

# Software Requirement Analysis And Specification

---

- SRS is the medium through which the client and the user needs are accurately specified.
- Software requirement is the description and specification of a system.
- SRS is a document that completely describes WHAT the proposed system should do without describing HOW the software will do it.
- The basic goal of requirement phase is to produce the SRS, which describes the complete external behavior of the proposed software.
- A high quality SRS is a pre-requisite to high quality software.
- A high quality SRS reduces the development cost

## Characteristics of good SRS



## **Correctness:**

User review is used to provide the accuracy of requirements stated in the SRS.

SRS is said to be perfect if it covers all the needs that are truly expected from the system.

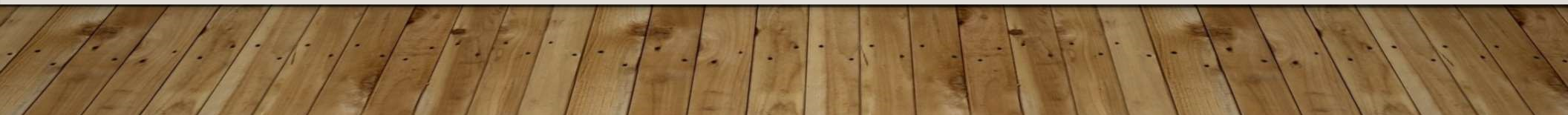
**Completeness:** The SRS is complete if, and only if, it includes the following elements:

- (1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.
- (2). Definition of their responses of the software to all realizable classes of input data in all available categories of situations.
- (3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

## **Consistency:**

The SRS is consistent if, and only if, no subset of individual requirements described in it conflict.

There are three types of possible conflict in the SRS:



- ). The specified characteristics of real-world objects may conflict. For example,
- (a) The format of an output report may be described in one requirement as tabular and in another as textual.
  - (b) One condition may state that all lights shall be green while another states that all lights shall be blue.
- ). There may be a reasonable or temporal conflict between the two specified actions. For example,
- (a) One requirement may determine that the program will add two inputs, and another may determine that the program will multiply them.
  - (b) One condition may state that "A" must always follow "B," while another requires that "A and B" co-occurs.
- ). Two or more requirements may define the same real-world object but use different terminology for that object.
- For example, a program's request for user input may be called a "prompt" in one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.





## **Unambiguousness:**

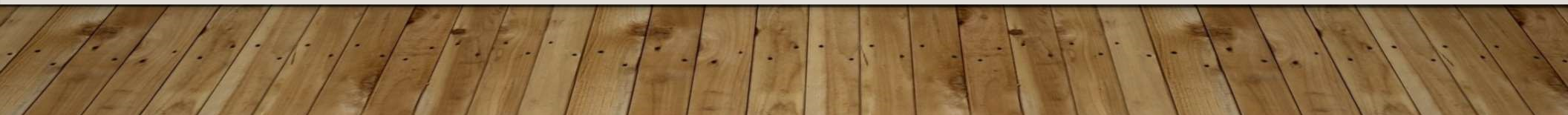
- SRS is unambiguous when every fixed requirement has only one interpretation.
- This suggests that each element is uniquely interpreted.
- In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

## **Ranking for importance and stability:**

- The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.

## **Modifiability:**

- SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent.
- Modifications should be perfectly indexed and cross-referenced.



## Verifiability:

- SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements.
- The requirements are verified with the help of reviews.

## Traceability:

- The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.
- **Backward Traceability:** This depends upon each requirement explicitly referencing source in earlier documents.
- **2. Forward Traceability:** This depends upon each element in the SRS having a unique name or reference number.

## Design Independence:

- There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.



## **1. Testability:**

- An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

## **2. Understandable by the customer:**

- An end user may be an expert in his/her explicit domain but might not be trained in computer science.
- Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

## **3. The right level of abstraction:**

If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas, for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.



# What Should The SRS Address

---

- The basic issues that the SRS writer(s) shall address are the following:
- **Functionality:** What is the software supposed to do?
- **External interfaces:** How does the software interact with people, the system's hardware, other hardware and other software
- **Performance:** What is the speed, availability, response time, recovery time of various software functions etc.
- **Attributes:** What are the portability, correctness, maintainability, security etc. considerations?
- **Design constraints imposed on an implementation:** Are there any required standards, effect, implementation language, policies for database integrity, resource limits, operating environments etc.



# Definition of Requirement

---

- Requirement is the descriptions and specifications of a system
- Software requirements analysis is necessary to avoid creating a software product that fails to meet the customer's needs
- Data, functional, and behavioral requirements are elicited from the customer and refined to create a specification that can be used to design the system
- Software requirements work products must be reviewed for clarity, completeness and consistency

# Types Of Requirements

---

## System requirements

- A structured document setting out detailed descriptions of the system services.
- Written as a contract between client and contractor.

