15 quality goals for requirements

- Justified
- Correct
- Complete
- Consistent
- Unambiguous
- Feasible
- Abstract
- Traceable

- Delimited
- Interfaced
- Readable
- Modifiable
- Verifiable
- Prioritized*
- Endorsed

Marked attributes are part of IEEE 830, see below
* "Ranked for importance and/or stability"

Verifiable requirements

Adapted from: IEEE

Non-verifiable :

- The system shall work satisfactorily
- The interface shall be user-friendly
- The system shall respond in real time

Verifiable:

- The output shall in all cases be produced within 30 seconds of the corresponding input event. It shall be produced within 10 seconds for at least 80% of input events.
- Professional train drivers will reach level 1 of proficiency (*defined in requirements*) in two days of training.

Practical advice

Favor precise, falsifiable language over pleasant generalities

Complete requirements

Complete with respect to what?

Definition from IEEE standard (see next) :

An SRS is complete if, and only if, it includes the following elements:

- *All significant requirements, whether relating to functionality, performance, design constraints, attributes, or external interfaces. In particular any external requirements imposed by a system specification should be acknowledged and treated.*
- *Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input values.*
- *Full labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure.*

IEEE 830-1998

"IEEE Recommended Practice for Software Requirements Specifications"

Approved 25 June 1998 (revision of earlier standard)

Descriptions of the content and the qualities of a good software requirements specification (SRS).

Goal: "The SRS should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, traceable."

IEEE 830-1998

"IEEE Recommended Practice for Software Requirements Specifications"

Approved 25 June 1998 (revision of earlier standard)

Descriptions of the content and the qualities of a good software requirements specification (SRS).

Goal: "The SRS should be correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable, traceable."

15 quality goals for requirements

- Justified
- Correct
- Complete
- Consistent
- Unambiguous
- Feasible
- Abstract

- Traceable
- Delimited
- Interfaced
- Readable
- Modifiable
- Testable
- Prioritized
- Endorsed

IEEE Standard: definitions

Contract:

A legally binding document agreed upon by the customer and supplier. This includes the technical and organizational requirements, cost, and schedule for a product. A contract may also contain informal but useful information such as the commitments or expectations of the parties involved.

Customer:

The person, or persons, who pay for the product and usually (but not necessarily) decide the requirements. In the context of this recommended practice the customer and the supplier may be members of the same organization.

Supplier:

The person, or persons, who produce a product for a customer. In the context of this recommended practice, the customer and the supplier may be members of the same organization.

User:

The person, or persons, who operate or interact directly with the product. The user(s) and the customer(s) are often not the same person(s).

IEEE Standard

Basic issues to be addressed by an SRS:

- Functionality
- External interfaces
- Performance
- Attributes
- Design constraints imposed on an implementation

IEEE Standard

Recommended document structure:

1. Introduction

- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, acronyms, and abbreviations ← Glossary!
- 1.4 References
- 1.5 Overview

2. Overall description

- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 Constraints
- 2.5 Assumptions and dependencies

3. Specific requirements

Appendixes

Index

Practical advice

Use the recommended IEEE structure

Practical advice

Write a glossary

Recommended document structure

1. Introduction

- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, acronyms, and abbreviations
- 1.4 References
- 1.5 Overview

2. Overall description

- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 Constraints
- 2.5 Assumptions and dependencies

3. Specific requirements

Appendixes

Index

Example section: scope

- Identify software product to be produced by name (e.g., Host DBMS, Report Generator, etc.)
- Explain what the product will and will not do
- Describe application of the software: goals and benefits
- Establish relation with higher-level system requirements if any

Example section: product perspective

Describe relation with other products if any.

Examples:

- System interfaces
- User interfaces
- Hardware interfaces
- Software interfaces
- Communications interfaces
- Memory
- Operations
- Site adaptation requirements

Example section: constraints

Describe any properties that will limit the developers' options

Examples:

- Regulatory policies
- Hardware limitations (e.g., signal timing requirements)
- Interfaces to other applications
- Parallel operation
- Audit functions
- Control functions
- Higher-order language requirements
- Reliability requirements
- Criticality of the application
- Safety and security considerations

Recommended document structure

1. Introduction

- 1.1 Purpose
- 1.2 Scope
- 1.3 Definitions, acronyms, and abbreviations
- 1.4 References
- 1.5 Overview

2. Overall description

- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 Constraints
- 2.5 Assumptions and dependencies

3. Specific requirements

Appendixes

Index

Specific requirements (section 3)

This section brings requirements to a level of detail making them usable by designers and testers.

Examples:

- Details on external interfaces
- Precise specification of each function
- Responses to abnormal situations
- Detailed performance requirements
- Database requirements
- Design constraints
- Specific attributes such as reliability, availability, security, portability

Possible section 3 structure

3. Specific requirements

- 3.1 External interfaces
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communication interfaces
- 3.2 Functional requirements
  - …
- 3.3 Performance requirements
  - …
- 3.4 Design constraints
  - …
- 3.5 Quality requirements
  - …
- 3.6 Other requirements
  - …

Consider a small library database with the following transactions:

1. Check out a copy of a book. Return a copy of a book.
2. Add a copy of a book to the library. Remove a copy of a book from the library.
3. Get the list of books by a particular author or in a particular subject area.
4. Find out the list of books currently checked out by a particular borrower.
5. Find out what borrower last checked out a particular copy of a book.

There are two types of users: staff users and ordinary borrowers.

Transactions 1, 2, 4, and 5 are restricted to staff users, except that ordinary borrowers can perform transaction 4 to find out the list of books currently borrowed by themselves. The database must also satisfy the following constraints:

- All copies in the library must be available for checkout or be checked out.
- No copy of the book may be both available and checked out at the same time.
- A borrower may not have more than a predefined number of books checked out at one time.