CSC 540 2.0
Software Engineering

M. K. A. Ariyaratne

Department of Computer Science
Faculty of Applied Sciences
University of Sri Jayewardenepura
Sri Lanka

UNIT 3
Requirements Engineering

Requirements Engineering - Topics to cover

1. Functional and Non functional requirements
2. Software Requirements elicitation
3. The software requirements document
4. Software Requirements Specification
5. Requirements Validation
6. Requirement Management

What are Requirements of a System - Requirements Engineering

- Requirements are **descriptions of the services** that a software system must provide and the **constraints under which it must operate**.
- Requirements reflect the needs of customers.
- The process of **finding out**, **analyzing**, **documenting** and **checking these services and constraints** is called **requirements engineering (RE)**.

Requirements Engineering (RE)

- It focuses on assessing if the system is useful to the business **(feasibility study)**,
  discovering requirements **(elicitation and analysis)**,
  converting these requirements into some standard format **(specification)**,
  and checking that the requirements define the system that the customer wants **(validation)**.
- Practically, RE isn't sequential process, it's an iterative process in which activities are interleaved.
  (For example, you iterate first on the **user requirements**; elicitation, specification, and validation, and repeat the same steps for the **system requirements**.)

Requirements??

- A requirement can be an abstract statement or a detailed description.

  A company wishes to let a contract for a large software development project, it defines its needs in a sufficiently abstract way that a solution is not predefined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.

User and System Requirements

1. Requirements can be categorized in several different ways.
2. One way is diving them as user and system requirement.
3. The difference between user requirements and system requirements lies in their scope, audience, level of detail, and purpose. Here's a detailed explanation:

User Requirements

Definition:

- High-level descriptions of what the system should do.
- Expressed in natural language, diagrams, or use cases.

Audience:

- Written for stakeholders, including end-users, clients, and non-technical personnel.

Purpose:

- To capture the goals and expectations of the system from the user's perspective.

User Requirements: Characteristics

- Focuses on **what** the users need, not **how** it will be implemented.
- Described in terms of the user's tasks, workflows, or problems to be solved.
- Uses simple language to ensure clarity for non-technical stakeholders.

User Requirements: Examples

- *Example 1:* **"The system should allow users to reset their passwords via email."**
- *Example 2:* **"The system should generate monthly sales reports in PDF format."**

System Requirements

Definition:

- Detailed specifications that define the system's functionality, performance, and constraints.
- Intended for developers and technical teams.

Audience:

- Engineers, designers, developers, and testers.

Purpose:

- To provide a precise and detailed blueprint for building and testing the system.

System Requirements: Characteristics

- Focuses on **how** the system will implement user requirements.
- Includes technical details such as data models, algorithms, and hardware/software specifications.
- Often divided into functional and non-functional requirements.

System Requirements: Examples

- *Functional Requirement:* **"The password reset functionality shall use a secure email token valid for 24 hours."**
- *Non-Functional Requirement:* **"The system shall handle 1000 concurrent users with a response time of less than 2 seconds."**

Comparison of User and System Requirements

Aspect	User Requirements	System Requirements
Focus	High-level goals and user needs	Technical implementation and detailed specifications
Audience	Non-technical stakeholders (users, clients)	Technical teams (developers, testers, engineers)
Level of Detail	General and abstract	Detailed and specific
Format	Natural language, diagrams, or use cases	Technical documents, specifications, models
Purpose	Define **what** the system should do	Define **how** the system will achieve the requirements

Table: Comparison of User and System Requirements

Summary

- **User Requirements:** Focus on *what* the user wants to achieve.
- **System Requirements:** Focus on *how* the system will fulfill those needs.
- Both are crucial for successful system development:
  - User requirements ensure alignment with stakeholder goals.
  - System requirements guide technical implementation.

Functional and Non functional requirements

1. Another way of categorizing requirements is divide them as Functional and Non functional requirements.
2. The difference between functional requirements and non-functional requirements lies in what they describe about a system.

