

2

DTIC FILE COPY

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

FLEET NUMERICAL OCEANOGRAPHY CENTER
SOFTWARE DEVELOPMENT STANDARDS: AN
IMPLEMENTATION OF DOD-STD-2167A

by

William T. Livings

September 1989

Thesis Advisor:

Barry A. Frew

Approved for public release; distribution is unlimited

AD-A219 606

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No 0704 0188	
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S)			5 MONITORING ORGANIZATION REPORT NUMBER(S)		
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b OFFICE SYMBOL (if applicable) Code 54		7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000			7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		
8a NAME OF FUNDING/SPONSORING ORGANIZATION		8b OFFICE SYMBOL (if applicable)		9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c ADDRESS (City, State, and ZIP Code)			10 SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO	PROJECT NO	TASK NO
			WORK UNIT ACCESSION NO		
11 TITLE (Include Security Classification) FLEET NUMERICAL OCEANOGRAPHY CENTER SOFTWARE DEVELOPMENT STANDARDS: AN IMPLEMENTATION OF DOD-STD-2167A					
12 PERSONAL AUTHOR(S) Livings, William T.					
13a TYPE OF REPORT Master's Thesis		13b TIME COVERED FROM _____ TO _____		14 DATE OF REPORT (Year, Month, Day) 1989, September	
15 PAGE COUNT 125					
16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP			
			Software Engineering;		
			Software Development Standards		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) Software development standards are integral to any organization's software development efforts and are essential to the development life cycle. They are vital in ensuring on-time delivery of more reliable and maintainable software products. The trend in software development is toward a structured, systems engineering approach based on standard practices, methodologies and rigorous management control. DOD-STD-2167A establishes uniform requirements for software development that are applicable throughout the system life cycle. It provides a basis for government insight into a contractor's software development, testing, and evaluation efforts. This thesis examines the possibility of developing a generic, tailored version of DOD-STD-2167A that would apply to an activity's or general project category's software development needs. The analysis indicates that a tailored version of the standard can be developed to at least eliminate					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21 ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Barry A. Frow			22b TELEPHONE (Include Area Code) (408) 646-2924		22c OFFICE SYMBOL Code 54EW

DD Form 1473, JUN 86

Previous editions are obsolete

S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

#19 - ABSTRACT - (CONTINUED)

some requirements for a project manager when dealing with an activity's or project category's software requirements, thereby reducing superfluous and duplicative activities.

Approved for public release; distribution is unlimited

Fleet Numerical Oceanography Center Software Development
Standards: An Implementation of DOD-STD-2167A

by

William T. Livings
Lieutenant, United States Navy
B.S., Louisiana State University, 1978

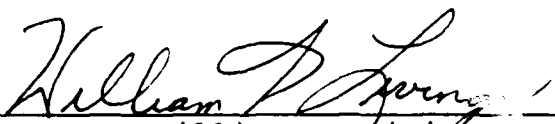
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

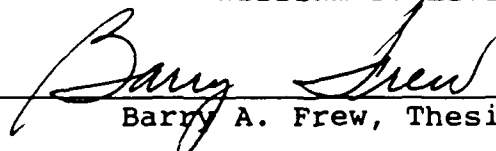
from the

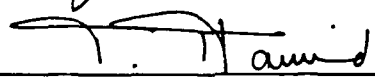
NAVAL POSTGRADUATE SCHOOL
September 1989

Author:


William T. Livings

Approved by:


Barry A. Frew, Thesis Advisor


Tarek Abdel-Hamid, Second Reader


David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

Software development standards are integral to any organization's software development efforts and are essential to the development life cycle. They are vital in ensuring on-time delivery of more reliable and maintainable software products. The trend in software development is toward a structured, systems engineering approach based on standard practices, methodologies and rigorous management control. DOD-STD-2167A establishes uniform requirements for software development that are applicable throughout the system life cycle. It provides a basis for government insight into a contractor's software development, testing, and evaluation efforts. This thesis examines the possibility of developing a generic, tailored version of DOD-STD-2167A that would apply to an activity's or general project category's software development needs. The analysis indicates that a tailored version of the standard can be developed to at least eliminate some requirements for a project manager when dealing with an activity's or project category's software requirements, thereby reducing superfluous and duplicative activities.

TABLE OF CONTENTS

I.	INTRODUCTION -----	1
A.	STANDARDS AND THE DEVELOPMENT PROCESS -----	1
B.	OVERVIEW OF THE COMMAND -----	5
C.	SCOPE OF RESEARCH -----	8
II.	TAILORING PROCESS/PROCEDURES -----	10
A.	INFORMATION SYSTEMS/SOFTWARE REQUIREMENTS ---	11
B.	DOD-STD-2167A -----	13
C.	METHODOLOGY -----	15
D.	REFERENCE DOCUMENTS: RELATIONSHIP OF DOD-STD-2167A TO OTHER STANDARDS -----	22
III.	SUMMARY/RECOMMENDATIONS -----	39
APPENDIX A:	MAJOR PROJECTS -----	44
APPENDIX B:	INTERMEDIATE PROJECTS -----	73
APPENDIX C:	MINOR PROJECTS -----	75
APPENDIX D:	SOFTWARE MAINTENANCE -----	102
APPENDIX E:	SOFTWARE QUALITY ASSURANCE -----	108
APPENDIX F:	SOFTWARE METHODOLOGY: PROTOTYPING -----	112
LIST OF REFERENCES	-----	117
INITIAL DISTRIBUTION LIST	-----	119

I. INTRODUCTION

A. STANDARDS AND THE DEVELOPMENT PROCESS

Software is one of the fastest growing and most expensive elements of information systems today. As computer hardware becomes more powerful and less costly to acquire, software development costs escalate. It becomes even more costly to maintain or modify, especially if it wasn't developed properly or didn't fulfill the user's requirements. In fact, software development has become the major, present day constraint or "bottleneck" to the timely delivery of effective, useful information systems.

