Unit-03

Requirements Engineering

3.1. Functional and Non-Functional Requirements

In software engineering, requirements are specifications of what the system should do and how it should perform. They are typically categorized into functional and non-functional requirements.

1. Functional Requirements

Functional requirements specify what the system should do. They describe the functions, features, and operations of a system. Examples of Functional Requirements are User Authentication, Data Processing, Order Management.

- Ensure the system provides the necessary functionality for users.
- Form the basis for system design and development.
- Are crucial for meeting user needs and business objectives.

Characteristics:

- Describe specific behaviors or functions of the system.
- Define inputs, outputs, and interactions.
- Typically expressed as actions the system must perform.

2. Non-Functional Requirements

Non-functional requirements specify how the system should perform. They describe the quality attributes, performance, and constraints of the system. Examples of Non-Functional Requirements are Performance, Usability, Reliability, Security.

- Ensure the system meets performance standards and user expectations.
- Address system constraints and environmental conditions.
- Are crucial for system quality, user satisfaction, and regulatory compliance.

Characteristics:

- Define system attributes such as performance, usability, reliability, and security.
- Often describe how well the system performs under specific conditions.
- Typically expressed as constraints or conditions.

3.2. Requirements Engineering Processes

Requirements engineering is the process of defining, documenting, and maintaining the requirements for a software system. It is a critical phase in software development that ensures the system will meet the needs of its stakeholders. The main processes in requirements engineering are:

1. Requirements Elicitation/derivation/expression

Gathering requirements from stakeholders through various techniques.

Activities:

- **Interviews**: Conducting discussions with stakeholders to gather information.
- **Surveys/Questionnaires**: Distributing written questions to stakeholders to collect responses.
- **Workshops**: Facilitating group sessions with stakeholders to identify requirements.
- **Observation**: Watching how users interact with the current system to identify needs.
- **Document Analysis**: Reviewing existing documentation and systems for relevant information.
- **Prototyping**: Creating preliminary versions of the system to gather feedback.

2. Requirements Analysis and Negotiation

Analyzing the gathered requirements to identify conflicts, priorities, and feasibility, and negotiating to resolve conflicts and agree on a final set of requirements.

Activities:

- **Conflict Resolution**: Identifying and addressing conflicting requirements.
- **Prioritization**: Ranking requirements based on their importance and urgency.
- **Feasibility Study**: Assessing the technical and economic feasibility of the requirements.
- **Requirements Classification**: Organizing requirements into categories for better management.
- **Validation and Verification**: Ensuring the requirements are complete, consistent, and unambiguous.

3. Requirements Specification/ Identification

Documenting the requirements in a clear, precise, and comprehensive manner.

Activities:

- **Requirements Documentation**: Writing detailed descriptions of functional and non-functional requirements.
- **Use Cases/User Stories**: Creating scenarios that describe how users will interact with the system.
- **Modeling**: Using diagrams and models (e.g., UML) to represent requirements visually.
- **Requirements Attributes**: Defining attributes like priority, status, and source for each requirement.

4. Requirements Validation

Ensuring the documented requirements accurately reflect the stakeholders' needs and are feasible to implement.

Activities:

- **Reviews**: Conducting formal and informal reviews of the requirements documents.
- **Walkthroughs**: Stepping through the requirements with stakeholders to validate their accuracy.
- **Prototyping**: Developing prototypes to validate requirements through user feedback.
- **Test Case Generation**: Creating test cases to verify that the requirements can be met.

5. Requirements Management

Managing changes to the requirements and maintaining the integrity of the requirements documentation over the project lifecycle.

Activities:

- **Change Control**: Establishing a process for requesting, evaluating, and approving changes to requirements.
- **Version Control**: Keeping track of different versions of requirements documents.

- **Traceability**: Ensuring each requirement can be traced back to its origin and throughout the development process.
- **Impact Analysis**: Assessing the effects of proposed changes on the project.

3.3. Requirements Elicitation

Requirements elicitation is the process of gathering requirements from stakeholders and other sources. It's a critical phase in the requirements engineering process, as it ensures that the development team understands the needs, constraints, and objectives of the system from the stakeholders' perspective. The techniques in requirements elicitation:

Techniques for Requirements Elicitation

1. **Interviews**

→ Direct discussions with stakeholders to gather information.

- **Types**:
    - **Structured Interviews**: Predefined set of questions.
    - **Unstructured Interviews**: Open-ended discussions.
- **Benefits**:
    - Detailed insights from stakeholders.
    - Clarification of ambiguous requirements.
- **Challenges**:
    - Time-consuming.
    - Requires skilled interviewers.

2. **Surveys/Questionnaires**

→ Written questions distributed to stakeholders to collect responses.

- **Types**:
    - **Open-Ended Questions**: Allow detailed responses.
    - **Closed-Ended Questions**: Provide specific options for responses.
- **Benefits**:
    - Can reach a large audience quickly.
    - Easy to analyze quantitative data.
- **Challenges**:
    - Limited depth of responses.
    - May not capture nuanced requirements.

3. **Workshops**

→ Facilitated group sessions with stakeholders to identify and discuss requirements.