# Software Specification

---

- A detailed software description which can serve as a basis for a design or implementation.
- Written for developers

# User Requirements

---

- Statements in natural language + diagram of the services the system provides and its operational constraints. written for customers
- Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge
- User requirements are defined using natural language, tables and diagram



# Problems With Natural Language

---

## **Ambiguity –**

- Readers and writers may not interpret words in the same way

## **Over-flexibility –**

- The same thing may be expressed in a number of different ways

## **Requirements amalgamation & confusion –**

- Several different requirements may be expressed together; functional and non-functional requirements may be mixed together

## **Lack of modularization –**

- NL structures are inadequate to structure system requirements



# Domain Requirements

---

- Requirements that come from the application domain of the system and that reflect characteristics of that domain
- May be new function requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

# Domain Requirement Issue

---

## Understandability

- Requirements are expressed in the language of the application domain – this is often not understood by software engineers developing the system

## Implicitness

- Domain specialists understand the area so well that they do not think of making the domain requirements explicit



# Functional And Non-functional Requirements

---

## Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations
- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail



# Non-Functional Requirements

---

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards
- Define system properties and constraints eg. reliability, response time and storage requirements. Constraints are I/O device capability, system representations etc
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.
- Non-functional requirements define the overall qualities or attributes of the resulting system

# CONTD...

---

## **Product requirements –**

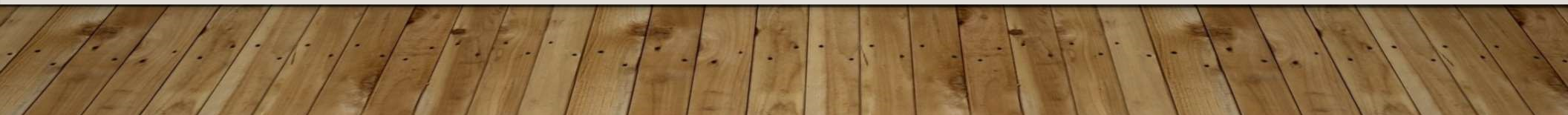
- Requirements which specify that the delivered product must behave in a particular way, e.g. execution speed, reliability etc.

## **Organizational requirements –**

- Requirements which are a consequence of organizational policies and procedures, e.g. process standards used, implementation requirements etc.

## **External requirements –**

- Requirements which arise from factors which are external to the system and its development process, e.g. interoperability requirements, legislative requirements