Building adequate computer-based systems is a significant challenge, and software is often a primary stumbling block. People create programs that don't function properly; programs that do work, but don't do what the users really want; programs that can't be easily changed or corrected when errors are found; and programs that are delivered for use months or even years too late. (Pressman, 1988, pp. 1-2) Software continues to absorb a larger and larger percentage of the overall development cost for computer-based systems. People/organizations spend billions of dollars each year on the development, acquisition, and maintenance of computer software. Pentagon expenditures for mission critical software totaled \$11 billion in 1985 and it is

predicted that by 1990 the amount will more than double accounting for roughly 20% of everything the Pentagon spends on weapons (Newport, 1986, p. 133). It is also reported that industry-wide, "75% of the time, businesses never use the software programs they undertake, either because they never complete them or because they arrive too late." (Newport, 1986, p. 132)

All of this alludes to a "software crisis" in the development of today's computer software and indicates that the traditional method of developing software is not working. Problems are not limited to software that "does not function properly." Rather, the "software crisis" encompasses problems associated with how we develop software, how we maintain a growing volume of existing software, and how we can expect to keep pace with a growing demand for more software (Pressman, 1988, p. 10).

There is no single best approach to solve the software crisis. However, by combining comprehensive methods for all phases in software development, better tools for automating these methods, more powerful building blocks for software implementation, better techniques for software quality assurance, and an overriding philosophy for coordination, control, and management, we can achieve a discipline for software development--a discipline called software engineering. (Pressman, 1988, pp. 11-12) In fact, the current trend in software development is transforming the

Development Life Cycle from a seemingly haphazard, trial and error process into a discipline based on standard practices, methodologies, and rigorous management control (Newport, 1986, p. 135).

Integral to this new way of thinking has been the development and establishment of software development standards. Software standards are vital in ensuring on-time delivery of quality software products and in minimizing maintenance costs. Standards are useful and equally important in all phases of the Software Development Life Cycle from requirements analysis and design, through coding and testing, and even into the implementation and maintenance phases. Standards must not only be developed, they must also be adhered to and their use enforced if they are to prove beneficial to your software development efforts. Focusing on standards to improve development efforts leads to reduced maintenance problems as the members of the development team are now following the same standardized set of development practices and techniques, thereby reducing the number of individual perceptions and misunderstandings of users' requirements and how they are to be achieved. The overall quality of software products can also be improved by standardizing the practices of programmers during the entire Life Cycle of the product. It is even reasonable to assume that consistent documented terminology and project standards improve communication among team members and result in fewer

interpretations (Poston, 1984, p. 96). Standards serve as a "written, usable formalization of experience--successful experience. Their use overcomes a common problem: most project experience is lost, or at best handed down by word of mouth or individual behavior." (Braverman, 1979, p. 81) Thus, standards "reduce the vulnerability of the project to personnel turnover and time lost getting new personnel up to speed." (Peters, 1981, p. 103)

Overall, benefits accrue by adopting and enforcing software development standards. The goals of standards are many--good schedule and cost performance, high product reliability, adequate documentation, increased productivity, smooth development and delivery, higher quality software, machine independence, more productive work force, and reduced production and maintenance costs (Tausworthe, 1978).

All of this evidence indicates that in the final evaluation, standards, if used, can contribute to better quality software. Standards "can be used to ensure that each and every module in a system, and the decisions which lead to this configuration are all established and documented at central checkpoints during the design effort." (Peters, 1981, p. 103) As a result, software both meets the requirements of the user, and is maintainable throughout its lifetime. In the long run it offsets an initial investment required to establish software development standards.

B. OVERVIEW OF THE COMMAND

The Fleet Numerical Oceanography Center (FNOC) in Monterey, California is the U.S. Navy's primary meteorological and oceanographic analysis and forecast facility providing unique global environmental information to the entire Naval establishment and to approved civilian/commercial users. FNOC's mission is "to provide operational numerical oceanographic and atmospheric products peculiar to the needs of the Department of the Navy; and to develop and test numerical techniques to solve oceanographic analytical and forecasting problems as directed by the Commander Naval Oceanographic Command." (FNOC 1986 Master Plan, 1986, p. 4)

FNOC provides worldwide environmental information support by monitoring a computerized forecasting process to ensure timely production of high quality products. Conventional and satellite environmental data from around the world is received through land-line and satellite data links. These raw data are automatically sorted, edited, and analyzed to represent current atmospheric and oceanic conditions. This analysis provides initial conditions from which numerical forecasts of the future state of the atmosphere and ocean evolve. A series of advanced atmospheric, ocean, and air-ocean prediction models generate a four-dimensional measure of the future air-ocean environment, typically, up to three days in advance. These forecast products are used to produce a variety of performance assessments for specific

tactical weapon systems and platforms. In addition, environmental analyses and forecasts are transmitted on high-speed computer links to Navy oceanography centers and detachments around the world to satisfy user requirements. (FNOC Change of Command Brochure, 1989, p. 2)

