

Automated Quality Analysis of Natural Language Requirement Specifications

William M. Wilson

Software Assurance Technology Center/GSFC
Bld 6 Code 300.1
Greenbelt, MD 20771 USA
+1 (301) 286-0102
William.M.Wilson@gsfc.nasa.gov

Linda H. Rosenberg, Ph.D.

Unisys Federal Systems/GSFC
Bld 6 Code 300.1
Greenbelt, MD 20771 USA
+1 (301) 286-0087
Linda.Rosenberg@gsfc.nasa.gov

Lawrence E. Hyatt

NASA Goddard Space Flight Center
Bld 6 Code 302
Greenbelt, MD 20771 USA
+1 (301) 286-7475
Larry.Hyatt@gsfc.nasa.gov

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. This paper describes the development and experimental use of the Automated Requirements Measurement (ARM) tool. The ARM tool searches the requirements document for terms the SATC has identified as quality indicators. Reports produced by the tool are used to identify specification statements and structural areas of the requirements document that need to be improved.

1. Introduction

The Software Assurance Technology Center (SATC) is part of the Office of Mission Assurance of the Goddard Space Flight Center (GSFC). 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 metric methodologies and tools that identify and assess risks associated with software performance and scheduled delivery. It is generally accepted that the earlier in the life cycle that potential risks are identified the easier it is to eliminate or manage the risk inducing conditions [1].

Despite the significant advantages attributed to the use of formal specification languages, their

use has not become common practice. Because 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]. Many words and phrases have dual meanings which can be altered by the context in which they are used. For example, Webster's New World Dictionary identifies three variations in meaning for the word "align", seventeen for "measure", and four for the word "model". Weak sentence structure can also produce ambiguous statements. "Twenty seconds prior to engine shutdown anomalies shall be ignored." could result in at least three different implementations. Using words such as "large", "rapid", and "many" produces inaccurate requirement specifications. Even though the words "error", "fault", and "failure" have been precisely defined by the Institute of Electrical and Electronics Engineers (IEEE) [5] they are frequently used incorrectly. Defining a large, multi-dimensional capability within the limitations imposed by the two dimensional structure of a document can obscure the relationships between individual 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 has 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.

2. Background

The SATC study was initiated by compiling a list of quality *attributes* that requirements specifications are expected to exhibit [6] [11]. The next step was to list of those aspects of a requirements specification that can be objectively and quantitatively measured. The two lists were analyzed to identify relationships between what can be measured and the desired quality attributes. This analysis resulted in the identification of categories and individual items that are primitive *indicators* of the specification's quality and can be detected and counted by using the document's text file.

Concurrent with development and analysis of the attribute and indicator lists, forty-six requirement specifications were acquired from a broad cross section of NASA projects. These documents were converted into ASCII text files. These files were used to develop a database containing the words and phrases used in the set of document and the number of times that they occurred. This database of basic words was used to refine the list of primitive indicators.

Using the refined list of primitive indicators, an initial version of the Automated Requirements Measurement (ARM) software was developed for scanning the specification files. ARM 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. The source documents are currently being independently reviewed to provide a basis of comparison with the conclusions arrived at using the ARM's reports.

While the source documents are being independently reviewed, the prototype ARM software is being used to aid selected NASA projects to strengthen their requirement specifications. The results of the engineering assessments and feedback from the selected NASA projects will be used to improve ARM's assessment processes and its user interface. **3. Specification Quality Attributes**

Desirable characteristics for requirements specifications are identified [2] [6] as:

- Complete
- Consistent
- Correct
- Modifiable
- Ranked
- Traceable
- Unambiguous
- Understandable
- Verifiable

As a practical matter, it is generally accepted that requirements specifications should also be **Validatable** and **Testable**. These eleven 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 software quality model [7] gives stability, modifiability (changeability), and testability as factors contributing to maintainability. A specification, obviously, cannot be correct if it is incomplete or inconsistent. It would also be difficult to validate a requirement specification that could not be understood.

Most, if not all, of these quality attributes are subjective. A conclusive assessment of a requirements specification's appropriateness requires review and analysis by technical and operational experts in the domain addressed by the requirements. Several quality attributes, 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.

3.1. 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. Since it is difficult to anticipate all real world situations, it is much easier to detect incompleteness than determine completeness. Use of the place holder "TBD" (to be determined) is undeniable evidence that the requirements specification is incomplete. The phrases "as a minimum" and "not limited to", depending on their context, may be more subtle indicators of incompleteness

3.2. 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.

