Object-Oriented Analysis and Design using JAVA

B.Tech (CSE/IT) 5th SEM
2021-2022

Lecture-10: System design

Introduction

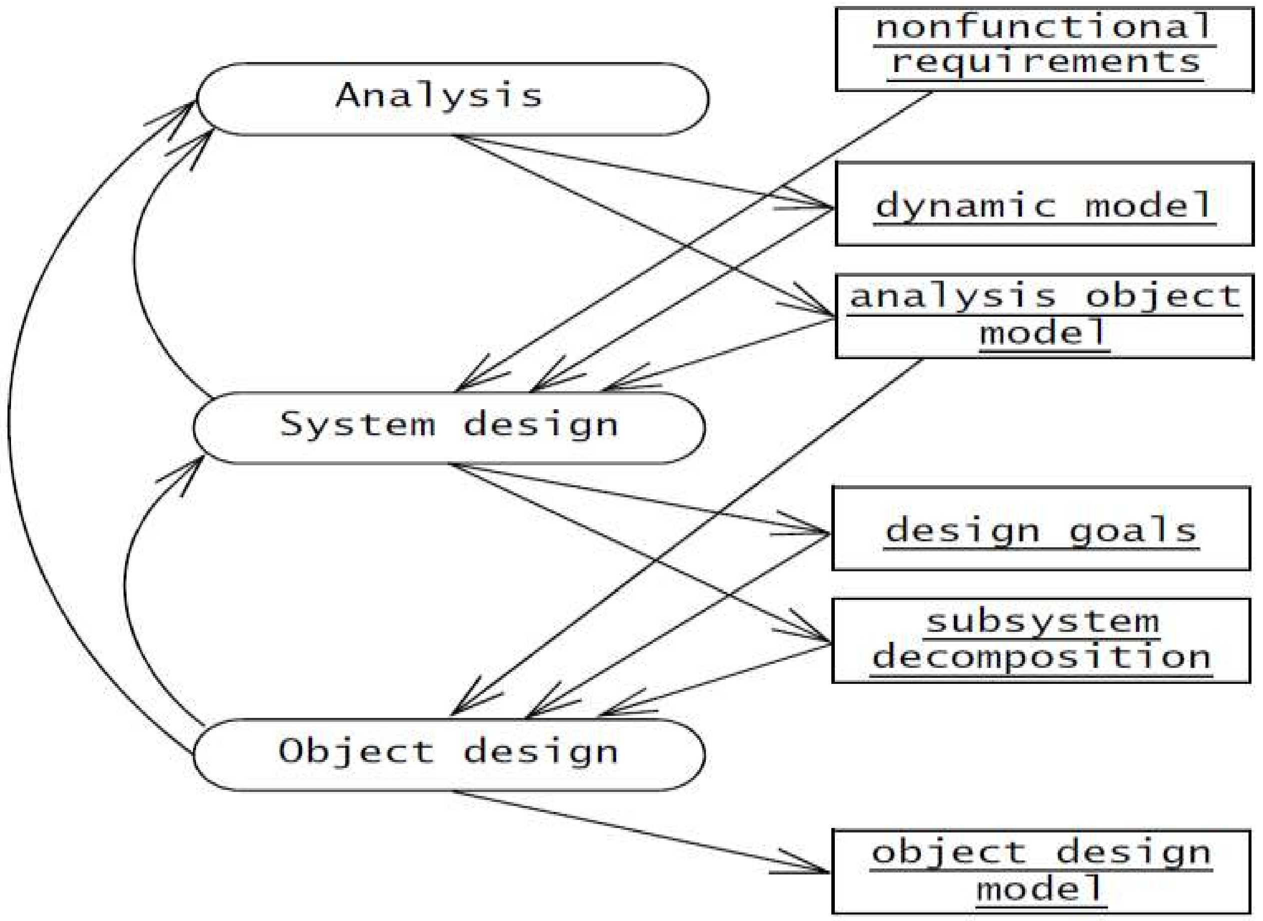
Analysis results in the requirements model described by the following products:	System design is the first step in this direction. System design results in the following products:
• A set of nonfunctional requirements and constraints, such as maximum response time, minimum throughput, reliability, operating system platform, and so on • A use case model, describing the system functionality from the actors' point of view • An object model, describing the entities manipulated by the system • a sequence diagram for each use case, showing the sequence of interactions among objects participating in the use case. • The analysis model describes the system completely from the actors' point of view and serves as the basis of communication between the client and the developers. • The analysis model, however, does not contain information about the internal structure of the system, its hardware configuration, or more generally, how the system should be realized.	• Design goals, describing the qualities of the system that developers should optimize • Software architecture, describing the subsystem decomposition in terms of subsystem responsibilities, dependencies among subsystems, subsystem mapping to hardware, and major policy decisions such as control flow, access control, and data storage • Boundary use cases, describing the system configuration, startup, shutdown, and exception handling issues.

