Object-Oriented Development

Linda M. Northrop

200710117 양다빈
200710116 박석희
200711475 허원선
200711476 홍창현

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

- The object and object attribute idea first appeared in the 1950's A.I.
- The real of object-oriented movement began in 1966 with the introduction of the **simula** language.
- Palo Alto Research Center (PARC) developed **smalltalk** in the early 1970's.
  - This time, the term "object-oriented" was coined.
  - Smalltalk is considered the first truly object-oriented language.
- Smalltalk influenced other object-oriented languages.
  - Objective-C, C++, Self, Eiffel, and Flavors.
- In 1980, Grady Booch pioneered the concept of object-oriented design (OOD).
- In 1985, the first commercial object-oriented database system was introduced.
- The 1990s brought an ongoing investigation of object-oriented domain analysis, testing, metrics, and management.
- The current new frontiers in object technology are design patterns, distributed object system and Web-based object applications.

2. Motivation-Because

- Need for greater productivity, reliability, maintainability and manageability.
- Viewing the world as Object is closer to human thinking.
- Objects are more stable than functions.
- Supports information hiding, data abstraction, and encapsulation.
- Easily modified, extended, and maintained.

2. Motivation- Advantage

- Object orientation extends across the life cycle.
  - The life cycle in that a consistent object approach is used from analysis through coding.
- Object approach spawns prototype.
  - Prototypes that support rapid application development.
- OO development supports open systems.
  - There is much greater flexibility to integrate software across applications.
- The use of object-oriented development encourages the reuse of software, design and analysis models.

2. Motivation- Advantage

- OO development supports the concurrency, hierarchy, and complexity.
  - It is currently necessary to build systems.
- Use of the OO approach tends to reduce the risk of developing complex systems.
  - System integration is diffused throughout the life cycle.
- Object technology facilitates interoperability.
  - That is the degree to which an application running on one node of a network can make use of a resource at a different node of the network.

3. Object-Oriented Model

- A new way of thinking about what it means to compute and how information can be structured.
- Systems are viewed as cooperating objects that encapsulate structure and behavior in a hierarchical construction.
- All functionality is achieved by messages that are passed to and from objects.

3. Object-Oriented Model

- Object-oriented model can be viewed as the framework with the following elements.
  - Abstraction.
  - Encapsulation.
  - Modularity.
  - Hierarchy.
  - Typing.
  - Concurrence.
  - Persistence.
  - Reusability.
  - Extensibility.

3. Object-Oriented Model

- Object-Orientation is the integration of procedural and data-driven approaches.
- Language evolution, in turn, has been a natural response to enhanced architecture capabilities and the ever increasingly sophisticated needs of programming systems.
- The impetus for object-oriented software development has followed this general trend.