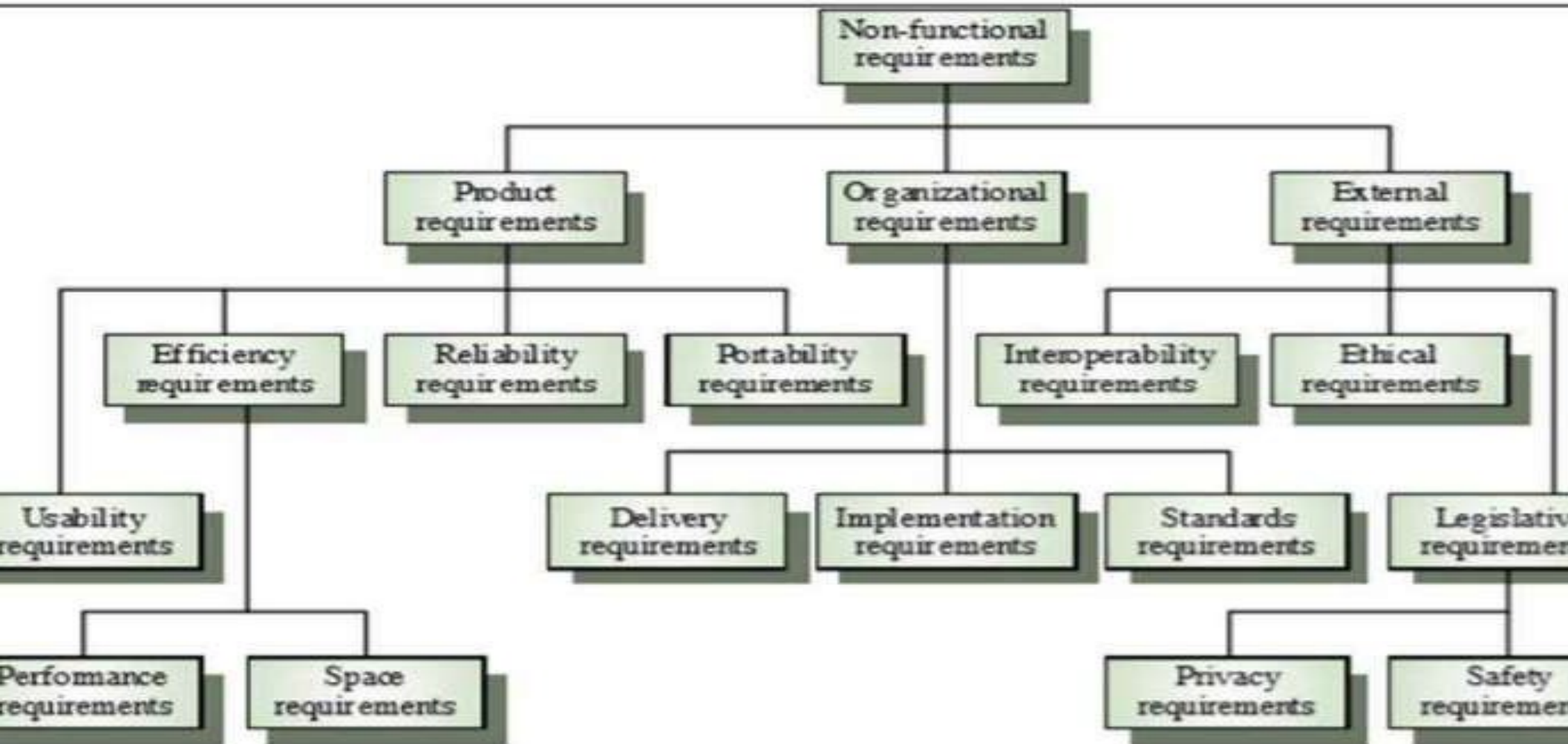


# These May Include

---

- Performance requirements
- Safety requirements
- Security requirements
- Usability requirements
- Scalability requirements

# Non-functional requirement types





# Requirement Engineering

---

- **RE** refers to the process of defining, documenting, and maintaining requirements in the engineering design process.
- Requirements Engineering provides the appropriate mechanism for understanding what the customer wants, analyzing the need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification and managing the requirements as they are transformed into a operational system
- Requirement Engineering is the disciplined application of proven principles, methods, tools, and notations to describe a proposed system's intended behavior and its associated constraints.
- SRS may act as a contract between developer and customer

# CONTD...

---

- It is the processes used to discover, analyze and validate system requirements.
- The processes used for RE vary widely depending on the application domain, the people involved and the organization developing the requirements. However, there are a number of generic activities common to all processes can be described in six distinct steps

