Unit-II

Requirements Analysis and Specification: Requirements Analysis – Software Requirements – Requirements Engineering – Eliciting Requirements – Developing Use Cases - Building the Requirements Model – Negotiating and Validating Requirements.

Requirements Analysis

Software requirement means requirement that is needed by software to increase quality of software product. These requirements are generally a type of expectation of user from software product that is important and need to be fulfilled by software. Analysis means to examine something in an organized and specific manner to know complete details about it. Therefore, **Software requirement analysis** simply means complete study, analyzing, describing software requirements so that requirements that are genuine and needed can be fulfilled to solve problem. There are several activities involved in analyzing Software requirements. Some of them are given below :

1. **Problem Recognition :**
   The main aim of requirement analysis is to fully understand main objective of requirement that includes why it is needed, does it add value to product, will it be beneficial, does it increase quality of the project, does it will have any other effect. All these points are fully recognized in problem recognition so that requirements that are essential can be fulfilled to solve business problems.

2. **Evaluation and Synthesis :**
   Evaluation means judgement about something whether it is worth or not and synthesis means to create or form something. Here are some tasks are given that is important in the evaluation and synthesis of software requirement :
   - To define all functions of software that necessary.
   - To define all data objects that are present externally and are easily observable.
   - To evaluate that flow of data is worth or not.
   - To fully understand overall behavior of system that means overall working of system.
   - To identify and discover constraints that are designed.
   - To define and establish character of system interface to fully understand how system interacts with two or more components or with one another.
3. **Modeling :**
   After complete gathering of information from above tasks, functional and behavioral models are established after checking function and behavior of system using a domain model that also known as the conceptual model.

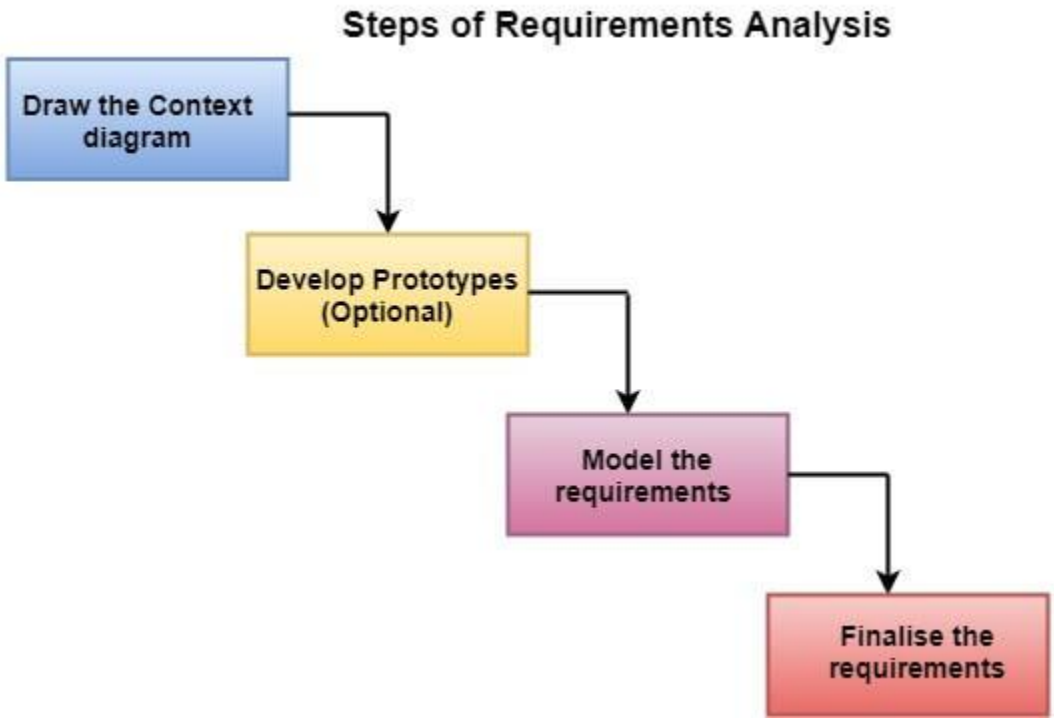
4. **Specification :**
   The software requirement specification (SRS) which means to specify the requirement whether it is functional or non-functional should be developed.

5. **Review :**
   After developing the SRS, it must be reviewed to check whether it can be improved or not and must be refined to make it better and increase the quality.