Functional Requirements

Definition:

- Functional requirements specify **what** the system should do.
- Describe the functionality or services the system is expected to provide.

Purpose:

- To define the primary behavior of the system.
- Ensure that the system meets the needs of its users.

Functional Requirements: Characteristics

- Clearly define **inputs, processes, and outputs**.
- Focus on **specific tasks** or user actions.
- Ensure compliance with user requirements.
- Testable through functional testing methods.

Functional Requirements: Examples

- *Example 1:* **"The system shall authenticate users using a username and password."**
- *Example 2:* **"The system shall generate monthly financial reports."**
- *Example 3:* **"The system shall allow users to search for products by name, category, or price range."**

Non-Functional Requirements

Definition:

- Non-functional requirements specify **how** the system should perform its functions.
- Define the quality attributes, constraints, or criteria for system performance.

Purpose:

- To ensure the system operates efficiently and effectively.
- Provide constraints for system design and implementation.

Non-Functional Requirements: Characteristics

- Focus on **performance, usability, reliability, and scalability**.
- Often apply across the entire system rather than specific functions.
- Testable through non-functional testing methods (e.g., performance testing).
- Defined using measurable criteria.

Non-Functional Requirements: Examples

- *Performance:* **"The system shall handle 1000 concurrent users with a response time of less than 2 seconds."**
- *Security:* **"The system shall encrypt all sensitive data using AES-256."**
- *Usability:* **"The system shall have an intuitive interface accessible to users with minimal training."**

Comparison of Functional and Non-Functional Requirements

Aspect	Functional Requirements	Non-Functional Requirements
Definition	What the system should do	How the system should perform
Focus	Specific tasks and functionalities	Quality attributes and constraints
Testability	Functional testing	Non-functional testing (e.g., performance)
Scope	Specific functions or features	System-wide attributes
Examples	User authentication, data retrieval	Performance, scalability, security

Table: Comparison of Functional and Non-Functional Requirements

Functional and Non-functional Requirements

State which requirements are functional / non-functional.

- The system must send an email whenever an order is placed.
- Emails should be sent with a latency of no greater than 12 hours from such an activity.
- Modified data in a database should be updated for all users accessing it within 2 seconds.
- The system shall display the heart rate of the patient.
- The system shall display the heart rate of the patient within three seconds.

Functional and Non-functional Requirements

List some functional and non-functional requirements for a smart phone.

- **Functional:**

  The system shall make a phone call.

  The system shall receive a phone call.

  The system shall allow the user to adjust volume.

- **Non-functional:**

  The system shall boot up in 60 seconds or less.

  The system shall respond to touch within 1 second.

Problems on Non-Functional Requirements

- Users or customers often propose these requirements as general goals. e.g.:**The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.**

- It is impossible to objectively verify the system goal, but goals (Non-Functional Requirements) can be written in more testable way. e.g. Medical staff shall be able to use all the system functions after **four hours** of training. After this training, the average number of errors made by experienced users **shall not exceed two per hour** of system use.

Measuring Non-Functional Requirements

- Whenever possible, you should write non-functional requirements **quantitatively**.