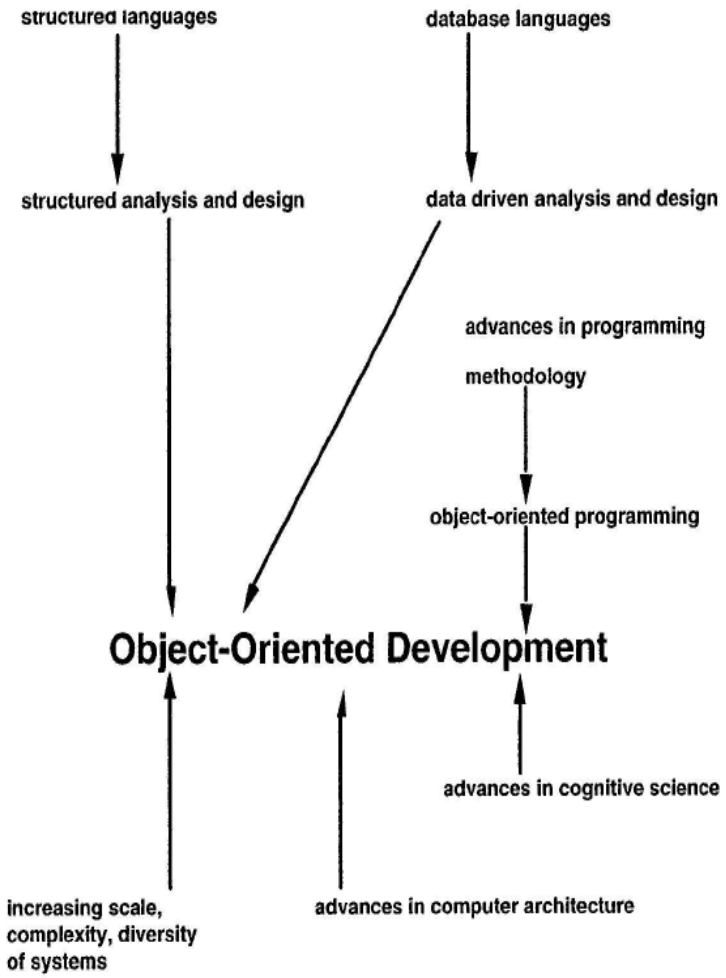

Figure 1. Influences on object-oriented development.

3. Object-Oriented Model

- The most significant factors are the advances in programming methodology.
- The support for abstraction in languages has progressed to higher levels.
- Abstraction progression
  - ADDRESS - machine languages.
  - NAME - assembly languages.
  - EXPRESSION - first generation languages (FORTRAN).
  - CONTROL - second generation languages (COBOL).
  - PROCEDURE AND FUNCTION - second and early third generation languages (PASCAL).
  - MODULES AND DATA - late third generation languages (Modula2).
  - OBJECTS - object-based and object-oriented languages.

3. Object-Oriented Model

- All object-oriented languages are not created equal nor do.
- No complete consensus on how to do object-oriented analysis and object-oriented design.
- Nevertheless, object-oriented development has proven successful in many application areas including.
  - Air traffic control .
  - Banking.
  - Business data processing.
  - Command and control systems.
  - Computer-aided design (CAD).
  - Databases.
  - And so on.
- Object-oriented technology has moved into industrial-strength software development.

4. Object Oriented Programming

- Concepts
  - Object Oriented languages are characterized by:
    - Object creation facility.
    - Message passing capability.
    - Class capability.
    - Inheritance.
  - Polymorphism.
- Languages
  - 4 Branches of object-oriented languages, with Simula being the common ancestor:
    - Smalltalk-based.
    - C-based.
    - LISP-based.
    - PASCAL-based.

4. Object Oriented Programming

- **Concepts**

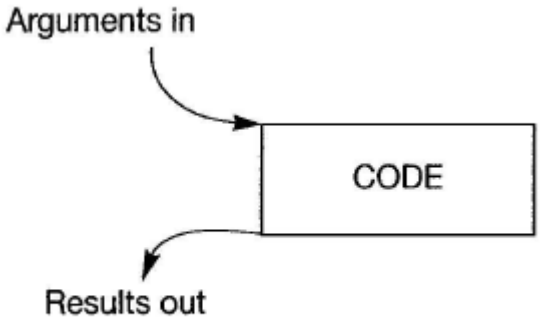

Figure 2. Procedural model.

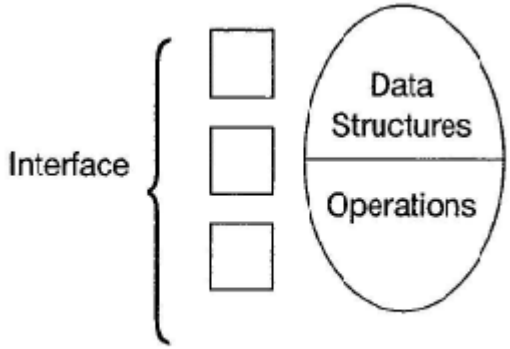

Figure 3. Object-oriented model.

- Object : entity that encapsulates state and behavior.
- State : information needed to be stored in order to carry out the behavior.
- Interface or protocol : set of messages to which it will respond.

4. Object Oriented Programming

- **Concepts**

Figure 4. Instantiation of objects from a class.

