Writing Effective Natural Language Requirements Specifications

William M. Wilson

This article details writing practices that will produce a stronger requirements specification document by avoiding three common documentation problems. Examples of these problems and the recommended solutions presented in this article were derived by analyzing 40 approved NASA requirements specification documents.

The system or software requirements specification (SRS) document is the first definitive representation of the capability that the provider is to deliver to the user or acquirer. The SRS document becomes the basis for all a project's subsequent management, engineering, and assurance activities. As such, it is a strong source of potential risks that could adversely impact the project's resources, schedules, and products. Because of the criticality of the SRS, it is important to prevent or correct shortcomings in both the form and content of the SRS document before it is established as a project baseline.

A study conducted by the Software Assurance Technology Center (SATC) located at NASA's Goddard Space Flight Center (GSFC) found that 40 approved SRS documents contained many instances of three common weaknesses. Deficiencies were found in

- the organization of requirement information.
- the structure of individual requirement statements.
- the language used to express requirements.

These defects can be prevented through a more disciplined and consistent approach to document design, formulation of specification statements, and selection of key words and phrases.

Organizing Requirements

To develop, deliver, and install system or software capability that successfully satisfies the expectations and needs of the user, the provider of the capability must have access to a wide variety of information. In addition to the requirements that prescribe a solution to the user's needs, descriptions of the user's current and future operational environments and a definition of the transition from one environment to the other must be provided. In some instances, support considerations make it necessary to dictate restrictions on the development environment and limit the technology content of the delivered capability.

Detail and Consistency

The problem of organizing the requirements information is compounded by the need for specific topics to be addressed in detail and in a manner that enhances comprehension and minimizes redundancy. The Institute of Electrical and Electronics Engineers (IEEE) [1] recommends that an SRS address each of the following topics.

- Interfaces.
- Functional capabilities.
- Performance levels.
- Data structures and elements.
- Safety.
- Reliability.
- Security and privacy.
- Quality.
- Constraints and limitations.

The first four topics address engineering requirements associated with individual elements of the needed capability. The last five topics address quality requirements that cross all aspects of the needed capability. These topics are not isolated subjects; they are coupled in various ways. Functions, interfaces, and data are closely linked. The question may arise: Should the section of the SRS that prescribes the requirements for a particular function include interfaces and data specifications, or should it point to sections of the SRS devoted to these topics? The first alternative distributes similar information across several sections of the document and creates problems in maintaining the document. The second alternative breaks the reader's train of thought when the referenced information is needed. The basic structural issue is how to organize these topics so that relationships and necessary detail can be stated clearly and succinctly.

Data Item Descriptions (DIDs) refine the SRS's design in most documentation schemes and are used to establish generic design solutions for each type of document. As with most general solutions, DIDs only resolve issues at the highest level of organization. Documentation standards developed by both the Department of Defense (DoD) [2] and NASA [3] include several DIDs to specify systems at various levels and from different aspects. The scope and number of DIDs included in these standards are intended to facilitate the documentation of large programs and projects. For smaller projects, these schemes are burdensome. Smaller projects tend to use fewer documents to decrease costs and facilitate information control.

Both DoD and NASA documentation standards provide for adaptation of DIDs to meet the needs of a particular project: DoD provides specific guidance for tailoring its DIDs, NASA provides guidance for tailoring and "rolling up" the concept, requirements, design, and other documents into a single volume.

Tailoring DIDs is a design activity. Its potential impact on a project's success is significant and should be undertaken with the same importance given to the design of any engineering product. The final design of the SRS must be a structure of sections and subparagraphs that encompass and address the concerns of all project participants and organizational stakeholders. The SRS structure must facilitate everyone's understanding of the totality of required capability, the particular attributes of a capability, and how their areas of responsibility will be affected by the capability. Sections of the documents must be constructed in light of the cohesiveness, coupling, complexity, and consistency necessary to achieve a balance between comprehensibility and completeness.

Information should never be arbitrarily grouped together—it makes the document difficult to understand and to maintain. Descriptions of the conditions and situations that the required capability will encounter must be located with the prescription of its required response; however, the description and prescription must be kept distinct from one another. Requirements that are parts of a single functional capability must be grouped together, e.g., functions that are connected in series by output-to-input relationships should appear in the SRS in the same sequence, if possible. Functions that share common inputs and outputs should be addressed within the same section of the SRS. If several processes must be accomplished in the same time frame, their specifications must be tied together by the document's structure to make this commonality clear. Similar functions need to be distinguished from one another but the similarities also need to be emphasized. Most of these restrictions can be satisfied by combining a requirements identification scheme that consistently uses similar numbers to number similar things and by using a writing style that uses short declarative sentences.

Both the DoD SRS DID, DPSC-8-1433, [2] and NASA-DID-P200, [4] provide an excellent starting point for the organization of a requirements document. Most structures provide for most environmental subjects as well as the nine essential topics of requirements information. It is highly desirable to have the same topics identified with the same number in related documents such as the Operational Concept Description, System/Subsystem Specification, SRS, and Software Product Specification; therefore, care must be taken to ensure that the integrity of paragraph numbering is maintained when the structure of the DID is shortened or extended. If not, any correlation of like information across the set of documents will be lost. Both NASA and DoD documentation tailoring instructions address this problem; the crux of their tailoring instructions is to not change subject numbers.

