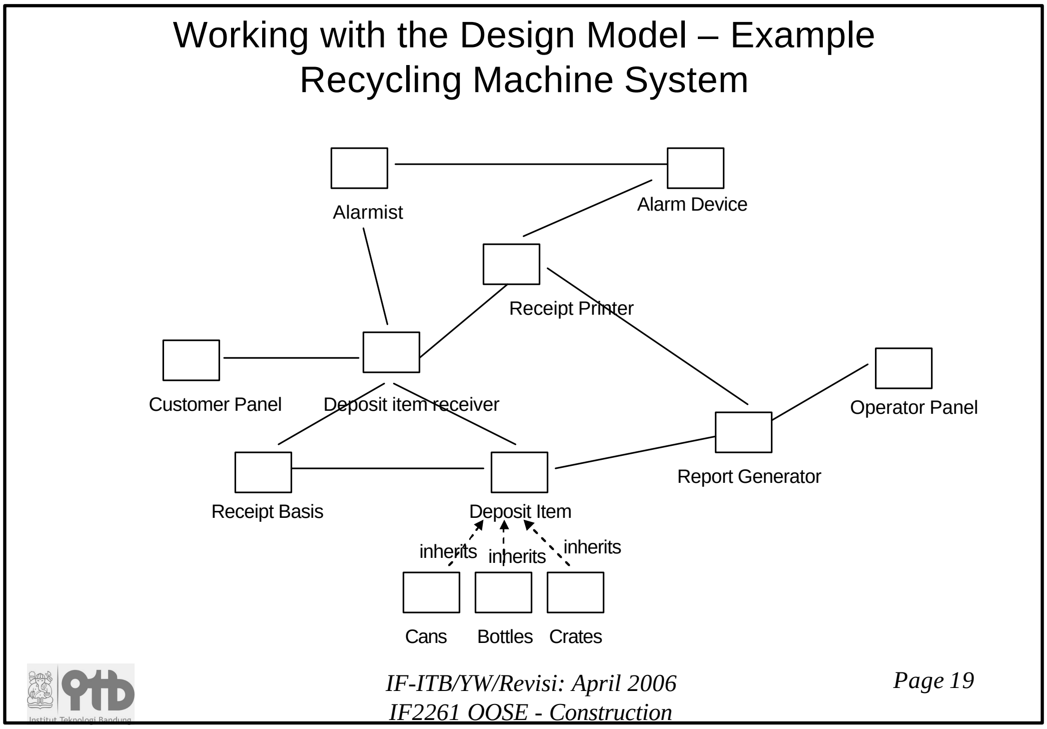
- Adding blocks
  - Adding blocks to handle the environment is a good change
  - Adding blocks for other functionality should not normally be done, since they should introduced through the analysis model
- Deleting blocks
  - More suspicious
  - You have to have good reasons for it
    (often implementation reasons)
  - Changing the logical structure of the system should be made in the analysis model first
- Splitting and joining blocks
  - Also suspicious changes
  - Will often decrease the robustness of the system
  - Should be done with a great care

Working with the design model (3)

- Changes associations
  - The most common change in the design model
  - Often come from the implementation environment
    - Synchronization and communication between processes
    - Actual implementation of association
      - Extension association has no direct implementation technique
      - Inheritance associations

Example – Change Association extends assc. → communication assc.



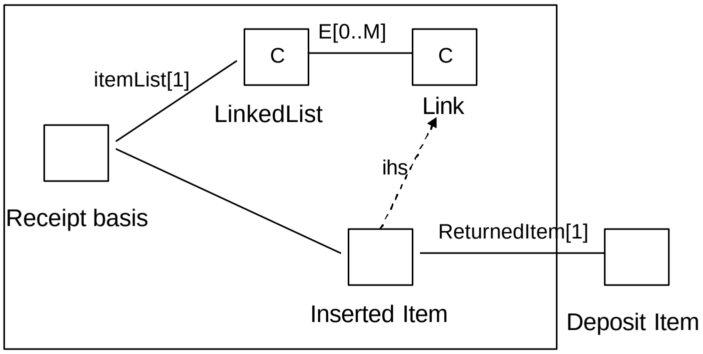


Working with the Design Model – Example Recycling Machine System

Use the component

- At an early stage, it is essential to decide upon which component libraries
  - When a component library exist, investigations should begin to find out what functionality it will offer
  - It is essential to be familiar with the library
- Rule of thumb for finding places where components could be used
  - Look for acquaintance association with cardinalty [0..N]
    - This will typically yield a list or array to hold several references

