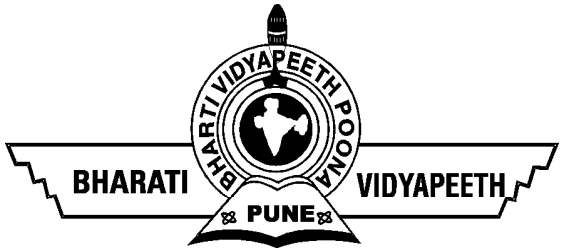


OBJECT ORIENTED SOFTWARE ENGINEERING

Architecture

&

Analysis

UNIT II

System Development as an Industrial Process

Documented by Spector and Gifford(1986) via an interview with Gerald Fox (experienced bridge designer)

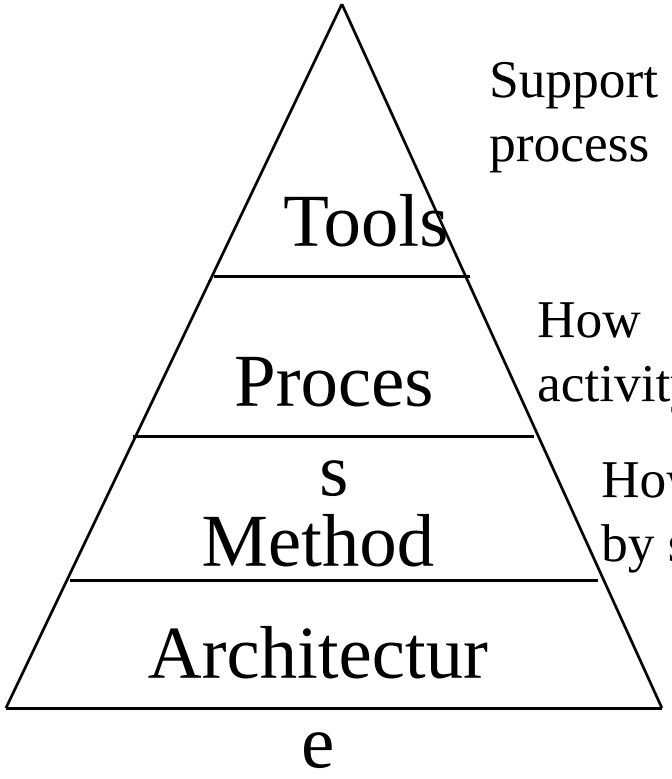
Learning Objectives

- The constitute of a rational enterprise philosophy
- The constituents of multiple activities of a rational enterprise
- Software Industry Direct Analogies with Rational Enterprise
- System development characteristics
- Part of a Larger Activity
- System Development
- Transition from Analysis to Construction
- The System Life Cycle
- System Development as a Process of Change

Learning Objectives Cont..

- System Development and Reuse
- Component
- System Development and Methodology
- Process Expresses more than Method
- Process Adaptation According to Organization Need
- Objectory

Constitute of a Rational Enterprise Philosophy



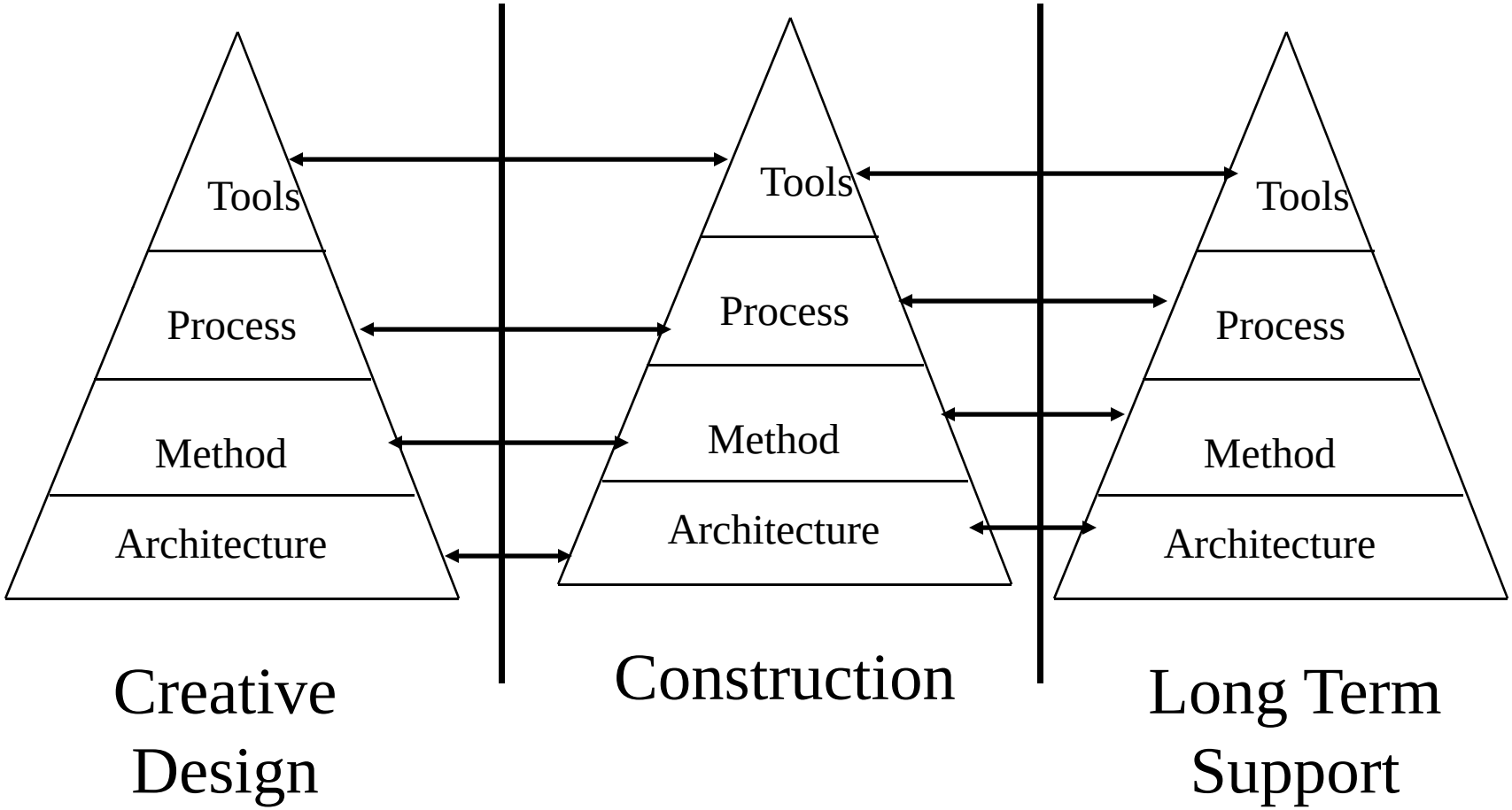
Support for architecture, method or process

How to scale up the method to industry activity

How to apply the architecture concept(step by step)

The selected approach from a universe of approaches (Foundation of concepts and technology)

Constituents of Multiple Activities of a Rational Enterprise



Software Industry Direct Analogies with Rational Enterprise

- Process must yield a forseeable result, irrespective of which individual performed the job
- Volume of output doesn't effect the process
- Possibility to allocate parts of the process to several manufacturers/subcontractors
- Possible to make use of predefined building blocks and components
- Possible to plan and calculate the process with great precision.
- Each person trained for an operation must perform it in a similar manner

System Development Characteristics

Part of a large activity

System Development

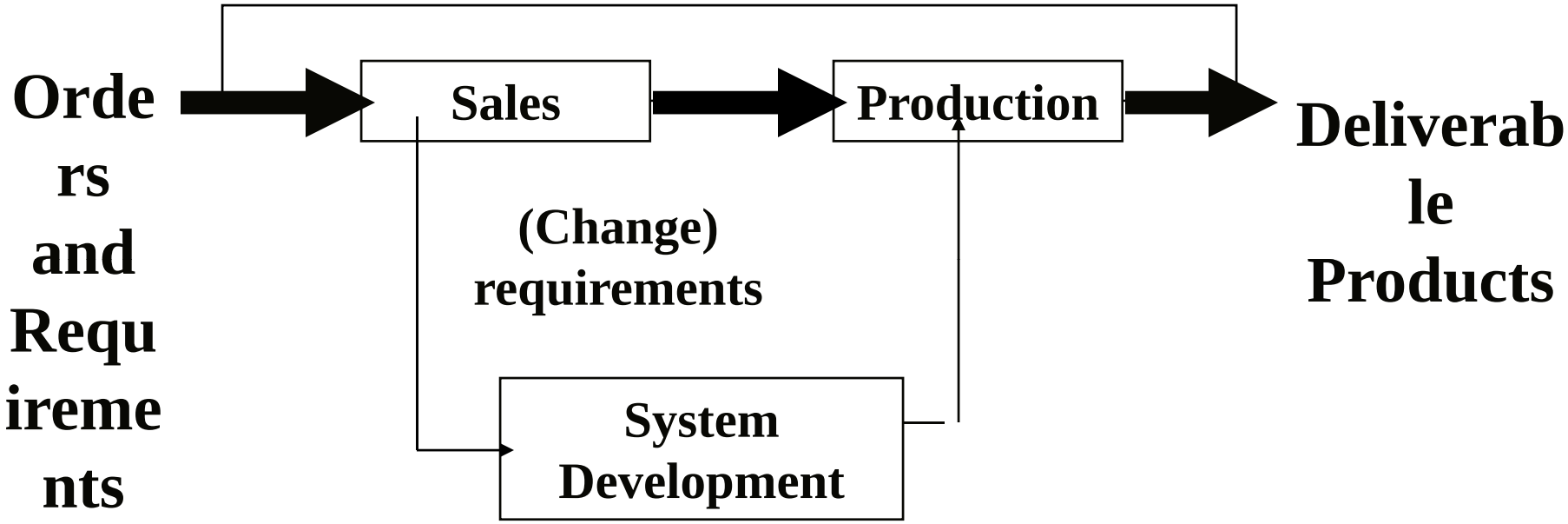
Transition from Analysis to Construction

Requirements are input to system development

A system is output system development

Parties interested in system development are customer, comp. dept., direct users, indirect users`

Part of a Larger Activity



Part of a Larger Activity Cont..

Only person involved in production

- skilled in duplicating products, assembling and configuring system and testing
- To Minimize the development department with customer contacts.

Product is described as sets of packages of functionality services termed as service packages

Service Packages are translated into building blocks

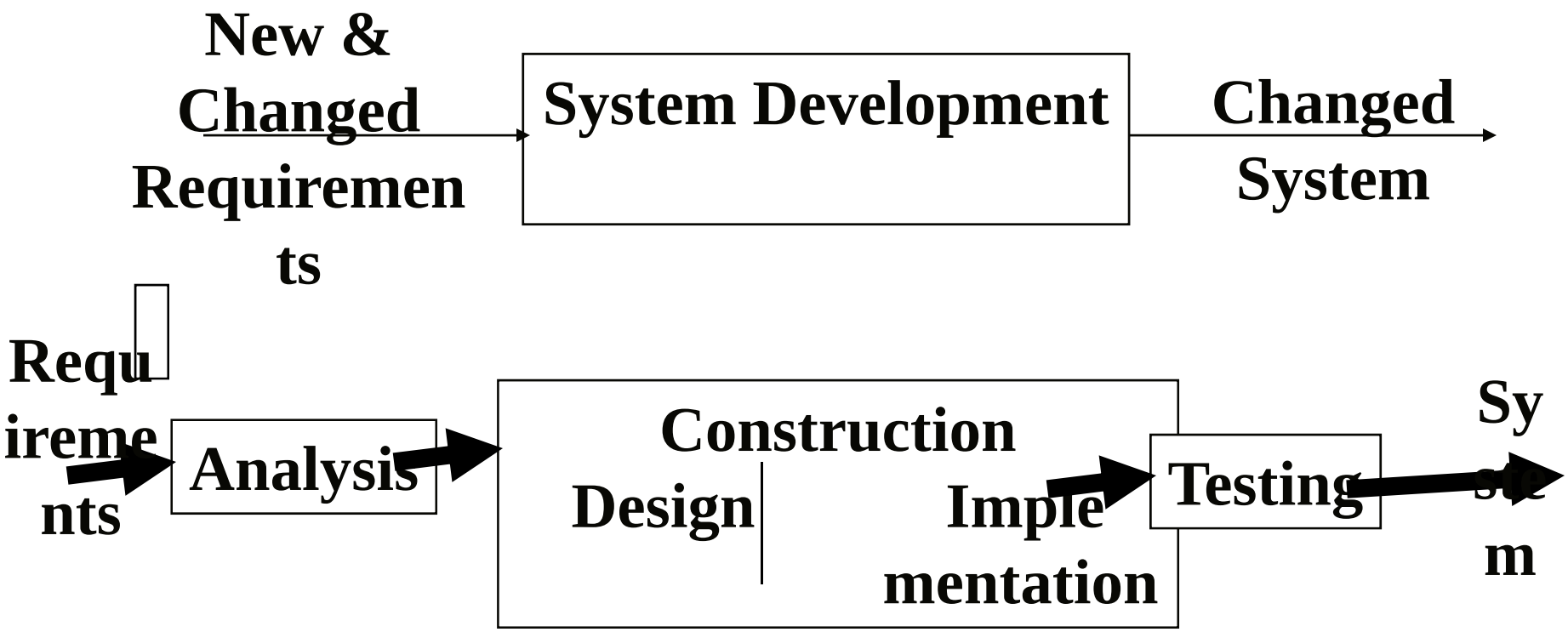
System Development

Gradual transformation of a sequence of models

Can be viewed as a process of producing model description

Consist of three distinct phases that follow each other seamlessly- analysis, construction and testing

System Development Cont..



Transition from Analysis to Construction

Guidelines for analysis

- Analysis must be independent of implementation
- Analysis must be application oriented
- Described in terms of service packages
- Should not be too elaborated

Requirements are Input to System Development

Technical applications

- Tactical command & control system; process control system; telecommunication system
- The role of system in its environment identified & requirements formulated in terms of behavior of sensors and actuators (interfaces to the environment)

Administrative systems

- Requirements based on a dialogue between customer and producer

A System is Output System Development

Set of descriptions

Basis for production in production department and for product description in sales department

The System Life Cycle

System Development as a process of change

System development and Reuse

System Development and methodology

Objectory

System Development as a Process of Change

Delta requirement specification

- Specifications for desired changes to an earlier version

Each new version is a delta version

Process of progressive change

System Development and Reuse

Not a central issue in other branches as so obvious and practiced widely

System is a set of application module

A module composed of other modules or component

Finest level of granularity of reuse is component

Decrease the product's life-cycle model

Other descriptions can be reused such as handbooks and educational material

Component

Already implemented unit

Not quantity but quality determines the success of reuse

Listed in catalogue

Browser to access catalogue

- Each iteration as a change to an existing system

Currently support for data structures and windows management systems in form of macro libraries, subroutine libraries, procedure libraries and so on

Changeable Applications

Must be designed to permit frequent alteration

Modules must be framed to handle change in requirement in all probability in few modules

System Development and Methodology

Important to know how the different steps of the method interact and how they fit into the development process as a whole

Architecture

- Internal structure key property of a system makes it easy to understand, change, test and maintain

- Analysis model, design model and implementation model may form an architecture

System Development and Methodology contd..

Method

- A planned procedure to approach a specified goal step by step
- Simplifies the development of system of a particular architecture

Process

- Natural scaling up of a method

System Development and Methodology Cont..

Method description and process description

Method presented as starting from nothing and resulting in a first system version

Project includes changing an existing system, tendering project, error handling project and others

Process offers to more than one project

Activity description should be reusable in different types of systems

System Development and Methodology cont..

Process will continue to exists so long as the developed system is in operation

A set of interacting sub process

Software factory describe the division of processes and sub processes

A process presuppose a method which, in turn presuppose an architecture

Process Expresses More Than Method

Transition from principles to practice

Describe how different subprocesses should cooperate and to what extent they should be carried out in parallel

Activity becomes less dependent on individual

System development as process of change

CASE

Support the process it is based on

Part of a greater whole

Process Adaptation According to Organization Need

Heterogeneous projects

Development case description about pros and cons of deselecting of subprocess

Process adaptor must have access of development case description

Objectory

Modern Development Technique

- Support of iterative development
- Support the entire chain from changed requirement to the functioning system

Object Factory for Software Development a development technique

Framework: Design with building blocks

Objectory Cont..