3.3. 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.

3.4. 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. As an example: 5. The XYZ system shall access the ABC, DEF, and GHI databases. 5.6 The XYZ system shall permit and restrict access based on application type. 5.1.7 Engineering applications shall have read access to all databases and write access to the ABC and DEF databases. 5.1.8 Administrative applications shall have read access to the DEF and GHI databases and write access to the GHI database.

3.5. 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. Ranking specifications according to stability and/or importance can conflict with structuring the document to be modifiable. This conflict often arises when there are safety and security requirements to be specified. Is it best to address them in separate documents, sections, or as subsections under the related functional

requirements?

3.6 Testable

In order for a specification to be testable it must be stated in such a manner that pass/fail or quantitative assessment criteria can be derived from the specification itself and/or referenced information [10]. "*The system shall be user friendly.*", can be subjectively interpreted and its implementation will be difficult to test objectively. "*The system's functions shall be activated and terminated by menu selections.*", is a specification that the implementation does or does not satisfy.

3.7 Traceable

Each statement of requirement 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. The example specification statements in paragraph 3.4, above, demonstrate this type of identification. A computer program can easily recognize this type of identification by detecting sentences that begin with strings of numbers separated and terminated with periods. The structure of a requirements document can be assessed by relatively simple algorithms if each specification is uniquely identified and expressed as a simple, uncomplicated statement. For the example referred to above there are specifications at three levels. There is one requirement specified at level one, one at level two and level at level three.

3.8. Unambiguous

A statement that specifies 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 such as "*as required*" or poor sentence structure, as demonstrated by the following sentence, will open the specification statement to misunderstandings. "*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.*"

3.9. Understandable

A requirements specification is understandable if the meaning of each of its statements is easily grasped by all of its readers [11]. This is the primary reason that most specifications are expressed in natural language.

3.10. Validatable

In order to validate a requirements specification each of the individuals and organizations having a vested interest in the system solution must be substantiate that the requirements are true as stated. 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].

3.11. Verifiable

In order to be verifiable requirement specifications at one level of abstraction must be consistent with those at another level of abstraction [11].

4. Specification Quality Indicators

Although most of the quality attributes of documented requirements are subjective, there are aspects of the documentation which can be measured and are indicators of quality attributes. Size, which is a primitive used in many quality metrics, can be directly measured. The size of an individual specification statement can be measured by the number of words it contains. 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. This count 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 specification statements at each level of the document's structure can also be counted. These counts provide an indication as to 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 specification statements are weak or strong.

4.1. Categories

Nine categories of quality indicators for requirement documents and specification statements 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, and structures 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
- Specification Depth

- Readability
- Text Structure

Imperatives

Imperatives are those words and phrases that command that something must be provided. The ARM report lists the total number of times each imperatives was 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 requirements documents that were judged to be the most explicit had the majority of their imperative counts associated with the upper list items.

- ***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 predefined. As an example, "*The XYZ 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*". In a few instances "shall" and "will" were used interchangeably within a document that contained both requirements specifications and descriptions of the operational environment. In those documents, the boundaries of the system being specified were not always sharply defined.
- ***Should*** is not frequently used as an imperative in requirement specification statements. However, when 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."

4.1.2. 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 specified requirements. 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:

4.1.3. 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. 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:

4.1.4. 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

4.1.5. 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

4.1.6. 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.

4.1.7. 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.

4.1.8. 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.

4.1.9. 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.

Since most documents at NASA contain scientific terms which tend to contain words of considerable length, the readability scores are skewed. As shown in Table 1. below, using the data from forty-six requirements documents, the grade levels indicated by the Flesch-Kincaid, Coleman-Liau and Bormuth have a high variance, and the mean for Coleman-Liau grade level is 27.6 with a maximum of 55.80. Alternative readability packages that allow for adjustment of word length are being investigated.

	Flesch-Kincaid	Grd Lvl	Grd Lvl	Grd Lvl
	Rdng	Ez	Lvl	Grd Lvl
mean	47.15	10.76	27.60	11.46
min	28.00	7.80	17.10	11.10
max	61.50	13.80	55.80	11.60
stdev	8.54	1.59	9.29	0.17

Table 1.

4.2. Quality Indicator/Attribute Relationships

Relationships between requirements specifications' quality attributes and categories of indicators measured by the ARM tool are shown below in Figure 1.

