

UNIT-

II

Software Requirements: Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

Requirements engineering process: Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management.

Software Requirements:

Functional and Non Functional Requirements

Requirements analysis is a very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and Non-functional requirements.



Understanding and distinguishing between these types of requirements is essential for the success of any project. Our comprehensive **System design course** covers these concepts in detail, providing you with the knowledge and skills to effectively gather, document, and analyze requirements.

Functional Requirements

These are the requirements that the end user specifically demands as basic facilities that the system should offer. 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 the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

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?

Non-Functional Requirements

These are the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. 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.

Extended Requirements

These are basically “nice to have” requirements that might be out of the scope of the System.

Example:

- Our system should record metrics and analytics.
- Service health and performance monitoring.

Difference between Functional Requirements and Non-Functional Requirements:

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 Requirements	Non 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) E mails 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

Requirements Engineering Process in Software Engineering:

•

Requirements Engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system. In this article, we'll learn about its process, advantages, and disadvantages.

Types of requirement:

- **User requirements**

Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

- **System requirements**

A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

What is Requirements Engineering?

A systematic and strict approach to the definition, creation, and verification of requirements for a software system is known as requirements engineering. To guarantee the effective creation of a software product, the requirements engineering process entails several tasks that help in understanding, recording, and managing the demands of stakeholders.

Requirements Engineering Process

1. **Feasibility Study**
2. **Requirements elicitation**
3. **Requirements specification**
4. **Requirements for verification and validation**
5. **Requirements management**

1. Feasibility Study

The feasibility study mainly concentrates on below five mentioned areas below. Among these Economic Feasibility Study is the most important part of the feasibility analysis and the Legal Feasibility Study is less considered feasibility analysis.

1. **Technical Feasibility:** In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development. Along with this, the feasibility study also analyzes the technical skills and capabilities of the technical team, whether existing technology can be used or not, whether maintenance and up-gradation are easy or not for the chosen technology, etc.
2. **Operational Feasibility:** In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining the usability of the product, Determining suggested solution by the software development team is acceptable or not, etc.
3. **Economic Feasibility:** In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.
4. **Legal Feasibility:** In legal feasibility, the project is ensured to comply with all relevant laws, regulations, and standards. It identifies any legal constraints that could impact the project and reviews existing contracts and agreements to assess their effect on the project's execution. Additionally, legal feasibility considers issues related to intellectual property, such as patents and copyrights, to safeguard the project's innovation and originality.
5. **Schedule Feasibility:** In schedule feasibility, the project timeline is evaluated to determine if it is realistic and achievable. Significant milestones are identified, and deadlines are established to track progress effectively. Resource availability is assessed to ensure that the necessary resources are accessible to meet the project schedule. Furthermore, any time constraints that

might affect project delivery are considered to ensure timely completion. This focus on schedule feasibility is crucial for the successful planning and execution of a project.

2. Requirements Elicitation

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed [here](#). Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage.

Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This is the first step in the requirements engineering process and it is critical to the success of the software development project. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system.

Several techniques can be used to elicit requirements, including:

- **Interviews:** These are one-on-one conversations with stakeholders to gather information about their needs and expectations.
- **Surveys:** These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.
- **Focus Groups:** These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.
- **Observation:** This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.
- **Prototyping:** This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

It's important to document, organize, and prioritize the requirements obtained from all these techniques to ensure that they are complete, consistent, and accurate.

3. Requirements Specification

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process. The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.

Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks.

The goal of this step is to create a clear and comprehensive document that describes the requirements for the software system. This document should be understandable by both the development team and the stakeholders.

Several types of requirements are commonly specified in this step, including

1. **Functional Requirements:** These describe what the software system should do. They specify the functionality that the system must provide, such as input validation, data storage, and user interface.

2. **Non-Functional Requirements:** These describe how well the software system should do it. They specify the quality attributes of the system, such as performance, reliability, usability, and security.
3. **Constraints:** These describe any limitations or restrictions that must be considered when developing the software system.
4. **Acceptance Criteria:** These describe the conditions that must be met for the software system to be considered complete and ready for release.