Essential idea behind the CCIT(Commite Consultatif International Telegraphique et Telephonique

Used in various systems information system, , real time system, process control system, CASE tools and graphic presentation systems. Adapted to different programming languages

Objectory Cont..

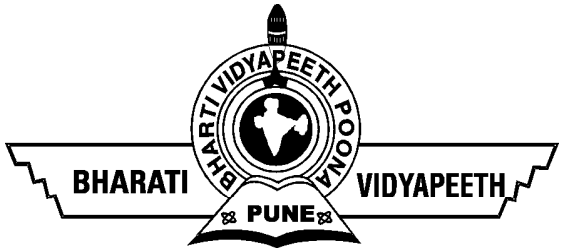
Simplified and scaled down to small scale projects with large projects

Three design techniques: building blocks, conceptual modeling and object oriented programming

Conceptual modeling extended with OO concepts and ability to describe dynamic behavior

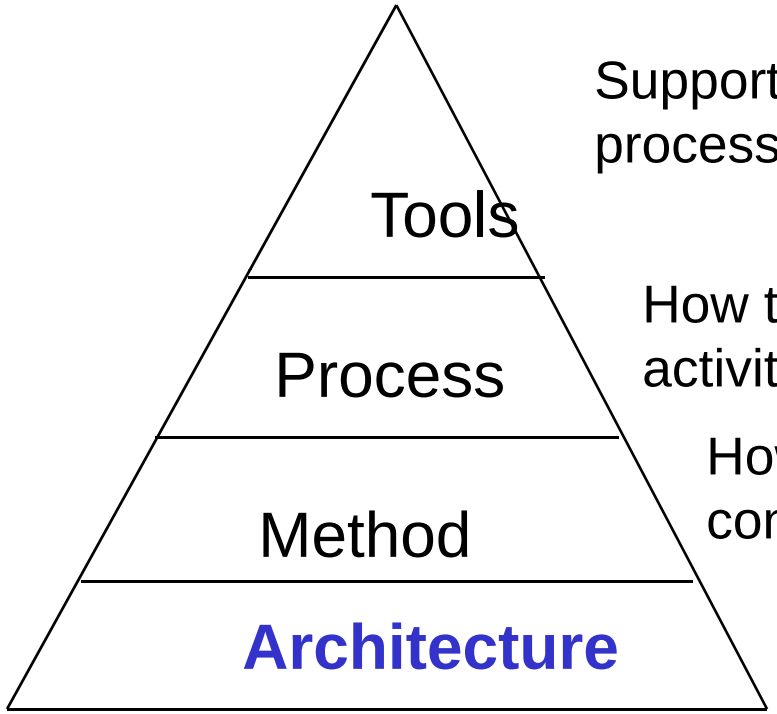
Learning Objectives

- **Architecture**
- Introduction
- System development is model building
- model architecture
- requirements model
- analysis model
- the design model
- the implementation model
- test model
- **Analysis**
- Introduction
- the requirements model
- the analysis model



Architecture

Learning Objective



Support for architecture, method or process

How to scale up the method to industry activity

How to apply the architecture concept(step by step)

The selected approach from a universe of approaches (Foundation of concepts and technology)

Learning Objectives Cont..

- System Development
- Basis for approach
- Models
- Architecture
- Development processes
- Processes and Models
- Model Architecture
- Requirement Model
- Problem Domain object Model
- Analysis Model
- Design Model
- Implementation Model
- Test Model

System Development

- The work that occurs when we develop computer support to aid an organization
- System development is model building
- Starts when a requirement of system identified
- Specification can be used for contract and to plan and control the development process
- Complex process handle poorly
- OOSE can used from start to end of system life cycle

Basis for Approach

Based on three technologies

- Object oriented programming
- Conceptual modeling
- Block design

Models

Five different models

- The requirement model
  - ✓**Aims to capture the functional requirements**
- Analysis model
  - ✓**Give the system a robust and changeable object structure**
- Design model
  - ✓**Adopt and refine the object structure to the current implementation environment**

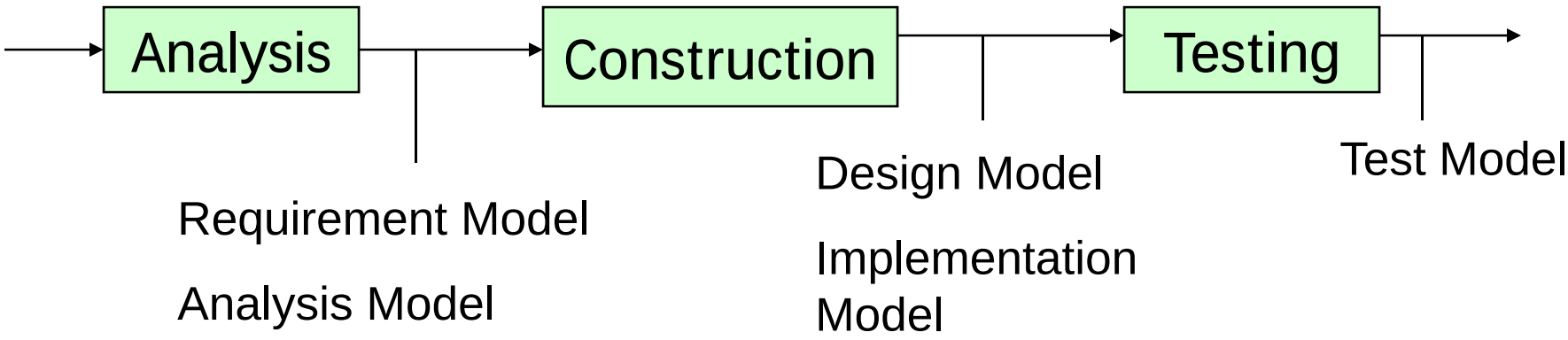
Models cont..

Implementation model

- Implement the system

Test model

- Verify the system



Models Cont..

- Seamless transition between the models is important
- The method layer define the transformation rules
- For maintainability traceability is very important between the models

Models Cont..

Models are tightly coupled to the architecture, and aim is to find concepts which

- Are simple to learn and use
- Simplify our understanding of the system
- Provide us with a changeable model of the system
- Are sufficiently powerful to express the information which is required to model the system
- Are sufficiently defined that different people can discuss the system in terms of these concepts without being misunderstood

Architecture

- System development includes the development of different models of a software system
- Aim is to find powerful modeling language, notation or modeling technique for each model
- Set of modeling techniques defines **architecture** upon which the system development method is based
- The architecture of a method is the denotation of its set of modeling techniques

Architecture Cont..

- Architecture can be view as the class of models that can be built with a certain modeling notation
- Modeling technique is described by means of syntax, semantics and pragmatics
  - Syntax (How it looks)
  - Semantics (What it means)
  - Pragmatics (heuristics or rules of thumb for using modeling technique)

Architecture Cont..

- Features of modeling techniques
  - Easy to use
  - Contain few but powerful modeling objects to enable easy learning
  - Help to handle complexity of the system

- To build these models method are required to show how to work with modeling techniques

Architecture Cont..

- The specific system architecture is formulated in terms of the modeling object used

- Specific system architecture is the result obtained after applying a method to a system.

Architecture Cont..

- System that share a similar high-level structure are said to have a similar architecture style
- Approaching reuse without understanding architecture style is like starting to construct a building without knowing if it will be a skyscraper or a garage
- Software of the same architecture style can be reused easier than those of different architecture styles

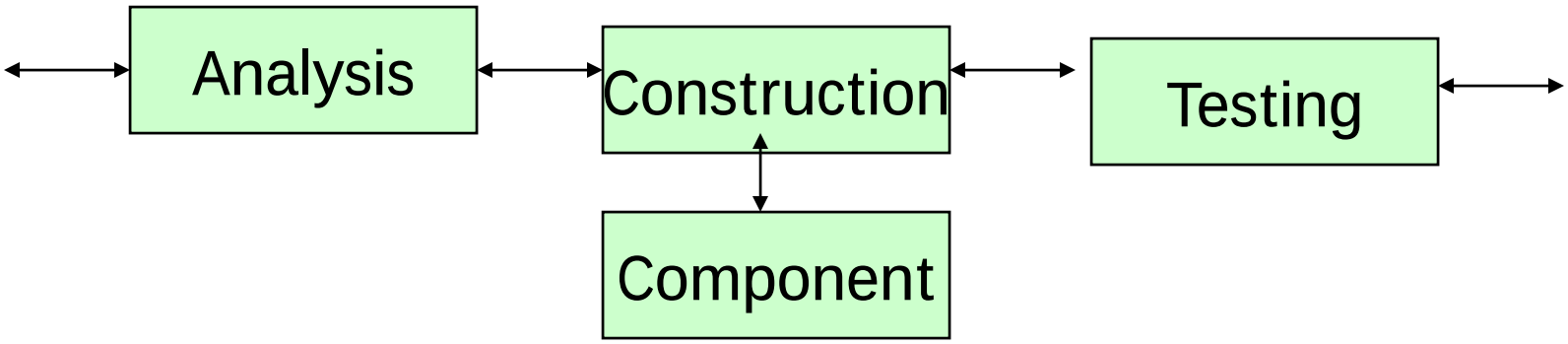
Development Processes

- Instead of focusing on how a specific project should be driven,the focus of the process is on how a certain product should be developed and maintained during its life cycle
- Divide the development work for a specific product into processes, where each of the processes describes one activity of the management of a product
- Process works highly interactively
- Process handles the specific activity of the system development

Development Processes cont..

- All development work is managed by these processes.
- Each process consist of a number of communicating sub processes
- Main processes are
  - Analysis
  - Construction
    - ✓**Component**
  - Testing

Development Processes Cont..



Analysis Process

- Creates conceptual picture
- Output are requirement model and Analysis model
- Requirement Model
  - ✓**Done by use cases in the use case model**
  - ✓**Form the basis of construction and testing process**
  - ✓**Forms the basis of analysis Model**

Development Processes Cont..

- **Analysis Model**
  - ✓**Basis of system structure**
  - ✓**Specify all the logical objects to be included in the system and how these are related and grouped**
  - ✓**Provide input for the construction process**

Construction Process

- **Develops design model and implementation model**
- **Includes the implementation and results in complete system**
- **Design Model**
  - ✓**Each object will be fully specified**
  - ✓**Consider the implementation constraints**

Development Processes Cont..

Testing Process

- Integrates the system, verifies it and decides whether it should be delivered

Component development process

- Communicates with the construction process
- Develops and maintain components

Development Processes Cont..

Analysis

Requirement Analysis

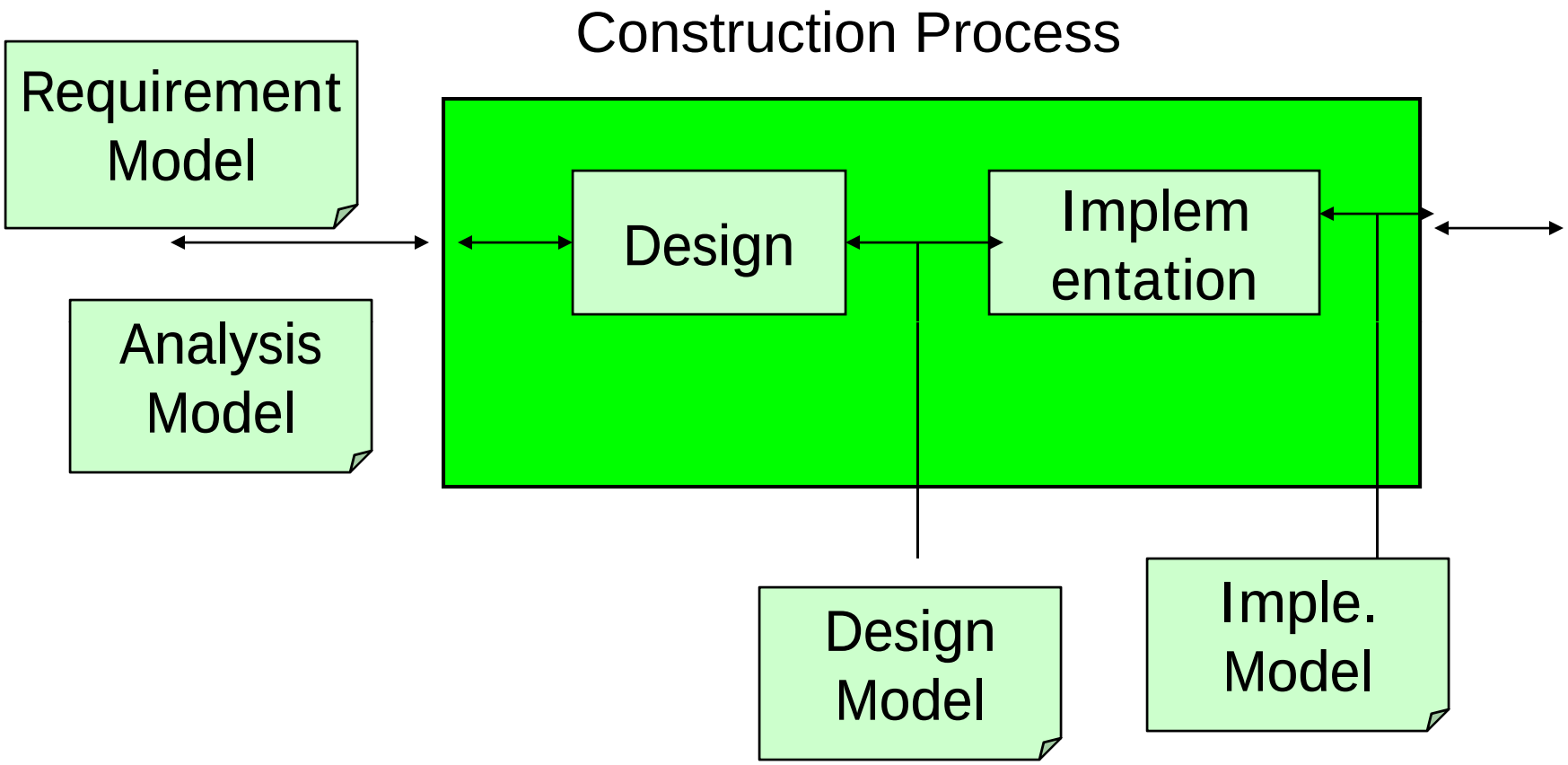
Robustness analysis

Requirements model

Analysis model

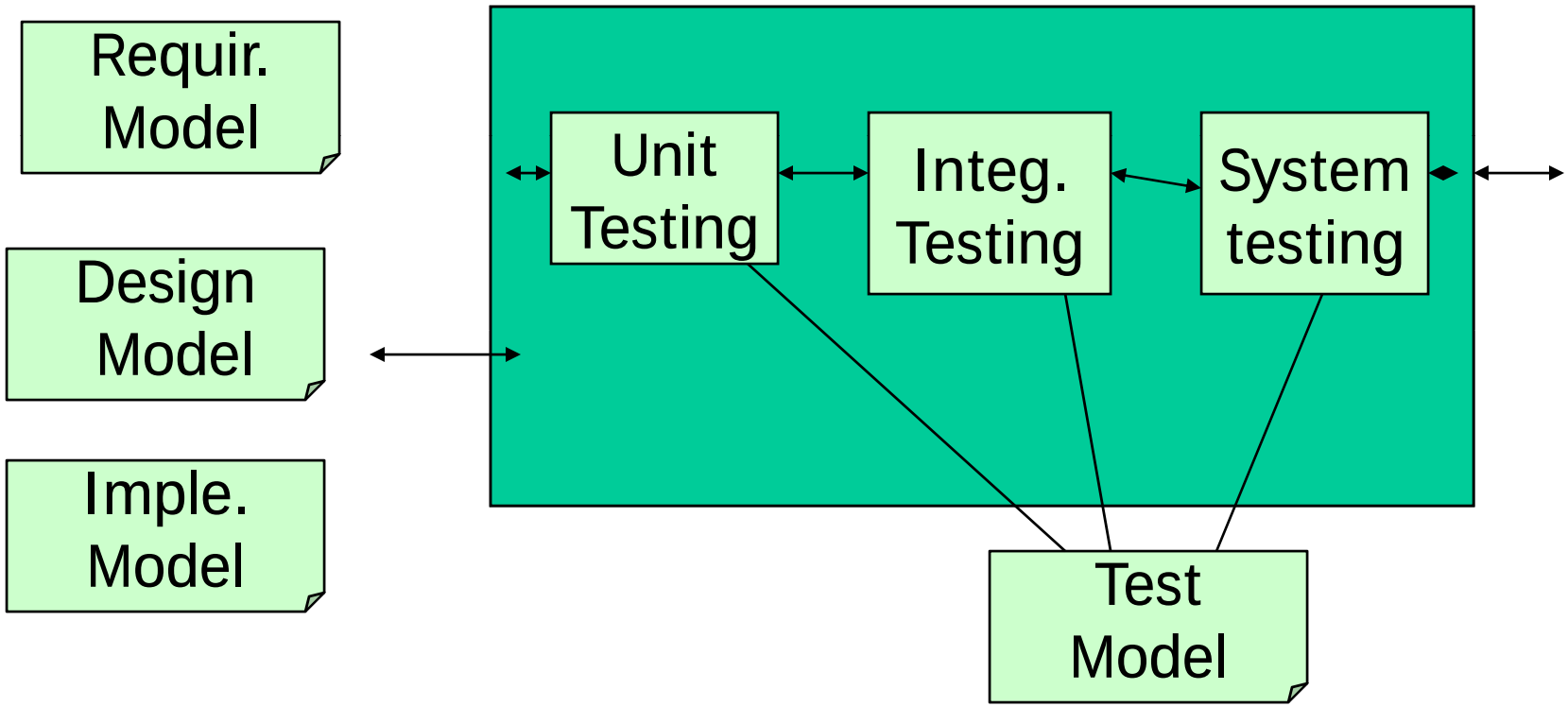
Requirements

Development Processes Cont..



