You Are Here.

Introduction

Requirements Engineering

• Vocabulary: Requirements (Analysis)
  • Usages of Requirements Specifications
  • Requirements Specification
  • Requirements Analysis
  • Desired Properties
  • Kinds of Requirements
  • Analysis Techniques
  • Documents
  • Dictionary
  • Specification
  • Requirements Specification Languages
  • Natural Language

Recall: Structure of Topic Areas

Example: Requirements Engineering

Vocabulary e.g. consistent, complete, tacit, etc.

Techniques informal semi-formal formal

Content
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later. (F. P. Brooks, 1995)

Introduction
Vocabulary: Requirements (Analysis)
Usages of Requirements Specifications
Requirements Specification
Requirements Analysis
Desired Properties
Kinds of Requirements
Analysis Techniques
Documents
Dictionary
Specification
Requirements Specification Languages
Natural Language

Requirements Analysis...
... in the sense of "finding out what the exact requirements are". "Analysing an existing requirements/feature specification " later.

In the following we shall discuss:
(i) desired properties of requirements specifications,
(ii) kinds of requirements hard and soft,
open and tacit,
functional and non-functional.
(iii) (a selection of) analysis techniques
(iv) documents of the requirements analysis:
• dictionary,
• requirements specification ('Lastenheft'),
• feature specification ('Pflichtenheft').

Note: In the following (unless otherwise noted), we discuss the feature specification, i.e. the document on which the software development is based. To maximise confusion, we may occasionally (inconsistently) call it requirements specification or just specification—should be clear from context. ...

Recall: one and the same content can serve both purposes; only the title defines the purpose then.
A requirements specification should be:

- **correct** — it correctly represents the wishes/needs of the customer,
- **complete** — all requirements (existing in somebody's head, or a document, or ...) should be present,
- **relevant** — things which are not relevant to the project should not be constrained,
- **consistent**, **free of contradictions** — each requirement is compatible with all other requirements; otherwise the requirements are not realisable,
- **neutral**, **abstract** — a requirements specification does not constrain the realisation more than necessary,
- **traceable**, **comprehensible** — the sources of requirements are documented, requirements are uniquely identifiable,
- **testable**, **objective** — the final product can objectively be checked for satisfying a requirement.

Correctness and completeness are defined relative to something which is usually only in the customer's head.

→ It is difficult to be sure of correctness and completeness.

→ "Dear customer, please tell me what is in your head!" is in almost all cases not a solution! It’s not unusual that even the customer does not precisely know...

The representation and form of a requirements specification should be:

- **easily understandable**, **not unnecessarily complicated** — all affected people should be able to understand the requirements specification,
- **precise** — the requirements specification should not introduce new unclarities or rooms for interpretation (→ testable, objective),
- **easily maintainable** — creating and maintaining the requirements specification should be easy and should not need unnecessary effort,
- **easily usable** — storage of and access to the requirements specification should not need significant effort.

Note: Once again, it’s about compromises.

→ A very precise objective requirements specification may not be easily understandable by every affected person. → provide redundant explanations.

→ It is not trivial to have both, low maintenance effort and low access effort. → value low access effort higher, a requirements specification document is much more often read than changed or written (and most changes require reading beforehand).

Pitfall: Vagueness vs. Abstraction

Consider the following examples:

- **Vague** (not precise):
  
  "the list of participants should be sorted conveniently"

- **Precise**, **abstract**:
  
  "the list of participants should be sorted by immatriculation number, lowest number first"

- **Precise**, **non-abstract**:
  
  "the list of participants should be sorted by
  
  public static <T> void Collections::sort( List<T> list, Comparator c );
  
  where T is the type of participant records,
  
  c compares immatriculation number numerically."

A requirements specification should always be as **precise** as possible (→ testable, objective). It need not denote exactly one solution; precisely characterising acceptable solutions is often more appropriate.

→ Being too specific may limit the design decisions of the developers, which may cause unnecessary costs.

→ Idealised views advocate a strict separation between requirements ("what is to be done?") and design ("how are things to be done?").
(a Selection of Analysis Techniques)

Requirements Analysis Techniques

• Analysis of existing data and documents

Requirements Specification

• Situation, situation, consequences

Multiple perspectives: non-functional requirements

Soft requirements for programming language, coding conventions, process model requirements, portability...

Functional requirements: non-functional

Clearly intended requirements

• Intentionally left open to be decided by developer.

• We often cannot provide this clarity;

• Clearly right/wrong.

• Because it requires something for the function

• Every constraint

• Examples:

- A car entertainment system which produces "noise" (due to limited bus bandwidth or CPU power)

- Software "traffic lights controller":
  - And no more inputs,
  - o
  - n: button pushed again

- Software "compute shipping costs":
  - o
  - S

- Cashing a cheque over

- Customer presses button, i.e. we want a clear right/wrong.

- Examples:

  - If a vending machine dispenses the selected item within 1 s, it’s clearly fine; if it takes 5 min., it’s clearly wrong — where’s the boundary?

