Requirement Engineering: Software Requirements, Types of Requirements, Requirement Engineering Cycle, Characteristics of Requirements, Requirement verification and validation.

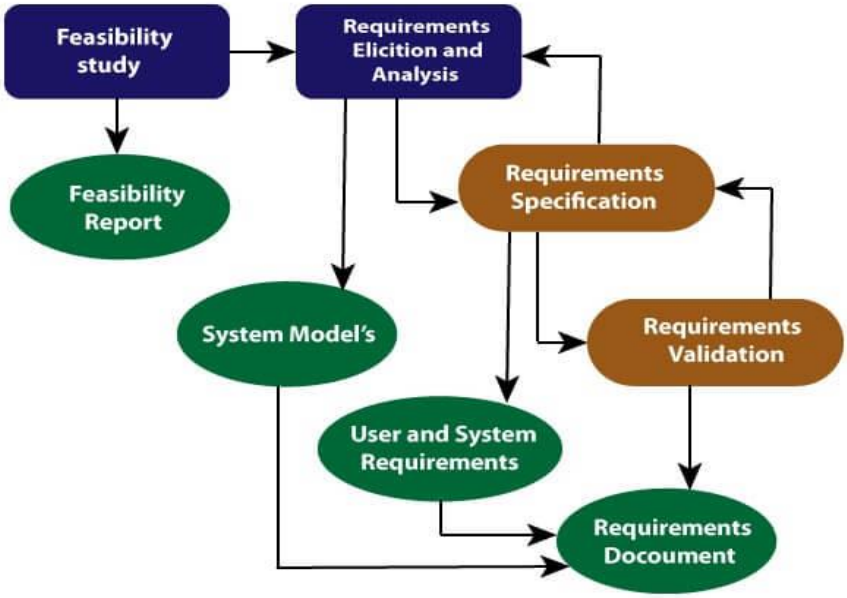
Requirement Engineering

Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

Requirement Engineering Process/ Cycle

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

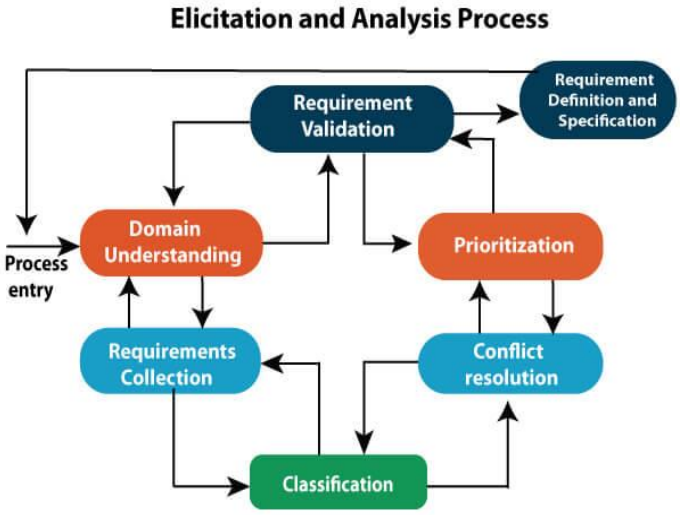
1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

2. Requirement Elicitation and Analysis:



This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

- Getting all, and only, the right people involved.
- Stakeholders often don't know what they want
- Stakeholders express requirements in their terms.
- Stakeholders may have conflicting requirements.
- Requirement change during the analysis process.
- Organizational and political factors may influence system requirements.

3. Software Requirement Specification:

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.

- **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.
- **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.
- **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.

4. Software Requirement Validation:

After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Validation Techniques

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.
- **Prototyping:** Using an executable model of the system to check requirements.
- **Test-case generation:** Developing tests for requirements to check testability.
- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

Software Requirement Management:

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

New requirements emerge during the process as business needs a change, and a better understanding of the system is developed.

The priority of requirements from different viewpoints changes during development process.

The business and technical environment of the system changes during the development.

Prerequisite of Software requirements

Collection of software requirements is the basis of the entire software development project. Hence they should be clear, correct, and well-defined.

A complete Software Requirement Specifications should be:

- Clear
- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable

- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source

Software Requirements: Largely software requirements must be categorized into two categories:

1. **Functional Requirements:** Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.
2. **Non-functional Requirements:** This can be the necessities that specify the criteria that can be used to decide the operation instead of specific behaviors of the system. Non-functional requirements are divided into two main categories:
    - **Execution qualities** like security and usability, which are observable at run time.
    - **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

Characteristics of requirements:

Requirements in software engineering are fundamental specifications that outline what a software system must accomplish. They serve as a bridge between client needs and the technical implementation. Here are some characteristics of requirements, presented in an understandable manner:

1. Clarity:

   Requirements should be clear and easy to understand for all stakeholders involved, including developers, testers, and clients. Ambiguity in requirements can lead to misunderstandings and errors in implementation.

