LECTURE3,4

Requirements Engineering Process in Software Engineering

Eliciting Software requirements

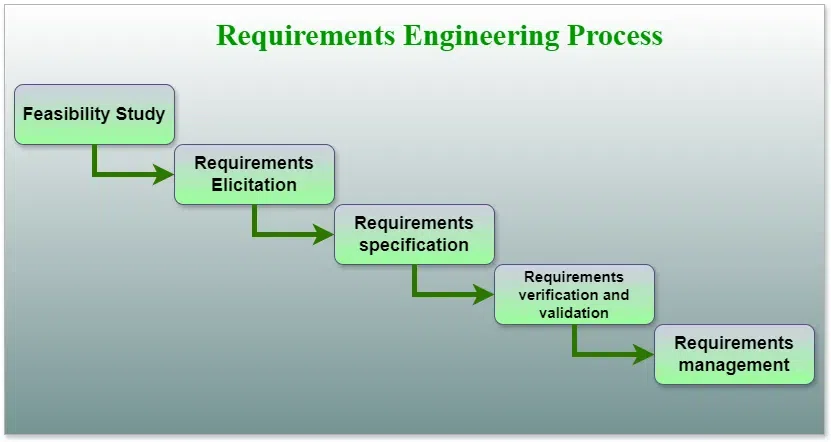
Requirement specifications, Requirements modeling

**Requirements Engineering** is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.

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.

**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.

**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.

**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.

**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.**

 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**

[**Functional Requirements**](https://www.geeksforgeeks.org/functional-vs-non-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.

[**Non-Functional Requirements**:](https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/) 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.

**Constraints:**These describe any limitations or restrictions that must be considered when developing the software system.

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:

* The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.
* The requirements should be complete in every sense.
* 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.

**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.

**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.**

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.**

**Requirements Management**

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:**

**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.

**Version control**: This involves keeping track of different versions of the requirements document and other related artifacts.

**Traceability**: This involves linking the requirements to other elements of the development process, such as design, testing, and validation.

**Communication:**This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.

**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
* Roleplaying
* Prototyping

Requirements documentation.

Use cases and User stories.

A use case diagram is a type of behavioral diagram created using Unified Modeling Language (UML) that represents the functionality of a system from a user's perspective. It depicts the system's use cases, the actors that interact with the system, and the relationships between the actors and the use cases. This helps in understanding the requirements and the interactions between users and the system. The main components of a use case diagram include:

1. **Actors:** Entities that interact with the system (e.g., users, other systems).
2. **Use Cases:** Specific functions or actions that the system can perform.
3. **System Boundary:** A rectangle that defines the scope of the system.
4. **Associations:** Lines that connect actors to use cases, indicating interaction.
5. **Relationships:** Includes generalization (inheritance between use cases), extend (optional behavior), and include (mandatory behavior).,**How do you identify actors in a use case diagram?**

Actors in a use case diagram can be identified by:

* Analyzing the roles that interact with the system.
* Considering all types of users (human and non-human) that will use the system.
* Identifying external systems or hardware that interface with the system.
* Reviewing the system requirements to find entities that interact with the system to perform a function.

### Relationships and Notations:**What is an association in a use case diagram?**An association is a line connecting an actor to a use case, indicating that the actor participates in that use case.**What is the difference between "include" and "extend" relationships?Include**: This relationship indicates that a use case incorporates the behavior of another use case. It is used when a use case is always performed as part of another use case.**Extend**: This relationship shows optional behavior. It indicates that a use case may extend the behavior of another use case, usually based on some condition.

### Examples and Scenarios **use case diagram for an online shopping system.**

* 1. **Actors**: Customer, Admin, Payment Gateway.
  2. **Use Cases**: Browse Products, Add to Cart, Checkout, Make Payment, Manage Orders, Update Inventory.
  3. **Relationships**:
     1. Customer → Browse Products, Add to Cart, Checkout, Make Payment.
     2. Admin → Manage Orders, Update Inventory.
     3. Checkout → (include) Make Payment (using Payment Gateway).

**Create a use case diagram for an ATM system.****or an e-commerce website**

Use Case Diagram for Library Management System

A use case diagram in UML helps to show the various ways in which a user could interact with a system. For a Library Management System, the use case diagram helps visualize the interactions between users (actors) and the system’s functionalities (use cases). This diagram provides a clear, simplified way to understand how the system operates and what it offers to its users.

