Chapter 4:

Object-Oriented Methodologies
Goals

- Object-Oriented Methodologies
  - The Rumbaugh et al. OMT
  - The Booch methodology
  - Jacobson's methodologies
Goals (Con't)

- Patterns
- Frameworks
- Unified Approach (UA)
- layered Architecture
Basic Definitions

• A methodology is explained as the science of methods.
• A method is a set of procedures in which a specific goal is approached step by step.
Too Many Methodologies

- **1986**: Booch came up with the object-oriented design concept, the **Booch method**.
- **1987**: Sally Shlaer and Steve Mellor came up with the concept of the **recursive design approach**.
Too Many Methodologies (Con’t)

• 1989: Beck and Cunningham came up with class-responsibility-collaboration (CRC) cards.

• 1990: Wirfs-Brock, Wilkerson, and Wiener came up with responsibility-driven design.

• 1991: Peter Coad and Ed Yourdon developed the Coad lightweight and prototype-oriented approach.
Too Many Methodologies (Con’t)

• **1991**: Jim Rumbaugh led a team at the research labs of General Electric to develop the object modeling technique (OMT).

• **1994**: Ivar Jacobson introduced the concept of the use case.
Survey of Some of the Object-Oriented Methodologies

- Many methodologies are available to choose from for system development.
- Here, we look at the methodologies developed by Rumbaugh et al., Booch, and Jacobson which are the origins of the Unified Modeling Language (UML) and the bases of the UA.
Rumbaugh et. al.’s Object Modeling Technique (OMT)

- OMT describes a method for the analysis, design, and implementation of a system using an object-oriented technique.
OMT (Con’t)

- OMT consists of four phases, which can be performed iteratively:
  - 1. **Analysis**. The results are objects and dynamic and functional models.
  - 2. **System design**. The result is a structure of the basic architecture of the system.
OMT (Cont’d)

3. **Object design.** This phase produces a design document, consisting of detailed objects and dynamic and functional models.

4. **Implementation.** This activity produces reusable, extendible, and robust code.
OMT Modeling

- OMT separates modeling into three different parts:
  - 1. An *object model*, presented by the object model and the data dictionary.
OMT Functional Model

Data Store

Process

Data Flow

External Entity

Client Account

Card Reader

PIN Code → Process PIN Code
The Booch Methodology

- The Booch methodology covers the analysis and design phases of systems development.
- Booch sometimes is criticized for his large set of symbols.
The Booch Methodology (Con't)

- The Booch method consists of the following diagrams:
  - Class diagrams
  - Object diagrams
  - State transition diagrams
  - Module diagrams
  - Process diagrams
  - Interaction diagrams
The Booch Methodology (Con't)

```
Car
  color
  manufacturer
  cost
  inherits
  superclass
Ford
  inherits
  Mustang
  Taurus
  Escort
```

- Car:
  - color
  - manufacturer
  - cost
  - inherits (superclass)

- Ford:
  - inherits
  - Mustang
  - Taurus
  - Escort
The Booch Methodology (Con't)

- **Silenced**
  - SoundAlarm to **Enabled**
  - SilenceAlarm to **Silenced**
- **Enabled**
- **Sounding**
  - SoundAlarm from **Enabled**
  - SilenceAlarm from **Sounding**
- **Disabled**
  - Enable to **Enabled**
  - Disable to **Disabled**
- **AlarmFixed** to **Disabled**

Operator::TurnOffAlarm
The Booch Methodology (Con't)

- The Booch methodology prescribes
  - A macro development process
  - A micro development process.
The Macro Development Process

• The macro development process consists of the following steps:
  – 1. Conceptualization
  – 2. Analysis and development of the model.
  – 3. Design or create the system architecture.
  – 4. Evolution or implementation.
The Micro Development Process

• The micro development process consists of the following steps:
  – 1. Identify classes and objects.
  – 2. Identify class and object semantics.
  – 3. Identify class and object relationships.
  – 4. Identify class and object interfaces and implementation.
The Jacobson et al. Methodologies