Development Processes Cont..

Testing Process



Development Processes Cont..

- Architecture forms the basis of the method and process, that is the concept of each model
- Development can be regard as a set of communicating processes
- System development iterates over these processes

Processes and Models

- Models of the system created during development
- To design models process description is required
- Each process takes one or several models and and transform it into other models
- Final model should be complete and tested, generally consists of source code and documentation
- Process follow a product and exist as long as the product exist

Model Architecture

- System development is basically concerned with developing models of the system

- Work is concerned with identifying and describing objects in a certain information space and with building models using these objects

Model Architecture Cont..

Object features

- Have an interpretation in reality
- Should be obvious
- Tangible
- Can be focus
- During evaluation of system object should be included or left out

Model Architecture Cont..

Features of good object model

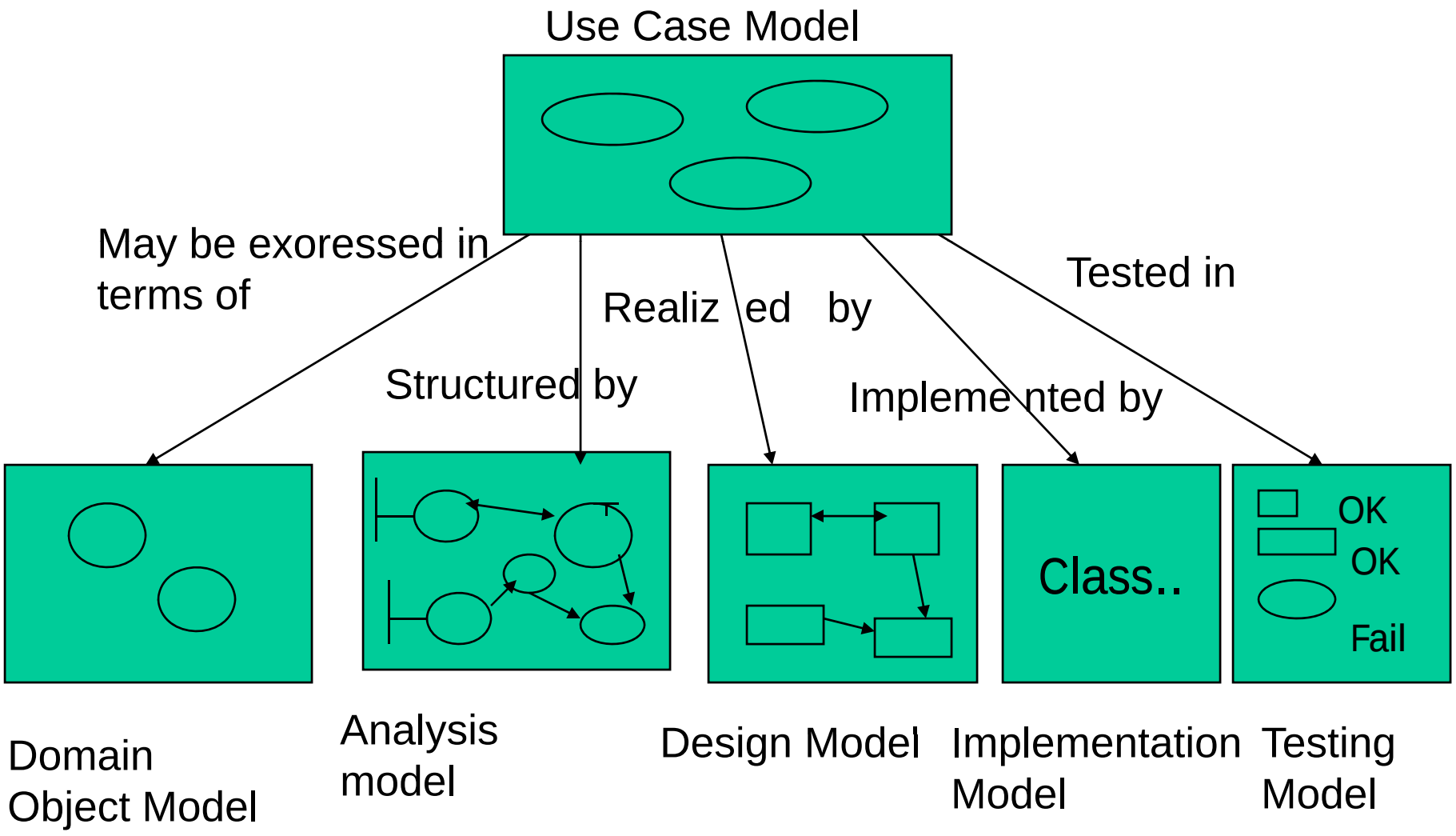
- Should be robust against modification
- Help the understanding of system

Requirement Model

Consist of

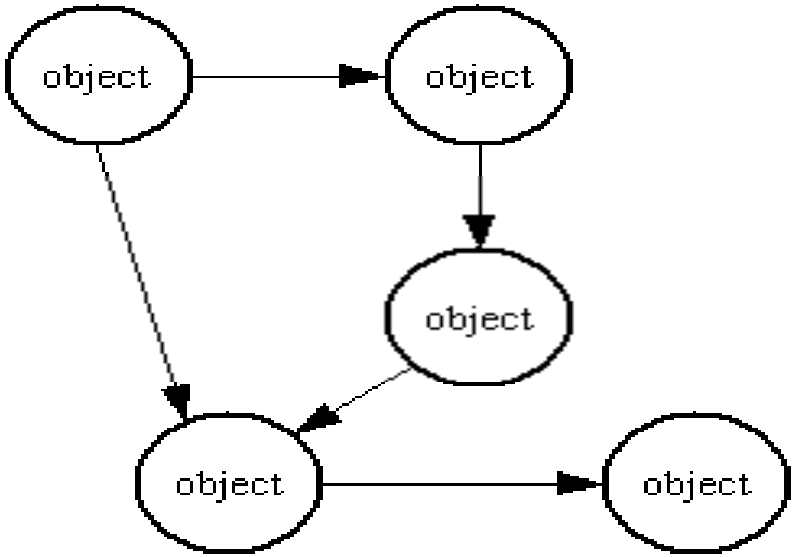
- A use case model
- Interface description
- Problem domain model

Requirement Model Cont..

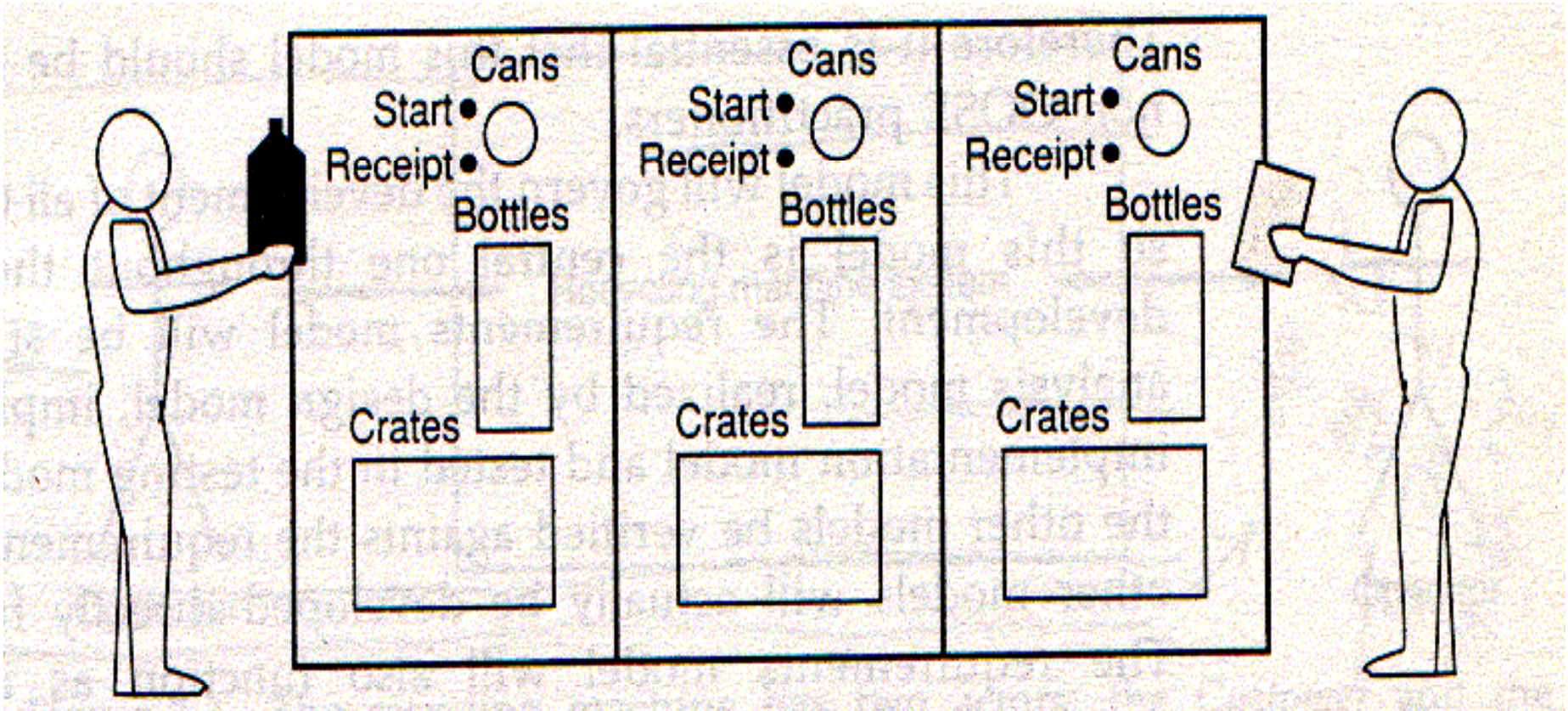


Problem Domain Object Model

Provides a logical view of the system, which is used to specify the use cases for use case diagrams



Example



The recycling machine can receive several different types of returnable items from several customers at the same time.

Example Cont..

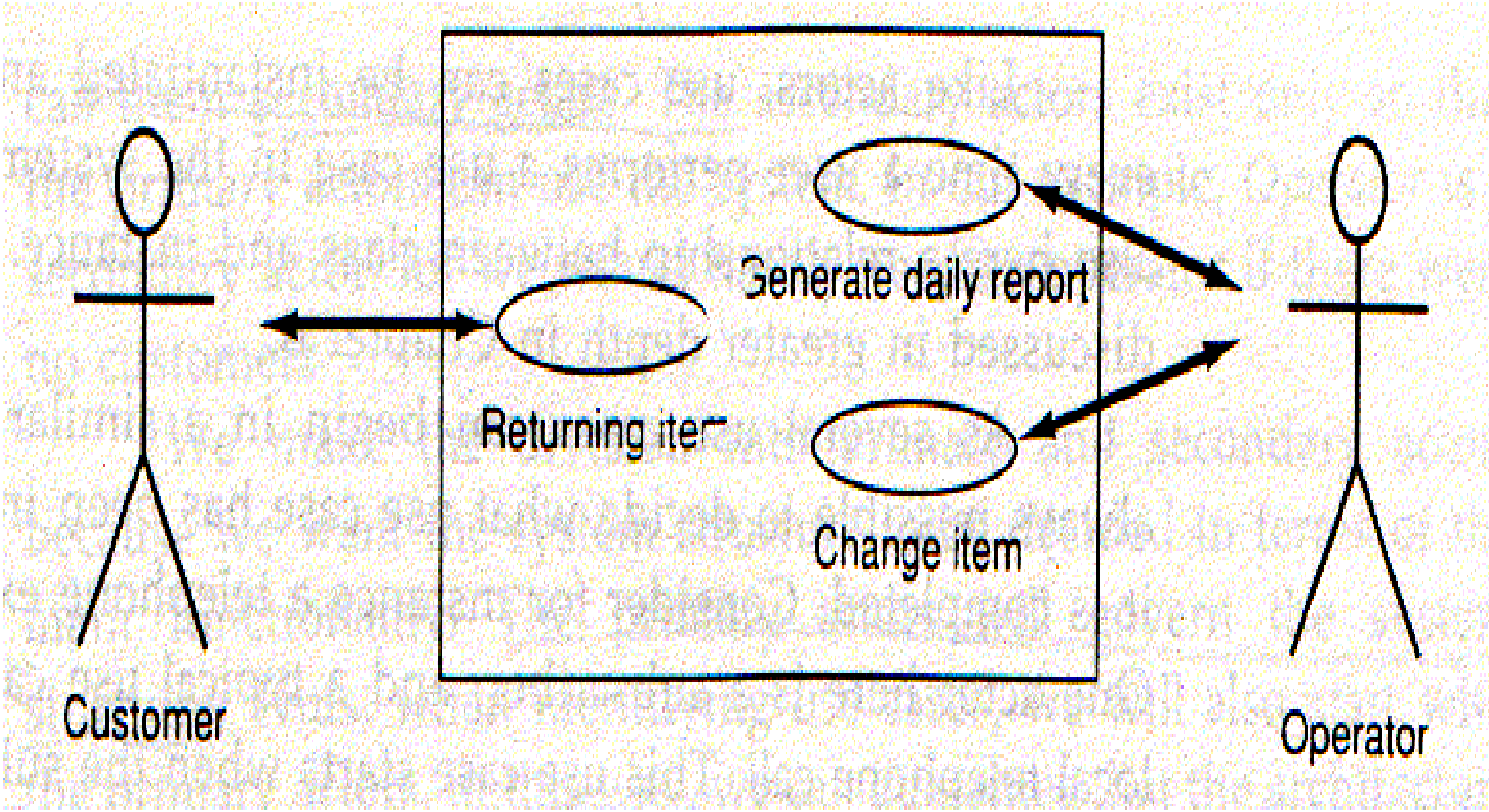
Throughout the discussion of the analysis and the construction activities, we will show how the different concepts are used in practice, by developing a system. The system controls a recycling machine for returnable bottles, cans and crates (used in Europe to hold several bottles). The machine can be used by several customers at the same time, and each customer can return all three types of items on the same occasion,

Since there may be different types and sizes of bottles and cans, the system has to check, for each item, what type was turned in. The system will register how many items each customer returns, and when the customer asks for a receipt, the system will print out what he turned in, the value of the returned items and the total return sum that will be paid to the customer.

Example Cont..

The system is also used by an operator. The operator wants to know how many items of each type have been turned in during the day. At the end of the day, the operator asks for a printout of the total number of items that have been turned in to the system on that particular day. The operator should also be able to change information in the system, such as the deposit values of the items. If there is something amiss, for instance if a can gets stuck, or if the receipt roll is finished, the operator will be called by a special alarm signal.

Use Case Diagram



Use Case Description

Returning Item is started by *Customer* when he wants to return cans, bottles or crates. With each item that *Customer* places in the recycling machine, the system will increase the received number of items from *Customer* as well as the daily total of this particular type. When *Customer* has turned in all his items, he will press the receipt button to get a receipt where the returned items have been printed as well as a total return sum.

Generate Daily Report is started by *Operator* when he wants to print out information about the returned deposit items of the day. The system will print out how many of each deposit item type have been received this day, as well as the overall total for the day. The total number will be reset to zero to start a new daily report.