Example – the use of component



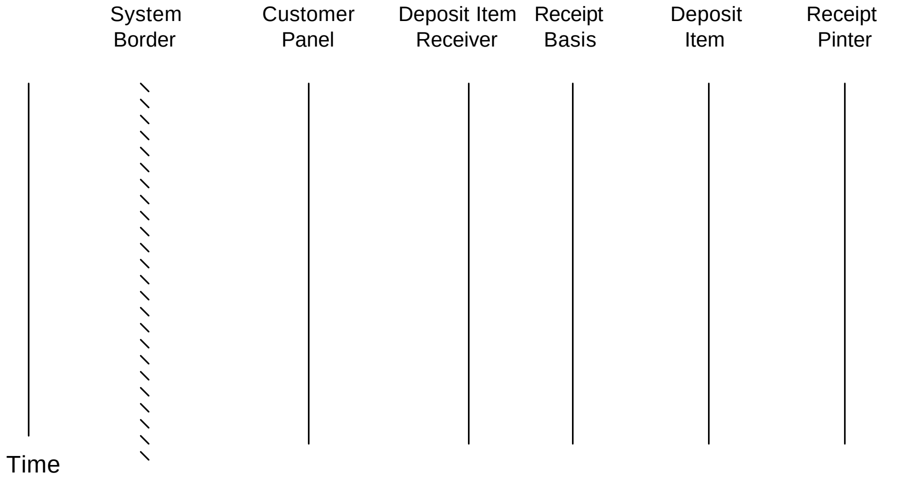
Interaction Diagram

- Describe how the blocks are to communicate by designing the use case
- The main purpose of the use case design is to define the protocols of the blocks
- The interaction diagram describes how each use case is offered by communicating objects
  - The diagram shows how the participating objects realize the use case through their interaction
  - The blocks send stimuli between one another
  - All stimuli are defined including their parameters
- For each concrete use case, draw an interaction diagram

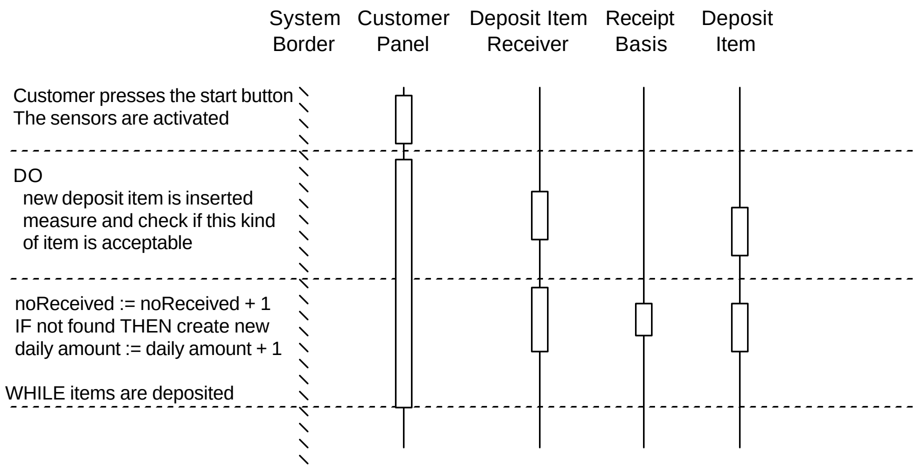
Building an interaction diagram

- Identify blocks
- Draw skeleton, consist of:
  - System border
  - Bars for each block that participates
- Describes the sequences
  - Structured text or pseudo-code
- Mark the bar to which operations belongs with a rectangle representing operation
- Define a stimulus
- Draw a stimulus as a horizontal arrow
  - Start: bar of the sending block
  - End: bar of the receiving block
- Structure the interaction diagram
  - Fork diagram
  - Stair diagram