- **Benefits**:
  - Encourages collaboration and consensus.
  - Can generate a large amount of information quickly.
- **Challenges**:
  - Requires careful planning and facilitation.
  - Group dynamics can influence outcomes.

4. Observation

→ Watching how users interact with the current system to identify their needs and challenges.

- **Types**:
  - **Passive Observation**: The observer does not interact with the users.
  - **Active Observation**: The observer interacts with users to understand their tasks.
- **Benefits**:
  - Provides real-world insights into user behavior.
  - Can identify hidden requirements.
- **Challenges**:
  - Can be time-consuming.
  - Users may change their behavior when observed.

5. Document Analysis

→ Reviewing existing documentation and systems for relevant information.

- **Sources**:
  - System manuals.
  - Process documentation.
  - Business plans.
- **Benefits**:
  - Provides historical context and background information.
  - Can identify existing requirements and constraints.
- **Challenges**:
  - Documents may be outdated or incomplete.
  - Requires thorough and careful examination.

6. Prototyping

→ Creating preliminary versions of the system to gather feedback from stakeholders.

- **Types**:
  - **Low-Fidelity Prototypes**: Basic, often paper-based models.
  - **High-Fidelity Prototypes**: Interactive and close to the final product.
- **Benefits**:
  - Visual representation of requirements.
  - Allows stakeholders to interact with the system early.
- **Challenges**:
  - Can be time-consuming to create.
  - May raise expectations for the final system.

3.4. Requirements Specification

Requirements specification is the process of documenting the gathered requirements in a clear, detailed, and structured manner. This documentation serves as a reference for both the development team and stakeholders throughout the software development lifecycle.

1. Functional Requirements

- **Definition**: Describe specific behaviors, functions, or features the system must perform.
- **Examples**:
  - User authentication: The system must allow users to log in using a username and password.
  - Order processing: The system must allow users to place, update, and cancel orders.

2. Non-Functional Requirements

- **Definition**: Describe how the system performs a function, encompassing system attributes such as performance, usability, reliability, and security.
- **Examples**:
  - Performance: The system must handle 1000 transactions per second.
  - Usability: The system must be easy to use and understand for non-technical users.
  - Reliability: The system must have 99.9% uptime.
  - Security: The system must encrypt all sensitive user data.

3. Use Cases/User Stories

- **Definition**: Scenarios that describe how users will interact with the system to achieve specific goals.
- **Examples**:
    - "As a user, I want to reset my password so that I can recover access to my account."
    - "As an admin, I want to generate monthly sales reports so that I can track performance."

4. Diagrams and Models

- **Definition**: Visual representations of requirements to help understand and communicate complex requirements.
- **Examples**:
    - Use case diagrams
    - Class diagrams
    - Sequence diagrams
    - State diagrams

5. Requirements Attributes

- **Definition**: Additional information about requirements such as priority, status, and source.
- **Examples**:
    - Priority: High/Medium/Low
    - Status: Approved/Pending/In progress
    - Source: Stakeholder name or document reference

3.5. Requirements Validation

Requirements validation is the process of ensuring that the documented requirements accurately reflect the needs and expectations of stakeholders and that they are feasible to implement. This step is crucial to prevent errors and misunderstandings that can lead to project delays, cost overruns, or failure to meet user needs.

Activities in Requirements Validation

1. Requirements Reviews

- **Description**: Systematic examination of the requirements documentation by a team of reviewers (stakeholders, developers, testers).
- **Objective**: Identify errors, ambiguities, omissions, and inconsistencies.

- **Types**:
  - **Formal Reviews**: Structured meetings with predefined agendas and roles.
  - **Informal Reviews**: Unstructured or ad-hoc discussions.

2. Prototyping

- **Description**: Developing a preliminary version of the system or a specific feature to gather feedback.
- **Objective**: Validate the requirements by allowing stakeholders to interact with a tangible representation.
- **Types**:
  - **Low-Fidelity Prototypes**: Basic models, often paper-based.
  - **High-Fidelity Prototypes**: Interactive and closer to the final system.

3. Model Validation

- **Description**: Using models (e.g., UML diagrams) to represent requirements and validate them through visualization.
- **Objective**: Ensure the completeness and correctness of the requirements by visual inspection.
- **Types**:
  - **Use Case Diagrams**: Validate functional requirements and interactions.
  - **Sequence Diagrams**: Validate the flow of processes and interactions over time.

4. Walkthroughs

- **Description**: Step-by-step presentation of the requirements documentation by the author to stakeholders.
- **Objective**: Gather feedback, identify issues, and ensure understanding.
- **Method**: Typically involves explaining the requirements and answering questions in an interactive session.

5. Inspections

- **Description**: Detailed examination of the requirements documents by a team, following a well-defined process.
- **Objective**: Detect defects early and ensure quality.
- **Focus Areas**: Consistency, completeness, correctness, and feasibility.

6. Test Case Generation

- **Description**: Creating test cases based on the requirements to ensure they are testable and verifiable.
- **Objective**: Validate that the requirements can be effectively tested and meet the desired criteria.
- **Types**:
  - **Functional Test Cases**: Derived from functional requirements.
  - **Non-Functional Test Cases**: Derived from non-functional requirements (e.g., performance, security).