SNS COLLEGE OF ENGINEERING

Kurumbapalayam (po), Coimbatore – 641 107
Accredited by NAAC-UGC with ‘A’ Grade
Approved by AICTE & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

Unit 3

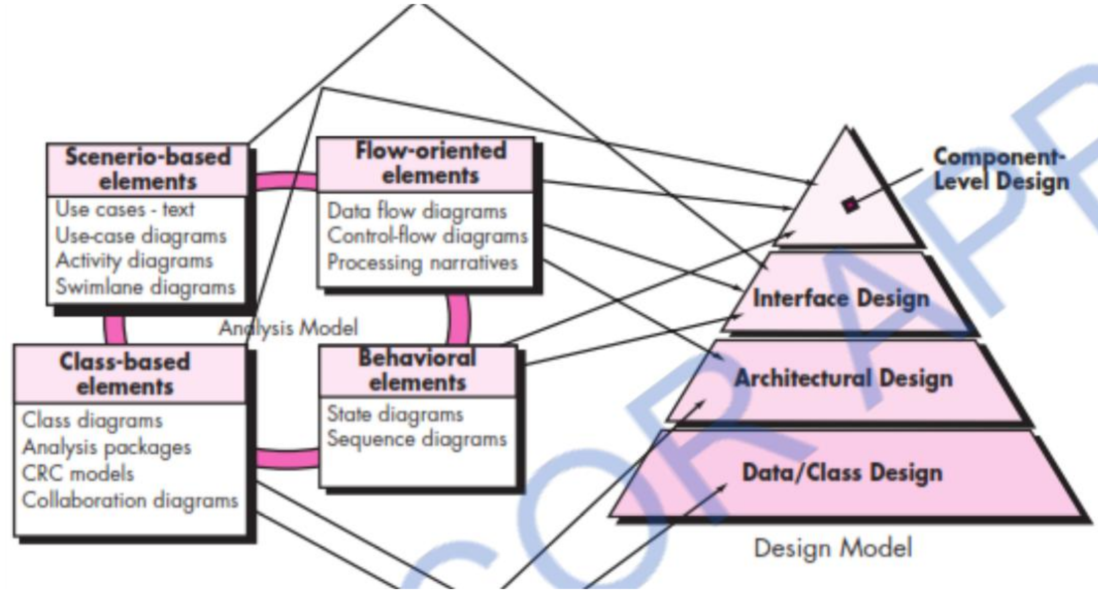
Introduction to Software Design

Software design is a **process** that translates the requirements of a system into a detailed plan or **blueprint** for building the software. This blueprint helps developers understand how the system should be structured and how all parts of the software will work together.

Key points to remember:

1. **Software Design is Important**: It involves applying principles, concepts, and best practices to ensure the software is of high quality. A good design is essential for building reliable, efficient, and maintainable software.
2. **Design Models**: The design process creates a **model** of the software, which includes details about **architecture**, **data structure**, **interfaces**, and **components**. These models guide the development team in creating the actual software.
3. **Technical Core**: Software design is a fundamental part of software engineering and is important no matter which development process or methodology is being used.
4. **Design Follows Requirement Analysis**: After the requirements of the software have been gathered and analyzed, design comes next. It is the final step before actual construction (coding and testing) begins.

Four Key Design Models for a Complete Software Design



When creating a software system, you need to address four important types of design models to ensure the system works correctly and efficiently. These models help turn the **requirement analysis** into a full design ready for development.

1. Data/Class Design

- **What it is**: This model focuses on how data is organized and how different **classes** (objects) are structured within the system. It converts the class models into **real implementations**.
- **How it works**: The **CRC (Class Responsibility Collaborator)** diagram helps define what each class will do, what data it holds, and how it interacts with other classes. This is the basis for defining the actual **data structures** and **attributes** in the design.
- **Example**: For a **banking system**, a class might be **Account** which contains attributes like **balance** and methods like **deposit()** or **withdraw()**.

2. Architectural Design

- **What it is**: This design model defines the **high-level structure** of the software. It describes how the major components of the system interact with each other and how the system will be structured overall.
- **How it works**: The architectural design includes the choice of **design patterns** (e.g., Model-View-Controller) and ensures that all parts of the system fit together. It also takes into account the **constraints** on the system, such as performance or security.
- **Example**: In a **web application**, the architectural design might include a **client-server** model where the client sends requests to a server, which processes the data and sends back the response.

3. Interface Design

- **What it is**: Interface design defines how different parts of the system will **communicate** with each other and with **users**.
- **How it works**: It focuses on how information flows between the system's components or with external systems. The interface should be easy for users to understand and interact with.
- **Example**: The **login page** of a web app is an **interface** between the user and the system. It defines how users will input their credentials and how the system will respond with feedback (success or error).

4. Component-Level Design

- **What it is**: This model focuses on the **detailed design** of each software **component** (i.e., the individual units that perform specific tasks).
- **How it works**: The component-level design takes the higher-level architectural design and breaks it down into **procedural descriptions** for each component. It also involves defining how each component will behave, based on the class models and flow models.
- **Example**: For the **Account** class mentioned earlier, the component-level design might specify the **withdraw()** method in detail: what steps are involved, how to check if there's enough balance, and how to update the account's balance.

Conclusion

To summarize, the design process in software engineering is about creating a **detailed plan** for building the software system. This plan is divided into four main models:

1. **Data/Class Design**: Organizing data and defining how classes work together.
2. **Architectural Design**: Structuring the system and defining how components interact.
3. **Interface Design**: Designing how the system communicates with users and other systems.
4. **Component-Level Design**: Detailing each individual component's functionality.

Each of these models helps guide the development process and ensures that the final product is well-structured, easy to understand, and able to meet the required functionality and quality standards.