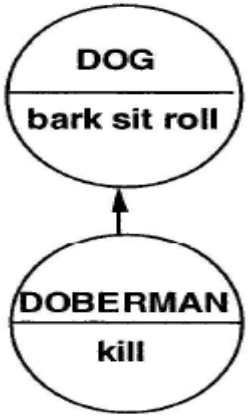

Figure 5. Inheritance.

- All the DOG objects respond in same way to the messages bark, sit and roll. Also have the same state(Data structures).
- Inheritance : the transfer of a class' capabilities and characteristics to its subclasses. And subclass will have behavior particular.
- Multiple inheritance : A given class to inherit from more than one superclass.

4. Object Oriented Programming

- **Concepts**

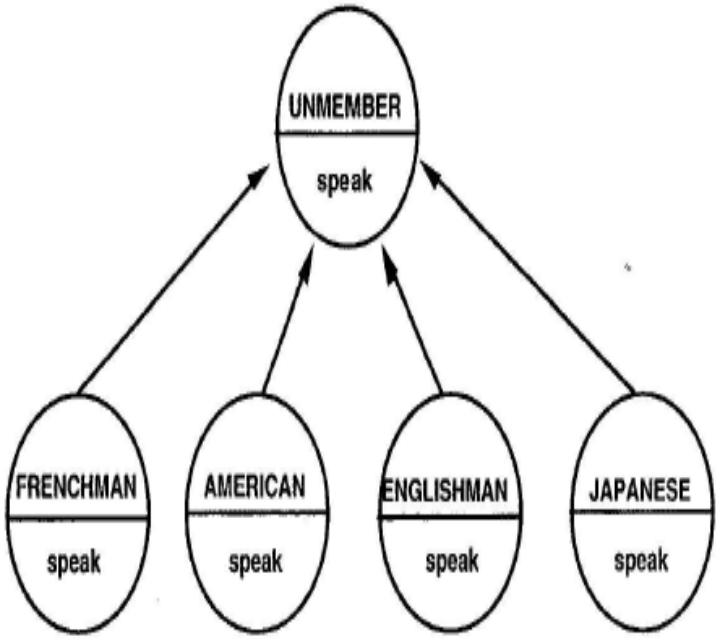

Figure 6. Polymorphism.

- Polymorphism : essentially describes the phenomenon in which a given message sent to an object will be interpreted differently at execution based upon subclass determination.

4. Object Oriented Programming

- **Languages**
  - Simula being the common ancestor:
    - Smalltalk-based (Smalltalk-80).
    - C-based (Objective C, C++, Java).
    - LISP-based (Flavors, XLISP, LOOPS).
    - PASCAL-based (Object Pascal, Turbo Pascal, Eiffel, Ada 95).
  - Object- based (Alphard, CLU, Euclid, Gypsy, Mesa, Ada).

5. Object-Oriented Software Engineering

- Life cycle.
  - Waterfall life cycle
  - Water fountain life cycle
- Object-oriented analysis(OOA) and object-oriented Design(OOD).
- Management issues.

5. Object-Oriented Software Engineering

- **Waterfall life cycle**
  - Waterfall consists of a sequential process, primarily in one direction.
  - Does not accommodate real iteration.
  - Criticized for placing no emphasis on reuse and having no unifying model to integrate the phases.