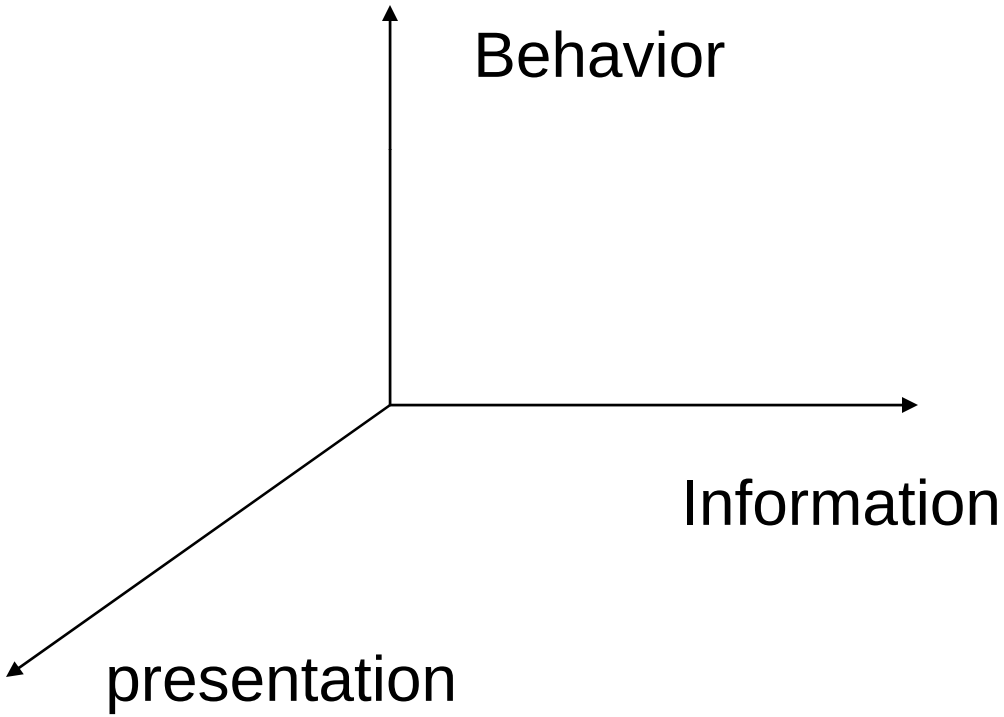
Change Item is used by *Operator* to change information in the system. The return value as well as the size of each returnable item can be changed, and new types of items can be added.

Analysis Model

- Requirement model aims to define the limitations of the system and to specify its behavior
- Development of actual system starts with analysis model
- Focus on the logical structure
- Define the stable, robust and maintainable structure that is also extensible
- Creating analysis model corresponding to assigning functionality of use cases to objects

Analysis Model Cont..

Dimensions of the analysis model



Analysis Model Cont..

Entity object

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

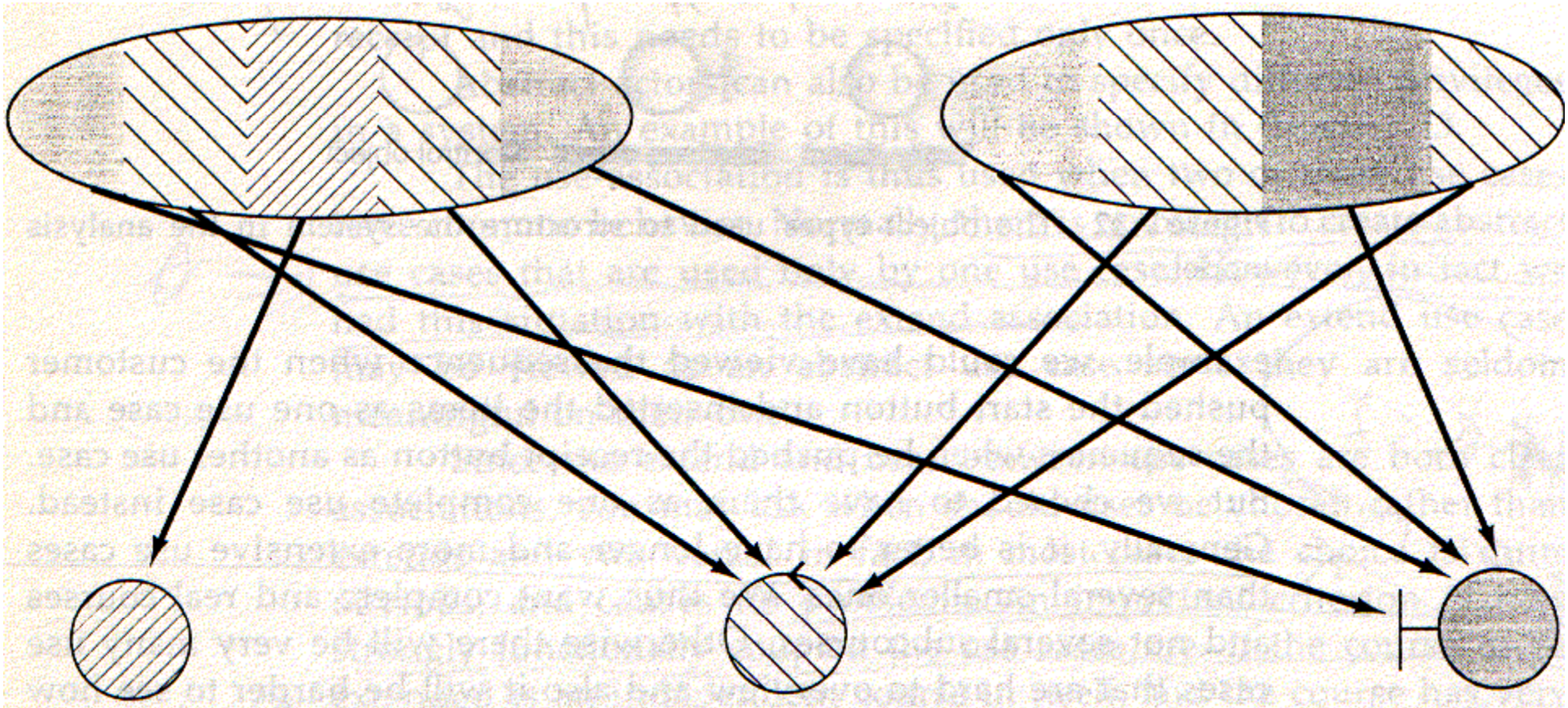
Control object

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

Interface object

- Interface objects interact directly with the environment

Requirement Model Structured in Analysis Model



Each use case is distributed among analysis objects. Several use cases can have objects in common.

Design Model

- Developed based on the analysis model
  - implementation environment is taken into consideration

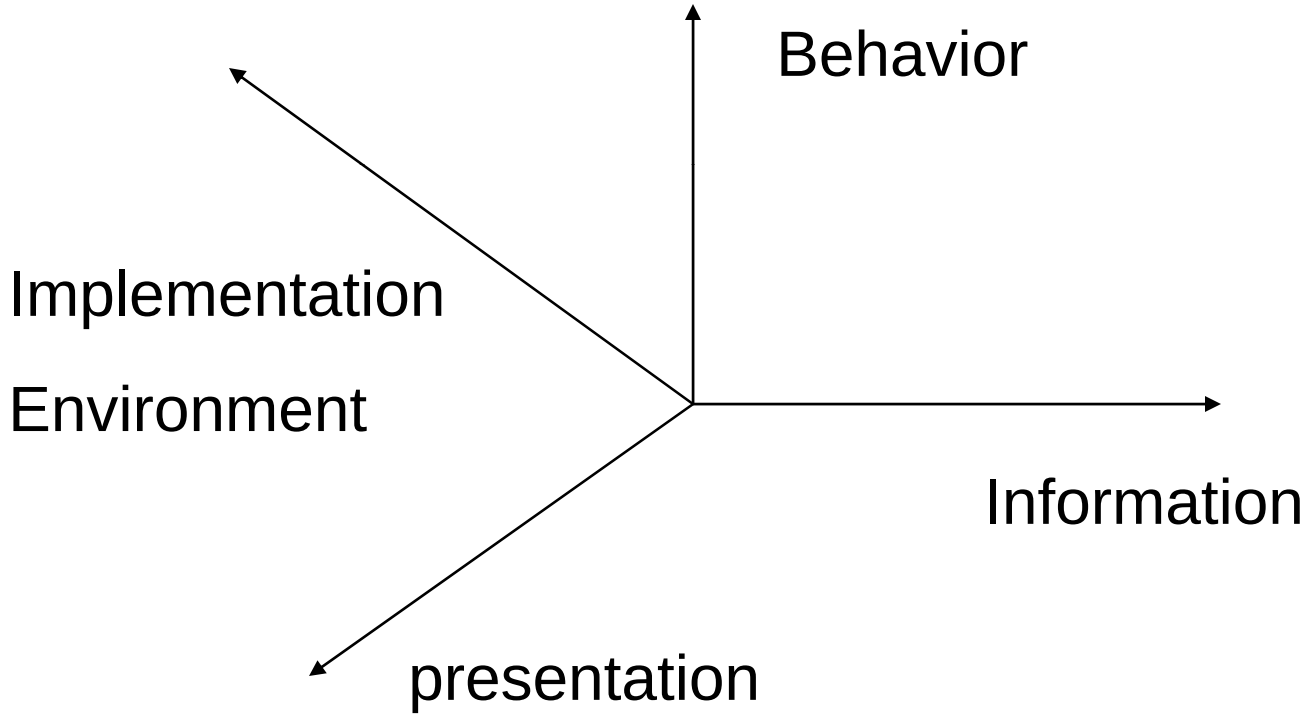
- The considered environment factors includes
  - Platform
  - Language
  - DBMS
  - Constraints
  - Reusable Components
  - Libraries
  - so on..

Design Model Cont..

- Design objects are different from analysis objects
- models
  - Design object interactions
  - Design object interface
  - Design object semantics
    - ✓**(i.e., algorithms of design objects' operations)**
- More closer to the the Actual source code

Design Model Cont..

Dimensions of the Design model



Design Model Cont..

- Use block term in place of object
- Stimuli is sent from one block to another to trigger an execution
- A typical block is mapped to one file
- To manage system abstractly subsystem concept is introduced
- Analysis Model is viewed as conceptual and logical model, whereas the design model should take as closer to the actual source code

Implementation Model

- Consist of annotated source code
- OO language is desirable since all fundamentals concepts can easily be mapped onto language constructs
- Strongly desirable to have an easy match between a block and the actual code module

Test Model

Fundamental concepts are test specifications and the test results

What we Learnt..

- ✓ System Development
- ✓ Basis for approach
- ✓ Models
- ✓ Architecture
- ✓ Development processes
- ✓ Processes and Models
- ✓ Model Architecture
- ✓ Requirement Model
- ✓ Problem Domain object Model
- ✓ Analysis Model
- ✓ Design Model
- ✓ Implementation Model
- ✓ Test Model

Analysis

Learning Objectives

- The Analysis Phase
- Analysis Model
- Analysis Artifacts
- Meta Model of Analysis Model
- Analysis workflow details
- Analysis model-rules of thumb
- Object Oriented Analysis
- Three ways to do Object Oriented Analysis
- Conceptual Model – Overview
- The Concept Category List
- Finding Concepts with Noun Phrase Identification

Learning Objectives Cont..

- Exercise
- How to make a conceptual model
- Drawing of Concepts
- Adding Associations
- Adding Attributes
- CRC cards & Role playing
- The Object Oriented Analysis Model (Jacobson)
- Subsystem
- Good Analysis class
- Bad Analysis class

The Analysis Phase

- Begins with a problem statements generated during system conception
- In software engineering, analysis is the process of converting the user requirements to system specification (system means the software to be developed).
- System specification, also known as the logic structure, is the developer's view of the system.
- Function-oriented analysis
  - **concentrating on the decomposition of complex functions to simply ones.**
- Object-oriented analysis
  - **identifying objects and the relationship between objects.**

Analysis Model

- Always in the language of the business
- Captures the big picture
- Contains artifacts that model the problem domain
- Tells a story about the desired system
- Is useful to a many of the stakeholders as possible