FNOC operations revolve around a large, intricate and growing computer and communications system. Five departments provide support to the command and ensure the successful accomplishment of its day-to-day support mission. The Computer Systems Department is the largest department, responsible for operating and maintaining the larger main-frame and super computers in the command. These include the Primary Environmental Prediction System (PEPS), the PEPS Upgrade (PEPSU), the Hydro Climatology Computer (HCC), and the Satellite Processing Center (SPC). The Data Integration Department is responsible for receipt and processing of raw environmental data, implementation, and maintenance of advanced meteorological and oceanographic analysis and forecast models, archiving of raw data and analyzed fields, quality control of all environmental analysis and forecast products, and management of FNOC's Environmental Satellite Data Processing Program (FNOC Change of Command Brochure, 1989, p. 4). The Fleet Applications Department meets the requirements of operating forces for specialized environmental information by creating a variety of tailored "end-user" products from analyses and forecasts of the air-ocean

environment, and includes such systems as the Optimum Track Ship Routing (OTSR) System and the Optimum Path Aircraft Routing System (OPARS). The Field Support Department provides engineering and logistics support to the diverse communications and display systems required by FNOC and other Naval oceanography command activities and includes the Naval Environmental Data Network (NEDN) and the Naval Environmental Display Station (NEDS). The Supply and Fiscal Department manages FNOC's budget and performs all other supply and financial related functions pertaining to the command.

Needless to say, this is a complex, intricate and expensive operation. FNOC is the Navy's only operational atmospheric and oceanographic forecasting center with advanced data processing and communications systems. The principal processing task at FNOC is the running of global environmental prediction models. This is accomplished with the primary environmental processing systems, PEPS and PEPSU. The PEPS system is dedicated to global analysis, global prediction and tailored product output to support Naval operations. Therefore, this system must be extremely responsive to fulfill all of the users' needs and requirements. However, much of the FNOC software running on the PEPS mainframes is very old and poorly documented.

FNOC's evolution of software has created an inventory of operational computer programs throughout the command which

is heavily machine and operating-system dependent and difficult to maintain. FNOC's method for addressing this problem is to use a special project, the Software Improvement Program (SIP). The SIP uses an incremental approach to the modernization of existing software. SIP maximizes the value of past software investments and at the same time, increases the reliability, efficiency, portability, and maintainability of the software. Software modifications range from simple translation of code to complete systems re-engineering. Whenever possible, existing code is salvaged and improvements are retrofitted. (FNOC SIP Macroplan, 1988, p. 3) The primary goal of the SIP is

to enable FNOC to more effectively carry out its mission by making its software more maintainable, portable, and reliable, thereby extending its useful life. A closely related goal is to make software more hardware--and operating-system--independent, thereby reducing the costs and difficulty of software conversions caused by competitive hardware acquisitions. (FNOC SIP Macroplan, 1989, p. 3)

The FNOC SIP will establish a systems development and maintenance environment which will help improve the overall quality of future software developed for, or maintained by, FNOC.

C. SCOPE OF RESEARCH

The scope of this thesis is to ascertain the feasibility and usefulness of tailoring the DOD-STD-2167A to meet Fleet Numerical Oceanography Center software development needs. Research will explore the options of developing a single

tailored version of the standard as opposed to multiple versions, according to some generic categorization of FNOC software development projects, to meet FNOC software needs. Initially, this research involved a review of the general need and importance of standards on software development followed by a determination of FNOC missions, functions and objectives, and the special importance that good, structured software development has on the successful accomplishment of its missions and objectives. Following this is a review and analysis of the DOD-STD-2167A and how it relates to FNOC software development. Tailoring of the DOD standard involved tailoring of the standard's requirements, data item descriptions, and any formal reviews and audits deemed appropriate for FNOC software needs and requirements. The resulting tailored versions of the standard are included in this thesis as appendices. Other appendices will address the areas of software maintenance and quality assurance, their relationship (or lack thereof) to the standard and their relative importance to software development in general. The last appendix provides general guidelines or standards to follow when employing prototyping as a part of the Life Cycle Methodology. DOD-STD-2167A's relation to other military standards and specifications are addressed in the body of the thesis.

II. TAILORING PROCESS/PROCEDURES

FNOC intends to use the tailored version(s) developed from this research effort as a starting point, or reference point for a much broader, more detailed command-wide instruction or set of instructions covering all phases of the Software Development Life Cycle. It will serve as an "umbrella" under which detailed sub-sets of instructions/methodologies will be placed. The SIP Software Engineering Technology is the approach FNOC will take to improve or reengineer their software. Its methodology consists of selected structured analysis, design, programming, and testing techniques and procedures; a set of supporting software tools; a set of standards and guidelines; a quality assurance and configuration management plan; and a training program designed to facilitate the implementation of the methodology. A baseline of software engineering elements will be developed, focusing initially on the SIP but with the ultimate goal of implementing those elements command-wide. This baseline may change as elements are used and evaluated as the SIP matures. (FNOC SIP Macroplan, 1988, p. 13) For example, DeMarco's methodology for structured analysis, Yourdon's methodology for design, and Chen's methodology for database/modeling will be used, supported with case tools, in their respective phases of the Life

Cycle. These methodologies will be refined to meet local needs, standardized and documented, and placed under the "umbrella."