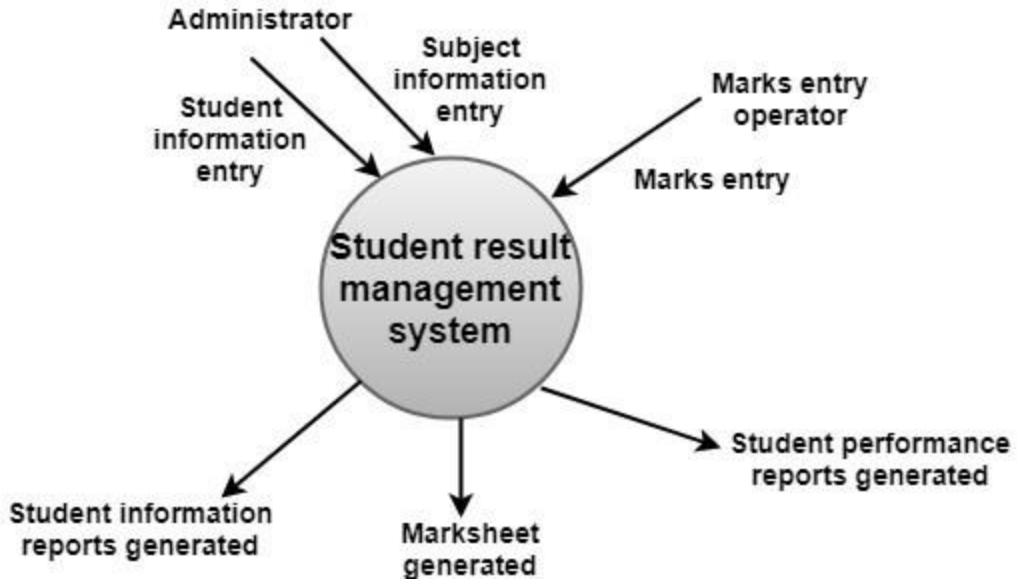
Requirements Analysis

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. This activity reviews all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly. Here, we may also use the interaction with the customer to clarify points of confusion and to understand which requirements are more important than others.

The various steps of requirement analysis are shown in fig:



(i) Draw the context diagram: The context diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world. It identifies the entities outside the proposed system that interact with the system. The context diagram of student result management system is given below:



(ii) Development of a Prototype (optional): One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the system they say they want.

We can use their feedback to modify the prototype until the customer is satisfied continuously. Hence, the prototype helps the client to visualize the proposed system and increase the understanding of the requirements. When developers and users are not sure about some of the elements, a prototype may help both the parties to take a final decision.

Some projects are developed for the general market. In such cases, the prototype should be shown to some representative sample of the population of potential purchasers. Even though a person who tries out a prototype may not buy the final system, but their feedback may allow us to make the product more attractive to others.

The prototype should be built quickly and at a relatively low cost. Hence it will always have limitations and would not be acceptable in the final system. This is an optional activity.

(iii) Model the requirements: This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships between them. The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data Flow diagram, Entity-Relationship diagram, Data Dictionaries, State-transition diagrams, etc.

(iv) Finalise the requirements: After modeling the requirements, we will have a better understanding of the system behavior. The inconsistencies and ambiguities have been identified and corrected. The flow of data amongst various modules has been analyzed. Elicitation and analyze activities have provided better insight into the system. Now we finalize the analyzed requirements, and the next step is to document these requirements in a prescribed format.

Requirements 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 it's process, advantages and disadvantages.

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. In order to guarantee the effective creation of a software product, the requirements engineering process entails a number of tasks that help in understanding, recording and managing the demands of stakeholders.

Steps in Requirements Engineering Process

The requirements engineering process is an iterative process that involves several steps, including:

Requirements Elicitation

This is the process of gathering information about the needs and expectations of stakeholders for the software system. This step involves interviews, surveys, focus groups, and other techniques to gather information from stakeholders.

Requirements Analysis

This step involves analyzing the information gathered in the requirements elicitation step to identify the high-level goals and objectives of the software system. It also involves identifying any constraints or limitations that may affect the development of the software system.

Requirements Specification

This step involves 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.

Requirements Validation

This step involves checking that the requirements are complete, consistent, and accurate. It also involves checking that the requirements are testable and that they meet the needs and expectations of stakeholders.

Requirements Management

This step involves managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant.

Requirement Engineering

The Requirements Engineering process 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.

Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. it is the disciplined application of proven principle , methods ,tools and notations to describe a proposed system’s intended behaviour and its associated constraints.

Tools Involved in Requirement Engineering

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

Requirements Engineering Process Consists of the Following Main Activities

- Requirements elicitation
- Requirements specification
- Requirements verification and validation
- Requirements management

1. 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 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.

There are several techniques that 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.

2. 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.

There are several types of requirements that 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.

In order 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.
3. 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 the process of 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. V&V 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 to relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible so as to incorporate changes in requirements specified by the end users at later stages too. Being able to modify 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.
There are several key activities that 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 in a timely manner.
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.