2. Completeness:
   Requirements should encompass all necessary functionalities and constraints of the software system. Any missing requirements can lead to gaps in the final product, causing dissatisfaction among users.
3. Consistency:

   Requirements should be consistent with each other and with the overall objectives of the software project. Inconsistencies can lead to confusion and conflicts during development.

4. Feasibility:

   Requirements should be feasible within the constraints of technology, budget, and time. Unrealistic requirements can lead to project delays, cost overruns, and dissatisfaction among stakeholders.

5. Relevance:
   Requirements should be relevant to the needs of the end-users and stakeholders. Including irrelevant requirements can waste resources and complicate the development process.
6. Testability:
   Requirements should be testable, meaning that it should be possible to verify whether the software meets the specified requirements. Testable requirements help ensure the quality and reliability of the software.

7. Modifiability: Requirements should be flexible enough to accommodate changes and updates during the development process and beyond. Software projects often evolve, and requirements should be able to adapt accordingly.

8. Traceability: Requirements should be traceable, meaning that there should be a clear link between each requirement and its source, rationale, and verification status. Traceability helps ensure accountability and facilitate change management.

9. Priority: Requirements should be prioritized based on their importance to the overall objectives of the project and the needs of stakeholders. Prioritization helps focus resources on the most critical aspects of the software.

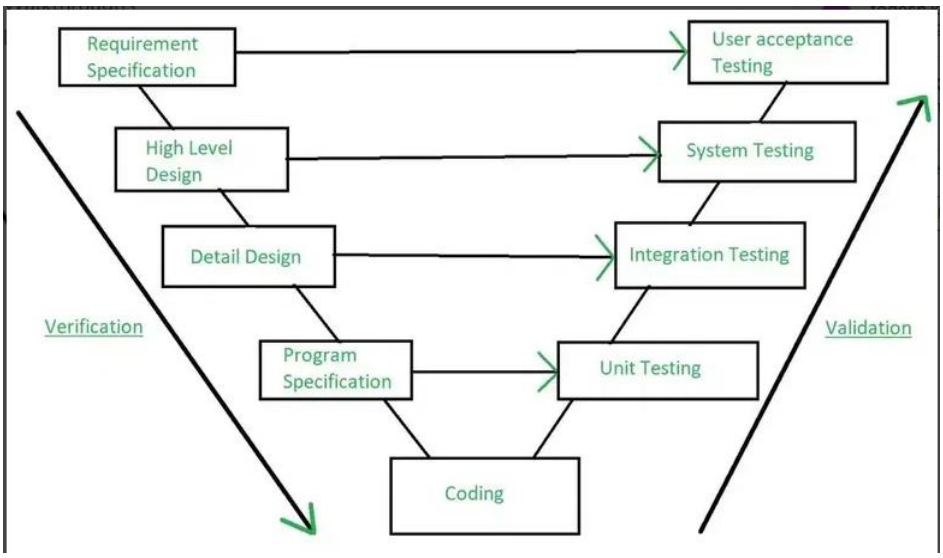
10. Verifiability: Requirements should be verifiable, meaning that there should be a way to confirm whether the software meets each requirement. Verifiable requirements help ensure that the software performs as expected.

Verification and Validation is the process of investigating whether a software system satisfies specifications and standards and fulfills the required purpose. **Barry Boehm** described verification and validation as the following:
Verification: *Are we building the product right?*
Validation: *Are we building the right product?*

Verification
Verification is the process of checking that software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is simply known as **Static Testing.**
Static Testing

Verification Testing is known as Static Testing and it can be simply termed as checking whether we are developing the right product or not and also whether our software is fulfilling the customer's requirement or not. Here are some of the activities that are involved in verification.

- Inspections
- Reviews
- Walkthroughs
- Desk-checking



Verification and Validation

What is Requirements Validation?

