Unit -II

SOFTWARE REQUIREMENT SPECIFICATION

Requirement analysis and specification – Requirements gathering and analysis – Software Requirement Specification – Formal system specification – Finite State Machines – Petri nets – Object modelling using UML – Use case Model–Class diagrams – Interaction diagrams–Activity diagrams–State chart diagrams –Functional modeling–Data Flow Diagram.

---

1. Requirement analysis and specification

The **Requirement Analysis** and **Specification** phase starts after the feasibility study stage is complete and the project is financially viable and technically feasible. This phase ends when the requirements specification document has been developed and reviewed. It is usually called the **Software Requirements Specification (SRS)** Document.

These activities are usually carried out by a few experienced members of the development team and it normally requires them to spend some time at the customer site. The engineers who gather and analyse customer requirements and then write the requirements specification document are known as **system analysts** in the software industry. System analysts collect data about the product to be developed and analyse the collected data to conceptualize what exactly needs to be done. After understanding the precise user requirements, the analysts analyse the requirements to weed out inconsistencies, anomalies and incompleteness.

We can conceptually divide the requirements gathering and analysis activity into two separate tasks:

- Requirements gathering
- Requirements analysis

REQUIREMENTS GATHERING:

It is also known as **Requirements Elicitation**. The primary objective of the requirements- gathering task is to **collect the requirements from the stakeholders**. A **stakeholder** is a source of the requirements and is usually a person or a group of persons who either directly or indirectly are concerned with the software.

1. Studying existing documentation:

The analyst usually studies all the available documents regarding the system to be developed before visiting the customer site. Customers usually provide a statement of purpose (SoP) document to the developers. Typically, these documents might discuss issues such as the context in which the software is required.

2. Interview:

Typically, there are many different categories of users of the software. Each category of users typically requires a different set of features from the software. Therefore, it is important for the analyst to first identify the different categories of users and then determine the requirements of each.

3. Task analysis:

The users usually have a black-box view of software and consider the software as something that provides a set of services. A service supported by the software is also called a **task**. We can therefore say that the software performs various tasks for the users. In this context, the analyst tries to identify and understand the different tasks to be performed by the software. For each identified task, the analyst tries to formulate the different steps necessary to realize the required functionality in consultation with the users.

4. Scenario analysis:

A task can have many scenarios of operation. The different scenarios of a task may take place when the task is invoked under different situations. For different types of scenarios of a task, the behaviour of the software can be different.

5. Form analysis:

Form analysis is an important and effective requirement-gathering activity that is undertaken by the analyst when the project involves automating an

existing manual system. During the operation of a manual system, normally several forms are required to be filled up by the stakeholders, and in turn, they receive several notifications.

In form analysis, the existing forms and the formats of the notifications produced are analyzed to determine the data input to the system and the data that are output from the system.

