Writing Requirements Properly

**2011 NDIA CMMI Conference
Denver, Colorado**

**Presented by
Al Florence
The MITRE Corporation**

Class Participation

Determine the problems with these 2 requirements:

- All computer-resident information that is sensitive shall have system access controls. Access controls shall be consistent with the information being protected and the computer system hosting the data.

The interval for propagating changes to suppliers shall be configurable.

Introduction

- A Government agency, while re-developing legacy systems, reversed engineered the existing requirements.

- The examples represent several legacy systems that are in the process of redevelopment in a modernization effort.

- The examples depict only the requirements effort – they do not reflect any other lifecycle activities: design, implementation, test or operation.

Introduction

- It needs to be noted that requirements do not "live alone"
  - They depend on other requirements and/or
  - on clarifying comments

    to present a complete view of the functionality associated with a related set of requirements.

- A related set of functional requirements may be introduced with a preamble describing the capability of the functional set.
  - The preamble does not itself establish requirements; this is done later in the requirements' specifications.

- Some requirements may be amplified with clarifying comments which are, again, not part of the requirements, but add understandability.

Introduction

- Some requirements are documented sequentially with the requirements stated first setting the "stage" for the following requirements which add more and more capability.

    - The later stated requirements depend on the earlier requirements to complete their functionally.

    - An example may be the use of the word "processing". If the processing of a functional set of related requirements has been described in earlier requirements the later requirements may amplify and/or reference the processing without having to restate the processing.

Critical Attributes

The following are some critical attributes that requirements must adhere to:

Completeness: **Requirements should be as complete as possible.**

(They should reflect system objectives and specify the relationship between the software and the rest of the subsystems.)

Traceability: **Each requirement must be traceable to some underlying source, such as a system-level requirement.**

(Each requirement should have a unique identifier so that the software design, code, and test plans can be precisely traced back to the requirement.)

Testability: **All requirements must be testable in order to demonstrate that the software end product satisfies its requirements.**

*(In order for requirements to be testable they must be specific, **unambiguous**, and quantitative whenever possible. Avoid vague, general statements.)*

Critical Attributes

Consistency: Requirements must be consistent with each other; no requirement should conflict with any other requirement.

(Requirements should be checked by examining all requirements in relation to each other for consistency and compatibility.)

Feasibility: Each requirement must represent a feasible representation.

(Requirements that have questionable feasibility should be analyzed during requirements analysis to prove their feasibility.)

Unique identification: Uniquely identifying each requirement is essential if requirements are to be traceable and testable.

(Uniqueness also helps in stating requirements in a clear and consistent fashion.)
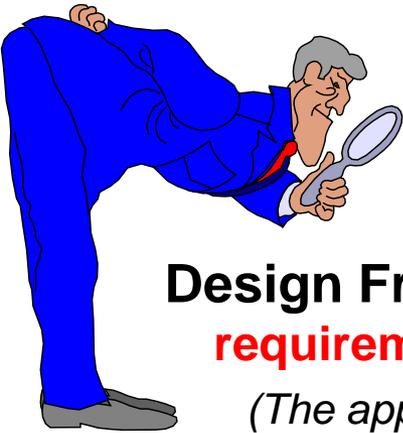
Critical Attributes

Design Free: **Software requirements should be specified at a requirements level not at a design level.**

(The approach should be to describe the software requirement functionally from a system point of view, not from a software design point-of-view, i.e. describe the system functions that the software must satisfy. A requirement reflects "what" the software shall accomplish while the design reflects "how" the requirement is implemented.)

Use of "shall" and related words: **In specifications, the use of the word "shall" indicates a binding provision.**

(Binding provisions must be implemented by users of specifications. To state non-binding provisions, use "should" or "may". Use "will" to express a declaration of purpose (e.g., "The Government will furnish..."), or to express future tense.[2)]

Ambiguity

- Requirements must be written in a clear, concise and unambiguous fashion

- Words and phrases that may have confusing and multiple interpretations must be avoided

  - Adequate
  - Ad hoc
  - All
  - Always
  - Appropriate
  - Clearly
  - Easy
  - Existing
  - Fast
  - Flexible
  - Future
  - If required
  - Immediately
  - Large
  - Light

  - Limited
  - Near real time
  - Periodic
  - Portable
  - Rapid
  - Several
  - Slow
  - Small
  - Sometimes
  - State of the art
  - Sufficient
  - Usable
  - User-friendly
  - Weight
  - When required

Also:
http://www.ppi-int.com/newsletter/SyEN-017.php#article

Examples

- With domain knowledge of the system, several teams reverse-

  engineered and defined requirements.
- They represented:
    - the users

    - the contractors

    - the acquisition organization

- This author was assigned as a consultant to guide the teams in the proper specification of requirements.