A. INFORMATION SYSTEMS/SOFTWARE REQUIREMENTS

Applications software at FNOC performs a number of specific functions, but in general terms it can be divided into five functional categories:

1. Software that manipulates input/raw data.
2. Software that analyzes and prepares data for models (data assimilation).
3. Software that models data (algorithmic calculations and forecasts).
4. Software that takes output from models and builds products.
5. Software that transports products (communication software).

FNOC software development is done primarily by contractors and Navy R&D laboratories, although recently FNOC has started to undertake in-house development of some of their software. Most of the in-house efforts enhance and maintain software and convert deliverables to a form which can be implemented operationally. Because of the complexity of some software, it is being maintained by the original programmer even though that individual may have been transferred to another department. Other reasons some of the software is difficult to maintain include:

1. Use of machine-specific assembly languages that few individuals at the command know.

2. A workload that does not allow programmers time to cross-train.
3. Lack of local software engineering standards.
4. A large number of undocumented software modifications applied over time.

Frequently, after delivery of a product, in-house conversion is necessary to create operational software due to inadequately written specifications or because different standards and procedures were used during the development process than are being used at FNOC. This conversion is also necessary to move the software into the operational run environment. (FNOC SIP Macroplan, 1988, p. 29)

Existing standards at FNOC primarily deal with coding and documentation in the PEPS environment. DOD Automated Information Systems (AIS) documentation standards (DOD-STD-7935A) is presently being used, but no other standards are being rigorously followed for directing the Development Cycle. Software Quality Assurance is performed almost entirely by the programmers of the software. Requests for new products or changes to existing products are handled in a variety of ways, from formal requests to simple phone calls from users. Changes to software are frequently made on the initiative of the responsible programmer, who uncovers problems or recommends improvements based on monitoring of operational products. (FNOC SIP Macroplan, 1988, p. 30) Obviously, one can see the need for standards not only in the area of full scale software development, but

also in the area of modifying or enhancing existing software. The SIP will be the vehicle for establishing a modern and effective systems development and maintenance methodology. Not only will this standard methodology be used at FNOC and on PEPS software, but eventually it will be used for other Information Systems (ISS) and at the Research and Development Activities that support FNOC. This will help prevent the existing software problem from recurring again in the future. (FNOC SIP Macroplan, 1988, p. 7)

B. DOD-STD-2167A

An integral part of the SIP will be the development/establishment and enforcement of standards. DOD-STD-2167A establishes uniform requirements for software development that are applicable throughout the System Life Cycle. It provides the means for establishing, evaluating, and maintaining quality in software and associated documentation. (DOD-STD-2167A, 1988, pp. iii/iv) The standard provides a broad framework for software development, leaving application specific details to the discretion of the user or program manager. It provides specific milestones or deliverables identified for each development phase in the form of its Data Item Descriptions (DIDs). DIDs describe a set of documents for recording the information required by DOD-STD-2167A.

SECNAVINST 5000.1B (1983) mandates the use of DOD-STD-2167A in the development of mission-critical computer

software by contractors or government agencies. The general framework of the standard can be tailored to any software project, thereby providing a formalization of the development process for both mission-critical and non-mission critical computer system software. The standard contains a set of requirements designed to be tailored for each contract by the contracting agency (DOD-STD-2167A, 1988, pp. 1/2). Per Department of Defense Directive (DODD) 5000.43, Acquisition Streamlining, "this standard must be appropriately tailored by the program manager to ensure that only cost-effective requirements are cited in defense solicitations and contracts." (DOD-STD-2167A, 1988, pp. iii/iv)

The tailoring handbook states "DOD-STD-2167A contains requirements for the acquisition, development, and support of software systems. For contracting agencies to effectively apply the standard, it is important that they tailor the requirements of the standard to meet the objectives of the specific system." (MIL-HDBK-287, 1989, Foreword) DOD-STD-2167A and its DIDs provide a maximum set of requirements. They are specifically designed to be tailored down to include only those requirements that are needed for a given project. The tailoring handbook also states, "tailoring must be performed for each acquisition, development, or support contract issued during the system acquisition process. As objectives and tasking change during that

process, tailoring decisions for each contract will change accordingly." (MIL-HDBK-287, 1989, p. 17)

Tailoring the DOD-STD-2167A from the "big picture" perspective is a two-step process. One must first tailor the requirements of the standard itself. These tailoring decisions form and are specified in the contract's statement of work (SOW). Secondly, DOD-STD-2167A is the parent standard for 17 data item descriptions (DIDs). Each DID specifies the required content for a particular deliverable document. Selection of appropriate DIDs and deletion of inappropriate content within those DIDs is a key part of the tailoring process (MIL-HDBK-287, 1989, p. 10). Tailoring of DIDs is specified on the Contract Data Requirements List (CDRL) form. Each deliverable is specified in a CDRL item.

It should also be noted that DOD-STD-2167A contains a number of selected requirements that are considered self-tailoring. They are referred to as "shell requirements." These are requirements whose specifics or tasks are not required unless specific direction is included in the contract. These paragraphs are discussed in paragraph 4.2.7 of the tailoring handbook. (MIL-HDBK-287, 1989, p. 10) Figure two of MIL-HDBK-287 presents these requirements.

