Requirement Engineering

Ashima Tyagi
Assistant Professor
School of Computer Science & Engineering

Outline

- Requirement
- Types of Requirements
- Requirement Engineering Tasks
- Requirements documentation

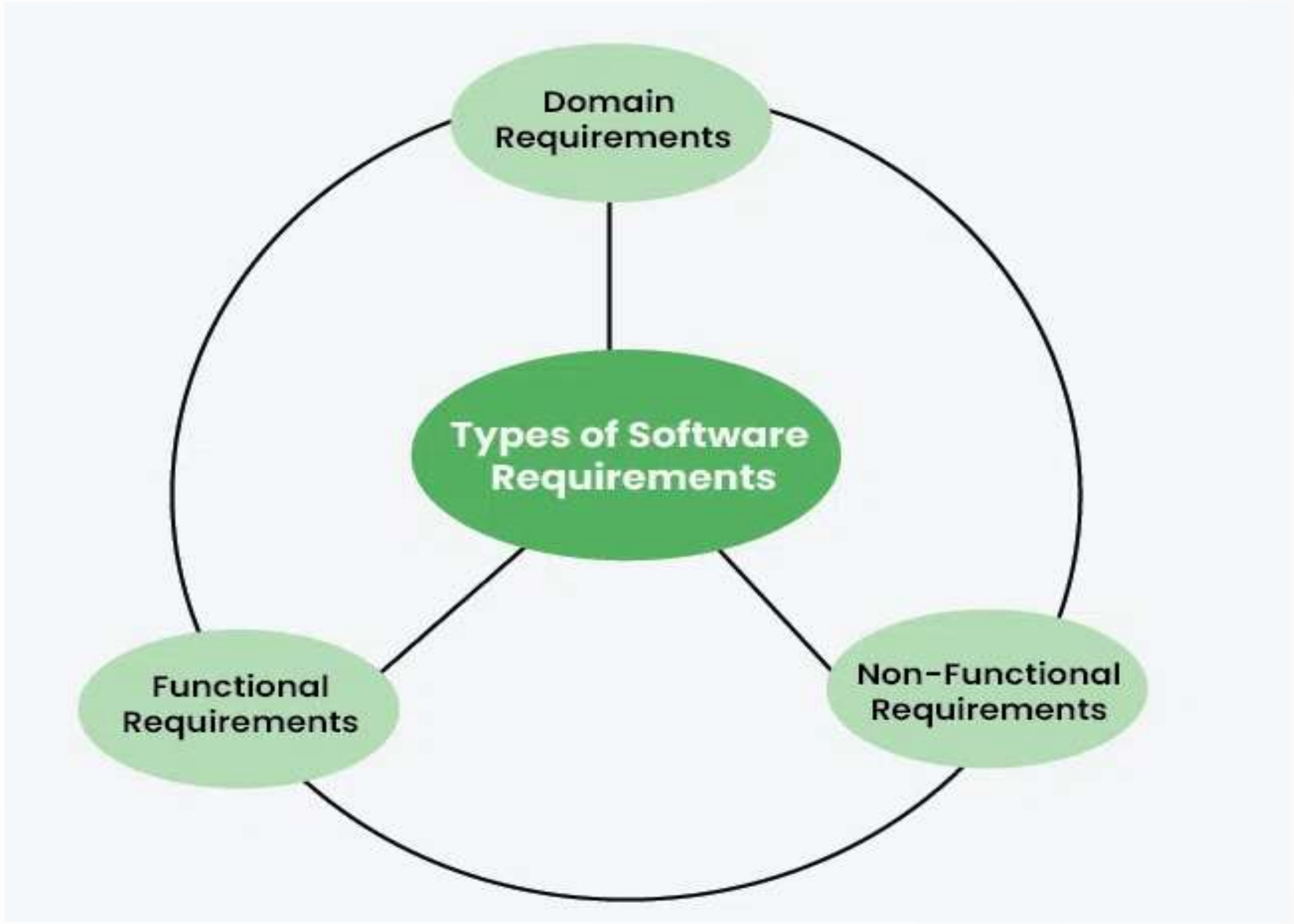
Requirements

A software requirement is a rule the software must conform to: what it must do, how well, and within what constraints or limits.

