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 ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Unit 3

DESIGN PROCESS

The **software design process** is the stage where the requirements for a software system are turned into a detailed plan, often referred to as a "blueprint" for building the system. This process is **iterative**, meaning it repeats over time, with each iteration refining and improving the design based on feedback and new insights. The goal is to ensure that the final software meets all the needs of the users, stakeholders, and technical requirements.

Key Goals of a Good Design

A **good software design** must follow certain guidelines to ensure the software is of high quality and can be effectively developed and maintained:

1. **Meeting All Requirements**:
    - The design must cover **both explicit and implicit requirements**. **Explicit requirements** are clear, written needs (like "The system must allow users to log in"). **Implicit requirements** might be things like **performance** or **security** that aren't always explicitly stated but are assumed by stakeholders.
2. **Readability and Understandability**:
    - The design should be **clear and easy to understand** for everyone involved in the project—**developers** who will code the system, **testers** who will check for bugs, and **support teams** who will help maintain it later.
3. **Complete Picture of the System**:

- The design should provide a full view of the system, covering **data** (how data is stored and used), **functionality** (what the software does), and **behavior** (how the system reacts to different inputs or conditions). This gives a comprehensive understanding of how everything works.

Quality Guidelines for Good Design

Good software design also follows some **quality guidelines** to ensure it is efficient, scalable, and easy to maintain:

1. **Architecture**:
   - The software's architecture should use recognized **patterns** or **styles** that are proven to work.
   - The components of the design should be well thought-out and fit together in a way that makes it easy to implement and test.
2. **Modularity**:
   - The design should divide the software into smaller, **logical parts** (modules or subsystems) that can be developed, tested, and maintained independently.
3. **Clear Representation**:
   - The design should clearly separate **data**, **architecture**, **interfaces**, and **components**, making it easy to understand how each part of the system fits together.
4. **Data Structures**:
   - The design should include **appropriate data structures** (e.g., arrays, lists, trees) that match the needs of the system and follow standard patterns that are well understood.
5. **Independent Components**:
   - Each component of the system should have **independent functionality**, meaning they can work on their own without relying too heavily on others.
6. **Simplified Interfaces**:
   - The design should create **simple interfaces** that reduce complexity, making it easier to connect different parts of the system and integrate with external systems.

7. **Repeatable Process**:
    - The design process should be **repeatable**, meaning that it follows a consistent method, informed by the requirements analysis, to ensure the software is built in a structured and predictable way.
8. **Effective Communication**:
    - The design should be documented using **notations and diagrams** (like UML) that clearly communicate the system's structure, behavior, and interactions to all stakeholders.

Quality Attributes (FURPS)

The **FURPS** acronym stands for five **key quality attributes** that software design should aim to achieve. These are essential targets for designing high-quality software:

1. **Functionality**:
    - Focuses on whether the software has all the features it needs. Is the system **secure**, and does it deliver the right functions as expected?
    - **Example**: A banking app that allows users to check balances, make transfers, and secure logins.
2. **Usability**:
    - Measures how **user-friendly** and **intuitive** the software is. Is it easy for people to use, and does it look good? Does it have good documentation?
    - **Example**: An online shopping website that is easy to navigate and understand, with clear instructions and helpful support.
3. **Reliability**:
    - Measures how often the software **fails** and how **predictable** and **accurate** it is. Can it recover from failures, and is the software dependable?
    - **Example**: A flight booking system that rarely crashes, provides correct booking information, and can handle failures gracefully (like offering retries or fixes when things go wrong).

4. **Performance**:
    - Refers to how **fast** the software works, how **efficiently** it uses resources (like memory or CPU), and how it handles large volumes of users or data.
    - **Example**: A social media platform that loads images and posts quickly even when many users are active at the same time.
5. **Supportability**:
    - Describes how easy it is to **maintain**, **update**, and **fix** the software. This includes **extensibility** (can we add new features?), **testability** (how easy it is to test), and **compatibility** (can the software work on different systems or devices?).
    - **Example**: A content management system (CMS) that allows new features to be added easily and integrates well with different web platforms.

Consideration of Quality Attributes During Design

It's important to keep these quality attributes in mind **from the very beginning of the design process**, not after the software is built. Different projects may prioritize different attributes, such as:

- An application may focus on **functionality** and security.
- A game might emphasize **performance** and responsiveness.
- A medical system might focus heavily on **reliability** to ensure data accuracy and uptime.

By considering these quality attributes early, you ensure that the software meets the stakeholders' needs and performs well across all aspects.