2

RADC-TR-90-348
Final Technical Report
December 1990

ASSISTANT FOR SPECIFYING QUALITY SOFTWARE (ASQS) MISSION AREA ANALYSIS

Advanced Technology, Inc.

Douglas Schaus

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Air Development Center
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RADC-TR-90-348 has been reviewed and is approved for publication.

APPROVED: Roger J. Dziegiel, Jr

ROGER J. DZIEGIEL, JR.
Project Engineer

APPROVED: Raymond P. Urtz Jr

RAYMOND P. URTZ, JR.
Technical Director
Directorate of Command & Control

FOR THE COMMANDER: Igor G. Plonisch

IGOR G. PLONISCH
Directorate of Plans & Programs

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE December 1990	3. REPORT TYPE AND DATES COVERED Final Mar 88 to Nov 89

4. TITLE AND SUBTITLE
ASSISTANT FOR SPECIFYING QUALITY SOFTWARE (ASQS)
MISSION AREA ANALYSIS

5. FUNDING NUMBERS
C - F30602-87-D-0094
Task 0004
PE - 63728F
PR - 2527
TA - QC
WU - 04

6. AUTHOR(S)
Douglas Schaus

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Advanced Technology, Inc.
2121 Crystal Drive
Arlington VA 22202

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Air Development Center (COEE)
Griffiss AFB NY 13441-5700

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RADC-TR-90-348

11. SUPPLEMENTARY NOTES RADC Project Engineer: Roger J. Dziegiel, Jr./COEE/(315)330-4063
This work was performed by Advanced Technology, Inc. under Subcontract to IIT Research Institute, Route 26N, Rome NY 13440.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report documents the Mission Area Analysis (MAA) conducted to identify mission and system characteristics that impact software quality. Characteristics were used to develop a set of rules that link characteristics to quality factors or characteristics to characteristics. This analysis did not determine what the actual software quality requirements are for the systems reviewed. This determination is made during a consultation with the Assistant for Specifying Quality Software (ASQS). The main objectives of the MAA were to: 1) develop mission profiles for the five Air Communications (C3); and Force-Mission Management; 2) determine what software quality factors are impacted by the respective missions; and 3) create rules and questions for use in ASQS.

The MAA tried, with limited success, to move the Software Quality Framework from the theoretical to general practice by creating processes for determining software quality goals and placing them into an expert system.

14. SUBJECT TERMS
Mission Area Analysis, Software Quality Framework, Software Quality, Software Quality Specification, Expert System

15. NUMBER OF PAGES
166

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

TABLE OF CONTENTS

Chapter	Section		Page
	Executive Summary		x
I.	Introduction		
	1.0	Background	1-1
	1.1	Purpose	1-2
	1.2	Study Objectives	1-2
	1.3	Software Quality Factor Definitions	1-2
II.	Approach		
	2.0	System Lifecycle	2-1
	2.1	Feasibility Analysis	2-2
	2.2	Knowledge Representation	2-3
	2.2.1	Mission Area and Software Types	2-5
	2.2.2	Projects and Functions	2-6
	2.3	Knowledge Acquisition	2-7
	2.3.1	Domain Experts	2-8
	2.3.2	Staffing	2-8
	2.3.3	Domain Considerations	2-9
	2.3.4	Quality Factor Definitions	2-9
	2.4	Knowledge Representation	2-10
	2.4.1	Rules and Queries	2-10
III.	Intelligence System Software Analysis		
	3.0	Scope	3-1
	3.1	Intelligence System Software Description	3-1
	3.2	Software Quality Factor Applicability	3-2
	3.3	Tactical Intelligence System Notional Architecture	3-5
	3.4	Applicability of Quality Factors by Software Function	3-6
	3.4.1	Message Processing	3-7
	3.4.2	Data Fusion/Correlation	3-7
	3.4.3	Situation Assessment	3-8

TABLE OF CONTENTS (Cont)

Chapter	Section	Page	
	3.4.4	User-System Interface	3-8
	3.4.5	System Monitor and Control	3-8
	3.4.6	System Software	3-9
	3.4.7	Collection Management	3-9
	3.4.8	General Considerations	3-9
	3.5	Mission-Oriented Rules and Queries	3-10
	3.6	System Security	3-11
	3.6.1	Class I Rules	3-11
	3.6.2	Class II Rules	3-18
	3.6.2.1	Determining the Level of System Requirements	3-18
	3.6.2.2	Data Exposure (DE)/System Risk (SR)	3-21
	3.6.2.3	Determining the Impact of Security Requirements on Quality	3-23
	3.7	Message Processing	3-26
	3.7.1	Class I Rules	3-26
	3.7.2	Class II Rules	3-26
	3.8	User-System Interface	3-28
	3.8.1	Class I Rules	3-28
	3.8.2	Class II Rules	3-29
	3.9	Indications and Warning/Intelligence Analysis	3-30
	3.9.1	Class I Rules	3-30
	3.9.2	Class II Rules	3-30
	3.10	Collection Management	3-32
	3.10.1	Class I Rules	3-33
	3.10.2	Class II Rules	3-33
	3.11	General Considerations	3-34
	3.11.1	Error Recovery	3-35
	3.11.1.1	Class I Rules	3-35
	3.11.1.2	Class II Rules	3-35
	3.11.2	Crisis/Time Constrained Operations (CT)	3-37
	3.11.2.1	Class I Rules	3-37
	3.11.2.2	Class II Rules	3-37

TABLE OF CONTENTS (Cont)

Chapter	Section		Page
	3.11.3	Authenication	3-38
	3.11.4	Communications Field Integrity	3-40
	3.11.5	Fault Tolerance (FT)	3-41
	3.11.6	High-Order Language (HOL)	3-44
	3.11.7	Interface Complexity	3-46
	3.11.8	Testing (TE)	3-47
	3.11.9	Architecture	3-49
IV. Satellite Software Analysis			
	4.0	Scope	4-1
	4.1	Satellite System Software Definition	4-1
	4.2	Software Quality Factor Applicability	4-1
	4.3	Satellite System Notional Architecture	4-3
	4.4	Applicability of Software Factors by Component	4-5
	4.4.1	Command and Control/Data Communications	4-5
	4.4.2	Attitude Control, Propulsion and Power	4-5
	4.4.3	General Considerations	
	4.5	Factor Applicability Conclusions	4-6
	4.6	Query Development	4-8
	4.7	Satellite Software Analysis	4-12
	4.7.1	Software Features	4-12
	4.8	Burn Time	4-14
	4.8.1	Query BT.1	4-14
	4.8.2	Query BT.2	4-15
	4.8.3	Query BT.3	4-16
	4.8.4	Querty BT.4	4-16
	4.9	Communication Frequencies	4-17
	4.9.1	Query CF.1	4-17
	4.9.2	Query CF.1.1	4-19
	4.9.3	Query CF.2	4-20
	4.9.4	Query CF.2.1	4-21

TABLE OF CONTENTS (Cont)

Chapter	Section		Page
	4.9.5	Query CF.3	4-21
	4.9.6	Query CF.4	4-22
	4.9.7	Query CF.5	4-23
	4.9.8	Query CF.6	4-23
	4.9.9	Query CF.7	4-24
	4.10	Data Encryption	4-24
	4.10.1	Query DE.1	4-24
	4.10.2	Query DE.2	4-25
	4.10.3	Query DE.3	4-25
	4.10.4	Query DE.4	4-27
	4.10.5	Query DE.5	4-27
	4.10.6	Query DE.6	4-28
	4.10.7	Query DE.7	4-29
	4.11	Fault Tolerance	4-30
	4.11.1	Query FT.1	4-30
	4.11.2	Query FT.2	4-31
	4.11.3	Query FT.3	4-31
	4.11.4	Query FT.4	4-32
	4.11.5	Query FT.5	4-33
	4.11.6	Query FT.6	4-33
	4.11.7	Query FT.7	4-34
	4.11.8	Query FT.8	4-35
	4.11.9	Query FT.9	4-35
	4.11.10	Query FT.10	4-36
	4.12	High-Order Language	4-37
	4.12.1	Query HOL.1	4-37
	4.12.2	Query HOL.2	4-38
	4.12.3	Query HOL.3	4-38
	4.12.4	Query HOL.3.1	4-39
	4.12.5	Query HOL.4	4-40
	4.12.6	Query HOL.5	4-40

TABLE OF CONTENTS (Cont)

Chapter	Section		Page
	4.12.7	Query HOL.5.1	4-41
	4.12.8	Query HOL.6	4-41
	4.12.9	Query HOL.7	4-42
	4.13	Interface Complexity	4-43
	4.13.1	Query IC.1	4-43
	4.13.2	Query IC.2	4-44
	4.13.3	Query IC.3	4-44
	4.13.4	Query IC.4	4-45
	4.13.5	Query IC.5	4-46
	4.13.6	Query IC.6	4-47
	4.14	Movable/Adjustable Subsystems	4-48
	4.14.1	Query MAS.1	4-48
	4.14.2	Query MAS.2	4-49
	4.15	Memory Utilization/Management	4-50
	4.15.1	Query MU.1	4-50
	4.15.2	Query MU.2	4-50
	4.15.3	Query MU.3	4-51
	4.16	Orbital Characteristics	4-52
	4.16.1	Query OC.3	4-52
	4.17	Processing Capabilities	4-53
	4.17.1	Query PC.1	4-53
	4.17.2	Query PC.2	4-53
	4.18	Power Sources	4-54
	4.18.1	Query PS.1	4-54
	4.18.2	Query PS.2	4-55
	4.18.3	Query PS.3	4-55
	4.18.4	Query PS.4	4-56
	4.19	Testing	4-57
	4.19.1	Query TE.1	4-57
	4.19.2	Query TE.2	4-58

TABLE OF CONTENTS (Cont)

Chapter	Section		Page
V.	**Conclusions and Recommendations**		
	5.0	Conclusions	5-1
	5.1	Recommendations	5-2
	List of References		R-1
	Appendix A-Mission Area Decompositions		A-1

LIST OF FIGURES

Number	Title	Page
2.1	Expert System Lifecycle	2-1
2.2	ASQS Context Tree	2-4
3.1	Tactical Intelligence System Notional Architecture	3-5
3.2	Quality Factor Increase/Decrease Scale	3-36
4.1	Satellite System Notional Architecture	4-4
4.2	Quality Factor Applicability Matrix	4-6

LIST OF TABLES

Number	Title	Page
1.1	Factor Definitions	1-3
1.2	Quality Factor Criteria Definitions	1-4
2.1	ASQS Feasibility Analysis	2-3
3.1	Tactical Intelligence System Functions	3-6
4.1	Satellite Component Attributes	4-8
4.2	Software Quality Factor Characteristics	4-9
4.3	Satellite Software Query Topics	4-10
4.4	Software Features	4-13

Executive Summary

This report documents the Mission Area Analysis (MAA) conducted to identify mission and system characteristics that impact software quality. These characteristics and their supporting rationale were used to develop rules for an expert system, the Assistant for Specifying Quality Software (ASQS). ASQS automates the quality specification process addressed in Specification of Software Quality Attributes, Software Quality Specification Guidebook, RADC-TR-85-37, Vol II. This process is part of a quality specification and measurement approach adopted by Rome Air Development Center (RADC) as the Software Quality Framework (SQF).

The MAA builds on a mission taxonomy used in the Software Test Handbook, RADC-TR 84-53, Vol. I that includes five Air Force mission areas; Armament, Avionics, Missile-Space, Command, Control and Communications (C^3) and Force-Mission Management. Each mission area was reviewed to identify those characteristics that impact system software quality according to the Software Quality Framework quality factor definitions The rationale behind each identified characteristic was captured based on expert opinions and document research. The result was a set of rules that linked characteristics to characteristics and characteristics to factors.

To optimize available resources, the reviews of the MAAs examined in detail two mission areas and two specific types of software. Conclusions of this comprehensive review of a limited MAA domain included:

- Additional review(s) should be conducted to cover all five Air Force Mission Areas
- Limitations of current expert systems technology dictate that the software quality framework be extended to achieve the precision needed for expert systems.

Rules were developed for a satellite and an intelligence systems. Some of them were limited based on the criteria (e.g., when estimating to what extent a particular mission attribute impacted software quality). The knowledge base and the Software Quality Specification Framework and its basic premises should be expanded to assist in continuing the development of ASQS. The effectiveness of future knowledge acquisition will be dependent on getting knowledge engineers involved in an actual development effort, with free access to experts, system requirements, and software designs. This is because trying to re-create, from expert recall, the undocumented rationale or key decisions made which influenced a program's quality goals is too subjective and resource intensive.

In summary, the MAA review has made progress in moving the software quality framework further into general practice. Additional work is recommended to enhance the depth of the

software quality framework. Then, expert systems technology can be applied to realize additional benefits in the useage of software quality specifications.

I. Introduction

This document is the final report on the Mission Area Analysis (MAA) conducted by IIT Research Institute (IITRI) and Advanced Technology Inc. (ATI) in support of the Assistant for Specifying Quality Software (ASQS), an expert system developed by Dynamics Research Corporation (DRC).

1.0 Background

Software quality metrics have not been widely used as a tool by acquisition managers because the utility of metrics is not clearly perceived by them. Software quality metrics, to be either predictive or evaluative value to the acquisition manager, must be described in a way that is relatable to the software products end use. Accordingly, if they are perceived as having a "real life" relationship to software functional and/or performance requirements, metrics might receive better acceptance among acquisition managers.

The bulk of the work done to date in the field of software metrics has concentrated on the software science aspects of metrics, as opposed to the utility of metrics for actual software development efforts. Studies, such as the RADC Software Quality Framework, have developed metrics that describe attributes of software, but they are not descriptive of the ability of the software to meet end use requirements. Complicating this picture is that the factors defined describe very general characteristics (efficiency, integrity, correctness, flexibility, etc.) as do the criterion measures (autonomy, distributedness, virtuality, generality, etc.) and the metrics (access control, self-sufficiency, accuracy checklist, etc.). This is not to say that the measures as measures are invalid. On the contrary, their validity is not the issue. What may be an issue is that these measures may be at a level of abstraction removed from the average acquisition manager's understanding of quality terms. There is nothing in the structure that either predicts or evaluates in a direct way if the software will meet the required functional and performance requirements. And this tends to be the acquisition manager's primary preoccupation: will the software meet the requirements?

The problem is twofold. First, the quality specification process must produce a relevant and tailored application of the current metrics to a user's acquisition program. Second, the metrics need to be reviewed, and if necessary, expanded to ensure they give meaningful measures of how well the software meets the program requirements. The former is the focus of ASQS and the Mission Area Analysis. The latter is a validation effort which is a natural follow-on to the MAA, but it was beyond the scope of this task.

1.1 Purpose

The purpose of this analysis was to identify characteristics of a mission-specific software systems that help determine software quality requirements as defined by the Specification of Software Quality Attributes, Software Quality Specification Guidebook. Characteristics were used to develop a set of rules that link characteristics to quality factors or characteristics to characteristics. This analysis did not determine what the actual software quality requirements are for the s, .ems reviewed. This determination is made during a consultation with ASQS.

1.2 Study Objectives

The main objectives of the Mission Area Analysis were to:

a. Develop mission profiles for the five Air Force mission areas, Armament, Avionics, Missile-Space; Command, Control, and Communications (C^3); and Force-Mission Management.

b. Determine what software quality factors are impacted by the respective missions.

c. Create rules and questions for use in ASQS.

1.3 Software Quality Factor Definitions

This analysis used the quality factors and definitions as stated in the Specification of Software Quality Attributes, Software Quality Specification Guidebook. For clarity and understanding, the Software Quality Framework (SQF) terms and definitions are summarized below.

The software quality factors are divided into three categories -- Performance, Design and Adaptation. These three categories are defined as follows:

Performance: The ability of the software to function as designed and to function in the presence of hardware and software faults and errors.

Adaptation: The process by which the software is used beyond its original requirements, such as extending or expanding capabilities and adapting the software for use in another application or a new environment.

<u>Design</u>: The process by which the software is systematically engineered, designed, and tested to ensure the minimization of software failure and the need for future redevelopment.

The three quality factor categories are subdivided into the thirteen software quality factors which are further divided into a criteria. TABLE.1.1 lists the categories and factor definitions. TABLE.1.2 lists the criteria definitions.

TABLE 1.1. Factor Definitions

Performance	
Efficiency:	The ability of the software to optimize its use (speed, size, execution, etc.) or resources (memory, data storage, processor speed, etc.).
Integrity:	The ability of the software to detect and repel attempts to access the system or its software by unauthorized users.
Reliability:	The ability of the software to accomplish the function it was designed to perform.
Survivability:	The ability of the software to operate in a degraded mode (presence of errors, software failures or hardware failures).
Usability:	The amount of effort required to properly use the software.
Adaptation	
Expandability:	The ability to change the software to increase its capability or performance by enhancing current functions or by adding new functions or data.
Flexibility:	The ability to change the software missions, functions, or data to satisfy other requirements.
Interoperability:	The ability to couple software of one system to software of one or more systems.
Portability:	The ability to transport software from one operating environment (hardware, configuration or software system) to another.
Reusability:	The ability to convert a software component for use in another application.
Design	
Correctness:	The extent to which the software design, coding, and implementation conforms to its specifications and standards.
Maintainability:	The degree to which software errors or failures can be located and fixed within a specified time period.
Verifiability:	The degree to which the software operation and performance can be tested and verified.

TABLE 1.2. Quality Factor Criteria Definitions

Performance	
Accuracy:	Those characteristics of software which provide the required precision in calculations and outputs
Anomaly Management	Those characteristics of software which provide for continuity of operations under and recovery from non-nominal conditions
Autonomy:	Those characteristics of software which determine its non-dependency on interfaces and functions
Distributedness:	Those characteristics of software which determine the degree to which software functions are geographically or logically separated within the system
Effectiveness-Comm:	Those characteristics of the software which provide for minimum utilization of communications resources in performing functions
Effectiveness-Processing:	Those characteristics of the software which provide for minimum utilization of processing resources in performing functions
Effectiveness-Storage:	Those characteristics of the software which provide for minimum utilization of storage resources
Operability:	Those characteristics of software which determine operations and procedures concerned with operation of software and which provide useful inputs and outputs which can be assimilated
Reconfigurability:	Those characteristics of software which provide for continuity of system operation when one or more processors, storage units, or communication links fails
System Accessibility:	Those characteristics of software which provide for control and audit of access to the software and data
Training:	Those characteristics of software which provide transition from current operation and provide initial familiarization

TABLE 1.2. Quality Factor Criteria Definitions (Cont)

Adaptation	
Application Independence:	Those characteristics of software which determine its nondependency on database system, microcode, computer architecture, and algorithms
Augmentability:	Those characteristics of software which provide for expansion of capability for functions and data
Commonality:	Those characteristics of software which provide for the use of interface standards for protocols, routines, and data representations
Document Accessibility:	Those characteristics of software which provide for easy access to software and selective use of its components
Functional Overlap:	Those characteristics of software which provide common functions to both systems
Functional Scope:	Those characteristics of software which provide commonality of functions among applications
Generality:	Those characteristics of software which provide breadth to the functions performed with respect to the application
Independence:	Those characteristics of software which determine its non-dependency on software environment (computing system, utilities, input, output routines, libraries)
System Clarity:	Those characteristics of software which provide for clear description of program structure in a non-complex and understandable manner
System Compatibility:	Those characteristics of software which provide the hardware, software, and communication compatibility of two systems
Virtuality:	Those characteristics of software which present a system that does not require user knowledge of the physical, logical or topological characteristics
Design	
Completeness:	Those characteristics of software which provide full implementation of the functions required
Consistency:	Those characteristics of software which provide for uniform design and implementation techniques and notation
Traceability:	Those characteristics of software which provide a thread of origin from the implementation to the requirements with respect to the specified development envelope and operational environment
Visibility:	Those characteristics of software which provide status monitoring of the development and operation
General	
Modularity:	Those characteristics of software which provide a structure of highly cohesive components with optimum coupling
Self-Descriptiveness:	Those characteristics of software which provide explanation of the implementation of functions
Simplicity:	Those characteristics of software which provide for definition and implementation of functions in the most noncomplex and understandable manner

II. Approach

2.0 System Lifecycle

The Mission Area Analysis covered three of the expert system lifecycle phases shown in Figure 2.1. The emphasis was placed on knowledge acquisition and knowledge representation with some effort allotted to validation. Like the lifecycle for traditional software, the phases shown below are not always distinct and sometimes involve some overlap. In performing the knowledge acquisition for ASQS, certain facts about the task and the domain were uncovered that normally are addressed during the feasibility analysis. They impacted this effort and will likely impact future efforts to populate the ASQS knowledge base.

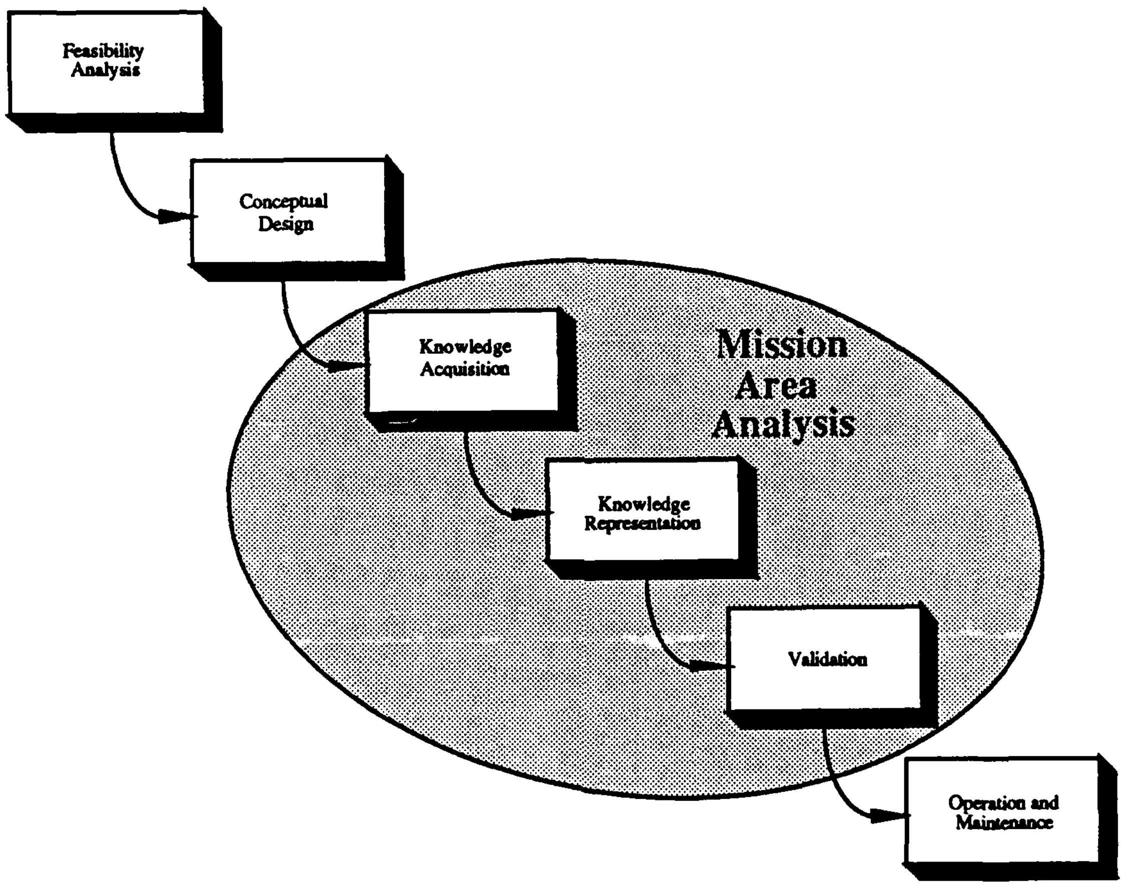

Figure 2.1. Expert System Lifecycle

2.1 Feasibility Analysis

One method of improving the likelihood of success in expert system construction is to choose a manageable task. The task should not be too difficult, and a good model of how the task is performed should exist. ASQS is an R&D activity attempting to automate a complex method in a useable, consistent manner. Other quality programs do not appear to be as comprehensive as ASQS, therefore it should not be expected that mission area experts will be thoroughly familiar with this technology and may require additional training. When evaluated against these criteria, ASQS has several shortcomings. First, the real-world model for determining and assessing software quality is varied and not well-defined. There are few, if any software quality axioms and the Software Quality Framework (SQF) attempts to formalize the process and provide standardization. A universally accepted quality specification process that can be captured for implementation in an expert system does not currently exist. Therefore the mission analysis required a search for the process in addition to an attempt to document the process.

Scope is the second problem which exists. ASQS attempts to capture the rationale for determining the software quality requirements of an unbounded number of Air Force missions. Although only five mission areas are explicitly mentioned in the ASQS operational concept, they actually represent a very broad range of Air Force missions. Accordingly, the number of requirements and system architectures that ASQS must address, which potentially impact software quality, is substantial. Multiply this by the number of judgements that are made to define the software requirements (and their resulting quality constraints) and the inherent risks to the success of ASQS are evident. It could be easily argued that ASQS is attempting to capture the system engineering process in an expert system.

TABLE 2.1 summarizes the key risk areas of ASQS. All of the risks are significant, however, the last one, What will the criteria be for success?', is perhaps the most important. The SQF defines software quality as a precise and quantifiable science.• The risk is high if ASQS is expected to exactly predict the proper quality levels for system software because the knowledge domain is so large and the real-world definition of quality is still evolving, but if the objective is to gain a better appreciation of the system software quality goals, then the risk is more reasonable.

TABLE 2.1. ASQS Feasibility Analysis

Criteria	Impact	Risk Level
Does ASQS have a narrow focus?	The best expert system tasks are rather limited in nature and have a narrow focus.	High
Are experts demonstrably better than amateurs?	It is generally difficult to attempt a task that has proved equally difficult for experts and novices. This suggests the task is inherently difficult.	Medium
Are the rules of software quality stable?	It is important to know whether the the criteria for performing the task are still changing; otherwise it will be difficult to capture knowledge.	High
Is a human expert available and committed?	It is essential to have access to experts who will be dedicated to the project and provide the domain knowledge.	Low
Is the skill of software quality specification taught or documented?	Teaching material or internal documentation and guidelines for performing the expert system task are valuable sources of information.	Medium
What will the criteria for success or failure be?	It is important to know what will constitute success for the expert system. Does the expert system have to be 100% correct, or will something less be acceptable?	High

2.2 Knowledge Representation

The knowledge representational scheme (i.e., the context tree) used by ASQS is shown in Figure 2.2. The context tree is a hierarchical view of how knowledge is organized within ASQS. It parallels a standard development process that starts with general mission requirements and continues through detailed system design. As one progresses through the tree, more system characteristics are known and quality attributes estimated. For this analysis, each level in the schema was matched to the appropriate specification and design documents. This helped define the type of knowledge required to support rule development at each level. Mission Area (MA) is equivalent to a mission needs or operational concept document; Software Type (ST) and Project (PR) are equivalent to an A-Level or System/Segment Specification and System Design Document; and Function (FN) and Main Specification (MS) are equivalent to the B-Level or Software Requirements Specification and Software Design Document.

