**Unit-2**

## Software Requirements

## Software requirements

* The software requirements are description of features and functionalities of the target system.
* It is the description of what the system should do.
* Requirements convey the expectations of users from the software product.
* Requirements should be clear ,correct, well defined.

### Functional Requirements

Requirements, which are related to functional aspect of software fall into this category.

They define functions and functionality within and from the software system.

### Non-Functional Requirements

Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.

Non-functional requirements include -

* Security
* Logging
* Storage
* Configuration
* Performance
* Cost
* Interoperability
* Flexibility
* Disaster recovery
* Accessibility

**User Requirements**User requirements are statements in **natural language** along with corresponding **diagrams** (tables, forms, intuitive diagrams) detailing the services provided by the system and operational constraints it must comply with.

Essentially, it entails the requirement that the user wants or ability to perform a functionality or action with the system. So, it outlines the **activities a user can perform** with the system. User requirements entail both functional and non-functional requirements that are understandable even by users without technical knowledge.

**User Requirements specification/document** is a document usually used in software engineering that specifies what the user expects the software to be able to do.It is a contractual agreement.

* Easy and simple to operate
* Quick response.
* Effectively using operational errors.
* Customer support

**System Requirements**

System requirements consist of a structured document that details the **system’s functions, services and operational constraints**. It’s mainly written and serves as a tool for the developers of the system to build and implement the system design. It contains the functionality that’s needed by the system in order for it to fulfill the user requirements.

It outlines both the **functional and non-functional requirements** the system should fulfil. Hence, it consists of all things the system must contain to construct or develop it. They’re often described as the expanded version of user requirements that engineers use at the beginning of a system’s design. Hence, the system requirements serve as a blueprint for the developer to follow defining the parts of a system that are to be implemented, so it acts as a contract between the client and contractor.

**Interface specification**

Application programming interface are specified in SRS.

What kind of interface customer desire.

**Requirements engineering (RE)**

**Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process.

**Requirement Engineering Process**

It is a five-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software requirement management

1. Feasibility Study:

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

**Types of Feasibility:**

**1.Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.

**2. Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.

1. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

**2. Requirements Elicitation and analysis**

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

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects etc.

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.
* **Domain Analysis** Every software falls into some domain category. The expert people in the domain can be a great help to analyze general and specific requirements.

**Brainstorming** An informal debate is held among various stakeholders and all their inputs are recorded for further requirements analysis.

3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

An SRS (Software Requirements Specification) is a document that outlines the requirements for a software project. A well-written SRS is essential for a successful software development project. Here are some tips for writing a good SRS for your project:

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

* **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
* **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
* **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

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

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

**Requirements Validation Techniques**

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

**Software Requirement Management:**

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

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

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

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

Tools support for Requirement Engineering:

* Observation
* Questionnaires(interview,survey and poll )
* Use case
* User stories
* Mind mapping
* Role playing
* Prototyping
* Requirement workshop

**System model­­**

In **software engineering**, a **system model** is a graphical representation that depicts the components of a software system and how they interact with each other. These models help software engineers understand, design, and communicate about the system.

It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the **Unified Modeling Language** (**UML**).

Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

Here are some common types of system models

1. context model
2. Behavioral model
3. Data models
4. Object models
5. Structured model

Models can explain the system from **different perspectives**:

* An **external** perspective, where you model the context or environment of the system.
* An **interaction** perspective, where you model the interactions between a system and its environment, or between the components of a system.
* A **structural** perspective, where you model the organization of a system or the structure of the data that is processed by the system.
* A **behavioral** perspective, where you model the dynamic behavior of the system and how it responds to events.

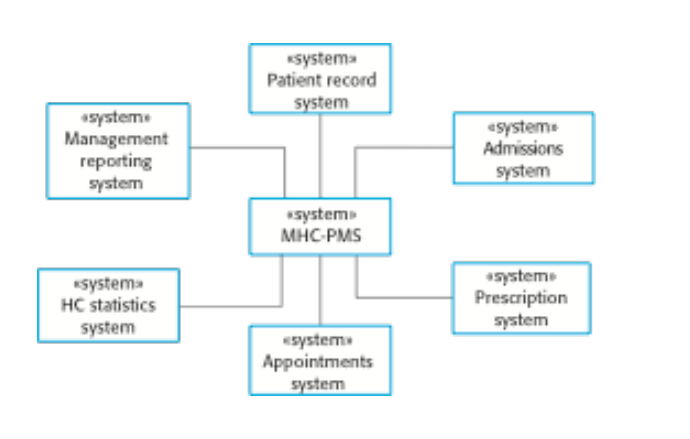
UML

UML diagram types

* Activity diagrams, which show the activities involved in a process or in data processing .
* Use case diagrams, which show the interactions between a system and its environment.
* Sequence diagrams, which show interactions between actors and the system and between system components. ²
* Class diagrams, which show the object classes in the system and the associations between these classes. State diagrams, which show how the system reacts to internal and external events.
* **Context models** Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
* Social and organisational concerns may affect the decision on where to position system boundaries.
* Architectural models show the system and its relationship with other systems.

System boundaries System boundaries are established to define what is inside and what is outside the system.

* They show other systems that are used or depend on the system being developed.
* The position of the system boundary has a profound effect on the system requirements. Defining a system boundary is a political judgement .
* There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.
* The context of the MHC-PMS (mental health care –patient management system)



* **Structural models**  **Structural models** (also known as **component models** or **static model.** Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
* Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
* You create structural models of a system when you are discussing and designing the system architecture.

**Behavioral models**  Behavioral models are used to describe the dynamic behavior of an executing system. This behavior can be modeled from the perspective of the data processed by the system, or by the events that stimulate responses from a system

The behavioral Model describes the overall behavior of the system. To represent system behavior, two models use one is the Data processing model, i.e., DFD (Data Flow Model), and another is the state machine model, i.e., state diagram.

**Data Flow Diagram:**

* We model the system data processing using the functional model, a graphical representation of an enterprise function within a defined scope. It shows the end-to-end processing of data. It takes an input–process–output view of the system. The representation of data object flow in the analysis model facilitates easy conversion into software. This diagram enables a software engineer to develop a model of the information domain and Functional domain at the same time. The data processing model is the core modeling activity in structured analysis.
* State diagram The state is the mode or condition of being. A state diagram is a dynamic model that shows changes in an object’s state during its lifetime in response to an event. It is used to help the developer better understand any complex functionality of specialized areas of the system. It depicts the system’s dynamic behavior in response to an external and internal event. Likewise, it uses various notations to show the function, such as initial, final, state, transition, event, action, history site, signal, the action inside the state, and self-transition.

#### Data Model

* Analysis modeling starts with data modeling. The software engineer defines all data objects required for the system. It describes the logical structure of the data processed by the system. ER (Entity Relation Attribute Model) is one type of data model that illustrates the entities in the system, their attributes, and the relationships between entities. Elements of data modeling help to provide appropriate information to understand the problem.
* Data modeling uses the concept of cardinality. ER diagram consists of information required for each entity or data object and shows the relation between objects. It shows the structure of the data in terms of the tables. Three relations exist between these objects – one-to-one, one-to-many, and many-to-many.
* Object Model Object model consists of former properties and procedures and methods which tell us how to access these properties. The goal of class modeling is to describe the object. An object is a concept, abstraction, or thing which identifies that has meaning for an application. The object model shows individual objects and the relation between them. It helps document test cases and discusses examples. Understanding uncovered rules, definitions of resources, and their relationship is beneficial. Object diagrams are valuable because they support the investigation of requirements by modeling the examples from the problem domain.