Use Case Diagram, referred to as a Behavior model or diagram.

Use Case Diagram for Library Management System

Let’s visually map out the relationships and interactions. Below is the textual description of what the diagram would look like:

**Actors:**

* User (Staff or Student)
* Librarian

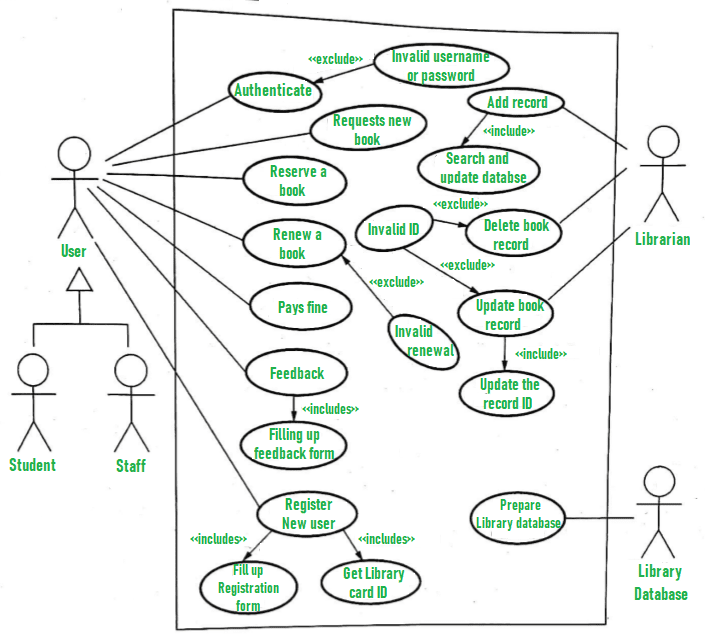
**Use Cases:**

* Register New User
* Issue Library Card
* Request New Book
* Reserve Book
* Renew Book
* Pay Fine
* Fill Feedback Form
* Manage Records
* Delete Records
* Update Database

**System Boundary:**

The system boundary will encompass all the use cases mentioned above.

**Below is the use case diagram of a Library Management System**



*Use Case Diagram For Library Management System*

Explanation of Use Case Diagram of a Library Management System

Here, we will understand the designing use case diagram for the library management system. Some scenarios of the system are as follows :

**User** who registers himself as a new user initially is regarded as staff or student for the library

system.

For the user to get registered as a new user, registration forms are available that is needed to be filled out by the user.

After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to cardholder or user.

After getting the library card, a new book is requested by the user as per there requirement.

After, requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.

Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.

If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.

User can fill the feedback form available if they want to.

***Librarian has a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying fine.***

***Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.***

***Updating database is the important role of Librarian.***

**user story**

A user story is a short, simple description of a feature or functionality from the perspective of the end user. It usually follows the format: "As a [type of user], I want [some goal] so that [some reason]."

The components of a user story typically include:

* **Title**: A short, descriptive title.
* **User Role**: The type of user the story is about.
* **Goal**: What the user wants to achieve.
* **Reason**: Why the user wants to achieve this goal.
* **Acceptance Criteria**: Conditions that must be met for the story to be considered complete.
* **Description**: Additional details or context (optional).

**Why are user stories important in Agile development** User stories help ensure that development is user-centered, provide a clear and concise way to capture requirements, facilitate communication among stakeholders, and help prioritize work based on user needs.

**As a registered user, I want to reset my password so that I can regain access to my account if I forget my password.**

User stories are less formal and more iterative than traditional requirements documentation. They are typically part of agile methodologies like Scrum and are used to create a product backlog, where each story is a placeholder for a future task. The key elements of user stories include:

**Role**: Who the story is about (e.g., end-user, administrator).

**Goal**: What the user wants to accomplish.

**Benefit**: Why the user wants to achieve this goal.

User stories also come with acceptance criteria, which define the conditions that must be met for the story to be considered complete. For example, acceptance criteria for the password reset story might include:

The user receives a password reset link via email.

The link allows the user to enter a new passwo,

**How would you write**

* **A user story for a login feature in an application ,**
* A **user story for a feature that allows users to reset their passwords.**
* **A user story for a feature that allows users to filter search results by various criteria in an online job portal.**
* **A user story for implementing a notification system in a social media app.**

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**