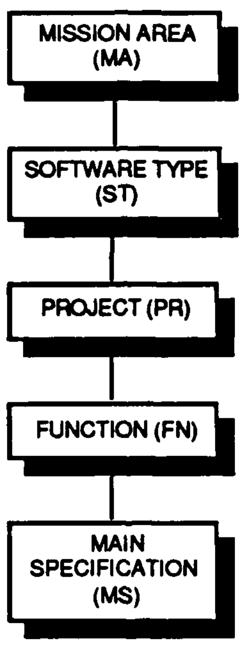

Figure 2.2. ASQS Context Tree

Since levels of the context tree potentially yield quality data the factors are general enough to justify conclusions with as little as a mission statement. The value of examining a system in more detail is to quantify to what extent the factor applies. If possible, this would require a detailed knowledge of the requirements and design constraints, and estimates of how each impacts a factor. For this analysis, confidence values were assigned to the ASQS rule conclusions, even though it was very subjective. Confidence values ranged from 0 to 1, with .3 and below defined as weakly indicated, greater than .3 and up to .5 defined as moderately indicated, and greater that .5 defined as strongly indicated. ASQS through its reasoning chain aggregates the confidence values and determines the factor rating.

The five Air Force Mission Areas (MA) are divided into Software Types (ST) which are broad categories of systems used to perform the selected mission. In general, MA- and ST-level rules act like metarules. They are not directly executed by the inference engine and are used to select the rules appropriate for a given mission area and system analysis. The next three levels Project (PR), Function (FN), and Main Specification (MS) contain rules that conclude functional or quality characteristics. PR- and FN-level rules determine the actual system and the functions it performs. The Main Specification (MS) is the quality specification associated with each function. MS-level rules determine most of the characteristic-to characteristic and characteristic-to-factor correlations.

2.2.1 Mission Area and Software Types

The Mission Area Decompositions (MAD) contained in Appendix A, summarize the major missions and system categories (software type) in five Air Force mission areas; Armament, Avionics, Missile-Space, C^3, and Force-Mission Management. Representative Air Force systems are listed under the software type as examples of systems associated with a specific software type. The actual number of fielded systems fitting under a given software type can be extensive, so a generic example called a generic project was developed based on a composite of the representative systems. The generic project allocates the major mission functions performed by a software type into a notional design or architecture, which can be used as a starting point for quality evaluations.

Missions are broad statements of the responsibilities and functions of an organization or system. Several sources such as Air Force System Command Regulations, Air Force product division fact books, A-Level Specifications, and Armed Forces Communications and Electronics Association (AFCEA) course material on command centers, were unsuccessfully used to develop mission definitions that could be categorized into distinct groupings of software systems. The alternative was to organize the mission areas according to the Air Force product divisions. Although somewhat arbitrary, it was a reasonable and fast approach for partitioning the mission and software domains.

The MAD builds on work done by Boeing Aerospace Corporation for the Software Test Handbook, RADC-TR-84-53, Vol II and reflects the mission responsibilities of the Air Force product divisions. This is especially true for the Armament, Avionics, Missile-Space, and C^3 mission areas which match the missions of the Armament Division, Eglin AFB, the Aeronautical Systems Division (ASD), Wright-Patterson AFB, the Space Systems Division (SSD), Los Angeles AFB, and the Electronic Systems Division (ESD), Hanscom AFB. The Force-Mission Management mission area is the only one that is not closely aligned with a product division and it includes systems that are not specific to one mission area. However, since Force-Mission Management systems tend to be command and control related, they have characteristics very similar to those of a C^3 system. Therefore, Force-Mission Management is addressed under the C^3 mission area reducing the mission areas to four.

Boeing categorized systems for the Armament Mission area into three software types; Threat Systems, Missile Systems, and Scoring Systems. Threat systems include defensive and offensive radar systems, targeting and tracking systems, and electronic countermeasure systems. Missile Systems are typically air-to-air and air-to-ground missile systems while Scoring Systems include proximity detectors for projectiles used in aerial gunnery training. After a review of Armament Division, the original software types were changed to Range Systems, Guided Weapon Systems, and Sensor Fuzed Weapons. This decomposition better reflects the system responsibilities of the Armament Division while eliminating the potential confusion between Armament missile systems and Missile-Space missile systems.

The Avionics mission area decomposition in Boeing's Test Handbook was mostly unchanged. The five software types, Avionics, Test Equipment, Air Crew Training Devices, Flight Controls, and Reconnaissance and C^3 were retained and only minor name changes were made. Avionics (software type) was renamed Aircraft Software, Reconnaissance and C^3 was renamed Reconnaissance and Electronic Warfare, and Air Crew Training Devices was renamed Training Simulator Software. These changes were made for clarity sake or to better reflect the current mission responsibilities of ASD.

Boeing's work under the C^3 mission area was based solely on the mission functions of the Strategic Air Command (SAC) and no software types were addressed. To give a more representative view of the Air Force C^3 mission area, four software types were created based on the mission responsibilities of ESD: Strategic C^3 systems, Tactical C^3 systems, Intelligence Systems and C^3 Countermeasures, and Advanced Decision Systems.

The Missile-Space mission area was changed and expanded. Boeing divided Missile-Space into Ground Control and Communications, Prelaunch Checkout and Launch Systems, Launch Vehicle Systems and Space Vehicle Systems. Based on an analysis of the Ballistic Missile Office and Space Division missions, the Prelaunch Checkout and Launch Systems software type was deleted since many of its mission functions fit under Launch Vehicle Systems. Ground Control and Communications was renamed Satellite Communications and Ground Control. This reflects the satellite emphasis in communications, navigation and meteorological support. A Defense and Surveillance Systems software type was added to accommodate mission functions relating to the Strategic Defense System (SDS).

2.2.2 Projects and Functions

Each software type at the PR level is associated with a notational architecture called a generic project. It shows a top-level design with its software configuration items or components. For example, a typical command and control system might have a communications component, message handling component, data base manager component, and user interface component. Each component performs functions whose attributes determine the component`s quality requirement or goal. Components in the generic project are assigned default quality goals based on expert validation. These default values are assumed to represent typical quality objectives for a software type. In ASQS, components (and their supporting rule base) can be copied from the generic project to a user's project if applicable.

Ideally, a component functional decomposition could be developed for a generic project which has a quality allocation extending down to the lowest defined function. For every function, there would be unique attributes that determined its quality goal to allow the user of ASQS added flexibility to find and copy a generic function and incorporate it into his functional decomposition. Conceptually this approach seemed feasible, but it was not possible to capture supporting quality data at such a level of detail and create function unique rules.

Functional attributes that might impact quality could only be estimated at a system or component level and even at this level, the data rarely supported more than a subjective estimate. Attempting to make quality assessments at lower levels of a functional decomposition was much too arbitrary to yield meaningful results. For this analysis, software requirements specifications and system design documents were used to identify the lower level functional characteristics. Some decompositions went as far as the fourth and fifth levels, but only to support a quality estimate at the component level. Several of the system or software requirements specifications used for the MAA explicitly stated the system quality requirements. The Worldwide Military Command and Control System (WWMCCS) Information System (WIS) and the Navy's Ocean Surveillance Information System (OSIS) specifications addressed most of the quality factors contained in the Framework. However, in reviewing the design documentation, it was not clear on how the specified quality requirements were allocated and implemented in the software design.

2.3 Knowledge Acquisition

Knowledge acquisition was performed using expert interviews combined with document research. Three experts were used with five to fifteen years experience in system acquisition and development. Their program background included the Air Force's Granite Sentry program, Intelligence Data Handling System (IDHS), Defense Satellite Communications System (DSCS), and Worldwide Military Command and Control System (WWMCSS) and the Navy's Ocean Surveillance Information System (OSIS) and Global Positioning System (GPS).

All the experts were given an overview of the Software Quality Framework, ASQS, and the ASQS context tree at the beginning of the interview process and attempted to describe software quality within the context of the SQF. None of the experts used the formal quality specification process in their respective programs. Correlation between the Framework and expert data had to be based on interpretation or extrapolation. Review of actual specification documents found no strong link between the SQF and the design process. The OSIS and WWMCCS specifications addressed the quality factors but this reflected more a compliance with a document format requirement rather than an analysis based on the SQF specification process.

2.3.1 Domain Experts

Two domain experts, one for Satellite Systems were selected to support the functional and quality analyses in the Missile-Space and C^3 mission areas. Their role was to help develop mission functional profiles, supplement the research data derived from the specifications, and help derive rules and queries. Other experts were used on a limited basis to assist with the functional decompositions in the remaining three mission areas.

The availability of qualified domain experts is an issue when dealing with any large domain. In this analysis, beside the number needed, the type of experience was critical. Many of the available experts had experience based on an acquisition or end-user viewpoint which was adequate for defining major mission functions, but unsatisfactory for understanding the detailed software aspects of a system. The experts used in this study generally addressed system quality issues from an A-specification level therefore, most of the detailed software characteristic data came from published books, journals, or proceedings. This information was relevant to the domain but lacked a the practical and focused input of an expert which meant the knowledge engineer assumed much of the expert's responsibility for identifying major domain concepts and reasoning chains.

2.3.2 Staffing

Effective knowledge acquisition requires the knowledge engineer to be familiar with the system under review. This helps make the entire process more productive from interviews to rule development. From the beginning of the task, one knowledge engineer should have been assigned for each mission area. Even with only two mission areas, the amount of data to be captured and analyzed was substantial for one person. It was not until the latter half of the task that the staffing was increased to two knowledge engineers.

2.3.3 Domain Considerations

The size of the domain was an early issue. Trying to capture software quality data for five Air Force mission areas is a sizeable task from a knowledge acquisition standpoint. As we proceeded with the functional decompositions, it became apparent that each mission area had a number of different software systems with different software quality concerns. The size problem was compounded by trying to find experts who could authoritatively address the various systems. The compromise was to focus the mission analysis in two mission areas, Missile-Space and C^3. In addition, specific software types were selected for study. For Missile-Space it was a satellite system and for C^3 it was an intelligence system.

Three representative system and/or software specifications were used during the analysis, the Multiple Satellite System (MSS), the Ocean Surveillance Information System (OSIS), and the World Wide Military Command and Control System (WWMCCS). Each was reviewed to determine top-level mission functions as well as software characteristics. Other system specifications and information were also used to provide supporting functional or characteristic data to include the Strategic Defense System (SDS) Phase I architecture studies, the Granite Sentry System Specification, and the Forward Area Air Defense Command, Control, and Intelligence (FAAD C^2I) Design Plan. Although functional decompositions were developed for each mission area, the emphasis was placed on the Missile-Space and C^3 mission areas.

2.3.4 Quality Factor Definitions

Software quality is an acknowledged concern but not a well-defined discipline. Like specification methodologies used to describe software systems, the quality framework provides a high-level abstraction mechanism to enhance the understanding of software quality issues. The definition and precision to completely describe a software quality design remains to be fully developed. A need for software quality was acknowledged by domain experts we interviewed and in some cases it could be addressed within the context of the framework definitions for some of the more commonly known factors such as reliability. Other factor definitions were subject to interpretation which made it difficult to determine the specific factor being impacted. For example, some see flexibility and expandability as measuring the same qualities while Others define integrity as data integrity and not security.

2.4 Knowledge Representation

2.4.1 Rules and Queries

The rules fall into two general classes. The first class of rules (Class I) are those that directly relate a functional or software characteristic(s) to one or more of the 13 software quality factors. The second class of rules (Class II) are those that relate one set of functional or software characteristics to another set. These two classes of rules provide a reasoning chain from software functions to quality factors.

Each rule contains parameters whose value must be known before the rule will execute. If ASQS is unable to determine parameter values from its known facts then it queries the user for the missing data. In general, the Class I rules would not have queries associated with their parameters. The Class I parameters would be determined by the applicable Class II rules.

The final rule and query set developed from the Mission Area Analysis is not contained in this report, but will be included in the ASQS Software Product Specification being developed by Dynamics Research Corporation. All rules in the ASQS knowledge base are cross-referenced to this document.

III. Intelligence System Software Analysis

3.0 Scope

This analysis is applicable to the Command, Control and Communications (C^3) mission area and the software type Intelligence Systems and C^3 Countermeasures.

3.1 Intelligence System Software Description

Intelligence system software is assumed to include all those functions performed to collect, process, evaluate, disseminate, and display intelligence data (Collection in this instance means to receive raw data from other systems or sensors that are external to the intelligence system). For the purpose of this analysis, intelligence systems are divided into two categories, strategic and tactical.

Strategic intelligence systems include those that store historical intelligence data, perform trend assessment, or help make intelligence estimates or predictions. Many are used as a source for planning new weapons development programs, developing strategic doctrine, or conducting technology assessments. They contain large encyclopedic data bases that are slowly changing and operate in a non- to near-real time mode. Strategic intelligence systems have minimal communication or security software requirements because they generally operate in highly secure facilities with no external communication interfaces.

Tactical intelligence systems are service specific, focus on a theater of operations, operate in a near real time mode and can be fixed or mobile. They generally have a communication segment, a sizeable data base, a complex man-machine interface, some type of security requirement and perform a specialized application (e.g., signal intelligence correlation). Tactical intelligence systems will often have multilevel security considerations driven by multiple data classifications, varying user clearances and communication interfaces with other secure and non-secure systems.

The following general rules can be derived from above:

Rule GE.1: IF Intelligence-Mission AND Tactical-System THEN Time-Critical

Rule GE.2: IF Intelligence-Mission AND Strategic-System THEN Not-Time-Critical

Query: Is the intelligence system considered a tactical system or strategic system?

This analysis uses a fixed tactical intelligence system as the basis for all functional and software quality assumptions.

3.2 Software Quality Factor Applicability

The first step is to determine if any of the thirteen software quality factors apply to intelligence system software. If factors are found to be non-applicable, this information will be used to write mission-level rules that will eliminate factors from further consideration based on the selection of the mission area and software type.

The principal mission of the tactical intelligence system is to receive, process and disseminate timely information on mobile targets of interest. This information is provided to Air Force and other services at all levels of command. A key aspect of a tactical intelligence system is its ability to process data, through both automatic and analyst interactive means, in near real time.

Typical inputs for the system come from external reports, queries, messages, and related data, and local analyst workstation interaction. These inputs are processed in both automatic and manual modes. Those inputs from external sources that require immediate attention are processed automatically by the system software. For example, the correlation of contact reports to existing tracks. Other automated system capabilities could include generating message processing summary reports or alert filtering.

A tactical intelligence system usually supports multiple users logged on to alphanumeric and graphic workstations. Inputs are entered through a combination of keyboard keystrokes, hard and soft function keys, and graphic input devices. Tasks performed include reviewing automatic processing results, generating summary reports, updating graphic displays, word processing, and accessing and updating a data base. External inputs can automatically trigger alerts to multiple analysts in the event of a change in contact reports or other significant external data. System outputs include target data updates, command support information, administrative reports, graphic displays, hard copy reports, and general system status data.

Given this system description, an attempt can be made to form some initial assumptions on the applicability of the quality factors to a tactical intelligence system by first examining those quality factors that make up the attribute of software Performance.

Efficiency. Efficiency is an important quality goal of intelligence software since a tactical intelligence system often processes several types of data to include human intelligence (HUMINT), communication intelligence (COMINT), electronic intelligence (ELINT), imagery, and general tactical reports. During normal operations this represents a large volume of data to process which increases considerably during a crisis situation. Bottlenecks and throughput become critical issues. Since it is not economically feasible to run every intelligence system on a CRAY computer, the alternative is to develop efficient software algorithms and routines that maximize the supporting hardware.

Integrity. Integrity is necessary because of its close relationship to security. Although many intelligence systems run in a system high mode where physical and personnel security measures play large roles, increased demands for DoD-wide information access and transfer are requiring systems with multi-level security software solutions. This increases the integrity requirements of the software.

Reliability. Reliability is a concern since tactical intelligence systems perform mission critical functions round-the-clock, reliability is a concern.

Survivability. Survivability is necessary since intelligence systems, because of their mission critical nature, must be able to operate in a degraded mode.

Usability. Usability is a concern.since intelligence systems operate in a near real time environment sometimes in crisis situations. Experts are not always available which means less experienced people must be able to run the system with a minimal learning curve.

Next, are the Design quality factors

Correctness. System security is a major driver in tactical intelligence systems. The design of system security measures for the software should be simple and small to allow for careful checking of its accuracy. This implies a need for correctness.

Maintainability. Maintainability is a concern because of an intelligence system's mission-critical nature, downtime or degraded operation due to software failures should be minimized. A maintainable design helps quickly isolate software problems and reduce the amount of downtime or degraded operation.

Verifiability. Intelligence systems help make operational assessments or tactical decisions based on some form of automatic or manual data analysis. If the system goes down due to a software failure, any fix must be quickly tested to ensure the correctness and accuracy of its operation.

The Adaptation factors.

Expandability. Expandability is applicable because intelligence systems have long life cycles over which several upgrades are made to enhance or improve existing mission capabilities.

Flexibility. Tactical intelligence systems perform highly specialized missions with security considerations th :t limit its value or use outside the intelligence domain. However, within the intelligence community, there is a trend to upgrade current systems to take advantage of new technologies or fulfill broader missions. This implies a need for more flexible software designs.

Interoperability. A design that reflects some level of software interoperability is required since a tactical intelligence system often must communicate with other intelligence or C^3 systems via satellite, long-haul packet network (e.g., Defense Data Network), or other types of communication nets. Software designs need to accommodate inter-system interfaces for increased DoD information access and transfer and greater operations-intelligence integration.

Portability. There is a need in the intelligence community to promote commonality and technology transfer to improve the Air Force's capability for monitoring and managing time-critical intelligence data. To achieve this goal, more software designs emphasize portability through the use of Ada or secure operating systems that run on widely used hardware (e.g., VMS on DEC).

Reusability. Tactical intelligence systems employ common communication, message processing, and data base designs that are strong candidates for reusability. Since reliability is a prime concern, new intelligence software would benefit from the use of operationally proven code.

In summary, all thirteen software quality factors are relevant at the system level for tactical intelligence systems. However, as the system is functionally defined, not all the factors are applicable to each function in software design. The next step is to take a closer look the tactical intelligence system functions to determine what is the quality allocation.

3.3 Tactical Intelligence System Notional Architecture

The notional architecture for a tactical intelligence system is shown in Figure 3.1. The major components in the notional architecture are message processing; data fusion/correlation; situation assessment; user-system interface; system monitor and control; system software; collection management; and general considerations. Each component is described below.

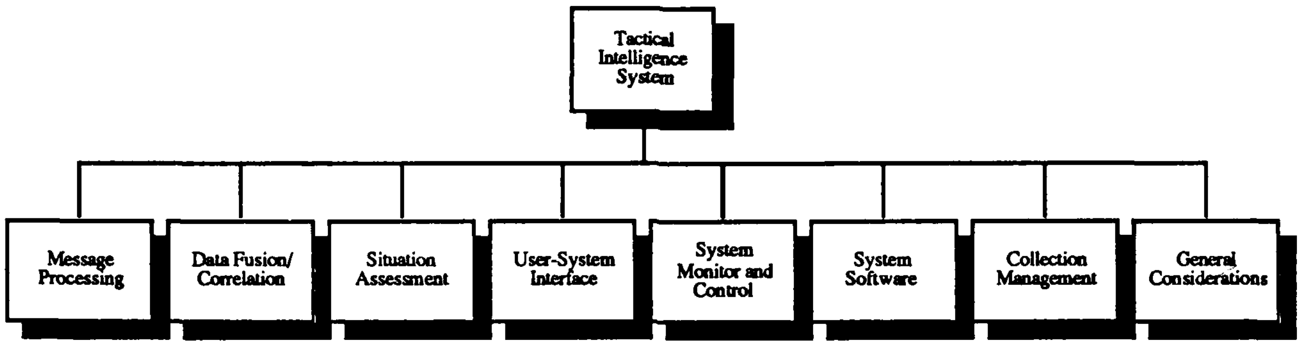

Figure 3.1. Tactical Intelligence System Notional Architecture

Message Processing. The message processing component is responsible for the receipt and preliminary processing of messages received from external communication circuits and for the assembly and transmission of messages generated as output to those circuits. The message processing function also contains the communications interface needed for transmission and receipt of data.

Data Fusion/Correlation. The data fusion component is a specialized application used to process intelligence data and perform assessments. This includes identifying contacts, processing ELINT data, and associating contacts to tracks.

Situation Assessment. The situation assessment component is a set of software tools used to evaluate threat situations and provide tailored intelligence products.

User-System Interface. The user-system interface component provides the man-machine interface such as the display of graphics and text. It also provides those interactive functions needed to develop input (e.g., word processing), manipulate files, or process alerts.

System Monitor and Control . The system monitor and control component provides system status monitoring, security services, operational configuration control, initialization and restart services, system management, performance monitoring, and network communications control.

System Software. The system software function provides data base management and operating system services.

Collection Management. The collection management component performs those functions that support the collection of intelligence data requests and sensor tasking.

General Considerations. General considerations is not an real architectural component, but rather a group of system attributes not attributed to a specific component. Such attributes include factors such as time constrained operations and error recovery.

3.4 Applicability of Quality Factors by Software Function

TABLE 3.1 provides representative mission functions that are performed by the components to give some insight into the operational requirements of the system that influence the quality allocation.

TABLE 3.1 Tactical Intelligence System Functions

Message Processing	Maintain communication status Message accountability Message receipt Message distribution Message generation Message validation Protocol processing
Data Fusion/Correlation	Sensor data analysis
Situation Assessment	Assessment aids Indications and warnings
User-System Interface	Text entry Text display Template displays Variable function keys Graphic processing and display

TABLE 3.1 Tactical Intelligence System Functions (Cont)

System Monitor and Control	System security Network management System administration System recovery and restart System initialization
System Software	Data base management Operating system services
Collection Management	Requirements collection Sensor tasking Report generation
General Considerations	Crisis/time constrained operations Error recovery Higher order language use

3.4.1 Message Processing

Message processing performs a series of functions that involve external and internal software interfaces. It includes providing the device drivers required by a communication interface controller as well as initiating the various protocol processing routines. This implies a need for interoperability and expandability. In addition, since communication protocols are somewhat standardized, the message processing software has a greater chance of being applied to other applications and a portable and reusable design is beneficial. During message processing, validation is performed for incoming and outgoing message traffic. It is the first and last security check for data that is sent to or received from external users or systems; therefore integrity is important. Since the message processing is a transparent function, usability is not applicable. Parsing of incoming messages and assembly of outgoing messages are functions that could cause data bottlenecks which requires an efficient software design.

3.4.2 Data Fusion/Correlation

The data fusion/correlation software is the heart of the automatic processing in a tactical intelligence system. It provides the ability to quickly assess and correlate incoming intelligence data generated by SIGINT, ELINT, COMINT, etc. sensor systems. Many of the routines are mathematically based and support various statistical computations. Their outputs are used to make operational assessment and confidence estimates so reliability and verifiability are important. Because many of the routines can be processing intensive, efficiency is also a concern. Data fusion algorithms are common to other intelligence applications and increases the need for portability, reusability, and interoperability in the software design.

3.4.3 Situation Assessment

The situation assessment software is a set of tools (assessment aids) used to analyze and respond to indications and warnings. It helps screen and filter intelligence data based on specific parameters defined by a user. These parameters trigger alerts (indications and warnings) which are flashed on a user's terminal. Filtering is a means of highlighting critical intelligence information used in making operational assessments and recommendations. Reliability, survivability, usability, correctness, and verifiability are important design concerns.

3.4.4 User-System Interface

The User-System Interface provides the man-machine interface for control of the text and graphic displays and provides the functions needed to manipulate, analyze, and send intelligence data. It must be highly usable to promote rapid and effective user responses Some level of security is enforced by the workstation software which implies a need for integrity. Expandability and flexibility are desired because of the different types of intelligent graphic and text display devices used by the system. In addition, changing display technology as well as evolving terminal display standards demand a flexible and expandable software design.

3.4.5 System Monitor and Control

The system monitor and control software provides the system security, software management utilities that maintains error and security logs, monitors and displays equipment status and CPU usage, manages process queues and performs system start, shutdown, and resource allocation. A primary attribute of this software is that it extends the security features of the system software to meet the system security requirements. This implies a need for integrity as well as correctness, maintainability, verifiability, and reliability. Security features often adversely impact the efficiency of a system so efficiency is a design concern. Many of the functions performed under system monitor and control involve some limited interface with a security or system administrator, therefore usability is a minor design concern. Similar to the system software, system monitor and control functions are essential to the basic operation of the system. It must operate at all times and support a degraded mode of operation so survivability is important.

3.4.6 System Software

The system software generally consists of non-developmental items such as the database manager and the operating system. They significantly influence the extent to which the quality factors will apply to the developmental software. Choosing an operating system that has been certified to the appropriate security level would likely decrease the need for integrity in the application software, an ideal situation from a quality viewpoint because the software would not only rate high in integrity, but also in correctness, reliability, verifiability, and maintainability. However, the number of off-the-shelf operating systems that are rated compliant for multilevel secure applications are very limited. The non-developmental software would perhaps contribute little to the software integrity in most tactical intelligence systems because efficiency, reliability, survivability are also impacted. Tactical intelligence systems store and process large amounts of data under near real time constraints using distributed architectures and they rely heavily on the operating system and the database manager to provide responsive, accurate and reliable support. If prioritized according to the mission requirements, tactical intelligence systems should emphasize reliability, integrity, efficiency, flexibility, and survivability for off-the-shelf or non-developmental software.

3.4.7 Collection Management

Collection management involves both data base and user-interface functions and provides critical mission support. Its software has attributes similar to those found in the user-system interface and system software, therefore there is a need for usability, reliability, survivability and correctness.

The Collection Management function monitors the status of data required by the system. It defines data requirements and will task the appropriate sensor(s) to acquire the required data. It will provide reports of the status of specific data collection through, and may contain algorithms to analyze alternate data sources to compensate/correlate for degraded sensor(s), erroneous sensor data, or conflicting sensor data, the latter typically being the result of fusion algorithms that may not adequately compensate for asynchronous sensor readings, or ambiguous sensor data.

3.4.8 General Considerations

General considerations are those system attributes that are not specific to any component. For example, the use of a high order language such as Ada might imply portability and reusability.

Time crisis/time constrained operations is obviously not confined to particular component but is considered a system attribute.

3.5 Mission-Oriented Rules and Queries

The following sections present sets of rules and associated questions that are focused on the intelligence mission area. A subset of the notional architecture and its functions were choosen based on the expert's knowledge of the subject or the feasibility of evaluating a functions quality attributes. In some cases, only the top level function could be evaluated (e.g. Message Processing), while others like System Monitor and Control, could be assessed at a lower functional level (e.g. System Security). The specific functions that were addressed are:

- System Security
- Message Processing
- User-System Interface
- Indications & Warning
- Collection Management
- Error Recovery
- Crisis/Time Constrained Operations
- General Considerations.

For each of the selected functions, a set of rules is presented that address many features within an intelligence system, but should not be considered exhaustive. The intent has been to provide a balanced mixture of depth and breadth within the rule set so that it is a viable building block for future expansion.

Each rule is presented in the following general format. The rule, itself, is expressed, followed by any associated questions that would be used during an ASQS consultation session. A short rationale explaining the rule is offered for validation of the conclusion(s) drawn.