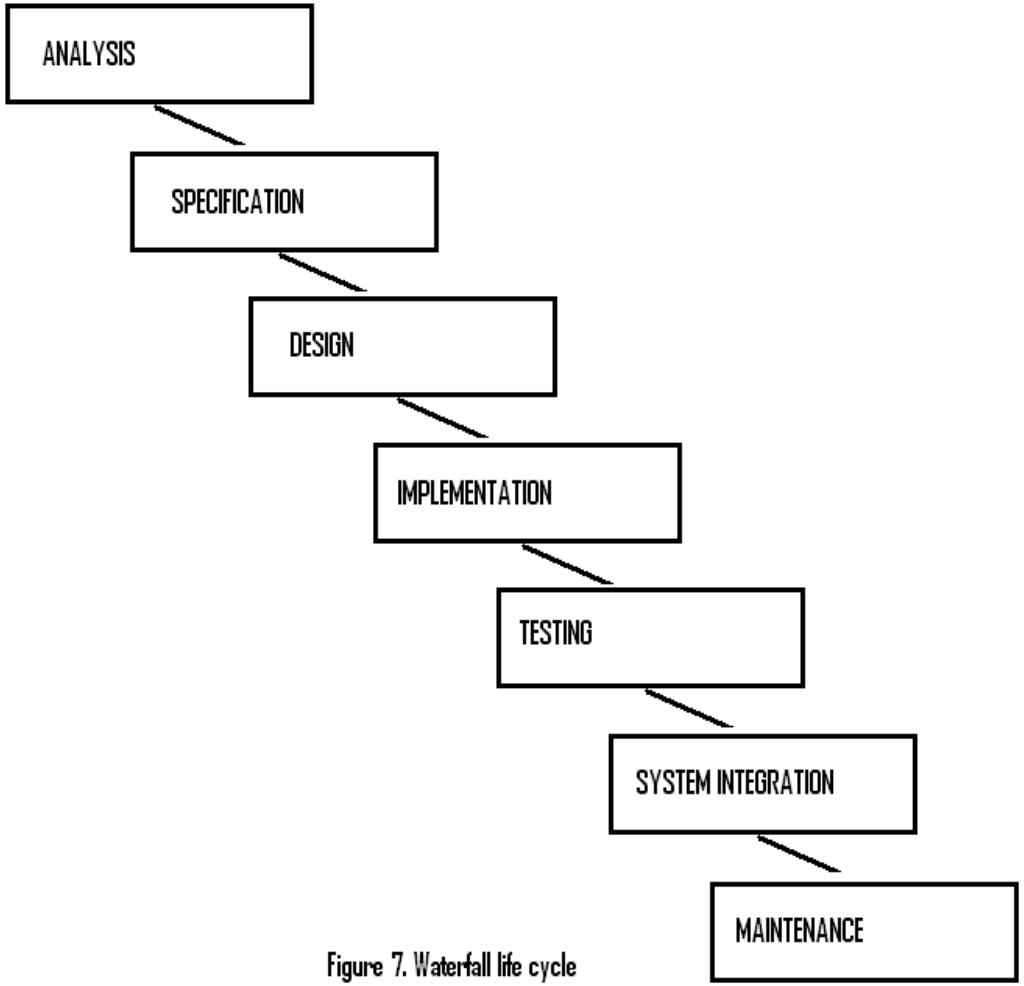

Figure 7. Waterfall life cycle

5. Object-Oriented Software Engineering

- **Water fountain life cycle**
  - Water fountain life cycle describes the inherent iterative and incremental qualities of object-oriented development.
  - Prototyping and feedback loops are standard.

