Writing Requirements

Analisis dan Desain SI

Fajar Pradana S.ST., M.Eng
Goal

- Define and understand the characteristics of and rules for writing good requirements.
- Understand the value of providing the rationale and the preliminary verification technique with each requirement.
Introduction

• Requirements engineering is a technical process
• It is certainly not like writing a novel, or a book like this; it is not even like the kind of “technical writing” seen in instruction manuals and user guides
• In writing a requirements document, two aspects have to be carefully balanced:
  ▪ The need to make the requirements document readable
  ▪ The need to make the set of requirements processable
Good Requirements Are SMART

• **Specific** -
  ▪ It must address only one aspect of the system design or performance
  ▪ It must be expressed in terms of the need (what and how well), not the solution (how).

• **Measurable** -
  ▪ Performance is expressed objectively and quantitatively
  ▪ E.g., an exact pointing requirement (in degrees) can be tested thus verified prior to launch.

• **Achievable** -
  ▪ It must be technically achievable at costs considered affordable
  ▪ E.g., JWST early designs specified an aperture requirement eventually descoped due to technical issues with deployment.
Good Requirements Are SMART

• Relevant -
  ▪ It must be appropriate for the level being specified
  ▪ E.g., requirement on the solar cells should not be designated at the spacecraft level.

• Traceable -
  ▪ Lower level requirements (children) must clearly flow from and support higher level requirements (parents).
  ▪ Requirements without a parent are referred to as orphans, and need to be assessed for necessity of inclusion.
Rules for Writing Good Requirements

• Requirements have mandatory characteristics:
  ▪ Needed
  ▪ Verifiable
  ▪ Attainable: technically, cost, schedule

• Each requirement should
  ▪ Express one thought
  ▪ Be concise and simple
  ▪ Be stated positively
  ▪ Be grammatically correct; free of typos and misspellings
  ▪ Be understood only one way; they are unambiguous
  ▪ Use consistent terminology to refer to the system/product and its lower level entities
  ▪ Comply with the project’s template and style rules
More Rules for Writing Good Requirements

• What a requirement must state:
  ▪ WHO is responsible
  ▪ WHAT shall be done
  ▪ Or HOW WELL something shall be done
  ▪ Or under what CONSTRAINTS something shall be done

• Requirement format: “WHO shall WHAT”
  ▪ Uses active not passive voice

• Example product requirements:
  ▪ The system shall operate at a power level of...
  ▪ The software shall acquire data from the...
  ▪ The structure shall withstand loads of...
  ▪ The hardware shall have a mass of...
More Rules for Writing Good Requirements

• Use the correct terms:
  ▪ Requirements are binding - Shall
  ▪ Facts or Declaration of purpose - Will
  ▪ Goals are non-mandatory provisions - Should
  ▪ Do NOT use “Must”
Ensuring Consistency Across Requirements

• A frequent concern in managing large sets of requirements is being able to identify conflicting requirements.
  ▪ The difficulty is in spotting that two statements many pages apart are in conflict

• classifying requirements in several ways, and using filtering and sorting techniques to draw together small numbers of statements that address the same topic
The Language of Requirements

• Using the following key words to indicate different priorities: shall, should, may
• A typical form to express capability requirements in the problem domain:
  The <stakeholder type> shall be able to <capability>.
• Associating performance and constraint aspects to a requirement:
  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.
The Language of Requirements

• A typical constraint requirement takes the following form:
  The <stakeholder type> shall not be placed in breach of <applicable law>.

• Example:
  The ambulance driver shall not be placed in breach of national road regulations.
The Language of Requirements

- A typical form to express function requirements associated with performance and constraint aspects in the solution domain:
  
  The <system> shall <function>
  not less than <quantity> <object> while <operational condition>.

