Object-Oriented Development

Linda M. Northrop

컴퓨터공학부
200711464 이해은

CONTENTS

1. Historical Perspective
2. Motivation
3. Object-Oriented Model
4. Object-Oriented Programming
5. Object-Oriented Software Engineering
6. Object-Oriented Transition
7. The Future

1.Historical Perspective

- 1950's artificial intelligence(A.I.) _ 최초의 object의 개념
- 1966 with the introduction of the Simula language _ 진정한 OO의 움직임
- PARC developed smalltalk in the early 1970's
  - Smalltalk 는 최초의 진정한 OO language

1.Historical Perspective

- Led other languages to support object-oriented programming
  - Objective-C, C++, Self, Eiffel, and Flavors
- 1980 Booch 는object-oriented design (OOD)개념을 개척했다.

2. Motivation

- Benefits are Greater:
  - Productivity
  - Reliability
  - Maintainability
  - Manageability

2. Motivation

- 대상을 객체로 바라보는 것이 보다 인간의 생각과 유사하다.
- Object 는 function 보다 안정적이다.
- OO development는 정보 은닉, 데이터 추상화, 캡슐화 등을 지원한다.
- 쉽게 유지 수정, 확장, 유지 할 수 있다.

3. Object-Oriented Model

: Compute가 의미하는 것이 무엇인가

정보를 어떻게 구조화 할 것인가

3. Object-Oriented Model

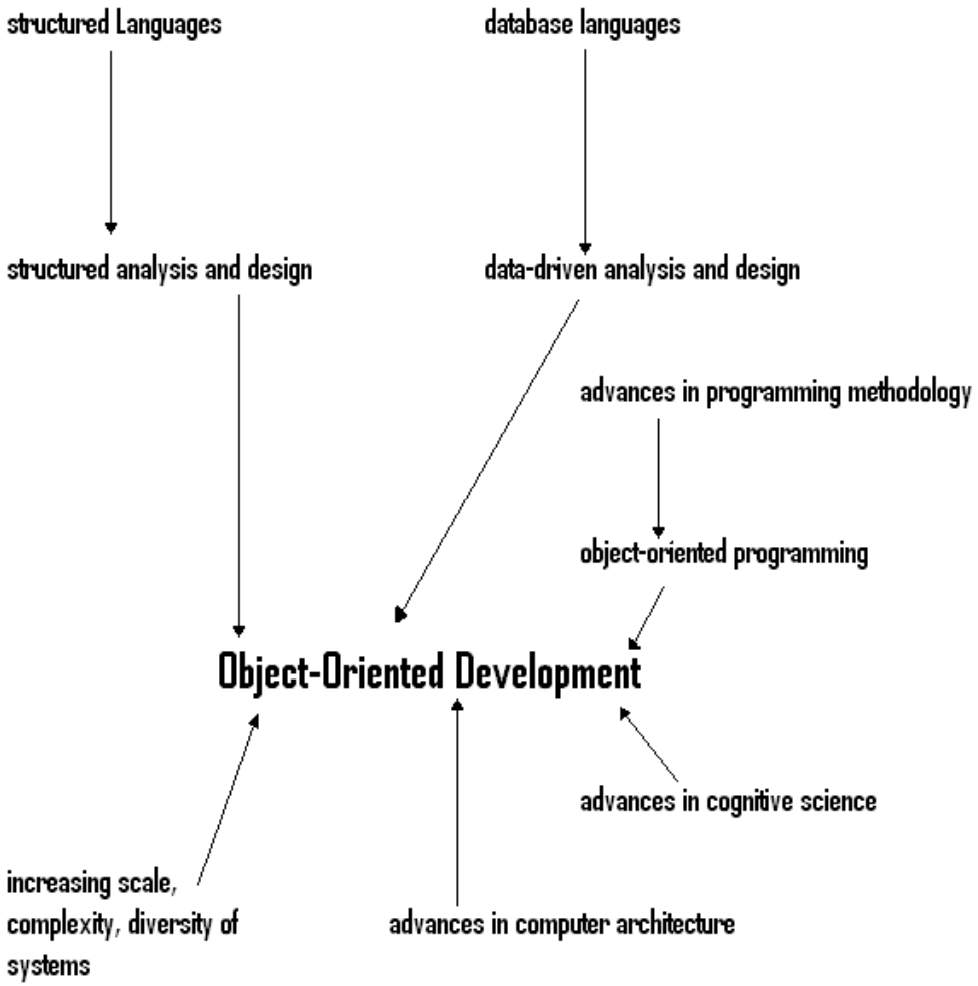
- 아래의 framework에 의해 설명할 수 있다.

