Topics covered

- **Software requirements**
- Functional, non-functional and domain requirements
- Desirable properties of requirements
- The software requirements document

Software requirements

- A software requirement [1] can be defined as:
  - a software capability needed by the user to solve a problem or achieve an objective
  - a software capability that must be met or possessed by a system or system component to satisfy a contract, specification, standard, or other formally imposed documentation
- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification

Value of requirements

- Requirements deficiencies are among the most important contributors to late and over-budget software
  - in nearly every software project which fails to meet performance and cost goals, requirements inadequacies play a major and expensive role in project failure [2]
  - development of the requirements specification "in many cases seems trivial, but it is probably the part of the process which leads to more failures than any other [3]

Benefits of requirements

- Include:
  - agreement among developers, customers, and users on the job to be done and the acceptance criteria for the delivered system
  - a sound basis for resource estimation (cost, personnel quantity and skills, equipment, and time)
  - improved system usability, maintainability, and other quality attributes
  - the achievement of goals with minimum resources (less rework, fewer omissions and misunderstandings)

Requirements perspectives

- User requirements
  - statements of the services the system provides and its operational constraints. Written for customers
- System requirements
  - detailed descriptions of the system services. Written as a contract between client and contractor
- Software specification
  - a detailed software description that serve as a basis for a design or implementation. Written for developers
- We will be merging user and system requirements

Overview

- Software requirements
- **Functional, non-functional and domain requirements**
- **Desirable properties of requirements**
- The software requirements document

Functional, non-functional and domain requirements

- Functional requirements
  - statements of services 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 and that reflect characteristics of that domain. May be functional or non-functional
- No hard and fast distinction between these groups

Examples of functional requirements

- 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 database.
- Every order shall be given a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Requirements precision

- Problems arise when requirements are not precisely stated
- Ambiguous requirements may be interpreted in different ways
- Consider the term appropriate viewers. Does it mean:
  - special purpose viewer for each different document type
  - provide a text viewer that shows the contents of the document

Requirements completeness and consistency

- Ideally, requirements should be complete and consistent
- Complete
  - no important facilities left out
- Consistent
  - no conflicts or contradictions in the descriptions of different facilities
- In practice, this is impossible

Non-functional requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements, I/O device capability, system representations, etc.
- Define process requirements, e.g., a particular programming language or development method
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

Types of 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.

Non-functional requirements examples

- Product requirement
  - documents inspected using the HyperCode system must be written in ASCII character set.
- Organisational requirement
  - the system must built using only freely-available software.
- External requirement
  - all official reports generated by the billing program must conform to official court-provided forms.

Expressing non-functional requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - the general intention of the user such as ease of use
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested

Examples

- A system goal
  - the billing program should be easy to use by experienced clients and should be organised in such a way that user errors are minimized.
- A verifiable non-functional requirement
  - Clients shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by these users shall not exceed two per day.

Domain requirements

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

Example domain requirement

- It must be possible to enter billing records for a case before receiving a case number from the court

Domain requirements problems

- 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

Topics covered

- Software requirements
- Functional, non-functional and domain requirements
- Desirable properties of requirements
- **The software requirements document**

Guidelines for writing requirements

- Invent a standard format and use it for all requirements
- Use language in a consistent way, e.g., use shall for mandatory requirements, should for desirable requirements
- Use text highlighting to identify key parts of the requirement
- Avoid the use of computer jargon

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

System requirements

- More detailed specifications of user requirements
- Serve as a basis for designing the system
- System requirements may be expressed using system models discussed in Chapter 7

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements
  - The system may inter-operate with other systems that generate design requirements
  - The use of a specific design may be a domain requirement

The requirements document

- The requirements document is the official statement of what is required of the system developers
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Requirements document requirements

- Specify external system behaviour
- Specify implementation constraints
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system i.e. predict changes
- Characterise responses to unexpected events

IEEE requirements standard

- 1.Introduction
  1. Purpose - (purpose of document, audience, system benefits, objectives and goals).
  2. Scope - (software products produced, overview of what software will do/not do)
  3 .Definitions, Acronyms and Abbreviations - (as used within the document, data dictionary)
  4. References - (references to other documents used)
  5. Overview - (how the software requirement specification is organized)
- 2.General Description
  1. Product Perspective - (identify the interface between the proposed software and existing system, including a diagram of major system components
  2. Product Function - (summary of major functions)
  3. User Characteristics - (express levels of expertise/training required to use system)
  4. General Considerations - (limitations, including hardware, software language)
  5. Assumptions and Dependencies - (factors that will effect the requirements i.e. OS type)
- 3.Specific requirements
  1. User requirements definition - defined as UML use cases and activity diagrams
  2. System models - the domain model represented as UML class diagrams, state diagrams and interaction diagrams

4.Appendices - any other important material

Key points

- A software requirements document is an agreed statement of the system requirements
- Requirements set out what the system should do and define constraints on its operation and implementation
- Functional requirements set out services the system should provide
- Non-functional requirements constrain the system being developed or the development process

References

1. Naur, P., and B. Randell, eds., Software Engineering: Report on a Conference Sponsored by the NATO Science Commission, Garmisch, Germany, 7–11 October 1968. Scientific Affairs Division, NATO, Brussels, January 1969.

2.Alford, M. W., and J. T. Lawson, "Software Requirements Engineering Methodology (Development)." RADC-TR-79-168, U.S. Air Force Rome Air Development Center, Griffiss AFB, NY, June 1979 (DDC-AD-A073132).

3.Schwartz, J.I., "Construction of Software, Problems and Practicalities," in Practical Strategies for Developing Large Software Systems, E. Horowitz, ed., Addison-Wesley, Reading, MA, 1975.

- Slides based on and adapted from Sommerville, chap. 5 and 6.