Writing Effective Requirements Specifications

Presenter: William M. Wilson

Track: Track 4

Day: Wednesday

Keywords: Requirements, Documentation, Specification Practices

Abstract

The Goddard Space Flight Center's (GSFC) Software Assurance Technology Center (SATC) has developed an early life cycle tool for assessing requirements that are specified in natural language. The Automated Requirements Measurement (ARM) tool was used to analyze more than 50 NASA System/Software Requirements Specification (SRS) documents. ARM reports were used to focus human analysis on specific aspects of the documentation practices exhibited by these documents. Several significant weaknesses were identified. This paper identifies the underlying problems that produce these deficiencies and recommends methods that can be used to prevent such problems.

1. Introduction

This paper is based on the results of a SATC study of NASA system/software requirements specification (SRS) documents written in natural language. Initial sections of this paper define the Software Assurance Technology Center's (SATC) organizational and mission context and its approach to the SRS study. Results of the study, the documentation deficiencies found and their causes are discussed in the body of this paper. The final section of this paper presents recommended documentation practices that will preclude the types of problems found by the SATC's SRS study.

Despite the significant advantages attributed to the use of formal specification languages, their use has not become common practice. Because the requirements that the acquirer expects the developer to contractually satisfy must be understood by both parties, specifications are most often written in natural language. The use of natural language to prescribe complex, dynamic systems has at least three severe problems: ambiguity, inaccuracy and inconsistency [11]. Defining a large, multi-dimensional capability within the limitations imposed by the two dimensional structure of a document can obscure the relationships between groups of requirements.

The importance of correctly documenting requirements has caused the software industry to produce a significant number of aids [3] to the creation and management of the requirements specification documents and individual specifications statements. Very few of these aids assist in evaluating the quality of the requirements document or the individual specification statements. This situation motivated the SATC to develop a tool to provide metrics that NASA project managers can use to assess the quality of their requirements specification documents and to identify risks that poorly specified requirements will introduce into their project. It must be emphasized that the tool does not attempt to assess the correctness of the requirements specified.

It assesses the structure of the requirements document and individual specification statements and the vocabulary used to state the requirements. The development of this tool has lead to an in-depth study of the documents used to evolve and test its capabilities. This study highlighted the most common deficiencies found in SRSs. These deficiencies were so pervasive that efforts to identify the underlying problems and potential solutions were initiated.

2. Background

Two of NASA's nine major field installations have primary responsibilities for unmanned space exploration missions. Goddard Space Flight Center's (GSFC) area of responsibility is near-Earth missions, while the Jet Propulsion Laboratory (JPL) is responsible for deep space missions. GSFC's missions are implemented by the Flight Projects Directorate. The Mission Operations and Data Systems Directorate (MODSD) provides GSFC flight projects with institutional communications and data handling support systems. The Office of Flight Assurance (OFA) supports GSFC's missions with systems review and project assurance management resources. OFA's Office of System Reliability and Safety (SR&S) supports GSFC's projects with expertise in the areas of safety, reliability and software assurance. SR&S also has the responsible for formulating GSFC's management policies and establishing minimum technical requirements in those three areas. The SATC aids SR&S in the execution of its software assurance responsibilities.

The SATC assists individual flight projects and MODSD organizations to adapt and use evolving software assurance technologies. These organizations also provide the data for most of the SATC's research activities, including the documents used in the SATC SRS study. The SATC's mission is to assist National Aeronautics and Space Administration (NASA) projects to improve the quality of software that they acquire or develop. The SATC's efforts are currently focused on the development and use of software metric methodologies and tools that identify and assess risks associated with performance and scheduled delivery. Software standards, guidebooks, tools and technical reports/papers produced by the SATC can be accessed at the Web address; http://satc.gsfc.nasa.gov/homepage.html.

2.1 System/Software Requirements Specification Study

System/Software Requirements Specifications SRSs are the first tangible product of most development life cycles and one of the major sources of problems in later phases. In order to better understand the source of these problems and seek a method for detecting them as early as possible, the SATC initiated a study of available NASA requirements specifications. The approach followed by the study effort was to (1) define the characteristics that should be exhibited by a SRS, (2) identify what could be measured, (3) develop indicators of quality, (4) develop a tool to perform the measurements, (5) analyze the SRS documents in light of the reports generated by the tool, (6) identify underlying problems and (7) develop recommended practices to preclude the problems.

