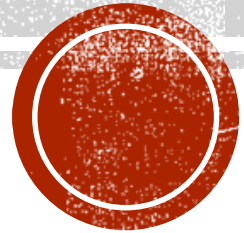
UNIT-I PART-II



- **Software Requirements:** Functional and non-functional requirements, user requirements, system requirements, interface specification, the software requirements document.

- **Requirements engineering process:** Feasibility studies, requirements elicitation and analysis, requirements validation, requirements management.

SOFTWARE REQUIREMENTS

- Introduction
- Functional and Non Functional Requirements
- User Requirements
- System requirements
- The software Requirement document

INTRODUCTION

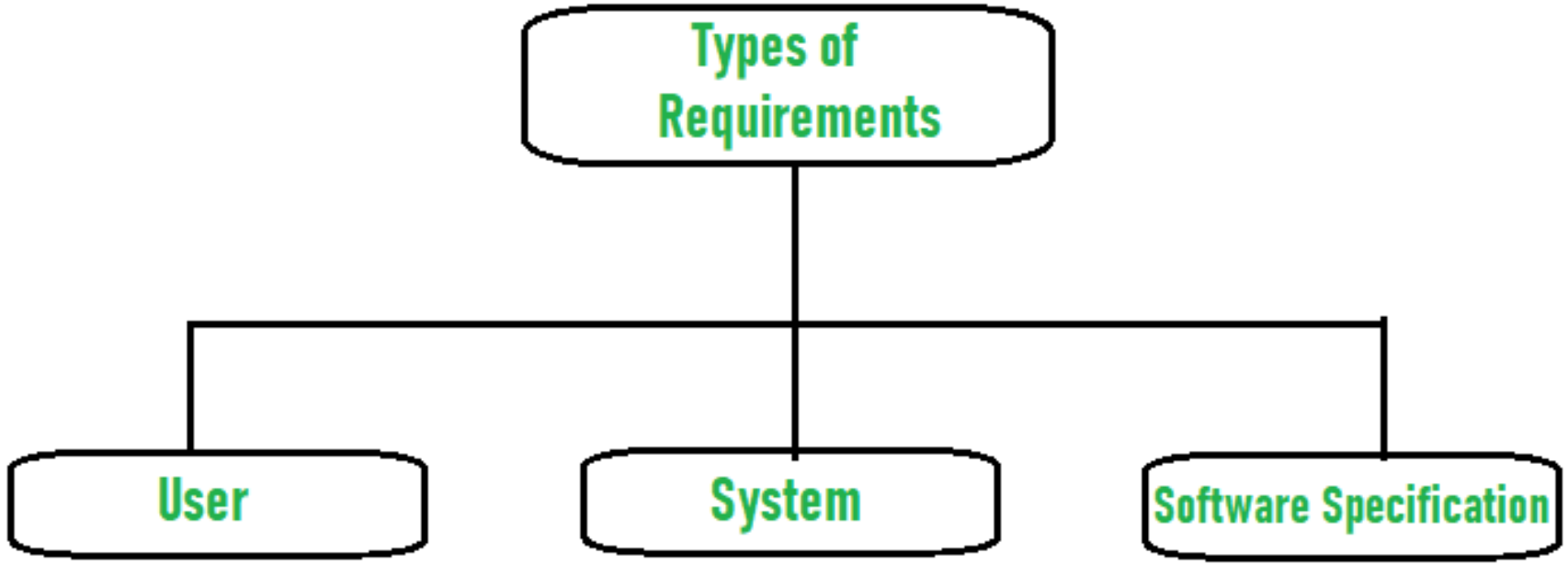
Requirement Engineering:

The process to gather the ***requirements from client,*** analyze and document them is known as "Requirement Engineering".

The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'Software Requirements Specification' document.

REQUIREMENTS

- Requirements are descriptions of services that a software system must provide and the constraints under which it must operate.
- Requirements help to describe what the software should do.