Analysis Artifacts

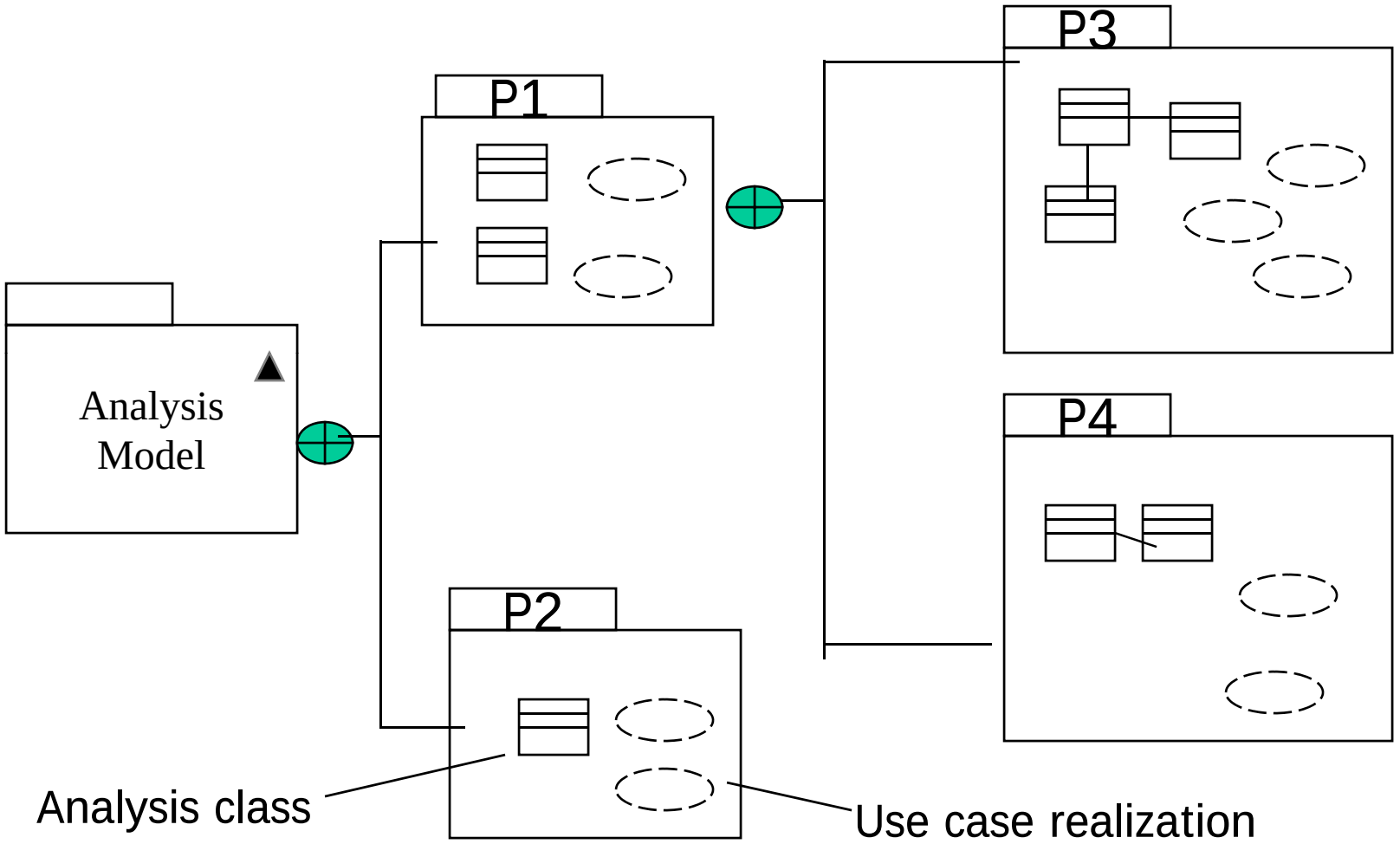
Analysis classes

- Model key concepts in the business domain

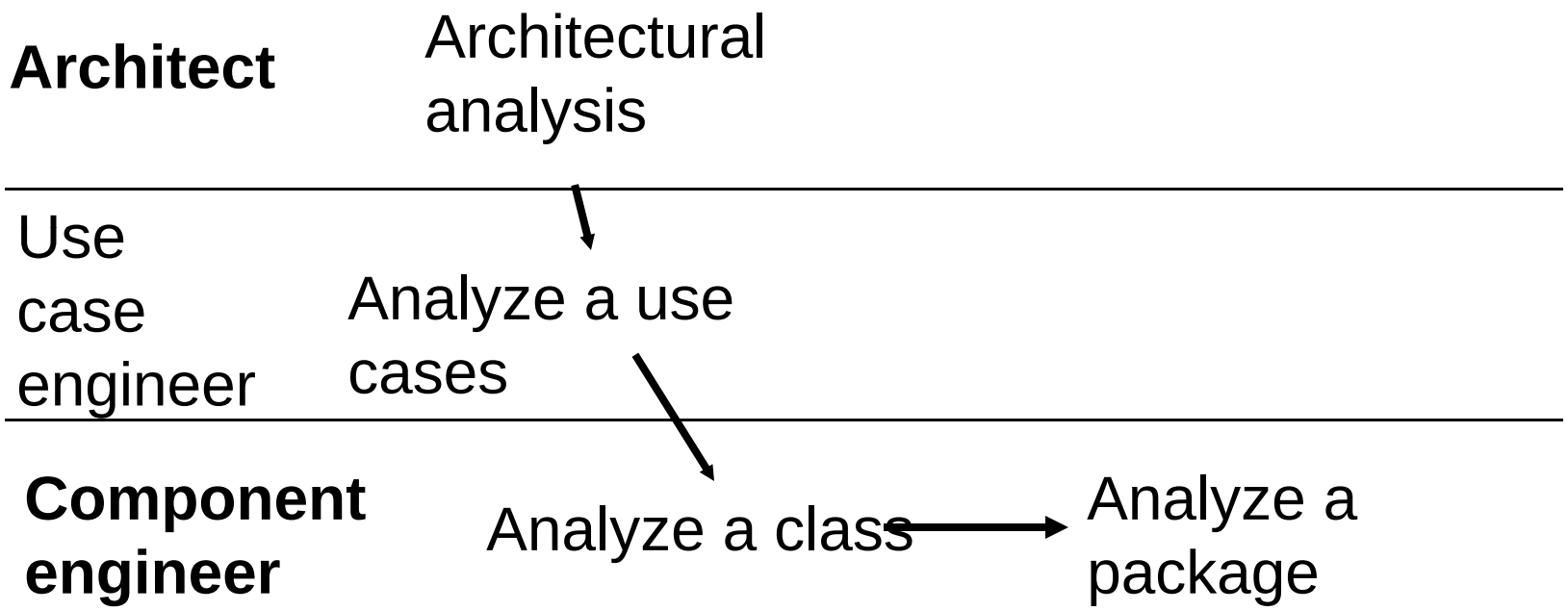
Structuring Use case

- Illustrate how instances of the analysis classes structured system in robust form specified by a use case

Meta Model of Analysis Model



Analysis Workflow Details



Analysis Model-Rules of Thumb

- 50 to 100 analysis classes in analysis model in average system
- Include classes from the vocabulary of problem domain
- Do not make implementation decisions
- Focus on classes & associations- minimize coupling
- Use inheritance when needed
- Keep it simple

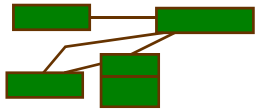
Object Oriented Analysis

- **Identifying objects**: Using concepts, CRC cards, stereotypes, etc.
- **Organising the objects**: classifying the objects identified, so similar objects can later be defined in the same class.
- **Identifying relationships between objects**: this helps to determine inputs and outputs of an object.
- **Defining operations of the objects**: the way of processing data within an object.
- **Defining objects internally**: information held within the objects.

Three ways to do Object Oriented Analysis

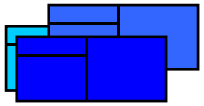
Conceptual model (Larman)

- **Produce a "light" class diagram.**



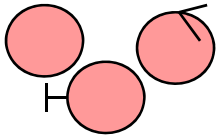
CRC cards (Beck, Cunningham)

- **Index cards and role playing**.



Analysis model with stereotypes (Jacobson)

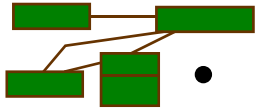
- **Boundaries, entities, control.**



A good analyst knows more than one strategy and even may mix strategies

Conceptual Model - Overview

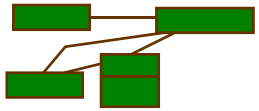
- Representation of concepts in a problem domain.



- In UML it is basically a “class diagram without operations”.

- It may show:
  - Concepts
  - Associations of concepts
  - Attributes of concepts

Conceptual Model Cont..



Strategies to Identify Concepts

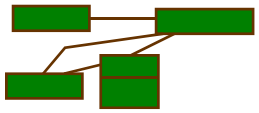
Finding Concepts with the

- **Concept Category List.**
- **Noun Phrase Identification.**

A central distinction between object oriented and structures analysis: division by concepts (objects) rather than division by functions.

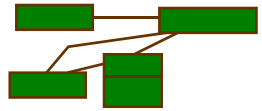
The Concept Category List

- physical or tangible objects
- specifications, designs, descriptions of things
- places
- transactions
- transaction line items
- roles of people
- containers of other things
- things in a container
- abstract noun concepts
- organisations
- events
- processes
- rules and policies
- catalogues
- records
- services
- manuals, books



Noun Phrase Identification

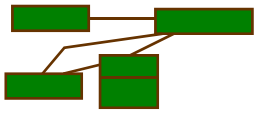
- Identify the noun and noun phrases in textual descriptions of a problem domain

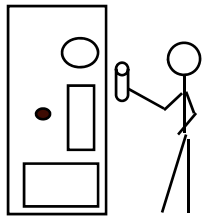


- Consider them as candidate concepts or attributes.
- Care must be applied with this method.
- Mechanical noun-to-concept mapping isn't possible
- Words in natural languages are ambiguous (especially English).

The “return item” Use case-Exercise: Find the Nouns!

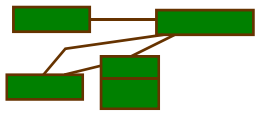
- The system controls a recycling machine for returnable bottles, cans and crates. The machine can be used by several customers at the same time and each customer can return all three types of item on the same occasion. The system has to check, for each item, what type has been returned.
- The system will register how many items each customer returns and when the customer asks for a receipt, the system will print out what was deposited , the value of the returned items and the total return sum that will be paid to the customer.
- An operator also … (not in “return item” Use Case)

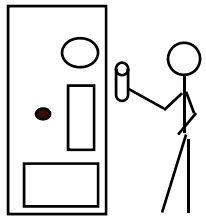




Case Study: Find the Nouns..

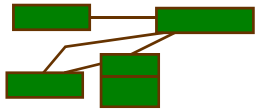
- The **system** controls a **recycling machine** for returnable **bottles**, **cans** and **crates**. The **machine** can be used by several **customers** at the same time and each **customer** can return all three **types** of **item** on the same occasion. The **system** has to check, for each **item**, what type has been returned.
- The **system** will register how many items each **customer** returns and when the **customer** asks for a **receipt**, the **system** will print out what was deposited , the value of the **returned items** and the total **return sum** that will be paid to the **customer**.
- An operator also … (not in “return item” Use Case)

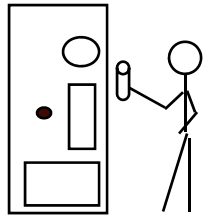




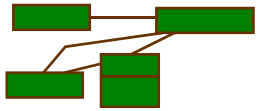
Case Study: Nouns found in the description:

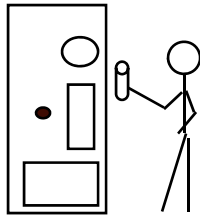
- recycling machine
- bottles, cans, and crates
- machine
- customers, customer
- types of item, item, type, returned items
- system
- receipt
- return sum





Case Study: Discussion of “recycling machine”.





recycling machine

bottles, cans and crates

machine

customers, customer

types of item, item, type, returned

sys

rec

ret

- This concept is the “overall system”
- As we consider only one single use case,
- better to name this concept in the context of this use case, e.g.
  - **Deposit item receiver**

Case Study: Discussion of “bottles, cans, and crates”.

deposit item receiver

bottles, cans, and crates

machine

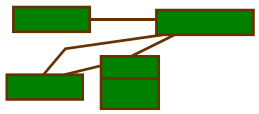
customers, customer

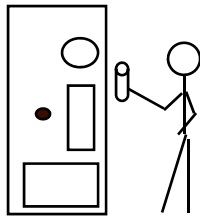
types of items

system

receipt

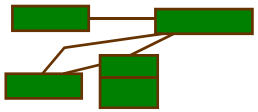
return su

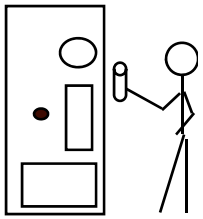




- Usually better to use singular and multiplicities instead of plural.
- As **bottle**, **can** and **crate** have much in common (they are processed as items),
- they could be generalised to an “item”. We should remember this for later (**inheritance**).

Case Study: Discussion of “machine” and “system”.





deposit item receiver

bottle, can, crate

machine

customers, customer

types of item, item items

system

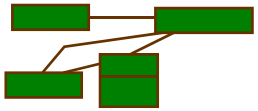
receipt

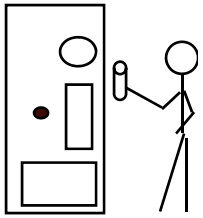
return sum

“Machine” and “System” mean here the same, namely the “Recycling machine”, i.e. the

- **Deposit item receiver**

Case Study: Discussion of “customers” and “customer”.





deposit item receiver

bottle, can, crate

customers, customer

types of item, item, type, re

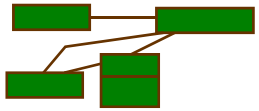
items

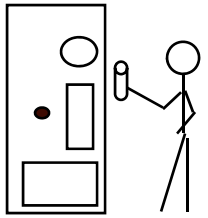
receipt

return sum

- The customer has already been identified as an actor.
- They are outside of the system.
- We establish a concept, that interfaces with the customer (and is inside the system):
  - **Customer panel**

Case Study: Discussion of “item” (etc.).





deposit item receiver

bottle, can, crate

customer panel

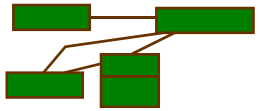
types of item, item, type, returned items

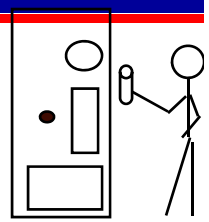
receipt

return sum

- The items that are inserted in the machine.
- Good candidate as superclass for bottle, can, crate.
- Let’s call it
  - **Deposit item**

Case Study: Discussion of “receipt”.





deposit item receiver

bottle, can, crate

customer panel

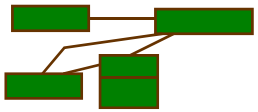
deposit item

receipt

return sum

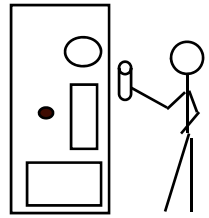
- The concept that “remembers” all items inserted in the machine.
- To distinguish it from the piece of paper returned to the customer, call it
  - **Receipt basis**

Case Study: Discussion of “return sum”.



deposit item receiver

bottle, can, crate



customer panel

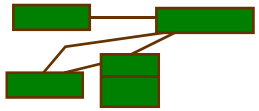
deposit item

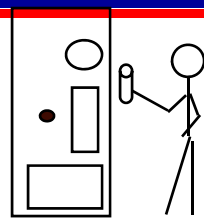
receipt basis

return sum

- The sum that it is returned to the customer is actually computed by adding up all values of the items stored in the receipt basis.
- The sum itself is only a **primitive data value,** and may therefore not be considered as a concept.

Case Study: Discussion of Other Concepts





deposit item receiver

bottle, can, crate

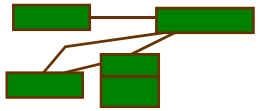
customer panel

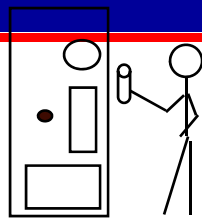
deposit item

receipt basis

- These are the concepts identified by nouns. Did we forget something?
- Check the "Concept Category List" !
- The system "**interfaces**" with the physical object "printer", so we add an interface concept
  - **Receipt printer**

Case Study: Summary of Concepts Identified in the Analysis





deposit item receiver

bottle, can, crate

customer panel

deposit item

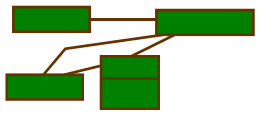
receipt basis

receipt printer

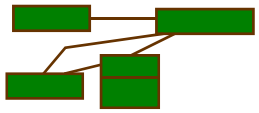
- So far we have identified:
  - Concepts
  - A generalisation relationship.
- **Next step: Finding associations.**

How to Make a Conceptual Model

- Find the concepts
- Draw them in a conceptual model
- Add associations
- Add attributes



Drawing of Concepts



Customer panel

Deposit item receiver

Deposit item

Receipt printer

Receipt basis

Can

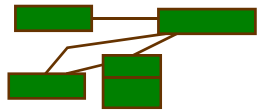
Bottle

Crate

Adding Associations

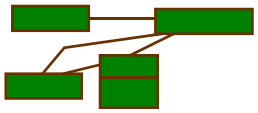
- If one concept needs to know of a another concept for some duration they should be linked by an association.

- Also use the Common Association list in order to identify associations.



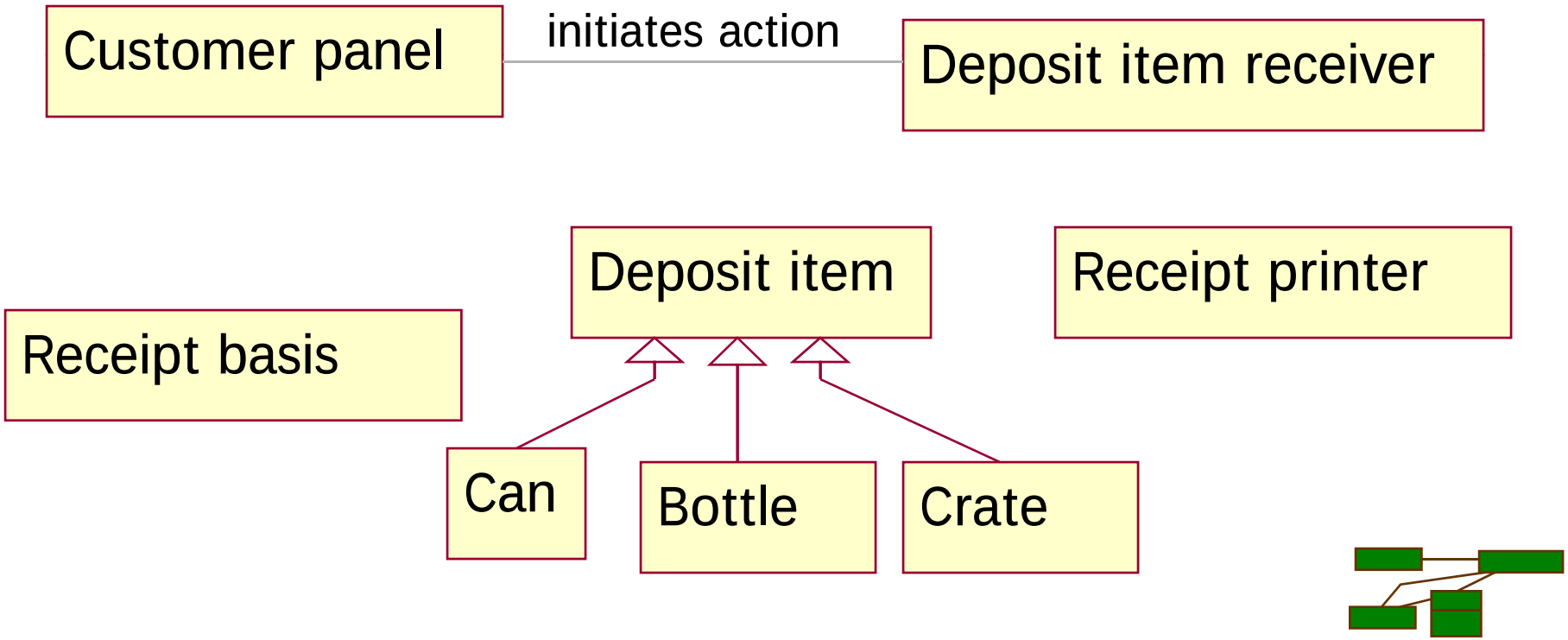
Common Association List

- A is a part of B
- A is contained in B
- A is a description for B
- A is a line item of a transaction or report B
- A is known or reported in B
- A is a member of B
- A is a organisational subunit of B
- A uses or manages B
- A communicates with B
- A is related to a transaction B
- A is a transaction related to another transaction B
- A is next to B
- A is owned by B

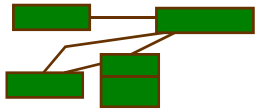


Adding Associations

- The Customer panel communicates to the receiver when an item is inserted.
- Also when the receipt is requested.