C. METHODOLOGY

Although DOD-STD-2167A was designed with the intent to be tailored to the given requirements of each individual project or contract, a major premise of this research effort

was to determine if some generic version of the standard could be developed that would apply to an activity's or general project category's software development projects. That is to say, some skeletonized version of the standard that would automatically eliminate and/or add requirements for a project manager when dealing with a particular activity's or project category's development requirements. Jane Radatz from Logicon, Inc., principal author (in conjunction with the DOD) of the DOD-STD-2167A and developer of the automated tailoring software package "Tailor," stated that the intent and design of DOD-STD-2167A was to tailor it to meet the individual needs and requirements of a specific system, development project or contract. She further commented that to apply generalized "tailored" versions of the standard to meet the development needs of categories of software would be difficult and only possible if the projects within a category were so similar in nature so as to produce the same set of deliverables. Even then further refinement of the "tailored version" would probably be necessary for each specific project. (Telephone conversation between Jane Radatz, Logicon, Inc., and the author, 17 May 1989)

David Maibor, principal author of DOD-STD-2167, also agreed, and felt that tailoring of the DOD-STD-2167A was mostly dependent on the personal biases of the project manager, the number of Computer Software Configuration Items

(CSCIs) in the project, and the level to which he/she may decide to decompose and test the Computer Software Components (CSCs) and Computer Software Units (CSUs). A tailored version for a category of software projects would be difficult to establish and still might not be useful in every case. (Telephone conversation between David Maibor and the author, 17 May 1989)

Several FNOC project managers and software personnel were interviewed and asked to respond to the 27 high-level, system project description questions in the "Tailor" software package. For the most part, responses were varied. In some cases, respondents were unfamiliar with certain aspects of DOD-STD-2167A and its terminology/acronyms, and therefore, did not know how to respond to some of the questions. In other cases, they would attach assumptions or constraints to their responses and on many occasions would respond with the comment "project dependent." When questioned, ALL respondents were of the consensus that to categorize FNOC software into "general" project categories, at least with regard to the DOD-STD-2167A, would be more practical and useful than trying to apply a tailored version of the standard to meet all FNOC software needs.

As a result, tailoring the standard according to the general project categories of Major, Intermediate, and Minor, as espoused by Barry Boehm, was determined to be the most applicable for FNOC's needs. Other categories were

considered, but were determined to be either too restrictive in scope or too dependent on one overriding characteristic. For instance, the project categories of Small, Medium, and Large denoted too much of a dependency on project size. Categorizing projects according to size, because of this dependency, may not always be appropriate. There may be a situation where a "Small" project's single product may be so complex and interdependent that it is more appropriate to tailor as a Large project. On the other hand, a "Large" project could result in a number of products that are relatively independent of one another and therefore does not require the same rigorously managed set of requirements and deliverables as some other project of the same category. Dividing projects into categories according to complexity also lent itself to a similar set of problems, restrictions, and variables. Dividing software development into categories of System Software and Application Software was considered, but was also felt to lead to unreliable results. There may be situations where project requirements for an application software development effort more closely resemble those of a System Software project (or vice versa), maybe because of scope, the number and types of products, and/or operating environment. As a result, a project may follow a version of the standard more applicable to the other category.

It was felt that Boehm's categories of Major, Intermediate, and Minor were general enough to afford FNOC the flexibility and latitude to apply tailored versions of the standard. This way FNOC personnel maintain a relatively comfortable level of assurance that the standard was appropriately being applied to the majority of the projects in each category. In addition, the command would still be receiving the benefits one should from the tailoring process as it was intended. That is to say, FNOC would be deleting non-applicable and/or redundant, superfluous requirements and activities in an effort to ensure cost-effective development efforts and contracts.

1. Criteria for Categorizing

a. General Guidance

- 1) The category chosen should be the highest one with any applicable criterion.
- 2) Lines of code are undocumented source code including commonly used subroutines.
- 3) When in doubt, start with the "Major" category and work down, deleting requirements as applicable.

b. Major Category

- 1) Greater than 20,000 lines of code.
- 2) Language other than standard FORTRAN.
- 3) Significant interfaces to other systems.
- 4) Significant risk or security implications.
- 5) Significant data base usage
- 6) Multiple users/many functional areas of the software.

- 7) Complex requirements including science, communications, system software, and/or complex data manipulation.
 - 8) May, on occasion, require additional hardware/firmware.
- c. Intermediate Category--Criteria for an "Intermediate category, by sheer definition, should fit somewhere between that for "Major" and "Minor" categories. Applicable criteria for this category could be:
- 1) Greater than 2000 and less than 20,000 lines of code.
 - 2) High-level language that is familiar to the programmer and requires a low learning curve for programming proficiency.
 - 3) Few interfaces that are well understood.
 - 4) Few and well understood security requirements.
 - 5) Multiple users/same functional area of the software.
 - 6) May require well understood modifications to hardware/firmware.

However, much difficulty was experienced in utilizing this criterion with "Tailor" to produce an applicable set of required deliverables (DIDs). A discussion pertaining to this category of software projects and its respective version of the standard can be found in Appendix B.

d. Minor Category

- 1) Less than 2000 lines of code.
- 2) Standard FORTRAN or comparable language with minimal-to-no extensions required.
- 3) No interfaces to other systems (stand alone).
- 4) No significant data base usage.
- 5) No significant risk involved including no security implications.
- 6) One single user of the software.

7) One person assigned to the job.

