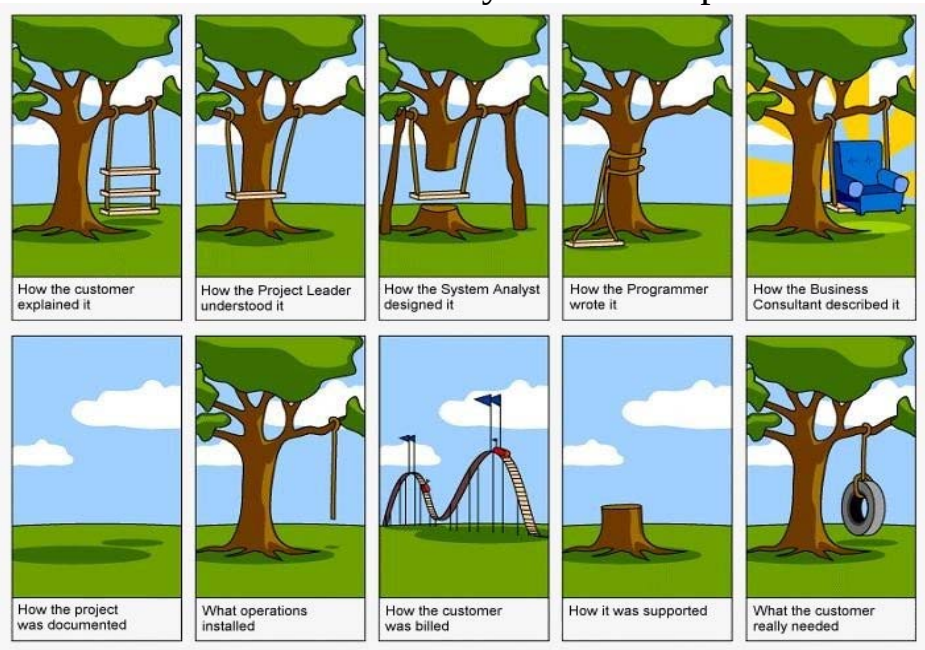
MIT-633 Object-Oriented System Analysis and Design

Assoc.Prof. Wichian Chutimaskul, Ph.D.
Email: wichian@sit.kmutt.ac.th

Facts of information system development



Introduction to Object-Oriented System Analysis and Design

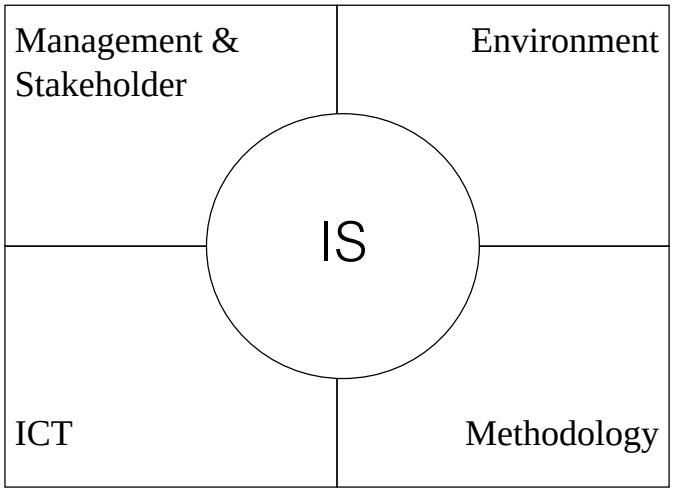
Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, 3rd Edition, A. Dennis - B.H. Wixon - D. Tegarden, Wiley 2010: Chapter 1

Contents

1. Introduction

- SA is a key person in SDLC who analyzes the business situation, defines opportunities, and designs IS for improvement.
- Many failed ISs were abandoned because SA tried to build wonderful systems without understanding the organization
- No silver bullet to guarantee the success of IS development → learn fundamental concepts & best practices

4+1 Framework of Information System Development



Stakeholder

- Actor
- Boundary
- Source & Sink (Destination)
- Terminator
- External Entity

- Who can be the actors?
  - Person
  - Unit or Organization
  - Other Information Systems (ISs)