To make the requirements specification clear, the requirements should be written in a natural language and use simple terms, avoiding technical jargon, and using a consistent format throughout the document. It is also important to use diagrams, models, and other visual aids to help communicate the requirements effectively.

Once the requirements are specified, they must be reviewed and validated by the stakeholders and development team to ensure that they are complete, consistent, and accurate.

4. Requirements Verification and Validation

Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.

Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. The main steps for this process include:

1. The requirements should be consistent with all the other requirements i.e. no two requirements should
- 2.
3. conflict with each other.
2. The requirements should be complete in every sense.
3. The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements verification and validation (V&V) is the process of checking that the requirements for a software system are complete, consistent, and accurate and that they meet the needs and expectations of the stakeholders. The goal of V&V is to ensure that the software system being developed meets the requirements and that it is developed on time, within budget, and to the required quality.

1. Verification is checking that the requirements are complete, consistent, and accurate. It involves reviewing the requirements to ensure that they are clear, testable, and free of errors and inconsistencies. This can include reviewing the requirements document, models, and diagrams, and holding meetings and walkthroughs with stakeholders.
2. Validation is the process of checking that the requirements meet the needs and expectations of the stakeholders. It involves testing the requirements to ensure that they are valid and that the software system being developed will meet the needs of the stakeholders. This can include testing the software system through simulation, testing with prototypes, and testing with the final version of the software.
3. Verification and Validation is an iterative process that occurs throughout the software development life cycle. It is important to involve stakeholders and the development team in the V&V process to ensure that the requirements are thoroughly reviewed and tested.

It's important to note that V&V is not a one-time process, but it should be integrated and continue throughout the software development process and even in the maintenance stage.

5. Requirements Management

Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality.

Several key activities are involved in requirements management, including:

1. **Tracking and controlling changes:** This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.
2. **Version control:** This involves keeping track of different versions of the requirements document and other related artifacts.
3. **Traceability:** This involves linking the requirements to other elements of the development process, such as design, testing, and validation.
4. **Communication:** This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.
5. **Monitoring and reporting:** This involves monitoring the progress of the development process and reporting on the status of the requirements.

Requirements management is a critical step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders and that it is developed on time, within budget, and to the required quality. It also helps to prevent scope creep and to ensure that the requirements are aligned with the project goals.

Tools Involved in Requirement Engineering

- Observation report
- Questionnaire (survey, poll)
- Use cases
- User stories
- Requirement workshop
- Mind mapping
- Role playing
- Prototyping

Advantages of Requirements Engineering Process

- Helps ensure that the software being developed meets the needs and expectations of the stakeholders
- Can help identify potential issues or problems early in the development process, allowing for adjustments to be made before significant
- Helps ensure that the software is developed in a cost-effective and efficient manner
- Can improve communication and collaboration between the development team and stakeholders
- Helps to ensure that the software system meets the needs of all stakeholders.

- Provides an unambiguous description of the requirements, which helps to reduce misunderstandings and errors.
- Helps to identify potential conflicts and contradictions in the requirements, which can be resolved before the software development process begins.
- Helps to ensure that the software system is delivered on time, within budget, and to the required quality standards.
- Provides a solid foundation for the development process, which helps to reduce the risk of failure.

Disadvantages of Requirements Engineering Process

- Can be time-consuming and costly, particularly if the requirements-gathering process is not well-managed
- Can be difficult to ensure that all stakeholders' needs and expectations are taken into account
- It Can be challenging to ensure that the requirements are clear, consistent, and complete
- Changes in requirements can lead to delays and increased costs in the development process.
- As a best practice, Requirements engineering should be flexible, adaptable, and should be aligned with the overall project goals.
- It can be time-consuming and expensive, especially if the requirements are complex.
- It can be difficult to elicit requirements from stakeholders who have different needs and priorities.
- Requirements may change over time, which can result in delays and additional costs.
- There may be conflicts between stakeholders, which can be difficult to resolve.
- It may be challenging to ensure that all stakeholders understand and agree on the requirements.