INDICATORS OF QUALITY ATTRIBUTES											
Categories of Quality Indicators	Quality Attributes										
	1. Complete	2. Consistent	3. Correct	4. Modifiable	5. Ranked	6. Testable	7. Traceable	8. Unambiguous	9. Understandable	10. Validatable	11. Verifiable
1. Imperatives	X			X			X	X	X	X	X
2. Continuances	X			X	X	X	X	X	X	X	X
3. Directives	X		X			X		X	X	X	X
4. Options	X					X		X	X	X	
5. Weak Phrases	X		X			X		X	X	X	X
6. Size	X					X		X	X	X	X
7. Text Structure	X	X		X	X		X		X		X
8. Spec. Depth	X	X		X			X		X		X
9. Readability				X		X	X	X	X	X	X

Figure 1.

5. RISK

Webster's New World Dictionary defines risk as the possibility of loss or injury, and the degree of probability of such a loss. As the first tangible representation of the needed capability, the requirements specification establishes the basis for all of the project's engineering, management and assurance functions. If its quality is poor, it can give rise to risks associated with the project's products and its resources.

5.1. Product Risks

Inadequacies in the requirements specification document introduces the possibility that the product's design will contain deficiencies that will be propagated as implementation faults. The possibility of these faults increases the probability that one or more of the products characteristics will be unsatisfactory.

5.1.1. Acceptance Risk

An *Acceptance Risk* is the probability that the product will not satisfy its acceptance criteria. A deficiency in any of the requirements specification's quality attributes may introduce a risk that the product will not be acceptable.

5.1.2. Availability Risk

An *Availability Risk* is the probability that the product will not be present or functional at a time when it is needed. A deficiency in any of the requirements specification's quality attributes may introduce a risk that the product will not be delivered on time or its functionality can not be

maintained in the operational environment.

5.1.3. Performance Risk

A *Performance Risk* is the probability that the product will not be capable of performing properly in the operational environment. A performance risk will arise if the requirements specification is difficult to understand or inadequately specifies the functional capabilities that are to be provided.

5.1.4. Reliability Risk

A *Reliability Risk* is the probability that the product will fail in the operational environment. A deficiency in any of the requirements specification's quality attributes can introduce a risk that the product will not achieve the level of reliability need to successfully perform its mission.

5.1.5. Reproducibility Risk

A *Reproducibility Risk* is the probability that the product can not be reproduced for replacement of the original product or for distribution to additional sites. If the requirements specification is poorly written and functions are inadequately prescribed there will be a risk that the product or components of the product cannot be replicated when replacement is necessary.

5.1.6. Supportability Risk

A *Supportability Risk* is the probability that the product can not be adequately maintained or logistically provided for in the operational environment. A deficiency in any of the requirements specification's quality attributes can introduce a risk that the resources needed to operate and maintain the product will not be provided and/or the product itself does facilitate maintenance.

5.1.7. Utility Risk

A *Utility Risk* is the probability that the product will less useful than demanded by the operational environment. A risk that the delivered capability will not provide the user with the full range of necessary functionality will arise if the requirements specification is difficult to understand or the functions that are to be provided are inadequately specified.

5.2. Resource Risks

The estimate of resources needed to provide a capability, including the time allocated for its acquisition, are based of the specification of the requirements that the capability is to satisfy. If the requirements are improperly specified the resource estimates will be incorrect. Product risks introduced by deficient requirements can result in delays and rework that introduce cost and schedule risks.

5.2.1. Cost Risk

A *Cost Risk* is the probability that the production or acquisition of the product will exceed allocated resources. available for that purpose. Any deficiency in the requirements specification, such as incompleteness or ambiguity, that causes the development effort to be under estimated or necessitates rework will introduce a risk that costs will exceed available funding.

5.2.2. Schedule Risk

A *Schedule Risk* is the probability that the product will not be delivered as scheduled. Any deficiency in the requirements specification's quality attributes can introduce a risk that the established schedule is inadequate or that product deficiencies will require unanticipated additional time to correct.

5.3. Quality-Risk Relationships

Figure 2 shows the areas of project risk that are directly impacted by the quality attributes of the requirements specification.