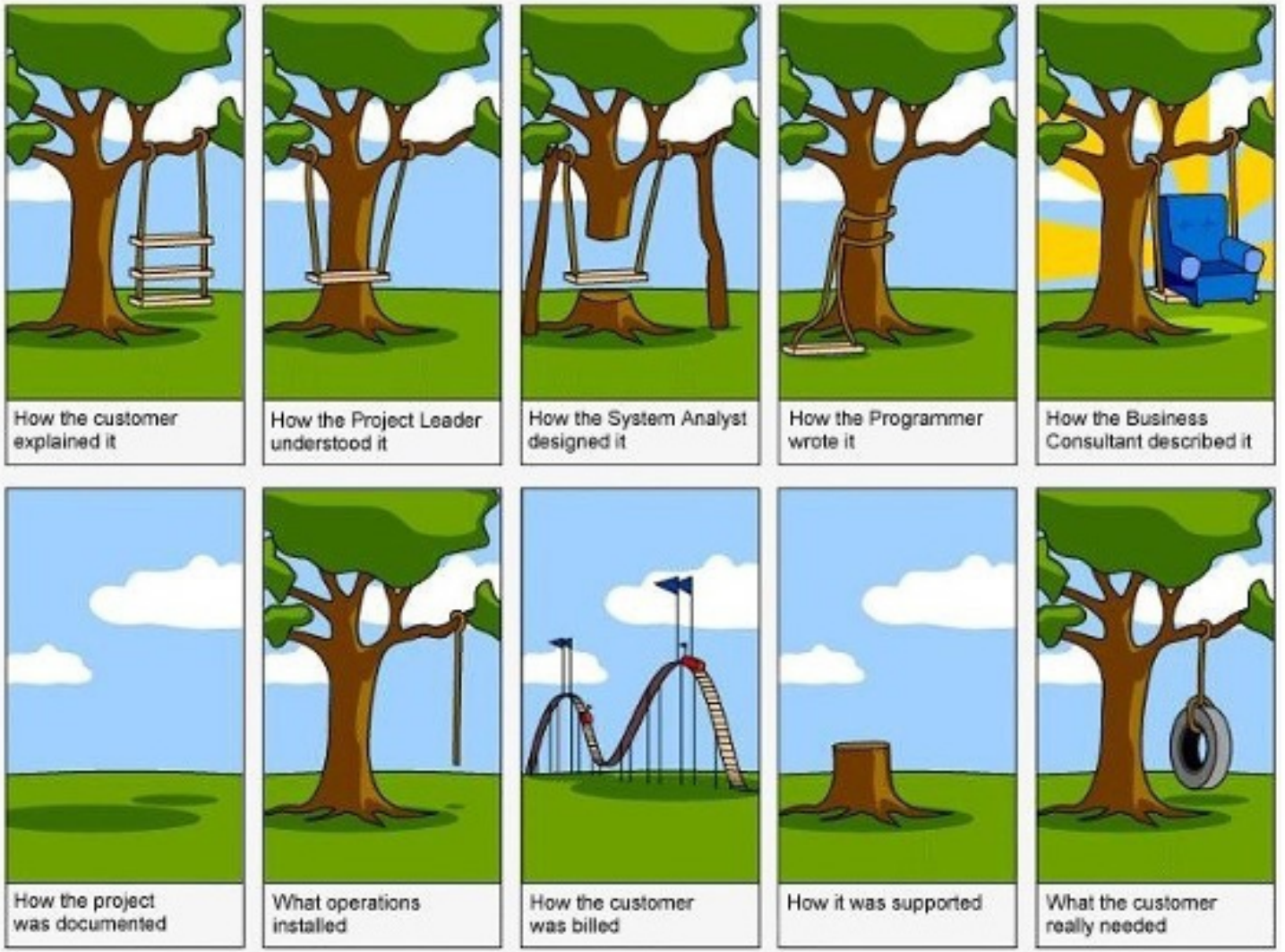
  so that they can be objectively tested.

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Summary

- **Functional Requirements:** Define *what* the system should do.
- **Non-Functional Requirements:** Define *how* the system should operate.
- Both types of requirements are essential for:
  - Ensuring the system meets user needs.
  - Delivering a reliable, efficient, and high-quality product.

Requirements Elicitation



Requirements Elicitation

- Requirement elicitation is the process of gathering and discovering the needs, expectations, and constraints of stakeholders to define the requirements for a software system.
- It is the first step in the requirements engineering process, ensuring that the project aligns with stakeholder goals.

Requirements Elicitation - Objectives

1. Understand Stakeholder Needs: Identify and clarify what stakeholders require from the system.
2. Capture Constraints: Document limitations like budget, time, technical constraints, or regulatory compliance.
3. Define Scope: Establish the boundaries of the system to avoid scope creep.