Advantages and Disadvantages

The advantages and disadvantages of the requirements engineering process in software engineering include:

Advantages

- 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 a clear and 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

- 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
- 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.

Stages in Software Engineering Process

Requirements engineering is a critical process in software engineering that involves identifying, analyzing, documenting, and managing the requirements of a software system. The requirements engineering process consists of the following stages:

- **Elicitation:** In this stage, the requirements are gathered from various stakeholders such as customers, users, and domain experts. The aim is to identify the features and functionalities that the software system should provide.
- **Analysis:** In this stage, the requirements are analyzed to determine their feasibility, consistency, and completeness. The aim is to identify any conflicts or contradictions in the requirements and resolve them.
- **Specification:** In this stage, the requirements are documented in a clear, concise, and unambiguous manner. The aim is to provide a detailed description of the requirements that can be understood by all stakeholders.
- **Validation:** In this stage, the requirements are reviewed and validated to ensure that they meet the needs of all stakeholders. The aim is to ensure that the requirements are accurate, complete, and consistent.
- **Management:** In this stage, the requirements are managed throughout the software development lifecycle. The aim is to ensure that any changes or updates to the requirements are properly documented and communicated to all stakeholders.
- Effective requirements engineering is crucial to the success of software development projects. It helps ensure that the software system meets the needs of all stakeholders and is delivered on time, within budget, and to the required quality standards.

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.

What is Requirement Elicitation?

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

1. It can be successful only through an effective customer-developer partnership. It is needed to know what the users require.
2. Requirements elicitation involves the identification, collection, analysis, and refinement of the requirements for a software system.
3. It is a critical part of the software development life cycle and is typically performed at the beginning of the project.
4. Requirements elicitation involves stakeholders from different areas of the organization, including business owners, end-users, and technical experts.
5. 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

1. **Compliance with Business Objectives:** The process of elicitation guarantees that the software development endeavours 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.

2. **User Satisfaction**: It is easier to create software that fulfils 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.
3. **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.
4. **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.
5. **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 Activities:

Requirements elicitation includes the subsequent activities. A few of them are listed below:

1. Knowledge of the overall area where the systems are applied.
2. The details of the precise customer problem where the system is going to be applied must be understood.
3. Interaction of system with external requirements.
4. Detailed investigation of user needs.
5. Define the constraints for system development.

Requirements Elicitation Methods:

There are a number of requirements elicitation methods. Few of them are listed below:

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.

1. In open-ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
2. 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:

1. Part of the environment that surrounds the system.
2. Produced by the system.
3. Used by the system.

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, 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.

1. **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.
2. **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.
3. **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.

The success of an elicitation technique used depends on the maturity of the analyst, developers, users, and the customer involved.

Steps Of Requirements Elicitation:

1. Identify all the stakeholders, e.g., Users, developers, customers etc.
2. List out all requirements from customer.
3. A value indicating degree of importance is assigned to each requirement.
4. In the end the final list of requirements is categorized as:
   - It is possible to achieve.
   - It should be deferred and the reason for it.
   - It is impossible to achieve and should be dropped off.

Features of Requirements Elicitation:

1. **Stakeholder engagement:** Requirements elicitation involves engaging with stakeholders such as customers, end-users, project sponsors, and subject-matter experts to understand their needs and requirements.
2. **Gathering information:** Requirements elicitation involves gathering information about the system to be developed, the business processes it will support, and the end-users who will be using it.
3. **Requirement prioritization:** Requirements elicitation involves prioritizing requirements based on their importance to the project's success.

4. **Requirements documentation:** Requirements elicitation involves documenting the requirements in a clear and concise manner so that they can be easily understood and communicated to the development team.
5. **Validation and verification:** Requirements elicitation involves validating and verifying the requirements with the stakeholders to ensure that they accurately represent their needs and requirements.
6. **Iterative process:** Requirements elicitation is an iterative process that involves continuously refining and updating the requirements based on feedback from stakeholders.
7. C**ommunication and collaboration:** Requirements elicitation involves effective communication and collaboration with stakeholders, project team members, and other relevant parties to ensure that the requirements are clearly understood and implemented.
8. **Flexibility:** Requirements elicitation requires flexibility to adapt to changing requirements, stakeholder needs, and project constraints.

Advantages of Requirements Elicitation:

1. **Clear requirements:** Helps to clarify and refine customer requirements.
2. **Improves communication:** Improves communication and collaboration between stakeholders.
3. **Results in good quality software:** Increases the chances of developing a software system that meets customer needs.
4. **Avoids misunderstandings:** Avoids misunderstandings and helps to manage expectations.
5. **Supports the identification of potential risks:** Supports the identification of potential risks and problems early in the development cycle.
6. **Facilitates development of accurate plan:** Facilitates the development of a comprehensive and accurate project plan.
7. **Increases user confidence:** Increases user and stakeholder confidence in the software development process.
8. **Supports identification of new business opportunities:** Supports the identification of new business opportunities and revenue streams.