8) Involves no hardware or firmware changes.

Note: Some of the individual criterion values, within their respective categories, may be different than those used by Barry Boehm as FNOC is establishing and/or using values to meet organizational needs and standards.

Subsequent to the determination of appropriate categories and their respective criteria, several sessions between the author and the FNOC Software Coordinator/ADP Manager were held to tailor the standard to each of the corresponding categories. It was felt that Logicon's Automated Tailoring Software Package was appropriate and adequate enough to tailor the requirements of the standard to such general categories. (Manual tailoring of the applicable DIDs, reviews, and audits for each category still needed to be performed.) Project description answers, tailor selections, statement of work reports, detail status reports, and action item lists were produced and placed in Appendices A, B, and C for Major, Intermediate, and Minor projects respectively. Appropriately selected DIDs, reviews and audits, depending on the responses given while tailoring the standard, were tailored manually and also placed in corresponding appendices.

D. REFERENCE DOCUMENTS: RELATIONSHIP OF DOD-STD-2167A TO OTHER STANDARDS

DOD-STD-2167A is part of an overall family of defense system acquisition standards. In fact, DOD-STD-2167A invokes or imposes other standards. These standards are:

1. DOD-STD-480A: Configuration Control--engineering changes, deviations, and waivers, dated April 1978.
2. MIL-STD-481: Configuration Control--engineering changes, deviations, and waivers (short form), dated 18 October 1972.
3. MIL-STD-490A: Specification practices, dated 4 June 1985.
4. MIL-STD-499A: Engineering Management, dated 1 May 1974.
5. MIL-STD-1521: Technical Reviews and Audits for systems, equipments, and computer software, dated 4 June 1985.

This referencing of other standards is permitted by Department of Defense Directive (DODD) 5000.43. Unless the invoking requirements (paragraphs) are tailored out of DOD-STD-2167A, the referenced standards are automatically on contract to the extent specified in DOD-STD-2167A. If no additional requirements from these standards are to be imposed, the standards need not be called out separately in the contract. However, when references to other standards are left in DOD-STD-2167A, the referenced standards should be reviewed. All incompatibilities must be resolved to have a correct, consistent Statement of Work (SOW) (MIL-HDBK-287, 1989, pp. 9-10). In the event of a conflict between the

text of DOD-STD-2167A and a reference, the text of DOD-STD-2167A shall take precedence (DOD-STD-2167A, 1988, pp. 3/4). Institute of Electrical and Electronic Engineers (IEEE) standards were also reviewed in the areas of Configuration Management, Software Maintenance, and Quality Assurance. A discussion of subsequent findings and relationships of these standards to the DOD-STD-2167A and software development, in general, can be found in appropriate sections and/or appendices of this thesis.

1. DOD-STD-480A and MIL-STD-481A

DOD-STD-2167A paragraph 4.5.5, General Requirements, requires the contractor or developing agency to prepare Engineering Change Proposals, or an abbreviated form thereof (MIL-STD-481A), in accordance with DOD-STD-480A, if and as specified in the contract. Both DOD-STD-480A and MIL-STD-481A delineate configuration control requirements and provide instructions for preparing and submitting proposed engineering changes, deviations and waivers, and related information. An engineering change is an alteration of a configuration item or item, delivered, to be delivered, or under development, after formal establishment of its configuration identification. A deviation is a specific written authorization, granted prior to the manufacture of an item, to depart from a particular performance or design requirement of a specification, drawing or other document for a specific number of units or a specific period of time.

An approved engineering change requires corresponding revision of the documentation defining the affected item. A deviation does not require revision of the applicable specification or drawing. A waiver, on the other hand, is a written authorization to accept a configuration item or other designated items that departs from the specified requirements, but nevertheless is considered suitable for use "as is" or after rework by an approved method.

Of the two standards, DOD-STD-480A covers the broader area and requires a more complete analysis of the impact if the engineering change described by an Engineering Change Proposal (ECP) were implemented. DOD-STD-480A requires that the data package submitted with an ECP contain a description of all known interface effects and information concerning changes required in the Functional/Allocated/Product Configuration Identification (FCI/ACI/PCI). A configuration identification is the current approved or conditionally approved technical documentation for a configuration item as set forth in specifications, drawings, and associated lists, and documents referenced therein. (DOD-STD-480A, 1978, p. iii)

MIL-STD-481A is intended for use in contracts/development efforts involving detailed design of the system at hand but was not developed by the present contractor/developer. MIL-STD-481A sets forth the requirements for preparation and submittal of an abbreviated engineering

proposal. Required information emphasizes the impact on the item under contract with limited description of the effect on interfaces and integrated logistic support (MIL-STD-481A, 1972, p. ii). In this case, the burden and responsibility for analysis of the impact of an ECP on associated items falls on the contracting/procuring activity vice the contractor.