Requirements Elicitation - Key Activities

1. Identifying Stakeholders:
   Determine who will be involved in or affected by the system, such as clients, end-users, developers, and domain experts.
2. Gathering Information:
   Collect relevant information about the domain, processes, and expectations.
3. Clarifying Requirements:
   Ensure all requirements are clear, consistent, and unambiguous.
4. Documenting Requirements:
   Record requirements in a structured format, such as a Software Requirements Specification (SRS) document.
5. Validating Requirements:
   Confirm with stakeholders that the captured requirements align with their needs and expectations.

Sources of information for Requirements Discovery (elicitation)

- The first step of requirement elicitation is identifying stakeholders. Stakeholders are the one who will invest in and use the product.
- Stakeholders range from end-users of a system through managers to external stakeholders.
  Eg: system stakeholders for a patient information system may include

1. Patients
2. Doctors
3. Nurses
4. Medical receptionists who manage patients' appointments.
5. IT staff responsible for installing and maintaining the system.
6. Health-care managers who obtain management information from the system.

Sources of information for Requirements Discovery (elicitation)

Other sources

- Documentation
- Workers
- Existing systems in the company
- Specifications of similar systems

Problems of Requirements elicitation

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organizational and political factors may influence the system requirements.
- The requirements change during the analysis process.
- New stakeholders may emerge and business environment may also likely to change.

Techniques used for Requirements Discovery

1. Interviewing
2. Questionnaires
3. Document Analyzing
4. Prototyping
5. Scenarios
6. Usecases
7. Ethnography

1. Interviewing

Interviewing for Requirements Discovery

- Can be formal or informal, done with stakeholders.
- Done by the requirements engineering team.
- Puts questions about the system that they currently use and the system to be developed.
- May be two types, Open or Closed.
  1. Closed interviews, where the stakeholder answers a pre-defined set of questions.
  2. Open interviews, in which there is no pre-defined agenda.
- In practice, interviews are normally a mixture of both of these. (Completely open-ended discussions rarely work well.)

Interviewing for Requirements Discovery - Problems

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- Interviews are not good for understanding domain requirements
  1. Requirements engineers cannot understand specific domain terminology;
  2. Some domain knowledge is so familiar that people find it hard to explain or think that it isn't worth explaining.

Interviewing for Requirements Discovery - Problems

- Interviews may not an effective technique for eliciting knowledge about organizational requirements and constraints.
- In general, most people are reluctant to discuss political and organizational issues that may affect the requirements.
- Time consuming.
- Interviews can also be costly.
- The outcome of interviews is largely dependent on the analyst's human relation, personality, and influencing skills.

2. Questionnaires

Questionnaires for Requirements Discovery

- Questionnaires are much more informal, and they are good tools to gather requirements from stakeholders in remote locations.
- Questionnaires can also be used when you have to gather input from dozens, hundreds, or thousands of people.

Benefits

- Less cost.
- Reach Large number of People.
- The responses are gathered in a standardized way.

Risks and Drawbacks

- Difficulties may arrive when filling by users.
- participants may forget important issues.
- Stockholders may not be willing to answer the questions.

3. Document Analysis

Document Analysis for Requirements Discovery

- It is the art of studying relevant business, system and project documentation with the objective of understanding the business, the project background and identifying requirements or opportunities for improvement.
- It's a means of gathering information before scheduling interviews or other elicitation sessions with stakeholders.
- Document analysis can be considered as a means of verifying requirements.