TYPES OF REQUIREMENTS

- **User Requirements:**
  - It is a collection of statements in *natural language* plus description of the services the system provides and its operational constraints. It is written for *customers.*
- **System Requirements:**
  - It is a structured documents that gives the detailed description of the system services. It is written *as a contract between client and contractor.*
- **Software specification:**
  - It is a detailed software description that can serve as basis for *design or implementation.* Typically it is written for software developers.

SOFTWARE REQUIREMENTS

- The software requirements are ***description of features*** and ***functionalities*** of the target system. Requirements convey the *expectation of users from the software product.*
- The requirements can be obvious/hidden, known (or) unknown, expected (or) unexpected from clients point of view.
- Software system requirements can be classified as functional and non functional requirements.

FUNCTIONAL REQUIREMENTS

- Functional requirements define ***a function that a system or system element*** must be qualified to perform and must be documented in different forms.
- The functional requirements describe the ***behavior of the system*** as it correlates to the ***system's functionality***.
- Functional requirements should be written in a simple language, so that it is easily understandable.
- Examples: functional requirements are authentication, business rules, audit tracking, certification requirements, transaction corrections, etc.

- These requirements allow us to verify whether the application provides all functionalities mentioned in the ***application's functional requirements.*** They support tasks, activities, user goals for easier project management.
- The most common way is that they are documented in the ***text form.***
- Other formats of preparing the functional requirements are ***use cases, models, prototypes, user stories, and diagrams.***

- ***Functional requirements specify product features.***
  - Determine what the system has to do
  - How the system should react to particular inputs
  - How the system should behave in particular situations
- In principle, requirements should be both
- **Completeness**: All services required by the user should be defined
- **Consistent**: there should be no conflicts or contradictions in the descriptions of the system facilities.
- ********In practice, it is difficult to achieve a completeness and consistent requirements document for a large system.*******
- ***Ex: Users can transfer money between accounts.***

NON FUNCTIONAL REQUIREMENTS

- Non-functional requirements are ***not related to the software's functional aspect.***
- They can be the necessities that specify the criteria that can be ***used to decide the operation instead of specific behaviors of the system.***
- Basic non-functional requirements are - usability, reliability, security, storage, cost, flexibility, configuration, performance, legal or regulatory requirements, etc.
- They are divided into two main categories:
- ***Execution qualities***
- ***Evolution qualities***

- **Execution qualities** like security and usability, which are observable at run time.
- **Evolution qualities** like *testability, maintainability, extensibility, and scalability* that embodied in the static structure of the software system.
- Non-functional requirements ensure that the software system must follow the ***legal and dedication rules***. The impact of the non-functional requirements is not on ***the functionality of the system, but they impact how it will perform.*** For a well-performing product, at least some of the non-functional requirements should be ***met***