Interface Specification in Software Engineering

- - Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
 - Three types of interface may have to be defined
 - **Procedural interfaces** where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. These interfaces are sometimes called Application Programming Interfaces (APIs)
 - **Data structures that are exchanged** that are passed from one sub-system to another. Graphical data models are the best notations for this type of description
 - **Data representations** that have been established for an existing sub-system
 - Formal notations are an effective technique for interface specification.

The user interface is the **front-end application** view to which the **user interacts** to use the software. The software becomes more popular if its user interface is:

1. **Attractive**
2. **Simple to use**
3. **Responsive in a short time**

4. **Clear to understand**
5. **Consistent on all interface screens**

Types of User Interface

1. **Command Line Interface:** The Command Line Interface provides a command prompt, where the user types the command and feeds it to the system. The user needs to remember the syntax of the command and its use.
2. **Graphical User Interface:** Graphical User Interface provides a simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, the user interprets the software.

Interface Specification in Software Engineering:

The **analysis** and **design** process of a user interface is iterative and can be represented by a **spiral model**. The analysis and design process of user interface consists of four framework activities.

1. User, Task, Environmental Analysis, and Modeling

Initially, the focus is based on the profile of users who will interact with the system, i.e., understanding, skill and knowledge, type of user, etc., based on the user's profile users are made into categories. From each category requirements are gathered. Based on the requirement's developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:

1. Where will the interface be located physically?
2. Will the user be sitting, standing, or performing other tasks unrelated to the interface?
3. Does the interface hardware accommodate space, light, or noise constraints?
4. Are there special human factors considerations driven by environmental factors?

2. Interface Design

The goal of this phase is to define the set of interface objects and actions i.e., control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by Theo Mandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.

3. Interface Construction and Implementation

The implementation activity begins with the creation of a prototype (model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.

4. Interface Validation

This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly, and it should be able to handle a variety of tasks. It should achieve

all the user's requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

User Interface Design Golden Rules

The following are the golden rules stated by Theo Mandel that must be followed during the design of the interface. **Place the user in control:**

1. **Define the interaction modes in such a way that does not force the user into unnecessary or undesired actions:** The user should be able to easily enter and exit the mode with little or no effort.
2. **Provide for flexible interaction:** Different people will use different interaction mechanisms, some might use keyboard commands, some might use mouse, some might use touch screen, etc., Hence all interaction mechanisms should be provided.
3. **Allow user interaction to be interruptible and undoable:** When a user is doing a sequence of actions the user must be able to interrupt the sequence to do some other work without losing the work that had been done. The user should also be able to do undo operation.
4. **Streamline interaction as skill level advances and allow the interaction to be customized:** Advanced or highly skilled user should be provided a chance to customize the interface as user wants which allows different interaction mechanisms so that user doesn't feel bored while using the same interaction mechanism.
5. **Hide technical internals from casual users:** The user should not be aware of the internal technical details of the system. He should interact with the interface just to do his work.
6. **Design for direct interaction with objects that appear on-screen:** The user should be able to use the objects and manipulate the objects that are present on the screen to perform a necessary task. By this, the user feels easy to control over the screen.

Reduce the User's Memory Load

1. **Reduce demand on short-term memory:** When users are involved in some complex tasks the demand on short-term memory is significant. So the interface should be designed in such a way to reduce the remembering of previously done actions, given inputs and results.
2. **Establish meaningful defaults:** Always an initial set of defaults should be provided to the average user, if a user needs to add some new features then he should be able to add the required features.
3. **Define shortcuts that are intuitive:** Mnemonics should be used by the user. Mnemonics means the keyboard shortcuts to do some action on the screen.
4. **The visual layout of the interface should be based on a real-world metaphor:** Anything you represent on a screen if it is a metaphor for a real-world entity then users would easily understand.
5. **Disclose information in a progressive fashion:** The interface should be organized hierarchically i.e., on the main screen the information about the task, an object or some behavior should be presented first at a high level of abstraction. More detail should be presented after the user indicates interest with a mouse pick.

Make the Interface Consistent

1. **Allow the user to put the current task into a meaningful context:** Many interfaces have dozens of screens. So it is important to provide indicators consistently so that the user know about the doing work. The user should also know from which page has navigated to the current page and from the current page where it can navigate.

