Software Engineering

Software
Requirements

Objectives

- To introduce the concepts of user and system requirements
- To describe functional and non-functional requirements
- To explain how software requirements may be organised in a requirements document
Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.
- Requirements may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
- Both these statements may be called requirements.

Types of requirements

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- 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.
**Functional and non-functional 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.

**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 but 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.
### Requirements imprecision, completeness and consistency

- Problems arise when requirements are not precisely stated.
- **Ambiguous requirements** may be interpreted in different ways by developers and users.
  - Consider the term ‘appropriate viewers’
    - User intention - special purpose viewer for each different document type;
    - Developer interpretation - provide a text viewer that shows the contents of the document.

- In principle, **requirements should be both complete and consistent**.
  - **Complete**
    - They should include descriptions of all facilities required.
  - **Consistent**
    - There should be no conflicts or contradictions in the descriptions of the system facilities.
  - In practice, it is impossible to produce a complete and consistent requirements document.

### Non-functional requirements

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

- **Examples**
  - **Product requirement**
    - 8.1 The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
  - **Organisational requirement**
    - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
  - **External requirement**
    - 7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.
Non-functional requirement types

- **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.

Goals and requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- **Goal**
  - A general intention of the user such as ease of use.
- **Verifiable non-functional requirement**
  - A statement using some measure that can be objectively tested.
- **Goals are helpful to developers as they convey the intentions of the system users.**
- **Examples**
  - A system goal
    - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
  - A verifiable non-functional requirement
    - Experienced controllers 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 experienced users shall not exceed two per day.
Requirements measures

<table>
<thead>
<tr>
<th>Property</th>
<th>Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Speed</strong></td>
<td>Processed transactions/second</td>
</tr>
<tr>
<td></td>
<td>User/Event response time</td>
</tr>
<tr>
<td></td>
<td>Screen refresh time</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>M Bytes</td>
</tr>
<tr>
<td></td>
<td>Number of ROM chips</td>
</tr>
<tr>
<td><strong>Ease of use</strong></td>
<td>Training time</td>
</tr>
<tr>
<td></td>
<td>Number of help frames</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Mean time to failure</td>
</tr>
<tr>
<td></td>
<td>Probability of unavailability</td>
</tr>
<tr>
<td></td>
<td>Rate of failure occurrence</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td><strong>Robustness</strong></td>
<td>Time to restart after failure</td>
</tr>
<tr>
<td></td>
<td>Percentage of events causing failure</td>
</tr>
<tr>
<td></td>
<td>Probability of data corruption on failure</td>
</tr>
<tr>
<td><strong>Portability</strong></td>
<td>Percentage of target dependent statements</td>
</tr>
<tr>
<td></td>
<td>Number of target systems</td>
</tr>
</tbody>
</table>

Requirements interaction

- **Conflicts** between different non-functional requirements are common in complex systems.
- **Spacecraft system**
  - To minimise weight, the number of separate chips in the system should be minimised.
  - To minimise power consumption, lower power chips should be used.
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?
**Domain requirements**

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

**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.

**User requirements**

- Should describe functional and non-functional requirements in such a way that they are **understandable by system users** who don’t have detailed technical knowledge.
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.
- **Problems with natural language**
  - Lack of clarity
    - Precision is difficult without making the document difficult to read.
  - Requirements confusion
    - Functional and non-functional requirements tend to be mixed-up.
  - Requirements amalgamation
    - Several different requirements may be expressed together.
- **Writing guidelines**
  - Invent a standard format and use it for all requirements.
  - Use language in a consistent way. 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.
System requirements

- More detailed specifications of system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system.
- They may be incorporated into the system contract.
- System requirements may be defined or illustrated using system models.

- 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.

Problems with natural language specification

- Ambiguity
  - The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.
- Over-flexibility
  - The same thing may be said in a number of different ways in the specification.
- Lack of modularisation
  - NL structures are inadequate to structure system requirements.

Alternatives

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured natural language</td>
<td>This approach depends on defining standard forms or templates to express the requirements specification.</td>
</tr>
<tr>
<td>Design description languages</td>
<td>This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.</td>
</tr>
<tr>
<td>Graphical notations</td>
<td>A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used.</td>
</tr>
<tr>
<td>Mathematical specifications</td>
<td>These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.</td>
</tr>
</tbody>
</table>
Representing requirements

- **Structured language specifications**
  - Limited by a predefined template for requirements.
  - All requirements are written in a standard way.
  - The terminology used in the description may be limited.
  - The advantage is that the most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification.

- **Form-based specifications**
  - Definition of the function or entity.
  - Description of inputs and where they come from.
  - Description of outputs and where they go to.
  - Indication of other entities required.
  - Pre and post conditions (if appropriate).
  - The side effects (if any) of the function.

- **Tabular specification**
  - Used to supplement natural language.
  - Particularly useful when you have to define a number of possible alternative courses of action.

- **Graphical models**
  - Graphical models are most useful when you need to show how state changes or where you need to describe a sequence of actions.

---

Sequence diagrams

- These show the sequence of events that take place during some user interaction with a system.
- You read them from top to bottom to see the order of the actions that take place.
- **Cash withdrawal from an ATM**
  - Validate card;
  - Handle request;
  - Complete transaction.

Stan Kurkovsky
**Interface specification**

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
- Three types of interface may have to be defined
  - Procedural interfaces;
  - Data structures that are exchanged;
  - Data representations.
- Formal notations are an effective technique for interface specification.

---

**The requirements document**

- The requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system requirements.
- It is NOT a design document. As far as possible, it should set WHAT the system should do rather than HOW it should do it.
### Requirements document structure

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

### Summary

- 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.
- 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.
- A software requirements document is an agreed statement of the system requirements.
- The IEEE standard is a useful starting point for defining more detailed specific requirements standards.