- The following examples show some of the requirements:
    - as initially specified by the teams

    - followed by this author's critique (against the critical attributes)

    - and as re-specified based on the critique

Example 1

Initial specification:

Software will not be loaded from unknown sources onto the system without first having the software tested and approved.

Critique:

- If it's tested and approved, can it be loaded from unknown sources?
- If the source is known, can it be loaded without being tested and approved?
- Requirement is ambiguous and stated as a negative requirement, which makes it difficult to implement and test.
- A unique identifier is not provided, which makes it difficult to trace.
- The word "shall" is missing.

Re-specification:

3.2.5.2 Software shall be loaded onto the operational system only after it has been tested and approved.

Example 2

Initial specification:

3.4.6.3 The system shall prevent processing of duplicate electronic files by checking a new SDATE record. An e-mail message shall be sent.

Critique: .

- Two "shalls" under one requirement number.
- Vague requirement: need to define the e-mail message.
- The requirement has design implications, SDATE record.
- A requirement should specify what the data in the record are and not the name of the record as it exists in the design and implementation..
- As specified it cannot be implemented or tested.

Re-specification:

3.4.6.3 The system shall:

      a. prevent processing of duplicate electronic files by checking the date and time of the submission, and

      b. send the following e-mail message:

            1. request updated submission date and time, if necessary, and

            2. the processing was successful, when successful.

12

Example 3

Initial specification:

3.2.5.7 The system shall process two new fields (provides production count balancing info to states) at the end-of-state record.

Critique:

- This requirement cannot be implemented or tested.

- It is incomplete. What are the two new fields?

- "Info" should be spelled out.

Re-specification:

3.2.5.7 The system shall provide the following data items (provides production count balancing information to states) at the end-of-state record:

    a. SDATE, and
    b. YR-TO-DATE-COUNT

13

Example 3

Re-Critique:

- This rewrite has design implications SDATE record and YR-TO-DATE-COUNT.

- From a requirements viewpoint it should specify what the data in the records are, not the name of the record as it exists in the design and implementation.

Re-Re-Specification:

3.2.5.7 The system shall provide the following data items (provides production count balancing information to states) at the end-of-state record:

a. submission date and time, and

b. year-to-date totals.

Example 4

Initial specification:

3.2.5.9 All computer-resident information that is sensitive shall have system access controls. Access controls shall be consistent with the information being protected and the computer system hosting the data.

Critique:

- Two "shalls" under one identifier.
- The requirement is vague and incomplete. Need to identify the sensitive information.
- What does "consistent" mean?
- As specified it cannot be implemented or tested.

Re-specification:

3.2.5.9 All sensitive computer-resident information shall have system access controls, consistent with the level of protection. (Reference *Sensitive Information,* Table 5.4.1 and *Level of Protection for Sensitive Information*, Table 5.4.2)

Example 5

Initial specification:

3.3.2.1 The system shall have no single point failures.

Critique:

- This is an ambiguous requirement. Needs identification of what components and/or functions the "no single point failures" applies to.

- As specified it cannot be implemented or tested.

Re-specification:

3.3.2.1 The *following* system components shall have no single point failures:

        a. host servers,
        b. networks,
        c. network routers,
        d. access servers,
        e. hubs,
        f. switches,
        g. firewalls, and
        h. storage devices.

Example 6

Initial specification:

3.2.7.1 The system shall purge state control records and files that are older than the operator or technical user-specified retention period.

Critique:

- Requirement is incomplete and vague without specifying the retention period or providing a reference as to where the information can be obtained.

- Requirement cannot be implemented or tested as stated.

