Parul® University

Software Engineering

(303105253)

Requirements Engineering

❖ Requirements Engineering in software engineering is a crucial process that focuses on eliciting, documenting, analyzing, and validating requirements for software systems.

❖ It is a systematic approach to defining what the software should do and how it should behave, ensuring that the final product meets the needs and expectations of stakeholders.

❖ The key aspects and activities involved in Requirements Engineering:

Image source : Google

Problem Recognition

❖ Problem recognition in Requirements Engineering is the initial phase.

❖ This phase is critical because it sets the stage for the entire RE process by clarifying the goals and scope of the software project.

Here's how problem recognition typically unfolds in requirement engineering.

Identifying Stakeholders: The process begins by identifying all stakeholders who are involved or impacted by the software system. Stakeholders can include end-users, customers, managers, domain experts, regulatory bodies, and others who have a vested interest in the software.

Image source : Google

Cont….

Understanding Current Issues: Analysts and stakeholders collaborate to understand the current problems, deficiencies, or opportunities that require addressing through the software system. This involves gathering feedback, conducting interviews, and analyzing existing systems or processes.

Gathering Requirements: Based on the identified issues or opportunities, requirements are elicited from stakeholders. This involves capturing both functional requirements and non-functional requirements.

Defining Goals and Objectives: Clear goals and objectives for the software project are established. These goals should be specific, measurable, achievable, relevant, and time-bound providing a clear direction for the development effort.

Image source : Google

Cont….

Prioritizing Requirements: Requirements are prioritized based on their importance to stakeholders and their impact on achieving project goals. This helps in focusing development efforts on the most critical and high-value features.

Identifying Constraints and Assumptions: Analysts also identify any constraints and assumptions that could affect the development and implementation of the software system.

Documenting the Problem Statement: The culmination of problem recognition is often a formal document known as the Problem Statement or Project Initiation Document. This document outlines the context of the project, the identified issues or opportunities, the goals and objectives, and the initial set of requirements to be addressed.

Image source : Google

Requirement Engineering tasks

Requirements Elicitation: This involves gathering requirements from stakeholders. Techniques such as interviews, workshops, surveys, and observations are used to capture the needs and expectations of users, customers, and other relevant parties.

Requirements Analysis: Once requirements are collected, they need to be analyzed to ensure they are clear, complete, consistent, and feasible. This involves refining requirements, identifying dependencies, and resolving conflicts or ambiguities.

Requirements Specification: Documenting requirements in a formal and structured manner. This includes creating artifacts such as use cases, user stories, functional and non-functional requirements, system models, and prototypes to describe what the software should do and how it should behave.

Image source : Google

Cont….

Requirements Validation: Ensuring that the specified requirements meet the needs of stakeholders and are technically feasible. Validation involves checking requirements for correctness, consistency, and completeness through techniques such as reviews, prototyping, simulations, and demonstrations.

Requirements Management: Managing changes to requirements throughout the software development lifecycle. This involves maintaining traceability of requirements, establishing baselines, controlling changes, and ensuring that all stakeholders are informed about the status and evolution of requirements.

Image source : Google

Cont….

Requirements Communication: Effectively communicating requirements to all stakeholders, including developers, testers, project managers, and customers. Clear and unambiguous communication helps ensure that everyone understands what needs to be built and how it should function.

Requirements Negotiation: Resolving conflicts and prioritizing requirements when stakeholders have different needs or priorities. Negotiation ensures that compromises are made when necessary to achieve consensus and balance competing interests.

Requirements Traceability: Establishing and maintaining traceability links between requirements and other artifacts such as design documents, test cases, and implemented code. Traceability helps ensure that all requirements are properly implemented and tested.

Image source : Google

Cont….

Requirements Documentation Management: Managing the documentation of requirements throughout their lifecycle, ensuring that documents are kept up-to-date, accessible, and organized. This includes version control and ensuring that changes to requirements are properly documented and communicated.

Requirements Evolution: Recognizing that requirements may change over time due to various factors such as new insights, stakeholder feedback, or changes in the business environment. Managing requirements evolution involves being responsive to changes while maintaining project stability and alignment with project goals.

Image source : Google

Requirement Engineering Processes

The Requirements Engineering process is a systematic approach used in software engineering to ensure that software systems meet the needs and expectations of stakeholders.