The rules fall into two general classes. The first class of rules, Class I, are those that directly relate a functional or software characteristic(s) to one or more of the 13 software quality factors. The second class of rules, Class II, are those that relate one set of functional or software characteristics to another set. These two classes of rules provide a reasoning chain from software functions to quality factors.

Each rule contains parameters whose value must be known before the rule will execute. If ASQS is unable to determine parameter values from its known facts, it queries the user for the missing data. In general, the Class I rules do not have queries associated with their parameters. The Class I parameters are determined by the applicable Class II rules.

An estimate of confidence values was made for the rule conclusions when implemented in ASQS. These values range from 0 to 1, with .3 and below defined in this analysis as weakly indicated, greater than .3 and up to .5 defined as moderately indicated, and greater that .5 defined as strongly indicated. ASQS through its reasoning chain aggregates the confidence values and determines the factor rating.

3.6 System Security

The following rules apply to issues related to multi-level security. The first set of rules relate the security function to the quality factors. The questions associated with these rules are very general, and will normally not be the basis for quality factor evaluation in the security area because of their generality. Instead, the second set of rules, which relate features of systems to security, will be the basis for concluding that one or more of the Class I security rules apply.

3.6.1 Class I Rules

Rule SE.1: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-High, THEN it is strongly indicated that Efficiency-Is-Needed.

Query: None.
Discussion: "Trusted" software for systems with a rating of A or B must execute continuous control of data and resources within the system, including user access privileges and external communications interfaces. For systems with a high security requirement, a considerable percent of processing bandwidth is expended in performing the access control and audit processes, which implies that the security software must be highly efficient so that computational resources are available for performing mission-oriented applications. Substantial effort must be expended during the design phase so that the security software possesses a high degree of efficiency.

Rule SE.2: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND the System-Security-Requirements-Low, THEN it is weakly indicated that Efficiency-Is-Needed.

Query: None.

Discussion: "Trusted" software for systems with a rating of C must execute continuous control of data and resources within the system, including user access privileges and external communications interfaces. Considerable processor bandwidth is required to perform the access control checking and auditing, which implies that the security software must be efficient enough so that adequate processing capacity exists to perform mission functions.

Rule SE.3: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-High, THEN it is strongly indicated that Integrity-Is-Needed.

Query: None.

Discussion: The software function is considered to enforce a security policy if the function, alone or in combination with other functions, implements the security mechanisms through which the specific requirements of the security policy are achieved. A security policy is the set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. When a software function is responsible for enforcing a security policy, the software implementing the function is "trusted". "Trusted" software is directly responsible for detecting and repelling unauthorized attempts to access the system or its data . For systems with a high security requirement (an "Orange Book" rating of B or above), significant effort must be expended to detect and repel unauthorized access attempts.

Rule SE.4: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-Low OR No-System-Security-Requirements, THEN it is weakly indicated Integrity-Is-Needed.

Query: None.

Discussion: For systems with a low security requirement (an "Orange Book" rating of C) or for systems with no security requirement (an Orange Book rating of D), the security related software must still detect and repel unauthorized access attempts. For this level of security requirement, commercial "best practice" is usually acceptable, that is, the security features normally available within commercial software is adequate to enforce security requirements. No increased integrity factor rating is necessary arising solely from the system security requirements.

Rule SE.5: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND User-Interaction, THEN it is weakly indicated Usability-Is-Needed.

Query: None.
Discussion: "Trusted" interface software must be easy to use so that users are not frustrated or denied service because of complex or lengthy data input or verification procedures.

Rule SE.6: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-High THEN it is strongly indicated Reliability-Is-Needed.

Query: None.
Discussion: "Trusted" software must accomplish the function(s) it was designed to perform so that the security policy defined for the overall system is executed. Security-related software must perform as designed if the rules governing access to data are to be enforced. For systems that qualify for a rating of B or higher, significant trust is placed in the security enforcement mechanisms' ability to perform its defined function(s).

Rule SE.7: IF a Function-Enforces-Security-Policy(i.e., implements, or supports the implementation of) AND the System-Security-Requirements-Low THEN it is weakly indicated Reliability-Is-Needed.

Query: None.
Discussion: "Trusted" software must accomplish the function(s) it was designed to perform so that the security policy defined for the overall system is executed. Security-related software must perform as designed if the rules governing access to data are to be enforced. For systems that qualify for a rating of C or lower, trust is placed in the security enforcement mechanisms' ability to perform its defined function(s).

Rule SE.8: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND the System-Security-Requirements-High, THEN it is strongly indicated that Survivability-Is-Needed.

Query: None.

Discussion: "Trusted" software must accomplish the function(s) it was designed to perform so that the security policy defined for the overall system is executed. Security-related software must perform its access control and audit functions under all operating conditions, including periods when the system is missing resources due to failure or removal. For systems where a high level of mandatory protection (i.e., Orange Book Division B or A), special care must be taken to ensure that the security-related functions of a system operate in a continuous manner.

Rule SE.9: IF a Function-Enforces-Security-Policy (i.e implements, or supports the implementation of) AND System-Security-Requirements-High, THEN it is strongly indicated Verifiability-Is-Needed.

Query. None.

Discussion: It must be possible to verify the correct performance of the security-related function. The degree to which security verification must be accomplished depends upon the specific security criteria applied to the system, e.g. a formal rating level such as defined in the "Orange Book." Security-related software for systems with a division rating of A or B must be subjected to rigorous testing so that high confidence may be placed in its operation and performance. Systems that qualify for a rating of B and higher must be verified so that all flaws are "removed or neutralized". This level of verification exceeds that normally required for command and control software.

Rule SE.10: IF Function-Enforces-Security-Policy (i.e implements, or supports the implementation of) AND System-Security-Requirements-Low OR No-System-Security-Requirements, THEN it is weakly indicated Verifiability-Is-Needed.

Query: None.

Discussion: It must be possible to verify the correct performance of the security-related function. The degree to which security verification must be accomplished depends upon the specific security criteria applied to the system, e.g. a formal rating level such as defined in the "Orange Book." Security-related software for systems with a division rating of C or D, must be subjected to rigorous testing so that high confidence may be placed in its operation and performance. Systems that qualify for a rating of C or D are essentially subject to the same level of verification as that normally required for command and control software. Security requirements alone do not indicate an increased need for verifiability.

Rule SE.11: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-High THEN it is strongly indicated that Correctness-Is-Needed.

Query: None.
Discussion: Security-related, or "trusted", software must conform precisely to its specifications and standards as defined in the system security policy. The basis for trusting the software-implemented security mechanisms in their operational setting is derived from the extent to which the software implementation conforms to the specifications from which it is built. For systems with a high security requirement, a significant dependency is placed upon the correct operation of the security software (and hardware).

Rule SE.12: IF a Function-Enforces-Security-Policy (i.e implements, or supports the implementation of) AND System-Security-Requirements-Low, THEN it is weakly indicated Correctness-Is-Needed.

Query: None.
Discussion: Security-related, or "trusted", software must conform to its specifications and standards as defined in the system security policy. The basis for trusting the software-implemented security mechanisms in their operational setting is derived from the extent to which the software implementation conforms to the specifications from which it is built. For systems with a rating of C , the correct operation of the security software (and hardware) is essential to meeting system mission objectives.

Rule SE.13: IF a Function-Enforces-Security-Policy (i.e implements, or supports the implementation of) AND System-Security-Requirements-High THEN it is weakly indicated Maintainability-Is-Needed.

Query: None.
Discussion: Security-related software is normally tested to such a degree that few latent errors remain in the operationally released version. The degree to which this is true depends upon the security rating determined for the overall system--the higher the rating the fewer latent errors are acceptable. System security requirements are high if the system qualifies for an Orange Book rating above level C. Because of the rigor of testing and repair before operational acceptance, the need for latent error identification and correction is small.

Rule SE.14: IF a Function-Enforces-Security-Policy (i.e implements, or supports the implementation of) AND System-Security-Requirements-Low OR No-System-Security-Requirements, THEN it is strongly indicated that Maintainability-Is-Needed.

Query: None.
Discussion: Security-related software is normally tested to such a degree that few latent errors remain in the operationally released version. The degree to which this is true depends upon the security rating determined for the overall system--the higher the rating the fewer latent errors are acceptable. Systems with a low security requirement (for example, levels C and D in the "Orange Book") may possess latent defects and must be repairable rapidly and easily so that operational systems may be corrected in a rapid manner.

Rule SE.15 : IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-High, THEN it is weakly indicated Expandability-Is-Needed.

Query: None.
Discussion: Changes to security-related software are usually only made if the security policy is changed. At present, the process of security certification and accreditation is so time and resource intensive that changes to an approved system security policy are infrequently made. Frequently, policy changes of any significance require major modifications of hardware and/or software, reflecting the close coupling between all the enforcement mechanisms that implement the security functions. The higher the security rating of the system, the more intensive the coupling, with the net effect that expandability becomes less practical within the existing state-of-the-art.

Rule SE.16: IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of) AND System-Security-Requirements-Low OR No-System-Security-Requirements, THEN it is weakly indicated Expandability-Is-Needed.

Query: None.
Discussion: Changes to security-related software are usually only made if the security policy is changed. At present, the process of security certification and accreditation is so time and resource intensive that changes to an approved system security policy are infrequently made. For systems with a security rating of C or below, changes are more technically and economically feasible because the degree of reverification involved is lower than for more formally verified systems (those with a rating of B and higher).

Rule SE.17 : IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of), THEN it is weakly indicated Flexibility-Is-Needed.

Query: None.
Discussion: Software which implements security mechanisms is usually intended to perform only those unique functions (e.g., user login). Because of the need to "trust" these software mechanisms, they are highly specialized and normally would not be applied to accomplish non-security functions.

Rule SE.18 : IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of), THEN it is weakly indicated Interoperability-Is-Needed.

Query: None.
Discussion: Software which implements security mechanisms must be able to interact with other software components. For example, the secure file control mechanism must be able to respond to application program/user requests for data, as well as requests for data from other systems.

Rule SE.19 : IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of), THEN it is weakly indicated Portability-Is-Needed.

Query: None.
Discussion: Software which implements security mechanisms is usually tightly bound with its operational hardware environment (see Rule SE.12 discussion). With the present state-of-the-art, only a limited degree of portability can be achieved, especially with "primitive" security functions (e.g. file access control) that are normally implemented within the operating system. Security functions that can be "layered" on more basic functions will admit a higher degree of transferability to other platforms.

Rule SE.20 : IF a Function-Enforces-Security-Policy (i.e., implements, or supports the implementation of), THEN it is strongly indicated Reusability-Is-Needed.

Query: None.
Discussion: Security-related software is independent of the specific intelligence application in the sense that common security requirements apply to many different intelligence mission systems. The requirements for verifiability and integrity result in the availability of extensive design and

performance data on security software elements, with the consequence that the software can be applied in many different specific applications where the policy enforced by the software/hardware combination remains constant.

3.6.2 Class II Rules

The following rules/questions provide a basis for determining whether the security Class I rules apply to a particular software function. These would be used during an ASQS consultation session and can be applied/asked in any order, except that the rules for determining the general system security level should be evaluated first.

3.6.2.1 Determining the Level of System Security Requirements

The rules given below may be used to determine the level of security required by the system. The rules/questions result in an assessment that High, Moderate, Low, or Not Applicable security requirements are indicated. A score of High implies that the probable security requirements equate to an "Orange Book" division rating of A (Verified Protection) or B (Mandatory Protection). A score of Low implies that the probable security requirements equate to a division rating of C (Discretionary Protection). A score of Not Applicable implies that there are no special requirements and equates to a division rating of D (Minimal Protection). The latter rating division is reserved for those systems that have been evaluated but do not meet the security evaluation criteria for a higher evaluation class.

Given that security issues apply at all, a binary (High/Low) security assessment is used here because such a division allows for some granularity in requirements without immersing the user in the full details of a security requirements analysis. Further, the Orange Book evaluation criteria require a significant difference in systems between each rating division (A-D). The largest "delta" in requirements occurs between divisions B and C, thus the use of two general ratings (High/Low) using the division distribution stated is consistent with the Orange Book formal criteria.

With these observations in mind, the rules presented here may be used to "painlessly" determine the general security aspects of the users system of interest. The rating assessment is based on the computer security requirements evaluation methodology developed by Mr. H. O. Lubbes, Senior Manager, Security Policy, Code 32-123, Space and Naval Warfare Systems Command (SPAWAR). The method, which is intended for use by acquisition managers, requires that certain risk factors be quantified and evaluated. A matrix that relates risk evaluation outcomes to Orange

Book division/class ratings is used to determine the system security level (A-C). The risk factors considered are: local processing capability, communications path, user capability, and data exposure (difference between clearance of lowest-cleared user and classification of the most-sensitive data).

The matrix recommends a B1 or higher level under the following conditions:

-Data exposure=1 and System risk>6

-Data exposure =2 and System risk >3

-Data exposure >2.

Data exposure is calculated as the difference between the level of the least-cleared user of the system and the maximum level of data processed by the system. Levels range from 0 (uncleared/unclassified) to 7(top secret-special background investigation with more than one compartment/top secret with two or more categories). Thus, the data exposure index ranges from 0 to 7. The system risk index ranges from 3 (representing a user capability for output-only on a store and forward network using a receive-only or fixed function terminal) to 9 (representing a user with full programming capability on a direct connect or interactive network using a programmable device/terminal). Here, the general assumption is that the system of interest requires that users (people) access sensitive data via some functional equivalent of a terminal.

Define data exposure as high if the data exposure index exceeds 1, that is, if there are users on the system with clearances at least 2 levels lower than the classification of the most sensitive data processed. Thus, the presence of secret-cleared users on a system that processes top secret data, would constitute a high level of data exposure risk.

Define data exposure as very high if the data exposure index exceeds 2, that is, if there are users on the system with clearances at least 3 levels lower than the classification of the most sensitive data processed. Thus, the presence of uncleared users on a system that processes secret data, would constitute a very high level of data exposure risk.

Define data exposure as low if the data exposure index is less than 2, that is, if the maximum classification of data processed on the system is no more than one level higher than the lowest clearance level of any user of the system. Thus, if a system processes top secret compartmented data and all users have a clearance of top secret with special background investigation and authorization for at least one compartment, the data exposure risk is rated as low.

Define system risk as high or Integrity-Is-Moderately-Indicated if the system risk index is greater than 3, that is, if there are users who:

-Have output-only access from any type of terminal using any type of duplex connection (such as, a store and forward network, via direct connection, or on an interactive network);

-Have transaction processing access from any type of terminal using any type of (duplex) connection;

-Have any type of access via a programmable device (e.g., workstation) using any type of connection.

Define system risk as very high or Integrity-Is-Strongly-Indicated if the system risk index is greater than 6, that is, if there are users who:

-Have output-only access from a programmable device via direct connection or an interactive network;

-Have transaction processing access from a fixed function interactive terminal via direct connection or an interactive network;

-Have transaction processing access from a programmable device using any type of (duplex) connection;

-Have full programming access from an interactive terminal or a programmable device.

Define system risk as low or Integrity-Is-Weakly-Indicated if the conditions described above are not met for users on the system of interest.

The following rules codify the definitions presented above. We assume that the system security requirements are initially not applicable, unless modified as a result of applying one or more of these rules.

Rule SS.1.1 IF Data-Exposure-Low AND System-Risk-Low THEN System-Security-Requirements-Low.

Query: None.
Discussion: This is case one from the Lubbes matrix (data exposure=1 and system risk<7).

Rule SS.1.2 IF Data-Exposure-Low AND System-Risk-High THEN System-Security-Requirements-Medium.

Query: None.
Discussion: This is case one from the Lubbes matrix (data exposure=1 and system risk>6).

Rule SS.2.1 IF Data-Exposure-Medium AND System-Risk-Low, THEN System-Security-Requirements-Medium.

Query: None.
Discussion: This is case two from the Lubbes matrix (data exposure=2 and system risk>3).

Rule SS.2.2 IF Data-Exposure-Medium AND System-Risk-High, THEN System-Security-Requirements-High.

Query: None.
Discussion: This is case two from the Lubbes matrix (data exposure=2 and system risk>3).
Rule SS.3.1 IF Data-Exposure-High, THEN System-Security-Requirements-High.

Query: None.
Discussion: This is case three from the Lubbes matrix (data exposure>2).

3.6.2.2 Data Exposure (DE)/System Risk (SR)

The next set of questions may be used to determine the low/high/very high settings for the data exposure level and the system risk level, again based upon the definitions above. The first series (DE.1-DE.3) establishes the data exposure level. The second series (SR.1-SR.7) determines the system risk level.

Rule DE.1: IF Sensitive-Data OR Classified-Data THEN Data-Exposure-Low.

Query: Will any sensitive or classified data be processed on the system?
Discussion: If yes, data exposure level is set as low. If no, security issues are not relevant, and no further questions are necessary. Set system security requirements as not applicable.

Rule DE.2: IF Data-Exposure-Low AND Lower-User-Clearance THEN Data-Exposure-Level-Medium.

Query SE.2.1: Will any user have a clearance that is less than the classification of data processed on the system (e.g. secret-cleared users on a system with top secret data)?
Discussion: If yes, set data exposure level as high . If no, data exposure remains at low.

Rule DE.3: IF Data-Exposure-Low AND Significantly-Lower-User-Clearance THEN Data-Exposure-High.

Query: Will any user have a clearance that is significantly less than the classification of data processed on the system (e.g. uncleared users on a system with secret data)?
Discussion: If yes, set data exposure level as very high . If no, data exposure remains as high.

Rule SR.1: IF Output-Only-Access AND Two-Way-Communication THEN System-Risk-High.

Query: Will users have output-only access from terminals connected to the system via any type of duplex (two-way) communications (such as, direct access, access via a LAN or a DDN circuit)?
Discussion: If yes, set system risk level as high. If no, system risk remains at low.

Rule SR.2: IF Transaction-Processing-Access AND Two-Way-Communication THEN System-Risk-High.

Query: Will users have transaction processing access from any type of terminal using any type of duplex (two-way) communications?
Discussion: If yes, set system risk level as high. If no, system risk remains as set.

Rule SR.3: IF User-Programmable-Device THEN System-Risk-High.

Query: Will users have any type of access from a programmable device using any type of communications?
Discussion: If no, system risk remains as set and no more questions are necessary.

Rule SR.4: IF Output-Only-Access AND User-Programmable-Device AND Direct-Connect OR Interactive-Network THEN System-Risk-High.

Query: Will users have output-only access from a programmable device using a direct connection or an interactive network (such as and LAN or a DDN circuit)?
Discussion: If yes, set system risk level as very high, and no more questions are necessary.

Rule SR.5: IF Transaction-Processing-Access AND User-Fixed-Function-Device AND Direct-Connect OR Interactive-Network THEN System-Risk-Very-High.

Query: Will users have transaction processing access from a fixed function interactive terminal using a direct connection or an interactive network (such as and LAN or a DDN circuit)?
Discussion: If yes, set system risk level as very high, and no more questions are necessary.

Rule SR.6: IF Transaction-Processing-Access AND User-Programmable-Device AND Two-Way-Communication THEN System-Risk-Very-High.

Query: Wil! users have transaction processing access from a programmable device using any type of duplex (two-way) connection?
Discussion: If yes, set system risk level as very high, and no more questions are necessary.

Rule SR.7: IF Full-Programming-Access AND Interactive-Terminal OR User-Programmable-Device THEN System-Risk-Very-High.

Query: Will users have full programming access from an interactive terminal or a programmable device?
Discussion: If yes, set system risk level as very high, and no more questions are necessary. If no, system risk remains as set, and no more questions are necessary .

3.6.2.3 Determining the Impact of Security Requirements on Quality

The following rules/questions use the software features/functions to determine whether any security-related processing occurs within the system of interest. These rules, in combination with the security requirements evaluation are translated into quality factor assessments using the Class I rules given above.

Rule SE.1.2 IF the function Controls-Access-To-Data OR Control-Access-To-Users, THEN Function-Enforces-Security-Policy (FESP).

Query: Does the function determine or control access between users and data or other users?
Discussion: Access mediation is a major function of the trusted computer base, and has a key role in implementing the security policy.

Rule SE.2.2 IF the function Creates-Data-Files OR Deletes-Data-Files, THEN Function-Enforces-Security-Policy

Query: Does the function create or delete data files?
Discussion: The security policy governs the creation/destruction of objects, in particular, data files. The software that does this must implement the governing policy.

Rule SE 3.2 IF Changes-User-Security-Characteristics, THEN Function-Enforces-Security-Policy.

Query: Does the function change or alter users security access characteristics, i.e., password, clearance, privileges?
Discussion: The security policy governs this process.

Rule SE.4.2 IF the function Changes-User-Security-Labels OR Changes-Data-Security-Labels, THEN Function-Enforces-Security-Policy.

Query: Does the function change security labels for users or for data?
Discussion: The security policy governs this process.

Rule SE.5.2: IF the function Controls-Data-Exchange-Between-System-Components, THEN Function-Enforces-Security-Policy.

Query: Does the function control data exchange between system components?
Discussion: The security policy governs this process.

Rule SE.6.2: IF the function Labels-Human-Readable-Output, THEN Function-Enforces-Security-Policy.

Query: Does the function perform or control the labelling of human-readable output (e.g., messages, displays, graphics, reports)?

Discussion: The security policy governs this process.

Rule SE.7.2: IF the function Controls-Security-Level-Terminal-User, THEN Function-Enforces-Security-Policy.

Query: Does the function change or control a users security level during a terminal session?
Discussion: The security policy governs this process.

Rule SE.8.2: IF the function Records-Access-Trail between users and data (or other users or interfaces), THEN Function-Enforces-Security-Policy.

Query: Does the function record an audit trail of accesses between users and data (or other users or interfaces)?
Discussion: The security policy governs this process.

Rule SE.9.2: IF the function Performs-Recovery OR Performs-Restart following a failure OR other discontinuity, THEN Function-Enforces-Security-Policy.

Query: Does the function perform recovery and/or restart following a failure or other discontinuity?
Discussion: The security policy governs this process.

Rule SE.10.2: IF the function performs System-Security-Admin-Services, THEN Function-Enforces-Security-Policy.

Query: Does the function perform security-related administrative services?
Discussion: The security policy governs this process.

Rule SE.11.2: IF the function performs a User-Logon-Service, THEN Function-Enforces-Security-Policy.

Query: Does the function perform a user log-on service?
Discussion: The security policy governs this process.

Rule SE 12.2: IF the function Mitigates-Denial-Of-Service, THEN Function-Enforces-Security-Policy.

Query: Does the function mitigate denial of service?
Discussion: The security policy governs this process.

3.7 Message Processing

These rules are primarily focused on functions performed by an intelligence system to receive, profile, store, and disseminate record message (e.g., AUTODIN/DSSCS) traffic.

3.7.1 Class I Rules

Rule MP.1: IF Automatic-Mode, THEN it is weakly indicated Integrity-Is-Needed, Reliability-Is-Needed, Correctness-Is-Needed,Verifiability-Is-Needed.

Query: None.
Discussion: Within an intelligence system, any function performed substantially by the software must perform as expected on each instance the function occurs (reliability). The processing performed must be correct, as determined through rigorous verification. If data security is involved, the quality of high integrity is essential.

Rule MP.2: IF Automatic-Mode, THEN it is weakly indicated Efficiency-Is-Needed, Usability-Is-Needed, Maintainability-Is-Needed, Expandability-Is-Needed.

Query: None.
Discussion: Automatic processes will generally be provided the resources necessary to accomplish their activities consonant with mission objectives (i.e., parsing and distributing a message within N seconds of receipt for each AUTODIN/DDN priority level). Such processes must be highly useable to the extent that operator/user participation is involved (e.g., correcting an invalid routing address, or determining the distribution for an unusual message). Because automatic processes encapsulate, generally one policy or approach, the software must be amenable to reasonable change as policies or requirements change.

3.7.2 Class II Rules

Rule MP.1.1: IF the function Receives-And-Store-Incoming-Data AND Processing-Unavailable, THEN Automatic-Process.

Query: Does the function receive and store incoming data during periods when the data cannot be processed upon receipt?
Discussion: Such a process is performing an automatic function of receiving and storing data pending processing by the unavailable processing unit (e.g., the message router is unavailable and incoming message traffic must be collected for later distribution).

Rule MP.2.1 IF the function Performs-Text-Processing OR Performs-Message-Text-Parsing OR Performs-Message-Profiling OR Performs-Message-Analysis THEN Automatic-Process.

Query: Does the function perform message text parsing, profiling, analysis, and/or routing?
Discussion: These functions are normally performed in an automatic mode by a message processing system.

Rule MP.3.1: IF the function performs Error-Detection-And-Correction, THEN Text-Analysis.

Query: Does the message handling function perform error detection and correction?
Discussion: Message handling systems generally perform some level of error detection and correction in an automatic mode, e.g., the use of EDC codes to prevent a garbled transmission.

Rule MP.4.1: IF the function Log-Messages OR Archive-Messages, THEN Function-Enforces-Security-Policy AND Automatic-Mode.

Query: Does the function log or archive a copy of all messages processed by the system?
Discussion: The logging function is normally an automatic process. Further, the message log file contains messages of differing classification. The security policy establishes the rules governing the message log. The logging software is a FESP component because it maintains the message log, and must be "trusted" to do it in accordance with the security policy.

Rule MP.5.1: IF the function performs Approved-Message-Release, THEN Function-Enforces-Security-Policy and Automatic-Process.

Query: Does the function perform message release for approved messages?

Discussion: Once the releasing official has approved the message for transmission, the transmission process is normally automatic. Generally, the security policy governs message release processing.

3.8 User-System Interface (US)

These rules are more general in nature, and focus on the interface between the software system and its human users. The rules center upon the concept that the human/machine interface, if any, is generally a critical function. The criticality of the interface to successful mission accomplishment is usually recognized in the term "user-friendly", as related to the system-user interface. For intelligence systems, the system-user interface is the communications pathway through which data processed by the software and hardware passes to the intelligence analyst (user). Because the analysts' assessments and actions are based, at least in part, on system processed data (e.g., messages profiled/distributed, historical database holdings, initial automated assessments) the form/fit/function of the system-user interface is an important element of the system. It is critical to mission accomplishment in the sense reflected here.

3.8.1 Class I Rules

Rule US.1: IF the function is a Critical-Function, THEN it is weakly indicated Reliability-Is-Needed, Survivability-Is-Needed, Usability-Is-Needed, Correctness-Is-Needed.

Rule US.2: IF Complex-And-Significant-Data AND Basis-For-Actions-And-Assessments THEN it is weakly indicated Usability-Is-Needed

Query: None.

Discussion: A critical system function is one that is essential to achievement of system mission objectives. Such a function must always be performed (reliability), even in a degraded mode of operation (survivability). Critical functions normally involve some level of operator/user involvement, so that usability is a significant issue. Because of the essential nature of the function, it must be possible to verify that the function is performed in a correct manner (correctness). The need for correctness implies a need for verifiability, but the function and its implementation in software are more important than the ease by which the implementation can be verified. That is, verifiability (or ease thereof) should not rate higher than the essential correctness of the implementation, even at the expense of requiring more work to perform the verification.