Unnecessary sections should be "stubbed" with an "N/A" as close to the main section of the document as possible. Stubbing retains the topic's title and identification number. Cutting sections removes the topic and reassigns its identification number to the following topic. This would apply through the rest of the document's numbers and destroy the document's congruence with the rest of the document set. New topics should extend the numbering established by the DID by adding a node (section) to the document tree that is appropriate to the new subject. They should be inserted as the last branch or leaf (at the end) and given the node's next available sequential number.

Statement Structuring

Poorly structured specification statements result in confusing requirements that are prone to incorrect interpretations. If a specification statement contains three or more punctuation marks, it probably needs to be restructured.

An example of a specification that is a prime candidate for restructuring follows:

> 3.1 The XYZ system shall provide variance/comparative information that is timely, itemized in sufficient detail so that important individual variances are not obscured by other variances, pinpoints the source of each variance, and indicates the area of investigation that will maximize overall benefits.

It is much easier to read when structured as follows:

3.1 The XYZ system shall provide variance/comparative information.

3.1.1 Variance/comparative information shall be timely.

3.1.2 Variance/comparative information shall be itemized in sufficient detail to do the following:

3.1.2.1 Prevent important individual variances from being obscured by other variances.

3.1.2.2 Pinpoint the source of each variance.

3.1.2.3 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 the following:

- Owner.
- Actor.
- Target.
- Constraint.
- Owned.
- Action.
- Object.
- Localization.

The recommended model for a specification statement's structure is as follows:

- Localization.
- Actor/Owner.
- Action.
- Target/Owner.
- Constraint.

For example, "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*.

Problems with Natural Language

Natural language's extensive vocabulary and commonly understood syntax facilitate communication and make it an inviting choice to express requirements. The informality of the language also makes it relatively easy to specify high-level general requirements when precise details are not yet known. However, because of differences among formal, colloquial, and popular definitions of words and phrases and the effort required to produce detailed information, these same attributes also contribute to documentation problems. The use of natural language to prescribe complex, dynamic systems has at least three common and severe problems: ambiguity, inaccuracy, and inconsistency [5].

The precise meaning of many words and phrases depends entirely on the context in which they are used. For example, Webster's New World Dictionary identifies three variations in meaning for the word "align," 17 for "measure," and four for "delete." Even though the words "error," "fault," and "failure" have been precisely defined by the IEEE [5], they frequently are used incorrectly in specifications.

Attention must be given to the role of each word and phrase when formulating specification statements. Words and phrases that are carelessly selected or carelessly placed produce statements that are ambiguous and imprecise.

The most simple word that is appropriate to its intended purpose in the specification is the one to use. The word "hide" is defined as "to put out of sight." The word "obscure" is defined as "lacking light or dim." Do not use *obscure* if you mean *hide*—the rules for the game "obscure and seek" are not well known.

Use the correct imperative and use it consistently. Remember that the word "shall" prescribes, "will" describes, "must" and "must not" constrain, and "should" suggests. Avoid weak phrases such as "as a minimum," "be able to," "capable of," and "not limited to." These phrases are subject to different interpretations and also set the stage for future changes to the requirements.

Do not use words or terms that give the provider an option regarding the extent to which the requirement is to be satisfied, such as "may," "if required," "as appropriate," or "if practical." Do not use generalities when numbers are required, for example, "large," "rapid," "many," "timely," "most," and "close." Avoid imprecise words that have relative meanings such as "easy," "normal," "adequate," or "effective."

The use of imprecise terms usually indicates that the specifications author was either lazy, incompetent, or did not have sufficient time to determine the exact requirements. Some writers seem to be afraid that their audience will be bored or will think them lazy if they use simple words and repeat themselves. When writing documents or software, being too fancy complicates things and make the resulting products hard to understand.

The previously given example specification could be further strengthened through a better selection of words and phrases.

3.1 The XYZ system shall provide variance/comparative information.
  3.1.1 Variance/comparative information shall be provided at the end of every reporting cycle.
  3.1.2 Variance/comparative information shall include the data necessary to
    3.1.2.1 Prevent important individual variances from being hidden by other variances.
    3.1.2.2 Pinpoint the source of each variance.
    3.1.2.3 Determine the frequency and severity of each variance.

Serious problems will always result from specifications statements such as "The system shall be user friendly and provide adequate resources to meet the user's operational needs." What is considered to be "user friendly," "adequate resources," and "user's operational needs" must be defined in detail through specification or by reference to an existing system that has the required characteristics.

Natural language often entices authors to write "stream of consciousness" specification statements that are difficult to understand, for example, "Users attempting to access the ABC database should be reminded by a system message that will be acknowledged and on page headings on all reports that the data is sensitive, and access is limited by their system privileges."

Restructured as shown below, this requirement, although longer, is easier to comprehend.