RISK AREAS IMPACTED BY SPECIFICATION QUALITY									
Specification Quality Attributes	1. Product Risk							2. Resource Risk	
	1.1. Acceptance	1.2. Availability	1.3. Performance	1.4. Reliability	1.5. Reproducibility	1.6. Supportability	1.7. Utility	2.1. Cost	2.2. Schedule
1. Complete	X	X	X	X	X	X	X	X	X
2. Consistent	X		X	X	X	X			
3. Correct	X	X	X	X	X	X	X	X	X
4. Modifiable	X	X	X	X	X	X	X	X	X
5. Ranked	X	X	X	X	X	X	X	X	X
6. Testable	X	X		X	X	X		X	X
7. Traceable	X			X	X	X			X
8. Unambiguous	X		X	X	X	X	X	X	X
9. Understandable	X		X	X	X	X	X	X	X
10. Validatable	X			X	X	X			X
11. Verifiable	X			X	X	X	X		X

Figure 2.

6. Specification Standards

Although many government and several professional organizations have published documentation standards that include standards for specifying requirements, none are universally accepted or extensively enforced. NASA has very explicit software documentation standards and style guides, but it allows wide latitude in establishing project standards. In many instances, to minimize effort and reduce costs, projects choose to accept the documentation standards and procedures of the contractor selected to provide the needed capability. Smaller projects, of which there are many in-house in addition to contractual acquisitions, are also frequently combine requirement and design documents to conserve project resources. The standards that are imposed seldom go beyond providing an outline and a general description of the information to be

provided. In many cases no style requirements are established. As a consequence of these circumstances, requirements documents from various sources bear little resemblance to one another.

The content outline of one of the IEEE's eight specification templates [6] and NASA's standard data item description (DID) for requirement specifications documents [8] are shown below and provide an example of the variance in scope that exists between standards.

IEEE 830 93

Software Requirement Specification

1. Introduction
 - 1.1 Purpose
 - 1.2 Scope
 - 1.3 Definitions, acronyms, and abbreviations
 - 1.4 References
 - 1.5 Overview
2. Overall description
 - 2.1 Product perspective
 - 2.2 Product functions
 - 2.3 User characteristics
 - 2.4 Constraints
 - 2.5 Assumptions and dependencies
3. Specific requirements
 - 3.1 External interfaces
 - 3.2 Functions
 - 3.3 Performance requirements
 - 3.4 Logical database requirements
 - 3.5 Design constraints
 - 3.6 Software system attributes
 - 3.7 Organizing the specific requirements
 - 3.8 Additional comments
- Appendixes
- Index

NASA-DID-P200

Requirements

- 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

Several problems related to the structure and organization of the source documents were frequently encountered when attempting to automate the scanning of specification files. The most troubling problem was the inconsistency in paragraph and specification identification across documents and within documents. The variations in 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)

Because of these numbering inconsistencies the current version of the Automated Requirement Measurement (ARM) software is implemented using the following scheme as the basis for recognizing paragraph and specification identifications.

Each requirement specification statement is assumed to be individually distinguished by one of the following markings:

- a. A simple number (i.e. a number without decimal. For example: 1, 23, 104, etc.)
- b. A hierarchical number (i.e. a number with decimals to indicate levels of structure. For example: 1.1.1, 23.4.5.6, etc.)
- c. A lettered hierarchical number (i.e. a hierarchical number immediately preceded by a letter. followed by a period. For example: L1.1.1, B23.4.5.6, etc.)
- d. An integer number (i.e. a simple number. For example: 1., 23., 104., etc.)
- e. A letter designation (i.e. a single capital/lower-case letter followed by a period. For example: A., P., b., x. etc.)

Another problem encountered when scanning requirement specification files is distinguishing statements that *prescribe* required system capabilities from those that *describe* the operational

environment. This problem has both structural and terminology aspects. In general three structural separations of descriptive and prescriptive information are usually used. The NASA documentation standards call for information to be presented in two distinct documents. The targeted operational environment is to be described within the NASA Concept Document, NASA-DID-P100 [8] while the standard shown above, NASA-DID-P200, is to contain prescriptive specifications. Other standards [2] [6] allocate descriptive and prescriptive information to separate parts of the same document. In several NASA contractor provided requirements documents the two categories of information were interlaced within the same sections of the document.

In some instances the ability to distinguish between statements prescribing requirements and those describing environmental features was further complicated by authors using the imperatives "shall", "will", and "should" interchangeably.