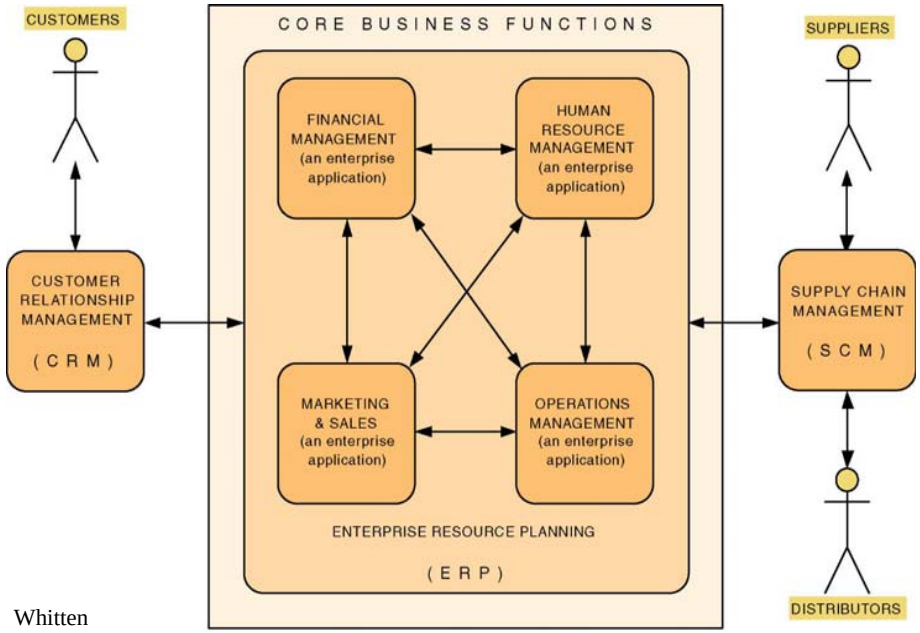
ICTs for Today's IS

- Networks and Internet
  - XML, Script language, Web-based, Intranet, Extranet, Portal
- Mobile and Wireless Technologies
  - PDA, Smart phone, etc.
- Object Technologies
- Collaborative Technologies
  - E-mail/ Instant messaging/ Groupware/ Work flow
- Enterprise Applications

Environment for Today's IS

a) Globalization of the economy
b) e-Commerce / e-Business
c) Security & Privacy
d) Collaboration and Partnership
e) Knowledge asset management
f) Continuous improvement & Total Quality Management (TQM)
g) Business Process Redesign (BPR)

Enterprise Applications



What is a Good System ?

1 Usability
- Users can learn it fast and get their job done easily

2 Efficiency ปสภ.
- It does not waste resources such as CPU time and memory

3 Reliability
- It does what it is required to do without failing

4 Maintainability
- It can be easily changed

2. Systems Development Life Cycle (SDLC)

- A system is a set of components that function together in a meaningful way.

- Characteristics of System:
  - High Cohesion (Abstraction)
  - Low Coupling (Encapsulation)

System Development Life Cycle (SDLC)

1 Planning
- Why should IS be built?

2 Analysis
- Who will use IS?
- What IS will do?,
- When/Where IS will be used?

3 Design
- How will IS work?

4 Implementation
- Which is IS actually built? (buy or build)

a) Planning

- Project Initiation
  - Identify business value
    - System Request: PIECES
  - Feasibility study
    - Operation, Technical, Economic, Schedule
- Project Management
  - Develop work plan
  - Staff the project
  - Control and direct project
  - Assessment

b) Analysis

- Scope Definition
- Problem Analysis
- Requirement Analysis

} Fact Finding
Information / Data Gathering
Requirement Elicitation

- Logical Model (Design)
  - Use-Case Modeling
  - Structural Modeling: Class & Object Modeling
  - Behavioral Modeling
- Decision Analysis

⇨ Requirement Specification

c) Design

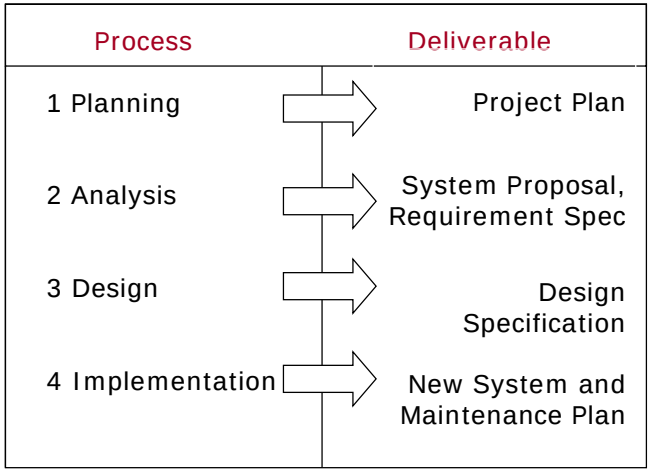
- Architecture Design
  - hardware, software, network infrastructure
- Database Design
- Process (Software) Design
- User Interface Design
  - Input/output design

⇨ Design Specification (Blueprint)

d) Implementation

- Construction & Testing
- Installation
  - Conversion
  - Training
- Operation & Support

