

Software Requirements¹

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

Requirements can range from high-level abstract statements of services or system constraints to detailed mathematical functional specifications

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

Requirements may serve a dual function:

- As the basis of a bid for a contract
- As the basis for the contract itself

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 pre-defined. 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.”

[A.M. Davis “*Software Requirements: Objects, Functions and States*”, Englewood Cliffs, 1993]

2.1 Types of Requirement

User requirements

- Statements in natural language plus diagrams of the services that the systems provides and its operational constraints.
- Written for customers

System requirements

- A structured document setting out detailed descriptions of the system services.

¹These notes are based on those provided by Ian Sommerville on his “Software Engineering” text website

- Written as a contract between client and contractor

Software specification

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

2.1.1 Example Definition and specifications

User Requirements definition:

The software must provide a means of representing and accessing external files created by other tools

System Requirements specification:

1. The user should be provided with facilities to define the type of external files
2. Each external file type may have an associated tool which may be applied to the file
3. Each external file type may be represented as a specific icon on the user's display
4. Facilities should be provided for the icon representing an external file to be defined by the user
5. When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of external file to the file represented by the selected icon

2.2 Functional, non-functional and domain requirements

Functional requirements

- Statements of services that the system should provide, how the system should react to particular inputs and how the system should behave in particular situations

Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.

Domain requirements

- Requirements that come from the application domain of the system that reflect the characteristics of that domain
- May be functional or non-functional

2.2.1 Functional requirements

- Describe functionality or system services
- Depend on the type of software, expected users and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do; functional system requirements should describe the system services in detail

Examples

- The user shall be able to search either all of the initial set of databases or select a subset from it
- The system shall provide appropriate viewers for the user to read documents in the document store
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area