Document Analysis for Requirements Discovery (2)

Type of Documents to be analyzed

- Functional specifications of existing system
- Business Rule Documentation
- Organizational Structure
- Company Memos
- Customer Suggestions/Feedback
- System Defect Reports

Document Analysis for Requirements Discovery (3)

Benefits

- Can come in useful where the stakeholder is unavailable or no longer with the organization
- Ensures that the analyst does not start from a blank page
- Acts as a means of cross-checking requirements with other sources

Risks and Drawbacks

- Time Consuming
- Conflicts
- Exhausted
- Not Found Real Figures

4. Scenarios


1
Sector analysis taking into account global and national trends
2
The main strategic alternatives in sector development
3
Possible sector development models
4
Sectors visions and scenarios of future development
5
Perspective S&T directions and technologies
➢Global and national trends analysis
➢Sector position in economy (share in GDP, exports, employment)
➢Domestic and foreign demand analysis
➢Relation with other sectors (suppliers and consumers of production)
➢Institutional structure and sector environment
➢Technologic level
➢Resource provision
➢Technologic alternatives
➢Institutional alternatives
➢Domestic and foreign demand alternatives
➢Regulative alternatives
➢Sector development model formation in dependence of different alternatives combination
➢Determination of the models depending on their probability
Sector vision scenarios description by the following characteristics
➢Institutional structure
➢Basic technologies and resource consuming
➢New technologies
➢Related sectors
➢Home and foreign demand
➢Positioning on the foreign markets
➢Sub-sectors vision
➢Geography
➢Resources
➢Effects for economy
➢Risks and limitations
Perspective technologies and technologic solutions characteristics
➢Invariant to selected scenario (inevitable future)
➢Specific for each of concerned scenarios

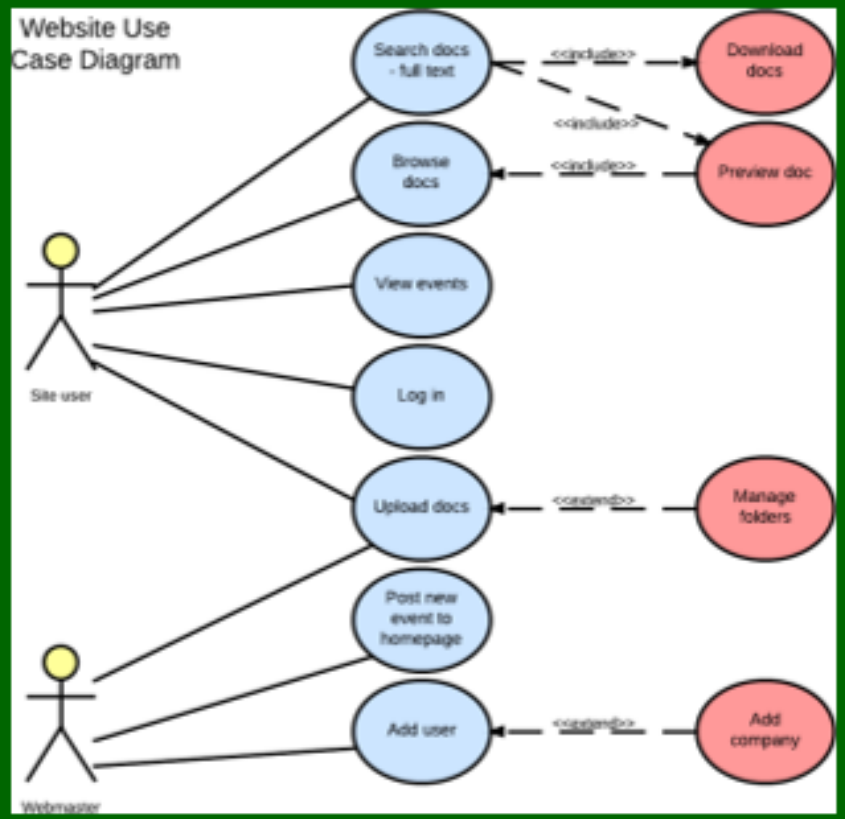
Scenarios for Requirements Discovery

- A concrete, focused, informal description of a single feature of the system used by a single actor.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.
- Scenario-based elicitation involves working with stakeholders to identify scenarios and to capture details to be included in these scenarios.

Scenarios for Requirements Discovery (2)

- Scenarios may be written as text, supplemented by diagrams, screen shots, etc.
- Alternatively, a more structured approach such as use cases may be used.

