**Lec#05**

**1. Software Design Methodologies (SDM):**

A **Software Design Methodology (SDM)** is a set of procedures used to create a **conceptual design** for fulfilling software requirements.

* + It depends on factors like:
    - **Type of software** (e.g., standalone, distributed).
    - **Scope of the project** (e.g., new system, revamp of an existing system).
    - **Resource constraints** (e.g., time, money, expertise).

**2. Common Software Design Approaches:**

**i). Structured (Function-Oriented) Design:**

* **Focus**: Procedures and functions.
* **Key Features**:
  + Decomposes the system into **processes and functions**.
  + Uses tools like **DFD (Data Flow Diagrams)** and **Flowcharts**.
  + Separates **data** from **procedures**.
* **Example**: **Inventory Management System**.
  + Functions like adding items, updating quantities, and generating reports are treated as separate processes.
* **Drawbacks**:
  + **Interdependencies** between functions.
  + **Reusability** is limited because data is not attached to functions.

**ii) Object-Oriented Design (OOD)**

* **Focus**: Objects that encapsulate **data** and **behavior**.
* **Key Principles**:
  + **Encapsulation**: Hiding internal details of objects.
  + **Inheritance**: Reusing code from parent classes.
  + **Polymorphism**: Allowing objects to take multiple forms.
* **Tools**: **Class Diagrams**, **Domain Models**, **Sequence Diagrams**.
* **Example**: **Online Shopping Platform**.
  + Entities like **products**, **customers**, and **orders** are modeled as objects with properties (e.g., price) and behaviors (e.g., adding to cart).
* **Benefits**:
  + Easier to manage and maintain.
  + Existing classes can be reused in future designs.
  + Models the real world better (everything is an object).

**iii) Data-Oriented (Data-Structure-Centered) Design**

* **Focus**: Organizing and processing **data efficiently**.
* **Key Features**:
  + Identifies **entities** and their **relationships**.
  + Uses tools like **ERD (Entity-Relationship Diagrams)**, **DFD**, and **Data Dictionaries**.
* **Example**: **Banking System**.
  + Entities like **customers**, **accounts**, and **transactions** are identified, and their relationships (e.g., a customer having multiple accounts) are modeled.
* **Benefits**:
  + Efficient **data storage** and **retrieval**.
  + Ensures **data integrity** and **security**.

**iv) Component-Based Design**

* **Focus**: Dividing the system into **reusable components**.
* **Key Features**:
  + Software is built using **predefined, reusable components**.
  + Promotes **modularity**, **reusability**, and **efficiency**.
* **Tools**: **Component-Based Diagrams**.
* **Example**: **Content Management System (CMS)**.
  + Components like **user authentication**, **content creation**, and **content management** are implemented as separate modules.
* **Benefits**:
  + Components can be reused across different systems.
  + Easy to add or remove components.

**v) Formal Methods**

* **Focus**: Using **mathematical techniques** to specify, develop, and verify software.
* **Key Features**:
  + Relies on **logic**, **set theory**, and **automata theory**.
  + Ensures **high reliability** and **correctness**, especially in **critical systems** (e.g., aerospace, medical devices).
* **Example**: **Air Traffic Control System**.
  + Mathematical models are used to verify the correctness of critical algorithms and protocols.
* **Benefits**:
  + Reduces ambiguity.
  + Eliminates design errors before implementation.
* **Drawbacks**:
  + **Expensive** to implement.

**3. Software Design Paradigms**

**1. Structured Design (Function-Oriented)**

* **Focus**: **Procedures** and **functions**.
* **Approach**: **Top-down** decomposition of the system into processes and functions.
* **Drawbacks**:
  + **Interdependencies** between functions.
  + **Reusability** is limited because data is not attached to functions.

**2. Object-Oriented Design (OOD)**

* **Focus**: **Objects** that encapsulate data and behavior.
* **Approach**: **Bottom-up** design, where smaller objects are combined to form a larger system.
* **Key Elements**:
  + **Objects**: Instances of classes.
  + **Classes**: Blueprints for objects.
  + **Encapsulation**: Hiding internal details.
  + **Inheritance**: Reusing code from parent classes.
  + **Polymorphism**: Allowing objects to take multiple forms.
* **Benefits**:
  + Easier to manage and maintain.
  + Existing classes can be reused.
  + Models the real world better.

**Lec#06**

**1. Requirements Fundamentals**

* **What are Requirements?**
  + **Definition**: A specification of what should be implemented. It describes how the system should behave, its properties, or constraints.
  + **Purpose**:
    - **Clear Goals**: Helps developers understand what to build.
    - **Avoid Miscommunication**: Ensures all stakeholders (clients, developers, testers) are aligned.
    - **Quality Assurance**: Helps verify and validate the system.
    - **Legal & Standard Compliance**: Ensures the system follows regulations.
* **Need vs Want**:
  + **Need**: Something you **must have** (essential).
  + **Want**: Something you **would like to have** (optional).

**2. Types of Requirements**

**1. User Requirements**

* **What?**: Statements in **natural language** (plus diagrams) about what services the system should provide to users and the constraints under which it must operate.
* **Example (Online Banking System)**:
  + "The system shall allow users to transfer money between their accounts."
  + "Users should be able to log in securely using a username and password."