It involves several interconnected activities that are performed iteratively throughout the software development lifecycle.

1. **Requirements Elicitation**

   Goal: Gather and collect requirements from stakeholders.

   Activities: Conduct interviews, workshops, surveys, and observations to understand stakeholders' needs and expectations.

   Outputs: Requirements elicitation document, stakeholder requirements, user stories, use cases, etc.

Image source : Google

Cont…

2. Requirements Analysis

Goal: Analyze and refine gathered requirements for clarity, completeness, consistency, and feasibility.

Activities: Review and prioritize requirements, identify dependencies, and resolve conflicts or ambiguities.

Outputs: Refined requirements document, prioritized requirements list, requirements models (e.g., use case diagrams, activity diagrams).

Image source : Google

Cont…

3. Requirements Specification

Goal: Document requirements in a clear, unambiguous, and structured manner.

Activities: Write detailed descriptions of functional and non-functional requirements, create system models (if applicable), and validate with stakeholders.

Outputs: Requirements specification document, system models (e.g., class diagrams, sequence diagrams), prototype (if used for validation).

Image source : Google

Cont…

4. Requirements Validation

Goal: Ensure that the specified requirements are correct, consistent, complete, and feasible.

Activities: Validate requirements through reviews, walkthroughs, prototyping, simulations, and demonstrations.

Outputs: Validated requirements document, feedback from stakeholders, identified changes or refinements.

Image source : Google

Cont…

5. Requirements Management

Goal: Manage changes to requirements throughout the software development lifecycle.

Activities: Establish baselines for requirements, manage version control, track changes, and ensure traceability between requirements and other project artifacts.

Outputs: Baseline requirements document, traceability matrix, change requests and impact analysis.

Image source : Google

Cont…

6. Requirements Communication

Goal: Ensure that requirements are clearly communicated to all stakeholders.

Activities: Present requirements in formats understandable to different stakeholders (e.g., technical specifications for developers, user stories for users), conduct meetings and status updates.

Outputs: Updated project documentation, meeting minutes, communication plans.

Image source : Google

Cont…

7. **Requirements Traceability**

Goal: Maintain traceability links between requirements and other project artifacts.

Activities: Establish and manage traceability matrices, ensuring that every requirement is traced to design elements, test cases, and implemented code.

Outputs: Traceability matrix, reports on traceability coverage, updated documentation.

Image source : Google

Cont…

8. **Requirements Evolution**

Goal: Manage changes in requirements over time.

Activities: Track changes in stakeholder needs, analyze impacts on existing requirements and project scope, prioritize and implement changes.

Outputs: Updated requirements documentation, change requests, revised project plans

Image source : Google

Software Requirement Specifications

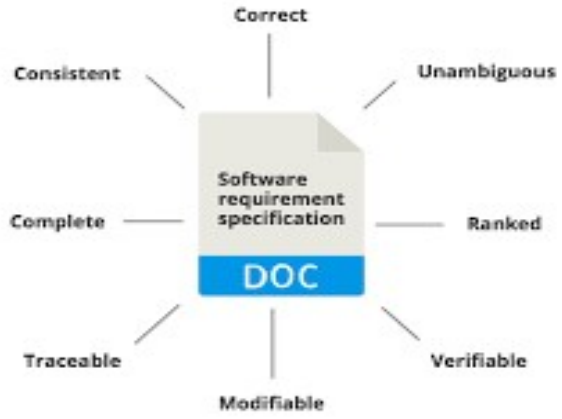
❖ A Software requirement specification (SRS) is a document that mentioned complete description about how the system is expected to perform, behavior of system, functional and non-functional requirements of the system.

❖ SRS is a formal report, that enables the customers to review whether it (SRS) is according to their requirements.

❖ It is usually signed of at end of requirements engineering phase.

User of SRS

1. Client
2. Development Team
3. maintenance team
4. technical Writers



Need of SRS Document

- It structures and formalizes all project requirements.
- It helps the development team build a product that exactly meets customer and target audience needs.
- It provides all team members with the necessary information while working on the project.
- It minimize the possible misunderstanding between the members of the development team and the customer.
- It clearly define the scope of work, and that allows developers to plan particular iterations and release dates.
- It allows estimating the required development budget.