Types of Requirements

There are two types of requirements:

- Functional
- Non-functional
- Domain



1. Functional requirements

- "A description of a behavior that a system will exhibit under specific conditions".
- For example, "If the user activates the 'log in' button, the login page will appear." Functional requirements answer the question, "What must the software do?"
- These are the requirements that the end user specifically demands as basic facilities that the system should offer.

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?

Functional requirements include:

- *User Requirements:* What the end users expect from the system. These can be high-level and may describe interactions between the system and its users.

Example: "The system should allow users to log in using their email and password.“

- System Requirements: Detailed specifications of what the system must do, usually describing the internal workings or features.

Example: "The system must authenticate users against the database using SHA-256 hashing."

2. Non-functional requirements

- "A description of a property or characteristic that a system must exhibit or a constraint that it must respect"
- For example, "If the user activates the 'log in' button, the login page will appear within 500 milliseconds." This nonfunctional requirement has a characteristic that the system must exhibit: responsiveness. Responsiveness is also called a quality attribute.
- Example about respecting a constraint is, "The GUI toolkit must be able to display non-rectangular windows."

Non-functional requirements include:

They are also called non-behavioral requirements. They deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Example:

- Each request should be processed with the minimum latency?
- System should be highly valuable.

Figure shows a simple example of a design to build a table for a house.

Sketch of house

Two Failed Requirements

Note. This rolling table fails the nonfunctional requirement of fitting through an average door and the functional requirement of having four legs.

Requirement Types

Functional Requirements	Non Functional Requirements
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies "What should the software system do?"	It places constraints on "How should the software system fulfill the functional requirements?"
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory.	It is not mandatory.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing, etc are done.	Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.
Usually easy to define.	Usually more difficult to define.
Example 1) Authentication of user whenever he/she logs into the system. 2) System shutdown in case of a cyber attack. 3) A Verification email is sent to user whenever he/she registers for the first time on some software system.	Example 1) Emails should be sent with a latency of no greater than 12 hours from such an activity. 2) The processing of each request should be done within 10 seconds 3) The site should load in 3 seconds when the number of simultaneous users are > 10000

3. Domain requirements

Domain requirements are specific to the domain or industry in which the software operates. They include terminology, rules, and standards relevant to that particular domain.

Examples:

- Healthcare: The software must comply with HIPAA regulations for handling patient data.
- Finance: The system should adhere to GAAP standards for financial reporting.
- E-commerce: The software should support various payment gateways like PayPal, Stripe, and credit cards.

Domain requirements reflect the unique needs and constraints of a particular industry. They ensure that the software is relevant and compliant with industry-specific regulations and standards.

Requirement Engineering

- The broad spectrum of tasks and techniques that lead to an understanding of requirements is called requirements engineering.

It encompasses seven **distinct tasks/crucial steps**:

1. Inception,
2. Elicitation,
3. Elaboration,
4. Negotiation,
5. Specification,
6. Validation, And
7. Management

1. **Inception:** At project inception,

- you establish a basic understanding of the problem,
- the people who want a solution,
- the nature of the solution that is desired,
- and the effectiveness of preliminary communication and collaboration between the other stakeholders and the software team.

2. **Elicitation:** This phase focuses on gathering the requirements from the stakeholders. One should be careful in this phase, as the requirements are what establishes the key purpose of a project. Understanding the kind of requirements needed from the customer is very crucial for a developer.

Types of Elicitation Techniques:

I. Interviews
II. Surveys and Questionnaires
III. Workshops
IV. Observation
V. Prototyping
VI. Use Cases and User Stories
VII. Document Analysis and Review
VIII. Brainstorming

The following problems can occur in the elicitation phase:

- Problem of Scope
- Problem of Understanding
- Problem of Volatility

3. **Elaboration:** This phase is the result of the inception and elicitation phase.

In the elaboration process, it takes the requirements that have been stated and *gathered in the first two phases and refines them*. Expansion and looking into it further are done as well.

The main task in this phase is to indulge in modeling activities and develop a prototype that elaborates on the features and constraints using the necessary tools and functions.

Prototyping in the Elaboration phase is used to validate key ideas, concepts, and high-risk areas of the system.