3.8.2 Class II Rules

Rule US.1.1: IF the function supports users who are "watch-standing", THEN Extensive-Human-Interaction.

Query: Does the function support users who are "watch-standing"?
Discussion: The watch officer support activities within an intelligence system place significant demands on the system-user interface. In particular, the officer acts based upon data presented to him via the system-user interface.

Rule US.2.1: IF Extensive-Human-Interaction AND Intelligence-Mission THEN the function is a Critical-Function AND a Basis-For-Actions-And-Assessments

Query: Does the function require extensive interaction with human users, such as analysts, watch officers, intelligence sensor specialists?
Discussion: Extensive user-system interaction in an intelligence-oriented mission implies that the system-user interface is a used as a basis for actions and assessments and performs a critical mission role.

Rule US.3.1: IF Alphanumeric-Workstation AND Graphic-Display, THEN Complex-And-Significant-Data.

Query: Does the workstation support both alphanumeric and graphic displays?
Discussion: The use of multi-media displays indicates that the system provides complex, and significant, data to its human users. For intelligence systems, the data is the basis for actions and assessments, therefore the user-system interface performs a critical function.

Rule US.4.1: IF the function must be self-explanatory, THEN Extensive-Human-Interaction.

Query: Does the function include an extensive "help" facility to aid users in the performance of their assigned tasks?
Discussion: A self-explanatory requirement implies that the user interface is extensive.

Rule US.5.1: IF Different-Classes-Of-Users, THEN the Extensive-User-Interface.

Query: Does the function support different groups of users, e.g., system administrator, intelligence analyst, who perform differing mission roles?
Discussion: For intelligence systems, the requirement to support different groups of users implies that the system-user interface must have customized elements, and therefore that the interface is extensive.

3.9 Indications and Warning (I&W)/Intelligence Analysis

Within an intelligence system (excluding dedicated sensor signal processing systems) there are two major functions: the quick analysis of near-real time data for I&W; and the longer-term continuing assessment of data (for trends identification and military technology capabilities determination). The rules that follow address this functional area by providing linkage from specific I&W/analysis activities to assessments concerning the quality factors.

3.9.1 Class I Rules

Rule IW.1: IF the function performs or contributes to the generation of Near-Real-Time-Intelligence-Assessment (I&W/intelligence assessment), THEN it is strongly indicated Reliability-Is-Needed and Survivability-Is-Needed, Usability-Is-Needed, Correctness-Is-Needed, Expandability-Is-Needed, Interoperability-Is-Needed. It is moderately indicated Efficiency-Is-Needed, Maintainability-Is-Needed, and Flexibility-Is-Needed.

Query: None.
Discussion: Because the data are the basis for actions and assessments, the processes that manipulate the data must perform correctly and accurately. Speed and reliability are major considerations, so that usability is significant. Because intelligence systems must accomodate new or upgraded sensor sources, expandability is significant. Interoperability is a factor because most intelligence analysis systems must share their "processed" intelligence with a variety of users. Efficiency is an important consideration, but not at the expense of reliability and survivability. Similarly, maintainability and flexibility must be considered, but generally do not override the performance quality factors.

3.9.2 Class II Rules

Rule IW.1.1: IF the function prepares Sensor-Data-Time-History-Display, THEN I&W-Intelligence-Assessment.

Query: Does the function prepare time/history displays of sensor-derived data?
Discussion: This is an I&W/intelligence assessment-related process.

Rule IW.2.1: IF the function provides Trend-Analysis-Tools, THEN I&W-Intelligence-Assessment.

Query: Does the function provide trend analysis tools (e.g. regression, correlation, ANOVA, histograms)?
Discussion: This is an I&W/intelligence assessment-related process.

Rule IW.3.1: IF the function retrieves data on Coverage OR Sensor-Characteristics OR Exploitation-Requirements THEN I&W-Intelligence-Assessment.

Query: Does the function retrieve data on coverage, sensor characteristics, exploitation requirements?
Discussion: This is an I&W/intelligence assessment-related process.

Rule IW.4.1: IF the function provides analysts files on Intelligence Data AND Trend-Analysis, THEN I&W-Intelligence-Assessment.

Query: Does the function provide analysts files on current/historical intelligence data for purposes of trend analysis?
Discussion: This is an I&W/intelligence assessment-related process.

Rule IW.5.1: IF the function provides Computational-Aids, THEN I&W-Intelligence-Assessment.

Query: Does the function provide analysts with computational aids (such as areas of probability, coverage calculations, line of site, closest point of approach)?
Discussion: This is an I&W/intelligence assessment-related process.

Rule IW.6.1: IF the function Defines-Alert-Levels OR Defines-Alert-Conditions from intelligence data, THEN I&W-Intelligence-Assessment AND Automatic-Process.

Query: Does the function define or identify alert levels or conditions from intelligence data without analyst involvement?
Discussion: This is an I&W process. Performing the process without user involvement makes the process subject to the automatic processing rules discussed above.

Rule IW.7.1: IF the function produces Externally-Transmitted-Data AND Alert-Conditions OR Trend-Assessments, THEN I&W-Intelligence-Assessment AND Critical-Function.

Query: Does the function produce externally transmitted data based upon alert conditions or trend assessments?
Discussion: This is an I&W/intelligence assessment-related process. Performing the process without user involvement makes the process subject to the critical function rules discussed above.

Rule IW.8.1: IF the function performs the Correlates-Sensor-Data to tracked objects, THEN I&W-Intelligence-Assessment AND Automatic-Process.

Query: Does the function perform the automatic correlation of sensor data to tracked objects?
Discussion: This is an I&W/intelligence assessment-related process. Performing the process without user involvement makes the process subject to the automatic processing rules discussed above.

Rule IW.9.1: IF the function Defines-New-Tracked-Object, THEN I&W-Intelligence-Assessment AND Automatic-Process.

Query: Does the function initiate the definition of a new tracked object without operator intervention/participation?
Discussion: This is an I&W/intelligence assessment-related process. Performing the process without user involvement makes the process subject to the automatic processing rules discussion.

3.10 Collection Management (CM)

As with the I&W/intelligence analysis functions, the collection management mission within intelligence systems is very important. The quality factor impacts of collection management requirements within an intelligence system are initially codified in the rules that follow.

3.10.1 Class I Rules

No explicit rules of this class are stated here. Instead, Rule CM.1.1 states a relationship between a collection management-related function and critical functions. These latter functions are related to the quality factors using Class I rules presented above.

3.10.2 Class II Rules

Rule CM.1.1: IF the function performs Collection-Management, THEN Critical-Function

Rule CM.1.2: IF the function performs Collection-Management, THEN Time-Constraint.

Query: None.
Discussion: Within an intelligence system, the collection management function is very important to mission success. This implies the rules apply that consider critical intelligence functions. Further, collection management often deals with quick response activities, implying the need to operate within time constraints. Thus the rules that consider time constrained operations apply.

Rule CM.2.1: IF the function performs the input, tabulation, retrieval, and/or display of sensor availability and status, THEN Collection-Management.

Query: Does the function perform the input, tabulation, retrieval, and/or display of sensor availability and status data?
Discussion: This is a collection management-related function.

Rule CM.3.1: IF the function provides Sensor-Location-Data Or Coverage-Data OR Footprint-Data, THEN Collection-Management.

Query: Does the function provide sensor location, coverage, or "foot print" data?
Discussion: This is a collection management-related function.

Rule CM.4.1: IF the function supports Aggregation-Of-Tasking-Requirements AND Resolution-Of-Tasking-Requirements-Conflicts, THEN Collection-Management.

Query: Does the function support aggregation and deconfliction of tasking requests/requirements?

Discussion: This is a collection management-related function.

Rule CM.5.1 IF the function performs the Processing-Collection-Requests AND Validation-Collection-Requests AND Input-Collection-Requests THEN Collection-Management.

Query: Does the function perform the input, processing, and validation of user collection requests?
Discussion: This is a collection management-related function.

Rule CM.6.1: IF the function alerts the operator/user that a tasking needs to be reviewed, THEN Collection-Management.

Query: Does the function alert the operator/user that a tasking needs to be reviewed?
Discussion: This is a collection management-related function.

3.11 General Considerations

This section will address the functional attributes of the intelligence system to determine queries and/or rules that are relevant to developing quality goals but might not be unique to a mission area or specific type of system. In some cases, rules are provided. Starting with section 3.11.3, Authentication, only queries are listed from which rules may be derived. For each query , we have estimated which quality factors are impacted. We have also attempted to place a magnitude on this impact by using pluses. The scale runs up to plus-three (See Figure 3.2). This method permits an impact weighting on a "best judgement" basis.

+	Slightly Increase
++	Moderately Increase
+++	Strongly Increase

FIG 3.1 Quality Factor Increase/Decrease Scale

3.11.1 Error Recovery

The proper restart/recovery of an automated intelligence system is another critical function. Frequently, the system must achieve some level of processing security as part of recovery, before activity can continue, therefore security issues play an important role in establishing quality requirements. In addition, restart/recovery mechanisms are increasingly accomplished using software algorithms, which places greater quality demands on those components that implement them. The rules that follow codify some important considerations related to error recovery. It should be noted, that these rules are somewhat general in nature, and could be applied to other than intelligence-oriented systems.

3.11.1.1 Class I Rules

No explicit rules of this class are stated here. Instead, Rule ER.1.1 states a relationship between restart/recovery, security processing, and automatic processing. These latter functions are related to the quality factors using Class I rules presented above.

3.11.1.2 Class II Rules

Rule ER.1.1: IF the function performs Error-Detection AND Corrective-Action, THEN Automatic-Processing and Function-Enforces-Security-Policy.

Query: None.
Discussion: The restart recovery mechanisms implemented in software must ensure that the system restarts in a secure state as directed by the systems security policy. The fact that many recovery activities are performed automatically upon detection of a fault implies that the factors associated with automatic processing are significant.

Rule ER.2.1: IF the function performs Component-Status-Checking THEN Error-Detection.

Query: Does the function periodically check the status of components?
Discussion: Self-evident.

Rule ER.3.1: IF the function executes Diagnostic-Routines, THEN Periodic-Component-Status-Checking.

Query: Does the function execute diagnostic routines?

Discussion: Self-evident.

Rule ER.4.1: IF Faulty-Condition-User-Alerts, THEN Component-Status-Checking.

Query Does the function alert the operator of a faulty condition, such as a communications line failure?

Discussion: Self-evident.

Rule ER.5.1: IF the function Detect-Failures OR potential Overflow-Conditions, THEN Component-Status-Checking.

Query: Does the function monitor system resources to detect failures or potential overflow conditions?

Discussion: Self-evident.

Rule ER.6.1: IF the function implements a Timeout-Condition AND Failure-To-Receive-Data, THEN Error-Detection AND Corrective-Action.

Query: Does the function implement a timeout condition(s) on failure to receive expected data?

Discussion: Self-evident.

Rule ER.7.1: IF the function implements Error-Handling AND Faulty-Condition OR Invalid-Data based upon a status indicator, THEN Error-Detection AND Corrective-Action.

Query: Does the function implement error handling for a faulty condition or for invalid data based upon a status indicator?

Discussion: Self-evident.

Note: When this rule is invoked, an additional question should be asked to determine if automatic error correction occurs.

3.11.2 Crisis/Time Constrained Operations (CT)

Many systems, those for intelligence being included, must perform their mission during different, and perhaps hostile, conditions. The rules that follow provide an initial look into this complex area, which should be examined in considerably more detail at some future time.

3.11.2.1 Class I Rules

Rule CT.1: IF the function must complete discrete operations within a Time-Constrained-Operations THEN it is strongly indicated Efficiency-Is-Needed, Verifiability-Is-Needed and Expandibility-Is-Needed.

Query: None.

Discussion: Functions that are time-constrained normally place a significant design burden on the designer to ensure that the time budgets are met within the profiled processing workload. Software efficiency is essential for that reason. Verifiability is very important because the implementation must be subjected to test against the specified timing conditions. Time-constrained operations can take on varying degress of importance depending on whether the system is operating in a routine or crisis/wartime mode. For this analysis, a tactical intelligence system is assumed to have timing requirements consistent with crisis/wartime operation that pose high risks if not met. In addition, significant timing problems are often not evident until after the system is delivered, therefore expandability is needed to meet current and future requirements.

3.11.2.2 Class II Rules

Rule CT.1.1: IF the function must support Crisis-Processing (e.g., elevated levels), THEN Time-Constrained-Operations

Query: Does the function support routine, crisis, and/or wartime missions in terms of information processing workload?

Discussion: Routine, crisis, and wartime requirements for intelligence systems are generally stated in terms of data processed within some time constraint.

Rule CT.2.1: IF the function must process incoming data at an increasing volume profile over time, THEN Time-Constrained-Operations.

Query: Does the function have a requirement to process data of increasing volume over time?

Discussion: Processing workload is normally associated with a time constraint, e.g., number of messages to be processed or distributed per second. For this rule, "over time" refers to future time, e.g., the requirement to handle an increase in message volume during the next year.

3.11.3 Authentication

AU.1 Is there a requirement to determine what individual, process, or device is at the other end of a communications network?

Response 1: YES

Discussion: Local Area Networks (LANs) and Wide Area Networks (WANs) are used in intelligence systems. Both networks are vulnerable to several security threats that would influence the software integrity requirements. An initial assessment should be made of what level of protection is needed regardless of whether or not the information is classified, sensitive, or national security related. This query attempts to determine whether network security is a potential consideration based on a need for identification.

Action: Ask query AU 1.1

Impact: Integrity +

Response 2: NO

Discussion: If identification is not needed, a basic element of software-enforced network security is missing so network software integrity requirements are minimal. In these cases, security measures such as physical security, administrative security, communications security, etc are likely used to provide network security. The network probably is housed within a single facility or all systems communicating on the network are evaluated at the same level of security or trust. Since integrity requirements are low but the mission is still intelligence, verifiability and reliability should be increased to ensure the validity of the output.

Action: Ask query CF.1.

Impact: Verifiability +
Reliability +

AU.1.1 Do you have to identify and authenticate the location of the hardware or operating system at the distant end-point or in any intermediate system involved in the network communication?

Response 1: YES

Discussion: Combined identification and authentication requirements imply some access control and therefore an increase in integrity. Identifying and authenticating the hardware location or operating system requires functionality at the data link layer or the network layer of the Open Systems Interconnection (OSI) model. Data is being exchanged between nodes and there is a need to ensure correct routing, detection of data errors, and reliable data transfer. No interactive capability is implied. Its likely this is a long haul packet switching network like DDN. A major security concern is with masquerading, where an unauthorized node or an authorized node that is untrusted or comprised acts like an authorized node to get access to data. Authentication is a protective measure against this threat. As integrity is increased so is correctness, reliability, and verifiability. These factors help determine a level of assurance in the software responsible for enforcing the security policy (i.e., trusted software). Correctness because trusted software must closely conform to specification; reliability because it must enforce the security policy without failure; and verifiability because trusted software operation must be well verified. Connecting systems with each other and the network requires increased interoperability.

Action: Ask query AU.1.2

Impact: Integrity ++
Interoperability ++
Reliability +
Correctness +
Verifiability +

Response 2: NO

Discussion: There is a requirement for identification but no authentication. The source or destination of the data is known but there is no mechanism to verify it. Some integrity is implied but the assurances are not a priority. Interoperability is slightly increased to accommodate the data exchange.

Action: Ask query CF.1

Impact: Integrity +
Interoperability +

AU.1.2 Do you have a requirement to identify and authenticate the application or user at the distant end-point involved in the network communication?

Response 1: YES

Discussion: There is a functionality requirement for authentication at the application layer. At this level of capability, full interactive processing is allowed as well as data exchange. Masquerading is still a concern. Integrity is strongly increased because of the higher level of system interaction and greater risk for unauthorized access. Full interactive capability increases interoperability considerations. A more secure design places an emphasis on assurances therefore correctness, verifiability, and reliability are increased along with the integrity.

Action: Ask query CF.1 under Communication Field Integrity

Impact: Integrity ++
Interoperability ++
Correctness ++
Verifiability ++
Reliability ++

Response 2: NO

Discussion: Users are only exchanging data over the network. The discussion under AU 1.1, "No Answer", applies.

Action: Ask query CF.1

Impact: None

3.11.4 Communications Field Integrity

CF.1 Do you have a requirement to protect communication against unauthorized modification such as message stream modification through insertion, deletion, or replay?

Response 1: YES

Discussion: Modification of data can be used to compromise communication between trusted nodes, even when some level of authentication is used. Protection measures are likely to include some type of encryption mechanism. Encryption places added quality and reliability requirements on software. For example, protocol control information must be sent around the encryption unit through a bypass. The software to implement the bypass must be trusted not to send user data through the bypass, this implies increased integrity. Encryption, depending on the algorithm, could adversely impact throughput therefore a greater need for efficiency. Integrity leads to greater assurance requirements. Assurance refers to a basis for believing that the functionality will be achieved. It involves verifiability and proof of correctness through testing.

Impact: Integrity ++
Reliability ++
Correctness ++
Verifiability ++
Efficiency +

Response 2: NO
Discussion: This situation is similar to Response 2 under query AU.1. Modification of data is likely a threat, but the software is not the enforcer of the security policy.
Action: Ask query CF.1
Impact: None

3.11.5 Fault Tolerance (FT)

FT.1 Is there a standardized method for performing error detection, handling, and recovery for each software function?

Response 1: YES
Discussion: When we say that there are standard error detection, handling, and recovery procedures, we can be sure that each function has a method to increase its reliability and survivability. Procedures (or features) to enhance reliability and survivability need to possess these features themselves; therefore reliability and survivability should be increased.
Action: Ask Query FT.2
Impact: Reliability ++
Survivability ++

Response 2: NO.
Discussion: A NO response can mean one of two things; (1) there is no error detection, or, (2) there are no standard procedures. If the former, it means that survivability has been discarded. Therefore, the reliability of all functions must be extremely high to ensure that there are no errors. This instance of no error detection is very rare. The latter implies that each function performs its own fault tolerance. Again, this would imply that reliability and survivability requirements for these functions should be moderate. However, without standard procedures, testing and error correction will be much more difficult. As such, the software needs to have a slight increase in maintainability to offset the absence of standard procedures.
Action: Ask query FT.2

Impact: (1) Reliability +++
(1) Survivability -
(2) Survivability ++
(2) Reliability ++
(2) Maintainability +

FT.2 Will critical hardware component redundancy be employed such as master-checker processor pairs?

Response 1: YES
Discussion: A YES response implies some type of fault tolerant design. The combination of additional processors as well as the supporting operating system helps increase the system survivability. This places less emphasis on application software reliability, survivability, maintainability, and verifiability.
Action: Ask query FT.2.1.
Impact: Reliability -
Survivability -
Maintainability -
Verifiability -

Response 2: NO
Discussion: The implication is that there is a single CPU responsible for providing its own fault tolerant features and that these features are limited. The application software must assume increased error detection and correction responsibility to function properly. Reliability, survivability, maintainability, and verifiability should all be increased.
Action: Ask query FT.3
Impact: Reliability +
Maintainability +
Verifiability +
Survivability +

FT.2.1 Does the system support dynamic changing of CPUs used in master-checker pairs as part of a fault tolerant design?

Response 1: YES

Discussion: Dynamic switching of CPUs when faults are detected implies operating software capable of automatically transition processing to a different CPU without interruption or loss of data. Some processors will notify the user of a detected error but require a processor shutdown and a system reconfiguration before processing is again started. In the first case, the operating software performs the fault tolerant functions, placing less of a burden on the application software.

Action: Ask query FT.3

Impact: None

Response 2: NO

Discussion: The operating system software does not possess strong fault tolerant features therefore fault recovery must be performed by the application software. An increase in verifiability and maintainability are appropriate to ensure the proper operation of the reconfigured system or resolve software problems that occur as a result of the reconfiguration.

Action: Ask query FT.3

Impact: Maintainability +
Verifiability +

FT.3 Will critical software routine redundancy be employed?

Response 1: YES

Discussion: Software redundancy is the use of multiple routines to perform identical functions. A calculation is a good example. The sine of a number can be calculated by using a routine that uses the definition of sine (with factorials and all) and another that uses a library function within a given language and, perhaps, a third which uses the cosine or tangent of the appropriate complementary angle. The results are compared and the majority answer is taken. If there is no majority, the software uses the value from the module that has been deemed most accurate or reliable. As this "polling" process increases the reliability of the calculation, the reliability of each individual module does not necessarily have to be increased. As such, there is no need to increase or decrease functional reliability if software redundancy is used.

Action: Ask query FT.4

Impact: None.

Response 2: NO

Discussion: The absence of software redundancy implies that each software module must be more reliable as it is the only module that performs that specific function. As such, reliability should be moderately increased. Further, the absence of backup implies that survivability and maintainability will also be important.

Action: Ask query FT.4

Impact: Reliability ++
Survivability ++
Maintainability ++

FT.4 Will the software provide techniques for recovering from all hardware failures?

Response 1: YES.

Discussion: (1) If FT.2 = YES, the requirements need to be stepped up to a moderate level for all factors. (2) If FT.2 = NO, Survivability levels should be highly increased. The reasoning here is that the software is the only method of countering hardware failures or errors.

Action: Ask query HOL.1.

Impact: (1) Reliability ++
(2) Survivability +++
(1,2) Survivability ++
(1,2) Maintainability ++

Response 2 NO.

Discussion: (1) If FT.2 = YES, no change from the FT.2 requirements. There is hardware redundancy and the software only has to counter some hardware faults (2) If FT.2 = NO, there is also no change from the FT.2 requirements. There is no hardware redundancy and the software only has to counter some hardware faults. Obviously, certain hardware faults may prove fatal as there are no recovery procedures. This is probably a cost/benefit decision.

Action: Ask query HOL.1.

Impact: None.

3.11.6 High-Order Language (HOL)

HOL.1 Will the software be exclusively developed in a high-order computer language such as PASCAL, FORTRAN, or Ada?

Response 1: YES.

Discussion: The use of a high-order language (HOL) does not impact the requirements of the software. However, it is one of the most important contributors to adaptability requirements. Use of a high-order language enhances program modularity, generality, self-descriptiveness, simplicity, functionality, and independence. These attributes add to the maintainability, expandability, reusability, reliability, and survivability of the system. Employing a high-order language in the development of system software might not require the software to possess these quality factors, but it will add to the overall system quality.

Action: Ask Query HOL.2.

Impact: None.

Response 2: NO.

Discussion: The use of a language other than an HOL (assembly, machine) does not aid in the attainment of the software quality factor criterion of modularity, generality, self-descriptiveness, simplicity, or independence. As such, there is a need for the software to compensate for these missing qualities that an HOL would provide. Therefore, the requirements for maintainability, survivability, reusability, reliability, and expandability all need to be increased to a moderate degree. This will ensure that the software will possess the same levels of these factors as code written in an HOL. In addition, the most dangerous language capability that could be offered to programmers is assembly language, with its unlimited capacity for controlling and subverting a system. If assembly language is used, then it is assumed that the system software supports compilers that accept assembly. This means that correctness, verifiability, and reliability should all be increased to counter the decreased integrity of the system software.

Action: Ask Query HOL.2

Impact: Survivability ++
Reusability ++
Maintainability ++
Reliability ++
Expandability ++
Correctness ++
Verifiability + +

HOL.2 Will intelligence system software be developed using identical languages and subsets thereof?

Response 1: YES.

Discussion: This is the preferred scenario as far as minimizing the impact on the software quality factors. There is no significant impact.

Action: Ask Query HOL.3

Impact: None.

Response 2: NO.

Discussion: The use of different languages and/or subsets thereof take away the inherent maintainability, expandability, and reusability characteristics that would be present were one subset or language used. As such, these factors should be slightly increased to ensure necessary software quality.

Action: Ask query HOL.3

Impact: Expandability +
Maintainability +
Reusability +

HOL.3 Does the language contain inherent fault tolerance detection and recovery capabilities such as Ada's exception handling?

Response 1: YES.

Discussion: This will make the implementation of software Reliability and Survivability easier; but, it does not impact the need for these factors.

Action: Ask query IC.1

Impact: None.

Response 2: NO.

Discussion: The absence of inherent fault tolerance features within the programming language puts an added requirement on the software -- that of providing those fault tolerance procedures the language does not possess. Working in C would be a nightmare here because there is no type checking whatsoever, as one example. Again, we get into the bind where the need to manually provide a feature doesn't necessarily mean the quality factor the feature will enhance is required by the feature. In this case, it is suggested that Reliability and Survivability requirements be slightly increased.

Action: Ask query IC.1

Impact: Reliability +
Survivability +

3.11.7 Interface Complexity (IC)

IC.1 Have all software function interfaces been standardized?

Response 1: YES.

Discussion Once again, an attractive attribute of the software but it doesn't impact the software quality factors. It does add to the correctness and maintainability of the interface software but does not imply a variation in the requirements for software quality.

Action: None.

Impact: None.

Response 2: NO.

Discussion: This implies that the interface between each software function is unique. This would impact correctness and maintainability in that the assurance that the interfaces conform to standards and specifications is doubtful, at best, and the uniqueness of the interfaces will prevent the normal learning curve to apply to the detection and correction of interface errors as each interface is different. As such, correctness and maintainability requirements need to be increased slightly.

Action: None.

Impact: Correctness +
Maintainability +

3.11.8 Testing (TE)

TE.1 Have all processing functions been assigned processing times to determine the probability of satisfactory functional performance?

Response 1: YES.

Discussion: This is yet another method for preventing error or fault manifestation. Each processing function has a time range assigned to it. If a result has been produced within this space of time, no error checking is performed. Should the processing time be less than or greater than this range, there is a reasonable doubt as to the accuracy, precision, and reliability of the output. The accuracy and reliability of this timing feature, however, needs to be assured so as to only call upon these output checking routines when necessary. Constant invocation of a checking routine could add considerable time to the overall processing time which could prove crucial depending upon the timeliness requirements of the data being checked. Further, the survivability and maintainability of this time checking routine is required to ensure the availability and operation of the routine. For all three factors, a moderate increase is required.

Action: Ask query TE.2.

Impact: Survivability ++

Reliability ++

Maintainability ++

Response 2: NO.

Discussion: Given a checking routine such as in query TE.3, and the absence of this time checker, the efficiency of the software most likely needs to be increased moderately. Why? Those other routines take time and they may seriously impact processing time. Increasing the efficiency of the code to permit faster average processing times will enable those other checking routines to offer a true benefit.

Action:	Ask query TE.2.
Impact:	Efficiency ++

TE.2 Are error tolerances specified for all input and output data?

Response 1: YES.

Discussion: This checking feature of the software will help to ensure reliability of results and to prevent possible software errors due to manipulations on data that are of the wrong type, out of range, etc. As a result of the addition of this reliability feature, reliability should be slightly increased. This feature will also enhance the survivability of the software, as input that may cause faults and errors will be detected before processing is commenced. Thus, it is in our best interest to increase the survivability of this software slightly to ensure that it operates in the presence of faults.

Action:	Ask query TE.3.
Impact:	Survivability +
	Reliability +

Response 2: NO.