The Skeleton for The nteraction Diagram



Example – Interaction Diagram for Use Case Returning Item



System Border | Customer Panel | Deposit Item Receiver | Receipt Basis | Deposit Item

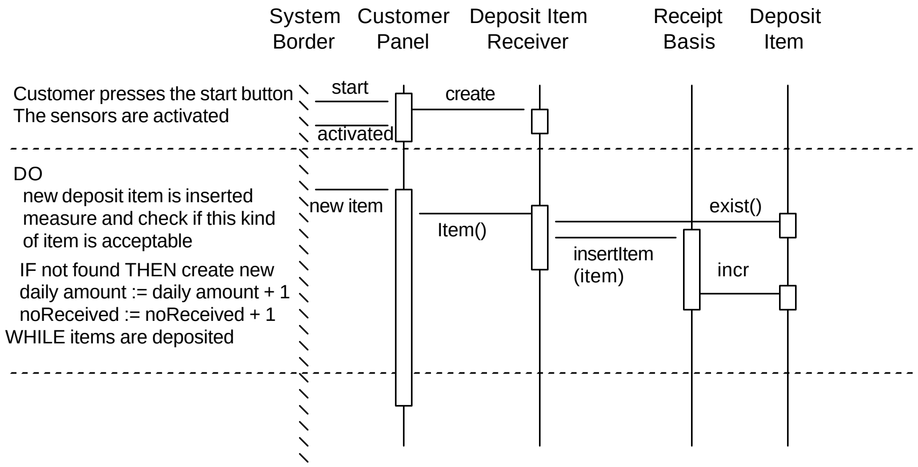
Customer presses the start button
The sensors are activated

DO
new deposit item is inserted
measure and check if this kind of item is acceptable

noReceived := noReceived + 1
IF not found THEN create new
daily amount := daily amount + 1

WHILE items are deposited

Example – Interaction Diagram for Use Case Returning Item



System Border | Customer Panel | Deposit Item Receiver | Receipt Basis | Deposit Item

Customer presses the start button
The sensors are activated

DO
new deposit item is inserted
measure and check if this kind of item is acceptable
IF not found THEN create new
daily amount := daily amount + 1
noReceived := noReceived + 1
WHILE items are deposited

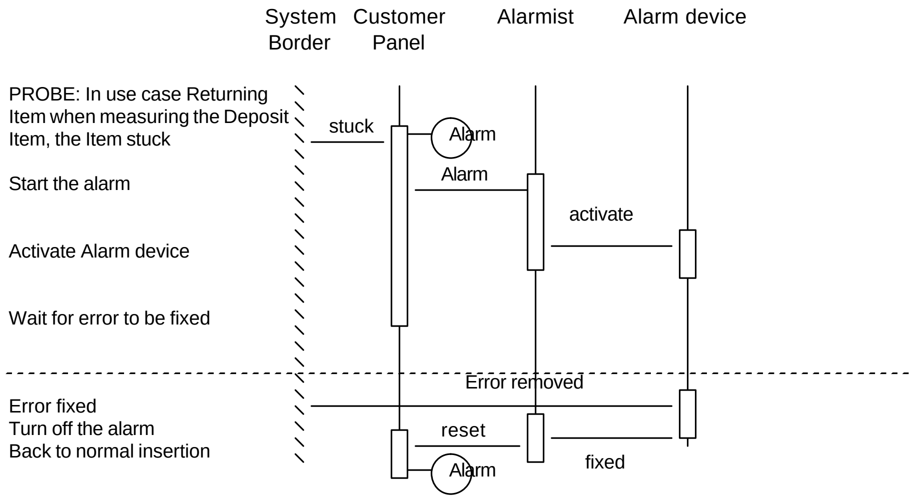
Structure of interaction diagrams

- *Fork*
  - indicates a centralized structure and is characterized by the fact that it is an object controls the other objects interacted with it.
  - This structure is appropriate when:
    - The operations can change order
    - New operations could be inserted
- *Stair*
  - indicates decentralized structure and is characterized by delegated responsibility.
  - Each object only knows a few of the other objects and knows which objects can help with a specific behavior.
  - This structure is appropriate when:
    - The operation have a strong connection. Strong connection exists if the objects:
      - form a ‘consist-of’ hierarchy
      - form an information hierarchy
      - form a fixed temporal relationship
      - form a (conceptual) inheritance relationship
    - The operation will always be performed in the same order