4. **Negotiation:** This phase emphasizes discussion and exchanging conversation on what is needed and what is to be eliminated. In the negotiation phase, negotiation is between the developer and the customer and they dwell on how to go about the project with limited business resources. Customers are asked to prioritize the requirements and make guesstimates on the conflicts that may arise along with it. Risks of all the requirements are taken into consideration and negotiated in a way where the customer and developer are both satisfied with reference to the further implementation.

The following are discussed in the negotiation phase:

- Availability of Resources.
- Delivery Time.
- Scope of requirements.
- Project Cost.
- Estimations on development.

5. **Specification:** In the specification phase, the requirements engineer gathers all the requirements and develops a working model. This final working product will be the basis of any functions, features or constraints to be observed. The models used in this phase include *ER (Entity Relationship) diagrams, DFD (Data Flow Diagram), FDD (Function Decomposition Diagrams), and Data Dictionaries.*

A software specification document is submitted to the customer in a language that he/she will understand, to give a glimpse of the working model.

This phase specifies the following:

- Written document.
- A set of models.
- A collection of use cases.
- A prototype.

6. **Validation:** This phase focuses on checking for errors and debugging. In the validation phase, the developer scans the specification document and checks for the following:

- All the requirements have been stated and met correctly
- Errors have been debugged and corrected.
- Work product is built according to the standards.

This requirements validation mechanism is known as the **formal technical review**. The review team that works together and validates the requirements include software engineers, customers, users, and other stakeholders. Everyone in this team takes part in checking the specification by examining for any errors, missing information, or anything that has to be added or checking for any unrealistic and problematic errors.

Some of the validation techniques are the following-

- Requirements reviews/inspections.
- Prototyping.
- Test-case generation.
- Automated consistency analysis.

7. **Requirements Management:** is a set of activities where the entire team takes part in identifying, controlling, tracking, and establishing the requirements for the successful and smooth implementation of the project.

In this phase, the team is responsible for managing any changes that may occur during the project. New requirements emerge, and it is in this phase, responsibility should be taken to manage and prioritize as to where its position is in the project and how this new change will affect the overall system, and how to address and deal with the change. Based on this phase, the working model will be analyzed carefully and ready to be delivered to the customer.

