System context

- Subject facet
- Usage facet
- IT system facet
- Development facet

Core activities

- Documentation
- Elicitation
- Negotiation

Requirements artefacts

- Goals
- Scenarios
- Solution oriented requirements
Document important information elicited or developed when performing a core the RE activity:
- i.e., documentation, elicitation, negotiation, validation and/or management
“Requirements Lifecycle”

Source: Adapted from Pohl, CAISE 1993
Requirements Specification

• What is Requirements Specification?

• Criteria for good requirements
Requirements Specification

→ How do we communicate the Requirements to others?
   - It is common practice to capture them in a specification
     - But an specification does not need to be a single paper document...

→ Purpose
   - Communication
     - explains the application domain and the system to be developed
   - Contractual
     - May be legally binding!
     - Expresses agreement and a commitment
   - Baseline for evaluating the software
     - supports testing, V&V
     - “enough information to verify whether delivered system meets requirements”
   - Baseline for change control

→ Audience
   - Customers & Users
     - interested in system requirements…
     - …but not detailed software requirements
   - Systems (Requirements) Analysts
     - Write other specifications that inter-relate
   - Developers, Programmers
     - Have to implement the requirements
   - Testers
     - Have to check that the requirements have been met
   - Project Managers
     - Have to measure and control the project
## Appropriate Specification

**A) Tiny project, 1 programmer, 2 months work**

programmer talks to customer, then writes up a 2-page memo

<table>
<thead>
<tr>
<th>Purpose of spec?</th>
<th>Project A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crystalizes programmer’s understanding; feedback to customer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Management view?</th>
<th>Spec is irrelevant; have already allocated resources</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Readers?</th>
<th>Primary: Spec author; Secondary: Customer</th>
</tr>
</thead>
</table>
Appropriate Specification

B) Large project, 50 programmers, 2 years work
   team of analysts model the requirements, then document them in a 500-page document

<table>
<thead>
<tr>
<th>Purpose of spec?</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Build-to document; must contain enough detail for all the programmers</td>
</tr>
<tr>
<td>Management view?</td>
<td>Will use the spec to estimate resource needs and plan the development</td>
</tr>
<tr>
<td>Readers?</td>
<td><strong>Primary</strong>: programmers, testers, managers; <strong>Secondary</strong>: customers</td>
</tr>
</tbody>
</table>
Procurement

A requirements specification may be written by...

...the procurer:
- specification is really a call for proposals
- Must be general enough to yield a good selection of bids...
- …and specific enough to exclude unreasonable bids

...the bidders:
- specification is a proposal to implement a system to meet the CfP
- must be specific enough to demonstrate feasibility and technical competence
- …and general enough to avoid over-commitment

...the selected developer:
- reflects the developer’s understanding of the customer’s needs
- forms the basis for evaluation of contractual performance

...or by an independent RE contractor!

Choice over what point to compete the contract

Early (conceptual stage)
- can only evaluate bids on apparent competence & ability

Late (detailed specification stage)
- more work for procurer; appropriate RE expertise may not be available in-house
Specifying Contents

The specification should address:

- **Functionality**
  - What is the software supposed to do?

- **External interfaces**
  - How does the software interact with people, the system's hardware, other hardware, and other software?
  - What assumptions can be made about these external entities?

- **Performance**
  - What is the speed, availability, response time, recovery time of various software functions, and so on?

- **Attributes**
  - What are the portability, correctness, maintainability, security, and other considerations?

- **Design constraints imposed on an implementation**
  - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) and so on?
Specification should not include…

→ **Project development plans**
  - E.g. cost, staffing, schedules, methods, tools, etc
    - Lifetime of SRS is until the software is made obsolete
    - Lifetime of development plans is much shorter

→ **Product assurance plans**
  - V&V, test, QA, etc
  - Different audiences
  - Different lifetimes

→ **Designs**
  - Requirements and designs have different audiences
  - Analysis and design are different areas of expertise
    - I.e. requirements analysts shouldn’t do design!
# IEEE Standard for Requirements Specification

## 1 Introduction
- **Purpose**
- **Scope**
- Definitions, acronyms, abbreviations
- Reference documents
- Overview

## 2 Overall Description
- **Product perspective**
- **Product functions**
- **User characteristics**
- **Constraints**
- Assumptions and Dependencies