2.1.1 Desirable SRS Quality Characteristics

The desirable characteristics for requirements specifications[2] [6] are:

- Complete
- Consistent

- Correct
- Modifiable
- Ranked
- Traceable
- Unambiguous
- Verifiable

As a practical matter, it is generally accepted that requirements specifications should also be Valid and Testable. These characteristics are not independent. For example, McCall's quality model [7] identifies tractability, completeness, and consistency as being factors which contribute to correctness. Also, the ISO 9126 sample quality model [7] gives stability, modifiable (changeability), and testability as factors contributing to maintainability. A specification, obviously, cannot be correct if it is incomplete or inconsistent.

a. Complete
A complete requirements specification must precisely define all the real world situations that will be encountered and the capability's responses to them [11]. It must not include situations that will not be encountered or unnecessary capability features.

b. Consistent
A consistent specification is one where there is no conflict between individual requirement statements that define the behavior of essential capabilities and specified behavioral properties and constraints do not have an adverse impact on that behavior [11]. Stated another way, capability functions and performance level must be compatible and the required quality features (reliability, safety, security, etc.) must not negate the capability's utility. For example, the only aircraft that is totally safe is one that cannot be started, contains no fuel or other liquids, and is securely tied down.

c. Correct
For a requirements specification to be correct it must accurately and precisely identify the individual conditions and limitations of all situations that the desired capability will encounter and it must also define the capability's proper response to those situations [11]. In other words, the specification must define the desired capability's real world operational environment, its interface to that environment and its interaction with that environment. It is the real world aspect of requirements that is the major source of difficulty in achieving specification correctness. The real world environment is not well known for new applications and for mature applications the real world keeps changing. The COBOL problem with the transition from the year 1999 to the year 2000 is an example of the real world moving beyond an application's specified requirements.

d. Modifiable
In order for requirements specifications be modifiable related concerns must be grouped together and unrelated concerns must be separated [6] [10]. This characteristic is exhibited by a logical structuring of the requirements document. Structuring the document to be modifiable may conflict the ranking individual specifications according to stability and/or importance.

e. Ranked
Ranking specification statements according to stability and/or importance is established in the requirements document's organization and structure [6]. The larger and more complex the

problem addressed by the requirements specification, the more difficult the task is to design a document that aids rather than inhibits understanding.

f. Testable
In order for a requirement specification to be testable it must be stated in such as manner that pass/fail or quantitative assessment criteria can be derived from the specification itself and/or referenced information [10]. Requiring that a system must be easy to use is subjective and therefore is not testable.

g. Traceable
Each requirement stated within the SRS document must be uniquely identified to achieve traceability [10]. Uniqueness is facilitated by the use of a consistent and logical scheme for assigning identification to each specification statement within the requirements document

h. Unambiguous
A statement of a requirement is unambiguous if it can only be interpreted one way [10]. This perhaps, is the most difficult attribute to achieve using natural language. The use of weak phrases or poor sentence structure will open the specification statement to misunderstandings.

i. Valid
To validate a requirements specification all the project participants, managers, engineers and customer representatives, must be able to understand, analyze and accept or approve it [11]. This is the primary reason that most specifications are expressed in natural language.

j. Verifiable
In order to be verifiable requirement specifications at one level of abstraction must be consistent with those at another level of abstraction [11]. Most, if not all, of these attributes are subjective and a conclusive assessment of the quality of a requirements specification requires review and analysis by technical and operational experts in the domain addressed by the requirements. These characteristics, however, can be linked to primitive indicators that provide some evidence that the desired attributes are present or absent. These primitives are alluded to in the attribute definitions below.

2.2 Indicators Of Strength And Weakness

Although most of the quality attributes of documented requirements are subjective, there are aspects of the documentation which can be measured that are indicators of attributes that are related to quality. Size, which is a primitive used in many quality metrics, can be directly measured. The size of a requirements document can be easily measured by the number of pages, the number of paragraphs, lines of text, or the number of individual specification statements it contains. The number of unique subjects addressed by specification statements within the requirements document can also be counted with relative ease. A count of unique specification subjects is an indication of the scope of the requirements encompassed by the document. The breadth and hierarchical depth encompassed by the document's specification statements can be measured using the document's internal identification scheme. These measures provide clues to the document's organization and depth of detail. The number of uniquely identified specification statements at each level of the document's structure can also be counted. These counts provide an indication of how the specification statements are organized and the level of detail to which

requirements are specified. It is also possible to count the occurrence of specific words and phrases that signal that a specification statements are weak or strong.

2.2.1 Categories Of Indicators

Nine categories of requirement document and specification statement quality indicators were established based on a representative set of NASA requirements documents selected from the SATC's library. Individual indicators were identified by finding frequently used words, phrases, structures and other aspects of the selected documents that were related to quality attributes and could be easily identified and counted by a computer program. These individual indicators were grouped according to their indicative characteristics. The resulting categories fall into two classes. Those related to the examination of individual specification statements and those related to the total requirements document. The categories related to individual specification statements are:

- Imperatives
- Continuances
- Directives
- Options
- Weak Phrases

The categories of indicators related to the entire requirements document are:

- Size
- Text Structure
- Specification Depth
- Readability

a. Imperatives
*Imperative**s*** are those words and phrases that command that something must be provided. The ARM tool lists imperatives and the number of times they were detected in the sequence that they are discussed below. This list presents imperatives in descending order of their strength as a forceful statement of a requirement. The NASA SRS documents that were judged to be the most explicit had the majority of their imperative counts associated with items near the top of this list.

