Software Engineering and Object-Oriented Analysis & Design

Instructor: Bhanu Bhakta Joshi

Topic 1: Software process and requirements

❖Software characteristics

- **Functionality:** Requirements and Specifications
- **Usability:** Easy to use and understand.
- **Reliability:** Free of defects and performs consistently and accurately under different conditions and scenarios.
- **Performance:** Runs efficiently and quickly handling large amounts of data or traffic.
- **Scalability:** Handle increasing workload and can be extended to meet the changing Requirements.
- **Testability:** Easy to test and validate, and it has a comprehensive test coverage.

- **Security:** Unauthorized access and safety from malicious attacks.
- **Maintainability:** Easy to change and update. Must be well-documented, so that it can be understood and modified by other developers.
- **Reusability:** Reused in other projects or applications

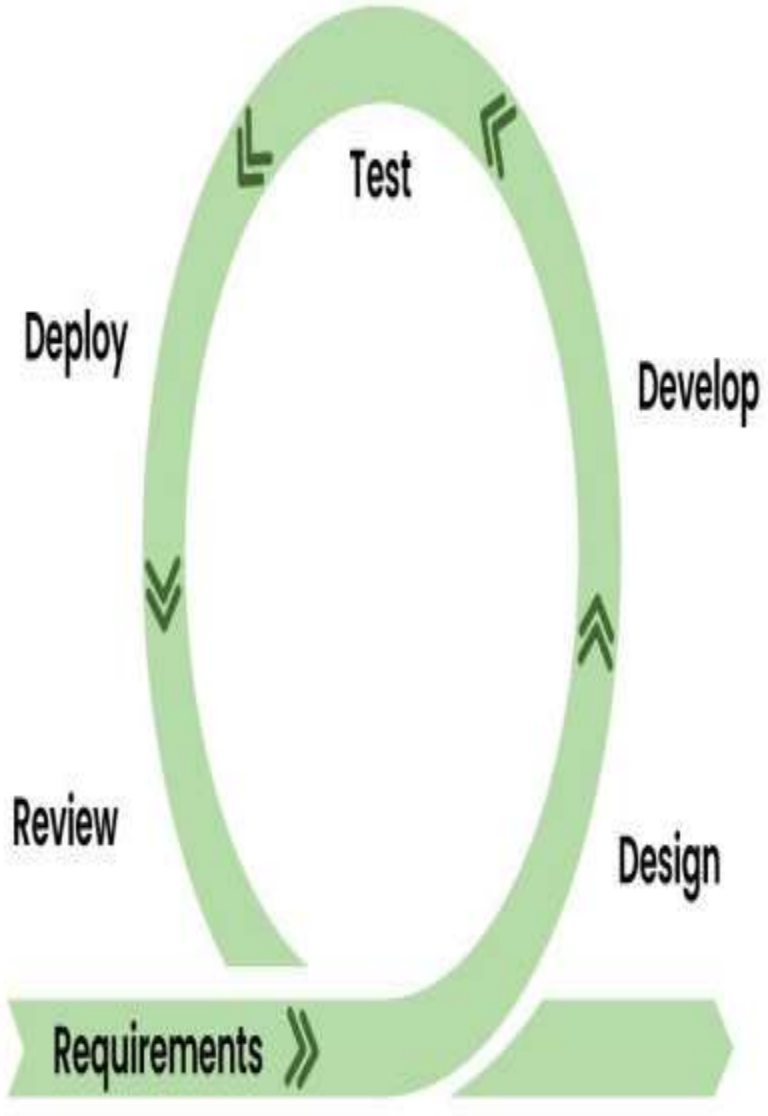
❖ Software quality attributes

- **Portability:** Transportable
- **Efficiency:** CPU time, memory and disk usage, response time.
- **Correctness:** SRS document is properly enforced.

❖ Software process model

1. Agile Model

- Combination of iterative and incremental models.
- Adaptability and Customer satisfaction.
- Created mainly to make changes in the middle of software development so that the software project can be completed quickly.
- Customer representative .
- Demo of working software is given to understand the customer’s requirements.
- Delivery of Incremental versions of the software.