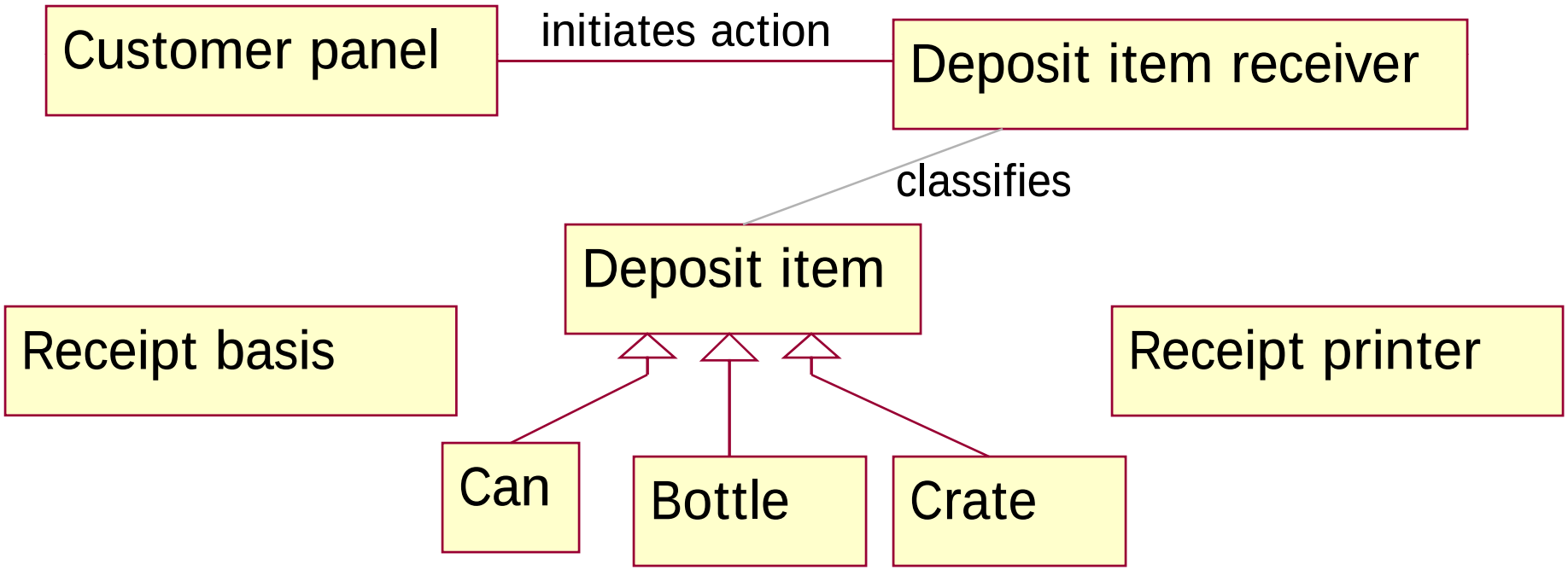


Adding Associations..

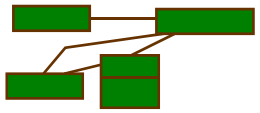


The Deposit item receiver manages Deposit items:

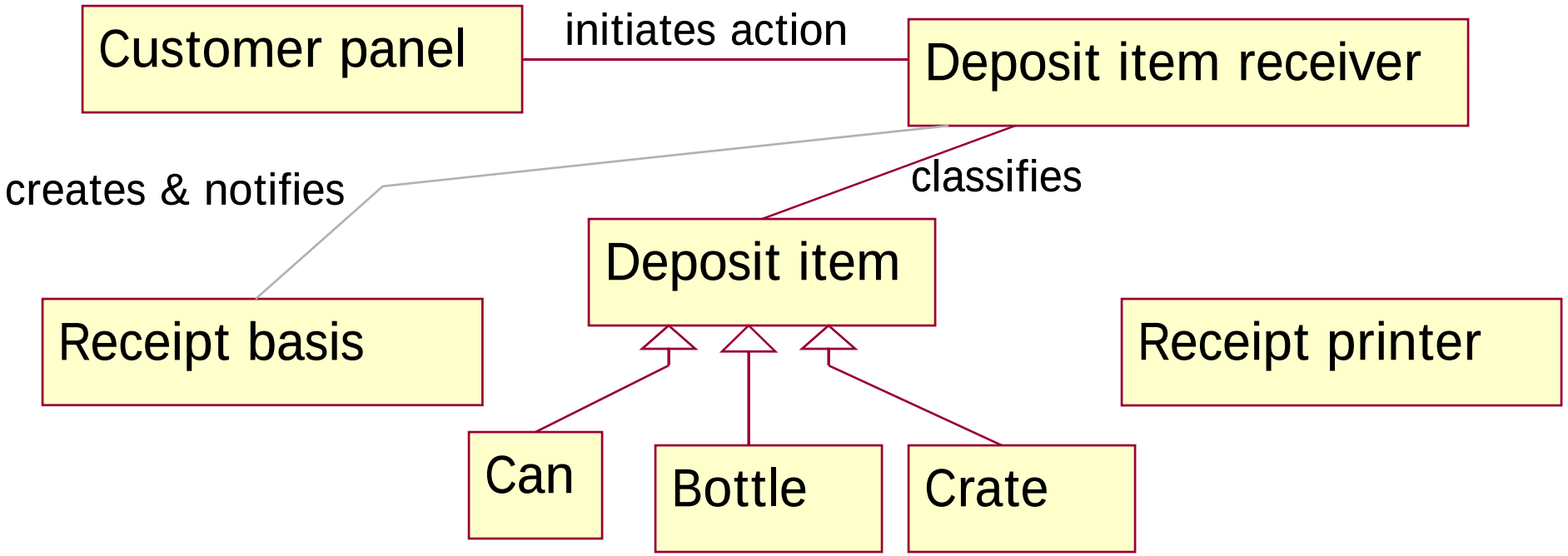
The items are classified.



Adding Associations..

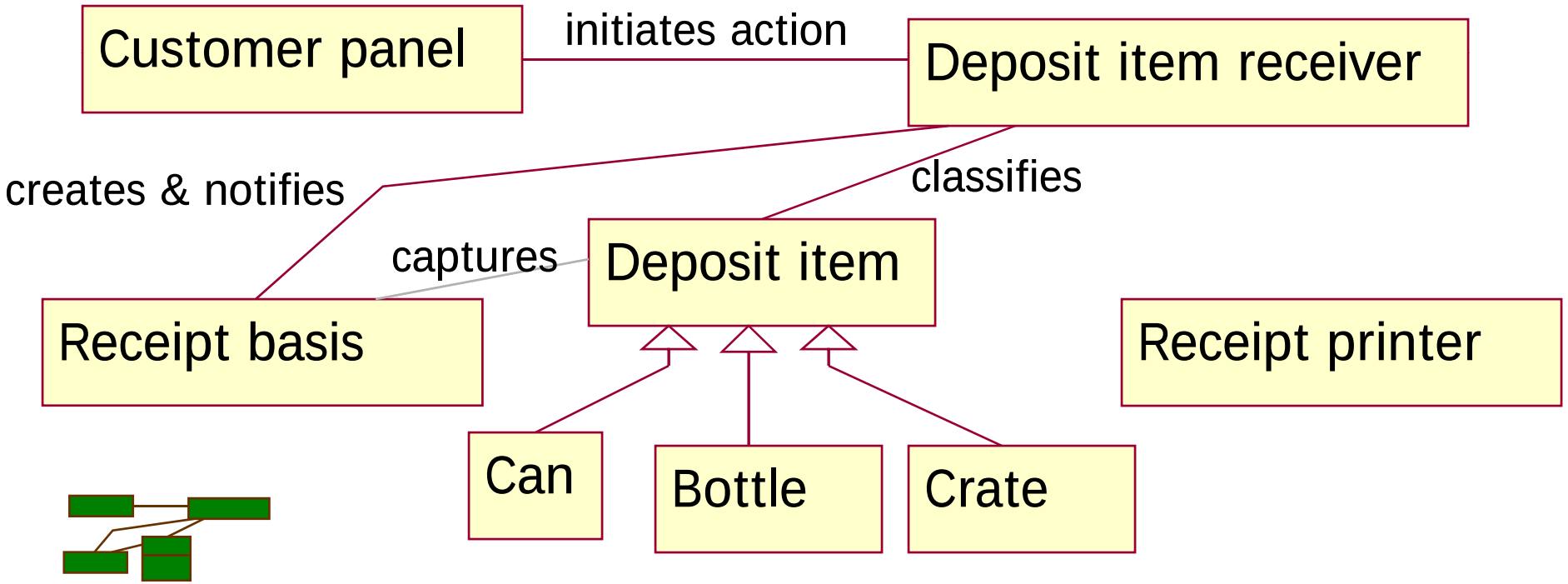


- The Deposit item receiver communicates to Receipt basis:
- Items received and classified are stored.
- It also creates the receipt basis when it is needed for the first time.



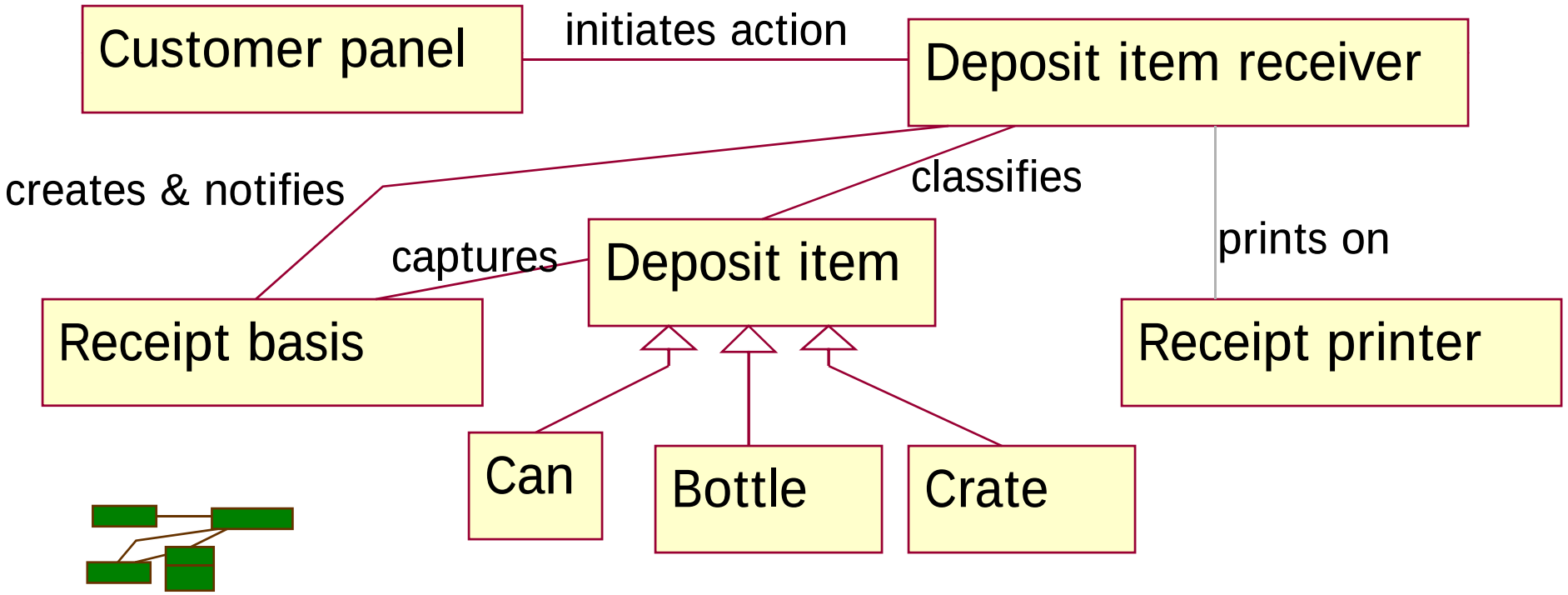
Adding Associations..

The Receipt basis collects Deposit items.



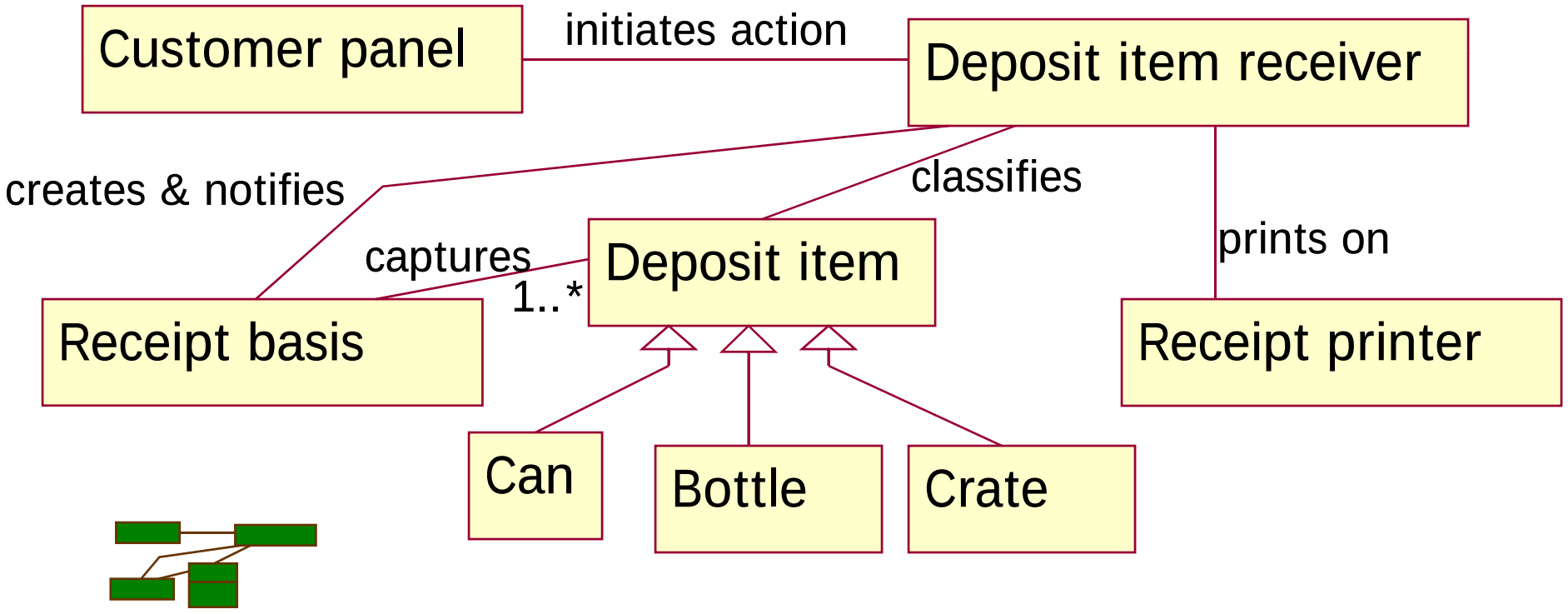
Adding Associations..

- On request by the Customer Panel the Deposit item receiver initiates printing of a receipt on the printer.



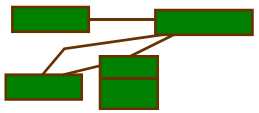
Adding Associations..

- Adding multiplicities
- Only one association here is a 1 to many relationship
- All others are 1 to 1.