The Jacobson et al. methodologies (e.g., OOBE, OOSE, and Objectory) cover the entire life cycle and stress traceability between the different phases.
Use Cases

• Use cases are scenarios for understanding system requirements.
• A use case is an interaction between users and a system.
• The use-case model captures the goal of the user and the responsibility of the system to its users.
Use Cases (Con't)

- The use case description must contain:
  - *How* and *when* the use case begins and ends.
  - The interaction between the use case and its actors, including *when* the interaction occurs and *what* is exchanged.
Use Cases (Con’t)

- **How** and **when** the use case will store data in the system.
- **Exceptions** to the flow of events.
Object-oriented software engineering (OOSE), also called Objectory, is a method of object-oriented development with the specific aim to fit the development of large, real-time systems.
Objectory (Con't)

- Objectory is built around several different models:
  - Use case model.
  - Domain object model.
  - Analysis object model.
  - Implementation model.
  - Test model.
Object-Oriented Business Engineering (OOBE)

- Object-oriented business engineering (OOBE) is object modeling at the enterprise level.
- Use cases again are the central vehicle for modeling, providing traceability throughout the software engineering processes.
Use-case model

Express in: Domain Object model
Structured by: Analysis model
Realized by: Design model
Implemented by: Implementation model
Tested in: Testing model

OK
NOT OK
OOBE (Con’t)

- OOBE consists of:
  - Analysis phase
  - Design
  - Implementation phases and
  - Testing phase.
Patterns

• A pattern is an instructive information that captures the essential structure and insight of a successful family of proven solutions to a recurring problem that arises within a certain context and system of forces.
Patterns (Con’t)

- The main idea behind using patterns is to provide documentation to help categorize and communicate about solutions to recurring problems.

- The pattern has a name to facilitate discussion and the information it represents.
Patterns (Con’t)

• A good pattern will do the following:

• *It solves a problem.* Patterns capture solutions, not just abstract principles or strategies.

• *It is a proven concept.* Patterns capture solutions with a track record, not theories or speculation.
Patterns (Con’t)

• **The solution is not obvious.** The best patterns generate a solution to a problem indirectly—a necessary approach for the most difficult problems of design.

• **It describes a relationship.** Patterns do not just describe modules, but describe deeper system structures and mechanisms.
Patterns (Con’t)

• The pattern has a significant human component.

• All software serves human comfort or quality of life; the best patterns explicitly appeal to aesthetics and utility.
Capturing Patterns

• Guidelines for capturing patterns:
  – *Focus on practicability.*
  – *Aggressive disregard of originality.*
  – *Nonanonomous review.*
  – *Writers' workshops instead of presentations.*
  – *Careful editing.*
Frameworks

- A framework is a way of presenting a generic solution to a problem that can be applied to all levels in a development.
- A single framework typically encompasses several design patterns and can be viewed as the implementation of a system of design patterns.
Differences Between Design Patterns and Frameworks

- Design patterns are more abstract than frameworks.
- Design patterns are smaller architectural elements than frameworks.
- Design patterns are less specialized than frameworks.
The Unified Approach

• The idea behind the UA is not to introduce yet another methodology.

• The main motivation here is to combine the best practices, processes, methodologies, and guidelines along with UML notations and diagrams.
Identify Actors

**Development**
- **Object-Oriented Systems Development**
- **Documentation & Traceability**
  - Use-cases, analysis, design, UI, past experiences
  - Patterns, UML Based Model Construction

**Construction**
- Component Based Development
- Continuous Testing
- User satisfaction, usability tests, quality assurance tests

**O-O Analysis**
- Repository of use-cases, analysis, design, UI, past experiences
  - Patterns, Documentation & Traceability
- O-O Analysis
  - Develop Use-Cases, activity diagrams prototyping
- Develop interaction diagrams
- Identify classes, relationships, attributes & methods
- Refine and iterate

