Software Requirements

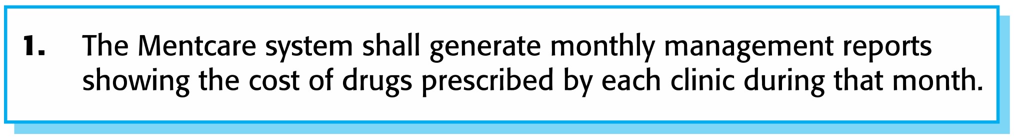
Requirements are descriptions of the services that a software system must provide and the constraints under which it must operate.

Requirements Engineering is the process of establishing the services that the customer requires from the system and the constraints under which it is to be developed and operated

Types of Requirements

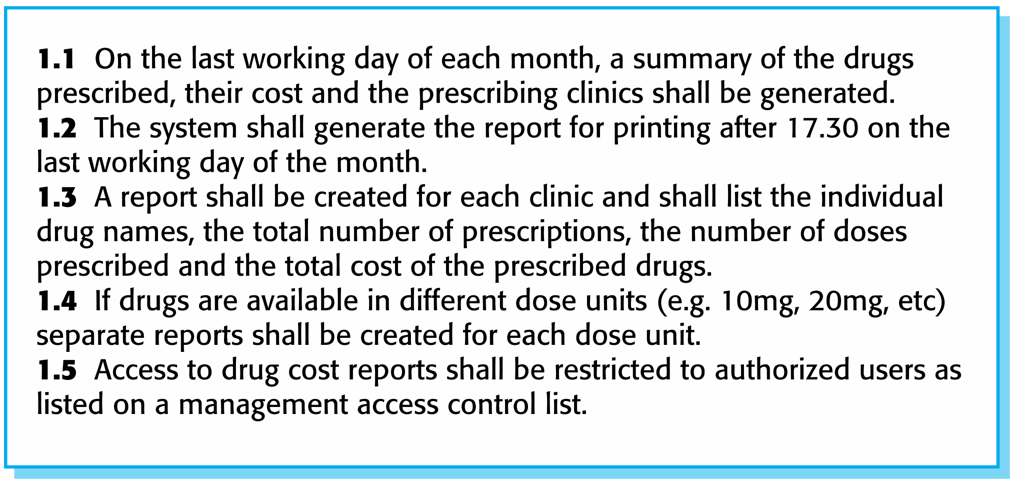
1. **User requirements**
   - ✓ Statements in natural language plus diagrams of the services that the system provides and its operational constraints.
   - ✓ Written for customers
2. **System requirements**
   - ✓ A structured document setting out detailed descriptions of the system's functions, services and operational constraints.
   - ✓ Defines what should be implemented so may be part of a contract between client and contractor.

User requirements definition



1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification



1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Software specification

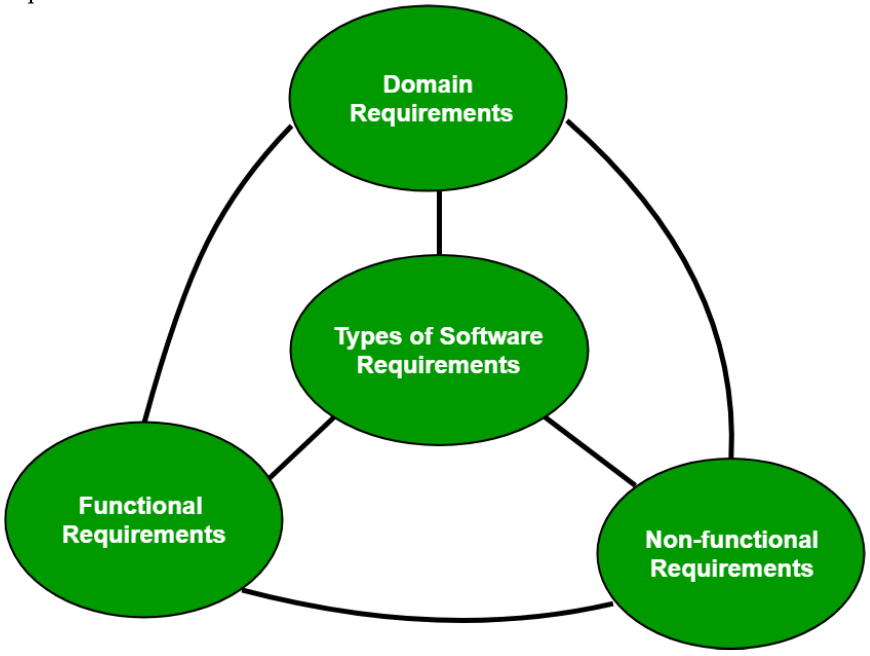
- ✓ A detailed software description which can serve as a basis for a design or implementation.
- ✓ Written for developers

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a 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 document.
3. A documented representation of a condition or capability as in 1 or 2.