**Step 1: Requirements elicitation**

**Step 2: Requirements analysis and negotiation**

**Step 3: Requirements specification**

**Step 4: System modeling**

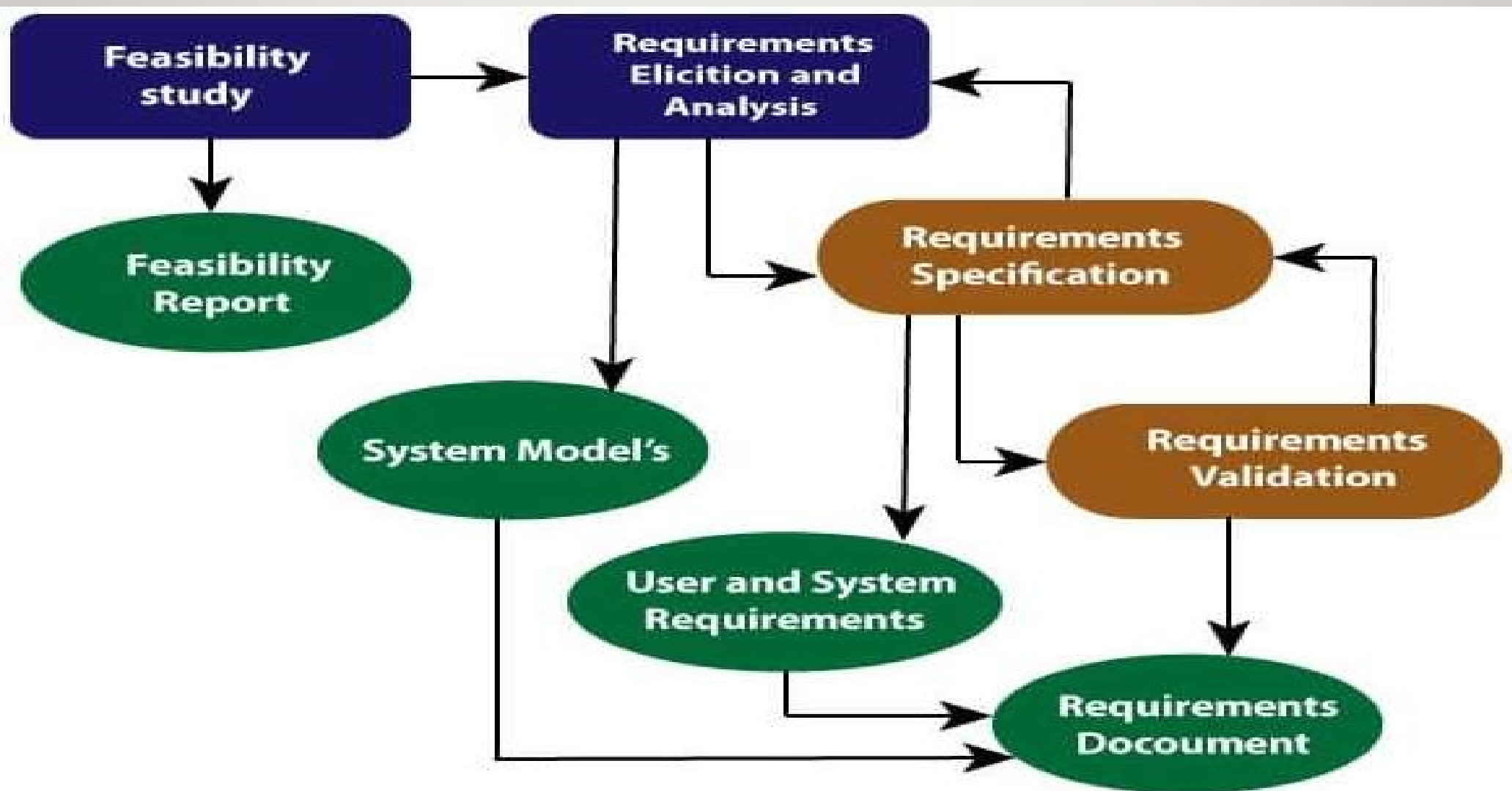
**Step 5: Requirements validation**

**Step 6: Requirements management**

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

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management





**Requirement Engineering Process**

# Requirements Elicitation

---

- It is the practice of obtaining the requirements of a system from users, customers and other stakeholders.
- The practice is also sometimes referred to as Requirement gathering.
- Find out from customers what the product objectives are, what is to be done, how the product fits into business needs, and how the product is used on a day to day basis.

# I. Software Requirements Elicitation

---

- Customer meeting are the most commonly used technique
- Use context free questions to find out customer's goals and benefits identify stakeholders gain understanding of problem, determine customer reactions to proposed solutions and assess meeting effectiveness
- If many users are involved, be certain that a representative cross-section of user is interviewed

## 2. Facilitated Action Specification Techniques(FAST)

---

- Meeting held at neutral site, attended by both software engineers and customers
- Rules established for preparation and participation
- Agenda suggested to cover important points and to allow for brainstorming
- Meeting controlled by facilitator(customer, developer or outsider)
- Definition mechanism(flip charts, stickers, electronic device etc) is used
- Goal is to identify problem, propose elements of solution, negotiate different approaches and specify a preliminary set of solution requirements



# Requirements Elicitation and Analysis Techniques

---

- Interviews
- Questionnaires
- Brain storming
- Use cases Approach
- User observation
- Workshops
- role playing
- prototyping



# Problem Of Requirement Elicitation

---

## Problems of scope:

- The boundary of system is ill-defined. or unnecessary details are provided

## Problems of understanding:

- The users are not sure of what they need and don't have full understanding of the problem domain

## Problem of volatility:

- The requirements change overtime

**Not having good skills to collect information and cannot understand the group problem.**

**Not well defined the actual Problem.**

**Not focus on the requirement of the system but more on the design which is useless on this stage.**

**The requirement should be changeable according to time.**

**Lack of analyst knowledge with the problem.**

**Lack of user's knowledge also creates problems for elicitation.**

**Problem with understanding the language between user and analyst.**

**Ignoring or omitting the actual problem.**

**The boundary of the problem should be well defined**

# **Guidelines of Requirements Elicitation**

Assess the business and technical feasibility for the proposed system

Identify the people who will help specify requirements.

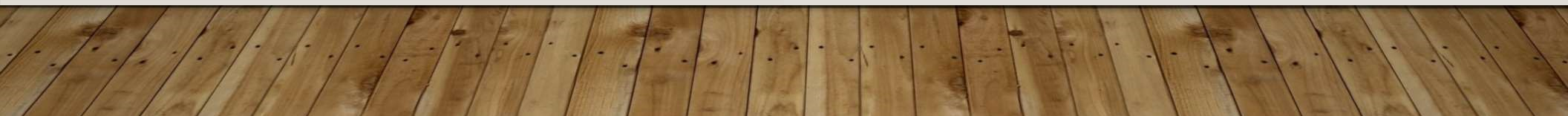
Define the technical environment (e.g. computing architecture, operating system, telecommunication needs) into which the system or product will be placed

Identify “domain constraints” (i.e. characteristics of the business environment specific to the application domain) that limit the functionality or performance of the system or product to build

Define one or more requirements elicitation methods (e.g. interviews, team meetings, ..etc)

Solicit participation from many people so that requirements are defined from different point of views.

Create usage scenarios of use cases to help customers/users better identify key requirements.



# Interviews

---

- Objective of conducting an interview is to understand the customer's expectations from the software.
- It is impossible to interview every stakeholder hence representatives from groups are selected based on their expertise and credibility.
- Interviews may be open-ended or structured.
- In open-ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
- In structured interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.