Discussion: The absence of input testing indicates that the software must be able to gracefully handle erroneous input. This implies an increase in survivability to a moderate level. The reliability of the software is not necessarily affected as, given the erroneous input, how could we expect the function to give a reasonable answer? Survivability would ensure the software wouldn't cough too much given the input. Maintainability is also needed to locate the source of input error and to perform timely corrections is required.

Action:	Ask Query TE.3.
Impact:	Survivability ++
	Maintainability +

TE.3 Will all computational outputs be tested to ensure they fall within expected ranges?

This query is very much like TE.2. There is one subtle difference, checking the inputs ensures the data is appropriate for the function it is about to enter. Checking output ensures the data is appropriate for the function it just left. One is necessary to ensure software failures will not occur because of bad input, the other ensures that the function has worked properly. Both features, however, can be designed and implemented in identical manners. As such, the requirements are identical.

Response 1: YES.
Discussion: See TE.2, Response 1.
Action: Ask query NA.1.
Impact: Survivability +
Reliability +

Response 2: NO.
Discussion: See TE.2, Response 2.
Action: Ask query NA.1.
Impact: Survivability ++
Maintainability +

3.11.9 Architecture (AR)

AR.1 Will the system be geographically dispersed?

Response 1: YES
Discussion: If a system is geographically dispersed then there is an implication of a distributed architecture that would require, at a minimum, some type of a communication interface. Since the system is likely to handle sensitive information, communication security would be an issue.
Action: Ask query AR.2
Impact: No quality factor impact. Determines need for communications.

Response 2: NO
Discussion: The system is located in one facility and it has limited or no communication requirements.
Action: Ask query AR.2

Impact: None

AR.1.1 Is the design an open architecture solution using only industry communication protocol standards?

Response 1: YES
Discussion: The use of industry standards such as IEEE or ISO improves interoperability and expandability of a system.
Action: None
Impact: Interoperability ++
Expandability ++

Response 2: NO
Discussion: If there is no adherence to communication standards then the added flexibility is needed to accommodate integration of the proprietary protocols with industry standard protocols.
Action: None
Impact: Flexibility ++

IV. Satellite System Software Analysis

4.0 Scope

This analysis is only applicable to the satellite software type within the Missile/Space mission area. The notional satellite system components are taken from the Missile/Space mission area decomposition (See Appendix A) and consists of Propulsion, Power, Data Communications, Command and Control, and Attitude Control. The specific mission packages of the satellite (i.e., Surveillance, Communications, Navigation, etc.) are not addressed.

4.1 Satellite System Software Definition

Satellite system software will include all those software functions accomplished either on-board the satellite or at a ground-based receiving station. Applicable ground-based operations will include those software functions that receive data from the satellite, generate commands to the satellite, or interpret the received data. Ground-based software that performs functions such as satellite tracking or software on-board a launch vehicle which places a satellite in its final orbit, will not be considered in this evaluation.

4.2 Software Quality Factor Applicability

Some of the thirteen software quality factors may not apply to satellite system software. If factors are found to be non-applicable, this information will be used to write mission-level rules that will eliminate factors from further consideration based on the selection of the mission area and software type.

For satellite system software, Adaptation the quality factors expandability, flexibility, portability, and reusability, take a slightly higher precedence. This is mainly due to the method by which satellite systems are developed. Generally speaking, satellite systems are developed in phased deployments, or blocks. Due to the complexity of technical requirements and the ongoing advancement of technologies applicable to satellite systems, deployed satellites do not usually satisfy all existing requirements. As such, follow-on systems, or block upgrades, are routinely planned to accommodate unfulfilled and new requirements using technology that has matured since the initial system was fielded. In this respect, adaptability of satellite software is key to the development, deployment, and operation of follow-on systems. As it is important in these followons, the need for initial designs to consider adaptability as a software requirement will reap great dividends as follow-on systems are planned and developed.

If we expressed this precedence in terms of a system-level rule, it would read:

IF satellite-software-type THEN expandability, flexibility, portability and reusability take precedence.

Interoperability-The software quality factor of interoperability can be eliminated from this analysis as not applicable to satellite software. Interoperability deals with the coupling of software between systems. Each satellite system is usually autonomous -- that is, it processes mission data and reports it to some source. This source, like NORAD for example, receives data from a variety of sources -- some satellites, some not satellites. Correlation of mission data is done in an environment that is removed from these sources and the sources do not communicate with each other. Very few satellite systems even permit communication between space platforms within the same system or alone other systems. When software is coupled, it implies that the action of one system's software has a direct impact on the action of another system and its software. An example where satellite systems would be coupled is presented here. A radar satellite detects a bomber raid with some certainty. It passes this information directly to an intelligence and reconnaissance (IR) satellite for verification. The IR satellite acts upon this data and commands its sensors to look in the proper direction. This is not the way this operation takes place. A central control center, like NORAD, would receive the radar detection and would then direct the IR satellite to verify. Interoperability is not deemed applicable to satellite software since there would be no direct communication between these two satellite systems. As such, interoperability is not deemed applicable to satellite software.

Usability-Usability has not been deemed applicable to satellite software and can be eliminated from this analysis as not applicable to satellite software. The vast majority of the system software is transparent to the user -- either located on-board the satellite or embedded in routines that the user cannot directly access. The user interface portion of the software represents a very small fraction of the total software.

Remaining Quality Factors-The remaining seven software quality factors are all deemed applicable to satellite software. Of these, extra emphasis can be placed on some of the factors. As previously stated, adaptability factors are. in general, important to satellite software. Additionally, because so much of the software is removed from programmer/analyst access, reliability and survivability need to be emphasized. The ability for the software to perform as anticipated, even in the presence of software failures and errors, is paramount to the success of the satellite's mission. Maintainability should be a prime concern in the development of ground-based software as this software is accessible and is critical to proper command generation. Efficiency, integrity, and correctness are important, but less then Adaptation and other Performance factors. Integrity ensures the unauthorized direction or use of a satellite. Correctness is important because the ability to recover from or change defective software is very limited once the satellite is launched. With an emphasis on correctness, the ability to verifiy and validate software is critical, so verifiability is applicable. Efficiency is closely tied to the satellite's mission. If it is a spaced-based radar, its algorithms for sampling incoming signals must be very efficient to maximize detection capability. Communication satellites have similar considerations. The general assumption for satellites is that efficiency is applicable.

4.3 Satellite System Notional Architecture

Figure 4.1 shows the satellite system notional architecture. Each component is described below with the exception of General Considerations. This component contains those attributes which are system, not component specific.

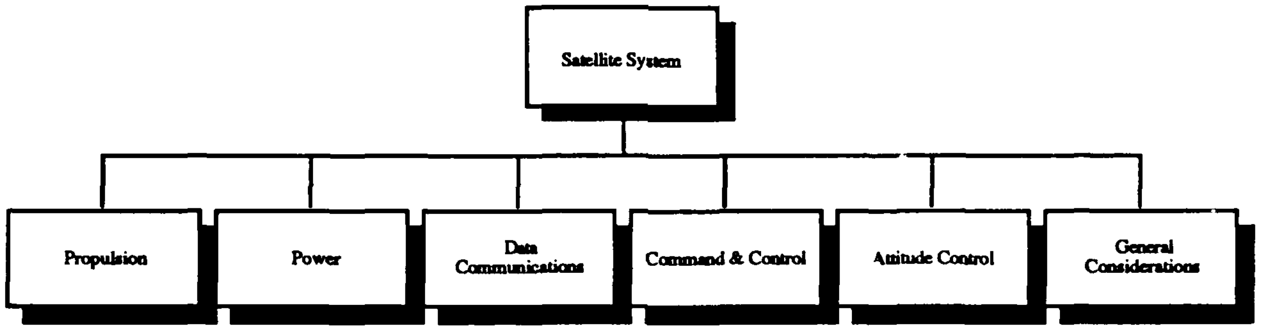

Figure 4.1. Satellite System Notional Architecture

Propulsion. This component provides the required thrust to perform platform attitude adjustments -- whether it be to stabilize the platform in its current orbit or to transfer the platform into a completely new orbit. This would include the firing of the on-board engines, adjustment of the burn to minimize the acceleration of the maneuver on the satellite hardware, the attainment of the proper delta V (change in velocity) to accomplish the attitude change, and the termination of engine activity upon maneuver completion.

Power. This component provides the electrical power necessary for the satellite to perform its mission as well as the means for the other generic functions to perform their required tasks. This unit consists of some sort of power source, usually a combination of solar antenna and batteries (although nuclear power/propulsion is on the way), as well as a means to distribute and monitor the electrical output to the various subsystems. Additionally, the Power unit must successfully monitor and dissipate excess heat created by the electrical generation process and microprocessor operation.

Data Communication. This component provides the communication capability between platform and either a ground, air, or sea terminal site on the earth. This includes the receiving/sending of commands from/to the earth terminal (and vice versa), the sending of telemetry and mission data and all functions required to establish the communication link, encode the commands, alter frequencies, etc. The hardware and software to perform these functions is co-located, both on the space platform and at various ground terminals.

Command and Control. The control center for the entire platform. Data from all sources is accepted, actions to be taken are determined, and commands issued to subsystems to accomplish these actions. This includes the receipt of ground commands and their execution, the monitoring of platform health and status, performance of the satellite's primary mission, data storage and retrieval, command decoding, and all other operational functions not specifically performed by the other four generic functions. The software to perform these activities is primarily located on-board the space platform.

Attitude Control. This component maintains the proper stability, orientation, navigation and guidance of the platform. It has the ability to sense the platform position relative to celestial navigation guides and to inform the command and control unit of any discrepancies to the norm. When the command and control subsystem deems an orbit correction is necessary, the attitude control unit controls the propulsion subsystem thruster burn and attitude correction.

General Considerations. This component contains those attributes which are system, not component specific. General considerations would contain functional requirements that can not be allocated to other top level functions. It may also contain functions that could be allocated to multiple top level functions. This could include functions such as "packaging" which would affect the software vulnerability to nuclear, chemical or biological effects.

4.4 Applicability of Software Factors by Component

This section will look at each satellite component and its functions to determine which of the software quality factors are applicable.

4.4.1 Command & Control and Data Communications

The importance of the functions performed by these components, deems that all software quality factors are applicable. Because it is a space platform most of the hardware and software functions necessitate reliability and survivability. The possibility of unauthorized command receipt and action necessitates the applicability of integrity. The limitations of on-board resources causes efficiency to be an applicable factor. Correctness is self-evident for all functions in that any software developed for the Department of Defense (DoD) must conform to a rigid set of specifications and standards. Maintainability is applicable to those portions of the software that are easily accessible to technicians and programmers. Reusability and expandability are applicable mainly because of our previous discussion on satellite software development and adaptability. Verifiability is important because of the autonomy of satellite systems. Portability and flexibility are important from a program development and Life-Cycle-Cost perspective.

4.4.2 Attitude Control, Propulsion, and Power

These three components all have two major characteristics in common -- one, they are all physically located on-board the space platform and, two, they have no direct interaction with ground-based functions. These components all receive and send commands to each other or the on-board portion of the command and control component. As such, the need for integrity is not foreseeable. The software is embedded within the subsystems and even the command and control component doesn't physically access them, but merely sends commands to them. As integrity is an integral part of the data communication and command and control functions, we can eliminate it here.

Expandability, for much the same reason, is also deemed unnecessary. While we do desire the software to be reusable, its isolation from human access makes it a poor candidate to be expanded within the framework of its existing application. Whatever capability that is designed into the software that is put on-board the satellite is all that will be available, no matter how easily the software could be expanded. No further consideration will be given to the expandability of this software.

4.4.3 General Considerations.

General considerations is not an actual architectual component, but rather a group of system attributes not attributed to a specific component. Such attributes include factors such as time critical operations and error recovery.

4.5 Factor Applicability Conclusions

The discussions in the previous four sections have helped us to narrow the problem to those factors and functions that have been deemed applicable. Figure 4.2 is an applicability matrix of software quality factors to software functions. Those shaded areas have been deemed non-applicable.

S/W QUALITY FACTOR / SATELLITE COMPONENT	EFFICIENCY	INTEGRITY	RELIABILITY	SURVIVABILITY	CORRECTNESS	MAINTAINABILITY	VERIFIABILITY	EXPANDABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY
COMMAND & CONTROL											
ATTITUDE CONTROL		▨						▨			
PROPULSION		▨						▨			
POWER		▨						▨			
DATA COMMUNICATIONS											

Figure 4.2. Quality Factor Applicability Matrix

To support the applicability conclusions, we could establish the following rules. The rules are placed in two groups, system level or function level, depending on the scope of impact.

<u>System Level</u>

Rule SAT.1: IF Satellite-Software-Type THEN NOT System-Interoperability

Rule SAT.2: IF Satellite-Software-type THEN NOT System-Usability

Rule SAT.3: IF Satellite-Software-type THEN NOT System-Flexibility

Rule SAT.4: IF Satellite-Software-Type THEN NOT System-Verifiability

Rule SAT.5: IF Satellite-Software-Type THEN NOT System-Portability

Rule SAT.6: IF Satellite-Software-Type AND Generic-Project THEN a Command-And-Control component.

Rule SAT.7: IF Satellite-Software-Type AND Generic-Project THEN an Attitude-Control component.

Rule SAT 8: IF Satellite-Software-Type AND Generic-Project THEN a Propulsion component.

Rule SAT.9: IF Satellite-Software-Type AND Generic-Project THEN a Power component.

Rule SAT.10: IF Satellite-Software-Type AND Generic-Project THEN a Data-Communications component.

Function Level

Rule PRO.1: IF the Propulsion component AND Generic-Project THEN NOT Function-Integrity

Rule PRO.2: IF the Propulsion component AND Generic-Project THEN NOT Function-Expandability

Rule PWR.1: IF the Power component AND Generic-Project THEN NOT Function-Integrity

Rule PWR.2: IF the Power component AND Generic-Project THEN NOT Function-Expandability

Rule ATC.1: IF the Attitude-Control component AND Generic-Project THEN NOT Function-Integrity

Rule ATC.2: IF the Attitude-Control component AND Generic-Project THEN NOT Function-Expandability

4.6 Query Development

The next stage in the analysis is to develop queries that probe into the details of the system functions and operations. A four step process was used in this development. The first two steps determined important attributes of the satellite components and software quality factors. These two lists are contained in TABLES 4.1 and 4.2, respectively. TABLE 4.1 gives a list of activities that the components must accomplish or concern themselves with. TABLE 4.2 lists characteristics of the quality factors. The process to develop rules and queries can then draw upon each software function and attempt to apply these operations to the individual software quality factors. Generally speaking, these lists aided the analyst in quantifying what had to be investigated.

TABLE 4.1. Satellite Component Attributes

Propulsion:	Amount of fuel Type of fuel Burn time
Power:	Maximum power requirements Type of batteries Type of power source Size and weight of power source Minimization of energy emittance
Data Communications:	Frequency of transmissions receiver/transmitter Size and flexibility of communication antenna
Command & Control:	On-board memory capacity On-board processing capacity Encryption/decryption techniques Error detection and recovery

TABLE 4.1. Satellite Component Attributes (Cont)

Attitude Control:	Orbit altitude Degree of precision required Need to change orbital planes Flexibility of communication and mission hardware
General Considerations:	High-order language usage

TABLE 4.2. Software Quality Factor Characteristics

Efficiency:	Compiler efficiency (optimization) Memory utilization/management Processing efficiency Assembly vs High-order language use
Integrity:	Authorized access to software Encryption of data Authentication messages
Reliability:	Precision of calculations (engineering accuracy) Fault tolerance S/W failures
Survivability:	Fault tolerance (error recovery, reconfigurability) Interface complexity Separate power sources Redundancy (H/W and S/W) Internal testing
Correctness:	Inputs, processing, & outputs clearly defined Standard I/O protocol between software units Standardized data representations
Maintainability:	Error detection/recovery Software modifications Integration complexity
Expandability:	Spare memory storage capacity Spare I/O channel capacity Spare processing capability Spare communication channels Interface compatibility
Reusability:	Standard subset of implementation language Modularity of design

The third step in this process was one of organization and correlation. The attributes as shown in TABLE 4.3, are organized into query topics which cover both functional aspects of the system as well as important characteristics of the software. For each topic, the affected quality factors and components are listed.

TABLE 4.3 Satellite Software Query Topics

Query Topic	Quality Factor	Component
Communication Frequencies	Efficiency Expandability Integrity Maintainability Reliability Reusability Survivability	Data Comm C^2
Data Encryption	Integrity Maintainability Reliability Survivability	C^2, DC
Fault Tolerance	Efficiency Maintainability Reliability Survivability	All
High-order Language	Efficiency Expandability Maintainability Reliability Reusability Survivability	All
Interface Complexity	Correctness Efficiency Expandability Maintainability Reliability Reusability Survivability	All

TABLE 4.3 Satellite Software Query Topics (Cont)

Movable/Adjustable Subsystems	Efficiency Maintainability Reliability Survivability	All
Memory Utilization/Management	Efficiency Expandability Reliability Survivability	All
Processing Capabilities	Correctness Efficiency Maintainability Reliability Survivability	All
Propellents	Reliability Efficiency Survivability Maintainability	Propulsion Attitude Control Power
Power Source	Efficiency Reliability Survivability	All
Testing	Correctness Efficiency Maintainability Reliability Survivability	All

The final step in the process was the development of queries. This analysis was based on expert knowledge of actual Air Force space systems as well as literature available on the Voyager spacecraft and on all ongoing Air Force satellite systems.

Explaining the reasoning and logic in moving from TABLE 4.3 to the and queries is very difficult. As the primary purpose of this analysis is to determine the impact upon the function's software quality factors, these queries were developed with this in mind -- what needs to be asked within this query topic that will tell us something about how much of a particular quality factor needs to be present within the software. The analysis starting in Section 4.7 discusses the responses to the queries and evaluates them in terms of their impact on the software functions and quality factors.

4.7 Satellite Software Analysis

This section will take a look at each satellite function and determine the impact of each possible response on the software quality factors. This analysis will accomplish several things:

1. Determine the order, within a query topic, in which the questions should be asked.
2. Provide a judgement as to the relative requirement for one software quality factor over another for each software function.
3. Address issues pertaining to the evaluation of software quality factor requirements.
4. Give some insight into satellite software functions and software quality factors.

The impact of each quality factor on a function is measured in terms of indicated (+), moderately indicated (++), or strongly indicated (+++), respectively. In many of the queries, the factor impact could be traced to a particular function. In those cases where no function is listed under the impact section, the quality impact is system wide and not function specific.

4.7.1 Software Features

Many of the queries ask if the system contains a certain feature. By feature, we mean a software routine that performs a function that aids in the enhancement of certain software quality factors. For example, a software routine may check each received command for authenticity to enhance the system integrity, or, a polling routine accesses three different modules and then takes the majority answer to enhance the reliability of a calculation, or, a diagnostic routine is implemented that detects errors and aids in the maintenance of the software. The question is this -- does a routine that enhances a particular software quality factor need to possess a greater degree of that quality factor itself? Should a feature that enhances Integrity possess more integrity than one that ensures something else? Does a reliability enhancement need to be more reliable than the function it enhances? Does a fault tolerance enhancement need to be more maintainable than the modules it is checking? These are not always simple questions to answer.

The issue initially arose in the analysis of some of the data encryption queries. While the features in question did add a certain degree of integrity to the system, it was not clear as to whether or not these features, themselves, required an increased degree of integrity. After all, we're trying to determine what the software requirements are, not the requirements for the system. In this case we ruled that, no, integrity was not needed in any increased degree within the software that implemented this feature. But what about reliability? An enhancement that aids in the reliability of the system would, so it would seem, need to be reliable. In fact, even the feature to implement integrity should be reliable. Similarly, maintainability of a maintainability enhancement is not necessarily required.

Ideally, we would wish all software ever written to be efficient, reliable, survivable, maintainable, reusable, correct, expandable, and to possess integrity. Unfortunately, this is never the case, and if it were, the software would be unaffordable. So, we had to make a decision -- what software features should contain the quality factor they seek to enhance at a system level. TABLE 4.4 follows and indicates the choices we have made.

TABLE 4.4 Software Features

NO	YES
Correctness	Reliability
Efficiency	Survivability
Expandability	
Integrity	
Maintainability	
Reusability	

We can discuss each one to present the reasoning behind these selections. Reliability and integrity have already been discussed. Survivability is similar to reliability in that if the survivability enhancement fails, the survivability of the entire system could be jeopardized. Correctness, reusability, efficiency, expandability, and maintainability can all be addressed together. Software that ensures that operational code contains these respective characteristics is usually run during testing and verification. As this testing does not impact the mission, there is no need for them to possess the characteristic they ensure. A claim for reusability could be made, but then again, these routines are not usually part of operational code, such as the case with reliability and survivability features. They can be characterized as off-the-shelf diagnostic programs that read operational code and report on the containment of the various software quality factors.

4.8 Burn Time

4.8.1 Query BT.1

Query: Is there sufficient spare fuel to permit a reasonable amount of additional platform attitude adjustments in the case where the system has been deployed and, for one reason or another, more are required than were planned?

Response 1: YES
Discussion: There is spare fuel. This implies an increase in overall satellite weight. This means that more fuel is necessary to make the early attitude adjustments. We need more fuel to move the satellite as it is now heavier due to the addition of spare fuel -- there obviously is an optimum point for which adding more fuel produces no additional (and perhaps fewer) attitude adjustments. More adjustments imply more firings of the engines. The more times the engines are fired, the higher the probability that they will fail. Since more engine firings are now necessary, this firing mechanism (hardware and software portions) must be more reliable. This firing mechanism is part of the Propulsion function, therefore, Propulsion reliability needs to be increased moderately.

Packing extra fuel on-board indicates the possibility of a longer performance life. A longer satellite life implies the hardware and software components of all functions will be required to operate longer, under the adverse conditions of the space environment, yet still perform their intended functions reliably and in the presence of potential hardware and software failures. This impacts the quality factors of efficiency, reliability, and survivability for all the satellite functions, not just Propulsion. Reliability needs to be slightly increased, while survivability should be moderately increased as there is a great potential, over time, for the components to develop errors and faults.

Action:	None		
Impact:	Propulsion	Power	Data Comm
	Reliability ++	Reliability +	Reliability +
	Survivability ++	Survivability ++	Survivability ++
	Command & Control		Attitude Control
	Reliability +		Reliability +
	Survivability ++		Survivability ++

Response 2: NO

Discussion: There is no spare fuel, there is no chance of providing additional attitude adjustments. The Attitude Control function must now be highly reliable. We only have so many engine firings at our disposal. For the Attitude Control function to tell us to move when, in fact, we may not have to, would have a significant impact on the mission. A slight increase in Propulsion reliability is also required. For each attempt at a burn, some small amount of fuel (depending upon the type, method of ignition, etc.) may be expended. With a limited fuel resource, this ignition system must perform its job with a minimum number of misfires.

Action:	None	
Impact:	Attitude Control	Propulsion
	Reliability +++	Reliability +

4.8.2 Query BT.2

Query: Are the thruster requirements only for one, continuous burn?

Response 1: YES
Discussion: This response implies that there are no attitude adjustment burns. One continuous burn would imply that the satellite is permanently changing its orbital plane or has the potential for one maneuver to evade a potential physical attack. If it is the latter, the Propulsion function must be highly reliable. If is the former, there is no great need for a change in reliability. A YES response to this query also implies that there is either no Attitude Control function or that the attitude adjustments are performed by a means other than propulsion. In the latter case, Attitude Control reliability needs to be moderately increased to ensure proper orientation of the satellite and platform stability in order for the satellite mission to be performed.

Action:	None	
Impact:	Attitude Control	Propulsion
	Reliability ++	Reliability +++

Response 2: NO
Discussion: A NO response implies intermittent burns -- most likely to perform attitude control functions. Many of the items discussed in the YES response to Query BT.1 apply here, save the extended satellite life discussion. Multiple firing imply moderate Propulsion reliability requirements. Likewise, Attitude Control reliability is highly required.

Action:	Ask Query BT.3	
Impact:	Attitude Control	Propulsion
	Reliability +++	Reliability ++

4.8.3 Query BT.3

Query: For intermittent thruster firing requirements, will the average burn time be less than 20 seconds?

Response 1: YES

Discussion: Many attitude adjustments. None, or few, orbital maneuvers. The BT.2 NO response criteria applies.

Action:	Ask Query BT.1	
Impact:	Attitude Control	Propulsion
	Reliability +++	Reliability ++

Response 2: NO

Discussion: Few attitude adjustments. One or more orbital maneuvers, evasive or otherwise. BT.2 YES responses apply.

Action:	None	
Impact:	Attitude Control	Propulsion
	Reliability ++	Reliability +++

4.8.4 Query BT.4

Query: Is the design of the Propulsion system and choice of fuel type one that optimizes fuel efficiency or burn time, i.e., change in velocity parameters?

Response 1: Fuel Efficiency.

Discussion: This implies longer, steadier burns -- those usually used for orbital transfers. Also, fuel efficiency implies a moderately higher requirement for Attitude Control reliability. If we are to conserve fuel, there should be fewer requirements to adjust the platform.

Action:	Ask Query BT.2
Impact:	Attitude Control
	Reliability ++

Response 2: Burn Time (delta V)

Discussion: This implies shorter burns and greater changes in velocity. More attitude adjustments and potential maneuvers against threats. Attitude Control reliability is slightly decreased as platform adjustments are expected to be made frequently. Propulsion reliability is highly increased. The potential for quick, threat evading maneuvers and frequent attitude adjustments imply a multitude of engine firings that must be accomplished to permit the satellite to perform its mission.

Action:	Ask Query BT.2	
Impact:	Attitude Control Reliability	Propulsion Reliability +++

4.9 Communication Frequencies

4.9.1 Query CF.1

Query: Are there methods for transmitting/receiving information under enemy jamming conditions?

Response 1: YES

Discussion: A YES response indicates that the developers believe that jamming presents a reasonable threat to the system. As such, the design of the satellite has most likely incorporated some techniques to permit transmission and receipt of data during those periods where the normal mode of communication is unavailable. Probably the most common technique is the location of ground sites in locations that the enemy will not be able to jam -- Australia, the U.S. mainland, the South Pacific, etc. A second simple technique would be to only transmit when the conditions are favorable -- when no jamming is present. More complicated techniques would include predetermined frequency shifts and intersatellite relays. The more simplistic of these techniques really requires no additional software robustness but are more scheduling and planning problems. Frequency hopping and intersatellite relays require a bit more sophistication -- both in hardware and software requirements. Under frequency hopping conditions, the hardware and software must be able to send and receive signals at a multitude of frequencies. The increased bandwidth capabilities of ground- and space-based jammers require these frequency shifts to be across a number of bands (X, S, C, etc.) rather than merely a shift of 1 or 2 GHz within the X band, for example. This places tremendous requirements on the hardware in terms of optimizing weight and size of instrumentation. Software features such as expandability (a wide spectrum of radio frequencies), reliability (effective and timely shifts of frequencies, availability of all modules), efficiency (optimizing code to reduce memory storage requirements), and survivability (use of alternate frequencies should partial hardware failure occur) are necessary to insure confident communications operations. These factors would all apply to the Data Communications functions. Each of these factors should be significantly increased over standard requirements.