7. Results

The graphs presented in this section are based on data collected by the ARM processor. The assessments accompanying the graphical presentations are based on an examination of the source documents in light of the data used to create the graphs.

7.1. Document Size And Imperatives

The relationship between document size, measured by total number of imperatives and total lines of text found in each specification document is depicted by Figure 3., below.

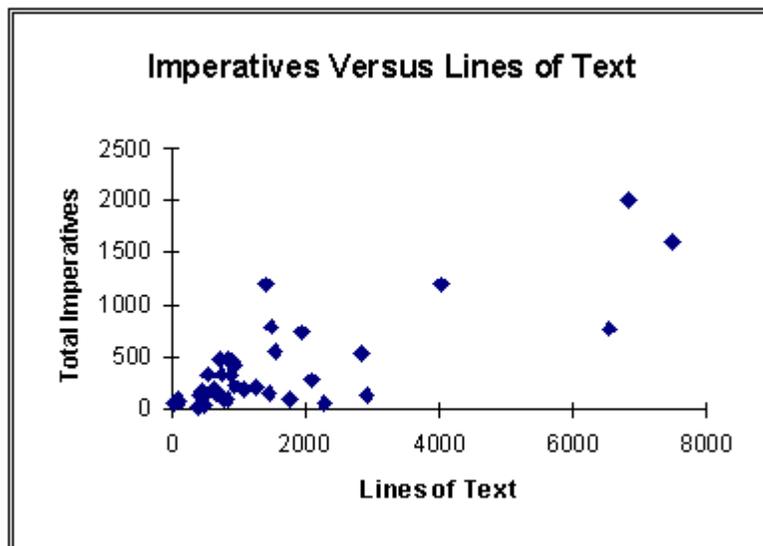


Figure 3.

The ratio of lines of text to imperatives for each specification document is shown in Figure 4. Most documents have ten lines of text, or less, for each imperative. When inspected, the documents with ratios that exceed one-to-one, appeared to have been created using a requirements analysis methodology or tool. The low ratios resulted of the fact that the documents

exclusively addressed software requirements and many single statement contains multiple imperatives connected by "and" or "or". Those specification documents with ratios above ten lines of text per imperative appear to have been developed based on an ad-hoc documentation standards with scopes that includes the description of the operational environment as well a the prescription of system requirements

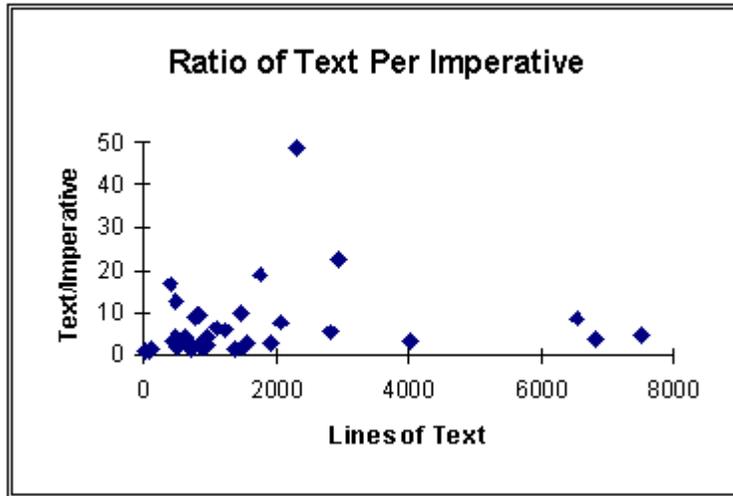


Figure 4.

The document with the highest ratio of text to imperatives also included descriptions of the project and requirements for the development environment. This extreme data point was removed in Figure 4a. to improve the visibility of the remaining data points.

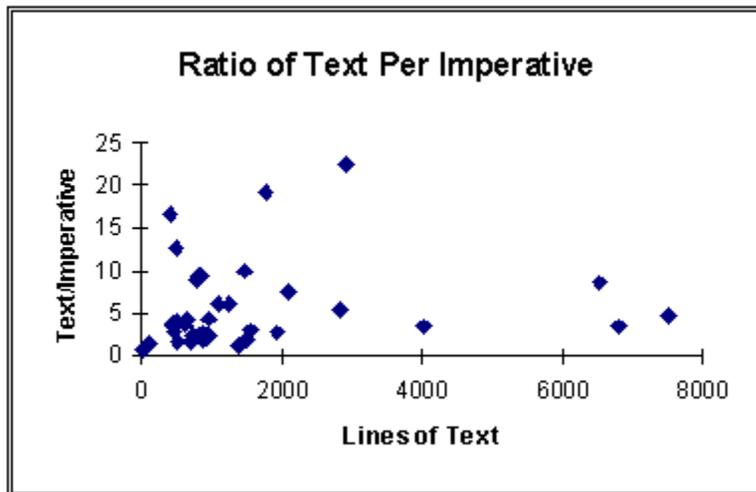


Figure 4a.