Requirements Documentation

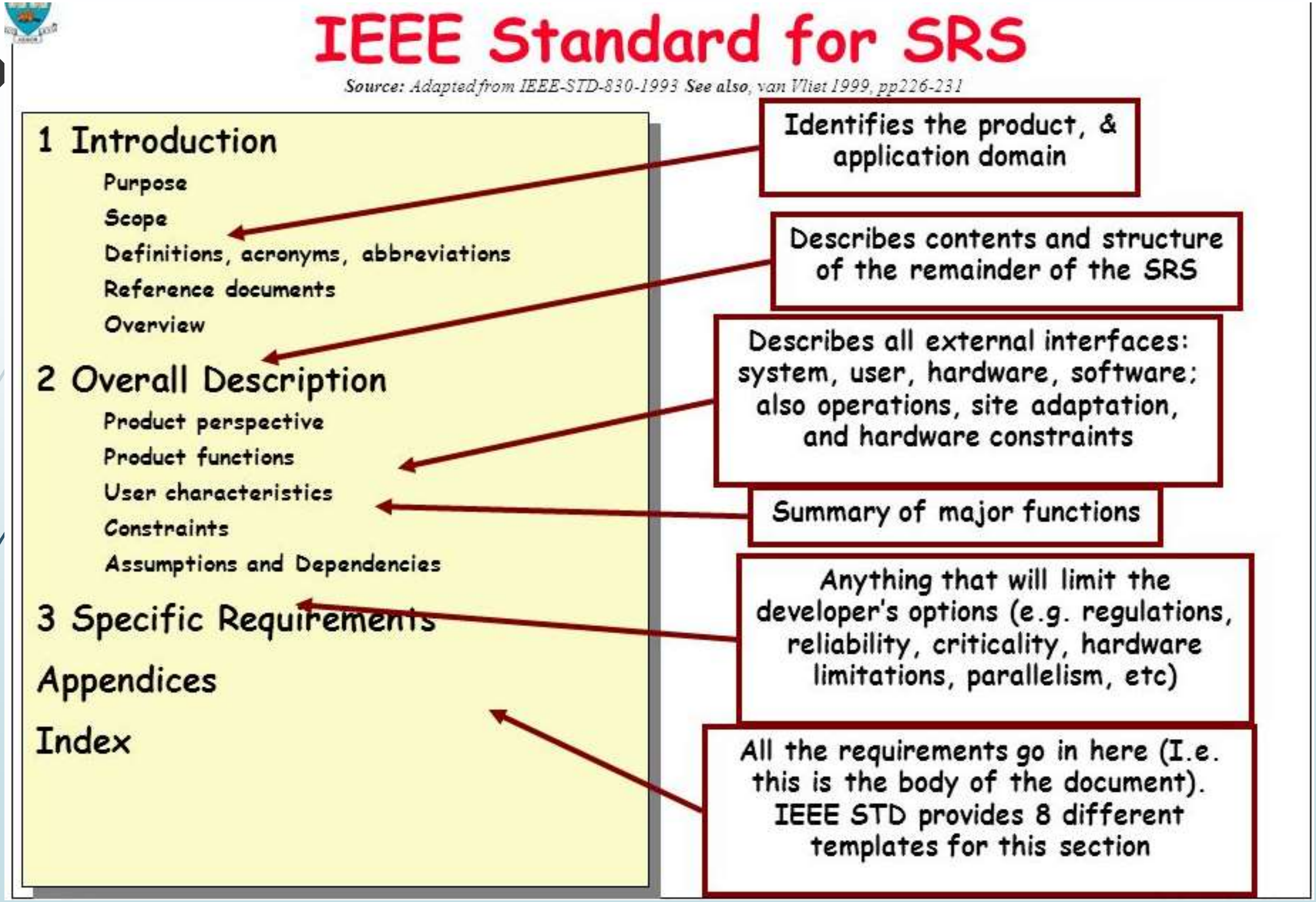
- Requirements Documentation is a detailed record of all the requirements for a software system.
- It serves as a communication tool between stakeholders, including business analysts, developers, designers, and clients.
- The goal is to document what the system must do (functional) and how it must perform (non-functional).
- A well-prepared requirements document helps in preventing misunderstandings, reduces ambiguity, and sets a clear baseline for development.

Understanding the Nature of Software

- The nature of software refers to the inherent characteristics and properties that define software as a product.
- These properties influence how software is developed, maintained, tested, and used.
- Unlike physical products, software is intangible, flexible, and constantly evolving, which brings both advantages and challenges.

Software Requirement Specification (SRS)

- Software Requirement Specification (SRS) is a document that describes the functional and non-functional requirements of a software system.
- It serves as a blueprint for the development team and other stakeholders, outlining what the software should do and how it should behave.



IEEE Standard for SRS

Source: Adapted from IEEE-STD-830-1993 See also, van Vliet 1999, pp226-231

1 Introduction
- Purpose
- Scope
- Definitions, acronyms, abbreviations
- Reference documents
- Overview

2 Overall Description
- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and Dependencies

3 Specific Requirements

Appendices

Index

Identifies the product, & application domain

Describes contents and structure of the remainder of the SRS

Describes all external interfaces: system, user, hardware, software; also operations, site adaptation, and hardware constraints

Summary of major functions

Anything that will limit the developer's options (e.g. regulations, reliability, criticality, hardware limitations, parallelism, etc)

All the requirements go in here (I.e. this is the body of the document). IEEE STD provides 8 different templates for this section

The SRS document contains the following information:

1. **Introduction:** Provides an overview of the software, its purpose, and scope.
2. **Functional Requirements:** Describes the specific functions and features of the software, including inputs, outputs, and behavior.
3. **Non-Functional Requirements**: Specifies the quality attributes of the software, such as performance, security, and usability.
4. **Design Constraints:** Specifies any limitations or restrictions that apply to the software.
5. **Performance Requirements**
6. **System Features:** Provides a detailed description of each feature or module of the software.
7. **User Stories or Use Cases:** Describes the interactions between the software and its users, including specific scenarios and workflows.
8. **Assumptions and Dependencies:** Lists any assumptions made during the requirements gathering process and dependencies on other systems or components.
9. **Preliminary Schedule and Budget**

Thank You