Process and Deliverable



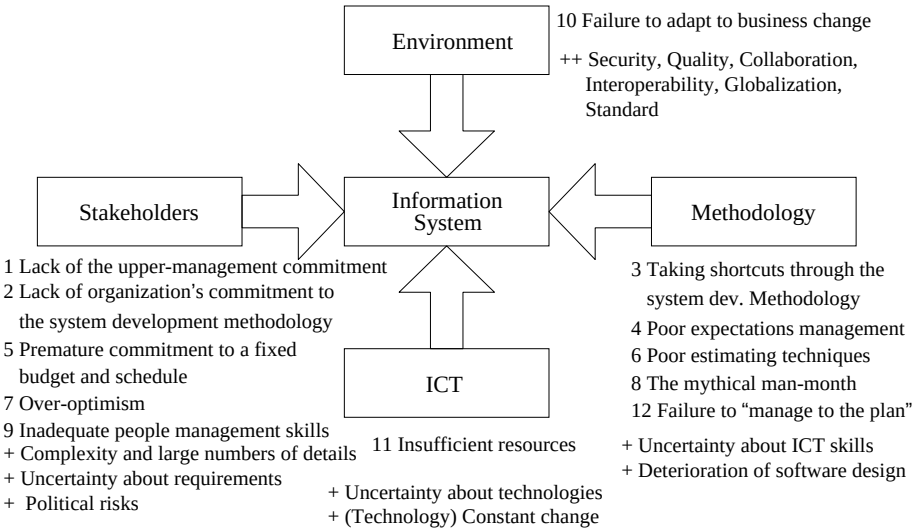
Principles of System Development

1 Get the owners and users involved
2 Use a problem-solving approach
3 Establish phases and activities
4 Establish standards
5 Justify systems as capital investments
6 Don't be afraid to cancel or revise scope
7 Divide and conquer
8 Design systems for growth and change

Case Study: H.J.Waston, et al., Data Warehouse Failure: Case Studies and Findings, The Journal of Data Warehousing, 4(1):44-54, 1999

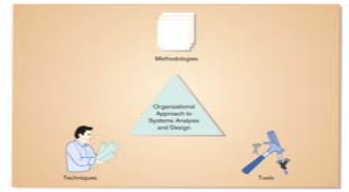
- A real-estate group in the federal government cosponsored a data warehouse with the IT department. A formal proposal was written by IT in which costs were estimated at $800,000, the project duration was estimated to be eight months, and the responsibility for funding was defined as the business unit's. The IT department proceeded with the project before it even knew if the project had been accepted.
- The project actually lasted two years because requirements gathering took nine months instead of one and a half, the planed user base grew from 200 to 2,500, and the approval process to buy technology for the project took a year. Three weeks prior to technical delivery, the IT director canceled the project. This failed endeavor cost the organization and taxpayers $2.5 million.
- Why did this system fail? Why would a company spend money and time on a project and them cancel it? What could have been done to prevent this?

Causes of Failed IS Projects



3. What Is a Methodology?

- A formalized approach to implement SDLC
- Categories
  - Process-centered
  - Data-centered
  - Object-oriented centered



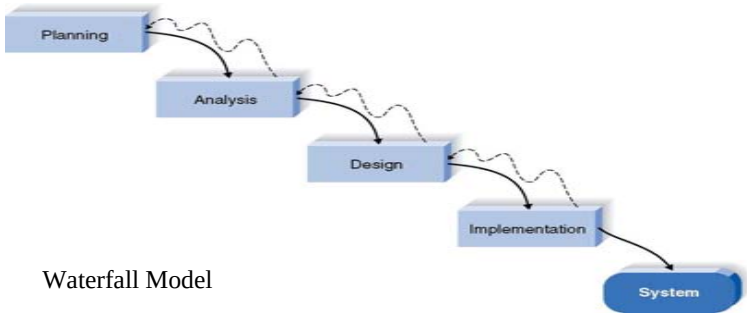
Using Methodology → Walk to the right direction
BUT not guarantee the success
NEED experience

Evolution of Systems Development

- Structured Design
  - Waterfall development
- Rapid Application Development (RAD)
  - Phased Development
  - Incremental and Throw-Away Prototyping
- Agile Development
  - Programming centric methodology: Extreme programming
- Object-Oriented Analysis and Design

Structured Design 1

- Projects move methodically from one to the next step
- Generally, a step is finished before the next one begins

