Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT I

Chapter 2 – Software Requirements

Concepts:

Requirements

Characteristics of software requirements

Types of Non – functional requirements

User requirements

System requirements

Documentation of Software Requirement Specification

Requirements

Requirement is a condition or capability possessed by the software or system component in order to solve a real-world problem. The problems can be to automate a part of a system, to correct shortcomings of an existing system, to control a device, and so on. IEEE defines requirement as (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).'

1) Requirements describe how a system should act, appear or perform. For this, when users request for software, they provide an approximation of what the new system should be capable of doing. Requirements differ from one user to another and from one business process to another.

In some cases, a requirement is simply a high level, abstract statement of service that the system should provide or a constraint on the system. At the other extreme it is a detailed formal definition of a system function. Some of the problems that arise during the requirement engineering process are a result of failing to make a clear separation between these different levels of description.

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 users. The requirements themselves are the descriptions of the system services and constraints that are generated during the requirement engineering process.

The requirement engineering is distinguished between the terms:

User requirements – It means the high-level abstract requirements. User requirements are statements in natural language plus diagrams of what services the system is expected to provide and the constraint under which it much operates.

System requirements – It sets out the system functions, services, operational constraints in detail. System requirement means the detail description of what the system should do. The system requirement document should be precise. It should define exactly of is to be implemented. It may be a part of contract between system buyer and software developer.

Example: Library System

User Requirement - Library system shall keep track of all data required by copyright licensing agencies**.**

System Requirement –

- On making a request for a document from the library system the requester shall be presented with a form that records details of user and system mode.
- Library system request forms shall be stored in the system for 5 years from the date of request.
- All library system request forms must be indexed by user, by the name of the material requested and by the supplier of the request.
- Library system shall maintain a log of all requests that have been made to the system

Classification of software requirements

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. **Software system requirements are often classified as functional requirements, non – functional requirements or domain requirements.**

1) **Functional requirements**: In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. In some cases, the functional requirements may also explicitly state what the system should not do. These requirements depend on the type of software being developed, the expected users of the software and the general approach taken by the organization when writing requirements.

Functional requirements in software engineering help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform. **Functional requirements indicate what a system must do.**

2) **Non – functional requirements** - A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load? A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. On-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is > 10000. Description of non-functional requirements is just as critical as a functional requirement. **Non-functional requirements support them and determine how the system must perform.**

Types of non – functional requirements

These are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards. Non – functional requirements often apply to the system as a whole.

The types of non – functional requirements are:

1) **Product requirements**: Requirements which specify that the delivered product must behave in a particular way e.g., execution speed, reliability, etc.

2) **Organizational requirements**: Requirements which are a consequence of organizational policies and procedures e.g., process standards used, implementation requirements, etc.

3) **External requirements**: Requirements which arise from factors which are external to the system and its development process e.g., interoperability requirements, legislative requirements, etc.

Metrics of specifying Non – functional requirements	
Speed	Processed transactions/Event response time, Screen refresh time.
Size	K bytes, Number of RAM chips
Ease of use	Training time, Number of help frames
Reliability	Mean time to failure, Portability of unavailability, Rate of failure occurrence, viability
Robustness	Time to restart after failure, Percentage of events causing failure, Portability of data corruption on failure.
Portability	Percentage of target-dependent statements, Number of target systems

User Requirements

User requirements, often referred to as user needs, describe what the user does with the system, such as what activities that users must be able to perform. User requirements are generally documented in a User Requirements Document (URD) using narrative text. User requirements are generally signed off by the user and used as the primary input for creating system requirements. The user requirements for a system should describe the functional and non-functional requirements so that they are understandable by system users without detailed technical knowledge. They should only specify the external behavior of the system and should avoid as far as possible system design characteristics. An important and difficult step of designing a software product is determining what the user actually wants it to do.

System Requirements

System requirements are expanded versions of the user requirements that are used by software engineers as the starting point for the system design. System requirements are the building blocks developers use to build the system. These are the traditional “shall” statements that describe what the system “shall do”. They add detail and explain how the user requirements should be provided by the system. They may be used as apart of the contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system.

Domain requirements

Domain requirements reflect the environment in which the system operates so, when we talk about an application domain, we mean environments such as train operation, medical records, e-commerce etc. Domain requirements may be expressed using specialized domain terminology or reference to domain concepts. Because these requirements are specialized, software engineers often find it difficult to understand how they are related to other system requirements.

