Object-Oriented Development

Linda M. Northrop

200611517 정훈섭
200711420 권준수
200710118 유희찬
200711448 오희수

INDEX

HISTORICAL PERSPECTIVE

- Object-oriented model
  - Exceedingly attractive as the best answer
  - Object-oriented versions of most languages have been or are being developed.
- Simula(1966)
  - By Kristen Nygaard & Ole-Johan Dahl
- SmallTalk(1972)
  - By PARC (Palo Alto Research Center)
- In 1985, the first commercial object-oriented database system was introduced.

MOTIVATION

- Objects are more stable than functions
- Object-oriented development support
  - Information hiding
  - Data abstraction
  - Encapsulation
    - Easily modified, extended, and maintained
- Object-oriented development
  - Reduce the risk of developing complex systems
    - System integration is diffused throughout the life cycle.

OBJECT-ORIENTED MODEL(1)

- OBJECT-ORIENTED MODEL’S CONCEP
  - Abstraction, Encapsulation, Modularity, Hierarchy
    Typing, Concurrence, Persistence, Reusability, Extensibility

- There are many and varied influences on object-oriented development.

- This approach has not reached maturity, there is still some diversity in thinking and terminology.

OBJECT-ORIENTED MODEL(2)

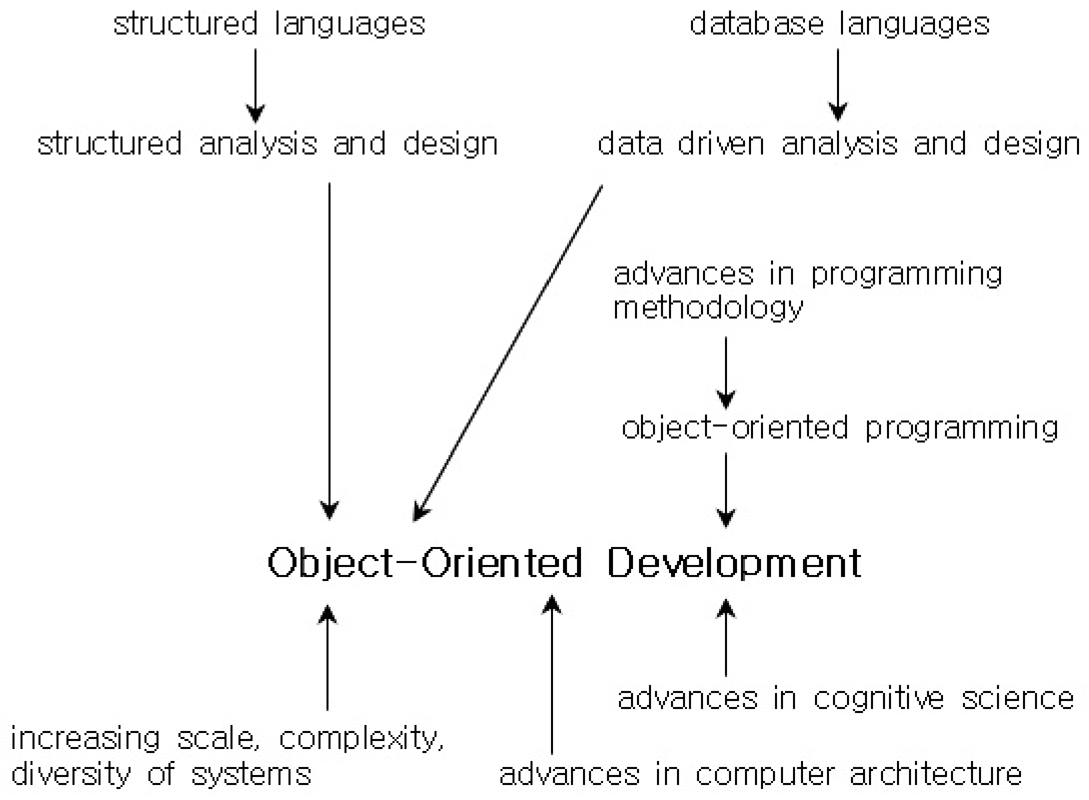
structured languages
structured analysis and design
database languages
data driven analysis and design
advances in programming methodology
object-oriented programming
Object-Oriented Development
increasing scale, complexity, diversity of systems
advances in computer architecture
advances in cognitive science

Figure 1. Influences on object-oriented development

OBJECT-ORIENTED PROGRAMMING(1)

❖ **Concepts**

- OBJECT
  - Functionality is achieved through communication with the interface of an object.

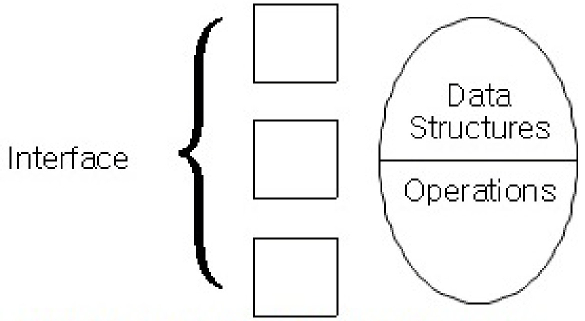

Figure 3. Object-Oriented model.