- ***Shall*** is usually used to dictate the provision of a functional capability.
- ***Must*** or ***must not*** is most often used to establish performance requirements or constraints.
- ***Is required to*** is often used as an imperative in specifications statements written in the passive voice.
- ***Are applicable"*** is normally used to include, by reference, standards or other

documentation as an addition to the requirements being specified.

- ***Responsible for*** is frequently used as an imperative in requirements documents that are written for systems whose architectures are already defined. As an example, "*The XYS function of the ABC subsystem is responsible for responding to PDQ inputs."*

- ***Will*** is generally used to cite things that the operational or development environment are to provide to the capability being specified. For example, *"The building's electrical system will power the XYZ system."*

- ***Should*** is not frequently used as an imperative in requirement specification statements. However, when it is used, the specifications statement is always found to be very weak. For example, "*Within reason, data files should have the same time span to facilitate ease of use and data comparison*."

b. Continuances

Continuances are phrases such as those listed below that follow an imperative and introduce the specification of requirements at a lower level. The extent that continuances were used was found to be an indication that requirements were organized and structured. These characteristics contribute to the tractability and maintenance of the subject requirement specification. However, in some instances, extensive use of continuances was found to indicate the presence of very complex and detailed requirements specification statements. The continuances that the ARM tool looks for are listed below in the order most frequently found in NASA requirements documents.

- below:
- as follows:
- following:
- listed:
- in particular:
- support:

c. Directives

Directives is the category of words and phrases that point to illustrative information within the requirements document. The data and information pointed to by directives strengthens the document's specification statements and makes them more understandable. A high ratio of the total count for the Directives category to the documents total lines of text appears to be an indicator of how precisely requirements are specified within the document. The directives that the ARM tool counts are listed below in the order that they are most often encountered in NASA requirements specifications.

- figure
- table
- for example

- note

d. Options
Options is the category of words that give the developer latitude in satisfying the specification statements that contain them. This category of words loosen the specification, reduces the acquirer's control over the final product, and establishes a basis for possible cost and schedule risks. The words that the ARM tool identifies as options are listed in the order that they are most frequently used in NASA requirements documents.

- can
- may
- optionally

e. Weak Phrases
Weak Phrases is the category of clauses that are apt to cause uncertainty and leave room for multiple interpretations. Use of phrases such as "adequate" and "as appropriate" indicate that what is required is either defined elsewhere or, worse, that the requirement is open to subjective interpretation. Phrases such as "but not limited to" and "as a minimum" provide a basis for expanding a requirement or adding future requirements. The total number of weak phrases found in a document is an indication of the extent that the specification is ambiguous and incomplete. The weak phrases reported by the ARM tool are:

- adequate
- as a minimum
- as applicable
- easy
- as appropriate
- be able to
- be capable
- but not limited to
- capability of
- capability to
- effective
- if practical
- normal
- provide for

- timely
- tbd

f. Size

Size is the category used by the ARM tool to report three indicators of the size of the requirements specification document. They are the total number of:

- lines of text
- imperatives
- subjects of specification statements
- paragraphs

The number of lines of text in a specification document is accumulated as each string of text is read and processed by the ARM program. The number of subjects used in the specification document is a count of unique combinations and permutations of words immediately preceding imperatives in the source file. This count appears to be an indication of the scope of the document. The ratio of imperatives to subjects provides an indication of the level of detail being specified. The ratio of lines of text to imperatives provides an indication of how concise the document is in specifying the requirements.

g. Text Structure

Text Structure is a category used by the ARM tool to report the number of statement identifiers found at each hierarchical level of the requirements document. These counts provide an indication of the document's organization, consistency, and level of detail. The most detailed NASA documents were found to have statements with a hierarchical structure extending to nine levels of depth. High level requirements documents rarely had numbered statements below a structural depth of four. The text structure of documents judged to be well organized and having a consistent level of detail were found to have a pyramidal shape (few numbered statements at level 1 and each lower level having more numbered statements than the level above it). Documents that exhibited an hour-glass shaped text structure (many numbered statements at high levels, few at mid levels and many at lower levels) were usually those that contain a large amount of introductory and administrative information. Diamond shaped documents (a pyramid followed by decreasing statement counts at levels below the pyramid) indicated that subjects introduced at the higher levels were addressed at different levels of detail.

