Week 6 Lesson 12
Writing Requirements

Prof. Michael Zink
Rationale

The purpose of this lesson is to present the aspects of writing requirements that are common to every development layer. Wherever the generic process is instantiated, certain principles and techniques are constant in their application to the expression and structuring of requirements.
Objectives

• Become familiar with the concept writing requirements
• Understand how requirements documents should be structured
• Understand the role of attributes
• Become familiar with the language of requirements
• Become familiar with the writing of requirement statements
Overview

- Introduction
- Requirements for requirements
- Structuring requirements documents
- Key requirements
- Using attributes
- Ensuring consistency across requirements
- Value of a requirement
- The language of requirements
- Requirement boilerplates
- Granularity of requirements
- Criteria for requirements statements
Introduction

• Requirements engineering is technical process
• Unlike other kinds writing, even unlike technical writing seen in instruction manuals and user guides
• Two aspects have to be carefully balanced:
  • Need to make requirements document readable
  • Need to make the set of requirements processable
• Word processor alone is not sufficient to manage requirements, same is true for database
• Writing and reviewing of requirements should go hand-in-hand
Requirements for Requirements

- Review some of the objectives and purpose for writing requirements
- First, identification of stakeholders
- Capabilities required by various stakeholders
- Relation to how requirements and statements are written
- Basic elements to perform:
  - Identification
  - Classification
## Requirements for Requirements

### Stakeholders for Requirements

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Creates the requirements and incorporates changes</td>
</tr>
<tr>
<td>Publisher</td>
<td>Issues and archives the requirements document</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Reviews the requirements and suggests changes</td>
</tr>
<tr>
<td>Implementor</td>
<td>Analyses the requirements and negotiates challenges</td>
</tr>
</tbody>
</table>
Requirements for Requirements: Abilities I

- Ability to uniquely identify every statement of requirement
- Ability to classify every statement of requirement in multiple ways, such as:
  - By importance
  - By type (e.g., functional, performance, constraint, safety)
  - By urgency (when it has to be provided)
Requirements for Requirements: Abilities II

- Ability to track the status of every statement of requirement, in support of multiple processes, such as:
  - Review status
  - Satisfaction status
  - Qualification status
Requirements for Requirements: Abilities III

- Ability to elaborate a requirement in multiple ways, such as providing:
  - Performance information
  - Quantification
  - Test criteria
  - Rationale
  - Comments
Requirements for Requirements: Abilities IV

- Ability to view a statement of requirement in the document context, i.e., alongside its surrounding statements
- Ability to navigate through a requirements document to find requirements according to a particular context
- Ability to trace to any individual statement of requirement
Structuring Requirements Documents

• Requirements documentation can be very large
• E.g., requirements for aircraft carrier may fill many filing cabinets
• Having a well-understood, clearly documented structure for whole requirements set is essential for effective management of complexity
Structuring Requirements Documents

- Organizing requirements into the right structure:
  - *Minimizing* the number of requirements
  - *Understand* large amounts of information
  - *Find* sets of requirements relating to particular topics
  - *Detect* omissions and duplications
  - *Eliminate* conflicts between requirements
  - *Manage* iteration (e.g. delayed requirements)
  - *Reject* poor requirements
  - *Evaluate* requirements
  - *Reuse* requirements across projects
Structuring Requirements Documents

- Requirements may also contain variety of technical and non-technical text:
  - *Background information*: places the requirements in context
  - *External context*: describing the enclosing system
  - *Definition of the scope* of the requirements
  - *Definitions of terms* used in the requirements statements
  - *Descriptive text* which bridges different sections of the document
  - *Stakeholder descriptions*
  - *Summary of models* used in deriving requirements
  - *References* to other documents
Key Requirements

- Key User Requirements (KUR) or Key Performance Indicators (KPI)
- Small subset abstracted from the whole that capture essence of system
- Every key requirement should solicit a negative response to the question: If the solution didn’t provide me with this capability, would I still buy it?
- Or at the system level: If the system didn’t do this, would I still want it?
- Key requirements become those that are absolutely mandatory
Using Attributes

• Simple textual statement is not sufficient to fully define a requirement
• Other classification and status information that each requirement carries
• Rather than clutter text of a requirement, additional information should be placed in attributes (attached to requirements)
• Allows structure for ease of processing, filtering, sorting, etc.
• Attributes can support many of the abilities mentioned earlier
Using Attributes

[SH234] The ambulance control system shall be able to handle up to 100 simultaneous calls.