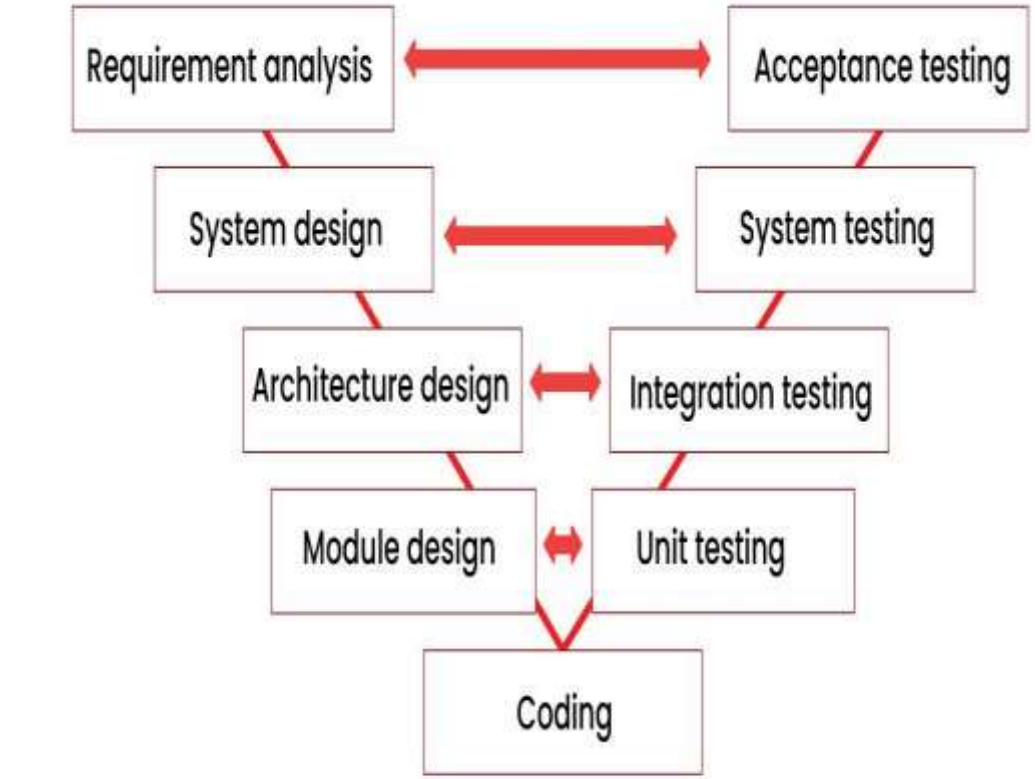
Agile Model

- **Agile has the following models:**

1. Scrum: Scrum master.
2. Crystal methods: Different projects require different processes.
3. DSDM: Dynamic system Development Method.
4. Feature driven development (FDD):Blends several industry-recognized best practices into a cohesive whole.
5. Lean software Development: Focuses on optimizing efficiency by eliminating waste, enhancing learning, making decisions as late as possible.
6. Extreme programming (xp):Key practices include pair programming, test-driven development, frequent releases in short development cycles.

2. V Model

- Also called Verification and Validation Model.
- The execution of each process is sequential.
- Software development and testing activities take place at the same time.
- V-Design the left side represents the development activity, the right side represents the testing activity.



V-Model

Key Characteristics of the V-Model:

1. Sequential and Rigid.
2. Verification and Validation at Each Step.
3. Early Detection of Defects.
4. No Overlapping Phases.

3. Iterative Model

- Started with some software specifications and the first version of the software is developed.
- After first version if there is a need to change the software, then a new version of the software is created with a new iteration.
- The Iterative Model allows accessing earlier phases, in which the variations can be made.
- Iterative model is used when requirements are defined clearly, easy to understand, large application and requirement of changes in future.

