INTRODUCATION TO SOFTWARE ENGINEERING

08. ANALYSIS AND DESIGN

Nguyen Thanh Hùng

hungnt@soict.hust.edu.vn

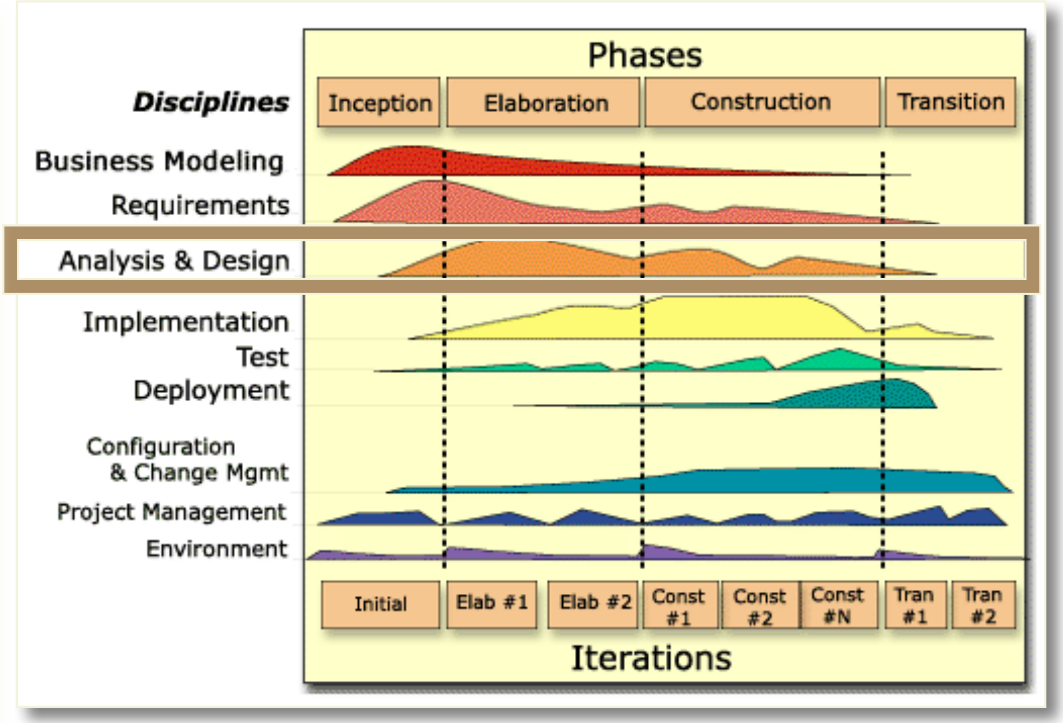
Content

1. What is Analysis & Design?
2. Analysis & Design methods
3. Case study

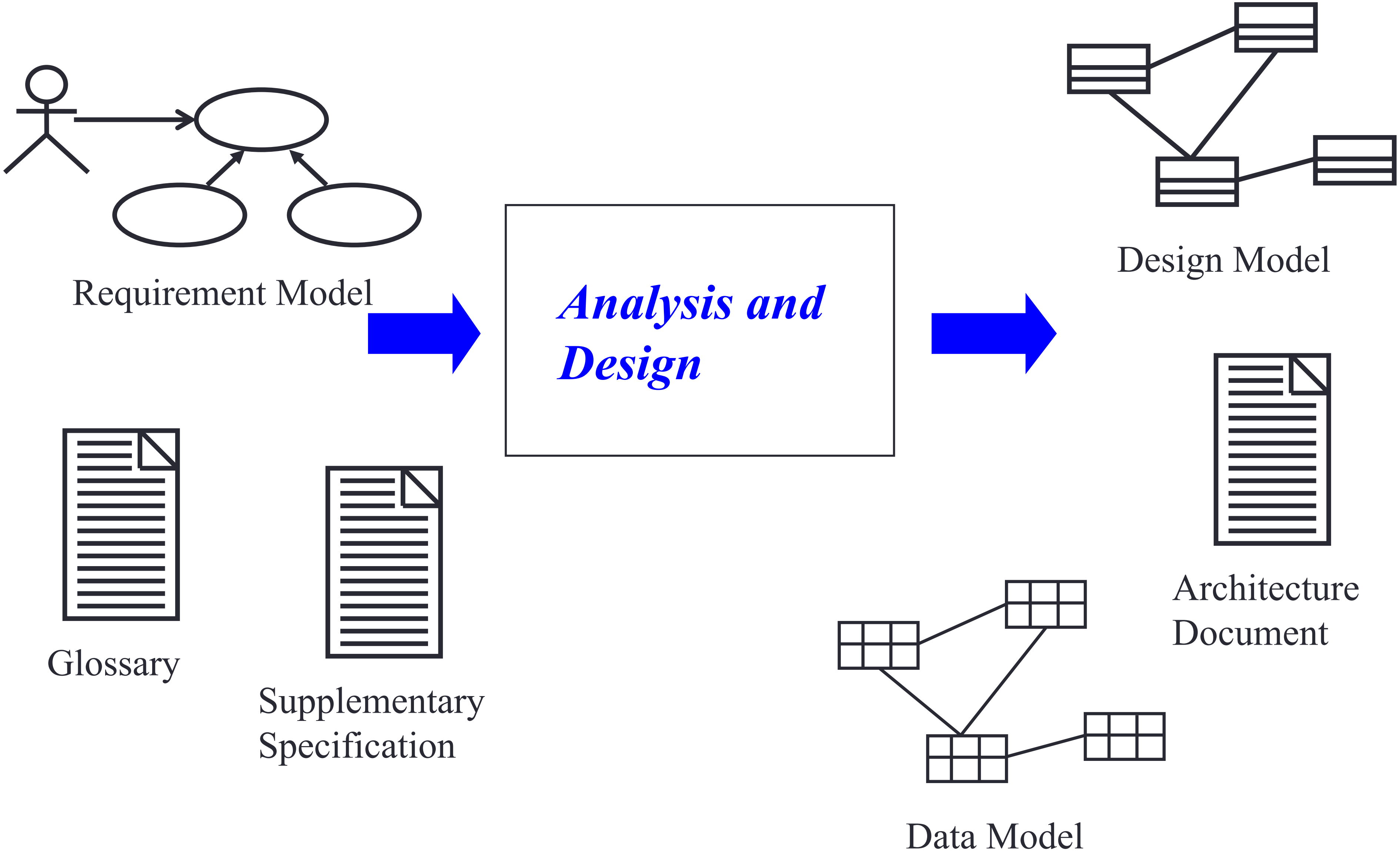
Analysis and Design in Context

The purposes of Analysis and Design are to:

- Transform the requirements into a design of the system-to-be.
- Evolve a robust architecture for the system.
- Adapt the design to match the implementation environment, designing it for performance.