**O-O Design**
- Design classes, their attributes, methods, association, structure ...
- Apply Design Axioms
- Build UML class diagram
- Design view and access Layers and prototypes
- User satisfaction & Usability tests based on use cases
- Continuous Testing

**Layered Approach**
- UML Based Modeling
The Unified Approach (UA)

- The unified approach to software development revolves around (but is not limited to) the following processes and components.
UA Processes (Con't)

- The processes are:
  - Use-case driven development.
  - Object-oriented analysis.
  - Object-oriented design.
  - Incremental development and prototyping.
  - Continuous testing.
UA Methods and Technology

• The methods and technology employed includes:
  – Unified modeling language (UML) used for modeling.
  – Layered approach.
  – Repository for object-oriented system development patterns and frameworks.
  – Promoting Component-based development.
UA Object-Oriented Analysis: Use-Case Driven

• The use-case model captures the user requirements.

• The objects found during analysis lead us to model the classes.

• The interaction between objects provide a map for the design phase to model the relationships and designing classes.
UA Object-Oriented Design

• Booch provides the most comprehensive object-oriented design method.
• However, Booch methods can be somewhat imposing to learn and especially tricky to figure out where to start.
• UA realizes this by combining Jacobson et al.'s analysis with Booch's design concept to create a comprehensive design process.
Iterative Development and Continuous Testing

• The UA encourages the integration of testing plans from day 1 of the project.

• Usage scenarios or Use Cases can become test scenarios; therefore, use cases will drive the usability testing.
Modeling Based on the Unified Modeling Language

- The UA uses the unified modeling language (UML) to describe and model the analysis and design phases of system development.
The UA Proposed Repository

- The requirement, analysis, design, and implementation documents should be stored in the repository, so reports can be run on them for traceability.

- This allows us to produce designs that are traceable across requirements, analysis, design, implementation, and testing.
The Layered Approach to Software Development

• Most systems developed with today's CASE tools or client-server application development environments tend to lean toward what is known as two-layered architecture: interface and data.
Two-Layer Architecture

- In a two-layer system, user interface screens are tied directly to the data through routines that sit directly behind the screens.
Problem With the Two-Layer Architecture

- This approach results in objects that are very specialized and cannot be reused easily in other projects.
Three-Layer Architecture

• Your objects are completely independent of how:
  – they are represented to the user (through an interface) or
  – how they are physically stored.
User Interface layer

This layer is typically responsible for two major aspects of the applications:

- Responding to user interaction
- Displaying business objects.
**Business Layer**

- The responsibilities of the business layer are very straightforward:
- model the objects of the business and how they interact to accomplish the business processes.
These objects should not be responsible for:

- Displaying details
- Data access details
Access Layer

- The access layer contains objects that know how to communicate with the place where the data actually resides,
- Whether it be a relational database, mainframe, Internet, or file.
Access Layer

- The access layer has two major responsibilities:
  - Translate request
  - Translate result
Three-Layered Architecture

Access Layer

Business Layer

View Layer

Object-Oriented Systems Development  Bahrami  ? Irwin/ McGraw-Hill
Summary

• we looked at current trends in object-oriented methodologies, which have been toward combining the best aspects of today's most popular methods.
Summary (Con’t)

- Each method has its strengths. Rumbaugh et al. have a strong method for producing object models.
- Jacobson et al. have a strong method for producing user-driven requirement and object-oriented analysis models.
- Booch has a strong method for producing detailed object-oriented design models.
Summary (Con’t)

• Each method has weakness, too. While OMT has strong methods for modeling the problem domain, OMT models cannot fully express the requirements.

• Jacobson, although covering a fairly wide range of the life cycle, does not treat object-oriented design to the same level as Booch, who focuses almost entirely on design, not analysis.
Summary (Con’t)

• The UA is an attempt to combine the best practices, processes, and guidelines along with UML notations and diagrams for better understanding of object-oriented concepts and object-oriented system development.