Frequency hopping also implies that we are able to detect the presence of jamming. This capability impacts both the space-borne and ground-based components of the satellite functions Data Communications and Command & Control. This feature is a survivability characteristic in that the failure to operate must be detected and steps taken to re-establish the communication links. As such, Data Communications and Command and Control software survivability need to be significantly increased. Integrity is also impacted as the software must discriminate between an acceptable command, a jamming environment, and an unauthorized command. This issue is examined in greater detail in section on Data Encryption. For our purposes here, however, we can conclude that software integrity requirements need to be slightly increased when frequency hopping techniques are employed.

The second of the more sophisticated techniques -- intersatellite relays -- requires a different set of requirements. Now, each satellite must be in view of at least one other satellite at all times. This will ensure the transferring of data to a satellite that may be jammed from normal space to ground operations. The capability requires a moderate increase in Data Communications reliability. The constellation will be operating more on its own, or autonomously. That is, there will be less direct ground control of each satellite. Not only does this require more reliability, it probably requires a separate design and development of software solely for this purpose. If autonomy was not an initial design consideration, the threat analysis has now deemed that it is necessary, to some degree. This new requirement may impact cost and schedule, but would not cause any significant increase or decrease in the other software quality factors.
Action: Ask Query CF.1.1
Impact: None.

Response 2: NO
Discussion: A NO response most likely means that the designers do not feel that the jamming of communication data is a threat to their system. This position may be based upon the orbital inclination or altitude, the frequency of transmission, or the knowledge of enemy intelligence with respect to their jamming capabilities. In any case, a NO response would not have a significant impact upon the functions and their software quality factors.
Action: Ask Query CF.5.
Impact: None.

4.9.2 Query CF.1.1

Query CF.1.1: What method will be used to counter this jamming?

Response 1: COMMUNICATION IN THE ABSENCE OF JAMMING.
Discussion: Mostly a scheduling and planning problem. The ability to detect the presence of jamming is required. Slightly increase Data Communications integrity requirements.
Action: Ask Query CF.5.

Impact:	Data Communications
	Integrity +

Response 2: FREQUENCY HOPPING.
Discussion: As previously mentioned, significant increases in Data Communications expandability, reliability, efficiency, and survivability. Slight increase in integrity. Command and Control survivability is significantly increased.
Action: Ask Query CF.5.

Impact:	Data Communications	Command & Control
	Expandability +++	Survivability +++
	Reliability +++	
	Efficiency +++	
	Survivability +++	
	Integrity +	

Response 3: INTERSATELLITE RELAYS.
Discussion: As previously mentioned, Data Communications reliability should be moderately increased.
Action: Ask Query CF.5.

Impact:	Data Communications
	Reliability ++

Response 4: NONE OF THE ABOVE.
Discussion: This implies some other technique which may be a combination of the aforementioned ones or some other method we haven't discussed is employed. The techniques mentioned above represent some of the common methods for countering a jamming threat. Other techniques that may influence the quality factors may exist. A "None of the Above" response, in this case, means no information exists to conclude a factor change.

Action: Ask Query CF.5.

Impact: Data Communications
Reliability +

4.9.3 Query CF.2

Query: Are the transmitting and receiving frequencies identical?

Response 1: YES.

Discussion: A YES response indicates that an adversary attempting to jam these communications could accomplish this task with less difficulty than were two, widely separated, frequencies used. Of course, if the frequency chosen was one for which an adversary had no jamming capability (see CF.4), then this would not be an unsound choice. There may, howe·er, be some added confusion involved with the discrimination of incoming and outgoing messages. This would necessitate a slight increase in Data Communications and Command and Control software integrity. The requirements for transmit/receive hardware would be reduced as the same equipment can perform in both modes. However, the reliability and efficiency of this hardware and accompanying software needs to be moderately increased to ensure operational availability. If the hardware or software fails, there is no communication capability. This further necessitate a moderate increase in the software survivability. These last three requirements are for the Data Communications function.

Action: Ask Query CF.3.

Impact:	Data Communications	Command & Control
	Integrity +	Integrity +
	Reliability ++	
	Efficiency ++	
	Survivability ++	

Response 2: NO.

Discussion: A NO response here does not really tell us too much. We know that separate hardware/software is required for each mode. There is also less susceptibility to a complete communications knockout. The answers to other CF queries will determine the true impact of a NO response.

Action: Ask Query CF.2.1.

Impact: None.

4.9.4 Query CF.2.1

Query: Are they within the same band range (X, C, S, etc.)?

Response 1: YES.
Discussion: Some of the same restrictions and needs as in the Query CF.2 YES response apply here. Dual mode hardware may still be employed depending upon the spread within the band. An additional item not mentioned in the aforementioned analysis concerns the maintainability of ground-based Data Communications software. Software failures to communications systems that do not have hardware redundancy or are used in both transmit and receive modes requires a significant increase in maintainability. Why? Because failures here knock out all communications. Software that is maintainable permits errors to be detected and corrected quickly and efficiently -- a must if all communications have ceased. The requirement for this need on-board the satellite is moot until techniques are developed to recover and repair satellites in orbit.
Action: Ask Query CF.3.
Impact: Data Communications (ground portion)
Maintainability +++

Response 2: NO.
Discussion: A NO response, on the other hand, indicates that there are most likely separate hardware and software that perform transmit and receive operations. There may also be hardware redundancy. Even without redundancy, the possibility may exist for the failed component to be temporarily replaced by the still active component. Survivability would then play a role as the ability to operate with existing faults or errors is imperative to ensure the remaining component was available and functioning properly. Moderately increase the survivability requirement.
Action: Ask Query CF.3.
Impact: Data Communications
Survivability ++

4.9.5 Query CF.3

Query: Are there other satellite systems operating at (or near) these frequencies?

Response 1: YES.
Discussion: Potential problems arise when a particular frequency or band become too crowded. Steps must be taken to insure there are unique characteristics for the signals. Discrimination

between signals becomes an issue. Command and Control integrity, both on the ground and on the platform, should be slightly increased. Depending upon the complete functionality of the Data Communications function, some (or all) of this discrimination may be accomplished here. A slight increase in this function's software integrity would also be necessary.

Action: Ask Query CF.4.

Impact:	Data Communications	Command and Control
	Integrity +	Integrity +

Response 2: NO.
Discussion: This could represent a greater design risk to the Data Communications function as a whole but there is no real effect on the degree of any particular software function. There may be reasons that there are no other systems developed at these frequencies -- the technology is not present, there is too great a jamming threat, etc.
Action: Ask Query CF.4.
Impact: None.

4.9.6 Query CF.4

Query: Are the data communication frequencies easily jammed by jammers.

Response 1: YES.
Discussion: If the frequencies are easily jammed from jammers, there are usually two reasons that the developers have continued to pursue these frequencies. One, the mission of the satellite is such that there is no perceived threat from these jammers -- an enemy might not care to jam this system. For military applications this case wold be very unlikely. The second reason would be that the developers are planning to use some of the techniques discussed in the CF.1 query, YES response. As such, no requirements determination can be made here. Instead, Query CF.1 should be asked.
Action: Ask Query CF.1.
Impact: None.

Response 2: NO.
Discussion: A NO response indicates that either the orbital altitude, inclination, or method of operation is such that the existing jammers are not deemed a threat. Under these circumstances, no variations in software requirements would be necessary.
Action: Ask Query CF.5.

Impact: None.

4.9.7 Query CF.5

Query: Are spare communications channels available?

Response 1: YES.
Discussion: Implies that there are methods for communicating in a jamming environment and/or in case of hardware failures. The criteria previously discussed under Query CF.1.1, FREQUENCY HOPPING response, apply.
Action: Ask Query CF.7.

Impact:	Data Communications	Command & Control
	Expandability +++	Survivability +++
	Reliability +++	
	Efficiency +++	
	Survivability +++	

Response 2: NO.
Discussion: Implies no frequency hopping capability. No reduction or increases in software requirements. Communications hardware redundancy should be employed to prevent the need for spare channels.
Action: Ask Query CF.7.
Impact: None.

4.9.8 Query CF.6

Query: Are there methods and procedures for the continuation of communications given a software failure within a transmitter or receiver?

Response 1: YES.
Discussion: This implies that some sort of on-board software redundancy or fault detection and recovery scheme is being utilized. These techniques may be applied in either the Data Communications or Command and Control functions. Software survivability requirements need to be moderately increased for these functions.
Action: None.

Impact:	Data Communications	Command and Control

Survivability ++   Survivability ++

Response 2: NO.
Discussion: No software backup. Therefore, reliability needs to be moderately increased for the Data Communications function.
Action: None.
Impact: Data Communications
Reliability ++

4.9.9 Query CF.7

Query: Is there a requirement for the satellites of the constellation to communicate with each other?

Response 1: YES.
Discussion: The requirements under Query CF.1.1, INTERSATELLITE RELAY response, apply.
Action: Ask Query CF.6.
Impact: Data Communications
Reliability ++

Response 2: NO.
Discussion: No reduction or increase in software requirements.
Action: Ask Query CF.6.
Impact: None.

4.10 Data Encryption

4.10.1 Query DE.1

Query: Will all telemetry, tracking, communications, and mission data be encrypted?

Response 1: YES.
Discussion: Integrity requirements are increased for Data Communications and Command and Control. Encryption implies a need for integrity.
Action: Ask Query DE.6.
Impact: Data Communications   Command and Control

Integrity ++ Integrity ++

Response 2: NO.
Discussion: Secure telemetry is not required and therefore Integrity is not an issue. Both the Data Communications and Command and Control functions are effected.
Action: Ask Query DE.4.
Impact: None

4.10.2 Query DE.2

Query: Will password and/or security clearances be required to access telemetry, tracking and communication data at the receiving terminals?

Response 1: YES.
Discussion: Merely implies that there will be some sort of control placed upon the distribution of system data. Password control is usually provided via the system software, therefore, a slight increase in Command and Control integrity is required. Security clearances are usually enforced physically or through software. If they are provided via the software, another increase in Command and Control integrity would be warranted, bringing the total requirement to a moderate level. These precautions are relatively easy to implement and usually prove difficult to bypass. System entry by unauthorized personnel using authorized passwords is another matter completely.
Action: Ask Query DE.3.
Impact: Command and Control
Integrity ++

Response 2: NO.
Discussion: Means that the data is, one, unclassified/non-sensitive (probably not too likely) or, two, that all potential users are cleared to access all data or, three, physical security is substantial. No direct impact on software quality.
Action: Ask Query DE.3.
Impact: None.

4.10.3 Query DE.3

Query: Will some sort of authentication process be employed by the on-board software to ensure commands received are genuine?

Response 1: YES.

Discussion: The implementation of this feature will greatly increase the system's integrity. As this feature is predominantly provided by the system software, unauthorized access to this software must be guarded. Therefore, integrity requirements for the Command and Control function should be moderately increased. It was at this point that the issues mentioned in section 4.7.1 were encountered. In rethinking the question, we came to the conclusion that since the implementation of this feature increased system integrity, it is not necessarily true that the feature itself needs to possess an increased level of integrity. Providing that this software is highly reliable, the requirement for the feature to possess Integrity has been significantly decreased (from its moderately increased state). If a potential unauthorized user cannot get past this on-board command authentication, the remainder of the system software is buffered from these erroneous commands. Reliability, however, remains high.

Action: Ask Query DE.5.

Impact: Command and Control

Reliability +++

Response 2: NO.

Discussion: Without this feature, unauthorized access to the system will present a greater risk to the system them with the authentication process. Therefore, if a system doesn't require or possess such a feature, chances are it is not necessary. This implies autonomous Command and Control to a certain extent. Then again, it is very unusual for a satellite to be designed that does not receive some type of ground communication. The key word here is "on-board" software. If the authentication is done somewhere else (on the ground, perhaps, via an echo of the message for confirmation, etc.). Assuming there is some sort of authentication, but not on-board, the requirements for reliability and integrity should not be impacted. A ground-based authentication process necessitates, however, an increase of maintainability and survivability of the software. Both in a moderate degree and to the Command and Control function. Why? Being ground-based, there is direct access to the software. The importance of this process in the overall operation of the satellite system is obviously crucial. Failures in this software need to be repaired quickly and to continue to function in a degraded mode while these fixes are being implemented.

Action: Ask Query DE.5.

Impact: Command and Control

Maintainability ++

Survivability ++

4.10.4 Query DE.4

Query: Will all up-linked system commands be encrypted?

Response 1: YES.
Discussion: Command and Control must be more reliable than if data was not encrypted? Not necessarily. It simply has an additional step in the processing of the command. Reliability should be slightly increased. This is to ensure the proper decoding of the command into a form that the system can implement. Because it is located on-board the satellite, the survivability of this software should also be slightly increased to ensure proper operation even if a failure (hardware or software) occurs to the components performing the function.
Action: Ask Query DE.2.
Impact: Command and Control
Reliability +
Survivability +

Response 2: NO.
Discussion: Non-encrypted uplinks can be intercepted and enable an adversary to counterfeit commands. There is no software procedure, even encryption, which will prevent an adversary from receiving a command if it is generated in a region where the adversary's antenna can pick it up. Therefore, proper planning and uplink procedures are required to ensure these links are not intercepted. There is no direct effect on software quality.
Action: Ask Query DE.2.
Impact: None.

4.10.5 Query DE.5

Query: Are there security procedures invoked when an unauthenticated command is received?

A follow on to Query DE.3. Primarily, these security procedures would notify the ground site that unauthorized commands have been received and, most likely, transmit the received command for ground analysis.

Response 1: YES.
Discussion: A YES response implies that there will be an additional system feature that is activated upon receipt and detection of an unauthorized message. This also implies that somehow the fact

that the message is unauthorized has been detected. If this detection routine was not functioning properly there could be some serious consequences to the system. For instance, an authorized command could be rejected or an unauthorized command (from an unauthorized user, not necessarily a command that is unrecognizable) could be implemented. As such, the reliability of this detection routine function, probably located in the Command and Control function, needs to be significantly increased.

Action: Ask Query DE.7.

Impact:	Command and Control
	Reliability +++

Response 2: NO.

Discussion: A NO response indicates that unauthorized messages, should they be detected, are simply ignored. This implies that there is no analysis done on these unauthorized commands to determine their source, their intention, etc. No need to beef up software quality to simply discard messages.

Action: Ask Query DE.7.

Impact: None.

4.10.6 Query DE.6

Query: Is there a capability to change or abandon the encryption codes once the system is deployed?

Response 1: YES.

Discussion: The ability to change or abandon encryption codes is an outgrowth of the detection and analysis of unauthorized commands. Or, perhaps, it could be a routine occurrence to counter the absence of this detection and analysis. In either case, the Data Communications and Command and Control functions would need to have increased reliability and survivability (moderate) to insure the successful transition from one code to the next and the successful operation of this transition should there be a partial hardware or software failure. For ground-based components of this process, a moderate increase in maintainability would aid in the timely correction of any ground based software failures or errors.

Action: Ask Query DE.4.

Impact:	Data Communications	Command and Control
	Reliability ++	Reliability ++
	Survivability ++	Survivability ++

Maintainability ++ Maintainability ++

Response 2: NO.
Discussion: No effect on the software quality factors.
Action: Ask Query DE.4.
Impact: None.

4.10.7 Query DE.7

Query: Is there a software verification procedure by which the Command and Control function ensures the received command is an acceptable command?

This questions differs from DE.3 in that DF 3 determines if the received command comes from an authorized user. This verification determines if that authorized command makes any sense -- i.e. is it a legitimate action for the system to take. This serves as an augmentation to the authentication process.

Response 1: YES.
Discussion: Permits the authentication process, if it exists, to be slightly less reliable as this procedure does serve as a second line of defense. However, without some sort of authentication procedure (highly unlikely), this module must be extremely reliable and survivable. This module is most likely located on-board the satellite which would rule out the need for excessive maintainability.
Action: None.
Impact: Command and Control
Reliability +++
Survivability +++

Response 2: NO.
Discussion: Unacceptable commands will cause software errors. Some of these errors may fail gracefully, some may not. Command and Control survivability (fault tolerance) must be significantly increased to ensure that an unacceptable command will not pull the entire system down but can be discarded without incident.
Action: None.
Impact: Command and Control
Survivability +++

4.11 Fault Tolerance

4.11.1 Query FT.1

Query: Is there a standardized method for performing error detection, handling, and recovery for each software function?

Response 1: YES.

Discussion: When we say that there are standard error detection, handling, and recovery procedures, we can be sure that each function, or the Command and Control function depending upon the answer to FT.4 and FT.5, has a method to increase its reliability and survivability. As mentioned earlier, procedures (or features) to enhance reliability and Survivability need to possess these features themselves. Therefore, based upon FT.4 and FT.5, either the Command and Control or all functions must have a moderate increase in reliability and survivability.

Action: Ask Query FT.4.

Impact: IF FT.4=Yes, THEN All Functions. IF FT.4-No, the Only the Command and Control Function.

Response 2: NO.

Discussion: A NO response can mean one of two things; (1) there is no error detection, or, (2) there are no standard procedures. If the first is the case, it means that survivability has been discarded. Therefore, the reliability of all functions must be extremely high to ensure that there are no errors. This instance of no error detection is very rare. The second case implies that each function performs its own fault tolerance. Again, this would imply that reliability and survivability requirements for these functions should be moderate. However, without standard procedures, testing and error correction will be much more difficult. As such, the software needs to have a slight increase in maintainability to offset the absence of standard procedures. For functions that have ground components, Data Communications and Command and Control, this requirement should have an extreme increase.

Action: Ask Query FT.4

Impact: (1) Reliability +++ (2) Reliability ++
Survivability ++
Maintainability +(+++ for ground components, data communications, and command and control)

4.11.2 Query FT.2

Query: Will critical hardware component (power distribution, command and control processor, communications, etc.) redundancy be employed?

Response 1: YES.
Discussion: Hardware redundancy implies that the mission life of the satellite is probably longer than were no redundancy employed. This implies that the software needs to be designed for a longer life-cycle than normal. This impacts reliability, survivability, and maintainability (for Data Communications and Command and Control ground-based software) over all functions that have hardware redundancy. Depending upon the nature of the redundancy (N-modular, triple modular, etc.) and the actual functions where this is applied, the software requirements will vary. A rule of thumb would be to at least slightly increase the requirements for these three factors across all functions. Should FT.5 = YES, only the Command and Control function should be affected.
Action: Ask Query FT.8.
Impact: IF FT.5=YES, THEN Only the Command and Control Function. IF FT.5=NO, THEN All Functions.

Response 2: NO.
Discussion: The absence of hardware redundancy implies that the software will have to perform even in the presence of hardware failures. The real question is, will the software have to recover from <u>all</u> hardware failures, or only a select group? Query FT.8 will answer this question. For now, we can assume that <u>some</u> software recovery is required. Therefore, survivability requirements need to be slightly increased. This applies to whatever functions are providing fault tolerance. Query FT.8 will determine if this requirement needs to be augmented.
Action: Ask Query FT.8.
Impact: Survivability + (only functions containing fault tolerance)

4.11.3 Query FT.3

Query: Will critical software routine redundancy be employed?

Response 1: YES.

Discussion: Software redundancy is the use of multiple routines to perform identical functions. A calculation is a good example. The sine of a number can be calculated by using a routine that uses the definition of sine (with factorials and all) and another that uses a library function within a given language and, perhaps, a third which uses the cosine or tangent of the appropriate complementary angle. The results are compared and the majority answer is taken. If there is no majority, the software uses the value from the module that has been deemed most accurate or reliable. As this "polling" process increases the reliability of the calculation, the reliability of each individual module does not necessarily have to be increased. As such, there is no need to increase or decrease functional reliability if software redundancy is used.
Action: Ask Query FT.6.
Impact: None.

Response 2: NO.
Discussion: The absence of software redundancy implies that each software module must be more reliable as it is the only module that performs that specific function. As such, reliability should be moderately increased. Further, the absence of backup implies that Survivability (on space-based routine) and maintainability (on ground-based routines) will also be important. Suggest a moderate increase in these factors as well.
Action: Ask Query FT.6.

Impact:	Reliability ++
	Survivability ++
	Maintainability ++

4.11.4 Query FT.4

Query: Will each software function provide its own fault tolerance procedures?

Response 1: YES.
Discussion: If FT.1 = YES, there are no additional requirements. If FT.1 = NO, then the requirements listed in FT.1, NO response, case (2) apply.
Action: Ask Query FT.2.
Impact: See discussion.

Response 2: NO.
Discussion: If FT.1 = YES, ask Query FT.5. If FT.1 = NO, assume there is no Survivability. Enact FT.1, NO response, case (2) requirements for each function.

Action: If FT.1 = YES,

Ask Query FT.5.

Otherwise,

Ask Query FT.2.

Impact: See Discussion.

4.11.5 Query FT.5

Query: Will all fault tolerance responsibility reside with the Command and Control function?

Response 1: YES.

Discussion: FT.1 = YES, FT.4 = NO, FT.5 = YES. Increase Command and Control survivability and reliability moderately.

Action: Ask Query FT.2.

Impact: Command and Control

Reliability ++

Survivability ++

Response 2: NO.

Discussion: FT.1 = YES, FT.4 = NO, FT.5 = NO. An unlikely combination. Most likely all fault tolerance is done by the hardware. Should this occur, software Reliability should be slightly increased for all functions to aid in hardware operation and availability.

Action: Ask Query FT.2.

Impact: Reliability +

4.11.6 Query FT.6

Query: Are error tolerances specified for all functional input data?

Response 1: YES.

Discussion: This checking feature of the software will help to ensure reliability of results and to prevent possible software errors due to manipulations on data that are of the wrong type, out of range, etc. As a result of the addition of this reliability feature, reliability should be slightly increased. This feature will also enhance the survivability of the software as input that may cause faults and errors will be detected before processing is commenced. Thus, it is in our best interest to increase the survivability of this software slightly to ensure that it operates in the presence of faults.

Action: Ask Query FT.7.

Impact: Survivability +
Reliability +

Response 2: NO.

Discussion: The absence of input testing indicates that the software must be able to gracefully handle erroneous input. This implies an increase in survivability to a moderate level. The reliability of the software is not necessarily affected as, given the erroneous input, how could we expect the function to give a reasonable answer? Survivability would ensure the software wouldn't cough too much given the input. Maintainability also arises in those ground-based components. A slight increase in the ability to locate the source of input error and to perform timely corrections is required.

Action: Ask Query FT.7.

Impact: Survivability ++

Command and Control	Data Communications
Maintainability +	Maintainability +

4.11.7 Query FT.7

Query: Are all computational outputs tested to ensure they fall within expected ranges, types, etc.?

This query is very much like FT.6. There is one subtle difference, checking the inputs ensures the data is appropriate for the function it is about to enter. Checking output ensures the data is appropriate for the function it just left. One is necessary to ensure software failures will not occur because of bad input, the other ensures that the function has worked properly. Both features, however, can be designed and implemented in identical manners. As such, the requirements are identical.

Response 1: YES.

Discussion: See FT.6, YES response.

Action: Ask Query FT.10.

Impact: Survivability +
Reliability +

Response 2: NO.
Discussion: See FT.6, NO response.
Action: Ask Query FT.10.
Impact: Survivability ++

	Command and Control	Data Communications
	Maintainability +	Maintainability +

4.11.8 Query FT.8

Query: Should the software provide techniques for recovering from all hardware failures?

Response 1: YES.
Discussion: (1) If FT.2 = YES, the requirements need to be stepped up to a moderate level for all factors. (2) If FT.2 = NO, survivability levels should be highly increased. The reasoning here is that the software is the only method of countering hardware failures or errors.
Action: Ask Query FT.3.

Impact:	(1)	Reliability ++	(2)	Survivability +++
		Survivability ++		
		Maintainability ++		

Response 2: NO.
Discussion: (1) If FT.2 = YES, no change from the FT.2 requirements. There is hardware redundancy and the software only has to counter some hardware faults; (2) If FT.2 = NO, there is also no change from the FT.2 requirements. There is no hardware redundancy and the software only has to counter some hardware faults. Obviously, certain hardware faults may prove fatal as there are no recovery procedures. This is probably a cost/benefit decision.
Action: Ask Query FT.3.
Impact: None.

4.11.9 Query FT.9

Query: Should the software provide alternative methods for onboard message routing should normal communication channels fail?

Response 1: YES.

Discussion: There needs to be spare message channels on-board the platform to ensure the interface between functions should a failure exist. This feature would prevent the isolation of one function from the remainder of the system. This would obviously improve the software survivability. As such, this feature needs to be survivable in and of itself to a slight degree. Maintainability would not be an issue as the communication links previously discussed provide spare channels for the ground-based portions of the Data Communications and Command and Control functions. Additionally, reliability needs to be slightly increased to ensure the successful switch from a failed channel to an alternate channel.

Action: None.

Impact: Reliability +
Survivability +

Response 2: NO.

Discussion: In the presence of communication failures and no alternate channels, the interface between functions is doomed. An extreme increase in the survivability of the software is required to prevent these failures from interrupting system operation. If hardware redundancy is employed (FT.2 = YES), this survivability requirement can be reduced to a moderate level as the redundancy provides a hardware backup.

Action: None.

Impact: Survivability +++ (++ if redundant hardware employed)

4.11.10 Query FT.10

Query: Does the satellite have a "watchdog timer" that determines the probability of satisfactory functional execution?

Response 1: YES.

Discussion: This is yet another method for preventing error or fault manifestation. A time limit is set for execution. If a result has been produced within this space of time, no error checking is performed. Should the processing time be greater than this range, there is a possibility of the software stuck in an endless loop, inadvertant jump in the program counter, etc. If this situation occurs, then a process such as that in Query FT.7 would be employed. The accuracy and reliability of this timing feature, however, needs to be assured so as to only call upon the test routines when necessary. Constant invocation of a checking routine could add considerable time to the overall processing time which could prove crucial depending upon the timeliness requirements of the data being checked. Further, the survivability and maintainability of this watchdog timer routine is required to ensure the availability and operation of the routine. For all three factors, a moderate increase is required.

Action: None.

Impact: Survivability ++
Reliability ++
Maintainability ++

Response 2: NO.

Discussion: Given a checking routine such as in Query FT.7, and the absence of this watchdog timer, the efficiency of the software most likely needs to be increased moderately. Why? Those other routines take time and they may seriously impact processing time. Increasing the efficiency of the code to permit faster average processing times will enable those test routines to offer a true benefit.

Action: None.

Impact: Efficiency ++

4.12 High-Order Language

Given the way in which satellite systems are developed and the need for satellite software to be adaptable, the use of a high-order language is one of the most important contributors to this adaptability requirement. Use of a high-order language enhances program modularity, generality, self-descriptiveness, simplicity, functionality, and independence. These attributes add to the maintainability, expandability, reusability, reliability, and survivability of the system. Employing a high-order language in the development of system software might not require the software to possess these quality factors, but it will add to the overall system quality.

4.12.1 Query HOL.1

Query: Will the software be developed in a high-order computer language such as JOVIAL, FORTRAN, or Ada?

Response 1: YES.

Discussion: The use of a high-order language (HOL) does not impact the requirements of the software.

Action: Ask Query HOL.2.

Impact: None.