# Brainstorming Sessions

---

- It is a group technique
- It is intended to generate lots of new ideas hence providing a platform to share views
- A highly trained facilitator is required to handle group bias and group conflicts.
- Every idea is documented so that everyone can see it.
- Finally, a document is prepared which consists of the list of requirements and their priority if possible.



# Use-Case Diagram

---

- This technique combines text and pictures to provide a better understanding of the requirements.
- A diagram that depicts the interactions between the system and external systems and users.
- It graphically describes who will use the system and in what ways the user expects to interact with the system
- A use case diagram shows the relationships between actors and their interactions with a system
- The use cases describe the 'what', of a system and not 'how'. Hence, they only give a functional view of the system.
- The components of the use case design includes three major things – Actor, Use cases, and Use case diagram.

# Use Case Symbols

---

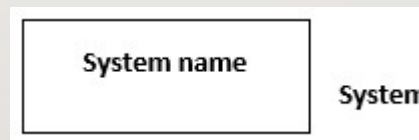
- **Use case-** subset of the overall system functionality. Represented graphically by an oval with the name of the use case inside the oval
- **Actor-** anything that needs to interact with the system to exchange information. Could be a human, an organization, another information system, an external device.
- Actor communicate by sending and receiving stimuli to and from the system. Each actor has name
- **Use case diagram –**  
A use case diagram graphically represents what happens when an actor interacts with a system. It captures the functional aspect of the system.
  - A stick figure is used to represent an actor.
  - An oval is used to represent a use case.
  - A line is used to represent a relationship between an actor and a use case.

# Basic Use Case Diagram Symbols And Notations

---

## System

- Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



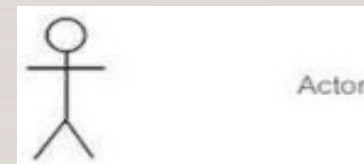
## Use Case

- Draw use cases using ovals. Label the ovals with verbs that represent the system's functions







## Actor

- Actors are the users of a system. Actors interact with the system by receiving or sending flow of information.