7.2. Document Size And Structure

The relationship between document size and the document's depth of text structure shown by Figure 5. This data is based on detection and automatic analysis of paragraph numbering and statement identifiers.

In general, the smaller documents have a greater percent of their statements numbered. Inspection of these documents indicates that they were probably developed using Computer Aided Software Engineering (CASE) tools. They are very hierarchical in structure. High level statements have been repetitively decomposed into subordinate statements. Every level of text is hierarchically identified and each statement containing an imperative has been given an additional unique sequential number.

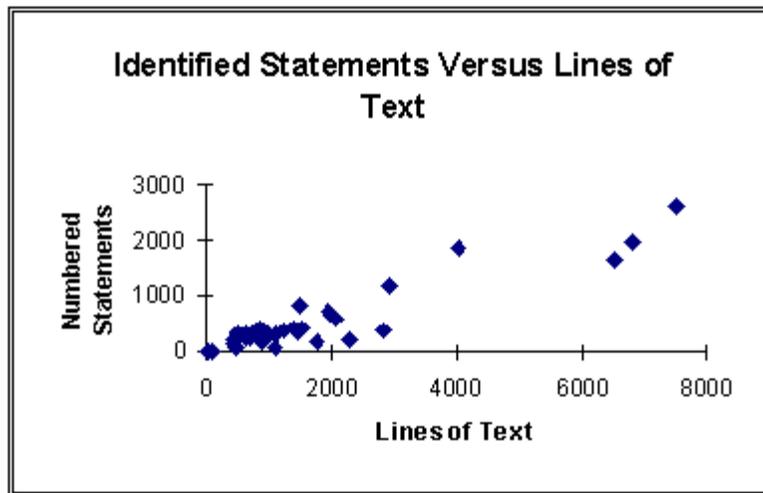


Figure 5.

Figure 6. shows the ratio of numbered statements to the number of imperatives within each specification document. Most of the documents containing less than 100 imperatives have a relatively high ratio. Inspection of these documents revealed that they were developed using a documentation format that was too detailed for the scope of the capability being specified. A significant number of the sections in these documents addressed administrative and general information rather than requirements specifications. In general, documents with more than 100 imperatives seem to have a ratio of numbered statements to imperatives close to one. This implies that these document's specification statements have a high degree of traceability.

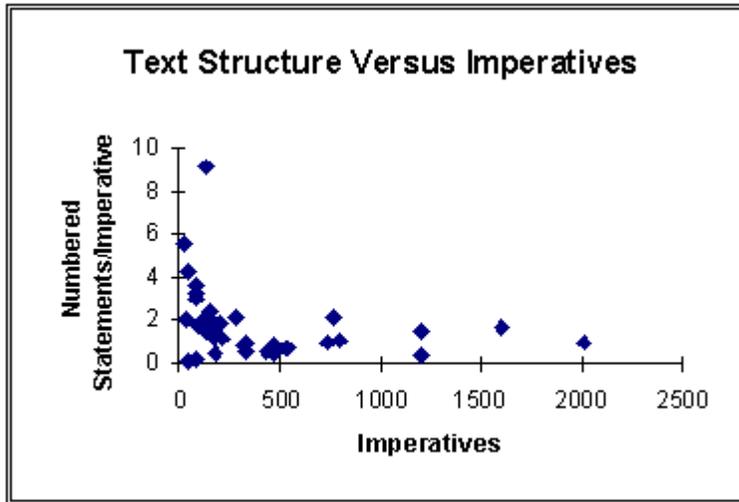


Figure 6.

