Lecture 5: Requirements Specification and its Quality

• Requirements specification

• Criteria for good requirements

• Specification quality

Thanks to Prof. Steve Easterbrook, University of Toronto
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

→ Consider two different projects:

A) Tiny project, 1 programmer, 2 months work
   programmer talks to customer, then writes up a 2-page memo

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></th>
<th>Project A</th>
<th>Project B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose of spec?</strong></td>
<td>Crystalizes programmer’s understanding; feedback to customer</td>
<td>Build-to document; must contain enough detail for all the programmers</td>
</tr>
<tr>
<td><strong>Management view?</strong></td>
<td>Spec is irrelevant; have already allocated resources</td>
<td>Will use the spec to estimate resource needs and plan the development</td>
</tr>
<tr>
<td><strong>Readers?</strong></td>
<td><strong>Primary</strong>: Spec author; <strong>Secondary</strong>: Customer</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

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

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
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

Requirement Shell
Volere template, 2010

---

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

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

---

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 :
Priority : The relative urgency of this requirement
Supporting Materials : Pointer to documents that illustrate and explain this requirement
History : Creation, changes, deletions, etc.

Customer Dissatisfaction :
Conflicts :
Other requirements that cannot be implemented if this one is
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

Lecture 5: Requirements Specification and its Quality

• Requirements specification

• Criteria for good requirements

• Specification quality
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*

áticas 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

áticas 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*

áticas 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*

áticas 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 %"

**Exercise**

→ The application shall verify the identity of all of its users before allowing them to use all its capabilities

→ The system shall let users to log in with passwords of at least 8 characters, containing both small and capital letters, numbers and special signs

→ The system shall use Norton antivirus protection

→ The application shall disinfect any file found to contain a harmful program if disinfection is possible

→ The system shall encrypt all confidential data using the RSA algorithm
Exercise

→ The application shall verify the identity of all of its users before allowing them to use all its capabilities.

→ The system shall let users to log in with passwords of at least 8 characters, containing both small and capital letters, numbers and special signs.

→ The system shall use Norton antivirus protection.

→ The application shall disinfect any file found to contain a harmful program if disinfection is possible.

→ The system shall encrypt all confidential data using the RSA algorithm.

Lecture 5:
Requirements Specification and its Quality

• Requirements specification

• Criteria for good requirements

• Specification quality
Desiderata for Specifications

→ Valid (or “correct”)
  - Expresses the real needs of the stakeholders (customers, users, ...)
  - Does not contain anything that is not “required”

→ Unambiguous
  - Every statement can be read in exactly one way

→ Complete
  - All the things the system must do...
  - ...and all the things it must not do!
  - Conceptual Completeness
    - E.g. responses to all classes of input
  - Structural Completeness
    - E.g. no TBDs!!!

→ Understandable (Clear)
  - E.g. by non-computer specialists

→ Consistent
  - Doesn’t contradict itself
  - Uses all terms consistently

→ Ranked
  - Indicates relative importance / stability of each requirement

→ Verifiable
  - A process exists to test satisfaction of each requirement

→ Modifiable
  - Can be changed without difficulty
    - Good structure and cross-referencing

→ Traceable
  - Origin of each requirement is clear
  - Labels each requirement for future referencing

SEQUAL

→ L - language extension
→ D - the domain
→ M - externalised model
→ $K_s$ - relevant explicit knowledge of the stakeholders
→ $K_m$ - relevant explicit knowledge of modellers
→ I - social actor interpretation
→ T - technical actor interpretation
→ G - organisationally-motivated goals of the modelling task
Physical quality

→ Although information systems specifications and models are not of the physical kind, any model can be represent physically - e.g. on disk or paper

→ Specification should be
  Electronically stored
Empirical quality

→ **Empirical** quality deals with the variety of elements distinguished, error frequencies when being written or read, coding and ergonomics for computer-human interaction, for documentation and tools.

→ Specification should be:
  - **Understandable**
    - all classes readers can easily comprehend the meaning of all requirements with a minimum of explanation.
  - **Concise**
    - it is as short as possible without affecting any other quality of the requirements specification.

Pragmatic quality

→ **Pragmatic** quality is the correspondence between the model and the audience’s interpretation of it.