**2. System Requirements**

* **What?**: Detailed descriptions of the system’s functions, services, and operational constraints.
* **Example (Online Banking System)**:
  + "The system shall authenticate users using a SHA-256 hashed password stored in the database."
  + "Funds transfer requests shall be processed within 5 seconds and use HTTPS encryption."

**3. Levels of Requirements**

**1. Business Requirements**

* **What?**: High-level objectives of the organization or customer requesting the system.
* **Example**:
  + **Objective**: Increase online sales by 20% in 6 months.
  + **Stakeholders**: Marketing team, IT team, customers.
  + **Scope**: Improve app UI, add a recommendation system, streamline checkout.
  + **Constraints**: Budget of $50,000, implementation within 3 months.

**2. User Requirements**

* **What?**: Statements of what services the system should provide to users and the constraints under which it must operate.
* **Example**:
  + "The system shall allow customers to search for products by name, category, and price."
  + "Users should be able to add items to a shopping cart and proceed to checkout."

**3. Functional Requirements**

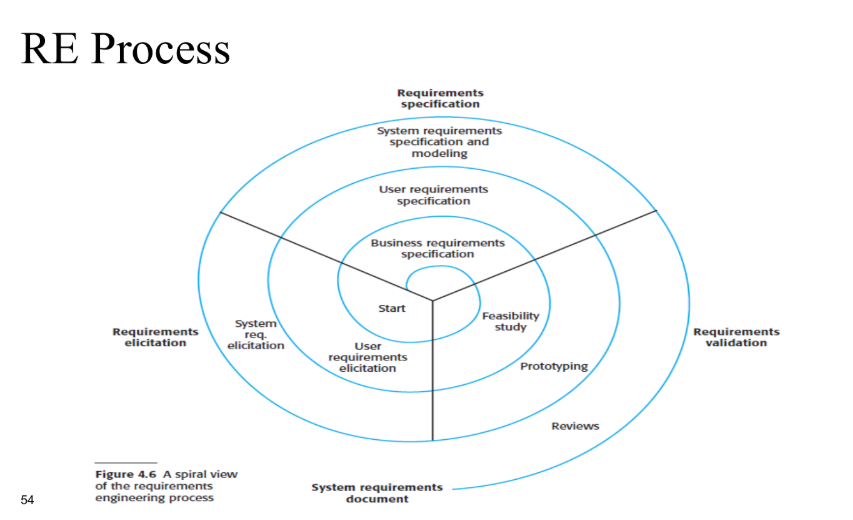
* **What?**: Detailed descriptions of the system’s functions, inputs, outputs, and exceptions.
* **Example (Mental care System)**:
  + "A user shall be able to search the appointments lists for all clinics."
  + "The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day."

**4. Non-Functional Requirements**

* **What?**: Constraints on the services or functions offered by the system.
* **Types**:
  + **Product Requirements**: Performance, reliability, usability, etc.
  + **Organizational Requirements**: Development process, standards, etc.
  + **External Requirements**: Regulatory, legislative, ethical, etc.
* **Example**:
  + "The system shall allow 100,000 hits per minute on the website."
  + "The system shall not have downtime of more than one second for continuous execution of 1,000 hours."

**4. Requirements Engineering (RE) Process**

* **What?**: The process of finding out, analyzing, documenting, and checking the services and constraints of a system.
* **Steps**:
  1. **Elicitation**: Gather requirements from stakeholders.
  2. **Analysis**: Understand and refine the requirements.
  3. **Specification**: Document the requirements clearly.
  4. **Validation**: Check if the requirements meet stakeholder needs.
  5. **Management**: Handle changes to requirements over time.
* **Challenges**:
  1. Stakeholders don’t know what they really want.
  2. Conflicting requirements from different stakeholders.
  3. Requirements change during the analysis process



**Tasks**

**Task 1: Write 5 functional and 5 non-functional requirements for a mobile telephone.**

* **Functional Requirements**:
  1. The phone shall allow users to make and receive calls.
  2. The phone shall support sending and receiving text messages.
  3. The phone shall have a camera for taking photos.
  4. The phone shall allow users to browse the internet.
  5. The phone shall support installing third-party apps.
* **Non-Functional Requirements**:
  1. The phone shall have a battery life of at least 24 hours on a single charge.
  2. The phone shall be water-resistant up to 1 meter for 30 minutes.
  3. The phone shall support 5G connectivity.
  4. The phone shall have a screen resolution of at least 1080p.
  5. The phone shall weigh less than 200 grams.

**Task 2: Write 5 functional and 5 non-functional requirements for an online examination system.**

* **Functional Requirements**:
  1. The system shall allow students to take exams online.
  2. The system shall automatically grade multiple-choice questions.
  3. The system shall allow instructors to upload exam questions.
  4. The system shall provide a timer for each exam.
  5. The system shall generate a report of exam results for each student.
* **Non-Functional Requirements**:
  1. The system shall handle up to 10,000 concurrent users without performance degradation.
  2. The system shall be accessible 24/7 with 99.9% uptime.
  3. The system shall encrypt all exam data to ensure security.
  4. The system shall be compatible with all major web browsers.
  5. The system shall load exam questions within 2 seconds.