- System design is the transformation of an analysis model into a system design model.
- During system design, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams.
- Developers also select strategies for building the system, such as the hardware/software strategy, the persistent data management strategy, the global control flow, the access control policy, and the handling of boundary conditions.
- The result of system design is a model that includes a subsystem decomposition and a clear description of each of these strategies.
- System design is not algorithmic. Developers have to make trade-offs among many design goals that often conflict with each other.

Major activities in system design

System design is decomposed into several activities, each addressing part of the overall problem of decomposing the system:

- *Identify design goals*. Developers identify and prioritize the qualities of the system that they should optimize.
- *Design the initial subsystem decomposition*. Developers decompose the system into smaller parts based on the use case and analysis models.
- Developers use standard architectural styles as a starting point during this activity.
- *Refine the subsystem decomposition to address the design goals*. The initial decomposition usually does not satisfy all design goals. Developers refine it until all goals are satisfied.


Analysis
System design
Object design
nonfunctional requirements
dynamic model
analysis object model
design goals
subsystem decomposition
object design model

System Design Concepts

Subsystem decompositions and their properties

- A subsystem is a set of classes and subsystems provide services to other subsystems.
- A **service** is a set of related operations that share a common purpose.
- **Coupling** measures the dependencies between two subsystems, whereas **cohesion** measures the dependencies among classes within a subsystem.
- Ideal subsystem decomposition should minimize coupling and maximize cohesion.
- **Layering** allows a system to be organized as a hierarchy of subsystems, each providing higher-level services to the subsystem above it by using lower-level services from the subsystems below it.
- **Partitioning** organizes subsystems as peers that mutually provide different services to each other

Subsystem and classes

- In order to reduce the complexity of the application domain, the smaller parts of the system called “classes” are identified and organized into packages.
- Similarly, to reduce the complexity of the solution domain, a system is decomposed into simpler parts, called “subsystems,” which are made of a number of solution domain classes.
- A **subsystem** is a replaceable part of the system with well-defined interfaces that encapsulates the state and behavior of its contained classes.
- A subsystem typically corresponds to the amount of work that a single developer or a single development team can tackle

Services and Subsystem Interfaces

- A subsystem is characterized by the services it provides to other subsystems. A **service** is a set of related operations that share a common purpose.
- A subsystem providing a notification service, for example, defines operations to send notices, look up notification channels, and subscribe and unsubscribe to a channel.
- The set of operations of a subsystem that are available to other subsystems form the **subsystem interface**.
- The subsystem interface includes the name of the operations, their parameters, their types, and their return values.
- System design focuses on defining the services provided by each subsystem, that is, enumerating the operations, their parameters, and their high-level behavior.
- Object design will focus on the **application programmer interface** (API), which refines and extends the subsystem interfaces.
- The API also includes the type of the parameters and the return value of each operation

Coupling and Cohesion

Coupling

- **Coupling** is the number of dependencies between two subsystems. If two subsystems are loosely coupled, they are relatively independent, so modifications to one of the subsystems will have little impact on the other.
- If two subsystems are strongly coupled, modifications to one subsystem is likely to have impact on the other.
- A desirable property of a subsystem decomposition is that subsystems are as loosely coupled as reasonable. This minimizes the impact that errors or future changes in one subsystem have on other subsystems

Cohesion

- **Cohesion** is the number of dependencies within a subsystem. If a subsystem contains many objects that are related to each other and perform similar tasks, its cohesion is high.
- If a subsystem contains a number of unrelated objects, its cohesion is low. A desirable property of a subsystem decomposition is that it leads to subsystems with high cohesion

Layers and Partitions

- A **hierarchical decomposition** of a system yields an ordered set of layers.
- A **layer** is a grouping of subsystems providing related services, possibly realized using services from another layer.
- Layers are ordered in that each layer can depend only on lower level layers and has no knowledge of the layers above it.
- The layer that does not depend on any other layer is called the bottom layer, and the layer that is not used by any other is called the top layer.
- In a **closed architecture**, each layer can access only the layer immediately below it. In an **open architecture**, a layer can also access layers at deeper levels

Architectural Styles

As the complexity of systems increases, the specification of system decomposition is critical. It is difficult to modify or correct weak decomposition once development has started, as most subsystem interfaces would have to change.

In recognition of the importance of this problem, the concept of **software architecture** has emerged. A software architecture includes system decomposition, global control flow, handling of boundary conditions, and inter subsystem communication protocols

- *Model/View/Controller* architectural style
- *Client/server* architectural style
- Peer-to-peer architectural style
- Pipe and filter architectural style

Conceptual and technical design

To transform requirements into a working system designers must satisfy both customers and the system builders. The customers should understand what the system it to do at the same time the system builders must understand how to do.

To accomplish these design is divided into two parts and is called as the two parts iterative process. The two parts are given below:

- Conceptual design or preliminary design.
- Technical design or detailed design.

First the conceptual design is produced that tells the customer exactly what the system will do. It beings to the what part of the solution. Once the customer approves it is translated into a much more detailed document i.e. the technical design which allows system builders to understand the actual hardware and software needed to solve the customers problem. It belongs to the how part of the solution.

Key references

Bernd Bruegge & Allen H. Dutoit - Object-Oriented Software Engineering: Using UML, Patterns, and Java

Thank You