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

- Introducing a periodicity constraint:

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

  Example:
  The coffee machine shall produce a hot drink
  every 10 seconds.
Capturing Constraint Requirements

• Collect all capability requirements first.
• Construct a list of all the different kinds of constraint that may need to be expressed.
  ▪ If this list is based on past experience of the same kind of system, then boilerplates should exist for each kind. Otherwise suitable boiler-plates may have to be defined.
• For each capability, consider each kind of constraint, and determine whether a constraint needs to be captured.
  ▪ A large table could be used for this; in each cell, indicate where constraints exist by entering the appropriate sub-ordinate clauses to the requirement; where no constraint is necessary, enter “N/A” in the appropriate cell.
• Select the boilerplate that best matches the constraint to be expressed, and instantiate it.
• The process is finished when every “cell” has been considered.
## Requirement Boilerplates

<table>
<thead>
<tr>
<th>Type of Constraint</th>
<th>Boiler-Plate</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> not less than <code>&lt;performance&gt;</code> times per <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> of type <code>&lt;qualification&gt;</code> within <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> Not less than <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> within <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> from <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> not less than <code>&lt;quantity&gt;</code> <code>&lt;object&gt;</code> within <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> composed of not less than <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> with <code>&lt;external entity&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> for <code>&lt;performance&gt;</code> <code>&lt;units&gt;</code> every <code>&lt;performance&gt;</code> <code>&lt;units&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> while <code>&lt;operational condition&gt;</code>.</td>
</tr>
</tbody>
</table>
Goodness Checklist: Is this Requirement...

• Free of ambiguous terms?
  ▪ Examples: as appropriate, etc., and/or, support, but not limited to, be able to, be capable of

• Free of indefinite pronouns?
  ▪ Examples: this, these

• Free of unverifiable terms?
  ▪ Examples: flexible, user-friendly, robust, light-weight, maximize, adequate, small, portable, easily - other “ly” words and other “ize” words

• Free of implementation?
  ▪ Requirement should state WHAT is needed, NOT HOW to provide it, i.e., state the problem not the solution.
Goodness Checklist: Is this Requirement...

• Necessary?
  ▪ Ask “Why do you need the requirement?”; the answer may lead you to the real requirement.

• Free of descriptions of operations?
  ▪ To distinguish between operations and requirements, ask “Does the developer have control over this?” “Is this a need the product must satisfy or an activity involving the product?”

• Free of TBDs (To Be Determined)?
  ▪ Use a best estimate and a TBR (To Be Resolved) with rationale when possible.
Rationale Captures the Motivation and Assumptions of a Requirement

• The rationale of each requirement defines
  ▪ Why a requirement is needed
  ▪ What assumptions were made
  ▪ What design effort drove the requirement
  ▪ Other data that will be needed to maintain the requirement over time

• Example
  ▪ Requirement: “The truck shall have a height of no more than 14 feet.”
  ▪ Rationale: 99% of all US interstate highway overpasses have a 14 foot or greater clearance. (Assumptions: The truck will be used primarily on US interstate highways for long-haul freight in the US.)
Criteria for Writing a Single Requirement Statement

- **Atomic**: each statement carries a single traceable element.
- **Unique**: each statement can be uniquely identified.
- **Feasible**: technically possible within cost and schedule.
- **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 a Set of Requirements Statements

- **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 another.
- **Structured**: there is a clear structure to the requirements document.
- **Satisfied**: the appropriate degree of traceability coverage has been achieved.
- **Qualified**: the appropriate degree of traceability coverage has been achieved.
‘Nightmare’ Requirements Statements

• The system shall perform at the 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 min 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 min.

• The system shall provide general word processing facilities which shall be easy to use by untrained staff and shall run on a 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.
Pitfalls

- **Avoid rambling**: conciseness is a virtue; it doesn’t have to read like a novel.

- **Avoid let-out clauses**: such as “if that should be necessary”; they render the requirements useless.

- **Avoid putting more than more requirement in a paragraph**: often indicated by the presence of the word “and”.

- **Avoid speculation**.

- **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.
‘Nightmare’ Requirements Statements – Review

• The system shall perform at the 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 min 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 min.
  ▪ There could be 12 requirements present.
  ▪ A better approach would be to identify clearly the four different operational modes of the aircraft: normal, emergency, emergency more than 15 min, and reduced rating exception, and express a separate requirement for each.
‘Nightmare’ Requirements Statements – Review

• The system shall provide general word processing facilities which shall be easy to use by untrained staff and shall run on a 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.
  ▪ ‘Let-out clause’ is not clear what the scope of the clause is...
  ▪ Is it necessary required or not?
Terima Kasih
Ada Pertanyaan