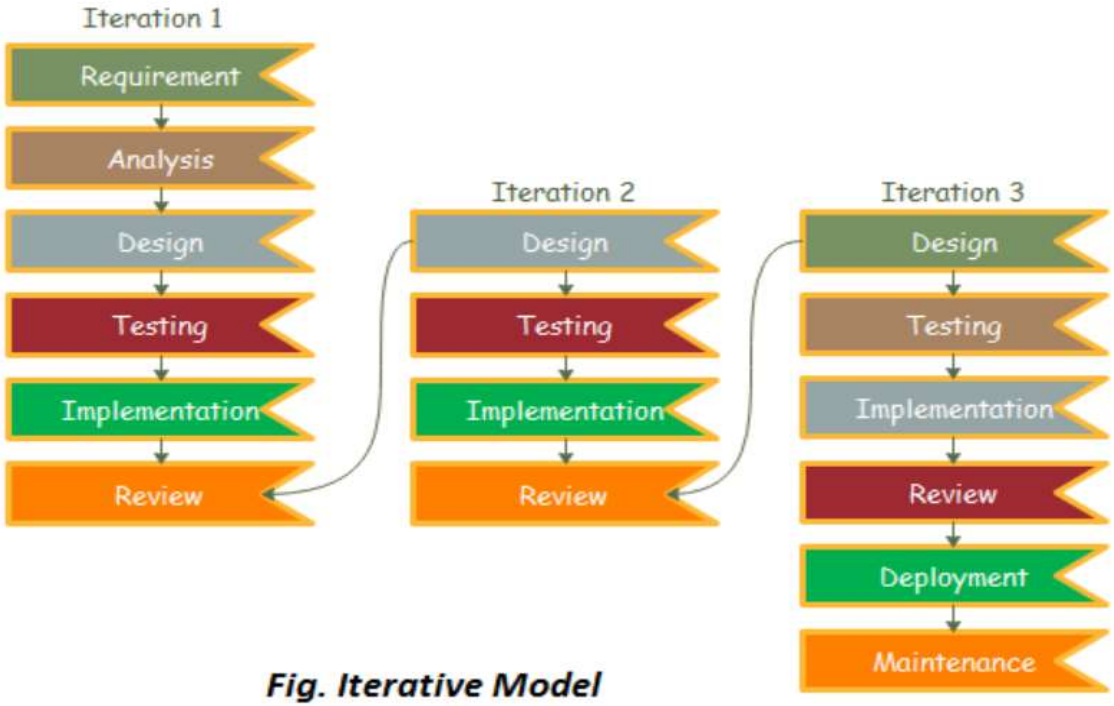


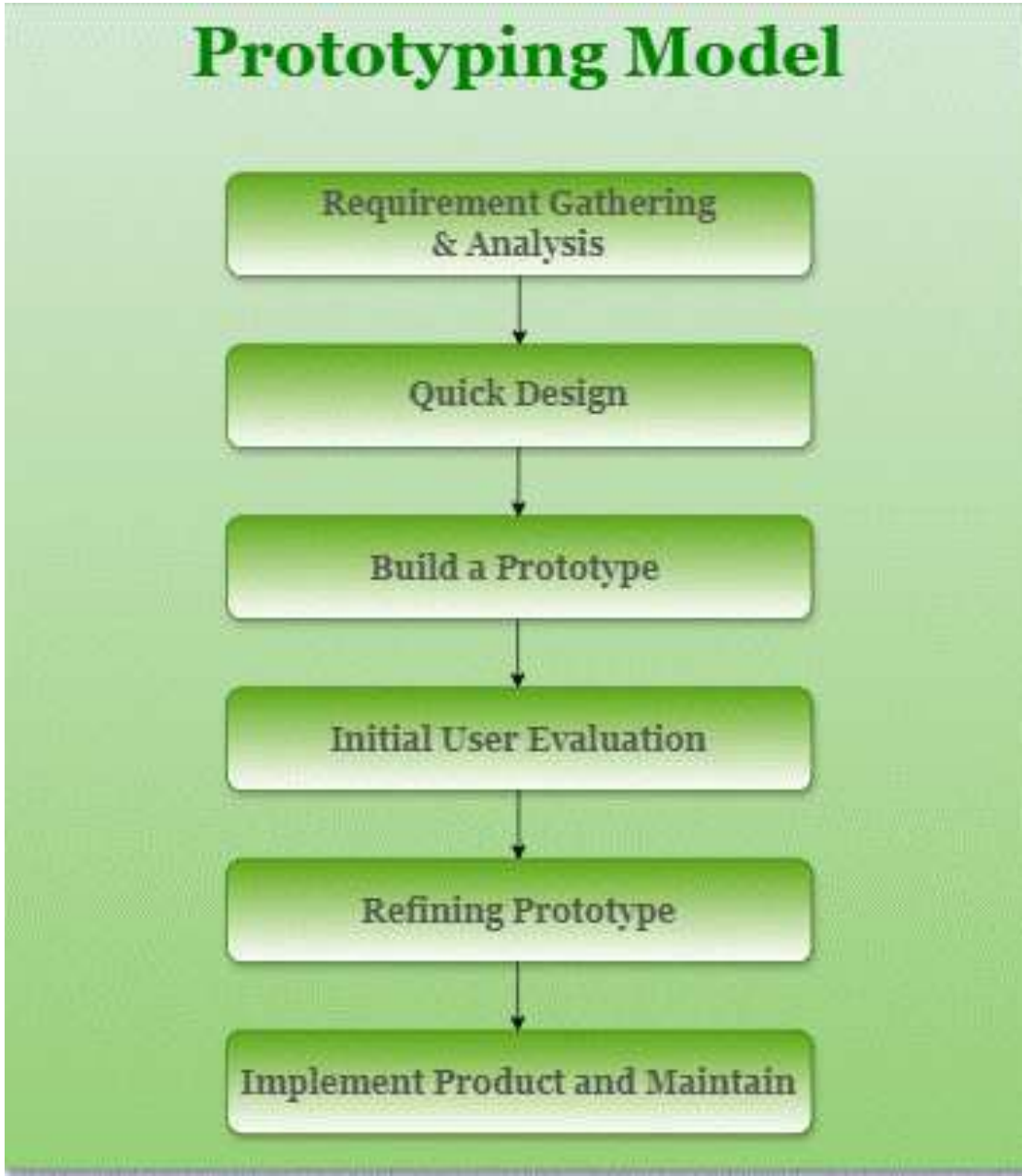
Fig. Iterative Model

Key Features of the Iterative Model

1. Repetition of Development Phases
2. Incremental Delivery
3. Flexibility to Change
4. Early and Continuous Feedback
5. Risk Management
6. Frequent Testing
7. Progressive Refinement

4. Prototype Model

- A prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.
- Used when the customers do not know the exact project requirements beforehand.



Key features of Prototype model

- Early Prototype Development
- User Involvement and Feedback
- Iterative Refinement
- Early Detection of Issues

5.Big Bang Model

- Do not follow any specific process**.**
- Begins with the necessary resources and efforts.
- The result may or may not be as per the customer's requirement because in this model the customer requirements are not defined.
- Ideal for small projects like academic projects or practical projects**.**
- Used when the size of the developer team is small, requirements are not defined and the release date is not confirmed.

