PARSHWANATH CHARITABLE TRUST'S

A.P. SHAH INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Data Science

Requirement Analysis and Cost Estimation

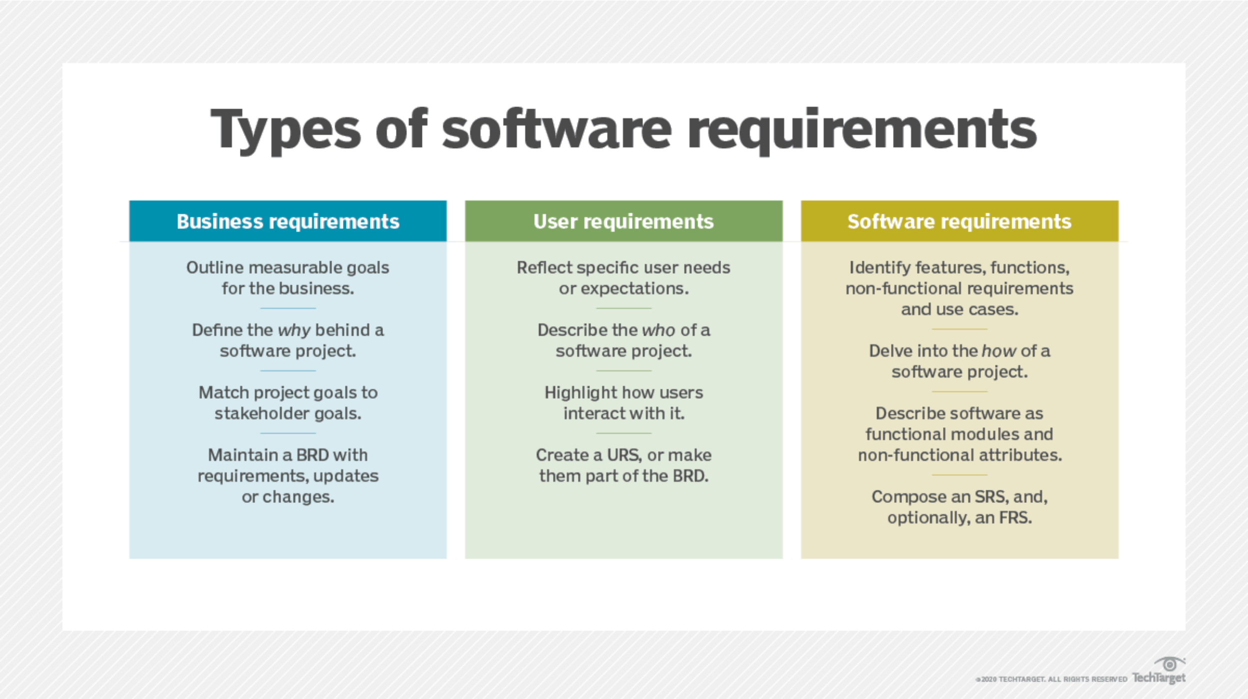
❖ Software Requirements: Functional & non-functional, User-system requirement engineering process – feasibility studies – elicitation – validation & management – software prototyping – S/W documentation – Analysis and modeling Requirement Elicitation.

The information which describes the user's expectations about the system performance is called as requirements.

Characteristics:

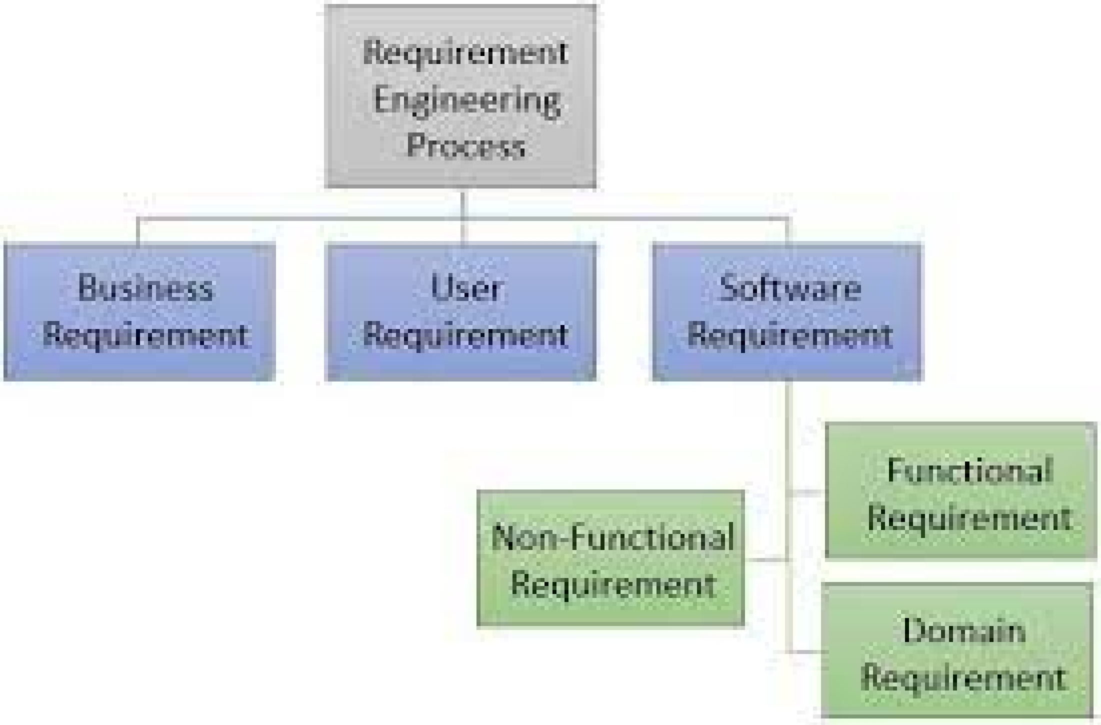
- Unambiguous
- Verifiable
- Clear
- Understandable
- Feasible
- consistent

Requirement Engineering: the procedure which collects the software requirements from customers , analyzes and documents them is called requirement engg.



Types of software requirements

Business requirements	User requirements	Software requirements
Outline measurable goals for the business.	Reflect specific user needs or expectations.	Identify features, functions, non-functional requirements and use cases.
Define the *why* behind a software project.	Describe the *who* of a software project.	Delve into the *how* of a software project.
Match project goals to stakeholder goals.	Highlight how users interact with it.	Describe software as functional modules and non-functional attributes.
Maintain a BRD with requirements, updates or changes.	Create a URS, or make them part of the BRD.	Compose an SRS, and, optionally, an FRS.



Functional Requirements

- These are the requirements that the end user specifically demands as basic facilities that the system should offer. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.
- All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected.
- They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.
- For example, in a hospital management system, a doctor should be able to retrieve the information of his patients.
- Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world.
- In order to accurately describe the functional requirements, all scenarios must be enumerated.
- There are many ways of expressing functional requirements e.g., natural language, a structured or formatted language with no rigorous syntax and formal specification language with proper syntax. Functional Requirements in Software Engineering are also called Functional Specification.

Non-Functional Requirement

- These are basically the quality constraints that the system must satisfy according to the project contract.
- Nonfunctional requirements, not related to the system functionality, rather define how the system should perform.
- The priority or extent to which these factors are implemented varies from one project to another. They are also called non-behavioral requirements.

- They basically deal with issues like:
  - Portability
  - Security
  - Maintainability
  - Reliability
  - Scalability
  - Performance
  - Reusability
  - Flexibility

Functional Requirements	Non Functional Requirements
A functional requirement defines a system or its components.	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 cases.	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 the user whenever he/she logs into the system. 2) System shutdown in case of a cyber attack. 3) A Verification email is sent to the 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

Requirement Engineering Tasks

The software requirements engineering process includes the following steps of activities:

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Requirements Management

Inception: This is the first phase of the requirements analysis process. This phase gives an outline of how to get started on a project. In the inception phase, all the basic questions are asked on how to go about a task or the steps required to accomplish a task. A basic understanding of the problem is gained and the nature of the solution is addressed. Effective communication is very important in this stage, as this phase is the foundation as to what has to be done further. Overall in the inception phase, the following criteria have to be addressed by the software engineers:

- Understanding of the problem.
- The people who want a solution.

- Nature of the solution.
- Communication and collaboration between the customer and developer.