7.3. Imperatives And Subjects

The relationship between the number of imperatives and the unique number of subjects of imperative statements is shown by Figure 7. Closer examination of these documents found that the real ratio between subjects and imperatives is actually lower than shown in Figure 8. In many instances the same subject was stated in different terms, apparently to introduce variety and hold the readers interest. The ARM software is not capable of recognizing different phrases as the same subject and counts each distinct combination of words immediately preceding the imperative as a unique subject. The larger documents were found to be much greater in scope and to addressed subjects at a higher level of capability. In these higher level specifications, multiple statements were used to define and bound each major functional capability

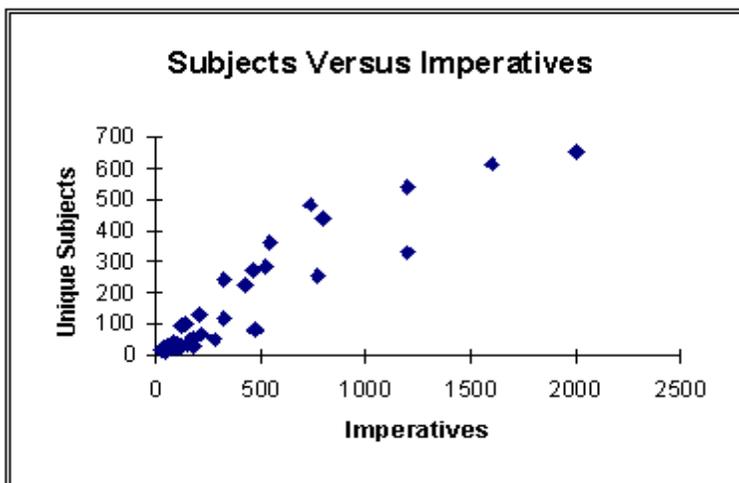


Figure 7.

Figure 8. shows the ratio of imperatives to subjects versus lines of text. Within a limited amount of variation, it appears that even in those documents containing many imperatives per subject, each imperative is supported with a nominal number of text lines

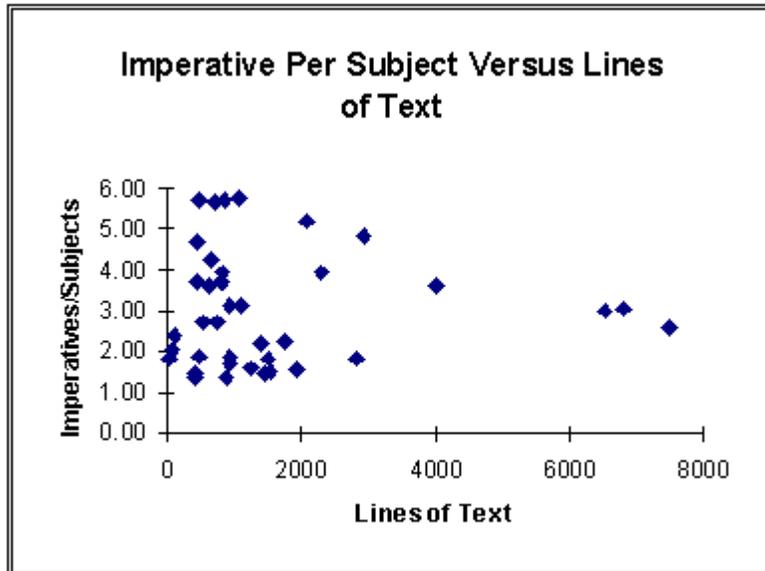


Figure 8.

7.4. Weak Phrases

Figure 9. shows the number of weak phrases versus the number of imperatives found in NASA requirements documents. This representation implies a somewhat linear relationship between these two indicators for large documents.

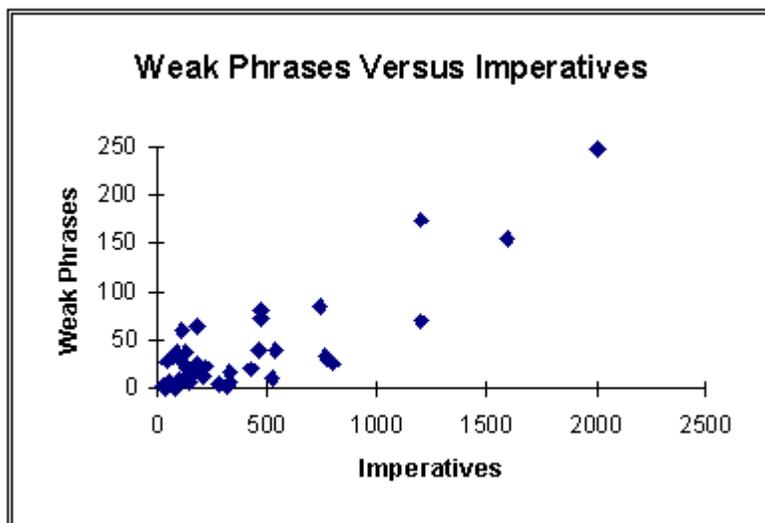


Figure 9.