Analysis and Design Overview



Requirement Model

Glossary

Supplementary Specification

Analysis and Design

Design Model

Architecture Document

Data Model

Analysis Versus Design

Analysis	Design
Focus on understanding the problem Idealized design Behavior System structure Functional requirements A small model	Focus on understanding the solution Operations and attributes Performance Close to real code Object lifecycles Nonfunctional requirements A large model

Analysis & Design workflow

- External Design / Architectural (Structure) Design
  - A top-level structure of software and its components
  - A top-level design for the interfaces external to the software and between its components
  - A top-level design for the database
- Internal design / Detailed Design

  (sufficiently detailed design for each software component to permit coding and testing)
  - Software units: can be coded, compiled, and tested

Internal / Detailed Design

- Functional partitioning and structuring
  - Partitioning sub-systems into programming units
  - Design of flows of data and processes among units
- Physical data design (File design)
  - Analysis of data characteristic

  ➔Decision of file organization, the medium and record layout

  ➔Database detail design
- I/O detailed design
  - Screen/Form design, Message design
  - Check method of input

Screen design

- Designing input format, output format
- Main points
  - Considerations of source of data, amount of data, number of items and digits, attributes...
  - Flow of input items from top to bottom and left to right.
  - Standardization of screen layout and operability
  - Consistent message presentations
  - Possibilities of aborting an operation midway or re-starting from the previous screen

Output design

- Output format of the system
  - Screen display
  - Printing (e.g. report)
- Easy of use and read
  - Consideration of the relationship among items set up the sequence and positions
  - Title appropriately expresses the content
  - Distinguishably various dates
  - Appropriate spacing for items and sufficient empty space for the entire report
  - Critical items can be immediately identified

Code design

- Structures of employee code, product code…
  - Properties: commonality, systematization, expandability, and clarity
  - Necessary functions

Identification	To distinguish individual data; e.g. two people with the same first and last names.
Classification	To group data; e.g. classifying by affiliation code (letters or series of numbers)
Listing	To sort data when listing; e.g. sorted by birthday, the same number of characters
Checking	To check input values; e.g. adding a check digit.

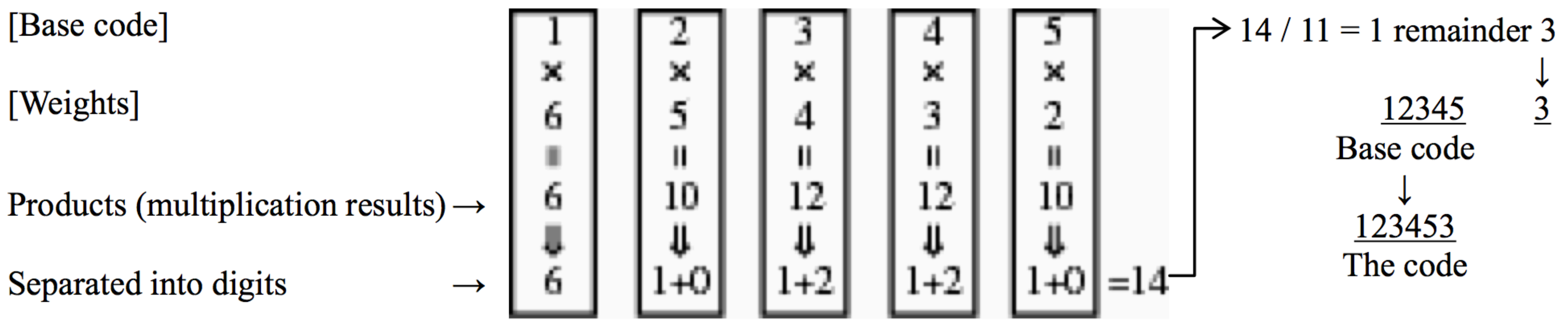
Check Methods

Method	Check content
Numeric check	Really a number?
Format check	In the predefined format? Are the digits not misaligned?
Limit check	Within the upper and lower bounds?
Range check	Within the correct range? (limit check)
Validity check	Logically valid? (e.g. Feb. 29 on a non-leap year)
Sequence check	Key item values listed in sequence?
Balance check	Paired items matched up correctly? (e.g., renters and landlords)
Collation check	Code value contained in the master file?

Check Digit Method

- Determining the error by performing the same calculation for code creation and code checking
  - Normally modulus 11 (or 10) & one digit at the end the base code

XXXXX	Y	→	XXXXXY
Base code	Check digit		Code



[Base code]	1	2	3	4	5	
	×	×	×	×	×	
[Weights]	6	5	4	3	2	
	=	=	=	=	=	
Products (multiplication results) →	6	10	12	12	10	
	⇓	⇓	⇓	⇓	⇓	
Separated into digits →	6	1+0	1+2	1+2	1+0	=14

→ 14 / 11 = 1 remainder 3

↓

12345 3

Base code

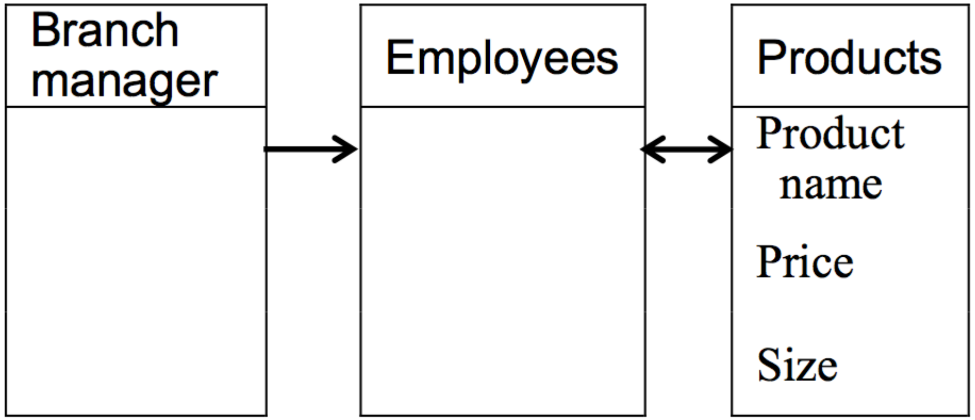
↓

123453

The code

Logical data design

- Identifying and analysis of data relationships by examining the business
- Deciding on data structure and items
- Preparation of candidates for databases & files (File Database concept design)
    - Entity-Relationship (ER) diagram