NON FUNCTIONAL REQUIREMENTS

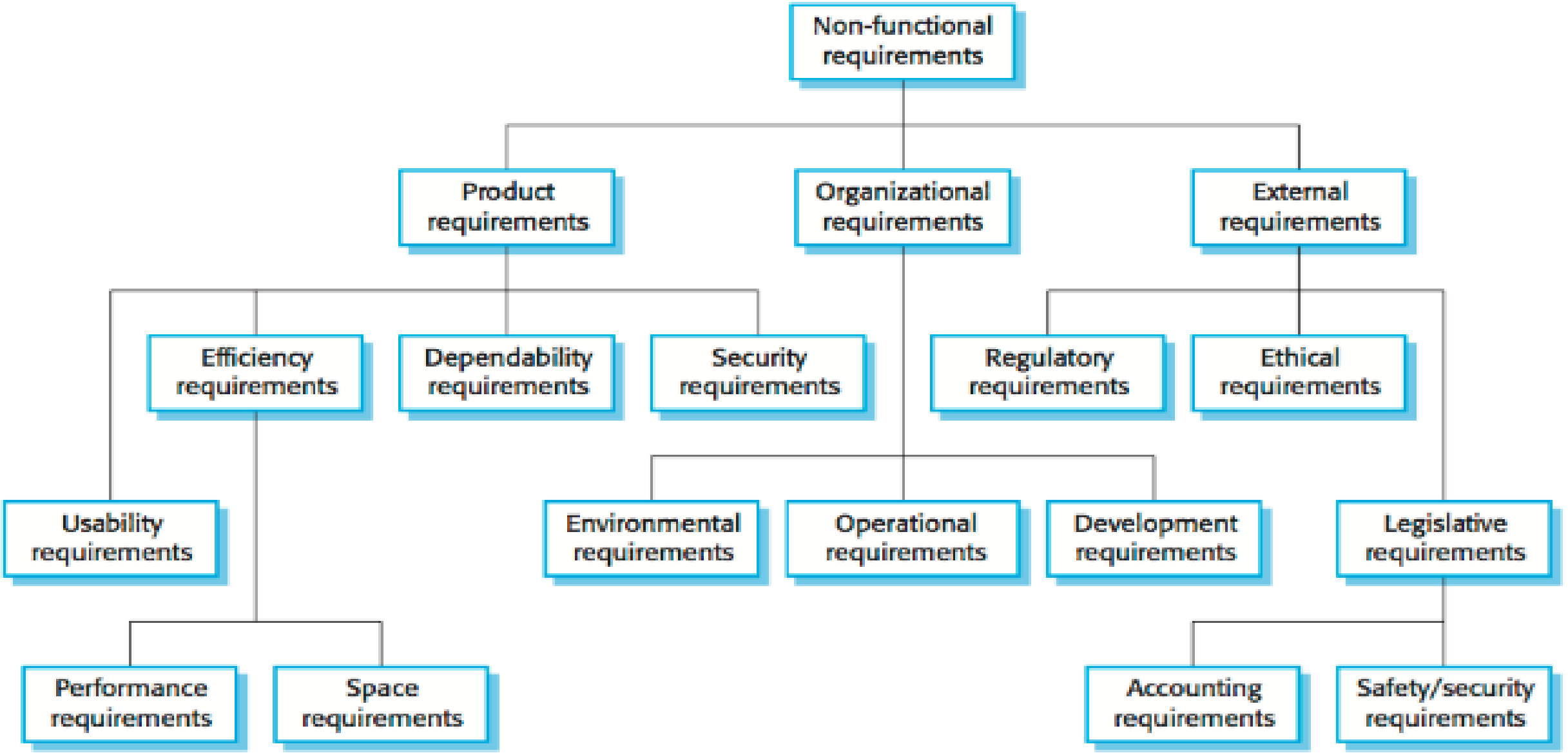
- Non functional requirements are **more critical** than functional requirements. If the non functional requirements ***do not meet*** then the complete system is of ***no use or useless.***
- Even though they don't directly relate to ***specific features or functions.*** Instead, they define *how a system performs and behaves under various conditions*.
- Non functional requirements are 3 types
  - ***Product Requirements***
  - ***Organizational Requirements***
  - ***External Requirements***

- Product requirements
  - Requirements which specify that the ***delivered product*** must behave in a particular way

  Ex: execution speed, reliability, etc.
- Organizational requirements
  - Requirements which are a ***consequence of organizational policies and procedures***

  Ex: process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are ***external to the system and its development process***

  Ex: interoperability requirements, legislative requirements, etc.


Non-functional requirements
Product requirements
Organizational requirements
External requirements
Efficiency requirements
Dependability requirements
Security requirements
Regulatory requirements
Ethical requirements
Usability requirements
Environmental requirements
Operational requirements
Development requirements
Legislative requirements
Performance requirements
Space requirements
Accounting requirements
Safety/security requirements