Source: R. Thomas
Priority: Mandatory
Release: 1
Review Status: Accepted
Verifiable: Yes
Verification: By simulation, then by system test
<table>
<thead>
<tr>
<th><strong>Category</strong></th>
<th><strong>Example Values</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identification</strong></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Unique reference</td>
</tr>
<tr>
<td>Name</td>
<td>Unique name summarizing subject of requirement</td>
</tr>
<tr>
<td><strong>Intrinsic characteristics</strong></td>
<td></td>
</tr>
<tr>
<td>Basic type</td>
<td>Functional, performance, quality factor, interface, constraint, non-requirement</td>
</tr>
<tr>
<td>Quality factor subtype</td>
<td>Availability, flexibility, integrity, maintainability, portability, reliability, safety, security, supportability, sustainability, usability, workmanship</td>
</tr>
<tr>
<td>Product/process type</td>
<td>Product, process, data, service</td>
</tr>
<tr>
<td>Quantitative/qualitative type</td>
<td>Quantitative, qualitative</td>
</tr>
<tr>
<td>Life-cycle phase</td>
<td>Pre-concept, concept, development, manufacturing, integration/test, deployment/delivery/installation, operation, support, disposal</td>
</tr>
<tr>
<td><strong>Priority and importance</strong></td>
<td></td>
</tr>
<tr>
<td>Priority (compliance level)</td>
<td>Key, mandatory, optional, desirable</td>
</tr>
<tr>
<td>Importance</td>
<td>1-10</td>
</tr>
</tbody>
</table>
Ensuring Consistency Across Requirements

• Often concern is to identify conflicting requirements
• What techniques can be applied to identify potential inconsistencies?
  • Clarifying requirements
  • Filtering and sorting techniques
• Requirements can touch on several aspects of system
  • E.g., requirement about engine performance may also contain safety element
  • Requirement should be viewed in both performance and safety context
Ensuring Consistency Across Requirements

- To facilitate this, requirements can be given primary and secondary classifications.
- Typically, single primary and multiple secondary classifications.
- Thorough review process can include filtering of statements by keywords used in classifications.
- This is an initial step to identify potential conflicts in requirements.
Value of a Requirement

• Non-negotiable requirements:
  • If not met, product is of no use
• Negotiable requirements:
  • System required to support 100 simultaneous users but delivered solution only supports 99
  • Most likely still of some value to customer
• Capturing the value of requirement can be challenging
  • Target: 100
  • Acceptable: 75
  • Unacceptable: < 50
  • Even better: 200
Value of a Requirement

• Provide several performance values
• Example, three-valued approach:
  • M: the mandatory lower (upper) limit
  • D: the desired value
  • B: the best value
• Values can be held in separate attributes within text labeled form:
  • The system shall support [M:50, D:100, B:200] simultaneous users
Value of a Requirement

- Alternative approach: supplying function that maps performance to some representation of value.
The Language of Requirements

• Use of consistent language makes it easier to identify different kinds of requirements
• E.g., “shall” as key work to indicate presence of requirement in text
• Language used will depend on level of requirement:
  • Stakeholder requirements that lie in problem domain
  • System requirements that lie in solutions domain
Stakeholder Requirements

• Concerned with capability and constrained on capability

• Express single capability required by one or more identified stakeholders:
  The <stakeholder type> shall be able to <capability>.

• Aspects of performance or constraints may also be stated in text:
  The <stakeholder type> shall be able to <capability> within <performance> of <event> while <operational condition>.

• Example:
  The weapons operator shall be able to fire a missile within 3 seconds of radar sighting while in severe sea conditions.
Stakeholder Requirements

- Constraint requirements:
  The <stakeholder> shall not be placed in breach of <applicable law>.