  - Abstraction - 추상화
  - Encapsulation - 캡슐화
  - Modularity - 모듈화
  - Hierarchy - 계층구조

3. Object-Oriented Model

- Typing - 정형성
- Concurrence - 병행성
- Persistence - 지속성
- Reusability - 재사용성
- Extensibility - 확장가능성

Influences on OO development



4. Object-Oriented Programming

- Concepts
- Languages

- Concepts

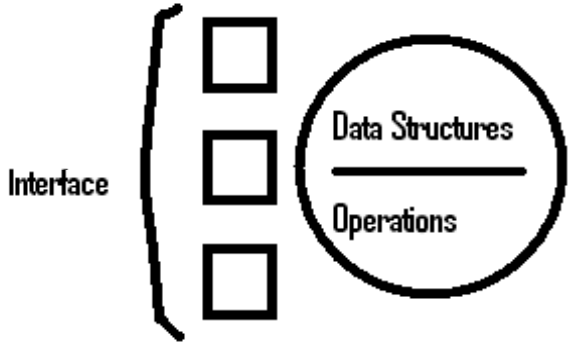


Figure 3. Object-oriented model

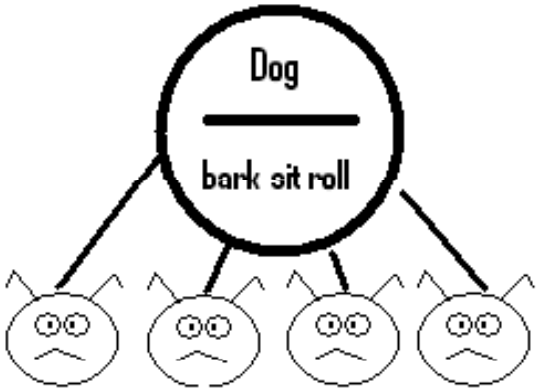


Figure 4. Instansiation of objects from a class

- Object

  :state와 behavior를 캡슐화하는 entity

  Class 의 실례

- Class

  :성질이 비슷한 object의 집합

  새로운 object를 만들기 위한 틀

- attributes, operation, state

- Concepts

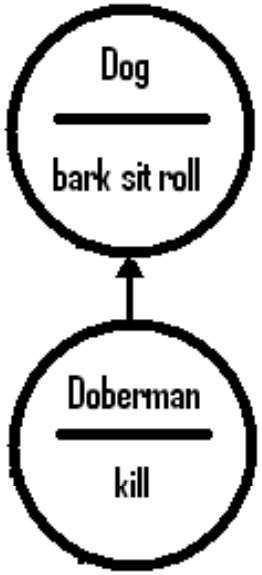


Figure 5. Inheritance

▪ Inheritance (상속)

: 클래스의 capability와 특성들이

그들의 subclass로 옮겨지는 것

객체가 어떤 메시지를 받을 때 상응하는

메소드를 자신의 클래스로부터 시작해

super class로까지 검사를 할 수 있다.

▪ Multiple inheritance (다중상속)

: 클래스가 한 개 이상의 super class로부터

상속받는 것

- Concepts
  - Polymorphism

    : object에 주어진 메시지가 각자의 determination을 기반으로 다르게 해석하여 수행
  - Abstract Class

    : 함수 바디를 가지지 않은 함수를 가지고 있는 클래스

    instance가 없고, 다만 subclass를 만들기 위해서 사용된다.