DIFFERENCE BETWEEN FUNCTIONAL & NONFUNCTIONAL REQUIREMENTS

Functional Requirements	Non functional Requirements
Functional requirements help to understand the ***functions of the system.***	They help to understand the ***system's performance.***
functional requirements are mandatory.	While non-functional requirements are not mandatory.
They are easy to define.	They are hard to define.
It concentrates on the user's requirement.	It concentrates on the ***expectation and experience*** of the user.
It helps us to verify the software's ***functionality.***	It helps us to verify the software's **performance.**
These requirements are specified by th***e user.***	These requirements are specified by ***the software developers, architects, and technical persons.***
Examples of the functional requirements are - Authentication of a user on trying to log in to the system.	Examples of the non-functional requirements are - The background color of the screens should be light blue.
Eg: for a library management system, allowing user to read the article online is a functional requirement.	Eg: For a library management system, for a user who wishes to read the article online must be authenticated first.
Ex:A Verification email is sent to user whenever he/she registers for the first time on some software system.	Ex:The site should load in 3 seconds when the number of simultaneous users are > 10000

USER REQUIREMENTS

- The user requirements should describe ***functional and non functional requirements*** in such away that they are understandable by ***system users who don't have detailed technical knowledge***.
- User requirements are define using natural language, tables and diagrams because these are the representations that can be understood by all users.
- **Guidelines for writing user requirements :**
- Prepare a standard format and use it for all requirements.
- Easy and simple to Operate
- Quik response
- Effectively handling operational errors
- The text which is mentioning the key requirements should be highlighted.
- Avoid the use of computer terminologies. It should be written in simple language.
- Customer support

- **user requirements specification (URS):**
- A user requirement specification (URS) is a document that outlines a system's requirements from the end user's perspective. It serves as a formal document that captures and defines ***what users expect from a system, product, or service***. The URS is a crucial step in the early stages of system development or project planning, as it provides a foundation for design, development, and testing.
- **It is a contractual agreement**

SYSTEM REQUIREMENTS

- System requirements are more detailed specifications of system functions, services and constraints than user requirements.
- System requirements are the minimum and/or maximum hardware and software specifications that a system or application must meet in order to function properly.
  - They are intended to be a basis for designing the system.
  - The system requirements can be expressed using system models.
  - The requirements specify what the system does and design specifies how it does.
  - It should simply describes the external behavior of the system and its operational constraints.
  - For a complex software system design it is necessary to give all the requirements in detail.
  - Usually natural language is used to write system requirements specifications and user requirements.

SYSTEM REQUIREMENTS

- The advantage of specifying requirements using this method is that requirements become understandable & expressive.
- The information can be represented using tables(or) graphical models.
- one way of using graphical models is use of sequence diagram
- The sequence diagram represent the sequence of actions that user performs while interacting the system.