## 3 Specific Requirements

### Appendices

### Index

---

- Identifies the product, & application domain
- Describes contents and structure of the remainder of the SRS
- Describes all external interfaces: system, user, hardware, software; also operations and site adaptation, and hardware constraints
- Summary of major functions, e.g. use cases
- Anything that will limit the developer’s options (e.g. regulations, reliability, criticality, hardware limitations, parallelism, etc)
- All the requirements go in here (i.e. this is the body of the document). IEEE STD provides 8 different templates for this section
3.1 External Interface Requirements
   3.1.1 User Interfaces
   3.1.2 Hardware Interfaces
   3.1.3 Software Interfaces
   3.1.4 Communication Interfaces

3.2 Functional Requirements
   this section organised by mode, user class, feature, etc. For example:
   3.2.1 Mode 1
      3.2.1.1 Functional Requirement 1.1
      ...
   3.2.2 Mode 2
      3.2.1.1 Functional Requirement 1.1
      ...
   3.2.2 Mode n
      ...

3.3 Performance Requirements
   Remember to state this in measurable terms!

3.4 Design Constraints
   3.4.1 Standards compliance
   3.4.2 Hardware limitations etc.

3.5 Software System Attributes
   3.5.1 Reliability
   3.5.2 Availability
   3.5.3 Security
   3.5.4 Maintainability
   3.5.5 Portability

3.6 Other Requirements
Organizing the Requirements

→ Example Structures - organize by...

- **External stimulus or external situation**
  - e.g., for an aircraft landing system, each different type of landing situation: wind gusts, no fuel, short runway, etc

- **System feature**
  - e.g., for a telephone system: call forwarding, call blocking, conference call, etc

- **System response**
  - e.g., for a payroll system: generate pay-cheques, report costs, print tax info;

- **External object**
  - e.g. for a library information system, organize by book type

- **User type**
  - e.g. for a project support system: manager, technical staff, administrator, etc.

- **Mode**
  - e.g. for word processor: page layout mode, outline mode, text editing mode, etc

- **Subsystem**
  - e.g. for spacecraft: command&control, data handling, comms, instruments, etc.

→ Requirements documents templates

https://www.volere.org/templates/volare-requirements-specification-template/
Requirements Specification

• Requirements specification

• Criteria for good requirements
Typical mistakes

- **Noise**
  - text that carries no relevant information to any feature of the problem.

- **Silence**
  - a feature that is not covered by any text.

- **Over-specification**
  - text that describes a detailed design decision, rather than the problem.

- **Contradiction**
  - text that defines a single feature in a number of incompatible ways.

- **Ambiguity**
  - text that can be interpreted in at least two different ways.

- **Forward reference**
  - text that refers to a terms or features yet to be defined.

- **Wishful thinking**
  - text that defines a feature that cannot possibly be validated.

- **Requirements on users**
  - Cannot require users to do certain things, can only assume that they will

- **Jigsaw puzzles**
  - distributing key information across a document and then cross-referencing

- **Duckspeak requirements**
  - Requirements that are only there to conform to standards

- **Unnecessary invention of terminology**
  - E.g. ‘user input presentation function’

- **Inconsistent terminology**
  - Inventing and then changing terminology

- **Putting the onus on the developers**
  - i.e. making the reader work hard to decipher the intent

- **Writing for the hostile reader**
  - There are fewer of these than friendly readers
Do not write like this

→ Ambiguity – or

➤ The same subsystem shall also be able to generate visible or audible caution or warning signal for the attention of security or business analyst

→ Multiple requirements – and, or, with, also

➤ The warning lamp shall light up when system intrusions is detected and the current workspace or input shall be saved
Do not write like this

→ Let-out clauses
  if, when, except, unless, although, always

  The fire alarm shall always be sounded when the smoke is detected, unless the alarm is being tested when the antivirus is deployed

→ Long rumpling sentences

  Provided that the designated input signals from the specified devices are received in the correct order where the systems is able to differentiate the designators, the security solution should comply with the required framework of Section 3.1.5 to indicate the desired security states
Do not write like this