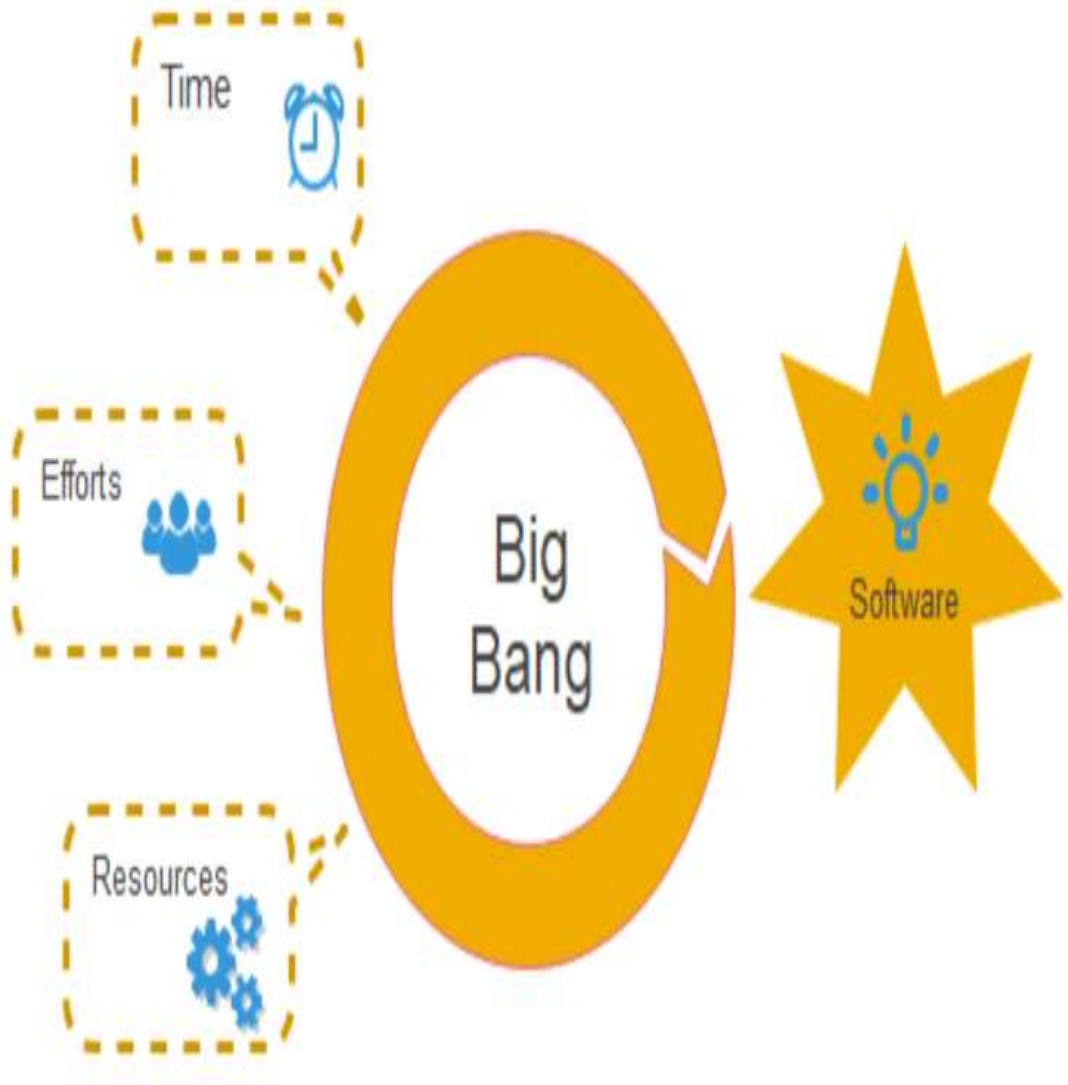


Fig. Big Bang Model

Key features of big bang model

- Single Development Phase
- No Detailed Planning
- All-in-One Release
- Late Testing and Debugging
- High Risk of Failure