2.2.2 Non-functional requirements

- 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.*

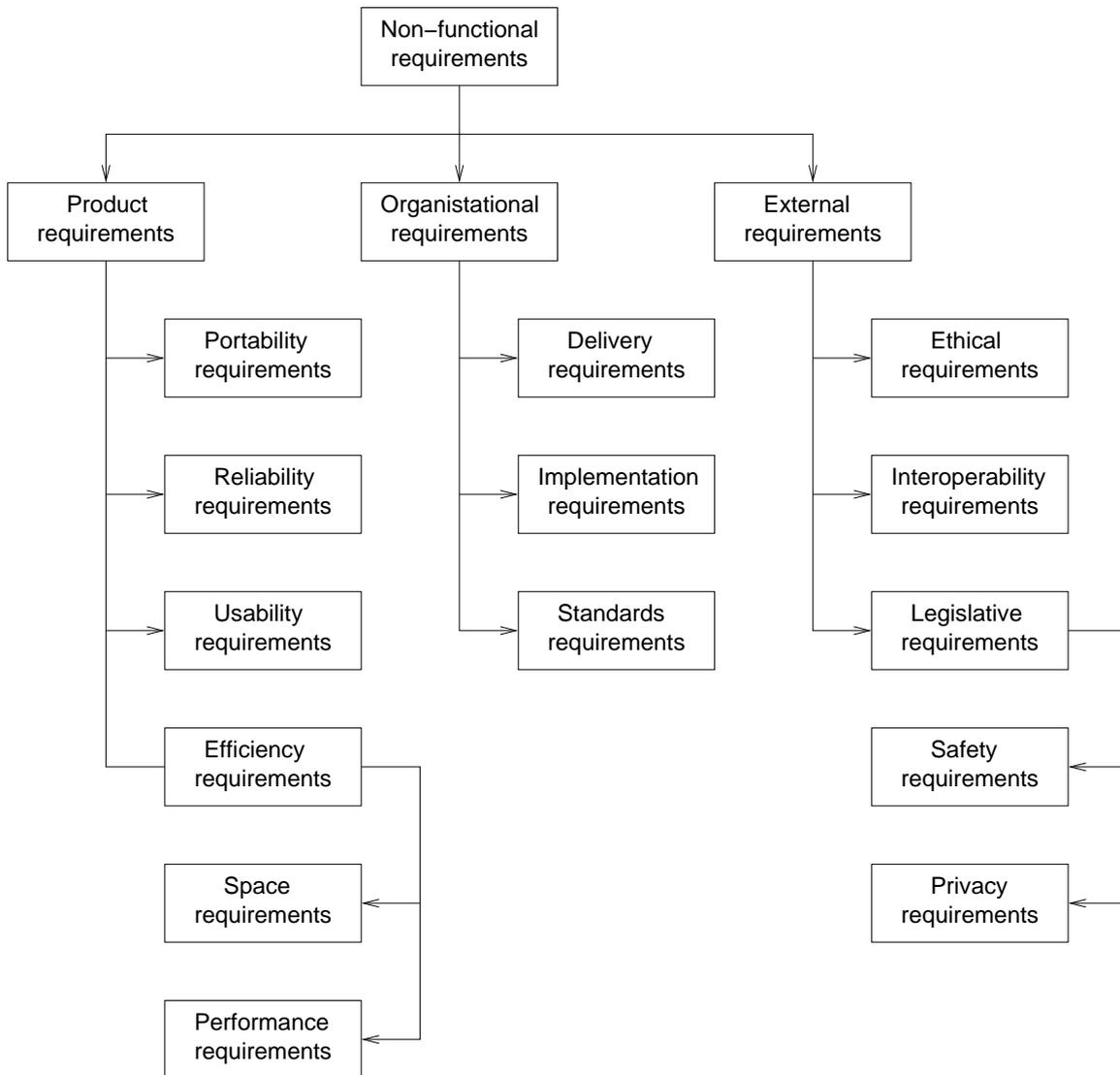


Figure 2.1: Non-functional Requirements

Metrics for non-functional requirements

See figure 2.2.

2.2.3 Domain requirements

- Describe system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements or may define specific computations
- If domain requirements are not satisfied, the system may be unworkable

Example: Library system

Property	Measure
Speed	Processed transactions/s User/Event response time Screen refresh time
Size	Kbytes Number of RAM chips
Ease of Use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	% events causing failure Time to restart after failure Probability of data corruption on failure
Portability	% target system dependent statements Number of target systems

Figure 2.2: Metrics for non-functional requirements

- Because of copyright restrictions, some received on-line documents must be deleted immediately after printing

Example: Train protection system

- Train deceleration shall be computed as

$$D_{train} = D_{control} + D_{gradient}$$

where $D_{gradient}$ is $9.81 \text{ ms}^2 * \text{compensated gradient}/\alpha$ and where the values of $9.81 \text{ ms}^2/\alpha$ are known for different types of train

Domain requirement issues

Understandability

- Requirements are expressed in the language of the application domain
 - this is often not understood by software engineers developing the system

Implicitness

- Domain specialists understand the area so well that they do not think of making the domain requirements explicit

2.3 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

2.3.1 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 switched on and off by the user

2.3.2 Revised requirement

“The editor shall provide a grid facility where a matrix of horizontal and vertical lines provides a background to the editor window. This grid shall be a passive grid where the alignment of entities is the user’s responsibility. *Rationale:* A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities ‘snap’ to grid lines can be useful, the positioning is imprecise; the user is the best person to decide where entities should be positioned.”

- Concentrates on the essential system feature
- Presents a rationale both its inclusion and for rejection of an automatic alignment feature

Separate statements are needed to cover the other parts of the original requirement, *e.g.*

- The grid can be turned on or off via an option in the control panel
- The grid can be in centimetres or inches
- The grid shall initially be in inches

2.3.3 Alternatives to NL 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 specifications

2.4 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

2.4.1 Requirements document structure

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

2.4.2 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

2.5 Summary

- Requirements set out what the system should do and define constraints on its operation and implementation
- Functional requirements set out services that the system should provide
- Non-functional requirements constrain the system being developed or the development process
- User requirements are high-level statements of what the system should do
- User requirements should be written using natural language, tables and diagrams
- System requirements are intended to communicate the functions that the system should provide
- System requirements may be written in structured natural language, a PDL or in a formal language
- A software requirements document is an agreed statement of the system requirements