  The ambulance driver shall not be placed in breach of national road regulations.
Systems Requirements

• Language is slightly different since requirements lie in solutions domain

• Focus is on function and constraint on the system:
  The <system> shall <function> not less than <quantity> <object> while <operational conditions>.

• Example:
  The communications system shall sustain telephone contact with not less than 10 callers while in the absence of external power.
Systems Requirements

• Periodicity constraint:
The <system> shall <function> <object> every <performance> <units>.

• Example:
The coffee machine shall produce a hot drink every 10 seconds.
Requirements Boilerplates

• Extension of the concept of “Boilerplates” shown earlier
• Good way of standardizing language for requirements
• Application to the collection and expression of constraint requirements
• Create palettes of boilerplates
Requirements Boilerplates

- Expressing requirements through boilerplate now becomes process of:
  - Selecting the most appropriate boilerplate from palette
  - Providing data to complete placeholders
Requirements Boilerplates: Global Templates

The <system> shall <function> <object>
every <performance> <units>

Requirement 347 = Template 34 +
<system> = coffee machine
<function> = produce
<object> = a hot drink
<performance> = 10
<units> = seconds

Requirement 348 = Template 34 +
<system> = coffee machine
<function> = produce
<object> = a cold drink
<performance> = 5
<units> = seconds
Requirements Boilerplates

- Separating templates has the following advantages:
  - *Global changes in style can be effected*: to change the way certain requirements are expressed, only centrally held boilerplate needs to be changed.
  - *System information can be processed more easily*: e.g., collection all “<operational conditions>” placeholders into a separate attribute allows for easy sorting and filtering.
  - *Confidential information can be protected*: boilerplates can be used to separate out parts that need to be protected.
**Requirements Boilerplates: Constraints**

- Approach to capturing constraint requirements:
  1. Collect all capability requirements first
  2. Construct a list of all different kinds of constraints that may need to be expressed
     - If list is based on past experience of same kind of system, boilerplates should exist for each kind.
     - Otherwise, suitable boilerplates may have to be defined
Requirements Boilerplates: Constraints

• Approach to capturing constraint requirements:
  3. For each capability, consider each kind of constraint, determine whether a constraint needs to be captured (see table on next slide)
  4. Select the boilerplate that best matches the constraint to be expressed, and instantiate it
  5. Process finished when every “cell” has been considered
## Requirements Boilerplates: Constraints

<table>
<thead>
<tr>
<th>Type of constraint</th>
<th>Boilerplate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance/capability</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>not less than</strong> <code>&lt;performance&gt;</code> <strong>times per</strong> <code>&lt;units&gt;</code></td>
</tr>
<tr>
<td>Performance/capability</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>of type</strong> <code>&lt;qualification&gt;</code> <strong>within</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code></td>
</tr>
<tr>
<td>Performance/capacity</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <strong>not less than</strong> <code>&lt;quantity&gt;</code> <code>&lt;object&gt;</code></td>
</tr>
<tr>
<td>Performance/timeliness</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>within</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> <strong>from</strong> <code>&lt;event&gt;</code></td>
</tr>
<tr>
<td>Performance/periodicity</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>not less than</strong> <code>&lt;quantity&gt;</code> <code>&lt;object&gt;</code> <strong>within</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code></td>
</tr>
<tr>
<td>Interoperability/capacity</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>composed of not less than</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> <strong>with</strong> <code>&lt;external entity&gt;</code></td>
</tr>
<tr>
<td>Sustainability/periodicity</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>for</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> <strong>every</strong> <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code></td>
</tr>
<tr>
<td>Environmental/operability</td>
<td>The <code>&lt;system&gt;</code> shall be able to <code>&lt;function&gt;</code> <code>&lt;object&gt;</code> <strong>while</strong> <code>&lt;operational condition&gt;</code></td>
</tr>
</tbody>
</table>
Granularity of Requirements

• How far to “split atom” in requirements engineering?
• Requirements statements can be decomposed into subclauses as long as it is assured that subclauses are always visible in context

<table>
<thead>
<tr>
<th>The communications system shall sustain telephone contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>With not less than 10 callers</td>
</tr>
<tr>
<td>While in the absence of external power</td>
</tr>
</tbody>
</table>
Granularity of Requirements

