# UNIT :2

# Requirement Analysis and Specification

Prepared By: Prof. Suhag Baldaniya

# Understanding the Requirement

- **Requirements** analysis, also called **requirements engineering**, is the process of determining user expectations for a **new** **or** **modified** product.
- These features, called **requirements**, must be quantifiable, relevant and detailed.
- In **software engineering**, such **requirements** are often called functional specifications.
- The goal of requirement engineering is to develop and maintain sophisticated and descriptive SRS (System Requirements Specification) document.
- The process to gather the software requirements from client, analyze and document them is known as requirement engineering.

## Understanding the Requirement (Cont…)

- A focused and detailed business requirements analysis can help you avoid problems like these.
- This is the process of discovering, analyzing, defining, and documenting the requirements that are related to a specific business objective.
- And it's the process by which you clearly and precisely define the scope of the project, so that you can assess the timescales and resources needed to complete it.
- **Remember**: to get what you want, you need to accurately define it – and a good business requirements analysis helps you achieve this objective.

## Understanding the Requirement (Cont…)

- Software requirements can be broadly classified into two groups:
  - **Functional** *or* **problem domain requirements**
  - **Non-functional** *or* **solution domain requirements**
- In a problem domain, the focus is on the functional or business requirements.
- It is recommended that you create a domain model of your functional requirements before you start thinking of the solution domain.
- In a solution domain, we focus on how to deliver the solution for functional or business requirements.

## Understanding the Requirement (Cont…)

- **Some of the important non-functional requirements of an application are**:

→Quality attributes such as security, high availability, scalability, performance, and reliability.

→ Message format, transport, and protocol

→ Data format and transformation

→ User interface

→ Integration

→ Runtime infrastructure

→ Networking and communication

→ Constraints such as the use of specific RDBMS System, protocols, and standards

## Requirement Engineering Process

**It is a four step process, which includes –**

1. Feasibility Study
2. Requirement Gathering
3. Software Requirement Specification
4. Software Requirement Validation

## Requirement Modeling

- Requirements modeling in software engineering is essentially the planning stage of a software application or system.
- Generally the process will begin when a business or an entity, for example an educational institution, approaches a software development team to create an application or system from scratch or update an existing one.
- Requirements modeling comprises several stages or '**patterns**': scenario-based modeling, data modeling, flow-oriented modeling, class-based modeling and behavioral modeling.

## Requirement Modeling (Cont...)

- Each of these stages/patterns examines the same problem from a different perspective.
- The appropriate approach to take will depend on the type of system and the organizational standards, in some cases they are domain specific modeling languages which are used.
- **Following are the requirement modeling strategies**:
  1. Flow Oriented Modeling
  2. Class-based Modeling

## Functional vs. Non-Functional Requirements

- A **functional requirement** specifies something that the application or system should do.
- Often this is defined as a behavior of the system that takes input and provides output.
- For example, a traveler fills out a form in an airline's mobile application with his/her name and passport details (input), submits the form, and the application generates a boarding pass with the traveler's details (output).
- **Non-functional requirements**, sometimes also called quality requirements, describe how the system should be, as opposed to what it should do.
- Non-functional requirements of a system include performance (e.g. response time), maintainability, and scalability, among many others.

## Requirement Modeling (Cont…)

### 1. Flow Oriented Modeling

- It shows how data objects are transformed by processing the function.
- **The Flow oriented elements are:**
  **i. Data flow model**
- It is a graphical technique. It is used to represent information flow.
- The data objects are flowing within the software and transformed by processing the elements.
- The data objects are represented by labeled arrows.
- Transformation are represented by circles called as bubbles.
- DFD shown in a hierarchical fashion. The DFD is split into different levels. It also called as 'context level diagram'.

## Requirement Modeling (Cont…)

ii. Control flow model :

- Large class applications require a control flow modeling.
- The application creates control information instated of reports or displays.
- The applications process the information in specified time.
- An event is implemented as a Boolean value. **For example,** the Boolean values are true or false, on or off, 1 or 0.

iii. Control Specification :

- A short term for control specification is CSPEC.
- It represents the behavior of the system.
- The state diagram in CSPEC is a sequential specification of the behavior.
- The state diagram includes states, transitions, events and activities.
- State diagram shows the transition from one state to another state if a particular event has occurred.