h. Specification Depth

Specification Depth is a category used by the ARM tool to report the number of imperatives found at each of the documents levels of text structure. These numbers also include the count of lower level list items that are introduced at a higher level by an imperative and followed by a continuance. This data is significant because it reflects the structure of the requirements statements as opposed to that of the document's text. Differences between the Text Structure counts and the Specification Depth were found to be an indication of the amount and location of text describing the environment that was included in the requirements document. The ratio of the total for specification depth category to document's total lines of text appears to be an indication of how concise the document is in specifying requirements.

i. Readability Statistics

Readability Statistics are a category of indicators that measure how easily an adult can read and understand the requirements document. Four readability statistics produced by Microsoft Word are currently calculated and compared:

- Flesch Reading Ease index is based on the average number of syllables per word and the average number of words per sentence. Scores range from 0 to 100 with standard writing averaging 60 - 70. The higher the score, the greater the number of people who can readily understand the document.
- Flesch-Kincaid Grade Level index is also based on the average number of syllables per word and the average number of words per sentence. The score in this case indicates a grade-school level. A score of 8.0 for example, means that an eighth grader would understand the document. Standard writing averages seventh to eighth grade.
- Coleman-Liau Grade Level index uses word length in characters and sentence length in words to determine grade level.
- Bormuth Grade Level index also uses word length in characters and sentence length in words to determine a grade level.

2.3 ARM Tool

The ARM tool scans a file designated by the user as the text file that contains the requirements specifications. During the scanning process, the ARM tool searches each line of text for specific words and phrases. These search arguments (specific words and phrases) are considered to be indicators of the document's quality as a specification of requirements.

At the conclusion of the ARM's processing, the user is given the option of printing the report file. This report file contains a summary report, a detailed imperative report and a detailed weak phrase report. It contains each line of text from the source file found to contain an imperative or a weak phrase and each of these lines of text is proceeded by a line of identification information.

2.3.1 ARM Analysis Process

The ARM tool looks at statement numbers/identifiers (3.1.5; a.,b.,c.) to analyze the structural level (how deep in the document) of the text. If an imperative (shall, must, etc.) is present, the line of text is considered to be a specification statement. The words immediately prior to the imperative are considered to be the subject of the specification. Unique subjects are counted. Specification statements are searched to identify continuances (and, :) and weak phrases (capability to, at a minimum).

In the example statement below, the ARM tool would record one line of text at level 3, one requirement at level 3 containing two continuances and two weak phrases, and three requirements at level 4 (a requirement at level 3 with a ":" continuance followed by 3 identified [a., b., c.] lines of text equals 3 requirements at level 4.

5.2.7 The XYZ system shall have the capability to generate reports showing detailed and summary information about the maintenance schedule for system hardware, system software,

and scientific software, including, at a minimum:
a. Routine maintenance schedules
b. Non-routine maintenance schedules
c. Upgrade maintenance schedule

2.3.2 Arm Analysis Of Document Set

The ARM tool was used to subject each specification file to a full text scan for occurrences of each of the quality primitives. The occurrence of primitives within each file were totaled and reported individually and by category. Correlation between all totals were examined for significant relationships. Files that exhibited anomalous data and off-norm counts were examined to determine the source of these aberrations. A tentative assessments of each requirements document's quality was made based on this analysis and examinations. Those requirements documents judged to be of poor quality were reviewed in detail to determine the source of their weakness.

3. What We Found

The SRS deficiencies found by the SATC study are the result of basic structural and language problems. These problems are common to all technical documents, especially those written in natural language. Structural problem arises due to the difficulty of organizing information that has detailed, complex, and multi-dimensional inter-relationships into a sequential media.

All languages have three problems: ambiguity, inaccuracy and inconsistency. The extent of these problems depends upon the degree of the language's lack of formality. Unfortunately, formal languages are difficult to learn and expensive to apply correctly. Most system users are not fluent in a formal language. This makes it extremely difficult, if not impossible, to validate requirements that are written in a formal language. Natural language, on the other hand, is extremely informal. Although natural languages' inherent problems are quite severe it does have significant advantages. The extensive vocabulary of natural language and common understanding of its rules of syntax facilitates communication among the projects' participants. The informality of the language and its extensive vocabulary also makes it relatively easy to specify high level general requirements when precise details are not yet known.

An examination of the NASA SRS documents judged to be of poor quality revealed the following deficiencies:

In several instances, documentation and style standards were not used. The documents appeared to be written using an ad hoc outline. In other cases documentation and style standards were misapplied. It was obvious that the standard had been used to drive the analysis, with no attempt to tailor the content to the real problem at hand. Some sections of the document were filled with meaningless information simply because the section was included in the standard and apparently the author did not want to leave it blank.