- CLASS
  - All objects are instances of classes, which are sets of objects with similar Characteristics.
  - A template from which new objects may be created.

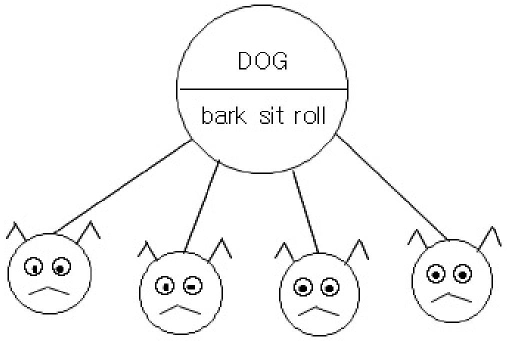

Figure 4. Instantiantion of objects from a class

OBJECT-ORIENTED PROGRAMMING(2)

❖**Concepts**

- INHERITANCE
  - Classes can be arranged in a hierarchy.
    - A subclass will inherit state and behavior from its superclass higher in the inheritance hierarchy structure.
    - Inheritance can be defined as the transfer of a class' capabilities and characteristics to its subclasses.

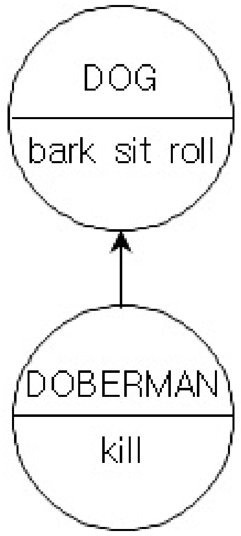

Figure 5. Inheritance.

OBJECT-ORIENTED PROGRAMMING(3)

❖**Concepts**

- POLYMORPHISM
  - Describes the phenomenon in which a given message sent to an object will be interpreted differently at execution based upon subclass determination.

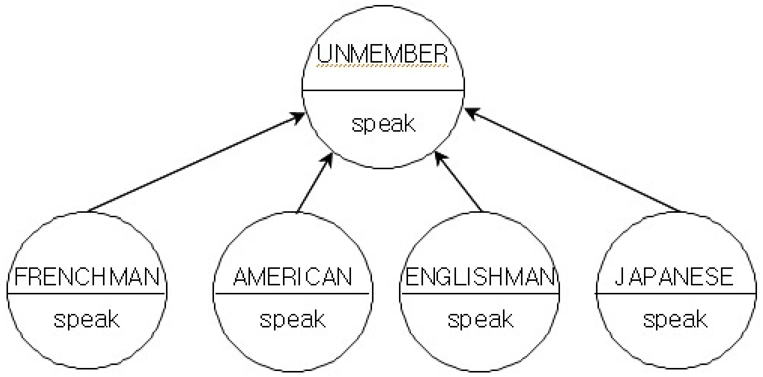

Figure 6. polymorphism.

OBJECT-ORIENTED PROGRAMMING(4)

❖**Languages**

- Four Object-Oriented Languages Based on the Simula
  - 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

OBJECT-ORIENTED SOFTWARE ENGINEERING(1)

❖**Life Cycle**

- Waterfall life cycle
  - The process is sequential.
  - Can’t involve iteration in real developing process.
  - Placing no emphasis on reuse and having no unifying model to integrate the phases

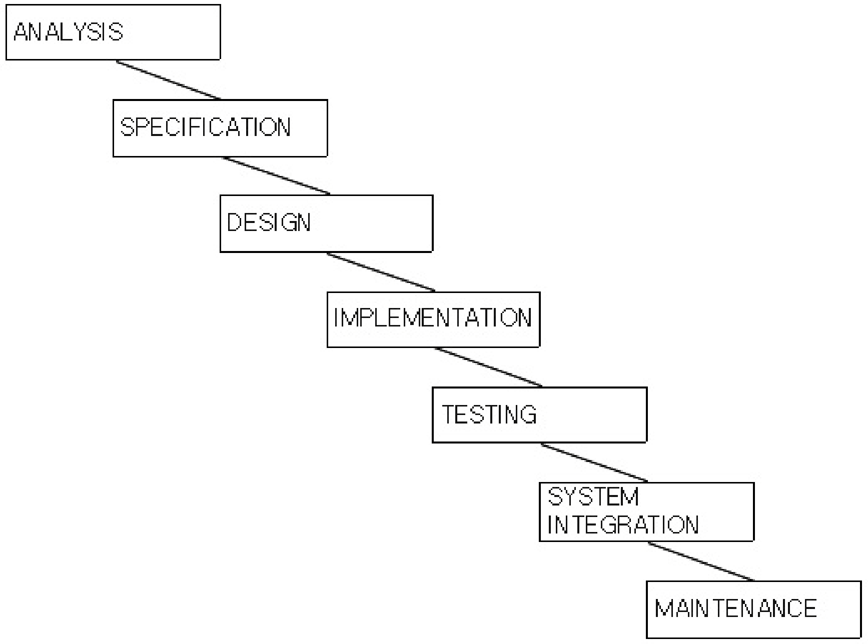

Figure 7. Waterfall life cycle.