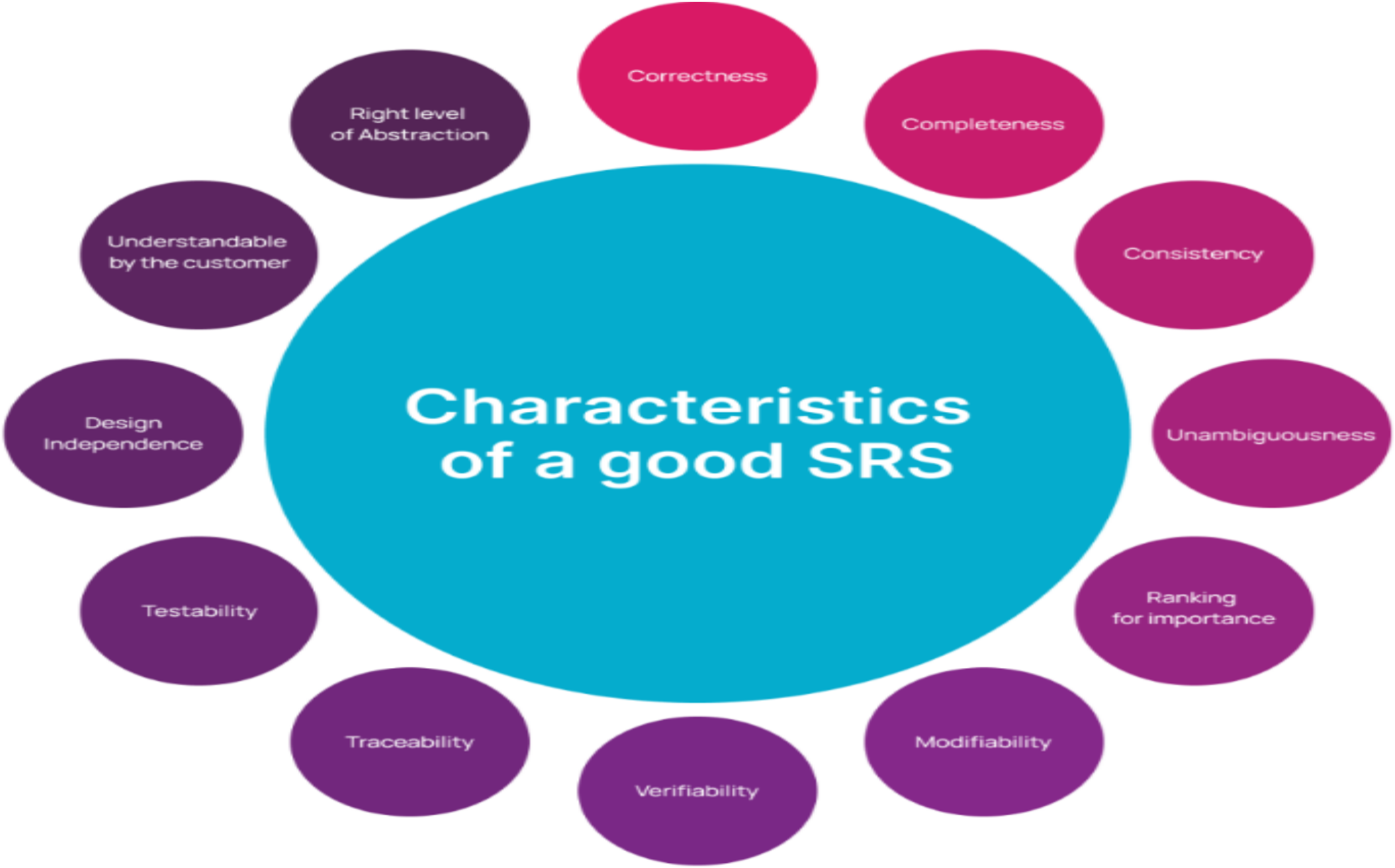
- **Characteristics of a Good System Requirements Document:**
- Completeness
- Correctness
- Modifiability
- Consistency
- Testability
- Design independence
- Understandable by the customer
- Traceability
- Verifiability

USER REQUIREMENTS	SYSTEM REQUIREMENTS
Written for customers	Written for implementation team
It Uses natural language	It Uses technical language
Describes the services and features provided by system	Describes the details services and features description and complete operations of a system
It May includes tables and diagrams	It May includes system models
Understandable by system user who don’t have technical knowledge	Understandable by implementation team who has technical knowledge
For client managers system users contract managers	System architecture, software developer client engineers.

SOFTWARE REQUIREMENT DOCUMENTS

- A software requirements specification (SRS) is a document that is created when a ***detailed description of all aspects of the software to be built must be specified before the project is to commence.*** It is important to note that a formal SRS is not always written.
- In fact, there are many instances in which effort expended on an SRS might be better spent in other software engineering activities. However, when software is to be developed by a third party, when a lack of specification would create severe business issues, or when a system is extremely complex or business critical, an SRS may be justified.

- First, the SRS could be written by the *client of a system*. Second, the SRS could be written by a *developer of the system*. The two methods create entirely various situations and establish different purposes for the document altogether.
  - The first case, SRS, is used to define the needs and expectation of the users.
  - The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.