2. **Maintain consistency across a family of applications:** in The development of some set of applications all should follow and implement the same design, rules so that consistency is maintained among applications.
3. If past interactive models have created user expectations do not make changes unless there is a compelling reason.

User interface design is a crucial aspect of software engineering, as it is the means by which users interact with software applications. A well-designed user interface can improve the usability and user experience of an application, making it easier to use and more effective.

Key Principles for Designing User Interfaces

1. **User-centered design:** User interface design should be focused on the needs and preferences of the user. This involves understanding the user's goals, tasks, and context of use, and designing interfaces that meet their needs and expectations.
2. **Consistency:** Consistency is important in user interface design, as it helps users to understand and learn how to use an application. Consistent design elements such as icons, color schemes, and navigation menus should be used throughout the application.
3. **Simplicity:** User interfaces should be designed to be simple and easy to use, with clear and concise language and intuitive navigation. Users should be able to accomplish their tasks without being overwhelmed by unnecessary complexity.
4. **Feedback:** Feedback is significant in user interface design, as it helps users to understand the results of their actions and confirms that they are making progress towards their goals. Feedback can take the form of visual cues, messages, or sounds.
5. **Accessibility:** User interfaces should be designed to be accessible to all users, regardless of their abilities. This involves considering factors such as color contrast, font size, and assistive technologies such as screen readers.
6. **Flexibility:** User interfaces should be designed to be flexible and customizable, allowing users to tailor the interface to their own preferences and needs.

Software Requirement Document:

- **Importance of formulating the SRS:** In software engineering, the Software Requirements Specification (SRS) is a document that describes the requirements of the software system. It is a key document in the software development process, as it provides a clear and concise description of what the software is supposed to do. The SRS is used by the software developers to design and build the software, and it is also used by the project manager to manage the project. The SRS is a living document, and it is updated as the project progresses. The SRS is a key document in the software development process, as it provides a clear and concise description of what the software is supposed to do. The SRS is used by the software developers to design and build the software, and it is also used by the project manager to manage the project. The SRS is a living document, and it is updated as the project progresses.

Document Title

Author(s)

Affiliation

Address

Date

Document Version

Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer's demand.

Introduction

- **Purpose of this Document** – At first, main aim of why this document is necessary and what's purpose of document is explained and described.
- **Scope of this document** – In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.
- **Overview** – In this, description of product is explained. It's simply summary or overall review of product.

General description

In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.

Functional Requirements

In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order. Functional requirements specify the expected behavior of the system-which outputs should be produced from the given inputs. They describe the relationship between the input and output of the system. For each functional requirement, detailed description all the data inputs and their source, the units of measure, and the range of valid inputs must be specified.

Interface Requirements

In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.

Performance Requirements

In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc. The performance requirements part of an SRS specifies the performance constraints on the software system. All the requirements relating to the performance characteristics of the system must be clearly specified. There are two types of performance requirements: static and dynamic. Static requirements are those that do not impose constraint on the execution characteristics of the system. Dynamic requirements specify constraints on the execution behaviour of the system.

Design Constraints

In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc. There are a number of factors in the client's environment that may restrict the choices of a designer leading to design constraints such factors include standards that must be followed resource limits, operating environment, reliability and security requirements and policies that may have an impact on the design of the system. An SRS should identify and specify all such constraints.

Non-Functional Attributes

In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.

Preliminary Schedule and Budget

In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.

Appendices

In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

Uses of SRS document

- Development team require it for developing product according to the need.
- Test plans are generated by testing group based on the describe external behaviour.
- Maintenance and support staff need it to understand what the software product is supposed to do.
- Project manager base their plans and estimates of schedule, effort and resources on it.
- customer rely on it to know that product they can expect.
- As a contract between developer and customer.
- in documentation purpose.

Requirement Engineering Process:**Feasibility Study:**

The feasibility study mainly concentrates on below five mentioned areas below. Among these Economic Feasibility Study is the most important part of the feasibility analysis and the Legal Feasibility Study is less considered feasibility analysis.