→ Specification should be:
  - **Executable/Interpretable/Prototypable**
    - there exists a software tool capable of inputting the requirements specification and providing a dynamic behavioural model.
  - **Organised**
    - its contents are arranged so that readers can easily locate information and logical relationships among adjacent sections are apparent.
  - **Cross-referenced**
    - Cross-references are used to relate sections containing requirements with other relative requirements
Social quality

→ The goal for social quality is agreement.

→ Tool support
  ➜ models created based on the different internal reality of the participants that are to agree

Syntactic quality

→ Syntactic quality includes correctness of lexicon, syntax and structural quality.

→ Syntactic errors:
  ➜ Syntactic invalidity
  ➜ Syntactic incompleteness

→ Automated tool support:
  ➜ Error prevention
  ➜ Error detection
  ➜ Error correction
Semantic quality

→ **Semantic** quality is the correspondence between the model and the modelling domain

→ **Feasibility:**

→ Attempts at reaching a state of total validity and completeness will lead to unlimited spending of time and money on the modelling activity.

→ The time to terminate a modelling activity is thus not when the model is “perfect” (*which will never happen*), but when it has reached a state where further modelling is regarded to be less beneficial than applying the model in its current state.

Semantic quality

→ **Complete**

- everything that the software is supposed to do is included;
- responses of the software to all realisable classes of input data in all recognisable classes of situations are included;
- all pages are numbered, all figures and tables are numbered, named, and referenced; all terms are defined; all units of measure are provided; and all referenced material are present;
- no sections are marked "To be determined".

→ **Correct**

- every requirement represent something required of the system to be built.

<table>
<thead>
<tr>
<th>User needs</th>
<th>Requirement: the system must respond button press within 5 seconds.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User needs that the system must respond button press within 5 seconds.</td>
<td>Requirement: the system must respond button press within 10 seconds.</td>
</tr>
</tbody>
</table>
Semantic quality

→ Internally consistent
  ➜ no subset of individual requirements stated therein conflict

| a) The light shall be lit when the button is pressed. |
| b) When the button is released, the light shall become lit. |

| a) The system should prompt the message |
| b) The system should cue the message |

→ Externally consistent
  ➜ no requirement stated therein conflict with any already base-lined project documentation

Semantic quality

→ Annotated by
  ➜ relative importance, relevant stability, version

→ Precise
  ➜ (1) numeric quantities are used whenever possible;
  ➜ (2) the appropriate levels of precision are used for all numeric quantities.

→ Traced
  ➜ the origin of each of its requirements is clear

The system shall respond to any occurrence of request X within 20 seconds.

→ Traceable
  ➜ it is written in a manner that facilitates the referencing of each individual statement

→ Verifiable
  ➜ there exist finite, cost effective techniques that can be used to verify that every requirement stated therein is satisfied by the system to be built.
Semantic quality

→ Achievable
- there could exist at least one system design and implementation that correctly implements all the requirements stated in the requirements specification

→ Design-independent
- there exist more than one system design and implementation that correctly implements all requirements stated in the requirements specification

→ At the right level of detail
- specific enough so that any system built that satisfies the requirements in the specification satisfies all user needs
- abstract enough so that all systems that satisfy all user needs also satisfy all requirements

Semantic quality

→ Modifiable
- structure and style are such that any changes can be made easily, completely and consistently

→ Unambiguous
- every requirement stated therein has only one possible interpretation

Up to 12 aircraft, the small display format shall be used. Otherwise the large display format shall be used.

<table>
<thead>
<tr>
<th>Aircraft that are non-friendly and have an unknown mission or the potential to enter restricted airspace within 5 minutes shall rise an alert.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft that are either non-friendly and have an unknown mission or have the potential to enter restricted airspace within 5 minutes shall rise an alert.</td>
</tr>
</tbody>
</table>
Orthogonal aspects

➔Reusable

❖ its sentences, paragraphs, and sections can be easily adopted and adapted for use in subsequent requirements specification.

❖ Physical quality
  ➢ a persistent form that is available to those who potentially will want to reuse it

❖ Syntactic quality
  ➢ syntactically correct

❖ Semantic quality
  ➢ Similar domains
  ➢ white-box reuse - modifiable, comprehensible and comprehended (techniques of pragmatic quality), annotated, and other

❖ Social quality
  ➢ model integration and conflict resolution can be useful to investigate to what extent the solutions based on the model to be reused, should be reused.

Lecture 5:
Requirements Specification and its Quality

• Requirements specification

• Criteria for good requirements

• Specification quality
THERE IS NO SUCH THING AS A PERFECT REQUIREMENTS SPECIFICATION!