5. Use cases


Website Use Case Diagram
Search docs - full text
Download docs
<<include>>
Browse docs
<<include>>
Preview doc
View events
Site user
Log in
Upload docs
<<extend>>
Manage folders
Post new event to homepage
Add user
<<extend>>
Add company
Webmaster

Use cases for Requirements Discovery

- In their simplest form, a use case identifies the actors involved in a use case and names the type of use case.
- This diagram is supplemented by additional information describing the interaction with the system.
- The additional information may be a textual description or one or more graphical models such as UML sequence or state charts.
- There is no hard and fast distinction between scenarios and use cases.
- A set of scenarios make a use case.

Use cases for Requirements Discovery (2)

- There would be a scenario for the normal interaction plus scenarios for each possible exception.
- Use cases identify the individual interactions between the system and its users or other systems. Each use case should be documented with a textual description.
- These can then be linked to other models in the UML that will develop the scenario in more detail.
- Scenarios and use cases are effective techniques for eliciting requirements from stakeholders who interact directly with the system.

Use cases for Requirements Discovery (3)

- Each type of interaction can be represented as a use case.
- However they are not as effective for eliciting constraints or high-level business and nonfunctional requirements.

6. Ethnography

Ethnography for Requirements Discovery

- Ethnography is an observational technique that can be used to understand operational processes.
- An analyst locate himself or herself in the working environment where the system will be used.
- The day-to-day work is observed and notes made of the actual tasks in which participants are involved.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.
- Requirements are derived from the way that people actually work.

Ethnography for Requirements Discovery (2)

- Ethnography is effective for understanding existing processes.
- Hardly use to identify new features that should be added to a system.

Benefits

- The ability to record and report all findings that are true
- It is more practical
- No long calculation has to be done

Ethnography for Requirements Discovery (3)

Risks and Drawbacks

- The viewer's or researcher's own perception
- few trials/studies/or objects observed to make an end conclusion
- results may contain human error

Comparison of few elicitation techniques

Technique	Good for	Kind of data	Plus	Minus
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is cruci Response rate may low. Responses ma not be what you wa
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee. Encourages contact between developers and users	Time consuming. Artificial environme may intimidate interviewee
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of domina characters
Naturalistic observation	Understanding context of user activity	Qualitative	Observing actual work gives insight that other techniques cannot give	Very time consumin Huge amounts of da
Studying documentation	Learning about procedures, regulations, and standards	Quantitative	No time commitment from users required	Day-to-day work w differ from documen procedures

Requirements Analysis

- Requirement analysis is the process of **refining, organizing, and evaluating the requirements** gathered during the requirement elicitation phase. Its goal is to ensure that all requirements are well-defined, consistent, complete, and feasible for implementation.

Relationship Between Elicitation and Analysis:

- Requirement Elicitation: Focuses on gathering and capturing stakeholder needs.
- Requirement Analysis: Focuses on refining, structuring, and validating those needs for implementation.

Key Activities in Requirement Analysis

1. Categorizing Requirements:
   Classify into functional, non-functional, and domain-specific requirements.
2. Validating Requirements:
   Check if requirements align with stakeholder goals and project scope.
3. Resolving Conflicts:
   Address discrepancies or contradictions between stakeholder needs. Documenting Requirements:
   Refine the requirements and create a structured specification document (e.g., SRS).
4. Developing Models:
   Use models to visually represent system behavior, data flow, or architecture.
5. Defining Acceptance Criteria:
   Establish measurable criteria for determining if a requirement is successfully implemented.

What is SRS?

- **Definition:** An official document describing a system's intended functions, constraints, and requirements.
- **Purpose:**
  - Acts as a contract between stakeholders and developers.
  - Guides design, development, and testing.
- **Importance:**
  - Ensures shared understanding among stakeholders.
  - Reduces misunderstandings and scope creep.