Information content was poorly and inconsistently organized. Descriptions of the project, the operational environment, and the requirements for the needed capability were intermingled. Inconsistent structuring of this information and the undisciplined use of "will", "shall", and "should" further blurred the distinction between "what is", "what will be", and "what must be".

Levels of detail varied greatly. It appeared that certain requirements were well understood and were dwelt upon at the neglect of other items. Again this implied that the SRS standard was being used in a "fill in the blanks" manner rather than as the repository of the results of a adequate analysis of requirements.

Document sections, paragraphs, and requirement statements were inconsistently identified.

The identification schemes most frequently encountered are shown below in the order of their prevalence.

- Hierarchical Numbers - 1.2.3., 1.2.3.4.., ...1.2.3.4.5.6.7.8., etc.
- Lettered Hierarchical Numbers P1.2.3., Q1.2.3.4.., .S1.2.3.4.5.6.7.8, etc.
- Integer Numbers - 1., 2.,3.,...10.; 1, 2, 3, 20, 21; 30; [1], [2], [3];[20]
- Letters A., B., C.; a., b., c.; a), b), c); (a), (b), (c)

Although most documents uses an identification scheme, in quite a few instances they were not consistently applied. This was particularly true in those document that were inconsistent in their level of detail. At the lower levels of specification, lettering and numbering of items was often done without regard for what had been used at the next higher level of the document.

Excessive wordiness was primarily found in the largest documents. In general these were the documents that attempted to combine concept and requirements. The authors appeared to resort to overblown descriptions in an attempt to make the document interesting or to hide the fact that there was a scarcity of hard information.

Individual requirements statements were poorly structured and poorly worded. Some statements were too lengthy, containing too many compound clauses and used imprecise language.

An examples of such a statement is:

"Users attempting to access the ABC database shall be reminded by a system message that must be acknowledged and page headings on all reports that the data is sensitive and access is limited by their system privileges."

4. Significance Of Structure And Language

To understand the structural and language problems and to identify an approach to preventing their occurrence, it was necessary to look at the general requirements that a requirements specification must satisfy. A SRS's quality characteristics are impacted by the document's structure, the structure of the individual specification statements and language used to express the specification statements. The organizational structure of the document can either enhance or undermine overall comprehension. If the document's structure is not properly tailored to the objective and purposes of the system solution that is being specified it will result in sections that lack cohesion and, similar to a poorly designed software capability, widely separated paragraphs will be tightly coupled..

5. What We Should Have Found

The documents should have exhibited structures that accommodated the SRS's objective, supported its purposes and encompassed those topics that sound engineering practice has found to be common to all technical specifications.

5.1 Requirement Specification Topics

The IEEE and other standards issuing organizations have identified nine topics that need to be addressed when specifying system or software requirements. They are:

1. Interfaces
2. Functional Capabilities
3. Performance Levels
4. Data Structures/Elements
5. Safety
6. Reliability
7. Security/Privacy
8. Quality
9. Constraints & limitations

The first four of these topics address engineering requirements associated with the individual elements of the needed capability. The last five topics address quality requirements that encompass all elements of needed capability

These nine topics are not isolated subjects and information is coupled at various levels. For example, functions, interfaces and data are closely linked. The issue is how best to discuss these topics so that the relationships are stated clearly but without too much redundancy. Should the discussion of a function include discussion of its interfaces and associated data, or should sections of the document where these topics are addressed be pointed to within the functional requirement? If the functional requirement includes discussion of topics addressed in detail elsewhere, this coupling creates problem in maintaining the document. If the other topics are only cited by reference, reading and understanding becomes cumbersome due to the necessity of finding and reading the referenced information out of sequence

5.2 SRS Objectives

The objective of the SRS is to define a capability that will satisfies a mission need or problem. In order to do this, the requirements phase of the project usually consists of two overlapping activities. The initial activity of the requirements phase is to study strategic enterprise and tactical environment in sufficient depth to enable the mission need to be defined for further analysis. Once the environment and the need/problem are sufficiently understood, they are analyzed in order to postulate a solution to the need/problem. The solution is documented in

terms of those aspects of the environment that impinge on the solution and requirements that must be satisfied in order to achieve a satisfactory solution.

The scope of the SRS and the amount of detail it must contain is a function of the first hand knowledge of the environment and the need/problem possessed by project participants or the availability of other documentation that accurately describes the situation.

5.3 SRS Purpose

The SRS's general purpose is to provide a medium for capturing and communication to all interested parties what is needed. Its most significant purpose is to serve as a contract between the acquirer and the capability provider by defining what is to be provided and in many instances the manner in which it is to be produced and the technology that it incorporates. In addition, the SRS provides a basis for most of the project's management and engineering functions such as assessing proposed engineering changes, resolving acquirer/provider disputes, developing test requirements, writing the preliminary user's manual, planning maintenance support activities and implementing operational enhancements.