Re-specification:

3.2.7.1 The system shall purge state control records and files that are older than the retention period input into the system by either the:

    a. operator, or
    b. technical user.

Example 7 1 OF 2

Initial specification:

3.2.6.3 The system shall receive and process state return data from the State Processing Subsystem. The system shall provide maintenance of the state data files and generate various reports.

Critique:

- Two "shalls" under one requirement number and multiple requirements in the specification.

- The word "process" in the first shall is vague. Need to define the processing required.

- The second "shall" does not provide for valid requirements; they cannot be implemented or tested as stated.
  - Needs identification of type/amount of maintenance required.
  - "various reports" is ambiguous.

Example 7 2 OF 2

Re-specification:

3.2.6.3 The system shall receive:

    a. production data that contains data from multiple states, and

    b. state total amount for one or more states,

extracted by the Returns Processing Subsystem.

3.2.6.4 The system shall parse multi-state data to respective state files.

3.2.6.5 The system shall display a summary screen reporting the results of processing for each state containing:

    a. state totals,

    b. state generic totals, and

    c. state unformatted totals.

Example 8

Initial specification:

3.2.7.1 The system shall not prevent the individuals from entering the year for which they intend the payment, but shall provide a check-point for them to ensure that they are not making a mistake in entering the correct year.

Critique:

- This is a negative requirement, negative requirements should not be specified. They cannot be implemented.

- A requirement should have all conditions that are required. If conditions are not required they will not be implemented.

- Two "shalls" under one requirement number.

- Suggest that this requirement be structured in a positive fashion.

Re-specification:

3.2.7.1 The system shall:

   a. allow individuals to enter the payment year, and

   b. provide a check-point to ensure that individuals enter the correct payment year.

Example 9 1 OF 2

Initial specification:

After the system receives the Validation file, the system shall:

- notify the individual about acceptance or rejection.
- the acceptance file must contain the name and ZIP code of the individual.
- rejected validation request must include the Reason Code.

Critique:

- The second and third bullets don't make sense, try to read them as such:
    - the system shall the acceptance file must...
    - the system shall rejected Validation…
- Use of both "shall" and "must".
- No unique identifier, use of bullets. Bullets cannot be traced.
- This requirement is ambiguous and cannot be implemented or tested.

Example 9 2 OF 2

Re-specification:

3.2.7.3 When the system receives a validation file, the system shall:

    a. reject the file if it does not contain the individuals:

        1. name, or

        2. ZIP code, and

    b. notify the individual about acceptance or rejection with a reason code. (Reference *Reason Code*, Table 5.4.8)

Example 10

Initial specification:

3.2.8.2 The enrollment process shall take from one to ten calendar days to complete for all payment types.

3.2.8.3 The enrollment process shall take no more than three days to complete for:
- a. credit payment, and/or
- b. note payment.

Critique:

These requirements are inconsistent and in conflict with each other.

Re-specification:

3.2.8.2 The enrollment process shall take:
- a. one to three calendar days to complete for:
  - 1. credit payment, and
  - 2. note payment, and
- b. one to ten calendar days to complete for all other payment types.

Example 11

Initial specification:

3.2.9.1 When doing calculations the software shall produce correct results.

Critique:

- Really? This is not a requirement.

- This type of requirements should not be specified!

- It should be deleted.

Re-specification:

Requirement deleted.

Summary

- The teams identified over 1000 requirements.
- The issues with their initial specification represented the entire spectrum of the following critical attributes:

  - completeness
  - traceability
  - testability
  - consistency

  - feasibility
  - unique identification
  - design free
  - use of shalls

- The teams were receptive to the critique, resolved issues and implemented the recommendations willingly.

- The requirements resulting from this effort were:
  - reviewed with senior management
  - accepted as specified
  - baselined, and
  - allocated to development teams for implementation.

Conclusion

- If sufficient time and proper effort is taken to validate requirements against critical attributes during their definition and specification, software projects will improve their probability of success considerably.

- If this is not done, projects pay the consequences during implementation, integration and test – not to mention during operation.

But you _knew_ that, didn't you?
(I hope!)

Contact Information

Al Florence

florence@mitre.org

703 395 8700 – Cell

303 955 2286 – Home