Response 2: NO.
Discussion: The use of a language other than an HOL (assembly, machine) does not aid in the attainment of the software quality factor criterion of modularity, generality, self-descriptiveness, simplicity, or independence. As such, there is a need for the software to compensate for these missing qualities that an HOL would provide. Therefore, the requirements for maintainability, survivability, reusability, reliability, and expandability all need to be increased to a moderate degree. This will ensure that the software will possess the same levels of these factors as code written in an HOL.
Action: Ask Query HOL.5.

Impact:	Survivability ++
	Reusability ++
	Maintainability ++
	Reliability ++
	Expandability ++

4.12.2 Query HOL.2

Query: Will a compiler be used to optimize machine code size and execution speed?

Response 1: YES.
Discussion: A compiler that optimizes code takes inefficient HOL code and turns it into efficient machine code. As such, the efficiency is a concern.
Action: Ask Query HOL.3.
Impact: Efficiency ++

Response 2: NO.
Discussion: A compiler that does not optimize machine code implies that the HOL code needs to be fairly efficient on its own accord. Therefore, the requirement for software efficiency is minimal.
Action: Ask Query HOL.3.
Impact: None

4.12.3 Query HOL.3

Query: Will the compiler produce machine code for a processor that is currently space-qualified?

Response 1: YES.
Discussion: This response does not impact the software quality factors.
Action: Ask Query HOL.4.
Impact: None.

Response 2: NO.
Discussion: Two reasons a compiler might be used that produces machine code for a non-space qualified processor are:
(1) The code will be run on a ground-based processor, or
(2) Space qualification for the processor is pending.
The first of these instances does not cause any adverse impact on the software quality. The second instance does, however. A processor that has not been successfully operated in a space environment presents a high risk to the satellite's development. The risks may, however, be reasonable given the processor's capability above those currently available. In any case, the reliability, survivability, and efficiency or the software should be increased to offset the unknown quantities of the processor. A moderate requirement should be sufficient.
Action: Ask Query HOL.3.1.
Impact: None.

4.12.4 Query HOL.3.1

Query: Is the processor ground-based?

Response 1: YES.
Discussion: No impact as previously discussed.
Action: Ask Query HOL.4.
Impact: None.

Response 2: NO.
Discussion: Moderate impact to reliability, survivability and efficiency as previously discussed.
Action: Ask Query HOL.4.

Impact:	Reliability ++
	Survivability ++
	Efficiency ++

4.12.5 Query HOL.4

Query: Will a standard subsets of the high-order language be utilized so as to minimize the need for development of specialty host processors?

Response 1: YES.
Discussion: No impact on the software quality.
Action: Ask Query HOL.5.
Impact: None.

Response 2: NO.
Discussion: Implications are the same as those of Query HOL.3.1, NO response.
Action: Ask Query HOL.5.
Impact: Reliability ++
Survivability ++
Efficiency ++

4.12.6 Query HOL.5

Query: If assembly code is used, will the assembly code be for a processor that is currently space-qualified?

Response 1: YES.
Discussion: No impact.
Action: Ask Query HOL.6.
Impact: None.

Response 2: NO.
Discussion: As with Query HOL.3, an additional query needs to be added that would determine if the processor was ground-based or space-based.
Action: Ask Query HOL.5.1.
Impact: None.

Response 3: NOT USED.
Discussion: Assembly language is not used.

•

Action: If HOL.1 = YES,

Ask Query HOL.7.

Otherwise, Ask Query HOL.6.

Impact: None.

4.12.7 Query HOL.5.1

Query: Is the processor ground-based?

Response 1: YES.
Discussion: No impact.
Action: Ask Query HOL.6.
Impact: None.

Response 2: NO.
Discussion: Much like that for Query HOL.3.1, NO response. Since we're talking assembly language, reusability and expandability requirements may be needed as well. If HOL.1 = YES, then both HOL and assembly languages are being used. If this is the case, expandability and reusability requirements should be moderately increased on those functions using assembly language.
Action: If HOL.1 = YES,

Ask Query HOL.7.

Otherwise Ask Query HOL.6.

Impact: (1) Reliability ++
Survivability ++
Efficiency ++

If HOL.1 = YES,
case (1) requirements above, plus,
Reusability ++
Expandability ++

4.12.8 Query HOL.6

Query: Will all satellite function software be developed using identical languages and subsets thereof?

Response 1: YES.
Discussion: This is the preferred scenario as far as minimizing the impact on the software quality factors. There is no significant impact.
Action: Ask Query HOL.7.
Impact: None.

Response 2: NO.
Discussion: The use of different languages and/or subsets thereof take away the inherent maintainability, expandability, and reusability characteristics that would be present were one subset or language used. As such, these factors should be slightly increased to insure necessary software quality.
Action: Ask Query HOL.7.
Impact: Expandability +
Maintainability +
Reusability +

4.12.9 Query HOL.7

Query: Will the language employed contain inherent fault tolerance detection and recovery capabilities such as Ada's exception handling?

Response 1: YES.
Discussion: This will make the implementation of software reliability and survivability easier; but, it does not impact the need for these factors.
Action: Ask Query IC.1.
Impact: None.

Response 2: NO.
Discussion: The absence of inherent fault tolerance features within the programming language puts an added requirement on the software -- that of providing those fault tolerance procedures the language does not possess. Working in C would be a nightmare here because there is no type checking whatsoever, as one example. Again, however, we get into the bind where the need to manually provide a feature doesn't necessarily mean the quality factor the feature will enhance is required by the feature. In this case, it is suggested that reliability and survivability requirements be slightly increased per our section 4.7.1 discussion.

Action: Ask Query IC.1.
Impact: Reliability +
Survivability +

4.13 Interface Complexity

4.13.1 Query IC.1

Query: Are the inputs, outputs, and processing functions clearly defined for all software functions?

Response 1: YES.
Discussion: One would hope that the answer to this question would be YES. A clearly defined description of functional activities and expected inputs and outputs adds tremendously to overall correctness of the final code, as well as its testing, verification, and maintenance. It also adds to the subsequent expansion and reuse of the code as its purpose, process, and data are all meticulously documented. While this process does contribute to code quality, there is no direct code quality requirements that this process specifies.
Action: Ask Query IC.2
Impact: None.

Response 2: NO.
Discussion: While this situation is not conducive to the development of reliable, survivable, efficient, maintainable, or correct code, it is also the situation where the best design is usually not implemented correctly. It is difficult enough to adequately develop code when the design is detailed and documented. When the design is sketchy and the documentation incomplete the problems are even more extreme. Where this is the case, at least a moderate increase in each of these factors for whichever functions are impacted is required. We can add expandability and reusability to this as we have already set the precedence of adaptability quality factors, and, next to using an HOL, a detailed design and full documentation is the next best activity in assuring software adaptability.

Action: Ask Query IC.2.

Impact: Reliability ++
Survivability ++
Efficiency ++
Maintainability ++
Correctness ++
Expandability ++
Reusability ++

4.13.2 Query IC.2

Query: Are there methods for operational testing of function interfaces to verify data content and format?

Response 1: YES.
Discussion: As a software feature that enhances system Reliability and Survivability, we can place a slight increase on the Command and Control software for these quality factors. As most of these interfaces will be located on-board the satellite, Maintainability is not an issue.
Action: Ask Query IC.3.
Impact: Command and Control
Survivability +
Reliability +

Response 2: NO.
Discussion: No operational testing would imply that these interfaces must be thoroughly tested and verified prior to deployment. Command and Control survivability and reliability should be increased moderately to insure their proper operation.
Action: Ask Query IC.3.
Impact: Command and Control
Survivability ++
Reliability ++

4.13.3 Query IC.3

Query: Have all software function interfaces been standardized?

Response 1: YES.
Discussion: Once again, an attractive attribute of the software but it doesn't impact the software quality factors. It does add to the correctness and maintainability of the interface software but does not imply a variation in the requirements for software quality.
Action: Ask Query IC.4.
Impact: None.

Response 2: NO.
Discussion: This implies that the interface between each software function is unique. This would impact correctness and maintainability in that the assurance that the interfaces conform to standards and specifications is doubtful, at best, and the uniqueness of the interfaces will prevent the normal learning curve to apply to the detection and correction of interface errors as each interface is different. As such, correctness and maintainability requirements need to be increased slightly.
Action: Ask Query IC.4.
Impact: Correctness +
Maintainability +

4.13.4 Query IC.4

Query: Are there methods for the Command and Control function to authenticate/verify receipt of messages to/from other software functions?

Response 1: YES.
Discussion: The authentication and verification process here is very similar to that performed on commands received from the ground sites. Commands still need to be verified in terms of the sender/receiver, the verification that the message received/sent is appropriate /correct, and, in the case of receipt from the Data Communications component, the command received is from an authorized user. The reliability of this software needs to be moderately increased to assure the proper performance of this feature. Incorrect operation could lead to catastrophic errors or system failure. Additionally, survivability and maintainability are impacted as the operation of this software in the presence of errors and faults, as well as the timely detection and correction of these errors and faults, is highly desired. A moderate increase in these factors is necessary.
Action: Ask Query IC.5.
Impact: Command and Control
Reliability ++
Survivability ++

Maintainability ++

Response 2: NO.
Discussion: The absence of procedures to authenticate/verify messages/commands within the Command and Control function implies that there must be extreme software reliability and survivability within this function. This is to ensure that any errors in message generation do not significantly impact the individual functions. Similarly, ground-based Command and Control activities need to increase maintainability requirements to a moderate level to ensure timely detection and correction of errors.
Action: Ask Query IC.5.
Impact: Command and Control
Reliability +++
Survivability +++
Maintainability ++

4.13.5 Query IC.5

Query: Have the functional interfaces been designed in a manner to permit minimal data transfer time and optimal processing efficiency?

Response 1: YES.
Discussion: This feature greatly benefits the efficiency of the functional interfaces. This has most likely been achieved through the use of assembly language routines (noted for fast processing speeds) or an optimizing HOL compiler. In either case, the software must possess some degree of efficiency above the norm. While this is different from the Query HOL.2 analysis -- the question there specifically stated that an optimizing compiler was used. If that question has been answered YES, then these requirements would coincide with the Query HOL.2, YES response. Otherwise, a slight increase in efficiency, as previously stated, is required.
Action: Ask Query IC.6.
Impact: If HOL.2 = YES
then No Impact
Otherwise,
Efficiency +.

Response 2: NO.

Discussion: Optimal processing times are usually critical in the fulfillment of space missions, especially where surveillance and communications are of concern. A design that has not optimized the functional interfaces is most likely relying on highly reliable and highly efficient software to do it for them. As such, the requirements for these factors should be at an extremely high level. This will ensure the requirements for timely mission data are delivered and that there is ample exchange of information within the system to make this timeliness possible.
Action: Ask Query IC.6.
Impact: If HOL.2 = YES
then Reliability ++
Efficiency +.
Otherwise,
Reliability +++
Efficiency +++.

4.13.6 Query IC.6

Query: Are there methods for recovering data that is lost due to a software interface failure?

Response 1: YES.
Discussion: If methods are available for recovering data it probably means that there is some kind of back-up memory which stores commands, data, etc. There is probably a limit on the number of commands/pieces of information that can be stored, with the older pieces of information being lost as new information is saved. This would permit, upon reconfiguration, the regeneration of the history back-up. Each function would need to maintain their own backup to ensure that commands that have already been acted upon are not regenerated. The methods for backing up, reconfiguring, and ensuring data recovery need to be reliable. Probably no more reliability is required than would normally be implemented. The efficient storage and accessing of the data, however, is most definitely required. This would permit the storage of the maximum number of commands/information possible which would greatly aid in the overall system survivability. As such, efficiency requirements should be moderately increased. Likewise, survivability requirements should be slightly increased.
Action: Ask Query IC.7.
Impact: Efficiency ++
Survivability +

Response 2: NO.

Discussion: The impact of absence of methods for recovering lost data is severe. Operational capability of the platform is in severe jeopardy. Reliability, survivability, and maintainability all need to be increased significantly. This will minimize the probability of interface failures and data loss, while increasing the probability of operation in the presence of faults and timely detection, correction, and recovery from these errors/faults.

Action: Ask Query IC.7.

Impact:	Reliability +++
	Survivability +++
	Maintainability +++

4.14 Movable/Adjustable Subsystems

4.14.1 Query MAS.1

Query: Are the mission, power, and communication requirements such that the attitude of the platform must be within strict tolerances in order for proper subsystem operation?

Response 1: YES.

Discussion: When platform attitude needs to be kept within strict tolerances, certain activities are implied. First, there will be many attitude adjustments. This implies a need for greater Attitude Control reliability. Second, more engine firing will be required to accommodate these extra adjustments. This means an increase in Propulsion efficiency and reliability, per our discussions with Burn Time. Requirements for each of these are identical to Query BT.2, NO response.

Action: Ask Query MAS.2.

Impact:	Attitude Control	Propulsion
	Reliability ++	Reliability ++
		Efficiency +

Response 2: NO.

Discussion: There is no need for increased Propulsion reliability or efficiency. There would, however, be additional requirements placed upon the Power and Data Communications functions in that they must be able to perform their functions when not necessarily in an optimum configuration. This would entail a slight increase in efficiency and reliability to ensure the optimum use of the hardware resources and the successful completion of their mission under adverse circumstances.

Action: Ask Query MAS.2.

Impact:	Power	Data Communications
	Reliability +	Reliability +
	Efficiency +	Efficiency +

4.14.2 Query MAS.2

Query: Will there be methods or procedures that ensure the continuation of mechanically steered antennas and electronically steered mechanism functions in the event of their failure?

Response 1: YES.
Discussion: For mechanically steered antennas and the requirement for strict platform attitude tolerances (MAS.1 = YES), a failure to the mechanisms that adjust these antennas could prove catastrophic. The most logical choices for the continuation of operations is either through hardware redundancy or, for software failures, ground generated commands that specifically task the Attitude Control function to move antennas, thus, perhaps, bypassing the failed module. Hardware redundancy, in this instance, has no direct impact on software requirements. Ground generated commands that "manually" maneuver these antennas do have impacts on software requirements. Software procedures that detect and analyze hardware and software failures need to be highly reliable. This would include any procedure within the Command and Control or Attitude Control functions that perform this activity. Additionally, there could be movable portions of the Power, Propulsion, or Data Communications functions that may not operate properly and would need to contain procedures to detect these failures. As such, maintainability (error detection), survivability (continuation of operation), and reliability need to be moderately increased to assure the combined operation of these functions.
Action: None

Impact:	Reliability ++
	Maintainability ++
	Survivability ++

Response 2: NO.
Discussion: Without defined procedures and methods of countering these failures, we must insure they do not happen. Hardware failure prevention is beyond the realm of software capabilities. Software failures, however, are not. Features such as input/output checking, timing constraints, software redundancy, etc. need to be employed to eliminate the probability of these errors occurring. This implies a high degree of Reliability, Survivability, and Maintainability in the system software.

Action: None

Impact: Reliability +++
Maintainability +++
Survivability +++

4.15 Memory Utilization/Management

4.15.1 Query MU.1

Query: Are there software procedures that will ensure the minimization of data loss due to memory failure?

Response 1: YES.
Discussion: The simplest procedure to ensure minimization of data loss is to back-up the data in various locations and to periodically update this back-up. This problem is very similar to Query IC.6, except this is at the data level rather than the command level. These procedures will enhance the reliability and survivability of the system and, as such, these factors should be slightly increased. Additionally, software efficiency is needed to a moderate degree to compensate for the increased execution time needed to maintain data backups.
Action: Ask Query MU.2.
Impact: Reliability +
Survivability +
Efficiency ++

Response 2: NO.
Discussion: Without procedures to ensure the minimization of data loss to memory failure, we must again rely on the hardware not to fail. There is no impact on software requirements.
Action: Ask Query MU.2.
Impact: None.

4.15.2 Query MU.2

Query: Will there be any software fault detection and recovery techniques used to specifically detect main memory failure and to provide restoration?
Response 1: YES.

Discussion: These procedures will most likely resemble those diagnostic routines that prepare, check, and verify RAM and disk storage space on a PC. These routines are inherently survivable. They detect errors and permit memory storage to continue, but do need to be moderately reliable in order to ensure data is not stored in what has been determined to be a bad sector or address of RAM.

Action: Ask Query MU.3.

Impact: Reliability ++

Response 2: NO.

Discussion: No real impact to the software. This implies that the hardware needs to be highly reliable.

Action: Ask Query MU.3.

Impact: None.

4.15.3 Query MU.3

Query: Is the storage of redundant files minimized throughout the system?

Response 1: YES.

Discussion: Assuming this minimization has not been done at the expense of ensuring reliable back-up, this would indicate the software needs to be moderately efficient in its back-up procedures. In addition, this procedure needs to be moderately reliable to ensure the data back-up is not compromised.

Action: None.

Impact: Efficiency ++
Reliability ++

Response 2: NO.

Discussion: A memory hog. Software efficiency should be significantly increased to help in the reduction of the redundant files. Other software factors are not impacted.

Action: None.

Impact: Efficiency +++

4.16 Orbital Characteristics

4.16.1 Query OC.3

Query: Have the communications capabilities been optimized for a specific orbital altitude and inclination?

Response 1: YES.
Discussion: There is no impact to software quality.
Action: Ask Query OC.4.
Impact: None.

Response 2: NO.
Discussion: If the communication capabilities have not been optimized for this specified orbit, there may be periods where communication between ground and satellite or between satellite and satellite may be difficult, if not impossible. The impossible is tough to correct with software, the difficult can become more probable if we ensure the availability and operation of the communication software. This implies that the reliability, survivability, maintainability, and efficiency of the Data Communications function be developed at a moderate increase over normal requirements. While this addition to the software's quality may not ease the difficulty of some transmissions, it should aid in those cases where the function is performing at the outer fringes of its capability. Whether the cost of achieving this additional capability is worth the benefit of having it remains to be seen.
Action: Ask Query OC.4.
Impact: Data Communications
Survivability ++
Reliability ++
Maintainability ++
Efficiency ++

4.17 Processing Capabilities

4.17.1 Query PC.1

Query: Will there be any on-board processing of mission data?

Response 1: YES.
Discussion: On-board processing of mission data implies an extra requirement for the Command and Control function -- data processing. This not only impacts the Command and Control function, but also the Data Communications function. With processed data now being downlinked, encryption will surely be used. The processing of mission data implies a need for the processed data to accurately reflect the contents of the unprocessed data. As such, reliability of the software that processes this data is important. Likewise, the survivability of this software is crucial. Processing speed and efficiency are also important as this processed data usually has some timeliness associated with it. After all, that's why the processing is done on-board.
Action: Ask Query PC.2.
Impact: Command & Control / Data Communications
Reliability ++
Survivability ++

Response 2: NO.
Discussion: There is no impact upon the software quality.
Action: Ask Query PC.2.
Impact: None.

4.17.2 Query PC.2

Query: Will each function have its own processing capability, as opposed to the Command and Control function accomplishing all processing?

Response 1: YES.
Discussion: Implies processing within each function. No additional software requirements.
Action: None.
Impact: None.

Response 2: NO.
Discussion: This implies centralized processing and would require a moderate increase in Command and Control efficiency, reliability, survivability, and maintainability.
Action: None.

Impact:	Command and Control
	Survivability ++
	Reliability ++
	Maintainability ++
	Efficiency ++

4.18 Power Sources

4.18.1 Query PS.1

Query: Are there means to operate the system should the main power source fail?

Response 1: YES.
Discussion: If the main power source should fail, there may be additional battery capacity to permit the satellite to continue operations until either the main power source is restored or these extra batteries are drained. As this deals with an extra hardware resource, the efficiency of the Power function should be moderately increased. As there may be power back-ups for each of the individual functions, the efficiency of all functions should be increased slightly.
Action: Ask Query PS.2.

Impact:	Power	All Other Functions
	Efficiency ++	Efficiency +

Response 2: NO.
Discussion: The absence of a power source back-up implies the power source should have a high probability of availability and operation -- this implies high reliability, survivability, and efficiency of the Power function.
Action: Ask Query PS.2.

Impact:	Power
	Efficiency +++
	Reliability +++
	Survivability +++

4.18.2 Query PS.2

Query: Are there methods for dissipating excess energy so as to minimize enemy detection and tracking which may be used to direct threats against the satellite?

Response 1: YES.
Discussion: If the excess heat is dissipated in an efficient manner, there should be no added threat to the system from enemy attack. This dissipation of excess energy implies a very efficient cooling system is in place for the Propulsion and/or Power functions. As this dissipation is crucial to the survivability of the platform and to the performance of the satellite's mission, survivability of the software needs to be extremely high.
Action: Ask Query PS.3.
Impact: Propulsion/Power
Efficiency +++
Reliability +++
Survivability +++

Response 2: NO.
Discussion: If there is no heat dissipation capability, it is because it is not needed -- i.e., the natural dissipation of heat in the space environment is sufficient to ensure proper operation and non-illumination of the platform. There are no added requirements placed upon the software's quality.
Action: Ask Query PS.3.
Impact: None.

4.18.3 Query PS.3

Query: What is the source of the platform's electrical power?

Response 1: SOLAR.
Discussion: A method of power generation that has been used on numerous satellites over the course of 30-odd years of satellite development. No additional software requirements are necessary.
Action: Ask Query PS.4.
Impact: None.

Response 2: NUCLEAR.
Discussion: Very similar to the OTHER response to FL.2. Heat dissipation becomes a tremendous problem and our experience with this type of power generation in space is very limited. Requirements are as in FL.2, OTHER response.
Action: Ask Query PS.4.

Impact:	Propulsion	Power
	Reliability +++	Reliability +++
	Survivability +++	Survivability +++
	Maintainability ++	Maintainability ++
		Efficiency +++

Response 3: BATTERIES ONLY.
Discussion: Batteries that cannot be recharged are used on small, short life-cycle satellites for very specific, many times highly classified, missions. This ensures little or no heat dissipation and very efficient, reliable operation. When the batteries are discharged, the mission is over. No additional software requirements are needed.
Action: Ask Query PS.4.
Impact: None.

4.18.4 Query PS.4

Query: Are there methods for continuing power distribution to a function in the event of a failure to the main method of distribution?

Response 1: YES.
Discussion: Operating in the presence of errors or failures implies increased software survivability -- to a moderate degree in this case. There is also a hardware component of this back-up distribution process. This would indicate a requirement for slightly increased efficiency to ensure optimal management of these back-up resources.
Action: None.

Impact:	Power
	Efficiency +
	Survivability ++

Response 2: NO.

Discussion: As in other stated cases, no back-ups mean higher reliability, survivability, and efficiency in the main systems. A significant increase is necessary due to the importance of the Power function to the overall mission of the satellite.

Action: None.

Impact: Propulsion
Efficiency +++
Reliability +++
Survivability +++

4.19 Testing

4.19.1 Query TE.1

Query: Will the software perform tests (e.g. Built-In Tests (BIT)) to assure on-board hardware operation and availability?

Response 1: YES.

Discussion: There will be software diagnostics to assess the status and health of the function hardware. This feature will ensure the reliability of the satellite functions and augment the survivability and maintainability of the function. The reliability of these diagnostic routines is important to their proper functioning, which in turn ensures the detection of errors, potential situations where failures may occur, and aid in the correction of these problems. Therefore, software reliability needs to be moderately increased. The operation of these diagnostics in the presence of errors is also important. The survivability needs to be increased moderately.

Action: Ask Query TE.3.

Impact: Reliability ++
Survivability ++

Response 2: NO.

Discussion: The absence of software initiated tests for all functions (or only some of the functions) presents a tremendous void in assuring the reliable, survivable, and efficient operation of the function hardware. For functions not containing testing procedures, these three factors need to be significantly increased to ensure the continuous operation of the function hardware and software.

Action: None.

Impact: Reliability +++
Survivability +++

Ffficiency +++

4.19.2 Query TE.2

Query: Will the softare be developed in such a manner as to permit modular testing?

Response 1: YES.
Discussion: This has no impact on software quality.
Action: Ask Query TE.4.
Impact: None.

Response 2: NO.
Discussion: This implies that the software needs to be redesigned or that the test plan needs to be rethought. There are DoD standards and, most likely, contract specifications which will require modular testing. This obviously impacts software correctness, as well as maintainability. If only system testing is allowed, module maintenance will be a nightmare. Correctness and maintainability requirements need to be significantly increased.
Action: Ask Query TE.4.
Impact: Correctness +++
Maintainability +++

4.19.3 Query TE.3

If Query TE.1 = NO, skip this Query.

Query: Will there be software procedures that will correct those discrepancies discovered during on-board testing?

Respoonse 1: YES.
Discussion: Rather than actually "correct those discrepancies", the software will literally permit operation in spite of them. This obviously requires a tremendous level of software survivability. Other factors are not directly influenced.
Action: Ask Query TE.4.
Impact: Survivability +++

Response 2: NO.
Discussion: There are no additional requirements.

Action: Ask Query TE.4.
Impact: None.

4.19.4 Query TE.4

Query: Has source code been thoroughly inspected and evaluated in order to reduce the redundancy by maximizing the use of macros, procedures, and functions?

Response 1: YES.
Discussion: No additional software requirements.
Action: None.
Impact: None.

Response 2: NO.
Discussion: This indicates that a comprehensive software review has not been accomplished. This impacts the correctness of the code. Additionally, unless the modularity of the code is optimized, maintainability, survivability, reliability, reuseability, and expandability are impacted. The only requirement of the software that this response implies, however, is a moderate increase in correctness. The other system factors benefit from correct code.
Action: None.
Impactt: Correctness +++

V. Conclusions and Recommendations

5.0 Conclusions

The main objective of this study was to build a baseline knowledge base for ASQS. Towards this objective, five Air Force mission areas were reviewed and two specific software types, Intelligence System and Satellite System were investigated. Domain specific knowledge was captured through expert interviews and research, the salient system characteristics were analyzed, and estimates were made on their quality impacts within the context of the Software Quality Framework.

The five Air Force mission areas, Avionics, Armament, Missile-Space, C^3 and Force-Mission Management, were examined as part of the Mission Area Analysis (MAA). A schema of missions and system categories (software types) was created based on Air Force product divisions such as Space Division, Electronics Systems Division, etc. and the types of systems they develop or manage.

The MAA uncovered two significant points. First, the five mission areas could be reduced to four. Force-Mission Management is a command and control mission which from a software perspective, is functionally similar to a C^3 system. Second, the five mission areas include a the large number of systems with different software characteristics that ASQS must address. From a knowledge acquisition standpoint, it is a sizeable domain to be examined. This posed several problems to include access and availability of domain experts, the amount of financial and personnel resources required to capture and validate the data, and the overall quality of the knowledge base. For these reasons, this analysis was reduced to the two software types, Satellite Systems and Intelligence Systems.

Notional architectures were developed for the generic projects under the intelligence and satellite systems. A-level, and to a limited degree, B-level specifications were reviewed to create the generic projects for a given software type. Based on the notional architectures, domain experts addressed the software quality issues but generally only at the system level. This reflected a general lack of understanding and experience with the Software Quality Framework. Because of this, rules linking functional and mission characteristics to software quality factors were primarily derived from interpretation of the expert or research data. No standard quality specification process was uncovered in this analysis.