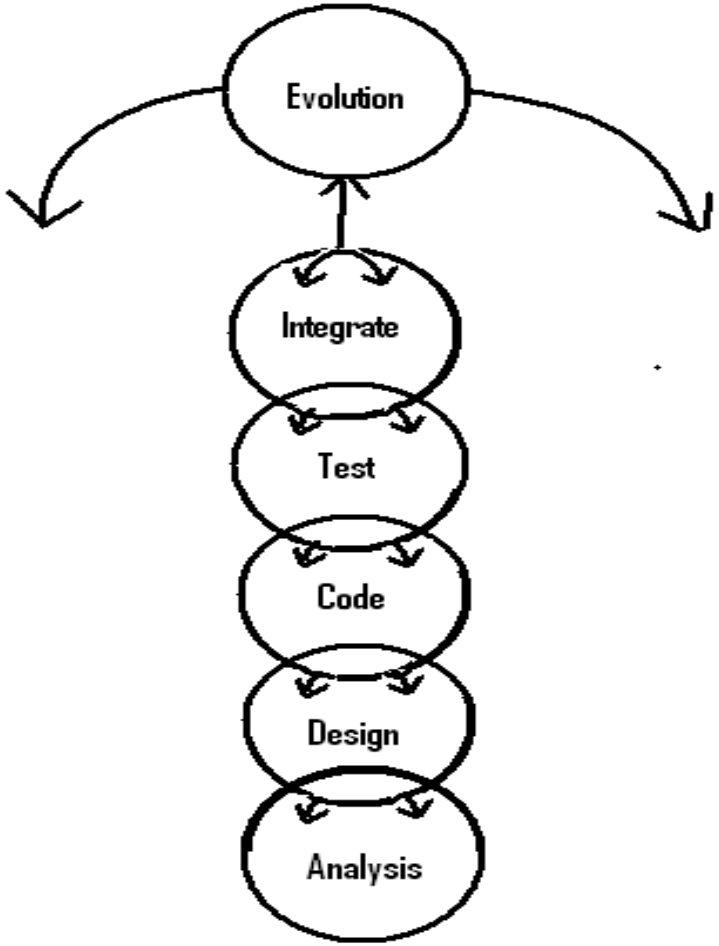

Figure 8. Water fountain life cycle for object-oriented software development

5. Object-Oriented Software Engineering

- **Object-oriented analysis(OOA).**
  - method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.
  - Scenarios can be used to determine necessary object behavior.
  - Frameworks have become very useful in capturing an object-oriented analysis for a given problem domain.
  - Framework is a skeleton of an application implemented by concrete and abstract classes.

5. Object-Oriented Software Engineering

- **Object-oriented Design(OOD).**
  - Object focus shifts to the solution domain.
  - OOD is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical and physical as well as static and dynamic models of the system under design.
  - 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.

5. Object-Oriented Software Engineering

- **OOA and OOD.**
  - There is difficulty in identifying and characterizing current OOA and OOD techniques.
  - because, the boundaries between analysis and design activities in the object-oriented model are fuzzy.

5. Object-Oriented Software Engineering

- **OOA and OOD.**
  - Some of OOA and OOD techniques:
    - Class diagrams, class category diagrams, class templates and object diagrams to record design. (Booch, 1991)
    - Class responsibility cards (CRC) - record class functionality and collaborators. (WirfsBrock, 1990)
    - Object Model - static structure of the objects in a system. (Object diagram)
    - Dynamic Model - aspects of a system that change over time. (State diagram)
    - Functional Model - data value transformation within a system. (Data Flow diagram)
    - find classes and objects, identify structures and relationships, determine subjects, define attributes, and define service, to determine a multilayer object-oriented model.
    - Use cases - basis for analysis model. (Objectory)

5. Object-Oriented Software Engineering

- **Management issues.**
  - The seamless, iterative, prototyping nature of object-oriented development eliminates traditional milestones.
  - New milestones have to be established.
  - LOC(lines of code) measurements are less valuable.
    - Number of classes reused.
    - Inheritance depth.
    - Class-to-class relations.
    - Coupling between objects.
    - Number of classes.
    - Class size are more valuable and meaningful.
  - Resource allocation needs to be reconsidered.
  - Incentives should be based on reuse, not LOC.

5. Object-Oriented Software Engineering

- **Management issues.**
  - Regarding Quality assurance, review and testing activities still essential, but timing and definition must be changed.
  - Tool Support - Object-Oriented Development environment needed along with the other components (incremental compiler, class debugger, browser for class libraries).
  - Estimates are cost of current and future reuse must be factored.
  - The risks involved in moving to an Object-Oriented approach.
    - Performance risks (cost of message passing, explosion of message passing, dynamic allocation).
    - Start-up risks (acquisition of appropriate tools, strategic and appropriate training).

6. Object-Oriented Transition

- The transition needs to progress through Levels of absorption before assimilation into a software development organization occurs.
- This transition period can take considerable time.
- Training is essential.
- Growing evidence that success requires a total object-oriented approach for at least the following reasons:
  - Traceability improvement.
  - Reduction in significant integration problems.
  - Improvement in conceptual integrity of process and product.
  - Maximization of the benefits of object orientation.

7. The Future

- In summary, Object-Oriented development is natural outgrowth of previous approaches.
- Object-Oriented development has not yet reached maturity, the full potential of objects has not been realized.
- Object orientation may eventually be replaced or absorbed into an approach that deals with a higher level of abstraction.
- In the not too distant future, talk about objects will no doubt be passe, but for now there is much to generate genuine enthusiasm.