# Requirement Modeling (Cont…)

## iv. Process Specification

- A short term for process specification is PSPEC.
- The process specification is <u>used to describe all flow model processes</u>.
- The content of process specification consists text, Program Design Language(PDL) of the process algorithm, mathematical equations, tables or UML activity diagram.

## Requirement Modeling (Cont...)

- Class based modeling represents the object. The system manipulates the operations.
- The elements of the class based model consist of classes and object, attributes, operations, class – responsibility – collaborator (CRS) models.
- <u>Classes </u>:
- Classes are determined using underlining each noun or noun clause and enter it into the simple table.
- **Classes are found in following forms:**
- **External entities:** The system, people or the device generates the information that is used by the computer based system.
- **Things**: The reports, displays, letter, signal are the part of the information domain or the problem.
- **Occurrences or events:** A property transfer or the completion of a series or robot movements occurs in the context of the system operation.

## Requirement Modeling (Cont…)

- **Roles**: The people like manager, engineer, salesperson are interacting with the system.
- **Organizational units**: The division, group, team are suitable for an application.
- **Places**: The manufacturing floor or loading from the context of the problem and the overall function of the system.
- **Structures:** The sensors, computers are defined a class of objects or related classes of objects.
- **Attributes :**
- Attributes are the set of data objects that are defining a complete class within the context of the problem.
- **For example,** 'employee' is a class and it consists of name, Id, department, designation and salary of the employee are the attribute.

## Requirement Specification (SRS)

- The requirements are specified in specific format known as **SRS**.
- This document is created before starting the development work.
- The software requirement specification is an official document.
- It shows the detail about the performance of expected system.
- SRS indicates to a developer and a customer what is implemented in the software.
- SRS is useful if the software system is developed by the outside contractor.
- SRS must include an interface, functional capabilities, quality, reliability, privacy etc.
- Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function.

## Requirement Specification (SRS)(Cont…)

- This is achieved through detailed and continuous <u>communications with the project team and customer throughout the software development process</u>.
- **Why SRS ?**
- In order to fully understand one's project, it is very important that they come up with a SRS listing out <u>their requirements, how are they going to meet it and how will they complete the project</u>.
- It helps the team to save their time as they are <u>able to comprehend how are going to go about the project</u>.
- Doing this also enables the team to find out about the limitations and risks early on.

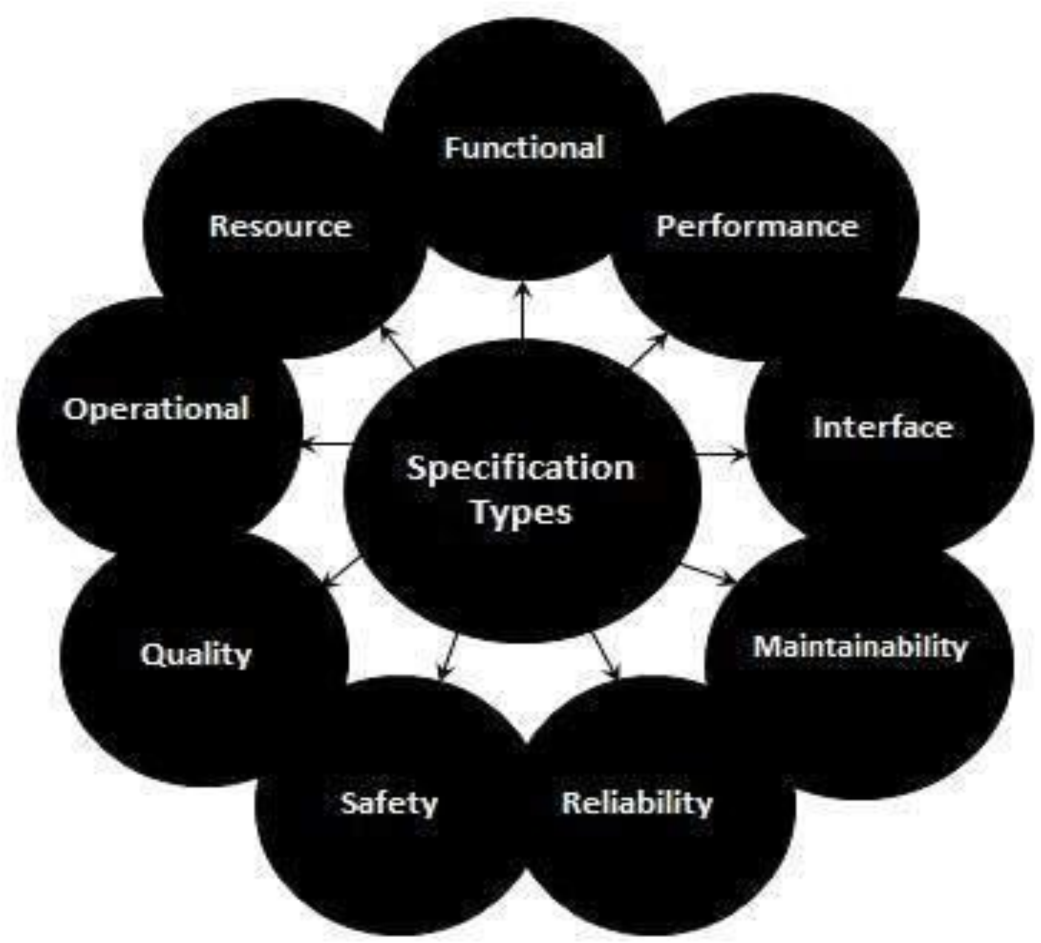
## The specific goals of the SRS are:

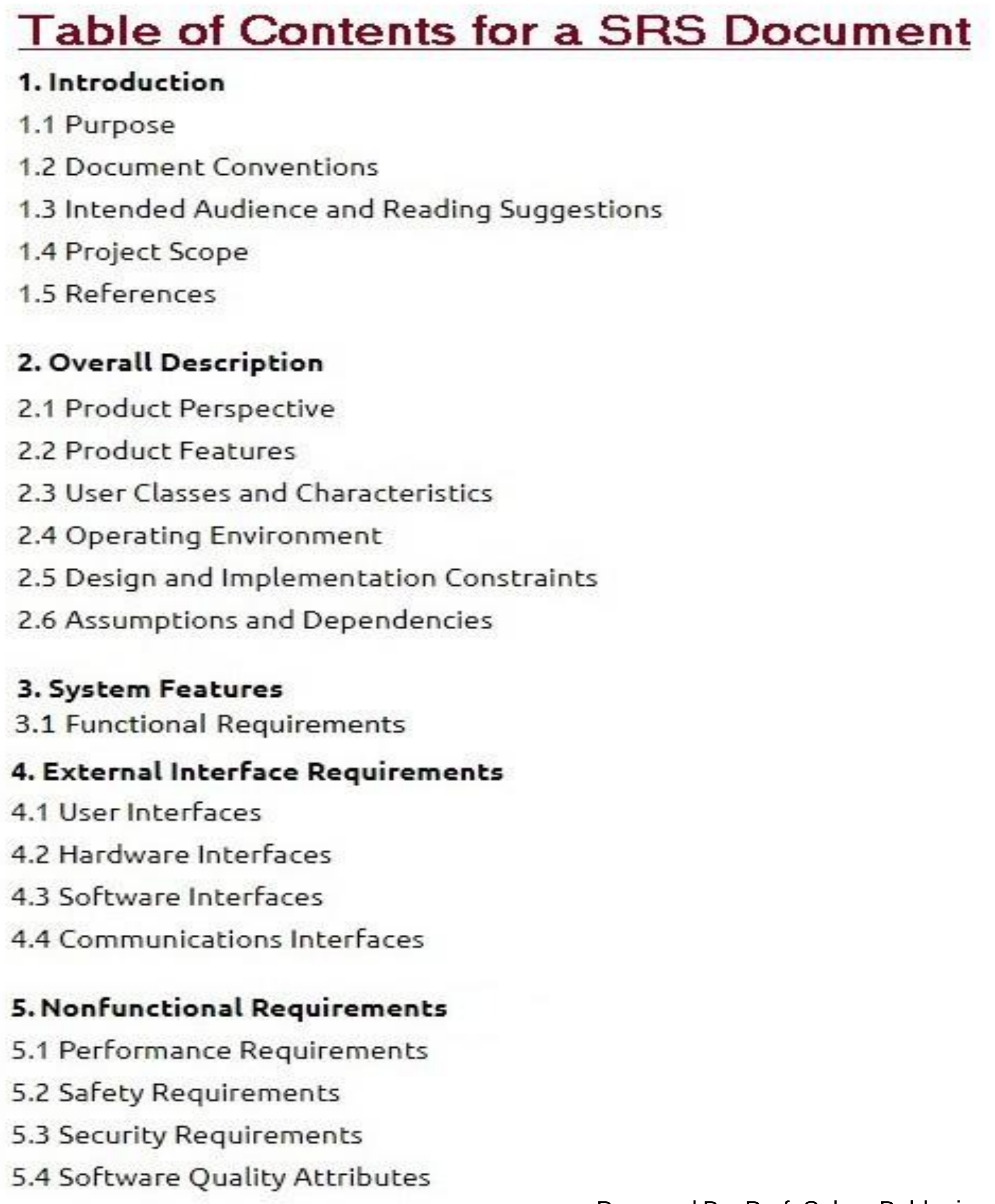
- Facilitating reviews.
- Describing the scope of work.
- Providing a reference to software designers (i.e. navigation aids, document structure).
- Providing a framework for testing primary and secondary use cases.
- Including **features** to customer requirements.
- Providing a platform for ongoing refinement (via incomplete specs or questions).