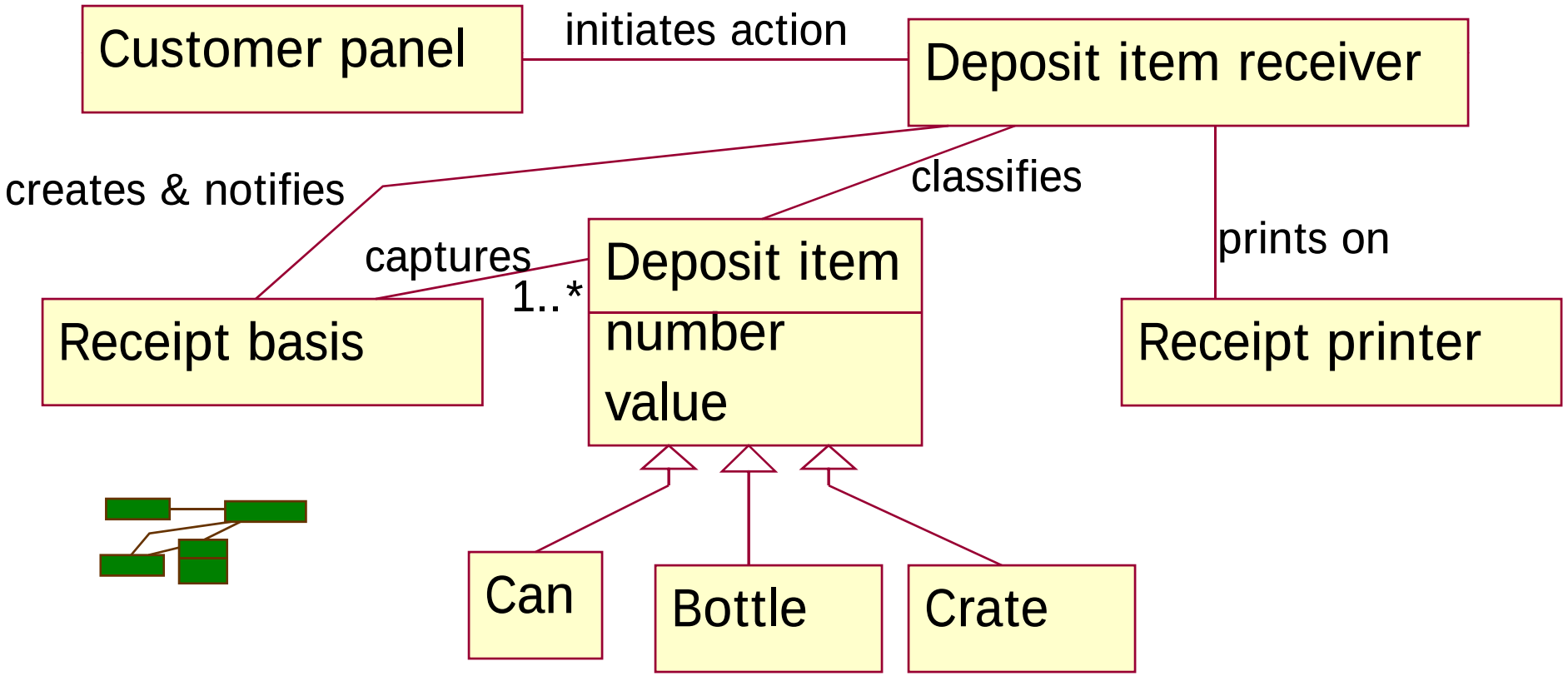
Adding Attributes

- An attribute is a logical data value of an object.
- Attributes in a conceptual model are **simple data values** as
  - Boolean, Date, Number, String (Text), Time, Address, Colour, Price, Phone Numbers, Product Codes, etc.
- Sometimes it is difficult to distinguish between attributed and concepts
  - E.g. Concept “Car” vs. attribute “Reg. Number”.



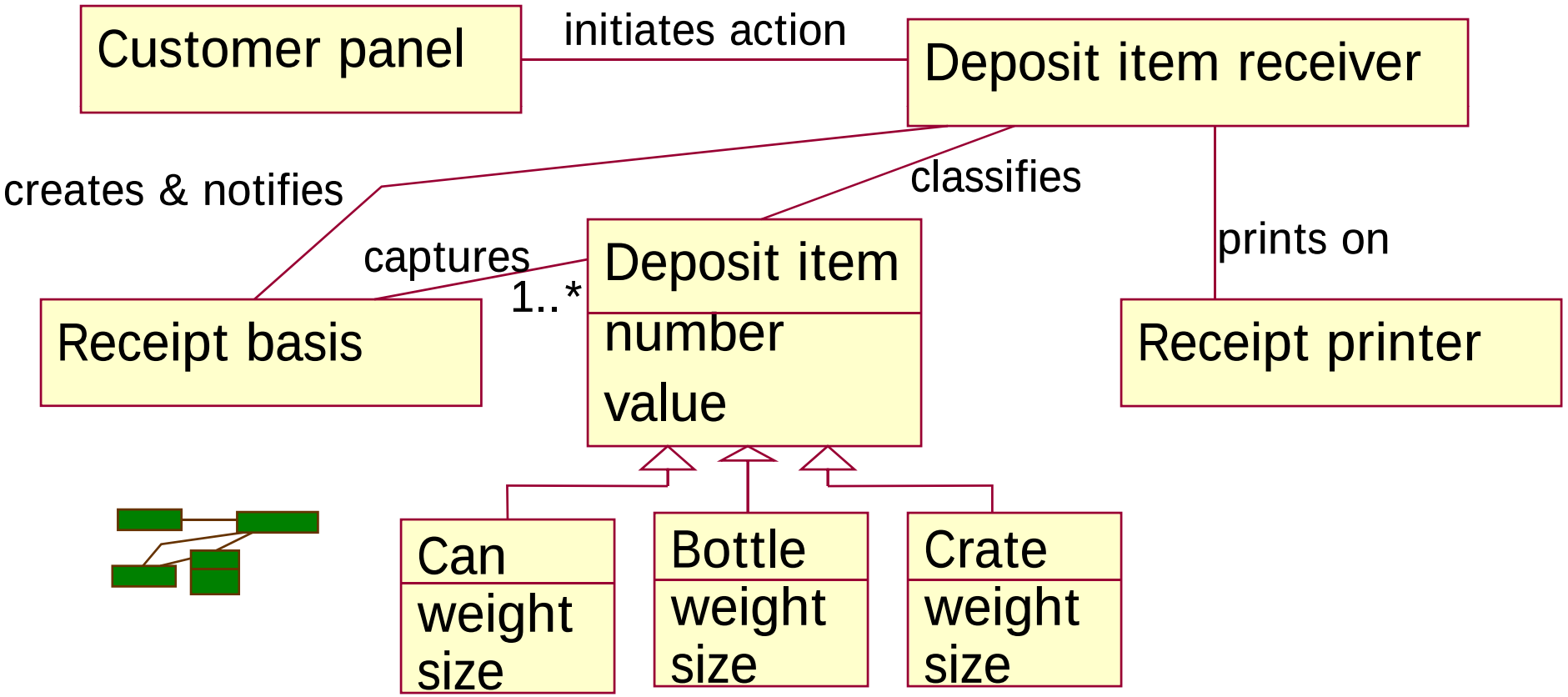
Adding Attributes..

- The Deposit item has a value.
- Also it will be assigned a number that shows later on the receipt.



Adding Attributes..

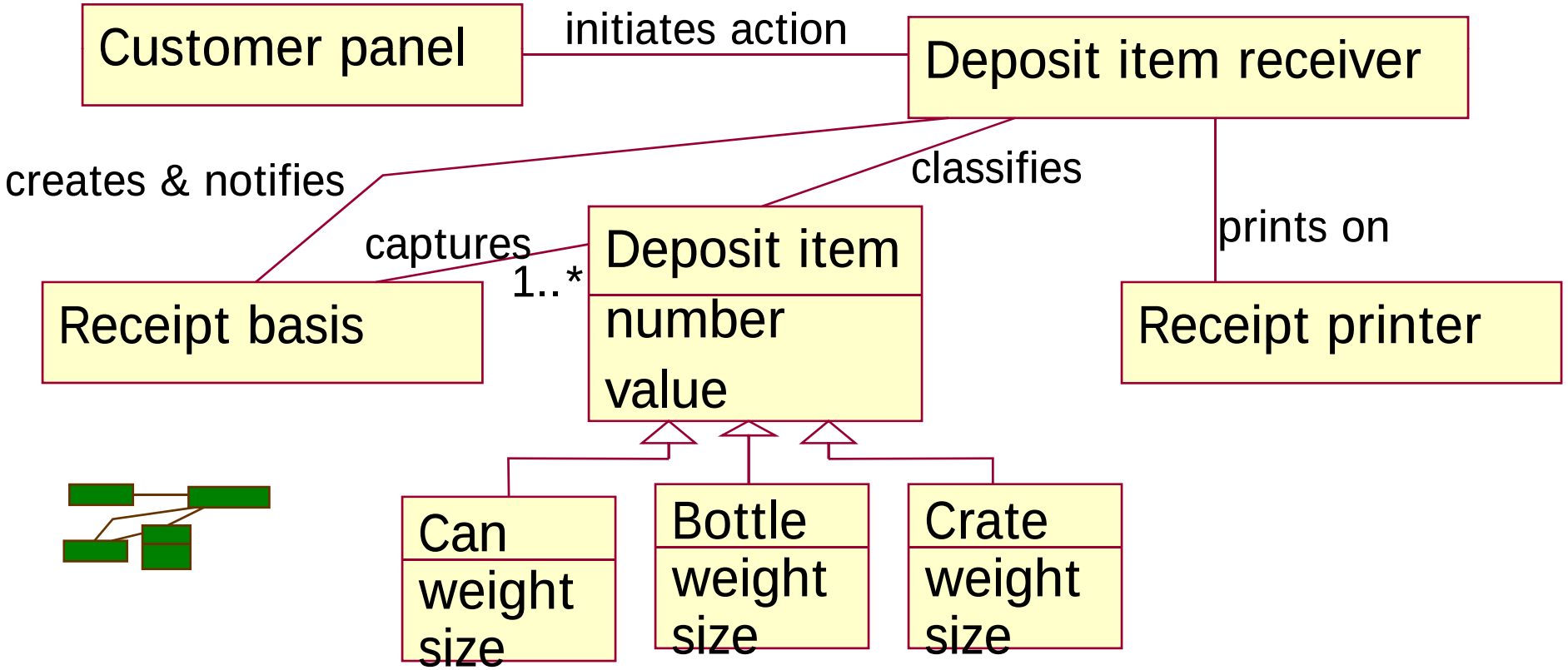
- In order to be classified by the Deposit item receiver each item has also a weight and a size.
- However this is the same for each type of item, but different *between* the types.



Summary

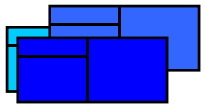
NOT in a conceptual model:

- arrows
- dotted lines
- methods, operations, functions



CRC Cards & Role Playing

- Not part of the UML design process but useful in detecting responsibilities of objects are CRC cards (developed by Kent Beck and Ward Cunningham).
- **CRC** stands for Class-Responsibility-Collaborator. They look like:

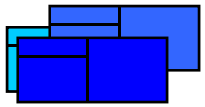


Name	Responsibilities
Collaborators	

CRC Cards & Role Playing..

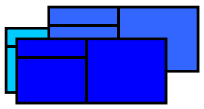
- CRC cards are index cards, one for each class, upon which the responsibilities and collaborators of a class are written.
- They are developed in a small group session where people *role play* being the various classes.
- Each person holds onto the CRC cards for the classes that they are playing the role of.

http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/crc_b/



Example: Recycling machine CRC cards

Customer panel	receive items receive print request
Deposit item receiver	

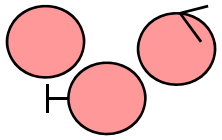


Dep. item receiver	classify items create Receipt Basis print receipt
Printer Deposit Item Receipt Basis	

Etc.

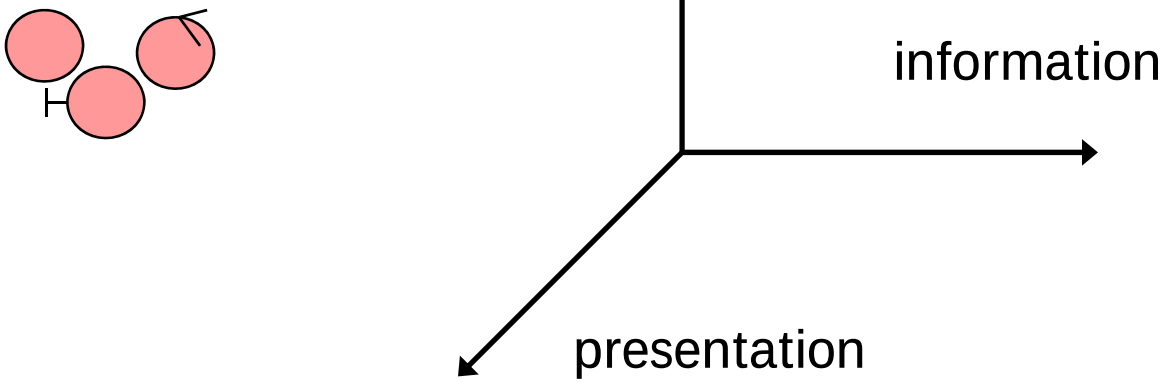
The Object Oriented Analysis Model (Jacobson)

- An analysis model is used to represent the system specification.
- To achieve robustness and stability the model must be **implementation environment independent.**
- Any change in the implementation environment will not affect the logical structure of the system.
- The model must be able to capture **information, behaviour** (operations) and **presentation** (inputs and outputs).



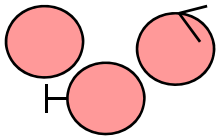
Behaviour - Information - Presentation

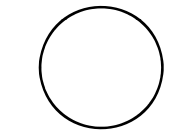
- **The model is defined in information - behaviour - presentation space.**



Syntax of the Object Oriented Analysis Model

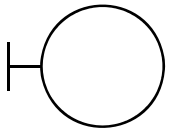
- Within an use case, we employ three types of objects
- in Rational Rose, known as three types of entities or stereotypes





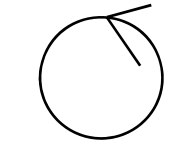
Entity

On information – behaviour plane and incline to information axis



Boundary / Interface

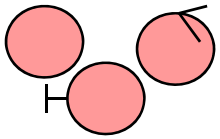
On the presentation axis

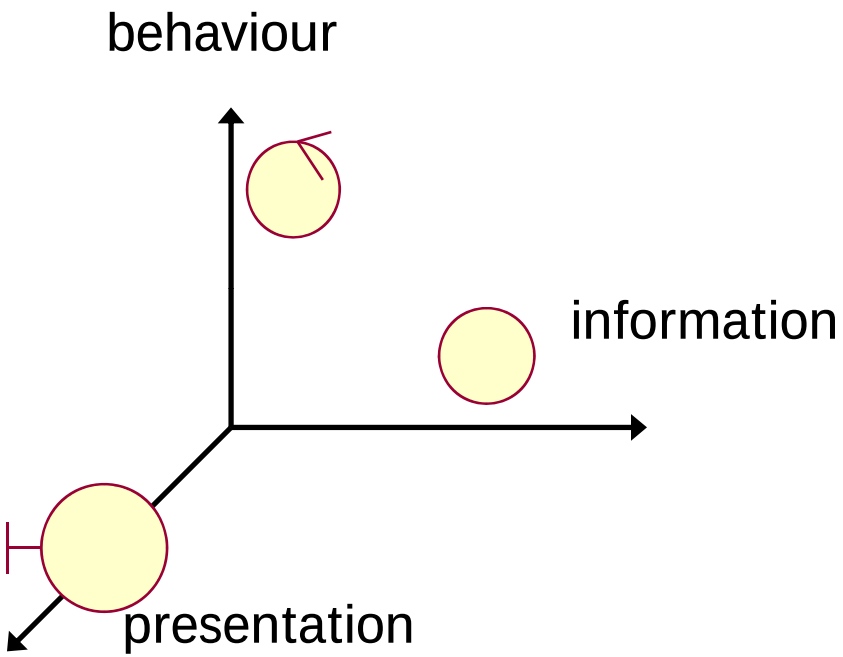


Control

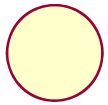
On information – behaviour plane but incline towards behaviour axis

Entity, Control, Interface

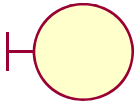




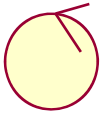
Semantics of the OO Analysis Model



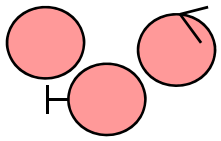
- An **entity object** models information that shows the state of a system.
  - This information is often used to record the effects of operations
  - related to the behaviours of the system.



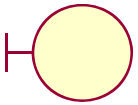
- A **boundary/interface object** models inputs and outputs and operations that process them.



- A **control object** models functionality/operations regarding to validate and decide whether to process and pass information from the interface object to the entity object or the way around.