In order for it to serve these purposes, the SRS must provide a definition of the future operational and support environments in which the required capability will exist as well as providing a prescription for the system intended as the solution to the mission need/problem. If necessary the SRS must also identify the methodologies and technologies that are to be used during the development life cycle.

These purposes cannot be achieved unless the form and content of the SRS enables it to be understood by all the projects participants. In most instances these participants include the acquirer, the provider, operational staff, support personnel, and users. Since these participants have different perspectives and interests of varying scope and depth, the SRS must be a very versatile document.

6. The Structural Problem

In order to be effective, the SRS must exactly describe the strategic and tactical environments and the system solution within those environments. The descriptions of these environments must address elements of the operational and support situations that will impinge on the solution as they are expected to exist during the system solution' operational life time. It is necessary to distinguish between those aspects of the environments (strategic, tactical, operational and support) that are continuations of the existing situation and those that are expected to be delivered in the future by the provider or by other sources. In addition, the SRS must address all of nine fundamental topics that are relevant for each of the environments.

The design of the SRS document should be based on a predetermined approach to structuring the specifications' information into sections and subparagraphs. Information should never be arbitrarily grouped together. This not only makes the document difficult to understand, it also makes it difficult to maintain. Statements that are part of a single function are normally grouped together.. Functions that are connected in series by output to input relationships should appear in the document in that same sequence if possible. Functions that share common inputs and outputs should be addressed within the same area of the document. If several processes must be

accomplished in the same time frame their specifications must be tied together by the document to make this commonality clear. Functions that are similar need to be distinguished from one another but the similarities also needs to be emphasized.

Data Item Descriptions (DID) that prescribe the SRS's outline and content are attempts to solve the structural problem for a general class of documents. As with any general solution, these DIDs only resolve issues at the highest level of organization. The DID that is to govern a particular SRS must be tailored to fit the type of project that is to use it and the category of system/software that is to be developed. This tailoring is a design activity which is not insignificant in impact or effort. If we look at DoD's and NASA's SRS DIDs in light of the topics that must be included in a requirements specification the structural problems come into focus.

6.1 Documentation Standards

The DoD Software Requirements Specification (SRS) DID, DI-IPSC-81433, (Figure 1.) specifies that information is to be provided within six major sections. Section 3 of the DID prescribes the structure for defining the technical requirements of the subject Computer Software Configuration Item (CSCI). Sections 1. and 2. are concerned with establishing the system context for the CSCI to be specified. Section 4 is to define the methods that will be used to ensure that the requirements specified in Section 3. have been satisfied. Section 5. is to trace each requirement specified in Section 3. back to its source in the system or subsystem level requirements document. Section 6. is a repository for any general information necessary to understand the requirements specified in Section 3.

SOFTWARE REQUIREMENT SPECIFICATION- DI-IPSC-81433

1. Scope
2. Reference Documents
3. Requirements
    - 3.1 Required states and modes
    - 3.2 CSCI capability requirements
    - 3.3 CSCI external interface requirements
    - 3.4 CSCI internal interface requirements
    - 3.5 CSCI internal data requirements
    - 3.6 Adaptation requirements
    - 3.7 Safety requirements
    - 3.8 Security & privacy requirements
    - 3.9 CSCI environment requirements
    - 3.10 Computer resource requirements
    - 3.11 Software quality factors
    - 3.12 Design and Implementation constraints
    - 3.13 Personnel-related requirements
    - 3.14 Training-related requirements
    - 3.15 Logistics-related requirements
    - 3.16 Other requirements
    - 3.17 Packaging requirements
    - 3.18 Precedence and criticality of requirements
4. Qualification Provisions
5. Requirements Traceability
6. Notes

A. Appendixes

Figure 1. Software Requirements Specification , DI-IPSCS-81433

Section 3 is organized into 18 subsections. Subsections 3.9 and 3.10 are respectively dedicated to requirements related to the operational *environment* and computer *resources*. Personnel, training, *logistics*, packaging, and other requirements related to the CSCI are to be presented in Subsections 3.13 through 3.17. Subsection 3.18 is dedicated to *ranking* the requirements specified in other subsections according to precedence and criticality.

External and internal **Interfaces** requirements are to be specified respectively in Subsections 3.3 and 3.4. Capabilities (I.e. **functions**) are to be specified in separate Subparagraphs of Subsection 3.2 **Performance levels** are to be specified within the Subparagraph of Subsection 3.2 that specifies the related function. Internal **data** requirements are to be specified in Subsection 3.5. **Safety** requirements are to be specified in Subsection 3.7. **Security/Privacy** requirements are to be specified in Subsection 3.8. **Quality** factors, including **reliability** are to be specified in Subsection 3.11. Design and implementation **constraints** are to be specified in Subsection 3.12.