Image source : Google

Structures of SRS

1. Introduction

1.1 Purpose
1.2 Intended Audience
1.3 Scope
1.4 Definition
1.5 References

2. Overall Description

2.1 User Interface
2.2 System Interface
2.3 Software & Hardware requirements
2.4 Constraints
2.5 User Characteristics

3. System features and requirements

3.1 Functional requirements
3.2 Use cases/ sequence diagram
3.3 External Interface requirements
3.4 Database requirements
3.5 References
3.6 Non-functional requirements

4. Delivery for approval

Characteristics of good SRS

1. **Correctness:** provide accurate functional and non-functional requirements as discussed with client.
2. **Completeness:** Should complete all essential features like functionality, performance, features, design, constraints & external interfaces.
3. **consistency:** Same abbreviation, tabular format, diagram format, naming format follow.
4. **Unambiguousness:** Should not be any confusion regarding understanding of the requirements. Everything mentioned uniquely and clearly.
5. **Ranking for importance and stability:** Every requirement is important. But some are urgent and must be fulfilled before other requirements and some could be delayed. It's better to classify each requirement according to its importance and stability

Image source : Google

Characteristics of good SRS

6. **Modifiability:** It is capable of quickly obtain changes to the system without affecting other.
7. **Verifiability:** the requirements are verified with the help of reviewers & stakeholders.
8. **Traceability :** Every requirements having unique number that is easy to use in future development.
9. **Design Independence:** There should be an option to select from multiple design alternatives for the final system. SRS should not contain any implementation details.
10. **Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.
11. **Understandable by the customer:** The language should be kept simple and clear.

Image source : Google

Use cases and Functional specification

https://www.slideshare.net/slideshow/uc-workshop-v2/41862983

Requirements validation

What is Requirements Validation?

The process of checking that the requirements accurately reflect the needs of the stakeholders.

Ensures the software meets user expectations, prevents costly rework, and improves overall project success.

Objectives of Requirements Validation

1. Ensure requirements are complete, consistent, unambiguous, and verifiable.
2. Identify and resolve errors, inconsistencies, and omissions early in the development process.
3. Improve stakeholder confidence in the project.

Image source : Google

Cont…

Requirements Validation Techniques

1. Reviews and Inspections
2. Prototyping
3. Test Case Generation
4. Modeling and Simulation

Challenges in Requirements Validation

1. Difficulty in defining complete and accurate requirements.
2. Stakeholder involvement and collaboration.
3. Balancing cost and quality.
4. Evolving requirements

Image source : Google

Cont…

Best Practices for Requirements Validation

1. Involve stakeholders early and continuously.
2. Use multiple validation techniques.
3. Document validation activities.
4. Iterative and incremental approach.
5. Prioritize requirements for validation

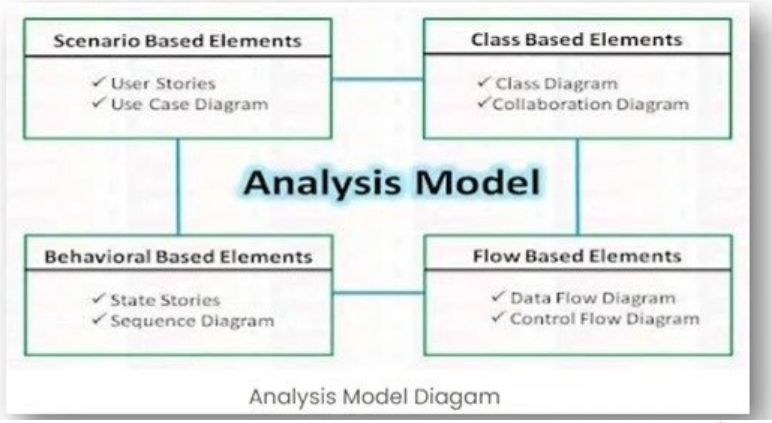
Image source : Google

Requirements Analysis

- Requirement analysis is significant and essential activity after elicitation.
- This activity reviews all requirements and may provide a graphical view of the entire system.
- After the completion of the analysis, the understandability of the project may improve significantly.

➢**Requirement Analysis Model / Elements of Requirement Model**

1. Scenario based Modelling
2. Class based Modelling
3. Flow oriented Modelling
4. Behavior Modelling



Analysis Model Diagam

Image source : Google

Cont..