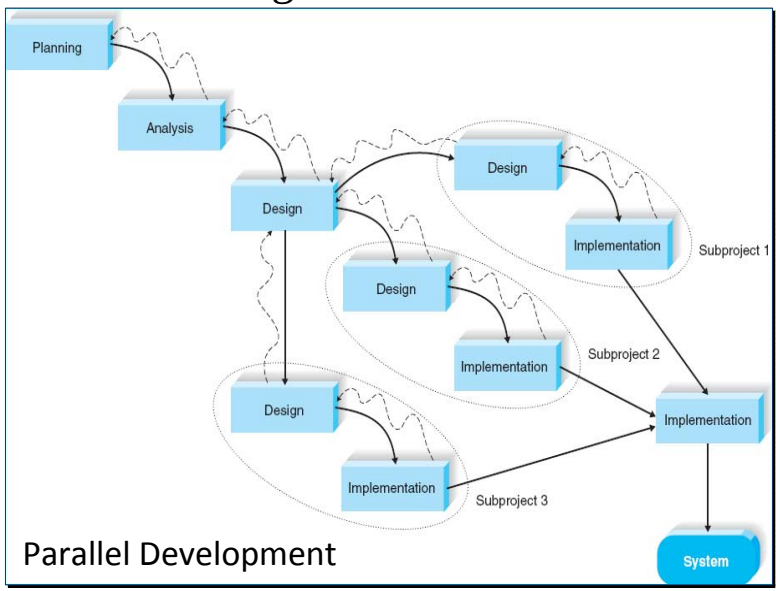


Waterfall Model

Pros and Cons of the Waterfall Model

Pros	Cons
Identifies systems requirements long before programming begins	Design must be specified on paper before programming begins Long time between system proposal and delivery of new system

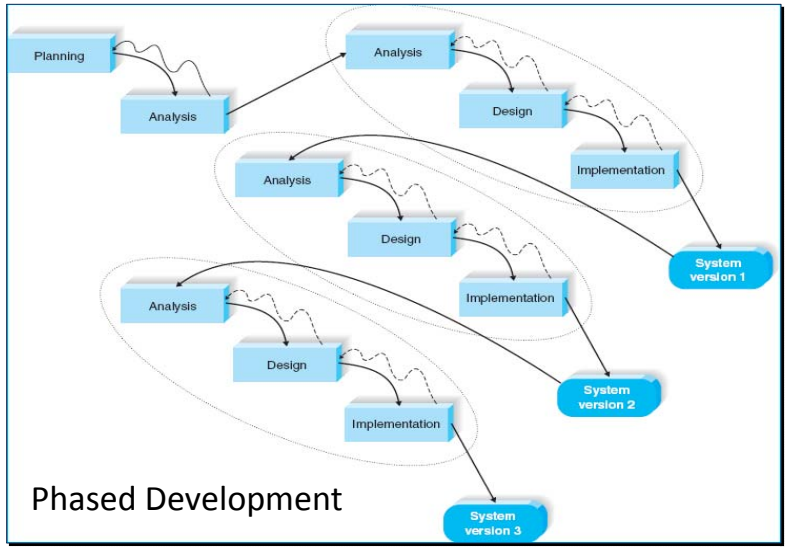
Structured Design 2



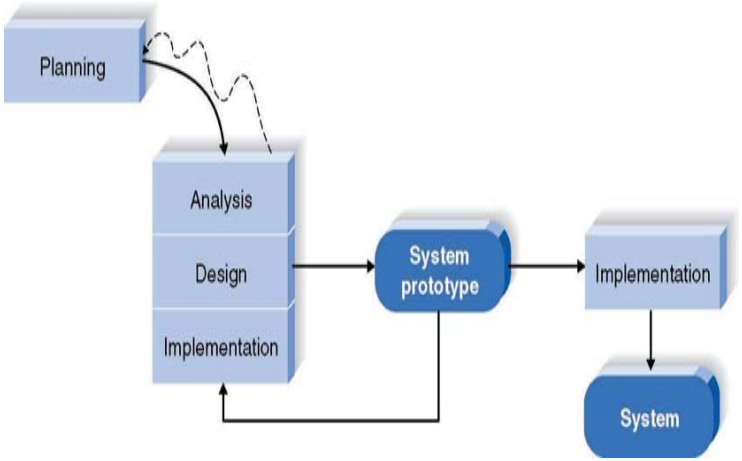
Rapid Application Development (RAD)

- Techniques emphasize user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the system development process
- A prototype is a smaller-scale, representative or working model of the users' requirements or a proposed design for an information system.
  - Incremental & throw-away prototypes