Content

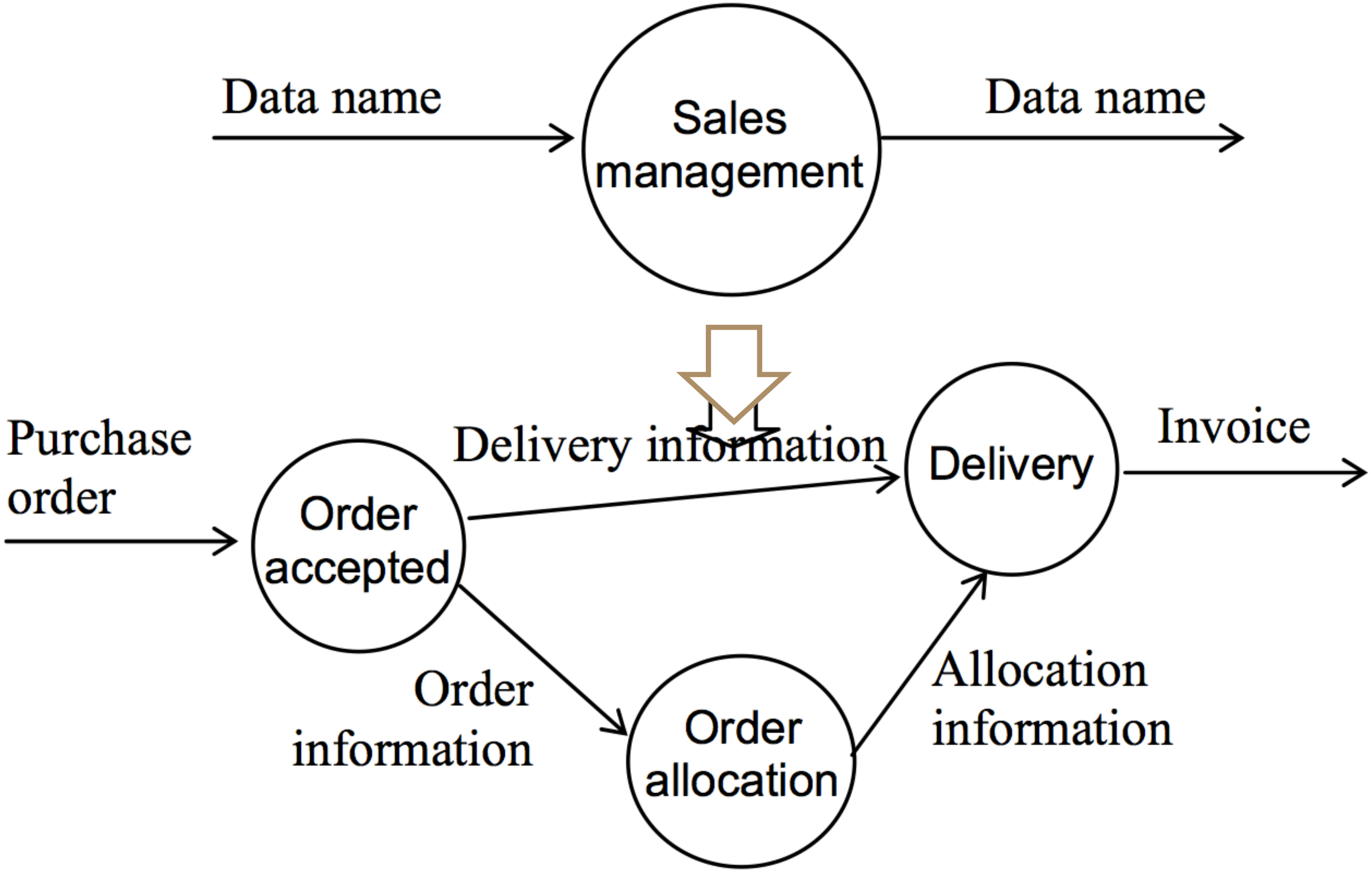
1. What is Analysis & Design?
2. Analysis & Design methods
3. Case study

2. Analysis & Design methods

- Structured Design (Process-oriented design)
  - Processes (DFD) => Program modules
  - Division of modules with a top-down approach
    - E.g. Bubble chart; STS partitioning; TR Partitioning
- Data-oriented Design (Based on data structure)
  - Processes integrated in data-specific procedures
  - e.g. Jackson method, Warnier method
- Object-oriented Design
  - Design of Object and Message sequence
    - i.e. Interaction diagrams
  - Design of Class and Relationship, i.e. Class diagrams