❖ Computer-aided software engineering

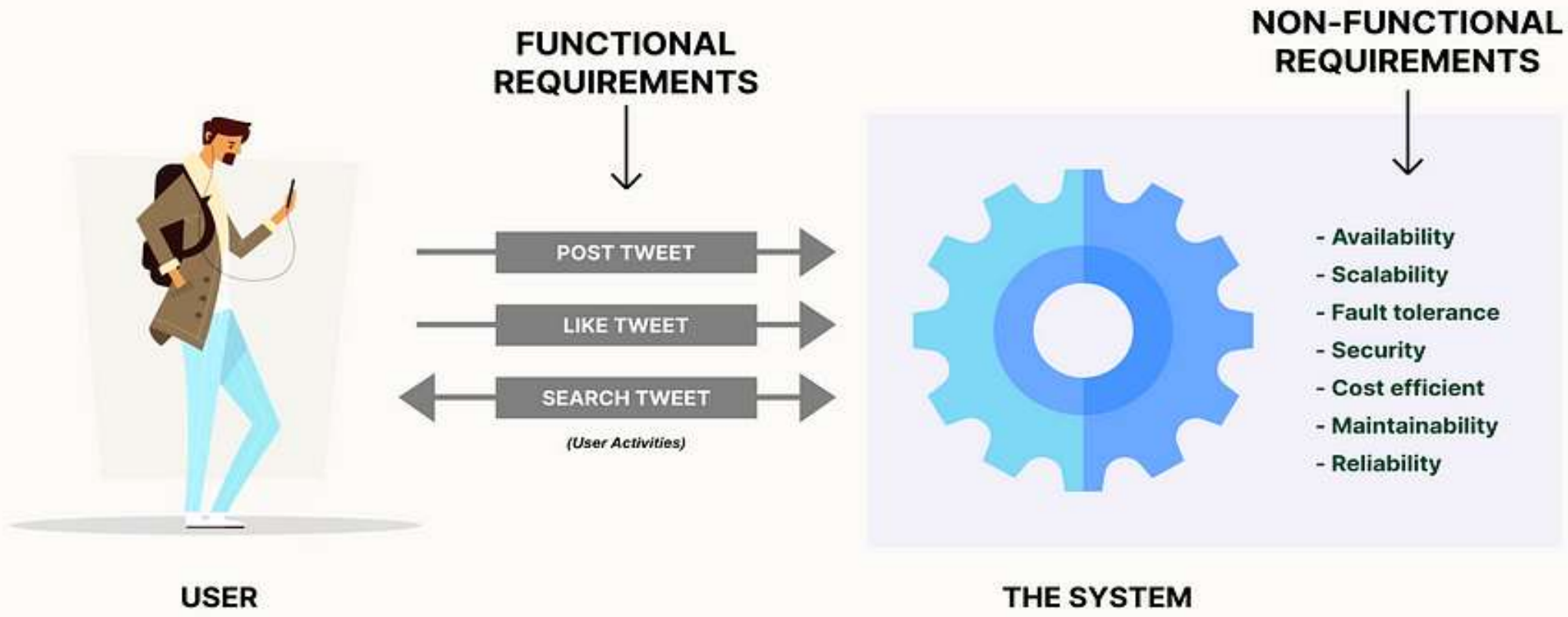
- Implementation of computer-facilitated tools and methods in software development.
- Ensures high-quality and defect-free software.
- Helps designers, developers, testers, managers and others to see the project milestones during development.
- CASE illustrates a wide set of labor-saving tools that are used in software development.
- The essential idea of CASE tools is that in-built programs can help to analyze developing systems that enhances quality and provide better outcomes.

Types of CASE Tools

- **Diagramming Tools:** Diagrammatic and graphical representations of the data and system processes, represents system elements, control flow and data flow among different software components and system structures in a pictorial form. For example, Flow Chart Maker tool for making state-of-the-art flowcharts.
- **Computer Display and Report Generators:** These help in understanding the data requirements and the relationships involved.
- **Analysis Tools:** It focuses on inconsistent, incorrect specifications involved in the diagram and data flow. It helps in collecting requirements, automatically check for any irregularity, imprecision in the diagrams, data redundancies, or erroneous omissions.
- **Central Repository:** It provides a single point of storage for data diagrams, reports, and documents related to project management.

- **Documentation Generators:** helps in generating user and technical documentation as per standards. It creates documents for technical users and end users. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.
- **Code Generators:** It aids in the auto-generation of code, including definitions, with the help of designs, documents, and diagrams.
- **Tools for Requirement Management:** It makes gathering, evaluating, and managing software needs easier.
- **Tools for Analysis and Design**: It offers instruments for modelling system architecture and behavior, which helps throughout the analysis and design stages of software development.
- **Tools for Database Management:** It facilitates database construction, design, and administration.
- **Tools for Documentation:** It makes the process of creating, organizing, and maintaining project documentation easier.

❖ Functional and non-functional Requirements



Functional requirements

- Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms.
- All these functionalities need to be necessarily incorporated into the system as a part of the contract.
- These are represented or stated in the form of input to be given to the system, the operation performed, and the output expected.
- **Example:**
- What are the features that we need to design for this system?
- What are the edge cases we need to consider, if any, in our design?

Non-Functional requirements

- These are the quality constraints that the system must satisfy according to the project contract.
- The priority or extent to which these factors are implemented varies from one project to another.
- They are also called non-behavioral requirements.
- Non-functional requirements specify the software's quality attribute.
- They ensure a better user experience, minimizes the cost factor.
- Basic non-functional requirements are - usability, reliability, security, storage, cost, flexibility, configuration, performance, legal or regulatory requirements, etc.