• The communications system shall sustain telephone contact
  With not less than 10 callers
  While in the absence of external power

• Traceability statements that can be cited from above:
  • The communications system shall sustain telephone contact
  • *The communications system shall sustain telephone contact*
    with not less than 10 callers
  • *The communications system shall sustain telephone contact*
    while in the absence of external power
Granularity of Requirements

While in the absence of external power,

- the communications system shall sustain telephone contact
  - with not less than 10 callers
- the communications system shall sustain radio contact
  - with not less than 15 ambulance drivers

• Now traceable statements that can be sited are:
  • While in the absence of external power, the communications system shall sustain telephone contact.
  • While in the absence of external power, the communications system shall sustain telephone contact with not less than 10 callers
  • While in the absence of external power, the communications system shall sustain radio contact with not less than 15 ambulance drivers
Criteria for Writing Requirements Statements

- Criteria every statement of requirement should meet:
  - Atomic: each statement carries a single traceable element
  - Unique: each statement can be uniquely identified
  - Feasible: technically possible within cost and schedule
Criteria for Writing Requirements Statements

- Criteria every statement of requirement should meet:
  - *Legal*: Legally possible
  - *Clear*: each statement is clearly understandable
  - *Precise*: each statement is precise and concise
  - *Verifiable*: each statement is verifiable, and it is known how
  - *Abstract*: does not impose a solution of design specific to the layer below
Criteria for Writing Requirements Statements

• Additional criteria that apply to set of requirements as a whole:
  • Complete: all requirements are present
  • Consistent: no two requirements are in conflict
  • Non-redundant: each requirement is expressed once
  • Modular: requirements statements that belong together are close to one
Criteria for Writing Requirements Statements

• Additional criteria that apply to set of requirements as a whole:
  • *Structured*: there is a clear structure of the requirements document
  • *Satisfied*: the appropriate degree of traceability coverage has been achieved
  • *Qualified*: the appropriate degree of traceability coverage has been achieved
Criteria for Writing Requirements Statements

The system shall perform a maximum rating at all times except that in emergencies it shall be capable of providing up to 125% rating unless the emergency condition continues for more than 15 minutes in which case the rating shall be reduced to 105% but in the event that only 95% can be achieved then the system shall activate a reduced rating exception and shall maintain the rating within 10% of the stated values for a minimum of 30 minutes.
Criteria for Writing Requirements Statements

The system shall provide general word processing facilities which shall be easy to use by untrained staff and shall run on thin Ethernet Local Area Network wired into the overhead ducting with integrated interface cards housed in each system together with additional memory if that should be necessary.
Criteria for Writing Requirements Statements

• Additional pitfalls that should be avoided:
  • Avoid rambling: conciseness is a virtue; it does not have to read like a novel;
  • Avoid let-out clauses: such as “if that should be necessary”, as they render the requirements useless;
  • Avoid speculation;
Criteria for Writing Requirements Statements

- Additional pitfalls that should be avoided:
  - *Avoid vague words*: usually, generally, often, normally, typically;
  - *Avoid vague terms*: user friendly, versatile, flexible;
  - *Avoid wishful thinking*: 100% reliable, please all users, safe, run on all platforms, never fail, handle all unexpected failures, upgradeable to all future situations
Examples

- FAA NAS Requirements Document: https://sep.faa.gov/file/get/2718
Summary

• Define and outline structure at the outset, preferably hierarchical, and improve it as you go
• Write down requirements as soon as possible, even if they are imperfect
• Determine in advance what attributes will be used to classify and elaborate the textual statement
• Produce an initial version rapidly to stimulate immediate feedback
• Perfect the requirements as you go, removing repetition, unwarranted design, and inconsistency
• Brainstorm and hold informal reviews continually, with rapid turnaround of versions
• Exposure to users is much better than analysis by “experts”
Post-Work

The rules to follow when writing requirements are as follows:
• Use simple, direct language
• Write testable requirements
• Use defined and agreed terminology
• Write one requirement at a time

With these rules in mind, specify a minimum of 10 requirements for the system you plan to work on for your course project.