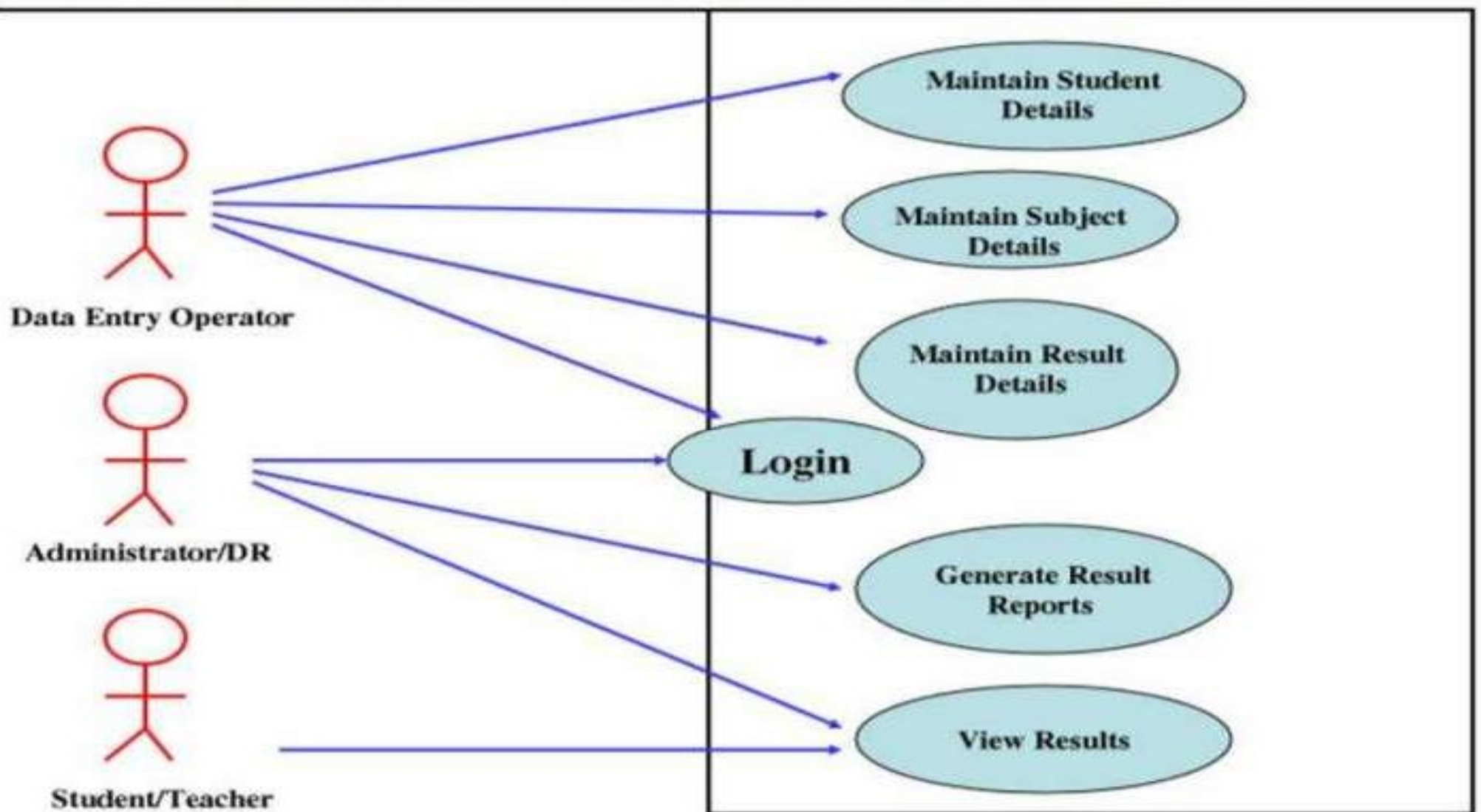
## Relationships

- Illustrate relationships between an actor and a use case with a simple line.
- For relationships among use cases, use arrows labeled either "uses" or "extends".
- A "uses" relationship indicates that one use case is needed by another in order to perform a task.
- An "extends" relationship indicates alternative options under a certain use case

Relationship	Symbol	Description
Communicates		An actor is connected to a use case using a line with no arrowheads
Includes		A use case contains a behavior that is common to more than one other use case. The arrow points to the common use case.
Extends		A different use case handles exceptions from the basic use case. The arrow points from the extended to the basic use case.
Generalizes		One UML "thing" is more general than another "thing." The arrow points to the general "thing."



## Use case diagram for Result Management System



# Purpose Of Use Case Diagram

---

- Use case diagrams are typically developed in the early stage(Analysis phase) of development and people often apply use case modeling for the following purposes:
  - Specify the context of a system
  - Capture the requirements of a system
  - Validate a systems architecture
  - Drive implementation and generate test cases
  - Developed by analysts together with domain experts

# Benefits Of Use-Case Modeling

---

- provides a tool for capturing functional requirements
- Assists in decomposing system scope into more manageable pieces
- Provides a means of communicating with users and other stakeholders concerning system functionality in a language that is easily understood
- Provides a means of identifying, assigning, tracking, controlling, and management system development activities, especially incremental and iterative development
- provides an aid in estimating project scope, effort, and schedule.
- Provides a baseline for testing in terms of defining test plans and test cases.
- Provides a baseline for user help systems and manuals as well as system development documentation.
- Provides a tool for requirements traceability.

# CONTD...

---

- Provides a starting point for the identification of data objects or entities.
- Provides functional specifications for designing user and system interfaces
- Provides a means of definition databases access requirements
- Provides a framework for driving the system development project

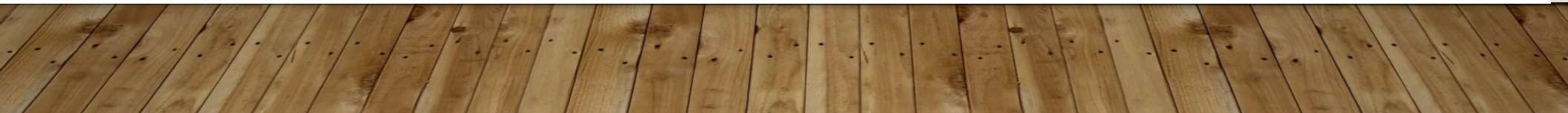


# Scenarios

Scenarios are real-life examples of how a system can be used.

They should include

- A description of the starting situation;
- A description of the normal flow of events;
- A description of what can go wrong;
- A description of the state when the scenario finishes.





# Ethnography

- Getting information by observation
- One of the goals is to use the viewpoint of the people within the system
- The terms emic and etic are used
- An etic view of the system is the 'outside view' or what the analyst see
- An emic view is the inside view or what the system users see
- The main characteristics of ethnographic studies are:
  - Analysts observe or possible even participate in such activities
  - Any interviews are conducted in situ, possibly as informal discussion rather than formal interview
  - There is emphasis on examining the interaction between users
  - There is emphasis on tracing communication links and
  - there is detailed analysis of artifacts



# Interviewing

---

- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
  - Closed interviews where a pre-defined set of questions are answered.
  - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders

# Software Requirement Specification(SRS):

---

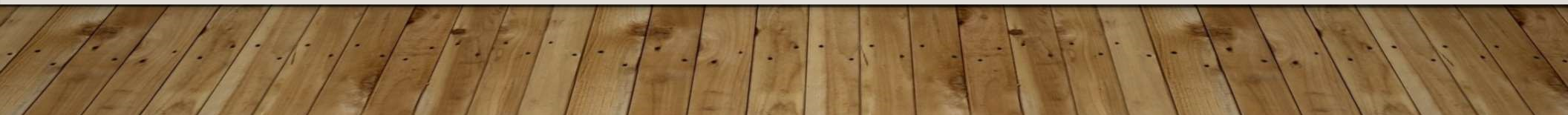
- Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language.
- It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.
- The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc
- A requirements specification for a software system- is a complete description of the behavior of system to be developed
- It includes a set of use cases that describe all the interactions the users will have with the software.
- In addition to use cases, the SRS also contains non-functional requirements

## 1. Data Flow Diagrams:

- Data Flow Diagrams (DFDs) are used widely for modeling the requirements.
- DFD shows the flow of data through a system.
- The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding.
- The DFD is also known as a data flow graph or bubble chart.

## 2. Data Dictionaries:

- Data Dictionaries are simply repositories to store information about all data items defined in DFDs.
- At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.



### 3. Entity-Relationship Diagrams:

- Another tool for requirement specification is the entity-relationship diagram, often called an **"E-R diagram."**
- It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.



# Requirements Validation

---

- The process of ensuring the specified requirements meet the customer needs.
- It is Requirements Quality Control.
- It's concerned with finding problems with the requirements.
- Requirements validation is done to make sure that requirements are complete and consistent according to user requirements.
- The requirements validation process detects errors in the software requirements specification (SRS).
- Ambiguities and conflicts in requirements are resolved during requirements validation.
- During the requirements validation process, different types of checks should be carried out on the requirements

**It checks for validity, consistency, completeness, realism, verifiability.**

- **Validity checks:** The functions proposed by stakeholders should be aligned with what the system needs to perform. You may find later that there are additional or different functions are required instead.
- **Consistency checks:** Requirements in the document shouldn't conflict or different descriptions of the same function
- **Completeness checks:** The document should include all the requirements and constraints
- **Realism checks:** Ensure the requirements can actually be implemented using the knowledge of existing technology, the budget, schedule, etc.
- **Verifiability:** Requirements should be written so that they can be tested. This means you should be able to write a set of tests that demonstrate that the system meets the specified requirements.



# Requirements Validation Techniques

---

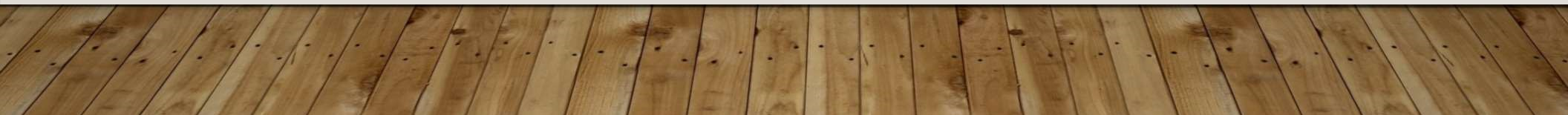
## I. Requirements reviews:

- Systematic manual analysis of the requirements
- A group of people from both organization side and user side carefully reviews the SRS.
- They review the document to check for any errors and ambiguity.
- After that, they gather up to discuss the issues and figure out a way to address the issues.
- A checklist is prepared that the reviewers fill up to provide a formal output of the review.
- After that, a final approval sign-off is done.

## 2. Prototyping

---

- The prototype of the system is presented before the end-user or customer, they experiment with the presented model and check if it meets their need.
- An executable model of the system is demonstrated to the customer and end-users to validate and ensure if it meets their needs.
- This type of model is generally used to collect feedback about the requirement of the user.
- Prototyping is usually used when the requirements aren't clear.
- So, we make a quick design of the system to validate the requirements. If it fails, we then refine it, and check again, until it meets the customer's needs.



# 3. Test-Case Generation

---

- Developing tests for requirements to check testability.
- The requirements need to be testable.
- If the tests for the requirements are added as part of the validation process, this often reveals requirements problems.
- In some cases test becomes difficult to design, which implies that the requirement is difficult to implement and requires improvement.



## 4. Automated Consistency Analysis

---

- Checking for the consistency of structured requirements descriptions.
- This approach is used for automatic detection of an error, such as nondeterminism, missing cases, a type error, and circular definitions, in requirements specifications.
- First, the requirement is structured in formal notation then CASE tool is used to check in-consistency of the system, The report of all inconsistencies is identified and corrective actions are taken.

# Advantages Of Using Requirements Validation Techniques

---

- Improved quality of the final product
- Reduced development time and cost. Validation techniques can reduce the likelihood of costly rework later on.
- Increased user involvement
- Improved communication: It can improve communication between stakeholders and developers, by providing a clear and visual representation of the software requirements.
- Easy testing and validation: A prototype can be easily tested and validated, allowing stakeholders to see how the final product will work and identify any issues early on in the development process.