- **Execution qualities** like security and usability, which are observable at run time.
- **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.
- **Example:**

➢Each request should be processed with the minimum latency?

➢System should be highly valuable.

❖ User requirements

- These requirements describe what the end-user wants from the software system.
- User requirements are usually expressed in natural language.
- Gathered through interviews, surveys, or user feedback.
- Characteristics of Good User Requirements:

➢Clear and Unambiguous

➢Complete

➢Consistent

➢Verifiable

➢Prioritized

➢Feasible

➢Traceable

❖ System requirement

- These requirements specify the technical characteristics of the software system, such as its architecture, hardware requirements, software components, and interfaces.
- System requirements are typically expressed in technical terms and are often used as a basis for system design.

❖ Interface specification

- A point where two systems ,subjects, organization's etc. meet and interact.
- Interface=system/environment.
- A specification is an agreement between the produce of the services consumer of that services.
- There are 4 types of interface specification:

➢ Procedural interface : Calling existing program by new program(API).

➢ Data structures : Passed from one subsystem to another. Graphical data model.

➢ Representation of data : Ordering of bits. Common in embedded and real time system.

➢ Message passing interface : Sub system requesting service from another sub system. Used in Distributed system and parallel processing.

Interface specification Cycle

❖ The software requirements documents

• Complete specification and description of requirements of the software.

SRS parts:

1. Introduction

• **Purpose:** Describe the SRS's purpose and target audience.

• **Scope:** Outline the software's scope and main functionalities.

• **Definitions, Acronyms, Abbreviations:** Define terms used.

• **References:** List relevant documents and resources.

• **Overview:** Briefly describe the document structure.

2. Overall Description

- **Product Perspective:** Context and relation to other systems.
- **Product Functions:** Key functions the software will perform.
- **User Characteristics:** Typical user profiles and expertise levels.
- **Constraints:** Design or implementation limitations.
- **Assumptions and Dependencies:** Assumptions made and external dependencies

3. Specific Requirements

- **Functional Requirements:** Detailed functions, often using use cases.
  - **Example Use Case: User Login**
    - **Primary Actor:** User
    - **Preconditions:** User must be registered.
    - **Postconditions:** User is logged in.
    - **Main Scenario:** Steps for a successful login.
    - **Extensions:** Handling invalid credentials.
- **Non-Functional Requirements:** Criteria like performance, usability, security.
  - **Performance:** Response time, throughput.
  - **Security:** Authentication, data protection.
  - **Usability:** User interface design.

4. External Interface Requirements

- **User Interfaces:** Look and feel of the interface.
- **Hardware Interfaces:** Interactions with hardware.
- **Software Interfaces:** Interactions with other software.
- **Communication Interfaces:** Communication protocols.

5. System Features

- Detailed description of system features, their purpose, and behavior.

6.Other Requirements

- **Data Management:** Data models and database requirements.
- **Operational Requirements:** Operating environment and support needs.
- **Legal and Regulatory Requirements:** Compliance requirements.

7. Appendices

- Additional information like glossaries, diagrams, and supporting documents.

❖ Requirement's elicitation and analysis

- Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development.
- There are various ways to discover requirements:

➢ **Interviews**

➢ **Surveys**

➢ **Questionnaires**

➢ **Task analysis**

➢ **Domain Analysis**

➢ **Brainstorming**

➢ **Prototyping**

➢ **Observation**

❖ Requirement’s validation and management.

- Requirements validation techniques are essential processes used to ensure that software requirements are complete, consistent, and accurately reflect what the customer wants.
- Helps to identify and fix issues early in the development process.
- reduces the risk of costly errors.

- Different type of test to check the requirements.

1.Completeness checks

2.Consistency checks

3.Validity checks

4.Realism checks

5.Ambiguity checks

6.Variability

Requirement validation techniques:

1.Test Case Generation

2.Prototyping

3.Requirement reviews

4.Automated Consistency Analysis: CASE tool is used

5. Walk throughs

6.Simulation

7.Checklists for Validation

THANK YOU