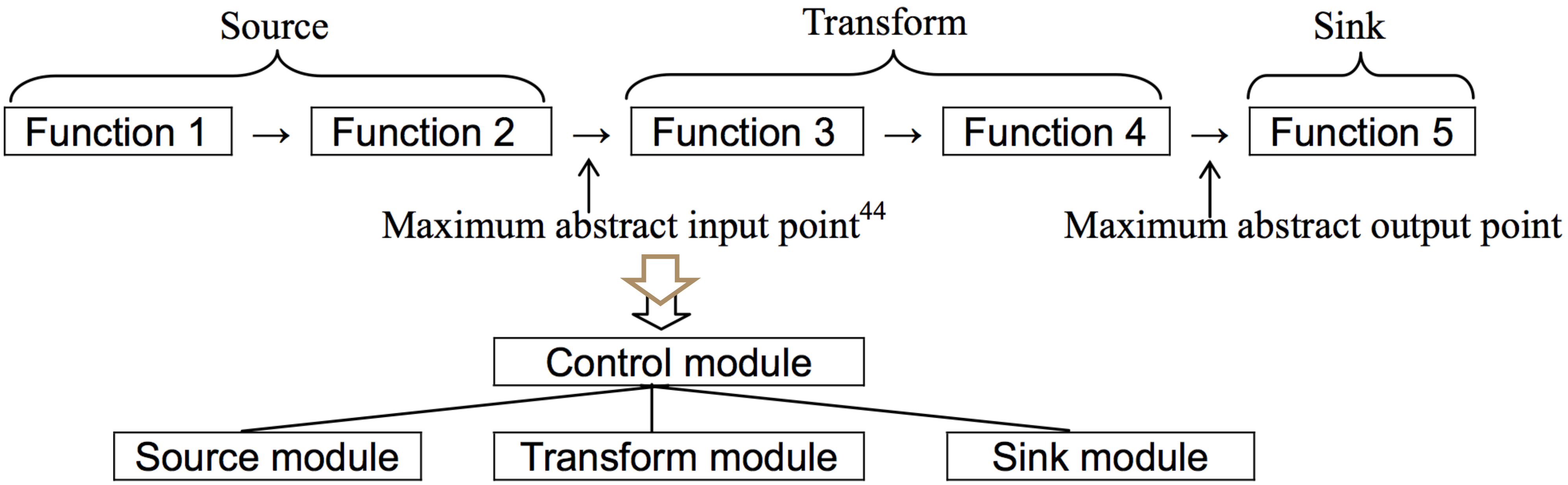
2.1. Structured Design – Bubble chart

- Sales management system broken down



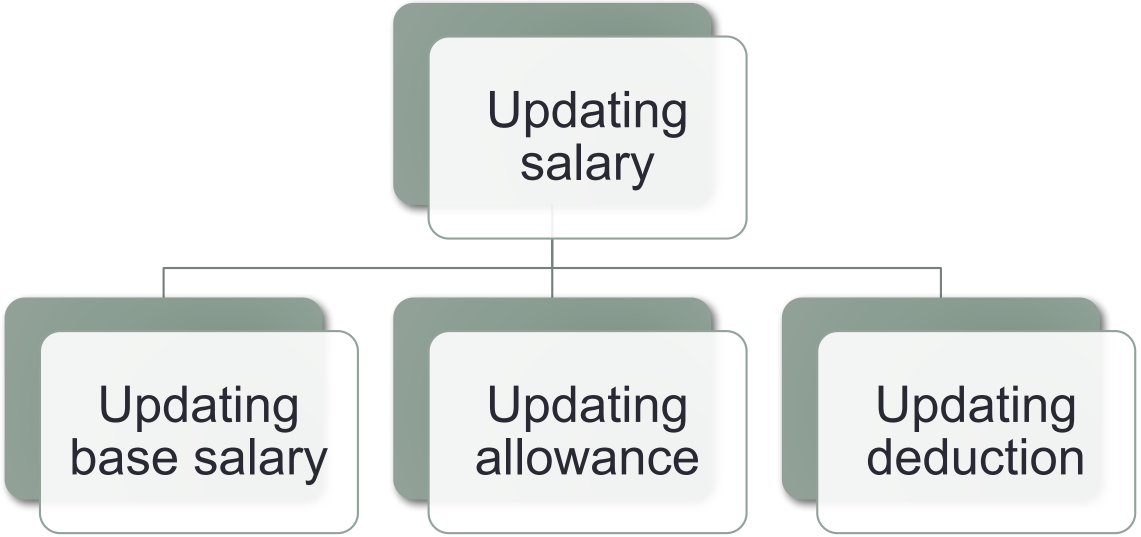
2.1. Structured Design – STS partitioning (source, transform, sink)

- Devision of program structure into
  - Source (input)
  - Transform (processes)
  - Sink (output)



2.1. Structured Design – TR Partitioning (Transaction Partitioning)

- Partition of transactions by branching flow of data => Modules
- Frequently in online systems
- E.g. Payroll program Seat reservation system