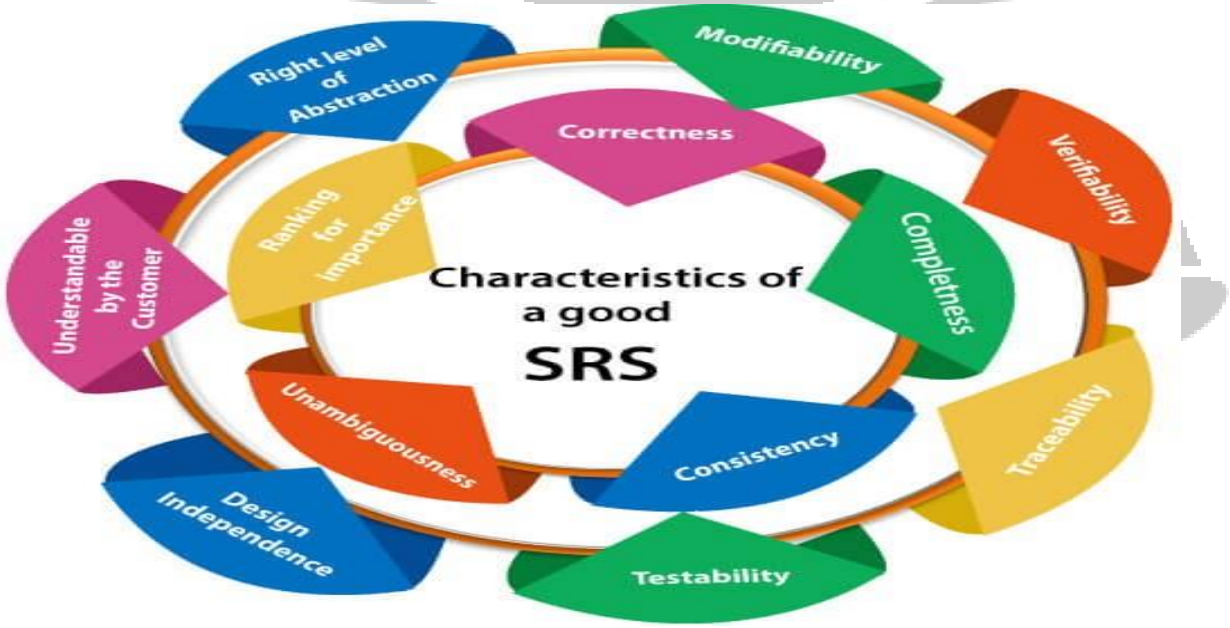
REQUIREMENT ANALYSIS AND SPECIFICATION:

After requirements gathering is complete, the analyst analyses the gathered requirements to form a clear understanding of the exact customer requirements and to weed out any problems in the gathered requirements. It is natural to expect the data collected from various stakeholders to contain several contradictions, ambiguities, and incompleteness.

The main purpose of the requirements analysis activity is to analyse the gathered requirements to remove all ambiguities, incompleteness, and inconsistencies from the gathered customer requirements and to obtain a clear understanding of the software to be developed.

During requirements analysis, the analyst needs to identify and resolve three main types of problems in the requirements

CHARACTERISTICS OF GOOD SRS



Following are the features of a good SRS document:

1. **Correctness:** User review is used to provide the accuracy of requirements stated in the SRS. SRS is said to be perfect if it covers all the needs that are truly expected from the system.

2. **Completeness:** The SRS is complete if, and only if, it includes the following elements:

(1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

(2). Definition of their responses of the software to all realizable classes of input data in allavailable categories of situations.

(3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions ofall terms and units of measure.

3. **Consistency:** The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:

(1). The specified characteristics of real-world objects may conflicts. For example,one requirement as tabular but in another as textual.

Unambiguousness: SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.

Ranking for importance and stability: The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.

Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable.

Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.

Modifiability: SRS should be made as modifiable as likely and should be capable of quicklyobtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.

Verifiability: SRS is correct when the specified requirements can be verified with a cost- effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.

Traceability: The SRS is traceable if the origin of each of the requirements is clear and if it facilitates the referencing of each condition in future development or enhancementdocumentation.

Design Independence: There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

Understandable by the customer: An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

The right level of abstraction: If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas,for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

The format of an output report may be described in

- Anomaly
- Inconsistency
- Incompleteness

Anomaly: It is an anomaly is an ambiguity in a requirement. When a requirement is anomalous, several interpretations of that requirement are possible. Any anomaly in any of therequirements can lead to the development of an incorrect system, since an anomalous requirement can be interpreted in several ways during development.

Inconsistency: Two requirements are said to be inconsistent if one of the requirementscontradicts the other.

Incompleteness: An incomplete set of requirements is one in which some requirements have been overlooked. The lack of these features would be felt by the customer much later, possibly while using the software. Often, incompleteness is caused by the inability of the customer to visualise the system that is to be developed and to anticipate all the features that would be required

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY
PALKULAM, KANYAKUMARI
OBSERVE OPTIMIZE OUTSPREAD