Elicitation: This is the second phase of the requirements analysis process. 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. In this process, mistakes can happen in regard to, not implementing the right requirements or forgetting a part. The right people must be involved in this phase. The following problems can occur in the elicitation phase:

- Problem of Scope: The requirements given are of unnecessary detail, ill-defined, or not possible to implement.
- Problem of Understanding: Not having a clear-cut understanding between the developer and customer when putting out the requirements needed. Sometimes the customer might not know what they want or the developer might misunderstand one requirement for another.
- Problem of Volatility: Requirements changing over time can cause difficulty in leading a project. It can lead to loss and wastage of resources and time.

Elaboration: This is the third phase of the requirements analysis process. 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.

Negotiation: This is the fourth phase of the requirements analysis process. 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.

Specification: This is the fifth phase of the requirements analysis process. This phase specifies the following:

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

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.

Validation: This is the sixth phase of the requirements analysis process. 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.

Requirements Management: This is the last phase of the requirements analysis process. 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 Elicitation

Requirements elicitation is the process of gathering and defining the requirements for a software system. The goal of requirements elicitation is to ensure that the software development process is based on a clear and comprehensive understanding of the customer's needs and requirements. This article focuses on discussing Requirement Elicitation in detail.

Requirements elicitation is perhaps the most difficult, most error-prone, and most communication-intensive software development.

- It can be successful only through an effective customer-developer partnership. It is needed to know what the users require.
- Requirements elicitation involves the identification, collection, analysis, and refinement of the requirements for a software system.
- It is a critical part of the software development life cycle and is typically performed at the beginning of the project.
- Requirements elicitation involves stakeholders from different areas of the organization, including business owners, end-users, and technical experts.
- The output of the requirements elicitation process is a set of clear, concise, and well-defined requirements that serve as the basis for the design and development of the software system.

Importance of Requirements Elicitation

- Compliance with Business Objectives: The process of elicitation guarantees that the software development endeavors are in harmony with the wider company aims and objectives. Comprehending the business context facilitates the development of a solution that adds value for the company.
- User Satisfaction: It is easier to create software that fulfills end users needs and expectations when they are involved in the requirements elicitation process. Higher user pleasure and acceptance of the finished product are the results of this.
- Time and Money Savings: Having precise and well-defined specifications aids in preventing miscommunication and rework during the development phase. As a result, there will be cost savings and the project will be completed on time.
- Compliance and Regulation Requirements: Requirements elicitation is crucial for projects in regulated industries to guarantee that the software conforms with applicable laws and norms. In industries like healthcare, finance, and aerospace, this is crucial.
- Traceability and Documentation: Throughout the software development process, traceability is based on requirements that are well-documented. Traceability helps

with testing, validation, and maintenance by ensuring that every part of the software can be linked to a particular requirement.

Requirements Elicitation Methods

1. 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 maybe 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 a structured interview, an agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.

2. 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.

3. Facilitated Application Specification Technique

Its objective is to bridge the expectation gap – the difference between what the developers think they are supposed to build and what customers think they are going to get. A team-oriented approach is developed for requirements gathering. Each attendee is asked to make a list of objects that are:

- Part of the environment that surrounds the system.

- Produced by the system.
- Used by the system.

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, the team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

4. Quality Function Deployment

In this technique customer satisfaction is of prime concern, hence it emphasizes on the requirements which are valuable to the customer.

3 types of requirements are identified:

- Normal requirements: In this the objective and goals of the proposed software are discussed with the customer. Example – normal requirements for a result management system may be entry of marks, calculation of results, etc.
- Expected requirements: These requirements are so obvious that the customer need not explicitly state them. Example – protection from unauthorized access.
- Exciting requirements: It includes features that are beyond customer's expectations and prove to be very satisfying when present. Example – when unauthorized access is detected, it should backup and shutdown all processes.

5. Use Case Approach

This technique combines text and pictures to provide a better understanding of the requirements.

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 include three major things – Actor, use cases, use case diagram.

- Actor: It is the external agent that lies outside the system but interacts with it in some way. An actor, maybe a person, machine etc. It is represented as a stick figure. Actors can be primary actors or secondary actors.

  - Primary actors: It requires assistance from the system to achieve a goal.
  - Secondary actor: It is an actor from which the system needs assistance.
- Use cases: They describe the sequence of interactions between actors and the system. They capture who(actors) do what(interaction) with the system. A complete set of use cases specifies all possible ways to use the system.
- 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.