4.4 The system shall notify users attempting to access the ABC database that
  a. The ABC data is classified as "sensitive."
  b. Access to the ABC data is limited to that allowed by the user's system privileges.
  c. Page headings on all reports generated using the ABC database must state that the report contains "sensitive" information.
  4.4.1 The system shall require the user to acknowledge the notification before being allowed to access the ABC database.

Summary and Conclusion

When natural language is used to specify requirements, several things must be kept in mind.

- The SRS is the medium to express requirements that have been identified and defined. The SRS's DID is not an outline for a method to derive requirements.
- The SRS is a software item and as such should be a product that is engineered to satisfy the project's needs.
- Begin the design process with the appropriate SRS DID.
- Use simple sentence structures and select words and phrases based on their formal definitions, not on how popular culture defines them.
- The SRS must be understandable. It does not have to be interesting. Aspire to be a good engineer, not a literary artist. ◆

About the Author

William M. Wilson is a retired principal systems engineering consultant formerly with the Software Assurance Technology Center (SATC). He has 40 years of professional engineering experience with NASA, DoD, and industry. He is a recognized author and instructor of software safety and reliability courses, an authority on the specification of software requirements, and a trained lead auditor under the TickIT software certification scheme. Before

The SSG Systems Engineering Process

This brief overview of the Standard Systems Group (SSG) Systems Engineering Process (SEP) summarizes the primary objectives of the SEP. For complete information, see http://web1.ssg.gunter.af.mil/sep/SEPver40/ssddview.html.

In May 1997, the SSG at Maxwell Air Force Base, Gunter Annex in Montgomery, Ala. was rated Level 3 according to the Software Engineering Institute Capability Maturity Model. SSG is one of the larger, more diverse government agencies to achieve this distinction. Continuous, sustained process improvements led to this maturity level, and the method by which it was achieved is embodied in the SEP, now in Version 4. A combination of management and engineering activities composes this standard organizational process that can be tailored to address project specifics.

The SEP is a pragmatic, disciplined approach to software systems engineering. It describes the essential elements of an organization's systems engineering process that must exist to ensure good systems engineering.

The SEP's goals for a product are to

- Meet customer's functional objectives.
- Minimize defects.
- Enhance look and feel of having been built by one person, though it does not depend on one person for maintenance.
- Reduce risk; eliminate rework.
- Improve predictable schedule and cost.
- Provide development insight.
- Enhance maintainability.
- Introduce industry best practices.
- Operationalize policies and directives.

Success in a market-driven and contractually negotiated market is often determined by how efficiently an organization translates customer needs into a product that meets those needs. Good systems engineering is key to that activity, and the SEP provides a way to define, measure, repeat, and enhance performance. The SEP acts as a framework to which continuous process improvement can be added.

Under the SEP, projects and systems experience productivity improvements of 200 percent to 300 percent, a hundredfold reduction in post-release defects, less overtime and fewer crises, a return on investment of up to a ratio of 7-to-1, reduced long-term sustainment costs, and improved interoperability. The employees also are able to feel more competitive.

The greatest benefit of the SEP is that it increases the ability to meet customer cost, schedule, and performance expectations.

Point of Contact: Barry Morton
SSG Software Engineering Process Group Facilitator
Voice: 334-416-3547 DSN 596-3547
E-mail: MortonB@ssg.gunter.af.mil

joining the SATC, he was vice president of Quong and Associates, a consulting firm specializing in quality engineering and assurance practices. He was responsible for aerospace industry software quality assurance standards and procedures. While manager of software engineering assurance in the Office of the Chief Engineer at NASA Headquarters, he established and directed the Software Management and Assurance Program, which produced NASA's first agency-wide software policies and standards. As a member of the Defense Communications Agency's National Military Command Systems Engineering Directorate, he was the project manager and systems engineer for the acquisition and development of several first-generation strategic command, control, and communications systems that support the National Command Authority. He has a bachelor's degree in electrical engineering. He is a member of the IEEE Computer Society, the Association for Computing Machinery, and the American Society for Quality Control.

Point of Contact: Linda H. Rosenberg
Goddard Space Flight Center
Code 300.1, Building 6
Greenbelt, MD 20771
E-mail: Linda.H.Rosenberg.1@gsfc.nasa.gov

References

1. IEEE Standard 830-1993, Recommended Practice for Software Requirements Specifications, Dec. 2, 1995.
2. MIL-STD-498, Software Development and Documentation, Dec. 5, 1994 (http://scpo.nosc.n&498.html).
3. Ganska, Ralph, John Grotzky, Jack Rubinstein, and Jim Van Buren, *Requirements Engineering and Design Technology Report*, Software Technology Support Center, Hill Air Force Base, Utah, October 1995.
4. NASA-STD-2100-91, *NASA Software Documentation Standard*, NASA Headquarters Software Engineering Program, July 29, 1991 (http://satc.gsfc.nasa.gov/assure/docstd.html).
5. Wilson, William, Linda H. Rosenberg, and Lawrence E. Hyatt, "Automated Quality Analysis of Natural Language Requirement Specifications," *Pacific Northwest Software Quality Conference Proceedings*, October 1996, pp. 140-151 (http://sate.gdcnasa.gov/SATC/PAPERS/PNQ/pncl.htnil).
6. 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.