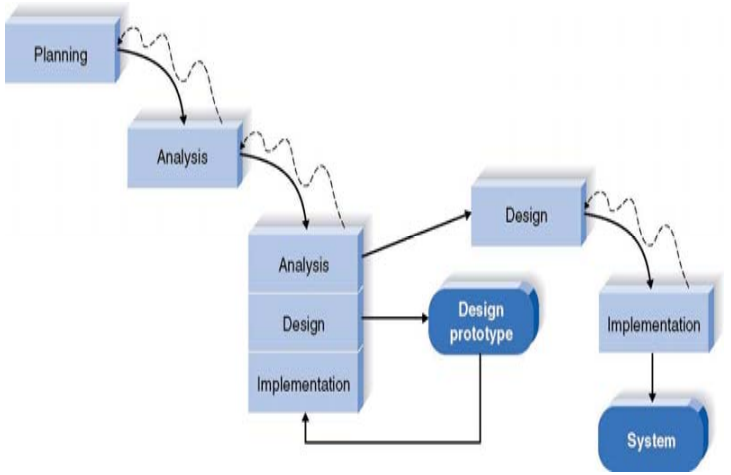
Rapid Application Development 1



Incremental Prototyping

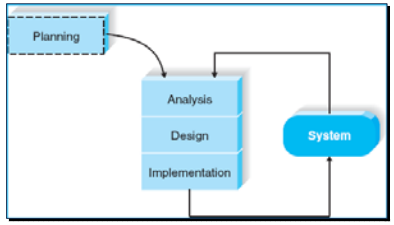


Throw-Away Prototyping



Agile Development: eXtreme Programming

- Short, incremental development cycles
- Automated tests
- Two-person programming teams
- Coding and testing operate together
- Advantages:
  - Communication between developers
  - High level of productivity
  - High-quality code



Selecting the Right Methodology

Usefulness for	**Waterfall**	**Parallel**	**Phased**	**Prototyping**	**Throwaway Prototyping**	**Extreme Programming**
Unclear user requirements	Poor	Poor	Good	Excellent	Excellent	Excellent
Unfamiliar technology	Poor	Poor	Good	Poor	Excellent	Poor
Complex systems	Good	Good	Good	Poor	Excellent	Poor
Reliable systems	Good	Good	Good	Poor	Excellent	Good
Short time schedule	Poor	Good	Excellent	Excellent	Good	Excellent
Schedule visibility	Poor	Poor	Excellent	Excellent	Good	Good

Object-Oriented Analysis and Design

- Attempts to balance emphasis on data and process
- Uses Unified Modeling Language (UML) for diagramming
  - Use-case Driven
  - Architecture Centric: functional, static, dynamic
  - Iterative and Incremental

4. Stakeholder in IS Development

1 Business analyst

2 System analyst

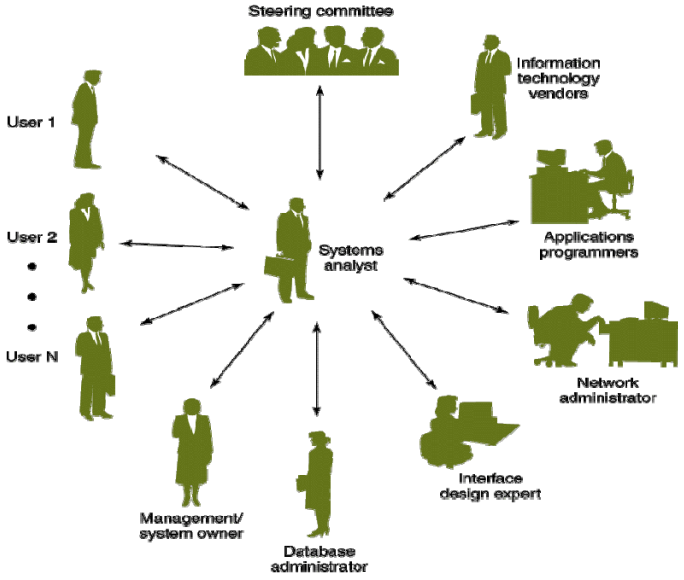
3 Infrastructure analyst

4 Change management analyst

5 Project manager

++ User / Owner / Designer / Programmer

Systems Analyst as a Facilitator



Knowledge & Skills: Systems Analysts

- Knowledge:
  - Information technology
  - Computer programming experience and expertise
  - General business knowledge
- Skills:
  - Problem-solving
  - Interpersonal communication
  - Interpersonal relations
  - Flexibility and adaptability
  - Character and ethics
  - Systems analysis and design

5. Object-Oriented Concepts and UML

1. Objects: an instance of some classes
2. Classes: template to define objects
3. Attributes: describe data aspects of the object
4. Operation: the processes the object can perform
5. Messages: instructions sent to or received from other objects
6. Encapsulation & Information Hiding
7. Inheritance
8. Polymorphism

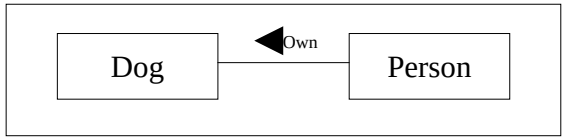
What is Object Orientation?

- Is the way of
  - managing complexity
  - understanding, viewing, modeling a system
- How to managing complexity
  - Thing & characteristics
    - name, address, DOB, etc
  - Function
    - walk, run
  - Thing & its parts
    - head, body, leg, etc.
  - Type of things
    - student, lecturer, etc.

Reducing complexity by means of abstract models

Reality

Model



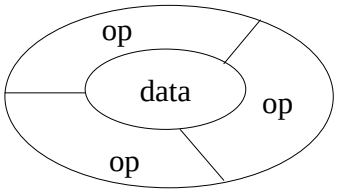
Objects

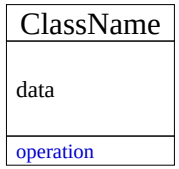
- An object is an instantiation of some classes.
  - Every object belongs to a class.
- An object is an abstraction of something in a problem domain, reflecting the capabilities of the system to
  - keep information about it,
  - interact with it,
  - or both."