→ System design:
  no names of components, materials, software objects/procedures, database fields

☞ The antenna shall be capable of receiving FM signals, using a copper core with nylon armoring and a waterproof hardened rubber shield

→ Mix of requirements and design:
  no references to system, design, testing, or installation

☞ The user shall be able to view the current selected channel number which shall be displayed in 14pt Swiss type on an LCD panel tested to standard 657-89 and mounted with shockproof rubber washers
Do not write like this

→ Speculation
   *usually, generally, often normally, typically*
   
   ✗ Users normally require early indication of intrusion into the system

→ Vague, undefinable terms
   *user-friendly, versatile, approximately, as possible, efficient, improved, high-performance, modern*

   ✗ Security-related messages should be versatile and user-friendly
   ✗ The OK status indicator lamp shall be illuminated as soon as possible after system security self-check is completed
Do not write like this

→ Wishful thinking

100% reliable/ safe/ secure. Handle all unexpected failures. Please all users. Run on all platforms. Never fail. Upgrade to all future situations.

☞ The gearbox shall be 100% secure in normal operation.
☞ The network shall handle all unexpected errors without crashing.
Good requirements

→ Use simple direct sentences

☞ Security analyst should be able to view system status

→ Use a limited vocabulary

☞ Security analyst should be able to change the infected component in less than 12 hours
☞ Security analyst should be able to reconfigure the infected component in less than 12 hours
Good requirements

→ Identify the type of user who wants each requirements

▷ The navigator shall be able to...

→ Focus on stating result

▷ ... view storm clouds by radar ...

→ Define verifiable criteria

▷ ... at least 100 km ahead.

▷ Acceptance criterion: Aircraft flying at 800km/h at 10,000 meters towards a known storm cloud indicated by meteorology satellite report; storm cloud is detected at a range of at least 100 km.
Criteria for Writing Good Requirements

→ **What, not how (external observability)**
  - Avoid premature design or implementation decisions

→ **Understandability, clarity (not ambiguous)**

→ **Cohesiveness (one thing per requirement)**

→ **Testability**
  - Somehow possible to test or validate whether the requirement has been met, clear acceptance criteria
  - Often requires quantification, this is more difficult for security than e.g. for performance
    - "The response time of function F should be max 2 seconds"
    - "The security of function F should be at least 99.9 %"
Criteria for Writing Good Requirements

→ What, not how (external observability)
  ✿ Avoid premature design or implementation decisions

→ Understandability, clarity (not ambiguous)

→ Cohesiveness (one thing per requirement)

→ Testability
  ✿ Somehow possible to test or validate whether the requirement has been met, clear acceptance criteria
  ✿ Often requires quantification, this is more difficult for security than e.g. for performance
    ➢ "The response time of function F should be max 2 seconds"
    ➢ "The security of function F should be at least 99.9 %"
Documenting requirements artefacts

- **Identifier**
  - 1
- **Description**
  - 1
- **Requirements artefact**
  - 0..* 0..*
  - 1
- **Goal**
  - 1..*
  - 1..*
  - example of satisfaction
- **Scenario**
  - 1..*
  - 1..*

**Relationships:**
- **has**
- **is realised by**
- **derived from**
- **complete, disjoint**
- **contributed to realisation of**
- **example of satisfaction**
- **has**
Requirement Shell
Volere template, 2010

Requirement # : Unique id
Requirement Type : Event/use case # :

Description : A one sentence statement of the intention of the requirement

Rationale : A justification of the requirement

Originator : Who raised this requirement?

Fit Criterion : A measurement of the requirement such that it is possible to test if the solution matches the original requirement

Customer Satisfaction :

Customer Disatisfaction :

Priority : The relative urgency of this requirement

Conflicts :

Supporting Materials : Pointer to documents that illustrate and explain this requirement

Other requirements that cannot be implemented if this one is

History : Creation, changes, deletions, etc.

Measure of stakeholder unhappiness if this requirement is not part of the final product. Scale from 1 = hardly matters to 5 = extremely displeased.

Degree of stakeholder happiness if this requirement is successfully implemented. Scale from 1 = uninterested to 5 = extremely pleased.
Things to Take Home

- Requirements specification
- Criteria for good requirements