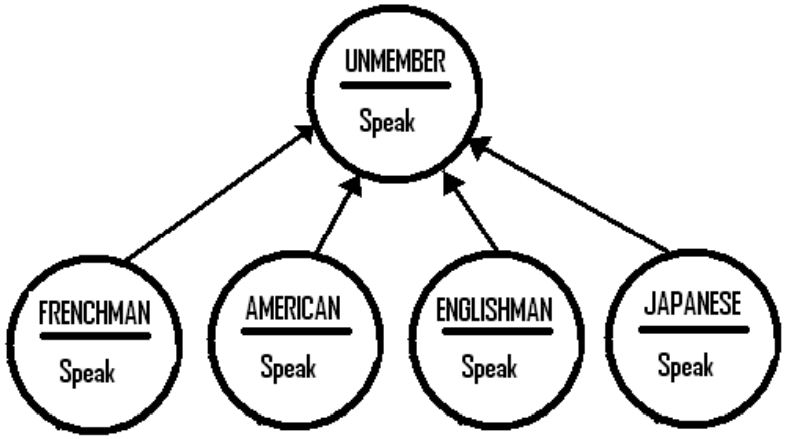


Figure 6. Polymorphism

- Languages

  - Simula 를 근간으로 하는 4 가지 OO언어

    - Smalltalk-based
    - C-based

      : Objective-C, C++, Java
    - LISP-based

      : Flavors, XLISP, LOOPS, CLOS
    - PASCAL-based

      : Object Pascal, Turbo Pascal, Eiffel, Ada 95

  -Object- based

    - Alphard, CLU, Euclid, Gypsy, Mesa, Ada

5. Object-Oriented Software Engineering

- Life Cycle
- Object-Oriented Analysis(OOA) and Object-Oriented Design(OOD)
- Management Issues

- Life cycle
  - Waterfall life Cycle

-하나의 방향으로 순차적으로 진행

-실제iteration을 수용하지 못한다

-Criticized for placing no emphasis on reuse and having no unifying model to integrate the phases