REQUIREMENTS - NASA DID-P200

1.0 Introduction
2.0 Related documentation
3.0 Requirements approach and tradeoffs
4.0 External interface requirements
5.0 Requirements specification
- 5.1 Process and data requirements
- 5.2 Performance and quality engineering requirements
- 5.3 Safety requirements
- 5.4 Security and privacy requirements
- 5.5 Implementation constraints
- 5.6 Site adaptation
- 5.7 Design goals

6.0 Traceability to parent's design
7.0 Partitioning for phased delivery
8.0 Abbreviations and acronyms
9.0 Glossary
10.0 Notes
11.0 Appendices

Figure 2. Requirements Specification, NASA DID-P200

NASA DID-P200 (Figure 2.) specifies that information is to be provided in eleven numbered sections. Information in Sections 1. and 2. is to provide the context for the capability being specified and the position of SRS within the project's documentation. Section 3. is to provide a description of the method(s) used to establish and analyze the requirements specified within the document. Section 4. provides for the specification of external interfaces. Section 5. prescribes the structure for defining the technical requirements of the subject software product. Section 6. is to trace each requirement specified in Section 5. back to its source in the system or subsystem

level requirements document. In Section 7 the requirements specified in Section 5. that are to satisfied by each phased delivery are to be identified. Sections 8. through 11. are repositories for acronym, terms, notes and appendices.

Section 5. is divided into seven subparagraphs. Process (**functional**) and **data** requirements are to be specified in Subparagraph 5.1. **Performance** and **quality** requirements are to be specified in Subparagraph 5.2. **Safety** requirements are to be specified in Subparagraph 5.3. Requirements for adaptation of the software to the physical site are to be specified in Subparagraph 5.6. Design goals, including **reliability**, are to be specified in Subparagraph 5.7. **Security and privacy** requirements are to be specified in Subparagraph 5.4. Implementation **constraints** are to be specified in Subparagraph 5.5

6.3 Statement Structuring

Poorly structured individual requirement statements will result in confusing specifications that are prone to incorrect interpretations. The following is such a statement:

The XYZ system shall provide variance/comparative information that is timely, itemized in sufficient detail so that important individual variances are not hidden because they cancel each other, pinpoints the source of each variance, and indicates the area of investigation that will maximize overall benefits."

This specification can more easily be understood if structured as follows:

"5.1 The XYZ system shall provide variance/comparative information.

5.1.1 Variance/comparative information shall be timely.

5.1.2 Variance/comparative shall be itemized in sufficient detail to:

5.1.2.1 Prevent important individual variances from being hidden.

5.1.2.2 Pinpoints the source of each variance

5.1.2.3 Indicate the area of investigation that will maximize overall benefits."

Each specification statement consists of four basic structural elements; Entities, Actions, Events, and Conditions. These elements can be used or modified by various cases such as:

1. Owner
2. Actor
3. Target
4. Constraint
5. Owned
6. Action
7. Object

8. Localization

The recommended model for a specification's structure is:

[Localization] [Actor|Owner] [Action] [Target|Owned] [Constraint]

For example: *"7.1 When three or more star trackers lose reference stars, the spacecraft shall immediately align its main axis on the Earth-Sun line unless the optical instrument's cover is not closed."*

Localization: "When three or more star trackers lose reference stars"
Actor|Owner: "spacecraft"
Action: "align"
Target|Owned: "main axis"
Constraint: "unless the optical instrument's cover is not closed"

If a specification statement contains more than 3 punctuation marks it probably can be restructured to make its meaning better understood.

7. The Language Problem

Attention must be given to the role of each word and phrase when formulating the specification statement. Words and phrases that are carelessly selected and/or carelessly placed produce specifications that are ambiguous and imprecise. Examples of specifications containing poor word selection are shown below:

*"The system shall be user **friendly**."*, can be subjectively interpreted and its implementation will be difficult to test objectively.

"*Users attempting to access the ABC database **should** be reminded by a system message that will be acknowledged and page headings on all reports that the data is sensitive and access is limited by their system privileges."*

8. Recommendations

The effectiveness of expressing requirements specifications with natural language can be greatly improved through relatively simple and readily available methods. Project managers must ensure that standards for requirements specifications and style are established at the outset of the project and that all project participants are trained in the use of those standards. It is also recommended that the technical nature and intended use of the SRS document be emphasized to encourage a straightforward writing style and simple sentence structures to specify each requirement. Technical documents are not required to be interesting, however, it is necessary for them to effectively communicate with the reader. In addition to structuring the document and individual specifications as discussed in Section 6 above, some of guidelines for producing a SRS that communicates are:

8.1 Words And Phrases

8.1.1 Use the most simple words appropriate to the intent of the statement. Hide is defined as "to put out of sight". Obscure is defined as "lacking light or dim". Don't use obscure if you mean hide.