Domain requirements are important because they often reflect fundamentals of the application domain. If these requirements are not satisfied, it may be impossible to make the system work satisfactorily.

Documentation of the software requirements

The software requirements document (sometimes called the software requirement specification or SRS) is the official statement of what the system developers should implement. It should include both the user requirements for a system and a detailed specification of the system requirements. In some cases, the user and system requirements are integrated into a single description. In other cases, the user requirements are defined in an introduction to the system requirements specification. If there are a large number of requirements the detailed system requirements may be presented in a separate document.

The requirements documents have a diverse set of users ranging from the senior management of the organization that is paying for the system to the engineers responsible for developing the software. The diversity of possible users means that the requirements has to be a compromise

between communicating the requirements to customers, defining the requirements in precise detail for developers and testers including information about possible system evolution.

Users of a requirement document	**System customers**	Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
	Managers	Use the requirements document to plan a bid for the system and to plan the system development process
	System Engineers	Use the requirements to understand what system is to be developed
	System Test Engineers	Use the requirements to develop validation tests for the system
	System Maintenance Engineers	Use the requirements to understand the system and the relationship between its parts

The most widely known standard is IEEE/ANSI 830-1998 (IEEE,1998). This IEEE standard suggests the following structure for requirements documents:

1. Introduction

1.1 Purpose of the requirements document

1.2 Scope of the product

1.3 Definitions, acronyms and abbreviations

1.4 References

Introduction: This provides an overview of the entire information described in SRS. This involves purpose and the scope of SRS, which states the functions to be performed by the system. In addition, it describes definitions, abbreviations, and the acronyms used. The references used in SRS provide a list of documents that is referenced in the document.

Overall description: It determines the factors which affect the requirements of the system. It provides a brief description of the requirements to be defined in the next section called 'specific requirement'. It comprises the following sub-sections.

Product perspective: It determines whether the product is an independent product or an integral part of the larger product. It determines the interface with hardware, software, system, and communication. It also defines memory constraints and operations utilized by the user.

Product functions: It provides a summary of the functions to be performed by the software. The functions are organized in a list so that they are easily understandable by the user:

User characteristics: It determines general characteristics of the users.

Constraints: It provides the general description of the constraints such as regulatory policies, audit functions, reliability requirements, and so on.

Assumption and dependency: It provide a list of assumptions and factors that affect the requirements as stated in this document.

Specific requirements: These determine all requirements in detail so that the designers can design the system in accordance with them. The requirements include description of every input and output of the system and functions performed in response to the input provided. It comprises the following subsections.

External interface: It determines the interface of the software with other systems, which can include interface with operating system and so on. External interface also specifies the

interaction of the software with users, hardware, or other software. The characteristics of each user interface of the software product are specified in SRS. For the hardware interface, SRS specifies the logical characteristics of each interface among the software and hardware components. If the software is to be executed on the existing hardware, then characteristics such as memory restrictions are also specified.

Functions: It determines the functional capabilities of the system. For each functional requirement, the accepting and processing of inputs in order to generate outputs are specified. This includes validity checks on inputs, exact sequence of operations, relationship of inputs to output, and so on.

Performance requirements: It determines the performance constraints of the software system**.**

Performance requirement is of two types: static requirements and dynamic requirements.

Static requirements (also known as capacity requirements) do not impose constraints on the execution characteristics of the system. These include requirements like number of terminals and users to be supported. **Dynamic requirements** determine the constraints on the execution of the behavior of the system, which includes response time (the time between the start and ending of an operation under specified conditions) and throughput (total amount of work done in a given time).

Logical database of requirements: It determines logical requirements to be stored in the database. This includes type of information used, frequency of usage, data entities and relationships among them, and so on.

Design constraint: It determines all design constraints that are imposed by standards, hardware limitations, and so on. Standard compliance determines requirements for the system, which are in compliance with the specified standards. These standards can include accounting procedures and

report format. Hardware limitations implies when the software can operate on existing hardware or some pre-determined hardware. This can impose restrictions while developing the software design. Hardware limitations include hardware configuration of the machine and operating system to be used.

Software system attributes: It provide attributes such as reliability, availability, maintainability and portability. It is essential to describe all these attributes to verify that they are achieved in the final system.

Other Requirements: Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.