Scenario Based Modelling / Element

- For building the design & analysis model it is important for software engineer to understand how end users & other actors want to interact with the system.
- Here, creation of scenario by using Use case Diagram, Activity Diagram & User Stories.

1. Use case Diagram:

- A use case diagram is used to represent the dynamic behavior of a system.

 : Actors

 : Use cases

 : Communication Link

 : System Boundary

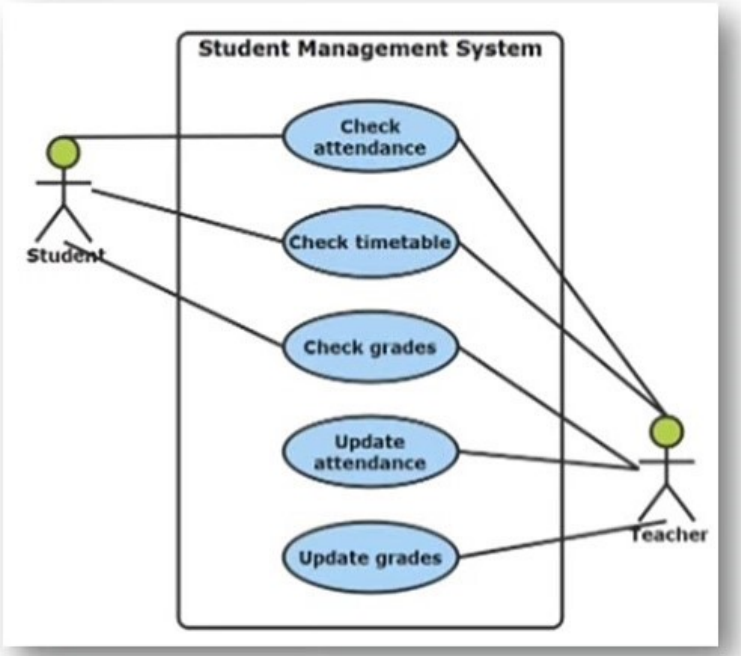


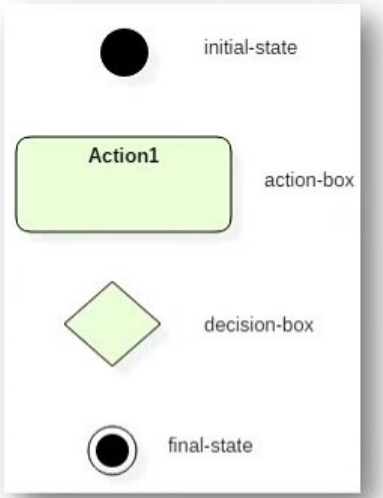
Image source : Google

Cont..

Scenario Based Modelling / Element

2. Activity Diagram:

- The activity diagram helps in predicting the workflow from one activity to another.
- It show condition of flow and the order in which it occurs.



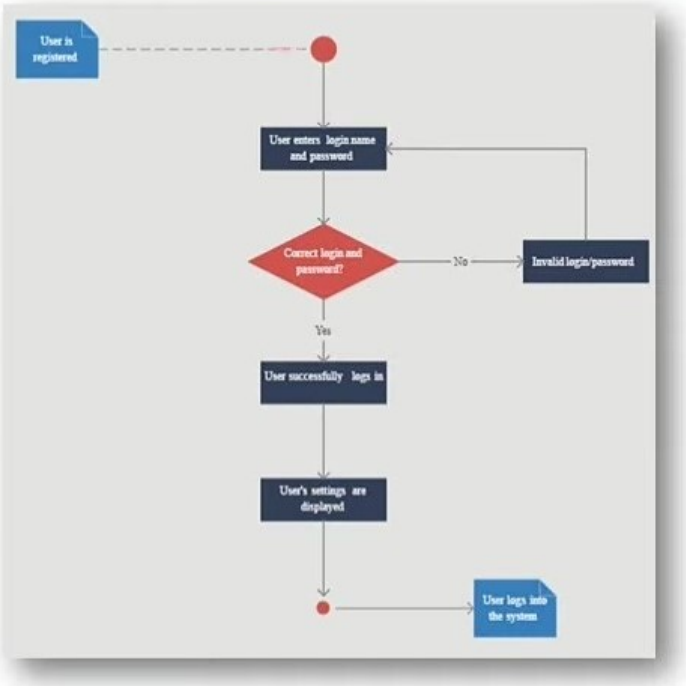


Image source : Google

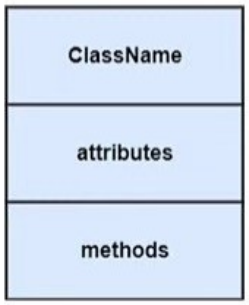
Cont..

