Standards for Writing Requirements

Standards for Requirements Documents

- Based on the ANSI/IEEE Guide to Software Requirements STD 830-1984
- Requirements use the “shall” language
  - The system shall allow users to only enter numerical data.
- Requirements are clearly numbered
- Requirements should not be confused with background information
- Requirements are concise

Characteristics of a Good Requirements Document

- A good Requirements Document is:
  1. Unambiguous
  2. Complete
  3. Verifiable
  4. Consistent
  5. Modifiable
  6. Traceable
  7. Usable during the Operation and Maintenance Phase

Unambiguous

- A Requirements Document is unambiguous if and only if every requirement stated therein has only one interpretation.
  1. As a minimum, this requires that each characteristic of the final product be described using a single unique term.
  2. In cases where a term used in a particular context could have multiple meanings, the term must be included in a glossary where its meaning is made more specific.

Complete

- A Requirements Document is complete if it possesses the following qualities:
  1. Inclusion of all significant requirements, whether relating to functionality, performance, design constraints, attributes or external inter-faces.
  2. Definition of the responses of the system to all realizable classes of inputs in all realizable classes of situations. Note that it is important to specify the responses to valid and invalid input values.
  3. Conformity to any standard that applies to it. If a particular section of the standard is not applicable, the Requirements Document should include the section number and an explanation of why it is not applicable.
  4. Full labeling and referencing of all figures, tables, and diagrams in the Requirements Document and definition of all terms and units of measure.

Verifiable

- A Requirements Document is verifiable if and only if every requirement stated therein is verifiable. A requirement is verifiable if and only if there exists some finite cost-effective process with which a person or machine can check that the system product meets the requirement.

Verifiable continued

Examples of non-verifiable requirements include statements such as:

- *The product shall work well,* or *The product shall have a good human interface.* These requirements cannot be verified because it is impossible to define the terms *good* or *well.*
- *The program shall never enter an infinite loop.* This requirement is non-verifiable because the testing of this quality is theoretically impossible.
- *The output of the program shall usually be given within 10 s.* This requirement is non-verifiable because the term *usually* cannot be measured.

Verifiable continued

- An example of a verifiable statement is
  - *The output of the program shall be given within 20 s of event X,* 60% *of the time; and shall be given within 30 s of event* X, 100% *of the time.* This statement can be verified because it uses concrete terms and measurable quantities.
- If a method cannot be devised to determine whether the system meets a particular requirement, then that requirement should be removed or revised.
- If a requirement is not expressible in verifiable terms at the time the Requirements Document is prepared, then a point in the development cycle (review, test plan issue, etc) should be identified at which the requirement must be put into a verifiable form.

Consistent

- A Requirements Document is consistent if and only if no set of individual requirements described in it conflict.
- There are three types of likely conflicts in a Requirements Document:
  1. Two or more requirements might describe the same real world object but use different terms for that object. For example, a program's request for a user input might be called a *prompt* in one requirement and a *cue* in another.

Consistent

2. The specified characteristics of real world objects might conflict. For example:
    1. The format of an output report might be described in one requirement as *tabular* but in another as *textual.*
    2. One requirement might state that all lights shall be green while another states that all lights shall be blue.
3. There might be a logical or temporal conflict between two specified actions. For ex- ample:
    1. One requirement might specify that the system will add two inputs and another specify that the system will multiply them.
    2. One requirement might state that *A* must always follow *B,* while another requires that *A and B* occur simultaneously.

Modifiable

- A Requirements Document is modifiable if its structure and style are such that any necessary changes to the requirements can be made easily, completely, and consistently. Modifiability generally requires A Requirements Document to:
  1. Have a coherent and easy-to-use organization, with a table of contents, an index, and explicit cross-referencing.
  2. Not be redundant; that is, the same requirement should not appear in more than one place in the Requirements Document.
     - Redundancy itself is not an error, but it can easily lead to errors.
     - Redundancy can occasionally help to make a Requirements Document more readable, but a problem can arise when the redundant document is updated. Assume, for instance, that a certain requirement is stated in two places. At some later time, it is determined that the requirement should be altered, but the change is made in only one of the two locations-. The Requirements Document then becomes inconsistent.
     - Whenever redundancy is necessary, the Requirements Document should include explicit cross-references to make it modifiable.

Traceable

- A Requirements Document is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.
- Two types of traceability are recommended:

Traceable continued

1. Backward traceability (that is, to previous stages of development) depends upon each requirement explicitly referencing its source in previous documents.
2. Forward traceability (that is, to all documents spawned by the Requirements Document) depends upon each requirement in the Requirements Document having a unique name or reference number.

When a requirement in the Requirements Document represents an apportionment or a derivative of another re-quirement, both forward and backward trace-ability should be provided.

Usable During the Operation and Maintenance Phase

- The Requirements Document must address the needs of the operation and maintenance phase, including the eventual replacement of the system.

Usable During the Operation and Maintenance Phase

1. Maintenance is frequently carried out by personnel not associated with the original development. Local changes (corrections) can be implemented by means of a well-commented code. For changes of wider scope, however, the design and requirements documentation is es-sential. This implies two actions;
   a) The Requirements Document should be modifiable as indicated previously..
   b) The Requirements Document should contain a record of all special provisions that apply to individual components such as:
      i. Their criticality (for example, where failure could impact safety or cause large finan-cial or social losses).
      ii. Their relation to only temporary needs (for example, to support a display that may be retired soon).
      iii. Their origin (for example, function X is to be copied from an existing product in its entirety).
2. Knowledge of this type is taken for granted in the developing organization but is frequently missing in the maintenance organization. If the reason for or origin of a function is not understood, it is frequently impossible to perform adequate system maintenance on it.