Software Engineering I

Object-Oriented Approach

Dr. Elham Mahmoudzadeh
Isfahan University of Technology
mahmoudzadeh@iut.ac.ir

2022

Introduction

- The primary difference between a traditional approach and an object-oriented approach is how a problem is decomposed.
- In traditional approaches, the problem-decomposition process is either process-centric or data-centric.
  - Processes and data are so closely related that it is difficult to pick one or the other as the primary focus.
- *Object-oriented methodologies* attempt to balance the emphasis between process and data by focusing the decomposition of problems on objects that contain both data and processes.

Modern object-oriented approach for developing information systems

- Use-case driven,
- Architecture-centric,
- Iterative and incremental.

- *Use case* is the primary modeling tool.

- A use case describes how the user interacts with the system to perform some activity, such as placing an order, making a reservation, or searching for information.

- A use case is used to identify and to communicate the requirements for the system to the programmers who must write the system.

- Also, use case is used for testing.

2- Architecture-Centric

- Any modern approach to systems analysis and design should be architecture-centric.

- Support at least three separate but interrelated architectural views of a system: *functional*, *structural*, and *behavioral*.

Three views of the system

- The *functional*, or *external view:* describes the behavior of the system from the perspective of the user.

- The *structural*, or *static view:* describes the system in terms of attributes, methods, classes, and relationships.

- The *behavioral*, or *dynamic view:* describes the behavior of the system in terms of messages passed among objects and state changes within an object.

3- Iterative and Incremental

- Modern object-oriented systems analysis and design approaches emphasize *iterative* and *incremental* development that undergoes continuous testing and refinement throughout the life of the project.

- This implies that the systems analysts develop their understanding of a user’s problem by building up the three architectural views little by little.

Iterative Development (I)

- Is a planned rework strategy.
- We use multiple passes to improve what we are building so we can converge on a good solution.
- Is an excellent way to improve the product as it is being developed.
- The biggest downside is that in the presence of uncertainty it can be difficult up front to determine (plan) how many improvement passes will be necessary.

Iterative Development (II)

- For example, we might start by creating a prototype to acquire important knowledge about a poorly known piece of the product. Then we might create a revised version that is somewhat better, which might in turn be followed by a pretty good version.

- In the course of writing this book, for example, I wrote and rewrote each of the chapters several times as I received feedback and as my understanding of how I wanted to communicate a topic improved.

Incremental Development (I)

- Based on the age-old principle of “Build some of it before you build all of it.”
- We avoid having one large, big-bang-style at the end of development.
- Instead, we break the product into smaller pieces so that we can build some of it, learn how each piece is to survive in the environment in which it must exist, adapt based on what we learn, and then build more of it.

Incremental Development (II)

- Slices the system functionality into increments (portions).
- Gives us important information that allows us to adapt our development effort and to change how we proceed.

- The biggest drawback to is that by building in pieces, we risk missing the big picture (we see the trees but not the forest).

Incremental Development (III)

- For example, while writing this book, I wrote a chapter at a time and sent each chapter out for review as it was completed, rather than trying to receive feedback on the entire book at once.

- This gave me the opportunity to incorporate that feedback into future chapters, adjusting my tone, style, or delivery as needed.

- It also gave me the opportunity to learn incrementally and apply what I learned from earlier chapters to later chapters.

Iterative and Incremental

- The systems analyst does this by working with the user to create a functional representation of the system under study.
- Next, the analyst attempts to build a structural representation of the evolving system. Using the structural representation of the system, the analyst distributes the functionality of the system over the evolving structure to create a behavioral representation of the evolving system.
- As an analyst works with the user in developing the three architectural views of the evolving system, the analyst iterates over each of and among the views.
- That is, as the analyst better understands the *structural* and *behavioral* views, the analyst uncovers missing requirements or misrepresentations in the *functional* view.

References

- **Dennis, Wixon, Tegarden, "System Analysis and Design, An Object Oriented Approach with UML", 5th Edition, 2015.**

What we will talk about next...

- RUP
- Scrum