## Characteristics of SRS

- The SRS should be complete and consistence.
- The modification like logical and hierarchical must be allowed in SRS.
- The requirement should be easy to implement.
- Each requirement should be uniquely identified.
- The statement in SRS must be unambiguous means it should have only one meaning.
- All the requirement must be valid for the specified project.

# Types of Requirements





## Table of Contents for a SRS Document

**1. Introduction**

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Project Scope

1.5 References

**2. Overall Description**

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 Assumptions and Dependencies

**3. System Features**

3.1 Functional Requirements

**4. External Interface Requirements**

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4.4 Communications Interfaces

**5. Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

## Requirement Analysis and Requirement Elicitation

- Analyst, requirement analysis is the most important part of any Job. It will **help you determining the actual needs of stakeholders**.
- At the same time, enable you to communicate with the stakeholders in a language they understand (like charts, models, flow-charts,) instead of complex text.

**Here are the main activities involve in requirement analysis:**

- Identify customer's needs.
- Evaluate system for feasibility.
- Perform economic and technical analysis.
- Allocate functions to system elements.
- Establish schedule and constraints.
- Create system definitions.

## Requirement Analysis and Requirement Elicitation (Cont…)

- Requirements analysis is critical to the success or failure of a systems or software project.
- The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

## Requirement Analysis and Requirement Elicitation (Cont…)

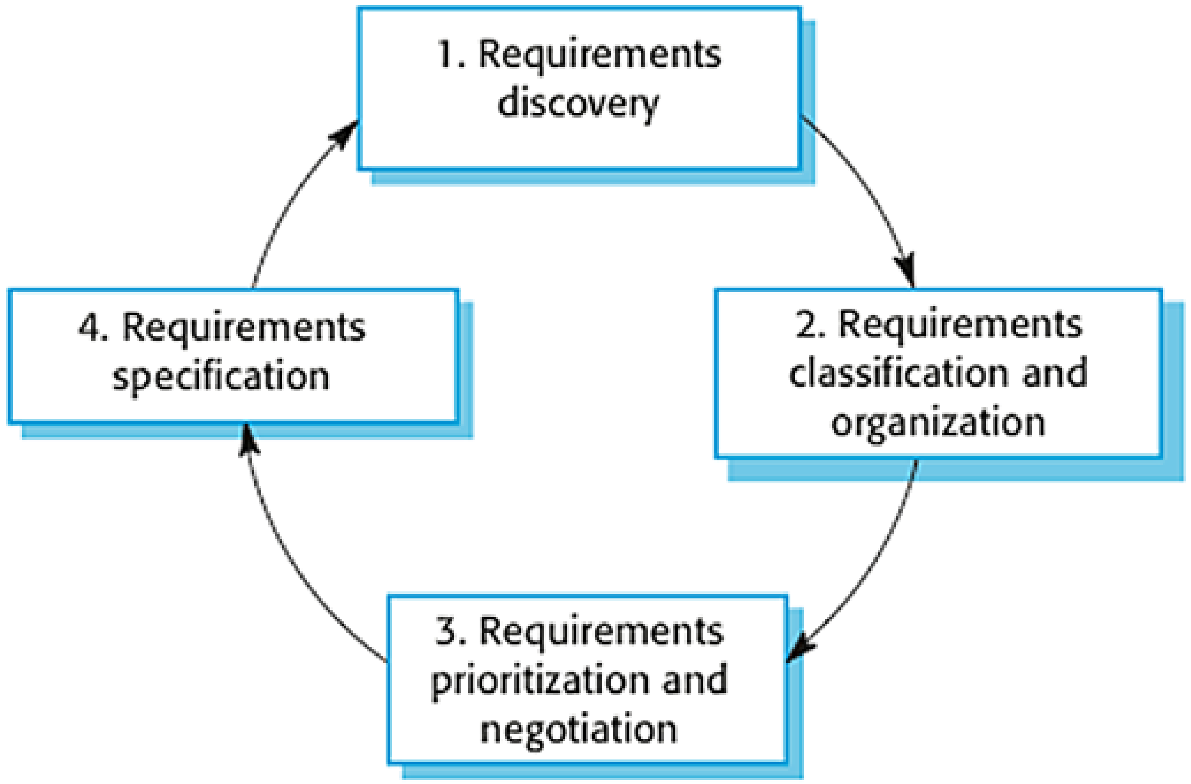

1. Requirements discovery
2. Requirements classification and organization
3. Requirements prioritization and negotiation
4. Requirements specification

**Figure** : Requirement Analysis and Requirement Elicitation

## Requirement Elicitation

- Requirement elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders.
- This practice is also sometimes referred to as "Requirement Gathering".
- **Requirements elicitation** is a part of the **requirements** engineering process, usually followed by **analysis** and specification of the **requirements.**
- **Requirements elicitation** practices include interviews, questionnaires, user observation, workshops, brainstorming, use cases, role playing and prototyping.
- Before **requirements** can be analyzed, modeled, or specified they must be gathered through an **elicitation** process.

## Requirement Elicitation (Cont…)

- **Requirements elicitation** is perhaps the most difficult, most error-prone and most communication intensive software development.
- <u>It can be successful only through an effective customer-developer partnership</u>. It is needed to know what the users really need.

**There are a number of requirements elicitation methods.**

**Few of them are listed below –**

1. Interviews
2. Brainstorming Sessions