Requirements validation is the process of checking that requirements defined for development, define the system that the customer wants. To check issues related to requirements, we perform requirements validation. We typically use requirements validation to check errors at the initial phase of development as the error may increase excessive rework when detected later in the development process. In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software Requirements Specification (SRS), these checks include:

1. **Completeness checks**
2. **Consistency checks**
3. **Validity checks**
4. **Realism checks**
5. **Ambiguity checks**
6. **Variability**

The output of requirements validation is the list of problems and agreed-on actions of detected problems. The lists of problems indicate the problem detected during the process of requirement validation. The list of agreed actions states the corrective action that should be taken to fix the detected problem.

Requirement Validation Techniques

There are several techniques that are used either individually or in conjunction with other techniques to check entire or part of the system:

1. Test Case Generation

The requirement mentioned in the SRS document should be testable, the conducted tests reveal the error present in the requirement. It is generally believed that if the test is difficult or impossible to design, this usually means that the requirement will be difficult to implement and it should be reconsidered.

2. Prototyping

In this validation technique 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. This type of model is mostly used to collect feedback about the requirement of the user.

3. Requirements Reviews

In this approach, the SRS is carefully reviewed by a group of people including people from both the contractor organizations and the client side, the reviewer systematically analyses the document to check errors and ambiguity.

4. Automated Consistency Analysis

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

5. Walk-through

A walkthrough does not have a formally defined procedure and does not require a differentiated role assignment.

- Checking early whether the idea is feasible or not.
- Obtaining the opinions and suggestions of other people.
- Checking the approval of others and reaching an agreement.

6. Simulation

Simulating system behavior in order to verify requirements is known as simulation. This method works especially well for complicated systems when it is possible to replicate real-world settings and make sure the criteria fulfil the desired goals.

7. Checklists for Validation

It employs pre-made checklists to methodically confirm that every prerequisite satisfies predetermined standards. Aspects like completeness, clarity and viability can all be covered by checklists.

Importance of Requirements Validation Techniques

1. Accuracy and Clarity

It makes sure that the requirements are precise, unambiguous and clear. This helps to avoid miscommunications and misunderstandings that may result in mistakes and more effort in subsequent phases of the project.

2. User Satisfaction

It confirms that the requirements meet the wants and expectations of the users, which helps to increase user happiness. This aids in providing a product that satisfies consumer needs and improves user experience as a whole.

3. Early Issue Identification

It makes it easier to find problems, ambiguities or conflicts in the requirements early on. It is more economical to address these issues early in the development phase rather than later, when the project is far along.

4. Prevents the Scope Creep

It ensures that the established requirements are well stated and recorded, which helps to prevent scope creep. By establishing defined parameters for the project's scope, requirements validation helps to lower the possibility of uncontrollably changing course.

5. Improving Quality:

It enhances the software product's overall quality. By detecting and resolving possible quality problems early in the development life cycle, requirements validation contributes to the creation of a more durable and dependable final product.

Advantages of Requirements Validation Techniques

1. **Improved quality of the final product:** By identifying and addressing requirements early on in the development process, using validation techniques can improve the overall quality of the final product.
2. **Reduced development time and cost:** By identifying and addressing requirements early on in the development process, using validation techniques can reduce the likelihood of costly rework later on.
3. **Increased user involvement:** Involving users in the validation process can lead to increased user buy-in and engagement in the project.
4. **Improved communication:** Using validation techniques can improve communication between stakeholders and developers, by providing a clear and visual representation of the software requirements.
5. **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.

6. **Increased alignment with business goals:** Using validation techniques can help to ensure that the requirements align with the overall business goals and objectives of the organization.
7. **Traceability:** This technique can help to ensure that the requirements are being met and that any changes are tracked and managed.
8. **Agile methodologies:** Agile methodologies provide an iterative approach to validate requirements by delivering small chunks of functionality and getting feedback from the customer.

Disadvantages of Requirements Validation Techniques

1. **Increased time and cost:** Using validation techniques can be time-consuming and costly, especially when involving multiple stakeholders.
2. **Risk of conflicting requirements:** Using validation techniques can lead to conflicting requirements, which can make it difficult to prioritize and implement the requirements.
3. **Risk of changing requirements:** Requirements may change over time and it can be difficult to keep up with the changes and ensure that the project is aligned with the updated requirements.
4. **Misinterpretation and miscommunication:** Misinterpretation and miscommunication can occur when trying to understand the requirements.
5. **Dependence on the tool:** The team should be well-trained on the tool and its features to avoid dependency on the tool and not on the requirement.
6. **Limited validation:** The validation techniques can only check the requirement that is captured and may not identify the requirement that is missed
7. **Limited to functional requirements:** Some validation techniques are limited to functional requirements and may not validate non-functional requirements.