Coad and Yourdon (1990)

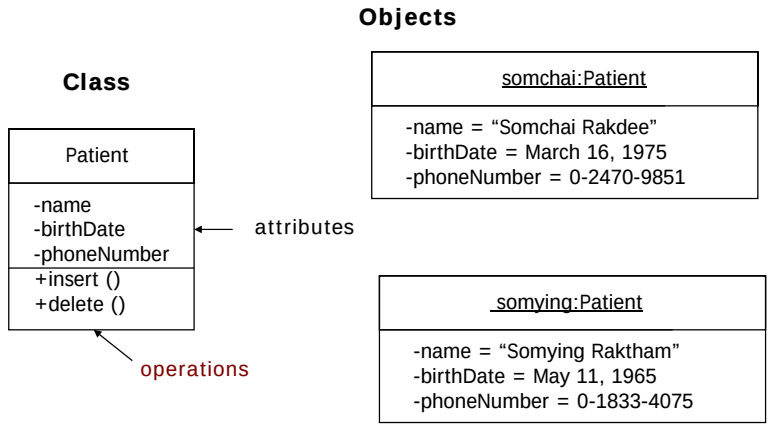
Classes

- A class is a set of objects that share a common structure (attributes or properties) and a common behavior (operations)
  - Represents similar objects (its instance)
- Classes are abstract (no values or specific behavior)
- A particular class member → object (instance)





Object and Class



Exercise: class or object

1. General Motor.
2. Automobile company
3. Student
4. Computer science student
5. Mary Smith.
6. Game
7. Board game
8. Chess.
9. Car
10. Mazda car
11. The game of chess between Tom and Jane which started at 2.30 p.m. yesterday.
12. The car with serial number JM188765T4.

Operation and Method

- Operation
  - A higher-level procedural abstraction that specifies a type of behaviour
  - Independent of any code which implements that behaviour
    - e.g. calculating area (in general)
- Method
  - An implementation of an operation
- Message – a instance of an operation

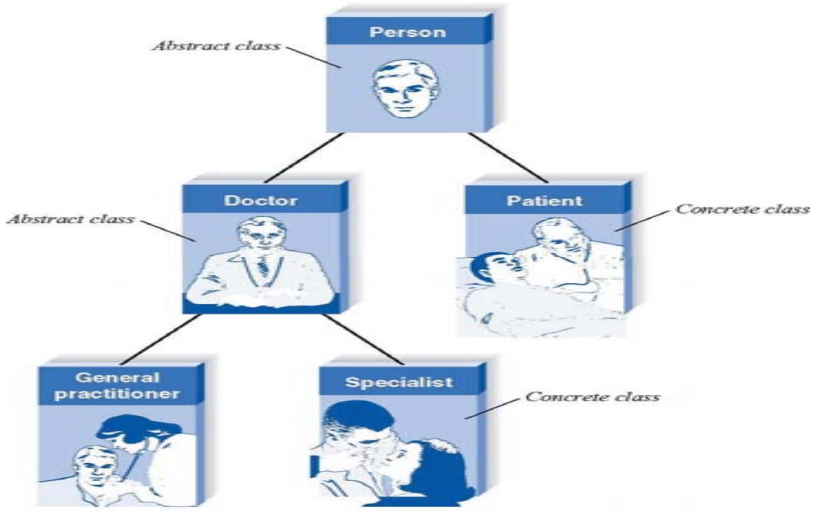
Key to Reusability

- Information hiding is the principle that only information required to use the object is available outside the object
- Encapsulation is the mechanism that combines data and processes in a single object