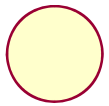


Pragmatics of the Object Oriented Analysis Model



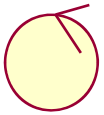
Identifying interface objects

- functions directly related to actors.



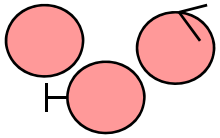
Identifying entity objects

- information used in an use case and functions of processing the information.



Identifying control objects

- functions that link interface objects and entity objects



Example: The Recycling Machine

Identifying interface objects

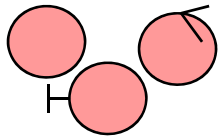
- Printer, Customer Panel

Identifying entity objects

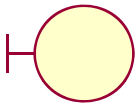
- Long term information: Crate, Bottle, Can
- Superclass: Deposit item
- Short term information: Receipt basis

Identifying control objects

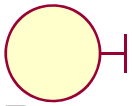
- Deposit item receiver



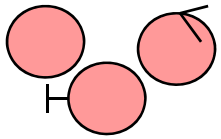
The Recycling machine: Interface Objects



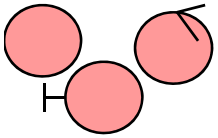
Customer panel

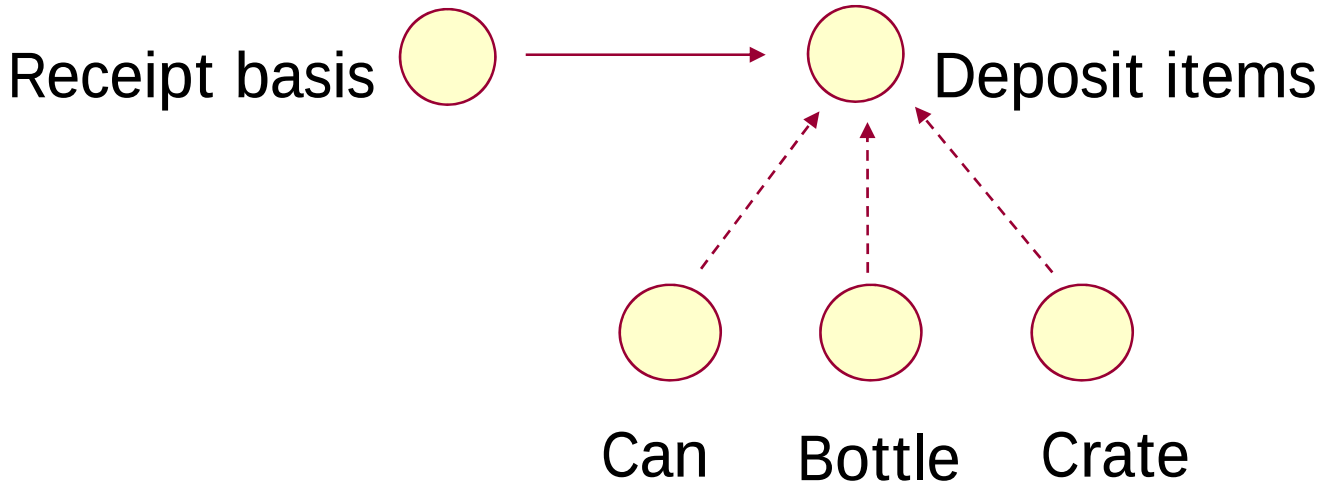


Receipt printer

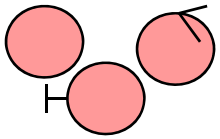


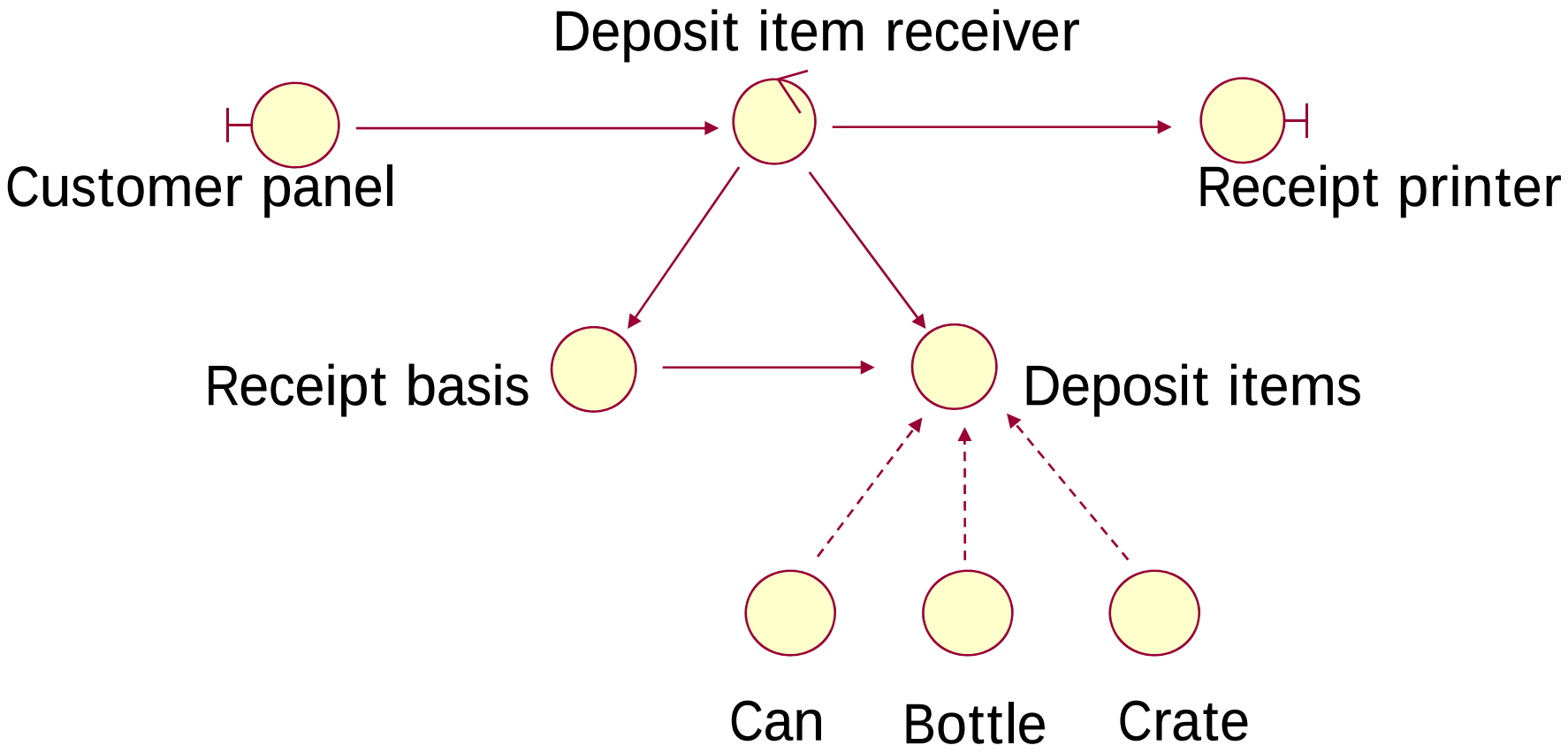
The Recycling machine: Entity Objects





Link Interface and Entity Objects by a Control Object





Subsystem

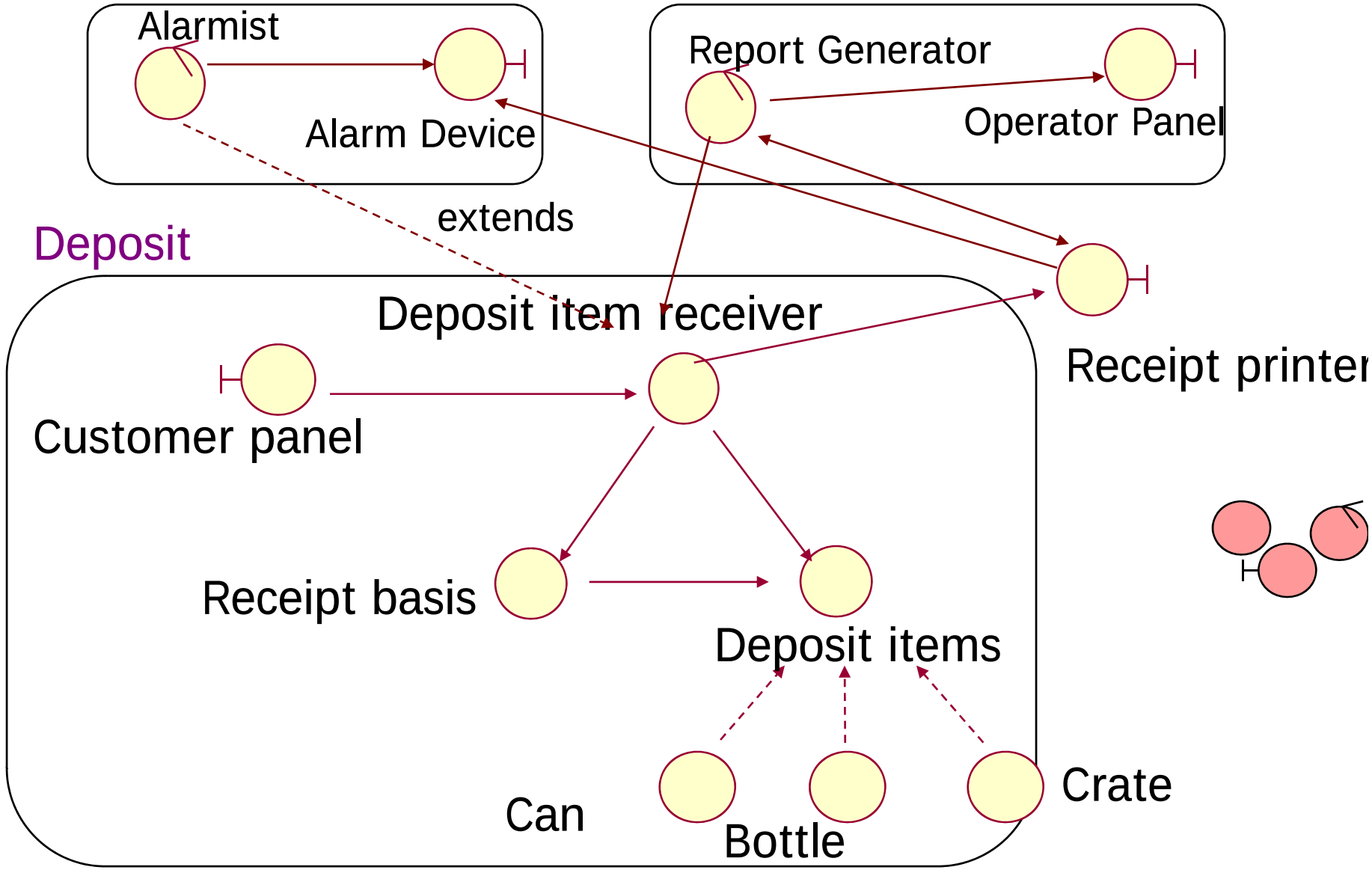
- Package the objects so that complexity is reduced
- Lowest level is service package
- Having objects with strong mutual coupling
- Little communication between different subsystem as possible

Subsystem cont..

- Whether two objects are strongly functionally
  - Will changes in one object lead to changes in the other object?
  - Do they communicate with the same actor?
  - Are both of them dependent on a third object, such as an interface object or an entity object?
  - Does one object perform several operations on the other?

- The aim is to have strong functional coupling within the subsystem and a weak coupling between subsytem

Subsystem in Recycling machine



Good Analysis class

- Name reflects its intent
- Crisp abstraction that models one specific element of the problem domain
- Maps to a clearly identifiable feature of the problem domain
- Has a small, well-defined set of responsibilities
- It has high cohesion
- Has low coupling

Bad Analysis class

- T is a functoid
- It is an omnipotent class
- Has a deep inheritance tree
- Low cohesion
- High coupling

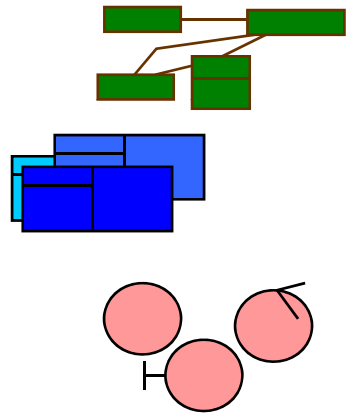
Summary - Object Oriented Analysis

The main task is identifying the objects.

Also: Relationships between objects.

Three strategies:

- Conceptual Model (concepts as objects)
- CRC cards (index cards as objects)
- Analysis Model (Stereotypes as objects)



Next step: Design.

What we Learnt..

- ✓ The Analysis Phase
- ✓ Analysis Model
- ✓ Analysis Artifacts
- ✓ Meta Model of Analysis Model
- ✓ Analysis workflow details
- ✓ Analysis model-rules of thumb
- ✓ Object Oriented Analysis
- ✓ Three ways to do Object Oriented Analysis
- ✓ Conceptual Model – Overview
- ✓ The Concept Category List
- ✓ Finding Concepts with Noun Phrase Identification

Learning Objectives Cont..

- ✓ Exercise
- ✓ How to make a conceptual model
- ✓ Drawing of Concepts
- ✓ Adding Associations
- ✓ Adding Attributes
- ✓ CRC cards & Role playing
- ✓ The Object Oriented Analysis Model (Jacobson)
- ✓ Subsystem
- ✓ Good Analysis class
- ✓ Bad Analysis class

UNIT II Learning's

- ✓ **Architecture**
- ✓ Introduction
- ✓ System development is model building
- ✓ model architecture
- ✓ requirements model
- ✓ analysis model
- ✓ the design model
- ✓ the implementation model
- ✓ test model
- ✓ **Analysis**
- ✓ Introduction
- ✓ the requirements model
- ✓ the analysis model

Objective Questions

Q1. Define Architecture.

Q2. Justify the statement System Development is a Model Building.

Q3. At what time, you will decide to start System Development.

Q4. In what way specifications can be used?

Q5. Define Conceptual modeling.

Q6. Define block design.

Q7. Define requirement model.

Q8. Define analysis model.

Q9. Define design model

Q10. Define Implementation Model.

Q11. Define Test Model.

Short Questions

Q1. Suggest some heuristics for identifying objects during object oriented analysis of problem.

Q2. Differentiate between analysis objects with examples.

Q3. Consider air ticket reservation system. Identify entity, control and interface objects.

Q4. Write short note on Architecture.

Q5. Differentiate Method and Process

Q6. What are the five different models for system development, as per the Jacobson approach?

Q7. How models are tightly coupled to the architecture? Discuss.

Long Questions

Q1. What are the features of analysis model and design? Explain with examples.

Q2. For a library management system make analysis model, design model and construction model.

Q3. Justify the statement “System development is model building”.

Q4. ”The goal if analysis model is to develop a model of what the system will do.” Explain the statement with the help of the steps that an analyst will follow throughout the analysis.

Q5. Describe what is done in Analysis with example?

Q6. Describe the system development process with model building.

Research Problems

Q1.Many people invest their money in a number of securities (shares). Generally, an investor has multiple portfolios of investments, each portfolio having investments in many securities. From time to time an investor sells or buys some securities and gets dividends for the securities. There is a current value of each security-many sites give this current value. It is proposed to build a personal investment management system (PIMS) to help investors keep track of their investments as well as on the overall portfolios. The system should also allow an investor to determine the net-worth of the portfolios.

- Discuss the problem analysis for the PIMS problem statement/
- Provide the use case based requirement analysis and specification
- Identify the conceptual objects and draw the Analysis model for PIMS

References

1. Ivar Jacobson, “Object Oriented Software Engineering”, Pearson, 2004.
2. Grady Booch, James Runbaugh, Ivar Jacobson, “The UML User Guide”, Pearson, 2004
3. R. Fairley, “Software Engineering Concepts”, Tata McGraw Hill, 1997.
4. P. Jalote, “An Integrated approach to Software Engineering”, Narosa, 1991.
5. Stephen R. Schach, “Classical & Object Oriented Software Engineering”, IRWIN, 1996.
6. James Peter, W Pedrycz, “Software Engineering”, John Wiley & Sons
7. Sommerville, “Software Engineering”, Addison Wesley, 1999.
8. http://www.gentleware.com/fileadmin/media/archives/userguides/poseidon_users_guide/userguide.html
9. http://www.gentleware.com/fileadmin/media/archives/userguides/poseidon_users_guide/statemachinediagram.html
10. http://www.developer.com/design/article.php/2238131/State-Diagram-in-UML.htm