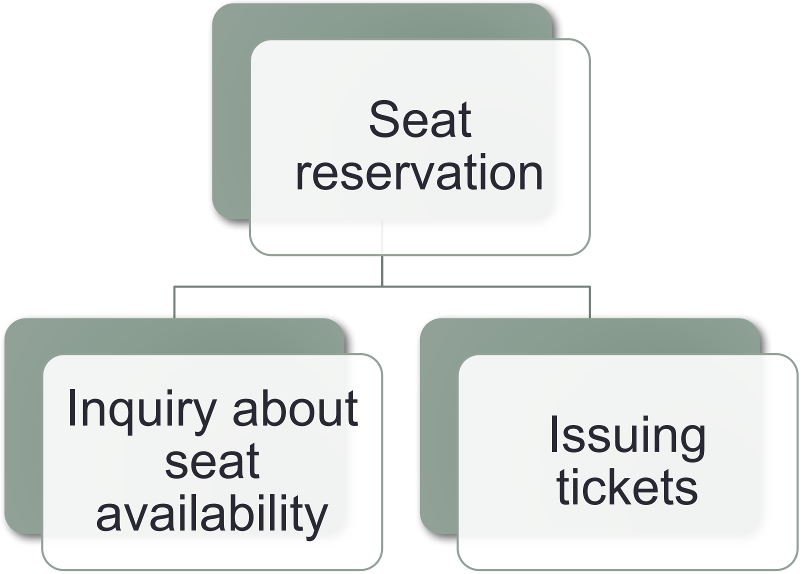


Updating salary

Updating base salary

Updating allowance

Updating deduction



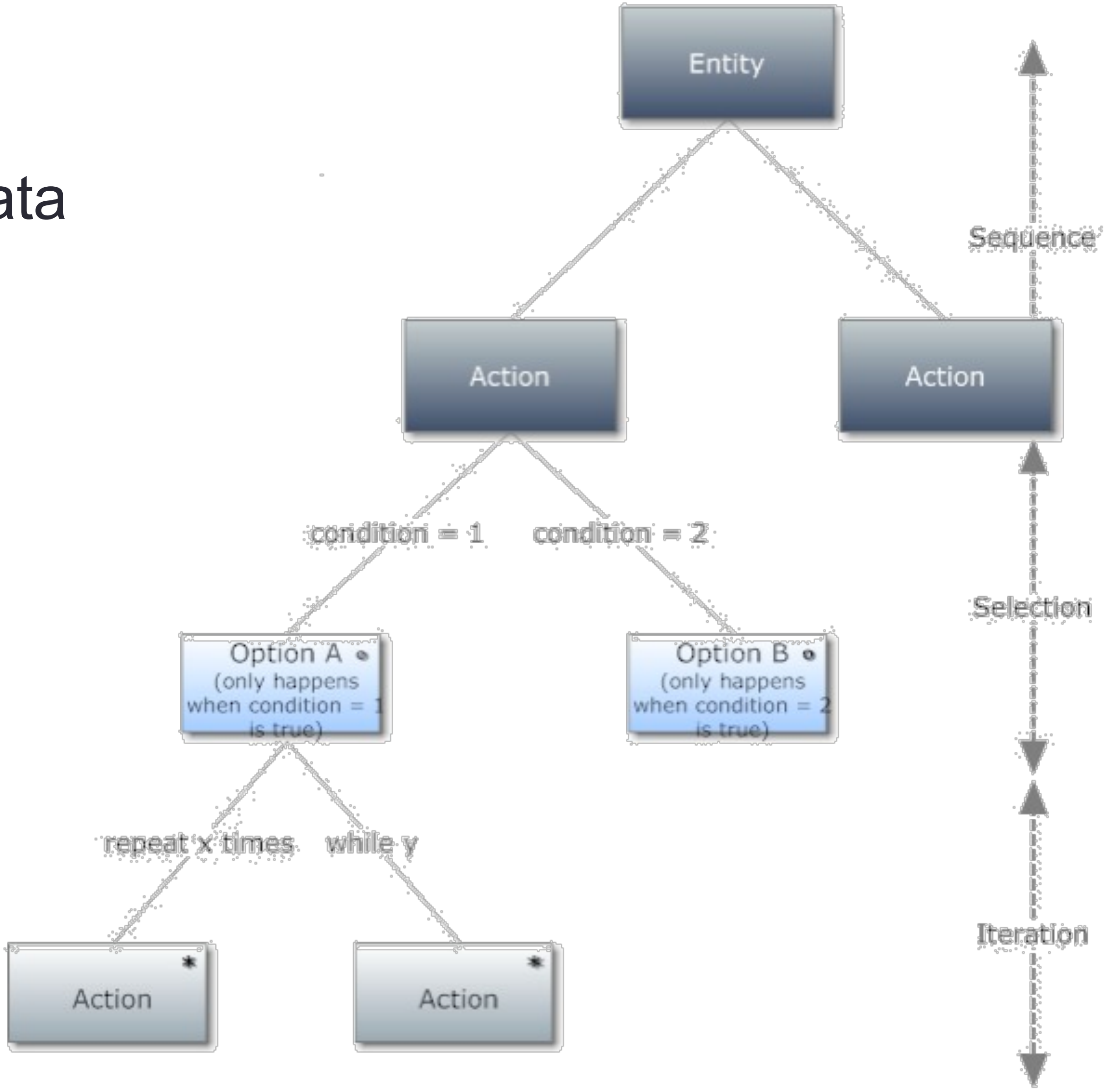
Seat reservation

Inquiry about seat availability

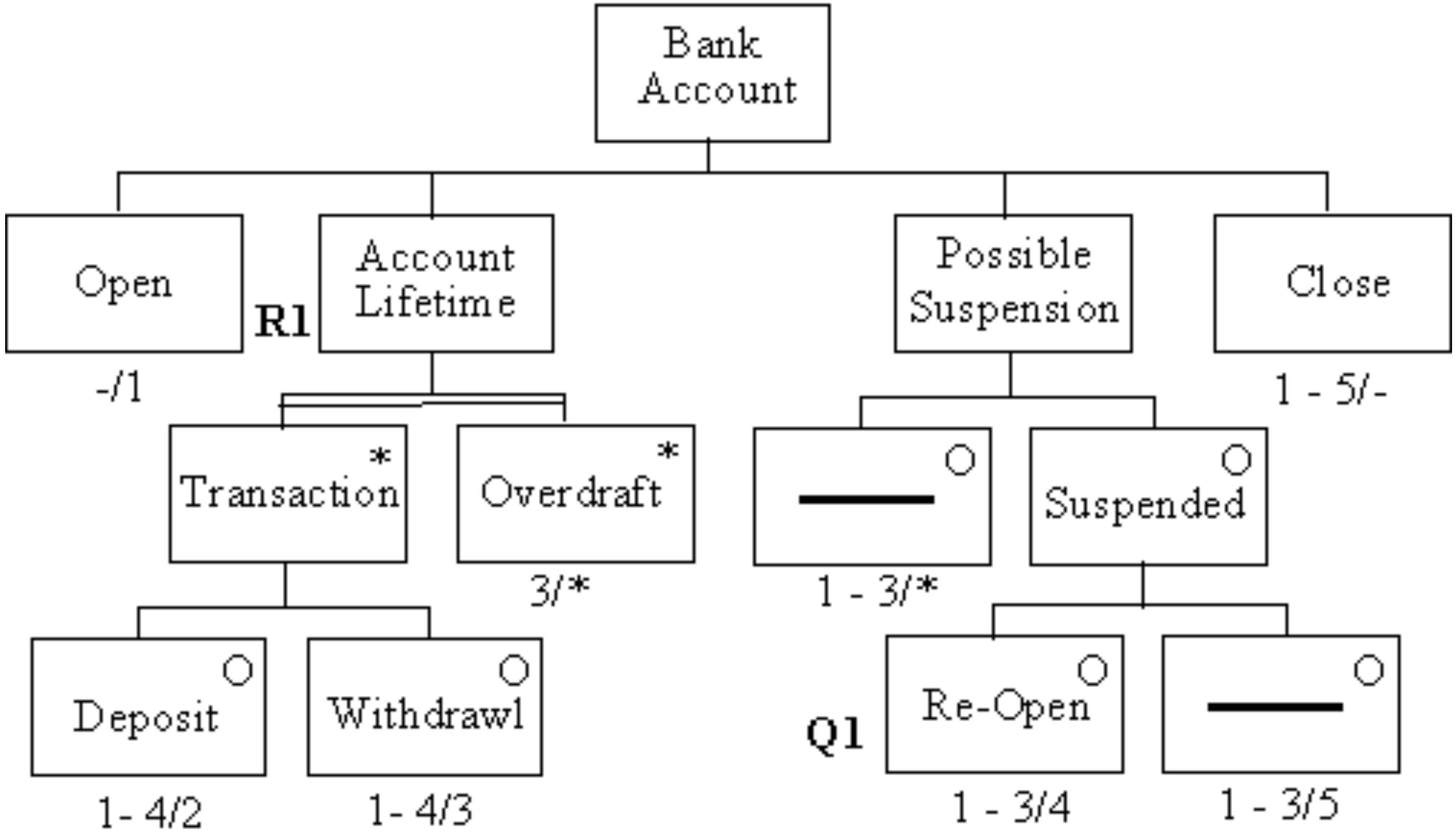
Issuing tickets

2.2. Data-oriented Design – Jackson method

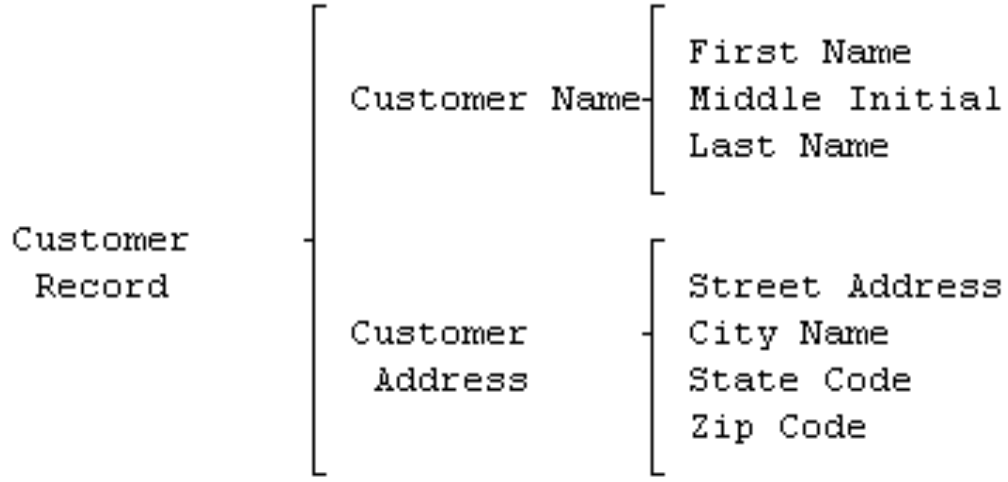
- Jackson Structured Programming (JSP)
- Idea of input/output data structure naturally determines the processes
- Data structure and program structure are expressed as tree structures