Characteristics of a good SRS
Correctness
Completeness
Consistency
Unambiguousness
Ranking for importance
Modifiability
Verifiability
Traceability
Testability
Design Independence
Understandable by the customer
Right level of Abstraction

CHARACTERISTICS OF GOOD SRS

- Correct:
  - The SRS must be correct. That means all the requirements must be correctly mentioned
- Complete:
  - The SRS is said to be complete only in following situations.
    1. When SRS consists of all the requirements related to functionality, performance, attributes, design constraints (or) external interfaces.
    2. When expected responses to the input data is mentioned by considering validity and invalidity of an input.

- Unambiguous:
  - When requirements are understand correctly then only unambiguous SRS can be written.
  - Unambiguous specification means only one interpretation can be made from the specified requirements.
  - If for particular term there are multiple meanings then, those terms should be mentioned in *glossary with proper meaning.*
  - If there are no conflicts in the specified requirements then SRS is said to be consistent.
- Stability:
  - In SRS, it is not possible to specify all the requirements. The SRS must contain all the essential requirements. Each requirement must be *clear and explicit.*
- Verifiable:
  - The SRS should be written in such a manner that the requirements that are specified with in it must be *satisfied by the software.*

- **Modifiability:** SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.
- **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.
- **Testability:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

- 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.
- Traceable:
  - References of the requirements are correctly mentioned then such a requirement is called as traceable requirements.
  - Various types of traceability's are
    1. Forward traceability → Each requirement is referred in the SRS document by its unique name.
    2. Backward traceability → If the reference to the requirement is mentioned in earlier document, then it is backward traceability.

SOFTWARE REQUIREMENT SPECIFICATION COMPONENTS

- **Functionality:** It addresses what software supposed to do
- **Performance:** It addresses the speed, response timings, availability, recovery time, software function, etc.
- **External interface:** It addresses how does the software interact with people, the system's hardware, other hardware, and other software.
- **Attributes:** It addresses the portability, correctness, security, reliability, maintainability, etc.
- **Design constraints imposed on an implementation:** It addresses the required standards in effect, implementation language, policies for database Integrity, resource limits, operating environments, etc.

SOFTWARE REQUIREMENTS SPECIFICATION TEMPLATE

Table of Contents

SOFTWARE REQUIREMENTS SPECIFICATION TEMPLATE

1. Introduction
   1. Purpose of this Document → Describes the purpose of the document
   2. Scope of this Document → Describes the scope of this requirements definition effort this section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs.
   3. Overview → Provides a brief overview of the product defined as a result of the requirements elicitation process.
2. General Description
   - Describes the general functionality of the product such as similar system information, user characteristics, user objective, general constraints placed on design team.

3. Functional Requirements

This section lists the functional requirements in ranked order.

Description → A full description of the requirement.

Criticality →Describes how essential this requirement is to the overallsystem.

Technical Issues → Describes any design (or) implementation issues involved in satisfying this requirement.

Cost & Schedule → Describe the relative (or) absolute costs of the system.

Risks → Describes the circumstances under which this requirement might not able to be satisfied.

Dependencies with other requirements → describes interactions with other requirements.

4. Interface requirements:

This section describes how the software interfaces with other software products (or) users for input (or) output.

User Interface →describes how this product interfaces with the user.

GUI → describes the graphical user interface of present.

CLI → describes the command line interface if present

API → describes the application programming interface if present.

Hardware interface → describes interfaces to hardware devices.

Communications interface → describes network interfaces.

Software Interface → describes speed and memory requirements.

5. Performance Requirements → specifies speed and memory requirements.

6. Other Non functional requirements

7. Design constraints

Specifies any other particular non functional attributes required by the system. Such as security, reliability, reusability, maintainability, portability, extensibility, serviceability.

8. Operational scenarios

This section should describes a set of scenarios that illustrate, from the users perspective, what will be experienced when utilizing the system under various situations.

9. Preliminary schedule

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies and their tentative start/stop dates.

10. Preliminary Budget

This section provides an initial budget for the project.

11. Appendices

Definitions, acronyms, abbreviations

Provides definitions terms, and acronyms can be provided.

References → provides complete situations to all documents and meetings referenced.