**“Requirements Lifecycle”**

*Source: Adapted from Pohl, CAISE 1993*

**Specification**

- **complete**
- **fair**
- **vague**

**Agreement**

- **common view**
- **personal view**

**Representation**

- **informal**
- **semi-formal**
- **formal**
Requirements Specification

• What is Requirements Specification?

• Criteria for good requirements

→ 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>Project A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose of spec?</strong></td>
</tr>
<tr>
<td><strong>Management view?</strong></td>
</tr>
<tr>
<td><strong>Readers?</strong></td>
</tr>
</tbody>
</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>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose of spec?</strong></td>
</tr>
<tr>
<td><strong>Management view?</strong></td>
</tr>
<tr>
<td><strong>Readers?</strong></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

Specification Contents

→ 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

<table>
<thead>
<tr>
<th>1 Introduction</th>
<th>Identifies the product, &amp; application domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>Scope</td>
<td></td>
</tr>
<tr>
<td>Definitions, acronyms, abbreviations</td>
<td></td>
</tr>
<tr>
<td>Reference documents</td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2 Overall Description</th>
<th>Describes contents and structure of the remainder of the SRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product perspective</td>
<td></td>
</tr>
<tr>
<td>Product functions</td>
<td></td>
</tr>
<tr>
<td>User characteristics</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td>Assumptions and Dependencies</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 Specific Requirements</th>
<th>Describes all external interfaces: system, user, hardware, software; also operations and site adaptation, and hardware constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendices</td>
<td>Summary of major functions, e.g. use cases</td>
</tr>
<tr>
<td>Index</td>
<td>Anything that will limit the developer’s options (e.g. regulations, reliability, criticality, hardware limitations, parallelism, etc)</td>
</tr>
<tr>
<td></td>
<td>All the requirements go in here (i.e. this is the body of the document). IEEE STD provides 8 different templates for this section</td>
</tr>
</tbody>
</table>
IEEE STD Section 3 (example)

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

   http://www.volere.co.uk/template.htm
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

Alexander and Stevens, Writing Better Requirements, 2002, Pearson Education Ltd

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

Alexander and Stevens, Writing Better Requirements, 2002, Pearson Education Ltd
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 %"
SecReq.1: Automation of the receipt generation or checking fuel capacity and sold amount of fuel as often as possible.

SecReq.2: Cash register computer must compare data about company car / employee ID to ERP and give warning if there is a mismatch.

SecReq.3: IT department must follow security advisories and patch the vulnerable systems, given that solutions are available.

SecReq.4: Implementing at least two workplaces for serving customers in filling station.

SecReq.5: Appropriate trainings shall be provided to all employees of PowerAB with regards to information security awareness.

SecReq.6: Database backup server accounts should be sufficiently protected against unauthorized access.

SecReq.7: IT managers should plan the placement of card readers near 24/7 cameras, which are monitored from a security officer.

SecReq.8: Have a backup server for redundancy and push uptime to near 100%.

SecReq.9: Using better security applications.

SecReq.10: The firewall should continuously monitor the communication channel and block suspicious software while transmitting message from account department to personnel department.

SecReq.1.1: Fuel system should check fuel capacity

SecReq.1.2: Receipt system should generate receipts

SecReq.2: Cash register computer should notify attendant about mismatch between car number and employee ID

SecReq.3: IT department should patch PowerAB software systems

SecReq.4: Management should create two workplaces for serving customers.

SecReq.5: Management should provide security awareness training for the PowerAB employees.

SecReq.6: Database backup server accounts should be protected against unauthorized access.

SecReq.7: Security officer should monitor the card readers.

SecReq.8: The backup server should ensure the uptime of 99.1%.

SecReq.9: “I do not know how to fix it !!!”

SecReq.10.1: The firewall should monitor the communication channel.

SecReq.10.2: The firewall should block the untrusted software. Explain "untrusted"!