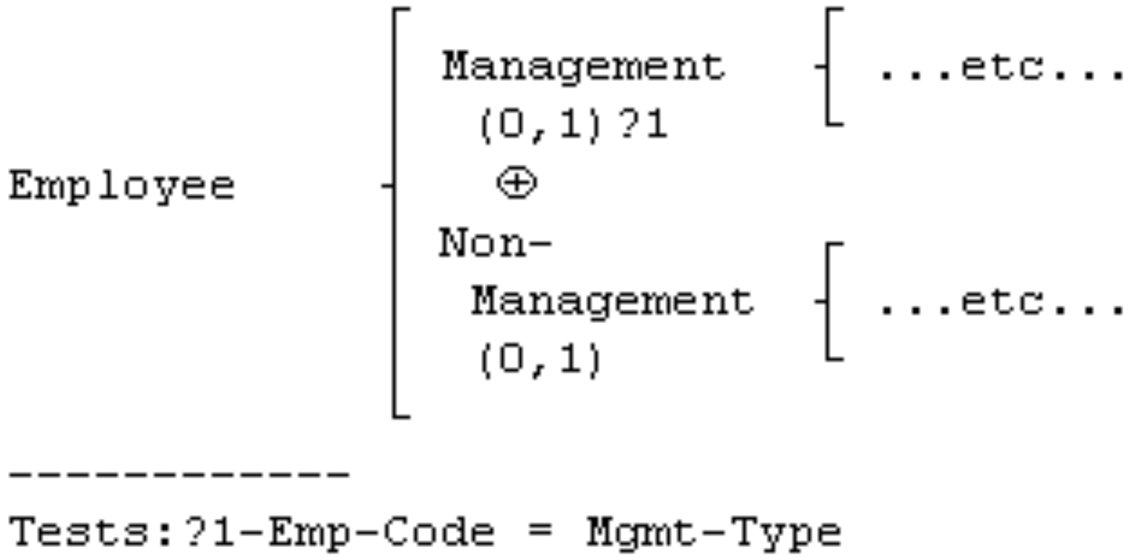


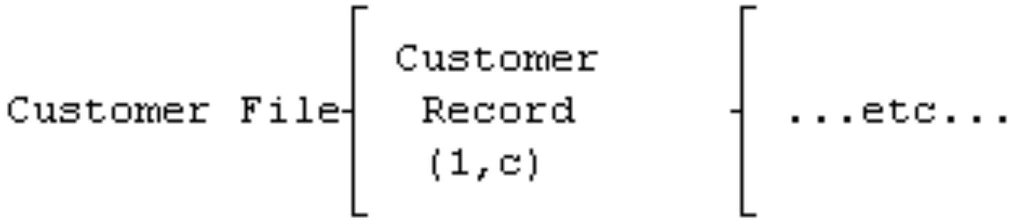
2.2. Data-oriented Design – Jackson method



2.2. Data-oriented Design – Warnier method







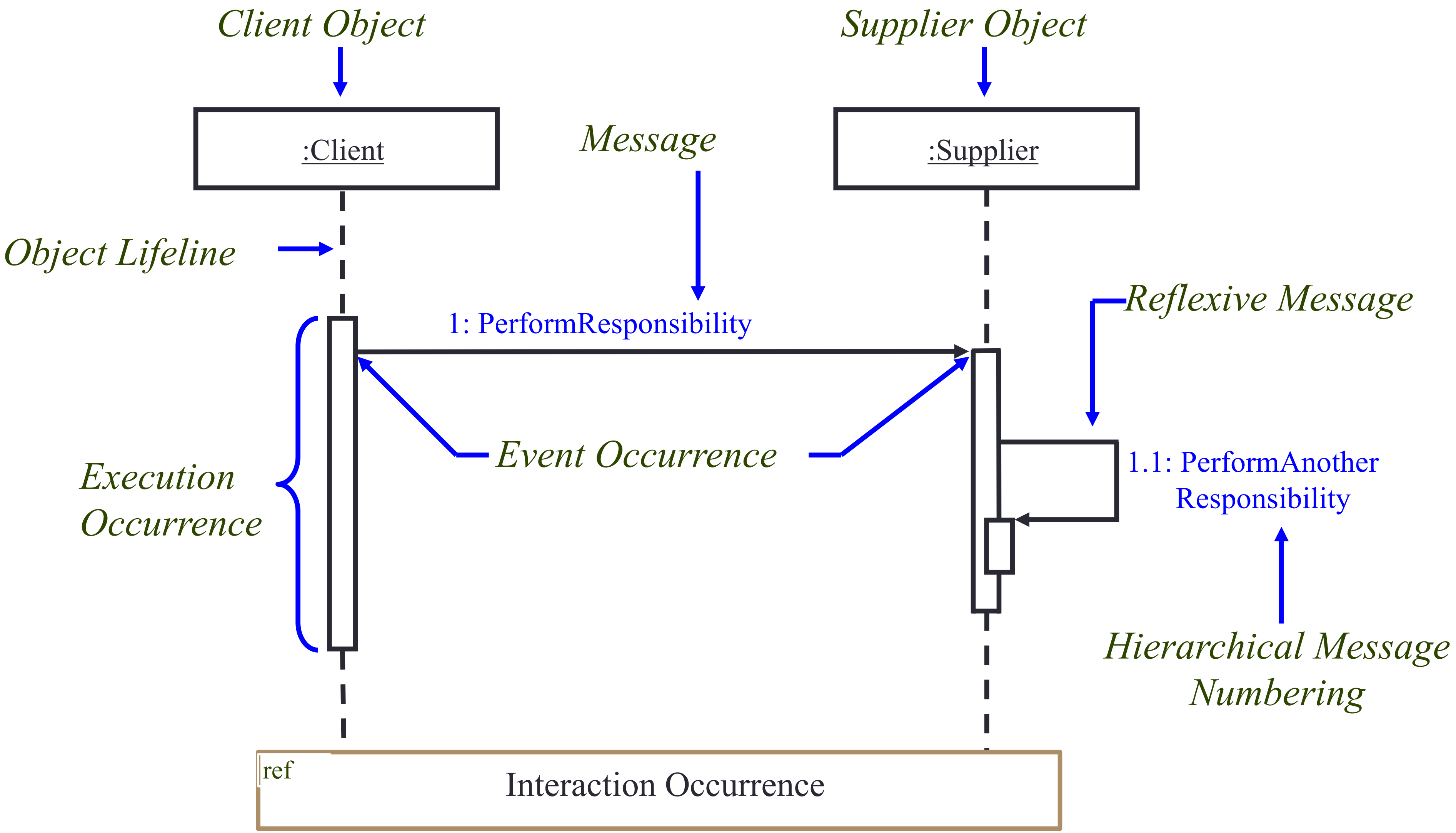
- Drawing so-called Warnier diagram, similar to a flowchart