Class Based Modelling / Element

- Class based modeling represent classes, object, attributes & operations of system.

1. Class Diagram:

- The class diagram shows a static view of an application.
- It represents the types of objects residing in the system and the relationships between them.
- It describes the major responsibilities of a system.



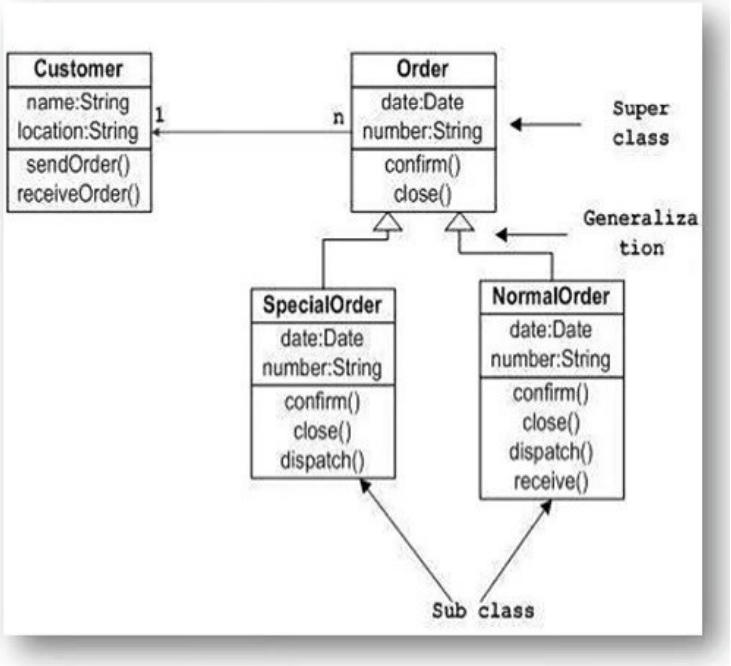


Image source : Google

Cont..

Flow Oriented Modelling / Element

- It shows how data objects are transformed by processing the function.

1. Data Flow Diagram:

- Data flow diagram is graphical representation of flow of data in an information system.
- It is capable of depicting incoming data flow, outgoing data flow and stored data.
- Components of DFD:

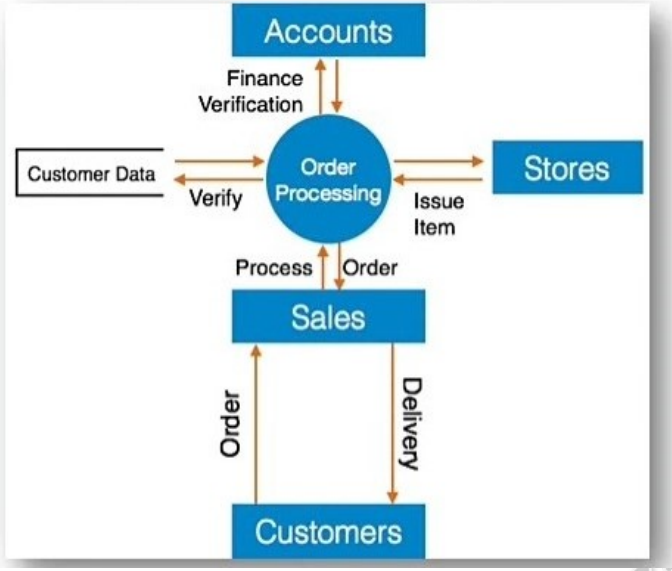


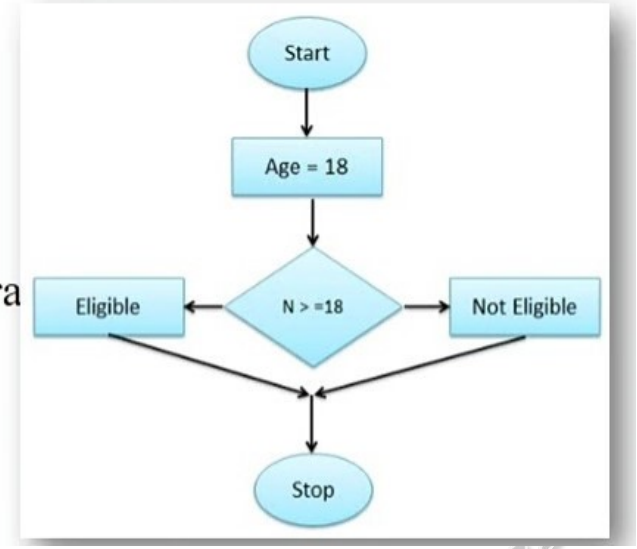
Image source : Google

Cont..

Flow Oriented Modelling / Element

2. Control Flow Diagram:

- It is a graphical representation of control flow or computation during the execution of programs or applications.
- Control flow graphs are mostly used in static analysis as well as compiler applications.
- Accurately represent the flow inside of a program unit.
- Used boolean values are true or false, on or off, 1 or 0.



➢ **Entry Block:**
Entry block allows the control to enter into the control flow gra

➢ **Exit Block:**
Control flow leaves through the exit block.

Image source : Google

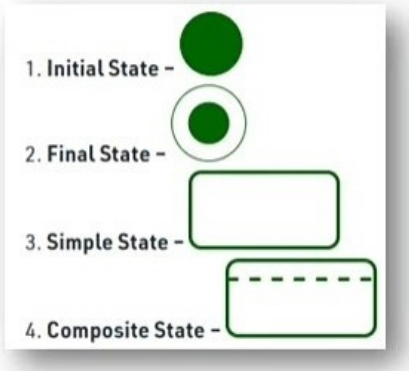
Cont..

Behavioral Modelling / Element

- Behavioral Model is specially designed to make us understand overall behavior and factors that influence behavior of a System.

1. State Transition Diagram:

- It usually describes overall states of system.
- And events which are responsible for a change in state of a system.



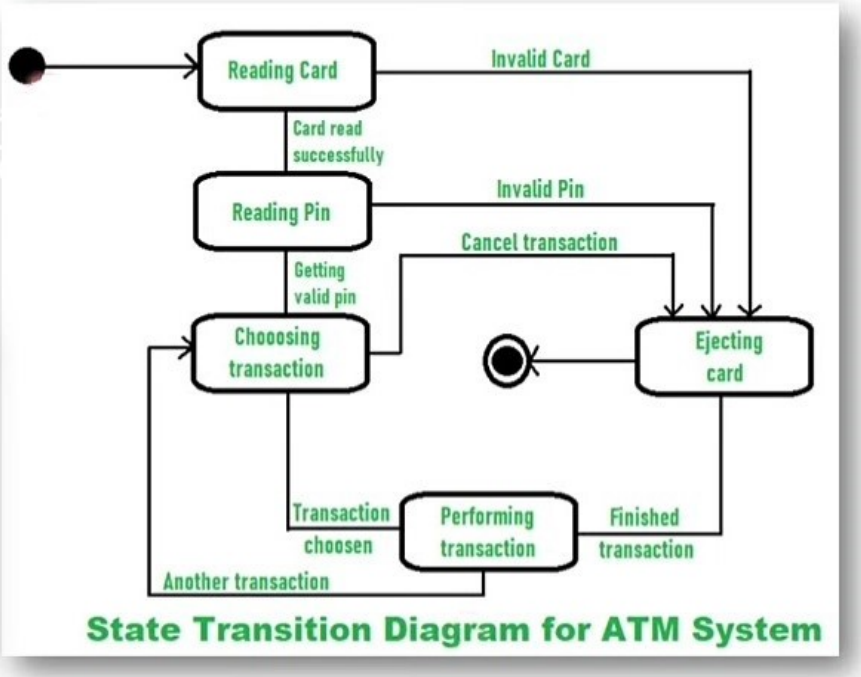


Image source : Google

Any Query?