1. **Technical Feasibility:** In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development. Along with this, the feasibility study also analyzes the technical skills and capabilities of the technical team, whether existing technology can be used or not, whether maintenance and up-gradation are easy or not for the chosen technology, etc.
2. **Operational Feasibility:** In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment. Along with this other operational scopes are determining the usability of the product, Determining suggested solution by the software development team is acceptable or not, etc.

3. **Economic Feasibility:** In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.
4. **Legal Feasibility:** In legal feasibility, the project is ensured to comply with all relevant laws, regulations, and standards. It identifies any legal constraints that could impact the project and reviews existing contracts and agreements to assess their effect on the project's execution. Additionally, legal feasibility considers issues related to intellectual property, such as patents and copyrights, to safeguard the project's innovation and originality.
5. **Schedule Feasibility:** In schedule feasibility, the project timeline is evaluated to determine if it is realistic and achievable. Significant milestones are identified, and deadlines are established to track progress effectively. Resource availability is assessed to ensure that the necessary resources are accessible to meet the project schedule. Furthermore, any time constraints that might affect project delivery are considered to ensure timely completion. This focus on schedule feasibility is crucial for the successful planning and execution of a project.

Requirements Elicitation:

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed [here](#). Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage.

Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This is the first step in the requirements engineering process and it is critical to the success of the software development project. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system.

Several techniques can be used to elicit requirements, including:

- **Interviews:** These are one-on-one conversations with stakeholders to gather information about their needs and expectations.
- **Surveys:** These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.
- **Focus Groups:** These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.
- **Observation:** This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.
- **Prototyping:** This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

It's important to document, organize, and prioritize the requirements obtained from all these techniques to ensure that they are complete, consistent, and accurate.

Requirements Verification and Validation:

Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.

Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. The main steps for this process include:

1. The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.
2. The requirements should be complete in every sense.
3. The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements verification and validation (V&V) is the process of checking that the requirements for a software system are complete, consistent, and accurate and that they meet the needs and expectations of the stakeholders. The goal of V&V is to ensure that the software system being developed meets the requirements and that it is developed on time, within budget, and to the required quality.

1. Verification is checking that the requirements are complete, consistent, and accurate. It involves reviewing the requirements to ensure that they are clear, testable, and free of errors and inconsistencies. This can include reviewing the requirements document, models, and diagrams, and holding meetings and walkthroughs with stakeholders.
2. Validation is the process of checking that the requirements meet the needs and expectations of the stakeholders. It involves testing the requirements to ensure that they are valid and that the software system being developed will meet the needs of the stakeholders. This can include testing the software system through simulation, testing with prototypes, and testing with the final version of the software.
3. Verification and Validation is an iterative process that occurs throughout the software development life cycle. It is important to involve stakeholders and the development team in the V&V process to ensure that the requirements are thoroughly reviewed and tested.

It's important to note that V&V is not a one-time process, but it should be integrated and continue throughout the software development process and even in the maintenance stage.

4. Requirements Management:

Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality.

Several key activities are involved in requirements management, including:

1. **Tracking and controlling changes:** This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.
2. **Version control:** This involves keeping track of different versions of the requirements document and other related artifacts.
3. **Traceability:** This involves linking the requirements to other elements of the development process, such as design, testing, and validation.
4. **Communication:** This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.
5. **Monitoring and reporting:** This involves monitoring the progress of the development process and reporting on the status of the requirements.

Requirements management is a critical step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders and that it is developed on time, within budget, and to the required quality. It also helps to prevent scope creep and to ensure that the requirements are aligned with the project goals.

UNIT-III

Design Engineering: Design process and design quality, design concepts, the design model. Creating an architectural design: software architecture, data design, architectural styles and patterns, architectural design, conceptual model of UML, basic structural modeling, class diagrams, sequence diagrams, collaboration diagrams, use case diagrams, component diagrams.

Software Design Process and quality – Software Engineering

-

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language. The software design process can be divided into the following three levels or phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design

Elements of a System

1. **Architecture:** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules:** These are components that handle one specific task in a system. A combination of the modules makes up the system.