A software requirement can be of 3 types:

- Functional requirements
- Non-functional requirements
- Domain requirements



Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

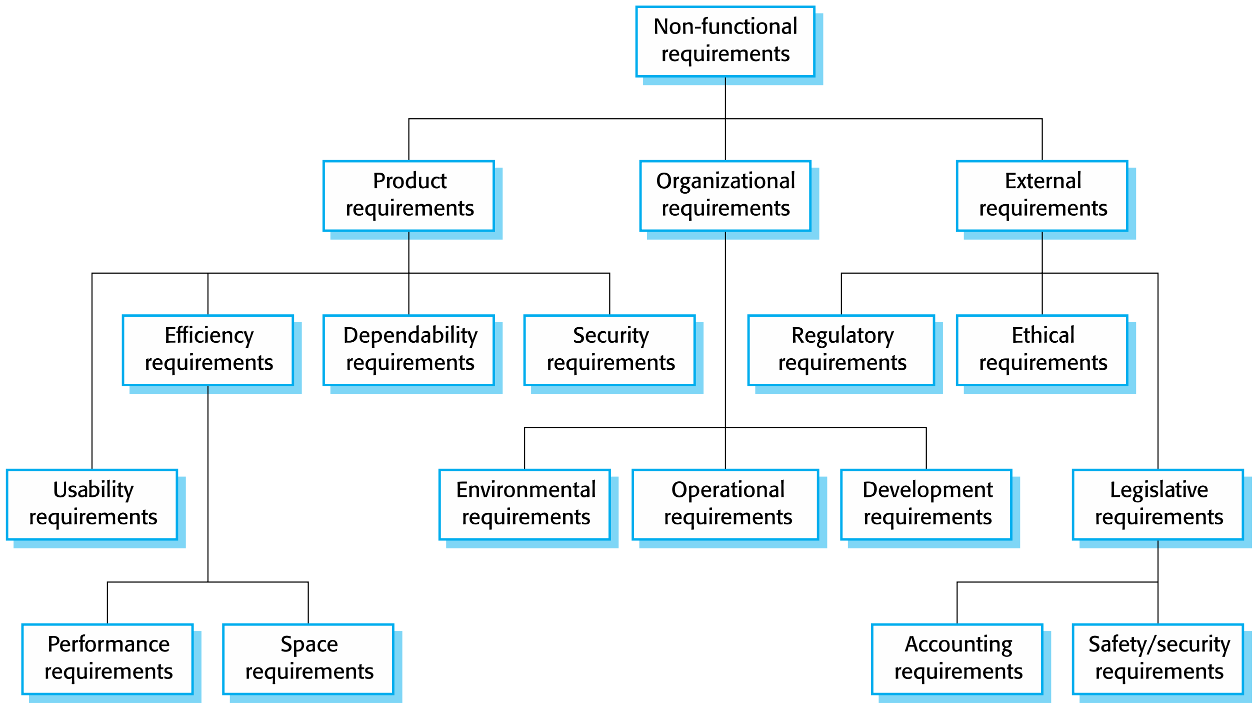
For example, in a hospital management system, a doctor should be able to retrieve the information of his patients. Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world. In order to accurately describe the functional requirements, all scenarios must be enumerated.

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. Sufficient network bandwidth may be a non-functional requirement of a system.

- ✧ Non-functional requirements are often called "quality attributes" of a system.
- ✧ Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.

- ✧ A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Types of nonfunctional requirement



- ✧ Product requirements
  - ▪ Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- ✧ Organisational requirements
  - ▪ Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.

- ✧ 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 for specifying nonfunctional requirements

Property	**Measure**
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Domain requirements: Domain requirements are the requirements which are characteristic of a particular category or domain of projects. The basic functions that a system of a specific domain must necessarily exhibit come under this category. For instance, in an academic software that maintains records of a school or college, the functionality of being able to access the list of faculty and list of students of each grade is a domain requirement. These requirements are therefore identified from that domain model and are not user specific.

User requirements

Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge User requirements are defined using natural language, tables and diagrams

Problems with natural language Ambiguity

- Readers and writers may not interpret words in the same way Over-flexibility
- The same thing may be expressed in a number of different ways Requirements amalgamation & confusion
- Several different requirements may be expressed together; functional and non-functional requirements may be mixed together Lack of modularisation
- NL structures are inadequate to structure system requirements.

Example The requirements for a CASE tool for editing software design models include the requirement for a grid to be displayed in the design window "To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel" This statement mixes up three different kinds of requirement A conceptual functional requirement stating that the editing system should provide a grid; it presents a rationale for this A non-functional requirement giving information about the grid units A non-functional user interface requirement defining how the grid is e grid is switched on and off by the user.

Alternatives to NL (natural language) specification

- Structured natural language
  - ✓ depends on defining standard forms or templates to express requirements specification
- Design description languages

  – Similar to programming languages but with additional, more abstract features
- Graphical notations

  – a graphical language, supplemented by text annotations, is used to define functional requirements (e.g. use-case diagrams)
- Mathematical/formal specifications

  – based on mathematical concepts such as finite-state machines or sets; unambiguous specifications reduce arguments between customers and contractors but most customers don't understand formal d formal speciation.

Requirements Documents

"If a company wishes to let a contract for a large software development project it must define its needs in a sufficiently abstract way that a solution is not predefined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system

The Requirements Document

- Official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- Should:
  – specify external system behaviour
  – specify implementation constraints
  – be easy to change (but changes must be managed)
  – serve as a reference tool for maintenance
  – record forethought about the life cycle of the system (i.e. predict changes) – characterise responses to unexpected events
- It is not a design document
  - it should state what the system should do rather than how it should do it

Requirements document structure

- ✓ Introduction
- ✓ Glossary
- ✓ User requirements definition
- ✓ System architecture
- ✓ System requirements specification
- ✓ System models
- ✓ System evolution
- ✓ Appendices
- ✓ Index

Requirements document users

System customers

Specify the requirements and read them back to check that they meet their needs; specify changes to the requirements

Development Managers

Use the requirements document to plan a bid for the system and to plan the system development process

Implementation Programmers

Use the requirements to understand what system is to be developed

Test programmers

Use the requirements to develop validation tests for the system

Maintenance programmers

Use the requirements to help understand the system and the relationships between its parts