Use Case with extension

- Described by a probe position in the interaction diagram
- The probe position indicates a position in the use case to be extended
  - Often accompanied by a condition which indicates under what circumstances the extension should take placed

Example – Probe position



System Border

Customer Panel

Alarmist

Alarm device

PROBE: In use case Returning Item when measuring the Deposit Item, the Item stuck

Start the alarm

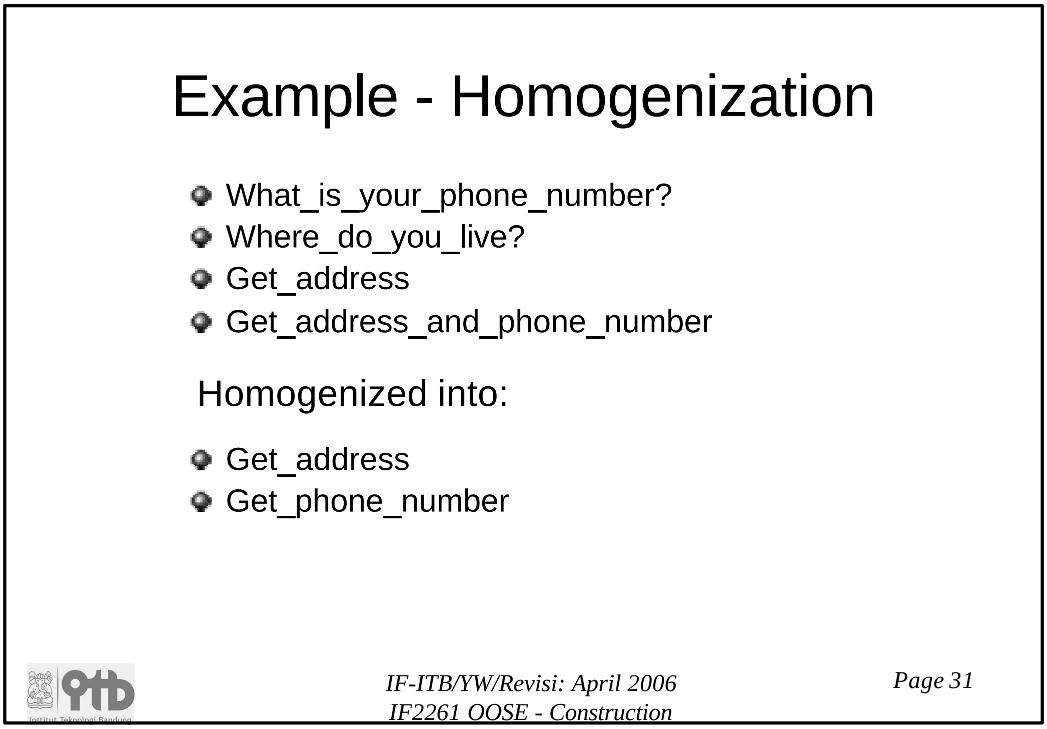
Activate Alarm device

Wait for error to be fixed

Error fixed
Turn off the alarm
Back to normal insertion

Homogenization

- In parallel design process, several stimuli with the same purpose or meaning are defined by several designers.
- These stimuli should be consolidated to obtain as few stimuli as possible.
  - Called *homogenization*.



Example - Homogenization

- What_is_your_phone_number?
- Where_do_you_live?
- Get_address
- Get_address_and_phone_number

Homogenized into:

- Get_address
- Get_phone_number