OBJECT-ORIENTED SOFTWARE ENGINEERING(2)

❖Life Cycle

- Water fountain life cycle
  - Shows that the development is inherently iterative and seamless.
  - Prototyping and feedback loops are standard.

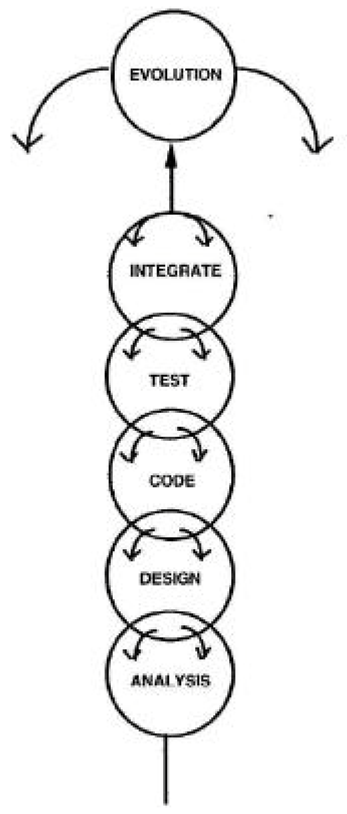

Figure 8. Water fountain life cycle for object-oriented software development.

OBJECT-ORIENTED SOFTWARE ENGINEERING(3)

❖**Life Cycle**

- Iterative/incremental life cycle
  - Analysis
    - to discover and identify the objects
  - Design
    - to invent and design objects
  - Implementation
    - to create objects

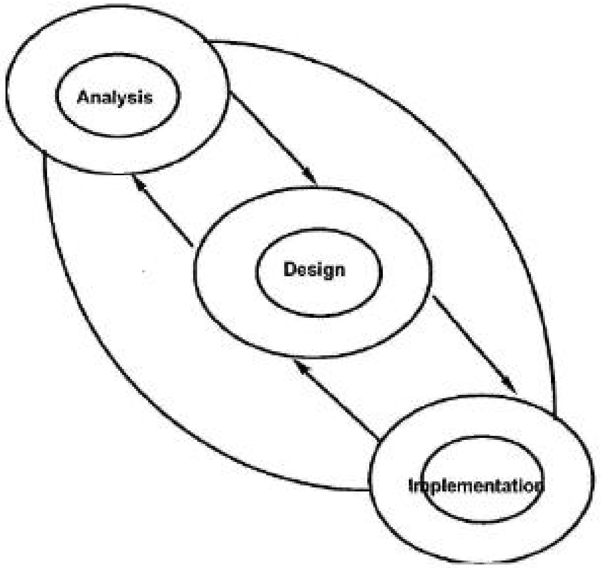

Figure 9. Iterative/incremental life cyle.

OBJECT-ORIENTED SOFTWARE ENGINEERING(4)

❖ **Object-Oriented Analysis (OOA)**

- Object-oriented analysis
  - Build on previous information modeling techniques
- Scenario
  - A sequence of actions that takes place in the problem domain
- Framework
  - A skeleton of an application or application subsystem implemented by concrete and abstract classes

OBJECT-ORIENTED SOFTWARE ENGINEERING(5)

- **Object-Oriented Design (OOD)**
  - OOD techniques were actually defined before OOA techniques were conceived.
  - A design pattern is a recurring design structure or solution that when cataloged in a systematic way can be reused and can form the basis of design communication.

OBJECT-ORIENTED SOFTWARE ENGINEERING(6)

- **Object-Oriented Analysis (OOA) & Object-Oriented Design (OOD)**
  - In both analysis and design, there is a strong undercurrent of reuse.
  - There is difficulty in identifying and characterizing current OOA and OOD techniques because, as described above, the boundaries between analysis and design activities in the object-oriented model are fuzzy.
- Some of the OOA and OOD techniques being used
  - Meyer, Booch's OOD techniques, Wirfs-Brock's OOD technique, Objectory(By Ivar Jacobson), etc.

OBJECT-ORIENTED SOFTWARE ENGINEERING(7)

❖ **Management Issues**

- Management activities that support software development also necessarily have to change.
  - New milestones have to be established.
  - An object-oriented development environment is essential.
- Risks involved in moving to an object-oriented approach.
  - Cost of message passing, explosion of message passing, class encumbrance, paging behavior, dynamic allocation, and destruction overhead.

OBJECT-ORIENTED TRANSITION & FUTURE

- **OBJECT-ORIENTED TRANSITION**
  - Object-Oriented Approach is the successful way for any project.
- **FUTURE**
  - Object-oriented development has not yet reached maturity.
  - Transparent information access across applications and environments is conceivable.
  - It is likely that the movement will continue to gain in popularity and techniques will mature significantly as experience increases.
  - It is also likely that object-orientation will eventually be replaced or absorbed into an approach that works at an even higher level of abstraction.

-THE END-

“Thank you for listening our presentation”