Both standards delineate situations for imposing standards on prime contractors. They also specify when to utilize an ECP, deviation or waiver. Definition of ECP classes and associated criteria, justification codes, and priorities are also documented. These standards are primarily designed to meet acquisition needs/requirements of hardware and/or other major "physical" systems (equipments) and show the relationship between the program phases and the ECP data required at each phase of the procurement/acquisition Life Cycle. A major portion of the standard is devoted to block by block instructions for the preparation of ECPs, deviations and waivers utilizing military (DD) forms and to the associated review and approval process that accompanies these proposals. Although designed for "hardware," they may also be applied to CSCIs, to the extent that these standards will be used to prepare ECPs, deviations, and waivers in relation to CSCIs and paragraph 4.5.1 of DOD-STD-2167A. The exact documentation that establishes the functional, allocated, and developmental configuration of any specific CSCI

are delineated by paragraph 5.x.5 of DOD-STD-2167A for each of the software development activities in the Life Cycle. DOD-STD-480A and MIL-STD-481A can be tailored, but are dependent on an individual project's functional and product specifics, and therefore cannot be appropriately applied to FNOC's general categories of software projects. However, when applied to specific, individual projects, no incompatibilities, duplication, or conflicts will be created if DOD-STD-480A or MIL-STD-481A is selected (MIL-HDBK-287, 1989, p. 113).

2. MIL-STD-490A

MIL-STD-490A establishes and sets forth uniform practices for the preparation, interpretation, change, and revision of program peculiar specifications to ensure the inclusion of essential requirements, and to aid in the use and analysis of specification content (MIL-STD-490A, 1985, pp. iii/iv). It is related to DOD-STD-2167A to the extent that DOD-STD-2167A paragraph 4.5.5, General Requirements, requires the contractor or developing agency to prepare Specification Change Notices (SCNs) in accordance with MIL-STD-490A. A Specification Change Notice is a document used to propose, transmit, and record changes to a specification utilizing DD Form 1696 and in accordance with paragraph 3.3.2.1 of MIL-STD-491A. In this capacity, no duplications or conflicts would be created if MIL-STD-490A is invoked.

Section Six of the standard contains a list of DIDs applicable to the standard. These DIDs cross-reference DOD-STD-2167A. Some DIDs have changed titles and others have been consolidated to form a single DID in DOD-STD-2167A.

MIL-STD-490A paragraph references are in parentheses:

1. System/Segment Specification, DI-CMAN-80008A
(3.1.3.1).
2. Software Requirements Specification, DI-MCCR-80025A
(3.1.3.2.5.1).
3. Interface Requirements Specification, DI-MCCR-80026A
(3.1.3.3.5).
4. Software Product Specification, DI-MCCR-80029A
(3.1.3.3.5.1).
5. Software Top Level Design Document, DI-MCCR-80012A
(3.1.3.3.5.1).
6. Software Detailed Design Document, DI-MCCR-80031
(3.1.3.3.5.1, 3.1.3.3.5.1).
7. Interface Design Document, DI-MCCR-80027A
(3.1.3.3.5.4).
8. Data Base Design Document, DI-MCCR-80028
(3.1.3.3.5.3).

The Software Top Level Design Document has been renamed the Software Design Document in DOD-STD-2167A, maintaining the same DID number. The Software Detailed Design Document and Data Base Design Document have been eliminated from DOD-STD-2167A and have been incorporated into the Software Design Document, DI-MCCR-80012A. These eight DIDs, listed above, are defined in the MIL-STD-490A, but their format and content preparation are required to be developed as specified by the approved Data Item Description (DD Form 1664).

Delivery should be in accordance with the approved CDRL (DD Form 1423) incorporated into the contract as specified in the DOD-STD-2167A.

The extent of the relationship between MIL-STD-490A and DOD-STD-2167A is in the area of SCNs, which is only a small part of MIL-STD-490A. Actual determination and preparation of hardware and software specifications (utilizing MIL-STD-490A) for individual projects is not in the scope of this thesis and must be the decision of FNOC personnel. However, it should be noted, if provisions of MIL-STD-490A beyond SCN preparation are invoked, one should make allowances for: (1) its non-current DID reference for System/Segment Specification, and (2) its citing of DOD-STD-2167 rather than DOD-STD-2167A software requirement and design documents (as previously addressed above) (MIL-HDBK-287, 1989, p. 113).

3. MIL-STD-499A

MIL-STD-499A was developed to assist government and contractor personnel in defining the system engineering effort in support of defense acquisition programs. The fundamental concept of the standard is to present a single set of criteria against which contractors/government agencies may propose their individual internal procedures as a means of satisfying the engineering requirements of the particular system under development. It specifies and prepares requirements for systems engineering for inclusion

in solicitation documents, contract work statements, and System Engineering Management Plans (SEMP). A SEMP is a comprehensive and complete entity within a contractor's proposal which describes how a fully integrated engineering effort will be managed and conducted. It consists of three parts: (1) technical program planning and control, (2) system engineering process, and (3) engineering specialty integration. (MIL-STD-499A, 1974, pp. ii,1)

The extent to which this standard is related to DOD-STD-2167A is cited in paragraph 1.2.1 of DOD-STD-2167A and delineates that DOD-STD-2167A should be used in conjunction with MIL-STD-499A when in the "total system" development context. ("Total system" includes hardware/firmware specifications and integration in addition to software development.) Other broad relationships exist, but are applicable to the "systems" level vice software development level. Technical, system level reviews are to be conducted in accordance with MIL-STD-1521B and identified in the SEMP. Configuration control change requirements (ECPs) and "system" configuration specifications for program-peculiar items are to be generated and prepared in accordance with DOD-STD-480A and MIL-STD-499A respectively. If software development does, however, take place in the context of "total system" development, no overlaps or conflicts will exist between MIL-STD-499A and DOD-STD-2167A.