  - A webshop may be required to display important web-shop items on the right hand side because the main users are socialized with right-to-left reading direction,

  - Buttons and screen of a mobile phone should be on the same side,

  - Customer not aware of something

  - Semi-tacit requirements

  - Because it requires something for the function

  - Every constraint

  - Customer knows domain new to domain

  - Examples

  - A vending machine which may only be allowed use a certain amount of bus capacity.

  - "We know what is" and we know what is

  - Customer doesn’t care

  - Distinguish between hard/soft

  - Requirements which maps

  - S

  - N

  - Must result in a new balance decreased by

  - Shipping parameters

  - Software "compute shipping costs":

  - Examples:

    - Shipping rates, etc. may be subject to functional requirements.

    - Energy consumption, timing, etc. may be subject to functional requirements.

    - We often cannot provide this clarity;

    - Clearly right/wrong.

Kinds of Requirements

• Functional and Non-Functional

• Hard and Soft Requirements

Kinds of Requirements: Functional and Non-Functional

Kinds of Requirements: Hard and Soft Requirements

Kinds of Requirements: Open and Semi-tacit

Experiments

• Prototyping

Vocabulary: Requirements (Analysis)

• Questioning with analysis techniques

• Open structured questions

Interview Analysis Techniques

• Observation

Participative development

Requirements Specification Languages

• Modelling Documents

• Dictionary

• Participative development
During a project on designing a highly reliable, EN-54-25 conforming wireless communication protocol, we had to learn that the relevant components of a fire alarm system are terminals and participants which receive messages from different assigned terminals.

Each terminal participant consists of exactly one wireless communication module and devices which provide sensor and/or signalling functionality.

A central unit is a part of the fire alarm system that communicates with the terminals and participants. A repeater is a device that repeats signals to extend the range of communication.

Requirements analysis should be based on a requirements specification. The resulting "raw material" is the basis of a formal specification. Users can be interviewed by a team of 2 analysts, ca. 90 min. During analysis, talk to customers, domain experts, or a core team.

Note: The customer decides, i.e. to specify requirements, but the customer chooses. (And the choice is documented.)

When "left alone" with a preliminary requirements specification, users can be interviewed by a team of 2 analysts, ca. 90 min. Analysis benefits from strong experience of analysis (3 to 4 people), domain knowledge, and expertise of the core team.

The customer wishes, and for this reason, the customer benefits from having technical background to anticipate exceptions, difficulties, contradictions, or radical change.

Good questions: How are things done today? What should be improved? Customers can not be assumed to be trained in stating/communicating requirements.

The "crystals" need to be sorted and "crystallize" to form a "crystal clear" set of requirements.

The requirements are a task of the analyst. There are technical difficulties, exceptions, and requirements that are relevant to the project and of open questions. The requirements are a test of one's own understanding and whether one knows what is wanted.

The project and requirements are a test of the analyst's knowledge, skills, and experience.
Requirements Specification

A document that specifies,

• in a complete, precise, verifiable manner,

• the requirements, design, behavior, or other characteristics of a system or component,

• and, often, the procedures for determining whether these provisions have been satisfied.

IEEE 610.12 (1990)

Software requirements specification (SRS)

— Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.

IEEE 610.12 (1990)

Structure of a Requirements Document: Example

1 INTRODUCTION

1.1 Purpose

1.2 Acronyms and Definitions

1.3 References

1.4 User Characteristics

2 FUNCTIONAL REQUIREMENTS

2.1 Function Set 1

2.2 etc.

3 REQUIREMENTS TO EXTERNAL INTERFACES

3.1 User Interfaces

3.2 Interfaces to Hardware

3.3 Interfaces to Software Products / Software / Firmware

3.4 Communication Interfaces

4 REQUIREMENTS REGARDING TECHNICAL DATA

4.1 Volume Requirements

4.2 Performance

4.3 etc.

5 GENERAL CONSTRAINTS AND REQUIREMENTS

5.1 Standards and Regulations

5.2 Strategic Constraints

5.3 Hardware

5.4 Software

5.5 Compatibility

5.6 Cost Constraints

5.7 Time Constraints

5.8 etc.

6 PRODUCT QUALITY REQUIREMENTS

6.1 Availability, Reliability, Robustness

6.2 Security

6.3 Maintainability

6.4 Portability

6.5 etc.

7 FURTHER REQUIREMENTS

7.1 System Operation

7.2 Customisation

7.3 Requirements of Internal Users

(Chen and Biswas, 2013) based on (IEEE, 1998)
Do not underestimate the value of a good natural language patterns.

Natural language is inherently imprecise, counter-measures:

- Distinguish easily understandable, precise, easily maintainable, easily usable requirements.
- It is the task of the analyst to make these requirements explicit.
- Identify simple, direct, clear statements.
- Recognise and refine assumptions that are not explained further.
- Check full verbs, active voice, defined verbs, nominalisations, quantifiers, conditions.
- Discover unclear substantives, nominalisations, quantifiers, conditions.
- Express processes by active voice.
- Identify implicit assumptions (here: there seems to be a firewall).
- Name the actors, indicate whether the user or the system does something.
- If the specification says that something is "possible", "impossible", or "may", "should", "must" happen, clarify who is enforcing or prohibiting the behaviour.
- Note: vague vs. abstract.

**References**