2.3. Object-oriented design

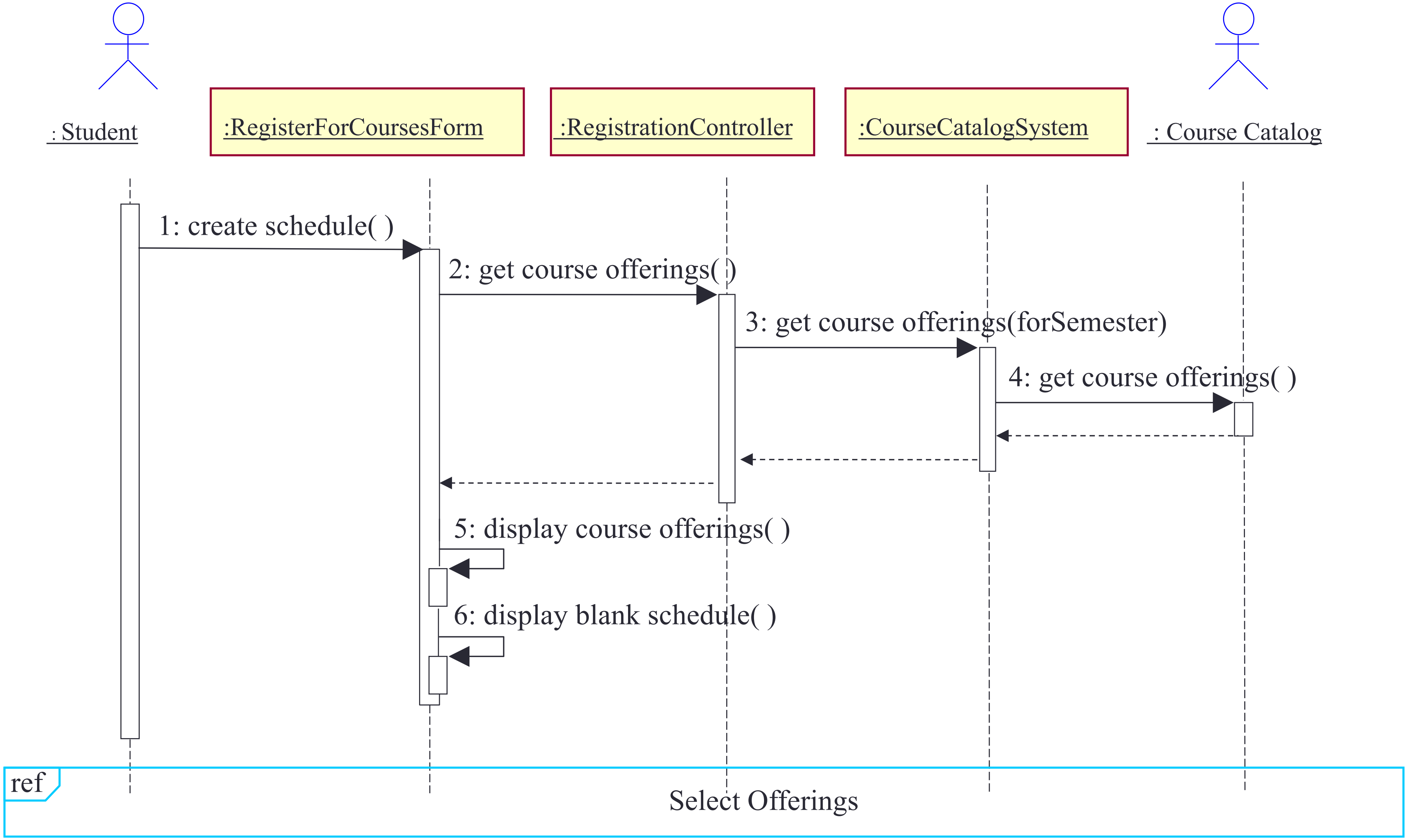
- Program = Objects + Messages
- Represented by UML diagrams
  - Interaction diagram, Class diagram…
- Object-oriented principles
  - Abstraction, Encapsulation, Modularity, Hierarchy
- Main concepts
  - Association, Inheritance, Polymorphism

(Object-Oriented Programming course)