Specification

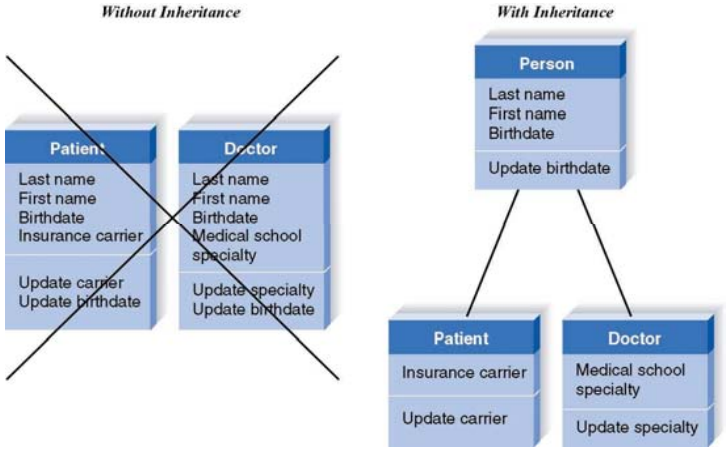
Implementation

Class Hierarchy

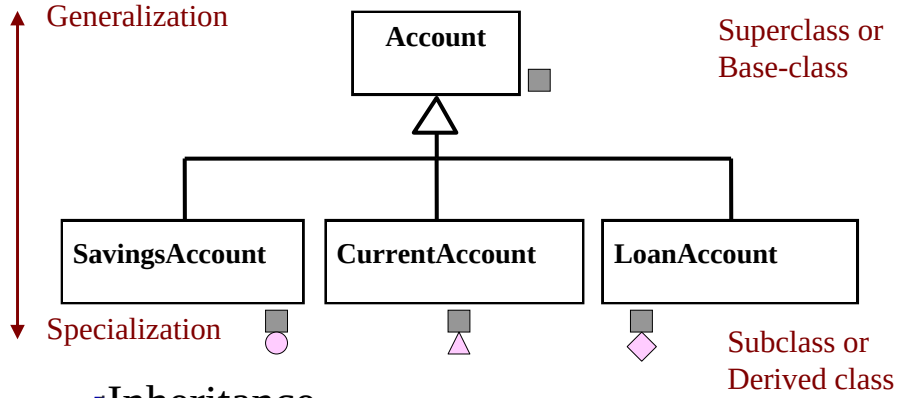


Inheritance

: classes can reuse properties that have defined in other classes



Example of Inheritance Hierarchy



- Inheritance
  - The implicit possession by all subclasses of features defined in its superclasses

Exercise:
Draw a class diagram that identifies all the commonalities between cars and vehicles.

Polymorphism

A property of object oriented software by which an abstract operation may be performed in different ways in different classes.

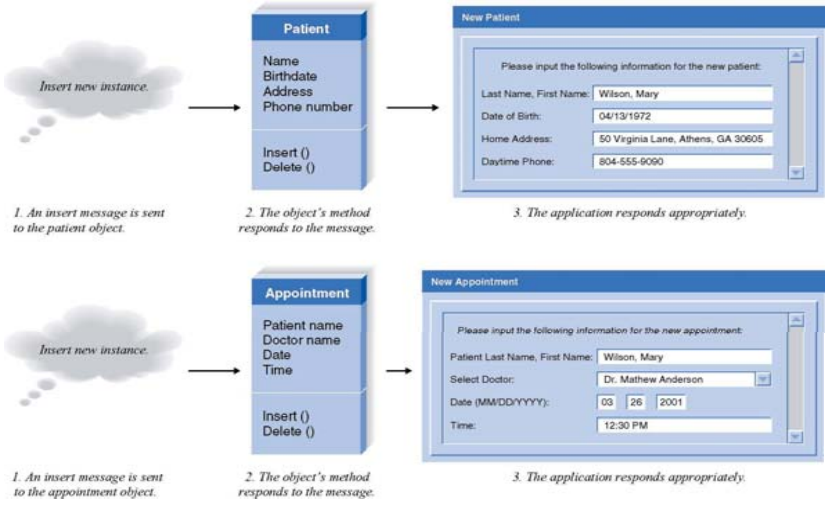
- Polymorphism: “able to assume many forms”.
- Requires that there be multiple methods of the same name
- The choice of which one to execute depends on the object that is in a variable
- Reduces the need for programmers to code many `if-else` or `switch` statements

Polymorphism or Overloading Vs. Overriding

1 + 1 =	+ → add-integer
1.0 + 1.0 =	+ → add-real
‘1’ + ‘1’ =	+ → concatenate
+: Polymorphism or Overloaded symbol	Overriding

Polymorphism

: The same symbol can be interpreted differently by different classes of objects



Advantages of O-O

- Save Effort → faster development
  - Reuse of generalized components cuts work, cost and time
- Higher Quality
  - Encapsulation increases modularity
  - Sub-systems less coupled to each other
  - Better translations between analysis and design models and working code
- Easier Maintenance

UML

- The full UML provides separate diagramming techniques
- Use case / class / interaction (sequence & collaboration) / state / activity / package / deployment