# Disadvantages Of Using Requirements Validation Techniques

---

- Increased time and cost.
- Risk of conflicting: it difficult to prioritize and implement the requirements.
- Risk of changing requirements:
- Misinterpretation and miscommunication
- Limited validation: The validation techniques can only check the requirement that are captured and may not identify the requirement that are missed

# Software Requirement Management

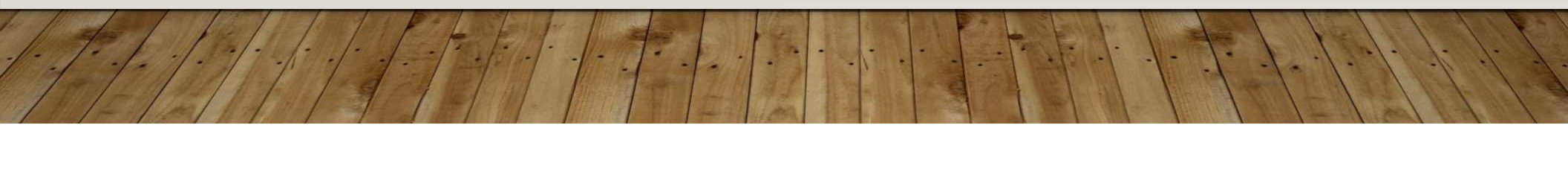
---

- Requirement management is the process of managing changing requirements during the requirements engineering process and system development.

## **Requirements are inevitably incomplete and inconsistent**

- New requirements emerge during the process as business needs change and a better understanding of the system is developed
- Different viewpoints have different requirements and these are often contradictory

## **Requirement change because:**

- The priority of requirements from different viewpoints changes during the development process.
  - System customers may specify requirements from a business perspective that conflict with end-user requirements.
  - The business and technical environment of the system changes during its development.
- 

# CONTD...

---

## Principal stages consists of:

- Problem analysis: Discuss requirements problem and propose change
- Change analysis and costing: Assess effects of change on other requirements
- Change implementation: Modify requirements document and other documents to reflect change



# Feasibility Study

---

- A feasibility study decides whether or not the proposed system is worthwhile.
- It is a study to evaluate feasibility of proposed project or system.
- It is one of stage among important four stages of Software Project Management Process.
- It is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view.
- It is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

# CONTD...

---

- A short focused study that checks
  - If the system contributes to organizational objectives
  - If the system can be engineered using current technology and within budget
  - If the system can be integrated with other systems that are used

# Types Of Feasibility

---

## Technical Feasibility -

- Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
- Current resources both hardware software along with required technology are analyzed/assessed to develop project.
- It provides information on whether the appropriate resources and technology required for use in project development are in place.
- In addition, the feasibility study also analyzes the technical strength and capabilities of the technical team, whether existing technology can be used, and whether the selected technology is easy to maintain and upgrade.

# Operational Feasibility

---

- Operational feasibility assesses the range in which the required software performs series of levels to solve business problems and customer requirements.
- Operational feasibility analyzes the level of service delivery according to requirements and the ease of operating and maintaining the product after deployment.
- Along with these other operational areas, it determines the product's usability, whether the software development team's decisions for the proposed solution are acceptable, and so on.

# Economic Feasibility

---

- Economic feasibility decides whether the necessary software can generate financial profits for an organization.
- Project costs and benefits are analyzed in a profitability study.
- This means that as part of this feasibility study, a detailed analysis of the costs of the development project will be made.
- This includes all costs necessary for the final development, such as hardware and software resources required, design and development costs, operating costs, etc.
- It is then analyzed whether the project is financially beneficial to the organization.





# View Points

---

- This principle states to understand the viewpoints of the stakeholders before specifying the requirements.
- We need to understand what are the actual problems of stakeholders or their business.

There are two key tools to cover Viewpoints:

- **a) Requirements Gathering** – Gathering essentially means to collect the piece of information that is available. This may include
  - Information that is stored in documents
  - Information that is stored in systems
  - Information that is already drawn out by the client end stakeholders

## **b) Requirements Elicitation –**

- Elicitation is an added but extremely significant step to fetch the information from stakeholders, which is not readily available on documents.
- It can be carried out via. processes like Interviewing, where RE team puts questions to stakeholders about the system that they use and the system that they want to be developed.

This basically revolves around answering 6 significant questions:-

- Who should be involved?
- What do they want?
- Where (in what context) could it work?
- Why do they want it?
- How will we know?
- When should we build it?



# Software Requirements Specification vs. System Requirements Specification

What is the difference between a *software* requirements specification document and a *system* requirements specification document?

“Software” and “system” are sometimes used interchangeably as SRS. But, a software requirements specification provides greater detail than a system requirements specification.

A **system requirements specification** presents general information on the requirements of a system, which may include both hardware and software, based on an analysis of business needs.

A **software requirements specification (SRS)** details the specific requirements of the software that is to be developed.



# Assignment Questions

---