Characteristics of a Good SRS

A good SRS must be:

- **Clear:** Free of ambiguity.
- **Complete:** Includes all functional and non-functional requirements.
- **Consistent:** No conflicting requirements.
- **Verifiable:** Testable requirements.
- **Modifiable:** Easy to update.
- **Traceable:** Linked to their sources or use cases.

Components of an SRS

Suggested Sections:

1. **Preface:** Readership, version history, and changes.
2. **Introduction:** Overview of system goals and scope.
3. **Glossary:** Definitions of terms used.
4. **User Requirements Definition:** Services and non-functional requirements.
5. **System Architecture:** Overview of system structure.
6. **System Requirements Specification:** Detailed requirements.
7. **System Models:** Graphical representations.
8. **System Evolution:** Assumptions and future considerations.
9. **Appendices:** Additional hardware or database requirements.
10. **Index:** For quick navigation.

Who Uses an SRS?

- **Developers:** Use SRS to design and build the system.
- **Testers:** Verify that the system meets requirements.
- **Clients and Stakeholders:** Ensure their needs are reflected accurately.

Types of Requirements in an SRS

- **Functional Requirements:**
  - Define specific functions the system must perform.
  - Example: "The system shall generate monthly sales reports."
- **Non-Functional Requirements:**
  - Define performance, usability, reliability, etc.
  - Example: "The system shall handle 1000 concurrent users."
- **Domain Requirements:**
  - Specific to the application domain.
  - Example: "The system must comply with HIPAA regulations." HIPAA stands for Health Insurance Portability and Accountability Act, a federal law that protects sensitive health information and sets standards for its handling

Techniques for Writing an SRS

1. **Natural Language:** Use plain text with clear, concise sentences.
2. **Structured Natural Language:** Use templates or forms to organize information.
3. **Graphical Models:** UML diagrams, sequence diagrams, or flowcharts.
4. **Formal Methods:** Mathematical specifications for high-reliability systems.

Examples of SRS Requirements

- **Functional Requirement:**
  - "The system shall send an email notification upon successful order placement."
- **Non-Functional Requirement:**
  - "The system shall maintain 99.99% uptime."
- **Domain Requirement:**
  - "All financial transactions must comply with PCI-DSS standards."

Challenges in Creating an SRS

- Ambiguities in requirements.
- Conflicting needs among stakeholders.
- Balancing completeness with conciseness.
- Ensuring requirements are testable and verifiable.

Benefits of a Well-Written SRS

- Aligns stakeholder expectations with deliverables.
- Serves as a reference throughout the project lifecycle.
- Reduces misunderstandings and errors.
- Facilitates better project planning and management.

Summary

- **User Requirements:** High-level goals from the user's perspective.
- **System Requirements:** Detailed technical specifications.
- **Purpose of SRS:** Ensures a clear, shared understanding among stakeholders.
- A well-written SRS is critical for project success.

What is Requirement Validation and Management?

- **Requirement Validation:** Ensures that the software requirements accurately represent the stakeholder's needs.
- **Requirement Management:** A process to handle requirements throughout the software development lifecycle.
- **Purpose:**
  - Prevent errors in requirements.
  - Maintain consistency and traceability.

Why is Validation and Management Important?

- Avoids costly changes later in the development process.
- Ensures that delivered software meets user expectations.
- Helps manage changing requirements effectively.
- Reduces risks of project failures due to unclear or incomplete requirements.

What is Requirement Validation?

- The process of checking that requirements define the system stakeholders truly need.
- Performed after requirements are gathered and documented.

Techniques for Requirement Validation

- **Reviews:**
  - Stakeholders review requirements for clarity and feasibility.
- **Prototyping:**
  - Develop prototypes to demonstrate requirements.
- **Test Cases:**
  - Derive test cases from requirements to validate correctness.

Common Issues in Requirement Validation

- Ambiguous requirements leading to multiple interpretations.
- Incomplete requirements missing critical details.
- Conflicting requirements between stakeholders.
- Unrealistic requirements that cannot be implemented.