UML 2.0 Diagram Summary

Diagram Name	Used to	Primary Phase
Structure Diagrams		
Class	Illustrate the relationships between classes modeled in the system.	Analysis, Design
Object	Illustrate the relationships between objects modeled in the system. Used when actual instances of the classes will better communicate the model.	Analysis, Design
Package	Group other UML elements together to form higher level constructs.	Analysis, Design, Implementation
Deployment	Show the physical architecture of the system. Can also be used to show software components being deployed onto the physical architecture.	Physical Design, Implementation
Component	illustrate the physical relationships among the software components.	Physical Design, Implementation
Composite Structure	Illustrate the internal structure of a class, i.e., the relationships among the parts of a class.	Analysis, Design
Behavioral Diagrams		
Activity	Illustrate business workflows independent of classes, the flow of activities in a use case, or detailed design of a method.	Analysis, Design
Sequence	Model the behavior of objects within a use case. Focuses on the time-based ordering of an activity.	Analysis, Design
Communication	Model the behavior of objects within a use case. Focuses on the communication among a set of collaborating objects of an activity.	Analysis, Design
Interaction Overview	Illustrate an overview of the flow of control of a process.	Analysis, Design
Timing	Illustrate the interaction that takes place among a set of objects and the state changes in which they go through along a time axis.	Analysis, Design
Behavioral State Machine	Examine the behavior of one class.	Analysis, Design
Protocol State Machine	Illustrates the dependencies among the different interfaces of a class.	Analysis, Design
Use-Case	Capture business requirements for the system and to illustrate the interaction between the system and its environment.	Analysis

FIGURE 2-6 UML 2.0 Diagram Summary

Benefits of the Object Approach

Concept	Supports	Leads to
Classes, objects, methods, and messages	■ A more realistic way for people to think about their business ■ Highly cohesive units that contain both data and processes	■ Better communication between user and analyst developer ■ Reusable objects ■ Benefits from having a highly cohesive system (see cohesion in Chapter 13)
Encapsulation and information hiding	■ Loosely coupled units	■ Reusable objects ■ Fewer ripple effects from changes within an object or in the system itself ■ Benefits from having a loosely coupled system design (see coupling in Chapter 13)
Inheritance	■ Allows us to use classes as standard templates from which other classes can be built	■ Less redundancy ■ Faster creation of new classes ■ Standards and consistency within and across development efforts ■ Ease in supporting exceptions
Polymorphism and Dynamic Binding	■ Minimal messaging that is interpreted by objects themselves	■ Simpler programming of events ■ Ease in replacing or changing objects in a syste ■ Fewer ripple effects from changes within an object or in the system itself
Use-case driven and use cases	■ Allows users and analysts to focus on how a user will interact with the system to perform a single activity	■ Better understanding and gathering of user neec ■ Better communication between user and analys
Architecture centric and functional, static, and dynamic views	■ Viewing the evolving system from multiple points of view	■ Better understanding and modeling of user neec ■ More complete depiction of information system
Iterative and incremental development	■ Continuous testing and refinement of the evolving system	■ Meeting real needs of users ■ Higher quality systems

FIGURE 2-8 Benefits of the Object Approach

6. Software Development

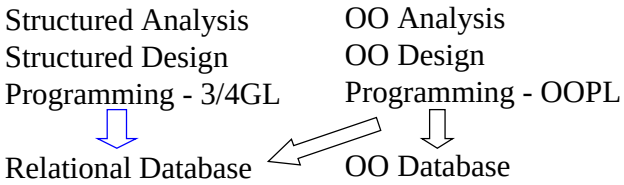
- Software Engineering
  - Time
  - Cost
  - fault-free

⇨ **Software Crisis**

- Understanding Users' Needs
- Requirement Process / Requirement Engineering / Requirement Elicitation

Software Development

- Economic → Long-term Cost
  - 2/3 software cost → Maintenance
  - S/W Engineer → better specification and design
- Software is to model the real world → real world is changing
- Structured Paradigm Vs. OO Paradigm



Software Development

- Software: code & document
  - The pressure to deliver a product on time is such that postpone document may never be completed.
- Software managers are excellent manager but know little about software development and maintenance

Summary

- Systems Development Life Cycle
- System Development Methodology
- IS Stakeholders
- Object-oriented Concepts
- Software Development

Further reading: www.rational.com/uml

Question and Answer

True & False Questions

1. The key person in the SDLC is a project manager.
2. The planning phase is the process of understand how the system should be built.
3. RAD stands for rapid application data.
4. An object is an instance of class.
5. The system analyst focuses on ensuring that the project is competed on-time, within budget, and according to the specification.

Exercise I

1. What is the difference between an object and an attribute?

2. Why is a Toyota automobile is specialization of a car and an engine is not?

Team Assignment: Form a team of 10 members

Project Title (xyz IS)

Team Code: OOSAD-xxx#
1 ID* name email-addr
2 ID** name email-addr
…
n ID name email-addr

Subject: MIT-633
……..

* Team Leader
** Auditor
(sort by ID)

Chapter 1 Introduction

1.1 Motivation

1.2 Objectives

1.3 Scopes

1.4 Benefits

(chapter 1 → only 2-3 pages)

Submission

Date: Two-day before the class

Time: 1.00 p.m.

Email to: wichian@sit.kmutt.ac.th

cc: your team members (to get all your email addr!)

Subject: OOSAD-Sun# (xxx)

(where # is your team number & “xxx” is an assignment#)

Constraint: Send me only one “.doc file”