Disadvantages of Requirements Elicitation:

1. **Time consuming:** Can be time-consuming and expensive.
2. **Skills required:** Requires specialized skills and expertise.
3. **Impacted by changing requirements:** May be impacted by changing business needs and requirements.
4. **Impacted by other factors:** Can be impacted by political and organizational factors.
5. **Lack of commitment from stakeholders:** Can result in a lack of buy-in and commitment from stakeholders.
6. **Impacted by conflicting priorities:** Can be impacted by conflicting priorities and competing interests.
7. **Sometimes inaccurate requirements:** May result in incomplete or inaccurate requirements if not properly managed.
8. **Increased development cost:** Can lead to increased development costs and decreased efficiency if requirements are not well-defined.

Developing Use Cases

What are Use Cases?

In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an *actor*) and a system, to achieve a goal. The actor can be a human, an external system, or time. In systems

engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. Another way to look at it is a use case describes a way in which a real-world actor interacts with the system. In a system use case you include high-level implementation decisions. System use cases can be written in both an informal manner and a formal manner. (Wiki)

What is the importance of Use Cases?

Use cases have been used extensively over the past few decades. The advantages of Use cases includes:

- The list of goal names provides the shortest summary of what the system will offer
- It gives an overview of the roles of each and every component in the system. It will help us in defining the role of users, administrators etc.
- It helps us in extensively defining the user's need and exploring it as to how it will work.
- It provides solutions and answers to many questions that might pop up if we start a project unplanned.

How to plan use case?

Following example will illustrate on how to plan use cases:

Use Case: What is the main objective of this use case. For eg. Adding a software component, adding certain functionality etc.
Primary Actor: Who will have the access to this use case. In the above examples, administrators will have the access.
Scope: Scope of the use case
Level: At what level the implementation of the use case be.
Flow: What will be the flow of the functionality that needs to be there. More precisely, the work flow of the use case.
Some other things that can be included in the use cases are:

- **Preconditions**
- **Postconditions**
- **Brief course of action**
- **Time Period**

Use Case Diagram

Below is a sample use case diagram which I have prepared for reference purpose for a sample project (much like Facebook). It would help us to understand the role of various actors in our project. Various actors in the below use case diagram are: **User and System.**
The main use cases are in the system and the diagram illustrates on how the actors interact with the use cases.For eg. During Sign Up, only users need to interact with the use case and not the system whereas when it comes to categorizing posts, only system would be required.

Building the Requirements Model

Requirements for a computer-based system can be seen in many different ways. Some software people argue that it's worth using a number of different modes of representation while others believe that it's best to select one mode of representation. The specific

elements of the requirements model

are dedicated to the analysis modeling method that is to be used.

- **Scenario-based elements :** Using a scenario-based approach, system is described from user's point of view. **For example**, basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases. Figure 1(a) depicts a UML

activity diagram for eliciting requirements and representing them using use cases. There are three levels of elaboration.

- **Class-based elements :** A collection of things that have similar attributes and common behaviors i.e., objects are categorized into classes. **For example**, a UML case diagram can be used to depict a Sensor class for the SafeHome security function. Note that diagram lists attributes of sensors and operations that can be applied to modify these attributes. In addition to class diagrams, other analysis modeling elements depict manner in which classes collaborate with one another and relationships and interactions between classes.
- **Behavioral elements :** Effect of behavior of computer-based system can be seen on design that is chosen and implementation approach that is applied. Modeling elements that depict behavior must be provided by requirements model.
- Method for representing behavior of a system by depicting its states and events that cause system to change state is state diagram. A state is an externally observable mode of behavior. In addition, state diagram indicates actions taken as a consequence of a particular event. To illustrate use of a state diagram, consider software embedded within safeHome control panel that is responsible for reading user input. A simplified UML state diagram is shown in figure 2.
- **Flow-oriented elements :** As it flows through a computer-based system information is transformed. System accepts input, applies functions to transform it, and produces output in a various forms. Input may be a control signal transmitted by a transducer, a series of numbers typed by human operator, a packet of information transmitted on a network link, or a voluminous data file retrieved from secondary storage. Transform may compromise a single logical comparison, a complex numerical algorithm, or a rule-inference approach of an expert system. Output produce a 200-page report or may light a single LED. In effect, we can create a flow model for any computer-based system, regardless of size and complexity.

Negotiating and Validating Requirements.

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.

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.