Lecture 6: Requirements Engineering

2016-05-12

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

You Are Here.

Introduction

- Requirements Specification
- Desired Properties
- Kinds of Requirements
- Analysis Techniques
- Documents
- Dictionary, Specification
- Specification Languages
- Natural Language
- Working Definition: Software
- Decision Tables
- Syntax, Semantics
- Consistency, Completeness, ...
- Scenarios
- User Stories, Use Cases
- Live Sequence Charts
- Syntax, Semantics
- Discussion

VL 6 ...
VL 7 ...
VL 8 ...
VL 9 ...

Recall: Structure of Topic Areas

Example: Requirements Engineering

Vocabulary: Requirements (Analysis)

Usages of Requirements Specifications

Requirements Specification

Desired Properties

Kinds of Requirements

Analysis Techniques

Documents

Dictionary

Specification

Specification Languages

Natural Language

Introduction
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)

Requirements Specifications

Requirements Analysis...

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

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.

Requirements on Requirements Specifications

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. ...!
Kinds of Requirements: Hard and Soft Requirements

**Hard Requirements**
- Must be fulfilled precisely and completely by the program.
- Examples:
  - A requirements specification should always be as precise as possible (form, representation).
  - The requirements specification should not introduce new unclarities or rooms for interpretation.
  - All affected people should be able to understand the requirements specification.
  - The requirements specification should be easy and should not need unnecessary effort.
- Note: Every constraint requires something for the function to be present.
- Examples:
  - A car entertainment system which produces "noise" (due to limited bus bandwidth or CPU power)
  - A vending machine dispenses the selected item within 1 s

**Soft Requirements**
- Are not as hard to define as hard requirements and require compromises.
- Examples:
  - A requirements specification should be as clear as possible.
  - It need not denote precisely all inputs, outputs, sequences of inputs, sequences of outputs, etc. may be subject to functional requirements.
  - The border between hard/soft is difficult to draw.
  - A requirements specification should be easily usable.
  - Precise, objective.
  - Precise may limit the design decisions of the developers, which may cause unnecessary costs.

- Examples:
  - "The list of participants should be sorted by immatriculation number, lowest number first"
  - "the list of participants should be sorted conveniently"
  - "the list of participants should be sorted by immatriculation number, numerically"
Good questions: How are things done today? What should be improved?

improvement, but (radical) change

Many customers do not want

Okay. Now what exactly does "morning" mean?

A:

The requirements.

elicit, i.e. to →

Then the first client will knock on the window.

C:

And if Mr. M is not available, too?

A:

More questions.

Then Mr. M opens the door.

C:

And if you are ill or on vacation?

A:

Note (formal) specification to communicate •

The customer decides: Y es, as I told you.

Customer

dialogue. . .( differentiate, but the customer chooses. (And the choice is documented.)

Then it also remains closed of course.

C:

"test" own understanding by •

making (different options to choose

proposals •

The customer decides: Y es.

Of course.

C:

Every morning?

A:

Customer/Client

It is the •

The "raw material" is basis of a •

Requirements Documents

requirements specification

overstrained •

As Analyst •

experts •

users

between •

contradictions •

limitations, •

corner-cases, •

technical difficulties, •

look out for contradictions, •

what is wanted, •

ask •

than what is not wanted, •

what is wanted, •

ask •

what is not wanted, •

how to communicate •

Technical difficulties

Users can be interviewed by a team of 2 analysts, ca. 90 min.

Interview

Requirements Analysis Techniques

•

Interview

•

Requirements Elicitation

•

Analysis of existing data and documents •

Interview

•

Observation •

Experiments

•

Modelling

•

Questioning with •

•

Experiments

•

Analysis of existing data and documents •

•

Modelling

•

Questioning with •

•

Experiments

•

Analysis of existing data and documents •

•

Modelling

•

Questioning with •

•

Experiments
The content and qualities of a good software requirements specification (SRS) are de-

scribed and several sample SRS outlines are presented. This recommended practice is aimed at specifying requirements of ... of in-house and commercial software products. Guidelines for compliance with IEEE/EIA12207.1-1997 are also provided.
Do not underestimate the value of a good dictionary. However, natural language patterns naturally need certain assumptions. Identify implicit assumptions (here: there seems to be a firewall).

If the specification says that something is "possible," "impossible," or "may," "should," "must" happen, clarify who is enforcing or prohibiting the behaviour.

Is the substantive used as a generic term or does it denote something specific? Is "user" generic or is a member of a specific classes meant?

Are sentences with "never," "always," "each," "any," "all" really universally valid? Are "all" really all or are there exceptions.

Conditions of the form "if-else" need descriptions of the if- and the then-case.

In "the component raises an error," ask whom the message is addressed to.

Nouns like "registration" often hide complex processes that need more detailed descriptions; the verb "register" raises appropriate questions: who, where, for what?

Not "is," "has" but "reads," "creates." Full verbs require information about who, where, when.

Discover defined verbs and, sometimes, verification protocols, used to develop, analyze, and document hard-ware or software requirements.

Express processes by nominalisations, in particular an object "offers" description of a function offered by the system to somebody, "is able if", usage of a function offered by a third party, under certain conditions.

Requirements Documents

Requirements Specification

- Specification language with special constructs
- A language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other assumptions.

- A specification language with special constructs
- A language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other assumptions.

- Requirements Specification Language

- Requirements Specification Language

- Other Pattern Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,
- Hard / soft,
- Easily understandable, precise, easily maintainable, easily usable

- Requirements Specification

- Requirements Specification

- Other Patter Example: RFC 2119

- Example Natural Language Patterns

- Requirements Representations

- Functional / non-functional,