What is Requirement Management?

- A continuous process to monitor, control, and maintain requirements.
- Helps in managing requirement changes effectively.

Key Activities in Requirement Management

1. **Requirement Identification:**
   - Assign unique identifiers for traceability.
2. **Requirement Tracking:**
   - Monitor changes and their impact.
3. **Version Control:**
   - Maintain versions of requirements.
4. **Change Management:**
   - Evaluate and document requirement changes.

Tools for Requirement Management

- **IBM Engineering Requirements Management DOORS:** For large-scale projects.
- **Jama Software:** Requirement tracking and collaboration.
- **JIRA:** Tracks requirements, tasks, and bugs.
- **ReqSuite RM:** Simplifies requirement documentation and tracking.

Benefits of Requirement Management

- Ensures alignment between stakeholders and developers.
- Improves communication and collaboration.
- Reduces risks of missing or misunderstood requirements.
- Facilitates easier project tracking and planning.

Challenges in Requirement Management

- Managing frequent changes in requirements.
- Handling conflicting requirements among stakeholders.
- Maintaining traceability for large projects.
- Allocating resources to handle requirement updates.

Summary

- **Validation:** Ensures requirements meet stakeholder needs.
- **Management:** Maintains and tracks requirements throughout the lifecycle.
- Both processes are essential to prevent costly changes and ensure project success.

Summary on Requirements Engineering

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.

Summary on Requirements Engineering (2)

- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- The requirements engineering process is an iterative process including requirements elicitation, specification and validation.
- Requirements elicitation and analysis is an iterative process. Elicitation process includes requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.

Summary on Requirements Engineering (3)

- You can use a range of techniques for requirements elicitation.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.

Question 1: Functional vs Non-Functional Requirements

Which of the following is an example of a non-functional requirement?

A. The system shall allow users to reset their password.

B. The system shall generate a monthly sales report.

C. The system shall support up to 100 concurrent users.

D. The system shall send a notification when an order is placed.

Answer: C. The system shall support up to 100 concurrent users

Question 2: System Requirements

System requirements are:

A. High-level objectives of the system

B. Low-level hardware and software specifications

C. Detailed descriptions of what the system must do

D. Descriptions of user satisfaction

Answer: C. Detailed descriptions of what the system must do

Question 3: Functional Requirements

Which of the following best describes functional requirements?

A. Constraints on the design of the system
B. Descriptions of the system's services and functionalities
C. Performance metrics for the system
D. Aesthetic considerations for the user interface

Answer: B. Descriptions of the system's services and functionalities

Question 4: Non-functional Requirements

Non-functional requirements are primarily concerned with:

A. What the system should do

B. Who the users of the system are

C. How well the system performs its functions

D. The cost of the system

Answer: C. How well the system performs its functions

Question 5: SRS Document

Which of the following is NOT a component of the Software Requirements Specification (SRS)?

A. Functional requirements

B. Non-functional requirements

C. Source code

D. Glossary

Answer: C. Source code

Question 6: Requirements Elicitation

Which technique is NOT typically used for requirements elicitation?

A. Interviews
B. Brainstorming
C. Unit testing
D. Prototyping

Answer: C. Unit testing

Question 7: Requirements Validation

Requirements validation ensures that:

A. The requirements are implementable within the project constraints.
B. The system meets the needs of stakeholders.
C. The requirements are free from ambiguity and errors.
D. All of the above.

Answer: D. All of the above

Question 8: Stakeholders

Stakeholders in a project include:

A. Only the project sponsor

B. Only the development team

C. Anyone who is affected by the system

D. Only the end-users

Answer: C. Anyone who is affected by the system

Question 09: Prioritization

What is the primary purpose of prioritizing requirements?

A. To ensure that all requirements are implemented
B. To focus on the most critical requirements first
C. To reduce project costs
D. To eliminate unnecessary requirements

Answer: B. To focus on the most critical requirements first