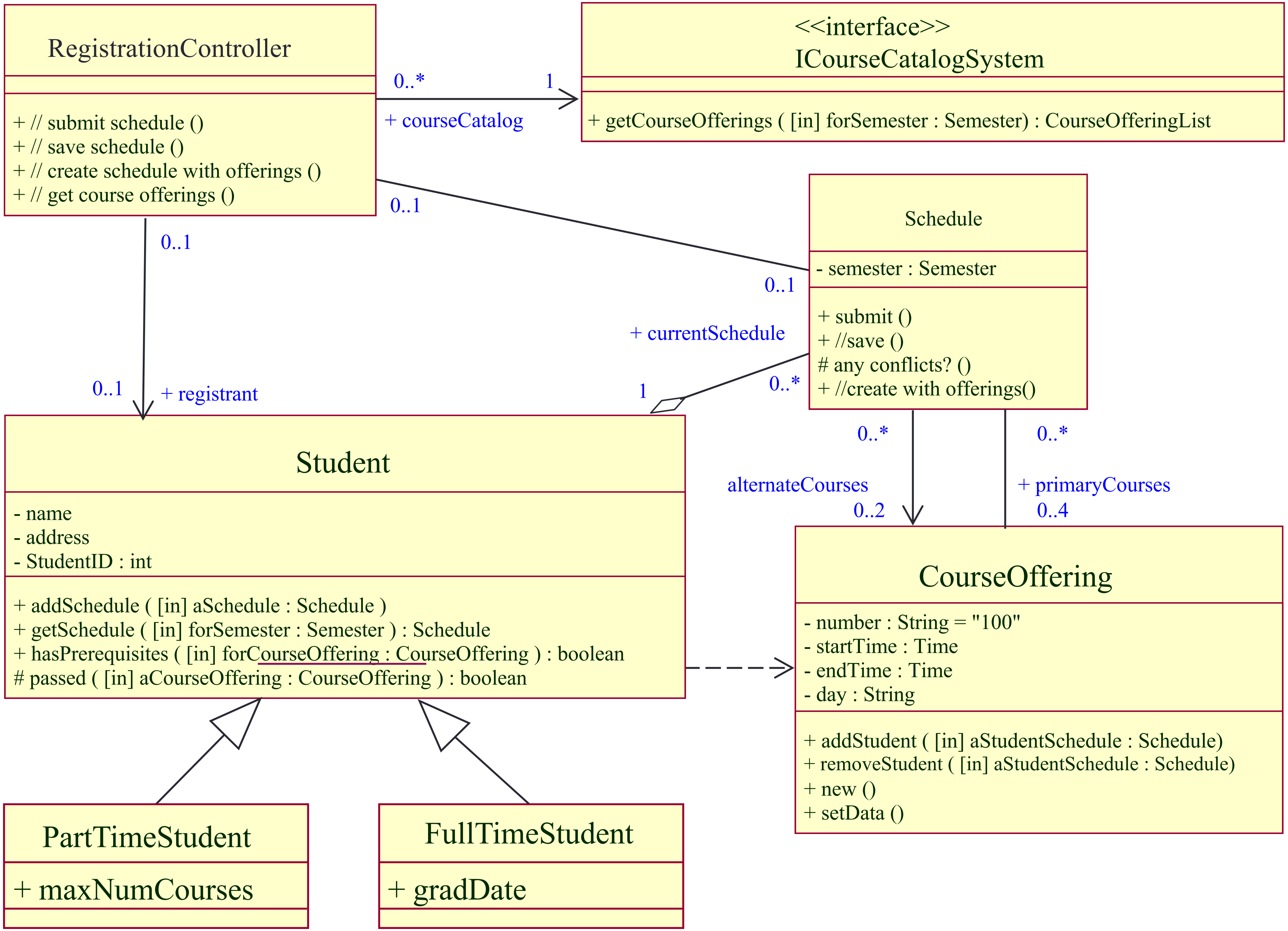
The Anatomy of Sequence Diagrams



E.g. Sequence diagram for "Create a Schedule" sub-flow


: Student
:RegisterForCoursesForm
:RegistrationController
:CourseCatalogSystem
: Course Catalog
1: create schedule( )
2: get course offerings( )
3: get course offerings(forSemester)
4: get course offerings( )
5: display course offerings( )
6: display blank schedule( )
ref
Select Offerings

E.g. Class diagram for UC "Register for course"



RegistrationController

+ // submit schedule ()
+ // save schedule ()
+ // create schedule with offerings ()
+ // get course offerings ()

<<interface>>
ICourseCatalogSystem

+ getCourseOfferings ([in] forSemester : Semester) : CourseOfferingList

Schedule

- semester : Semester

+ submit ()
+ //save ()
# any conflicts? ()
+ //create with offerings()

Student

- name
- address
- StudentID : int

+ addSchedule ([in] aSchedule : Schedule)
+ getSchedule ([in] forSemester : Semester) : Schedule
+ hasPrerequisites ([in] forCourseOffering : CourseOffering) : boolean
# passed ([in] aCourseOffering : CourseOffering) : boolean

CourseOffering

- number : String = "100"
- startTime : Time
- endTime : Time
- day : String

+ addStudent ([in] aStudentSchedule : Schedule)
+ removeStudent ([in] aStudentSchedule : Schedule)
+ new ()
+ setData ()

PartTimeStudent

+ maxNumCourses

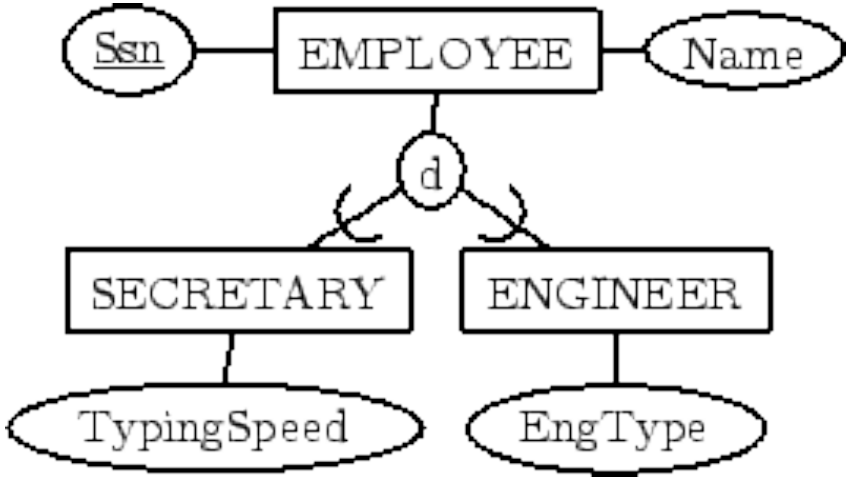
FullTimeStudent

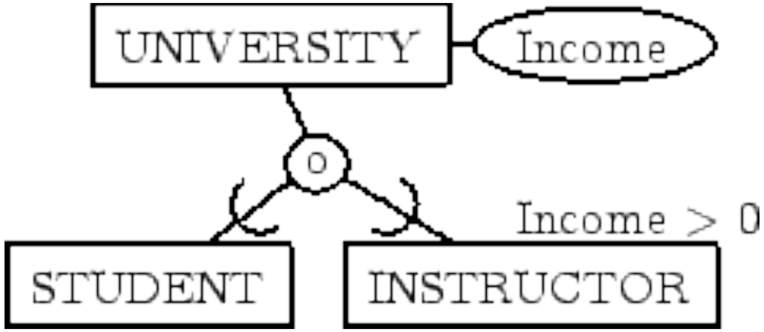
+ gradDate

Enhanced E-R diagram

- Or Extended E-R diagram

= E-R diagram + Superclass/Subclass (Generalization/Specialization)





Content

1. What is Analysis & Design?
2. Analysis & Design methods
3. Case study

Hệ thống quản lý bán hàng siêu thị

- Thảo luận

Question?