Figure 10. Presents the ratio of weak phrases to imperatives versus the number of imperatives contained in the document. In most of these documents it appears that one to ten percent of the individual specification statements contain weak phrases. Weak phrases were found in thirty to sixty percent of the individual specification statements in many of the smaller documents. These are same documents that were well structured as a result of using CASE tools.

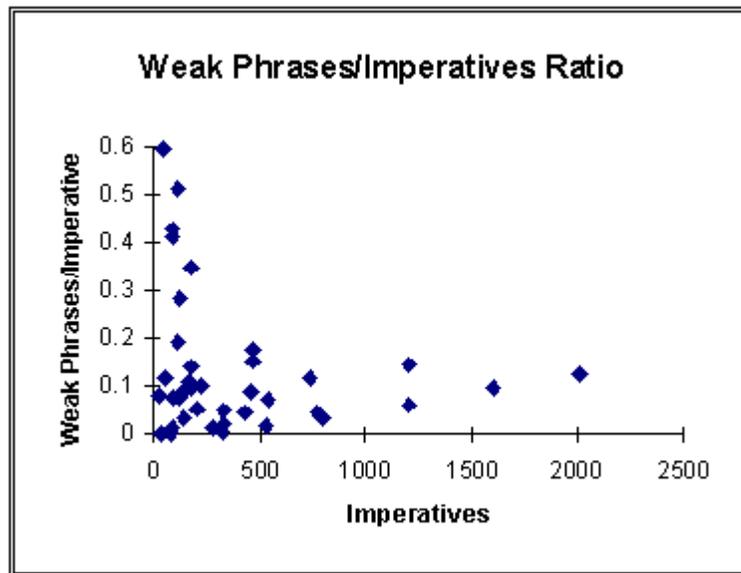


Figure 10.

8. Conclusions

Three initial general conclusions have arisen from the subject study, First, it is possible to gain insights into the quality of a requirements specification document through the use of data gathered by automated processing of the specification file. Second, the effectiveness of expressing requirements specifications with natural language can be greatly improved through relatively simple and readily available methods. Lastly, specifications developed using a proven methodology with the aid of an appropriate requirements definition tool are better structured, more consistently numbered, and crisper than those developed solely based on a documentation standard. Use of a CASE tool, however, is not a substitute for sound engineering analysis.

8.1. Management Recommendations

Based on the current results of the SATC study it is recommended that project managers ensure that requirements specification and style standards 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 requirements document be emphasized to encourage straightforward writing styles 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..

8.2. Technical Recommendations

Requirements analysis and development of specifications should occur prior to writing the requirements document, not as a by-product of the documentation activity. Requirement specifications should be developed using a methodology that is appropriate to the nature of the project and its products. CASE tools should be used if they support the project's development methodology and the technology being addressed by the project. Specification writers should be taught how to write simple direct statements. If conditional clauses are needed, they should be placed at the front of the sentence. Subjects, imperative/verb combinations, and objects of the verb should occur in that order. This will prevent interesting, but confusing statements, such as: "The Romans, the Greeks shall defeat if better prepared they be."

1. Future Work

Three types of enhancements are planned for the ARM software. The enhancement of immediate priority is to improve the ARM tool's user interface and to provide flexibility to tailor its reports to the user's areas of interest. This will facilitate enlisting additional projects to participate in the SATC study, add their requirements documents to the SATC database and provide user feedback.

The second ARM tool enhancement activity will take place in parallel with the improvement of its user interface and report generator. The results of the engineering review of the requirements specifications in the SATC database will be used to refine ARM's search and counting schemes. This will improve the validity of data produced and heighten user confidence in the conclusions that can be inferred from its reports

Lastly, once the ARM's search and counting schemes have been enhanced, words and phrases will be identified that can be used as indicators that the following types of requirements have been addressed within the requirements document.

- Acceptance Testing
- Data Handling
- Design Standards
- Integrity
- Maintainability
- Reliability
- Reusability
- Timing and Sizing

10. 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 Force 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, February 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.