3. Facilitated Application Specification Technique (FAST)
4. Quality Function Deployment (QFD)
5. Use Case Approach

## Requirement Engineering

- **Requirements** analysis, also called **requirements engineering**, is the process of determining user expectations for a new or modified product.
- These features, called **requirements**, must be quantifiable, relevant and detailed.
- In software **engineering**, such **requirements are** often called functional specifications.

## Why need Requirement Engineering ?

- The **Requirement Engineering** (RE) is the most **important** phase of the Software Development Life Cycle (SDLC).
- Therefore the **importance** of **Requirement Engineering** is enormous to develop effective software and in <u>reducing software errors at the early stage of the development of software.</u>
- **Requirement,** In product development and **process** optimization, a **requirement** is a singular documented physical or functional need that a particular design, product or **process** aims to satisfy.

## Requirement Engineering (Cont...)

- The process to gather the software requirements from client, analyze and document them is known as requirement engineering.
- The goal of requirement engineering is to develop and maintain sophisticated and descriptive '**System Requirements Specification**' document.
- Requirement Engineering Process

**It is a four step process, which includes** –

1. Feasibility Study
2. Requirement Gathering
3. Software Requirement Specification
4. Software Requirement Validation

## Requirement Engineering (Cont…)

### 1. Feasibility study :

- When the client approaches the organization for getting the desired product developed, it comes up with **rough idea** about what all functions the software must perform and which all features are expected from the software.
- Referencing to this information, the analysts does a detailed study about whether the desired system and its functionality are feasible to develop.
- This feasibility study is focused towards goal of the organization.
- This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization.

## Requirement Engineering (Cont…)

### 2. Requirement Gathering :

- If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user.
- Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

### 3. Software Requirement Specification :

- <u>**SRS** is a document created by system analyst after the requirements are collected from various stakeholders</u>.
- SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

## Requirement Engineering (Cont…)

### 4. Software Requirement Validation :

- After requirement specifications are developed, the requirements <u>mentioned in this document are validated</u>.
- User might ask for illegal, impractical solution or experts may <u>interpret the requirements incorrectly</u>.
- This results in huge increase in cost if not nipped in the bud.
- Requirements can be checked against following conditions –

→ If they can be practically implemented

→ If they are valid and as per functionality and domain of software

→ If there are any ambiguities (complex)

→ If they are complete

→ If they can be demonstrated

Thank You