8.1.2 Use imperatives correctly and be consistent. Remember, shall "prescribes", will "describes", must & must not "constrain", and should "suggests"

8.1.3 Avoid weak phrases such as "as a minimum", "be able to", "capable of", and "not limited to".

8.1.4 Do not use words or terms that give the provider an option on the extent that the requirement is to be satisfied such as "may", "if required", "as appropriate", or "if practical".

8.1.5 Do not use generalities where numbers are really required such as "large", "rapid", "many", "timely", "most", or "close".

8.1.6. Avoid fuzzy words that have relative meanings such as "easy", "normal", "adequate", or "effective".

8.2 Providing Examples

8.2.1 Immediately follow what is to be illustrated with the example.

8.2.2 Repeat an example if it is not already located on the same page. Its better to be repetitive than to divert the reader's attention.

8.2.3 Ensure that the example is not mistaken as part of the specification by the use of italics, quotes, or being explicit. *For example: "This is an example."*

8.3 Citing References

8.3.1 Identify all external documents in the section of the SRS designated for that purpose by the appropriate DID. For DI-IPSC-81433 and NASA DID-P200 its Section 2.

8.3.2 Identify each reference cited with a unique number or identifier, such as "2.14".

8.3.3 Cite references by short or common title, full title, version or release designator, date, publisher or source, and document number or other unique document identifier. For example: "2.14 NASA Software Management, Assurance, and Engineering Policy, NMI 2410.10, March 26, 1991."

1. Use the unique citation identifier when referencing information in the cited document. For example: " as defined by Section 3 of reference document 2.14."

8.4 Using Tables And Charts

1. Title and Identify each table and chart by a unique identifier.

8.4.2 List each table and chart in the RSR's table of contents by title, unique identifier and page number. Help the reader find it!

8.4.3 Identify the purpose of the table or chart in the text immediately preceding it. No surprises!

8.4.4 Explain each aspect or element of the table or chart (columns, rows, symbols, blanks, etc.) from right to left then top to bottom. No puzzles!

8.5 Summary & Conclusion

When using natural language to specify requirements several things must always be kept in mind.

8.5.1 The SRS is the media for expressing requirements not an outline for a methodology to derive the requirements.

8.5.2 The SRS is an item of software and as such should be an engineered product.

8.5.3 There are requirements that the SRS must satisfy. Be sure they are identified and used to decide on the details of the SRS structure.

8.5.4 Use proper sentence structures and select words and phrases based on their formal definitions, not what the popular culture thinks they mean.

And most of all, specification writers should be taught how to write simple direct statements and how to faithfully follow a disciplined approach to creating the SRS document. The SRS does not have to be interesting, but it must be understood. Statements such as: "The Romans, the Persians shall defeat." are acceptable from the Delphic Oracle but not from an software engineer.

References

[1] Brooks, Frederick P. Jr., No Silver Bullet: Essence and accidents of software engineering, IEEE Computer, vol. 15, no. 1, April 1987, pp. 10-18.

[2] DOD MIL-STD-490A, Specification Practices, June 4, 1985.

[3] Ganska, Ralph, Grotzky, John, Rubinstein, Jack, Van Buren, Jim, Requirements Engineering and Design Technology Report, Software Technology Support Center, Hill Air Fource Base, October, 1995.

[4] Gause, Donald C., and Weinberg, Gerald M., Exploring Requirements Quality Before Design, Dorset House Publishing, NY, NY, 1989

[5] IEEE Std 729-1983, Standard Glossary of Software Engineering Terminology, Feburary 18, 1983.

[6] IEEE Std 830-1993, Recommended Practice for Software Requirements Specifications, December 2, 1993.

[7] Kitchenham, Barbara, Pfleeger, Shari Lawrence, Software Quality: The Elusive Target, IEEE Software, Vol. 13, No. 1, January 1996, pp. 12-21.`

[8] NASA-STD-2100-91, NASA Software Documentation Standard, NASA Headquarters Software Engineering Program, July 29, 1991.

[9] Porter, Adam A., Votta, Lawrence G., Jr., and Basili, Victor R., Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment, IEEE Transactions on Software Engineering, Vol. 21, No. 6, June 1995, pp. 563-574.

[10] Sommerville, Ian, Software Engineering, Fourth Edition, Addison-Wesley Publishing Company, Wokingham, England, 1992.

[11] Stokes, David Alan, Requirements Analysis, Computer Weekly Software Engineer's Reference Book, 1991, pp. 16/3-16/21.

Writing Effective Requirement Specifications was presented at the Software Technology Conference, Utah, April 1997.