The satellite and intelligence systems software analysis produced suggested rules and queries. The rules were divided into Class I (characteristic to factor) and Class II (characteristic to characteristic) rules. The queries are questions presented to a user when additional information is needed by ASQS to conclude a functional characteristic or a factor requirement. Two approaches were used. The first was to develop Class I rules, followed by the Class II rules, and then the related queries. This approach was best for processes that were well-defined such as a risk index computation for determining computer security levels. The second was to develop queries that have software quality implications and then derive the rules. This approach was appropriate when there was no consensus on a process to decide the factor requirement (e.g., portability) but there was some evidence to suggest a rule. A majority of the framework factors seemed to fit this category. Establishing their requirements was more an arbitrary process then a systematic one.

5.1 Recommendations

The purpose of ASQS is to transition the concept of software quality into practical use. Much effort has been placed in developing the expert interface, but the greater problem is the knowledge base and its acquisition. For ASQS to be successful, it not only must overcome the difficulties in representing key elements of a quality process, it also must define those elements. The Software Quality Framework is suppose to define quality, but a lack of validation and general use has failed to make it an industry or DoD accepted standard.

The future of ASQS is tied to the Framework or possibly another quality specification and measurement process. In either case, knowledge acquisition must be conducted in conjunction with a development program with free access to the experts, their design process, and development environment. Such an effort would have the greatest chance of identifying the key requirements and design decisions that impact software quality.

LIST OF REFERENCES

Air Force Space Command, Electronic Systems Division, Granite Sentry Phase I System Specification, GS-SS-01-87-11, Draft, November 1987.

Abrams, M.D., Schaen, S.I., .and Schwartz, M.W., Strawman Trusted Network Interpretation Environments Guideline, Proceedings of the 11th National Computer Security Conference, October 1988.

Baker, E. and V.L. Cooper, Transitioning Software Quality Metrics Into General Use, Unpublished paper, September 1987.

Bowen, T.P, Wigle, G.B., and Tsai, J.T., Specification of Software Quality Attributes Software Quality Evaluation Guidebook, Rome Air Development Center, Report No. RADC-TR-85-37, Vol. III, February 1985.

Bowen, T.P, Wigle, G.B., and Tsai, J.T., Specification of Software Quality Attributes Software Quality Specification Guidebook, Rome Air Development Center, Report No. RADC-TR-85-37, Vol. II, February 1985.

Defense Communication Agency, Multiple Satellite System Specification, Draft, February 1987.

Department of Defense-Computer Security Center, Technical Rationale Behind CSC-STD-004-85: Computer Security Requirements, CSC-STD-004-85, June 1985.

Department of Defense-Computer Security Center, Trusted Computer System Evaluation Criteria, DoD-5200.28-STD, December, 1985.

Johnson, H.L.and Layne, Modeling Security Risk in Networks, Proceedings of the 11th National Computer Security Conference, October 1988.

Koss, W.E., The Selection and Management of Software Prototypes, Air Force Space Division, Report No. DoD-VA946-PM-89-1, March 1989.

Landwehr, C.E. and H.O. Lubbes, Determining Security Requirnents for Complex Systems with the Orange Book, Proceedings of the 8th National Computer Security Conference, September 1985.

Landwehr, C.E. and H.O. Lubbes, An Approach to Determining Computer Security Requirements for Navy Systems, Naval Research Laboratory Report No. 8897, May 1985.

Lubofsky, M and Koss, W.E., Evaluation Criteria for the Review of Software Requirements and Computer Program Development Specifications, Air Force Systems Command Space Division, Los Angeles, CA., October 1987.

Nugent, W, Computer Security Considerations for a Tactical Army System, Proceedings of the 11th National Computer Security Conference, October 1988.

Presson, E., Software Test Handbook Software Test Guidebook, Rome Air Development Center, Rome, NY., Report No. RADC-TR-84-53, Vol. II, March 1984.

Space and Naval Warfare Systems Command, OSIS Baseline Upgrade (OBU) Security Architecture, PD60-M01125, December 1988.

Space and Naval Warfare Systems Command, OSIS Baseline Upgrade (OBU) Program Design Specification/Program Design Document, PD60-S00960, August 1988.

Space and Naval Warfare Systems Command, OSIS Baseline Upgrade (OBU) Software Requirements Specification, PD60-S01105, April 1988.

Strategic Defense Initiative Office, SDI System Requirements and Rationale, CDRL Item B003, Contract MDA903-85-C-0068, February 1988.

Strategic Defense Initiative Office, Allocated Functional Requirements, CDRL Item B004, Contract MDA903-85-C-0068, February 1988.

PREFACE

To provide a focus, and develop meaningful results for this effort, it was necessary to develop top-level notional (functional) architectures for the Air Force Mission Areas. These mission area notional architectures provide a top level allocation of mission area functions. They are presented in this appendix in an "outline" format and correspond to the hierarchical formats presented in Sections 3 and 4.

These notional architectures are composites, developed by ATI from multiple Air Force, Navy, DOD, and NASA system specifications. The notional architectures are designed to be generic, that is, they are designed to be representative of multiple systems. A primary discriminator in the selection of the top level functions was the commonality of functions (consensus) among the different systems.

The goal of the notional architectures is to provide a standardized starting place/methodology that can be readily expanded for specific systems. These notional architectures were developed to a level that will allow users to have a strong "stepping-off" point, without burdening them with excessive detail that could preclude effective detailed functional allocations. The notional architectures are contained in the following pages of this appendix.

ARMAMENT

Range Systems

Representative Projects:	Defensive and Offensive Radar
	Tactical Threat Systems
	Strategic Threat Systems
	Advanced Threat Systems
	Targeting and Tracking Systems
	Electronic Countermeasures

Guided Weapon Systems

Representative Projects:	Advanced Medium Range Air-to-Air Missiles
	Remotely Piloted Vehicle

Generic RPV

Mission Management

Preflight Planning

- **Data Input**
- **Data Update**
- **Data Validation**

Communications

- **Uplink Management**
- **Downlink Management**

Training

- **Flight Dynamics Simulation**
- **Interact Image Simulation**

Maintenance

- **Recovery**
- **Console Operation**
- **Fault Detection**
- **Fault Location**
- **Emergency Procedures**

Mission Execution

Flight Execution

- **As Planned**
- **Deviation from Plan**
- **Handoff**
- **Lost Link**
- **Cued Landmark**

Navigation

- **Waypoint**
- **Dead Reckoning**
- **Operator Directed**
- **Preplanned**

Maneuver

- **Circle**
- **Racetrack**
- **Figure Eight**
- **Manual**

Payload Management

Reconnaissance

- **Scene Tracking**
- **Offset Scene Tracking**
- **Target Tracking**
- **Offset Target Tracking**
- **Passive Ranging**
- **Cued Target**

Area Search

Fire Support

Burst Correction

Laser Ranging

Designation

Laser Ranging

Detect Laser Reflected Energy

Compute Azimuth and Elevator Commands

Target Designation

Generic Air-to-Air Missile

Preflight Checkout and Initialization

Ensure that all the elements (sensors, processors, warhead, communcations, electronic counter measures/electronic counter-countermeasures [ECM/ECCM] are in full operating condition, with measured checkout values falling within tolerance limits. This process also involves inserting target acquisition data pertinent to navigation/guidance and to formulating terminal engagement tactics. How navigation/guidance and terminal engagement data are acquired, how and when they are inserted into the missile and the intrinsic quality of the data determine the degree of a missile's smartness.

Navigation and Guidance

This function is performed with respect to self-contained (inertial) reference or with respect to external references such as stars or terrain features; or navigation satellites, ground beacons or ground based navigational networks.

Engagement Tactics and Counter-counter Measures (CCM)

Safing, Arming and Fusing

Missile-borne Sensors

Sensor Fuzed Weapons

Representative Projects:	Proximity Detector Fuzes
	Laser Guided Bombs

Generic Fuze

Built-In-Test

I/O Test

ROM Test

Checksum Operation

Timer Test

RAM Test

Graphics

Trajectory Calculation

Performance and Statistics Calculation

Trajectory Calculation Algorithms

Analog-to-Digital Conversion

Telemetry (discrete and continuous)

Front End Formatting and Filtering

AVIONICS

<u>Aircraft Software</u>
Representative Projects: Multirole Fighter
Trainer Aircraft
Bomber

Generic Multirole Fighter
- **Control and Display Management**
  - <u>Head-Up Display</u>
    - **Project Collimated Symbology**
    - <u>Display Flight Information</u>
    - <u>Display Navigational Steering Info</u>
    - <u>Display Weapon Delivery Info</u>
  - <u>Digital Display Indicators</u>
    - **Store Mgmt Info**
    - **Caution and Advisory Info**
    - **Built-In-Test Info**
    - **Sensor Info**
  - <u>Horizontal Indicator</u>
  - <u>Up-Front Control</u>
    - **Autopilot**
    - **IFF**
    - **TACAN**
    - **Instrument Landing System**
    - **ADF**
    - **Data Link**
- **Status Monitoring**
  - <u>Built-In-Test</u>
    - **Store Fail Indications**
  - <u>Signal Data Recording</u>
    - **Aircraft Fatigue Strain Data**
    - **Engine Parameter Data**
    - **Target Parameter Data**
- **Electronic Flight Controls**
  - <u>Roll and Pitch Control</u>
  - <u>Autopilot</u>
  - <u>Automatic Throttle Control</u>
  - <u>Stall Warning</u>
- **Navigation and Flight Aids**
  - <u>Inertial Navigation</u>
    - **Ground Alignment**
    - **General Navigation Data**
  - <u>Air Data Computer</u>
    - **Provide Pressure Altitude**
    - **Determine Mach Number**
    - **Calculate True Airspeed**
  - <u>Magnetic Azimuth Detector</u>
- **Communication, Radio Navigation, and Identification**
  - <u>Threat Warning</u>
    - **Voice Alerting System**
    - **Audio Alerts**
  - <u>Secure Voice Communcation</u>
  - <u>Interrogation Receipt and Response</u>
  - <u>Airborne Vectoring</u>
  - <u>Bearing and Range Caluculation</u>
- **Stores Management**

AVIONICS

<u>Perform Weapon Inventory</u>
<u>Perform Weapon Status</u>
<u>Perform Weapon Selection</u>
<u>Perform Weapon Release</u>
    Rack/Launcher Lock/Unlock Operation
Electronic Warfare
<u>Detect and Decieve Fire Control and Guidance Radars</u>
<u>Dispense Chaff, Flare and Jammer Payloads</u>
<u>Threat Detection and Suppression</u>
Tactical Sensors
<u>Radar Set</u>
    Target Detection
    Target Designation
    Target Tracking
    Target Navigation
<u>Forwared Looking Infrared Set</u>
    Selectable Fields-Of-View Target Detection/Tracking
<u>Laser Spot Tracker/Strike Camera</u>
    Laser Illuminated Detection
    Laser Illuminated Tracking

<u>Test Equipment Software</u>

Generic ATE

Flightline Test Equipment

Intermediate Shop Test Equipment

Depot Test Equipment

<u>Operating System</u>

<u>Support Software</u>

Compilers/Interpreters

Assemblers

Linkers

Loaders

<u>Control Software</u>

Signal Generators

<u>Unit Under Test (UUT)</u>

Activate Control Software

Measure Response

<u>Training Simulator Software</u>

Generic Aircraft Trainer

Real World Modules

<u>World Model</u>

<u>Vehicle Model</u>

Engine

Weapons Systems

Rotor

Fuselage Characteristics

Cockpit Displays and Instruments

<u>Pilot Symbology</u>

Fuel System

Hydraulic System

Landing Gear System

Communication System

Navigation System

<u>Human Interface</u>

<u>Motion cues to the pilot</u>

<u>Visual cues to the pilot</u>

<u>Aerodynamic forces and equations of motion</u>

AVIONICS

System Control and Environment Modules

Executive Module

Global Database

Device Interface

Processing Units

Memory Units

Communication Links

Evaluation or Data gathering Modules

Data Collection and Reduction

Pilot Evaluation

Video and Digital Data Recording

Test Scenarios

Sampling Rates

Data Storage

Part Task Trainers

Weapon System Trainers

Flight Controls Software

Representative Projects: Primary Avionics Software System (PASS)

Generic Flight Controls

User Interface

System Control

Guidance Navigation and Control

Vehicle System Management

AVIONICS

Vehicle Systems Checkout

Flight Computer Operating System

Timing

Algorithmic

Fault Recovery

Reconnaissance and Electronic Warfare Systems

Generic Reconnaissance and EW

Strategic C^3 Systems
Representative Projects: Force Management Information System
Granite Sentry

Generic Command and Control System
System Monitor and Control
<u>Status Monitoring and Reporting</u>
Error Messages
<u>Performance Montoring and Reporting</u>
<u>Restart and Recovery</u>
Analyst Interaction
<u>Word Processing Interface</u>
<u>Graphics Display</u>
<u>Graphics Interaction</u>
<u>Function Select</u>
<u>Display and Sound Alarms</u>
Message Handling
<u>Message Delivery</u>
<u>Message Accountability</u>
<u>Message Receipt</u>
<u>Message Validation</u>
Communications
<u>Maintain Communication Status</u>
<u>Perform Data Integrity Checks</u>
Security
Test and Exercise
<u>Receive and Process Exercise Messages</u>
<u>Generate and Record Exercise Messages</u>
<u>Message Scenario Generation</u>
<u>Simulate Interceptor Status</u>
<u>Post Exercise Playback and Analysis</u>

Intelligence Systems and C^3 Countermeasures
Representative Projects: SAC On-line Analysis and Retrieval System
Intelligence Data Handling System
Automated Message Handling System

Generic Intelligence System
Message Processing
<u>Communications Handling</u>
TTY Handler
Data Communications Handler
Asynchronous Protocol Handler
Synchornous Protocol Handler
<u>Message Input Processing</u>
Security and Format Identification
Process Contact Messages/Reports
Process Query/Response Message
Process Narrative Message
Message and Security Logging
<u>Message Output Process</u>
Routing Process
Message Formatter
Secure Processing Check
Data Fusion/Correlation

COMMAND, CONTROL AND COMMUNICATONS

- Automatic Correlation
  - **Alert Routing**
  - **Ambiguity Routing**
  - **Dynamic Updates**
  - **Recovery**
  - **Status Logging**
- Data Fusion/Correlation Aids
  - **Review and Evaluation**
  - Ambiguity Processing
  - Display Data
  - Statistical Calculation
  - **Computational Aids**
  - Time/Speed/Distance Calculation
  - Projection and Interception
  - **Intelligence Data Base Management**
  - Sensor Characteristics Data Base Management
  - Platform Technical Characteristics Data Base Mgmt
  - Record Message File Data Base Mgmt
  - History of Coverage Data Base Mgmt
- **Situation Assessment**
- Report Generation
- Criteria File Management
- Analysis Support
- Area Management
- Threat Management
- Trend Analysis
- Knowledge Based Aids
- **Collection Management**
- User Requirements Input and Display
- User Requirements File Maintenance (Store, Retrieve, Modify)
- User Request Review
- Sensor Reference Data Review
- Sensor Selection
  - **Manual**
  - **Automatic**
  - **Tasking Request Generation**
  - **Selection Algorithm Replacement**
- Tasking Status Review
  - **Alert to Revisit Tasking**
- Tasking Message Generation
  - **Manual**
  - **Automatic**
- Report Generation
  - **Preformatted**
  - **Ad-hoc**
- **User-System Interface**
- Workstation Executive
- Batch Scheduling
- Login/Logout Control
- Recovery
- Word Processing Interface
- Map Display
- Graphic Tables
- Graphic Screen Annotation

<u>X-Y Display</u>
<u>Graphic Library</u>
<u>Alphanumeric Display</u>
System Monitor and Control
<u>System Control</u>
- **Master Controller**
- **Slave Controller**
- **Network Controller**
- **Interactive Controller**

<u>Data Processing Control</u>
- **Process Scheduler**
- **Queue Manager**

<u>System Monitor</u>
- **System Logging**
- **System Status and Statistics Display**
- <u>Security Audit Trail Log Reporting</u>
- <u>Status of Communicaton Lines</u>
- **Performance Instrumentation**

<u>Network Communications</u>
- **Message Router**
- **Network Communications Services**
- **Task Authorization**

<u>Distributed File Managment</u>
- **File Transfer**
- **File Print**

<u>Recovery</u>
- **Node Recovery**
- <u>Node Restart/Reconfiguration</u>
- <u>Standalone Workstation Recovery</u>
- **Processing Recovery**
- **Recovery from Process Abort**
- **Operation Recovery**
- **Recovery from Disk Failure**
- **External Communications Link Recovery**
- **Network Communicatons Link Recovery**

System Software
<u>Data Access Utilities</u>
- **DBMS Interface**
- **DBMS Load Utility**
- **Archive**
- **Restore**
- **File Management**
- **Data Access Security**

<u>DBMS</u>
<u>Operating System Software</u>

Tactical C^3 Systems
Representative Projects: JTIDS
Airborne Warning and Control Systems (AWACS)
Airborne Battlefield Command and Control Center (ABCC)
WWMCCS

Generic Tactical C3

COMMAND, CONTROL AND COMMUNICATONS

Surveillence
<u>Detection</u>
<u>Tracking</u>
- **Active and passive tracking**

<u>Identification</u>
- **Identification friend or foe**

Data Display and Control
<u>Weapons assignment and control</u>
<u>Air defense artillery</u>
<u>Air traffic control</u>
<u>Deployment control</u>

Data Processing
<u>Executive</u>
- **Program management**
- **Schedule execution sequence of comp prog components**
- **Failure recovery**
  - <u>Error detection</u>
  - <u>Error analysis</u>
  - <u>System reconfiguration</u>
  - <u>System restart</u>

<u>Surveillence</u>
- **Update azimuth, time, attitude, and navigation ref data**
- **Evaluate radar and IFF reports**
  - <u>Perform association and correlation</u>
  - <u>Display target reports</u>
- **Update flight plans**
- **Identify and monitor tracks**
  - <u>Initialize track data</u>
  - <u>Select the proper sensor data for smoothing</u>
  - <u>Test for target manuevers</u>
  - <u>Smooth sensor information</u>
  - <u>Determine track quality</u>
  - <u>Predict target position, size, velocity</u>

<u>Weapons</u>
<u>Controlled aircraft</u>
- **Executive**
  - <u>Update controlled aircraft parameters</u>
  - <u>Select and organize command information</u>
  - <u>Manage computer communciation pool</u>
- **Tactics and profile select**
  - <u>Select mission profile parameters</u>
  - <u>Monitor mission progress and guidance computations</u>
- **Guidance**
  - <u>Solve intercept equations</u>

<u>Air defense artillery</u>
- **Exchange track and command information with Army Air Defense Command posts.**

<u>Communications</u>
- **TADIL-A interupt processing**
  - <u>Interrupt processing of TADIL-A</u>
  - <u>Perform net controller functions</u>
  - <u>Schedules transmission of TADIL-A messages</u>
- **Encoder**
  - <u>Encode track, summary, and operational status data</u>

<u>Transfer to the output buffer</u>
<u>Sechedule messages for transmission</u>
<u>Prepare TADIL-C messages</u>
<u>Control time of TADIL-C message transmission</u>
Decoder
<u>Update database</u>

<u>Displays</u>
Display executive
<u>Display intialization</u>
<u>Determine display capacity alerts, alarms, console assignments, and presentation</u>
<u>Schedule sitatuation and tabular display generators</u>
Sensor display
<u>Convert sensor reports and strobes into symbols</u>
<u>Update the sensor history buffer</u>
<u>Format data for display</u>
Situation display
<u>Transform system data into symbols, alphanumerics, coordinates, lines and circles</u>
<u>Address display data to consoles</u>
Tabular display
<u>Update and generate tabular displays</u>
Interrogate data base
Extract data
Process and format data

<u>Switch Actions</u>
Presentation line generation
Processing control
Switch action processing
Switch action implementation

<u>Internal Simulations</u>
Message generation
Exercise tape processing
Flight simulation
<u>Radar and IFF reports</u>
<u>Positional updating</u>
<u>Radar ECM activity</u>
<u>Automatic umpiring</u>

<u>Battle Staff</u>
Mission monitoring
<u>Process data from the command/battle staff data base</u>
Alarms and monitoring
<u>Check of anomalies in mission profiles</u>
Late arrivals
Flight path deviations
Data base management
<u>Store data</u>
<u>Reallocate drum space</u>
<u>Retrieve data</u>
Special application processing

Communications

<u>System Maintenace Control</u>

Communications

<u>External and internal communications</u>
Secure voice and data transmission/reception

Console-to-console communications
Flightcrew intercom
Maintenance intercom
UHF, VHF/AM and TDMA relay
TADIL-C
Navigation and Guidance
Command
The command function includes both onboard command functions and those assigned by higher authority relative to external elements.
Supervision of internal operations
Resource management
Threat assessment
Air defense battle progress assessment
Monitor/evaluate offensive air operations
Direct E-3A aircraft
Report to external elements
Plan air operations
Support
Data recording
Simulation and excercise
Advanced Decision Systems
Representative Projects: MAC Command & Control Information Processing System

Generic Advanced Decision System
Warning
Situation Monitoring
Evaluation
Situation Assessment
Decision-Making and Support
Force Management
Force and Resource Monitoring
Operations Planning and Scheduling
Force Employment Monitoring
Order and Mission Instructions
Battle Management/Reconstitution
Employment and Execution of Forces (Initial & Reconstituted)
Hostilities Termination

<u>Satellite Communications and Ground Control</u>
Representative Projects: Air Force Satellite Control Network
Consolidated Space Operations
Center
NAVSTAR Global Positioning
System
Satellite Early Warning System

Generic Satellite
Command and Control
<u>Fault Detection and Correction</u>
Determine State-of-Health Status
Determine Payload Status
<u>Command Decoding and Generation</u>
Receive level and timing interrupts
Preprocess data as input
Perform intermediate processing
<u>Sequencing</u>
<u>Command decoding</u>
<u>Error recovery</u>
Generate commands for subsystems and telemetry as output
<u>Telementry Modulator/Demodulation</u>
Command detection
Telemetry modulation
Transmit
Receive
Perform Antenna Pointing
<u>Record Flight Data</u>
Attitude Control
<u>Orbit Adjust/Thruster Control</u>
<u>Sense orientation</u>
<u>Maintain stability</u>
Propulsion
Power
<u>Distribution and generation</u>
<u>Monitor and control battery charge</u>
<u>Automatic load shedding</u>
Data Communications
<u>Execute Routing and Congestion Control</u>
<u>Perform Link-Level Control</u>
Establish Link
Perform Link Access Control
<u>Accommodate variable length packets</u>
<u>Prioritize demand assignment</u>
<u>Multiplex link control data</u>
Perform Link Error Control
<u>Acknowledge/Retransmission Error Protocol</u>
Link Termination
<u>Priority Preemption</u>
<u>Minimum Link Quality Threshold</u>
<u>Radio Range Time</u>
Link Synchronization
<u>Clock Synchronization</u>

<u>Time Stamping</u>
Link Quality Monitoring
<u>Measure Signal to Noise Ratio</u>
<u>Bit and Packet Error Rate Measurement</u>
<u>Link Flow Control</u>
<u>Link Addressing</u>
<u>Perform Encryption</u>
<u>Perform Packet Transmission/Reception</u>
Demodulate and Decode Packet
Perform Baseband Store and Forward Processing
Inspect Packet Address
Apply Routing Rules

Generic Ground Control
Telemetry
Tracking
<u>Receive ranging data</u>
<u>Obtain antenna azimuth and elevation data</u>
<u>Transmit range tones</u>
Determine phase shift
Calculate slant range
Command

<u>**Space Vehicle Systems**</u>
Representive Projects: Space Shuttle

Generic Space Transportation System
Primary Mission Sensors
<u>Vehicle Acquisition and Tracking</u>
<u>Event Detection</u>
<u>Earth Observations</u>

<u>**Launch Vehicle Systems**</u>
Representative Projects: Atlas
Titan
Payload Assist Module
Inertial Upper Stage

Generic Launch Vehicle
Prelaunch Checkout and Countdown
Telemetry and Tracking
<u>Critical Performance Data</u>
<u>Receive/Decode Destruct Information</u>
Guidance
<u>Receive Target and Trajectory Specifications</u>
<u>Compute Required Velocity Vector</u>
<u>Send Steering Commands</u>
<u>Send Thrust Termination Signal</u>
Navigation
<u>Inertial Navigation</u>
Strap-down Inertial
<u>Detect Rotation</u>
<u>Compute Total Acceleration</u>
<u>Compute Velocity</u>
<u>Compute Position</u>

<u>Output to Guidance Computer</u>
Gimble-Angle Readout
<u>Detect Rotation</u>
<u>Compute Total Acceleration</u>
<u>Compute Velocity</u>
<u>Compute Position</u>
<u>Output to Display Device</u>
Control
<u>Launch Trajectory Control</u>
Guidance
<u>Compute Position, Velocity and Acceleration</u>
Thrust Termination
Vehicle Inspection
Vehicle Control
<u>Ignition Control</u>
<u>Orbital Flight Control</u>
Spin Stabilization
Attitude Control
Altitude Control
Programmed/Velocity
Inertial
Rendezvous
<u>Passive</u>
<u>Active</u>
Re-Entry
<u>Programmed/Velocity Cutoff</u>
<u>Inertial</u>
<u>Radio</u>
Transmit
Receive
Measure Signal Phase Shifts
Measure Signal Time Delay
Data Management
<u>Data Processing</u>
<u>Data Storage</u>
<u>Data Transmission</u>
<u>Signal Processing</u>
Communications
Propulsion

<u>Defense and Surveillance Systems</u>
Representative Projects: SDI

Generic Surveillence System
Surveillance, Acquisition and Track
<u>Detect Object</u>
Filter Out Background
<u>Receive Image or Signal</u>
<u>Convert to Coordinates</u>
<u>Compare with Background</u>
Filter Sensor Data
<u>Separate by Sensor Type</u>
<u>Determine Observation Type</u>
<u>Construct Object Type Entry</u>

- <u>Store Object Type Entry</u>
- **Associate Observation with Track**
- <u>Correlate Data to Track</u>
- <u>Update Track Entry</u>
- **Correlate Unknown Observations**
- <u>Correlate Object to Object</u>
- <u>Prioritize Sensor for Observation Data</u>
- <u>Compose New Track Entry</u>
- **Determine Initial Trajectory**

<u>Track Object</u>

- **Improve Trajectory**
- **Refine Trajectory**
- **Compare with Other Tracks**
- <u>Compare Track Location</u>
- <u>Update Missile Track</u>
- **Predict Trajectory**
- <u>Generate Observation Window</u>
- <u>Determine Impact Point</u>
- <u>Determine Sensor Coverage</u>
- **Task Sensor**
- <u>Determine Data Required</u>
- <u>Determine Sensor</u>
- <u>Generate Sensor Search Commands</u>
- **Receive Handover**

<u>Identify Object</u>

- **Convert to System Units**
- **Compare with Known Data**
- **Compare with Previous Data**
- **Request Object Tracking**
- **Task for Discrimination**
- <u>Determine Discrimination Method</u>
- <u>Determine Discriminator and Sensor</u>
- <u>Generate Discriminator Commands</u>

<u>Assess Objects for General Threat</u>

- **Determine Type of General Threat**
- **Prepare Target Data**
- **Prepare Resource Threat**