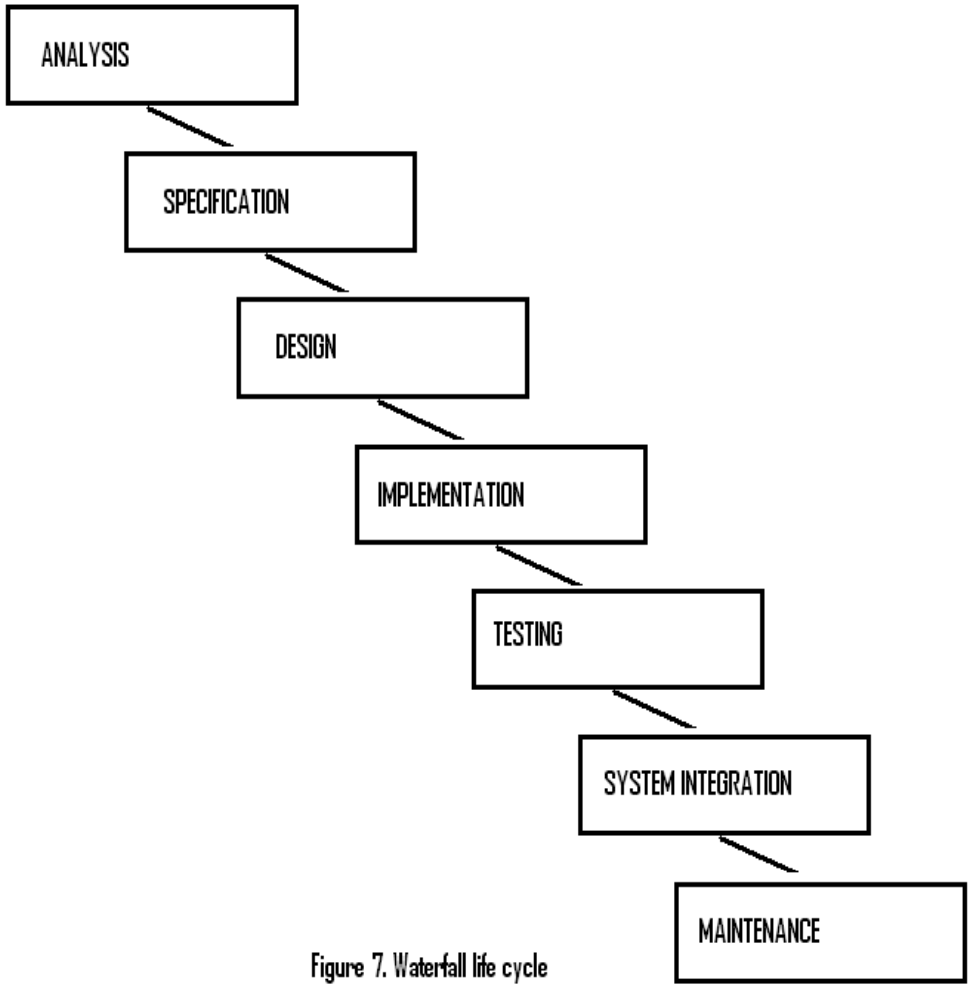


Figure 7. Waterfall life cycle

- Life cycle
  - Water fountain Life Cycle

    - describes the inherent iterative and incremental qualities of object-oriented development

    - Prototyping and feedback loops are standard
  - Iterative/incremental life cycle

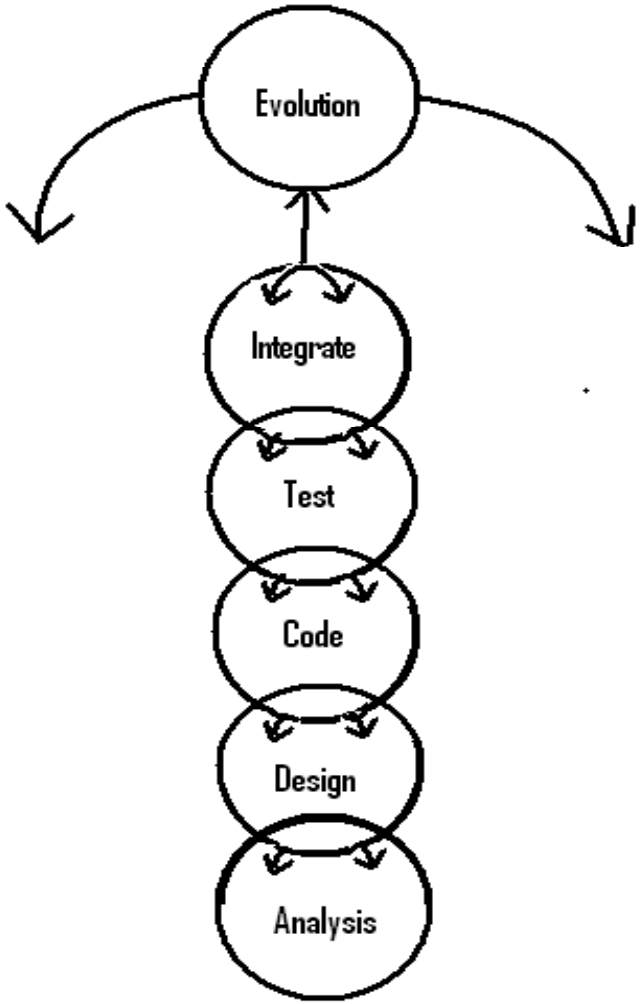


Figure 8. Water fountain life cycle for object-oriented software development

- OOA

  : 객체 지향 적으로 analysis 하는 것

  - Problem domain _ 특정 분야의 문제
  - Frame work _ 소프트웨어 뼈대 구조

- OOD

  : 객체지향적으로 design하는 것

  - 디자인 패턴

    : 다른 domain에 의해 적용되는 재사용성이 있는 asset

6. Object-Oriented Transition

- 이 Transition은 상당한 시간이 걸리고
- 트레이닝이 필요하다.

7. The Future

- 아직도 완성에 이르지 못했다 무한한 잠재력을 가지고 있다.
- OO는 더욱 인기를 얻을 것이고 성숙해 질 것이다.