4. MIL-STD-1521B

MIL-STD-1521B specifies requirements for scheduling and conducting reviews and audits on systems, equipment, and computer software. It delineates such requirements as location of reviews and audits, contractor requirements, and contracting agency participation responsibilities. With regard to DOD-STD-2167A, paragraphs 5.2.1, 5.3.1, 5.4.1 and 5.6.1 require the contractor to conduct Software Specification Reviews, Preliminary Design Reviews, Critical Design Reviews, and Test Readiness Reviews respectively.

The contracting agency shall tailor MIL-STD-1521B to require only what is needed for each individual acquisition (MIL-STD-1521B, 1985, p. 1/2). This ensures the cost effective application of the requirements of the standard when it is contractually invoked. Tailoring eliminates inapplicable and unnecessary requirements, provides for adding/modifying necessary technical review and audit factors not included in MIL-STD-1521B, and eliminates redundancy and inconsistency with other contract specifications and standards. MIL-STD-1521B is not a stand alone document and is dependent upon the work effort specified in the SOW. Tailoring of specifications should take place in all phases of military procurement, but is especially applicable to the initial stages of solicitation package preparation and contract negotiation. Depending upon the type of end-item(s) under procurement, the reviews and audits outlined

by MIL-STD-1521B may or may not be required for all programs. (MIL-STD-1521B, 1985, p. 117)

The update of MIL-STD-1521A in June 1985 provided compatibility with DOD-STD-2167, but no comparable revision has been undertaken to achieve consistency with DOD-STD-2167A. The documents cited for each review/audit are incompatible with those required by DOD-STD-2167A. Unless MIL-STD-1521B is tailored out of DOD-STD-2167A, the contractor and/or contracting agency needs to tailor MIL-STD-1521B to resolve the inconsistencies. (MIL-HDBK-287, 1989, p. 113) Guidelines for tailoring MIL-STD-1521B can be found in Appendix J of the standard and Appendix B of MIL-HDBK-287. Tailoring results of the MIL-STD-1521B in relation to FNOC's software project categories will be found in Appendices A, B, and C for Major, Intermediate, and Minor projects respectively.

5. MIL-STD-483A

MIL-STD-483A specifies and sets forth configuration management practices for hardware and software. It is designed to be tailored to specific programs and implemented by the SOW. The standard also establishes and supplements configuration management and control requirements which are not covered in DOD-STD-480A, MIL-STD-481A, MIL-STD-482, and MIL-STD-490A. (MIL-STD-483A, 1985, pp. iii/iv). This standard is not invoked or referenced by DOD-STD-2167A. DOD-STD-2167A paragraphs 4.5, General

Requirements, and 5.x.5, Detailed Requirements, specify requirements for the delivery of specific deliverables/ documents (DIDs) at the completion of each software development activity in the Life Cycle. These paragraphs are considered to be adequate and self-sufficient. (MIL-HDBK-287, 1989, p. 114)

Section six of MIL-STD-483A contains a list of DIDs applicable to the standard. These DIDs cross-reference DOD-STD-2167A. Some DIDs have changed titles and others have consolidated to form a single DID in DOD-STD-2167A. MIL-STD-483A paragraph references are in parentheses:

1. Software Development Plan, DI-MCCR-80030A (3.1.1).
2. Software Configuration Management Plan, DI-MCCR-80009 (3.1.1).
3. Version Description Document, DI-MCCR-80013A (80.5.4).
4. Software Requirements Specification, DI-MCCR-80025A (3.4.2, 3.4.7.1).
5. Interface Requirements Specification, DI-MCCR-80026A (3.4.2, 3.4.7.1).
6. Software Product Specification, DI-MCCR-80029A (3.4.7.3).
7. Software Top Level Design Document, DI-MCCR-80031 (3.4.7.2).
8. Software Detailed Design Document, DI-MCCR-80031 (3.4.7.2).
9. Interface Design Document, DI-MCCR-80027A (3.4.7.2).
10. Data Base Design Document, DI-MCCR-80028 (3.4.7.2).

The Software Configuration Management Plan and the Software Detailed Design Document have been eliminated from

DOD-STD-2167A as separate deliverables and have been incorporated into the Software Development Plan, DI-MCCR-80030A, and Software Design Document, DI-MCCR-80012A respectively. The reference to the Software Top Level Design Document in MIL-STD-483A is now the Software Design Document in DOD-STD-2167A. The DID number and contents are the same, only the name changed. The Data Base Design Document has also been eliminated from DOD-STD-2167A and has been incorporated into the Software Design Document. Although definitions for those DIDs listed above are contained in MIL-STD-483A, their format and content preparation are required to be developed as specified by the approved Data Item Description (DD Form 1664) and delivered in accordance with the approved CDRL (DD Form 1423) incorporated into the contract as specified in DOD-STD-2167A.

There are other areas of MIL-STD-483A that are related specifically to DOD-STD-2167A and software development. Appendix III concerns the System Specification/System Segment Specification and has been replaced by a System/Segment Specification, DI-CMAN-80008A in DOD-STD-2167A. Likewise, Appendix VI, Computer Software Configuration Item Specification has been divided and replaced by Software Requirements Specification, DI-MCCR-80025A, and Interface Requirements Specification, DI-MCCR-80026A, DIDs in

DOD-STD-2167A. Appendix XII on configuration audits has been replaced by MIL-STD-1521B, Appendices G, H, and I.

MIL-STD-483A is a totally complete and comprehensive document on configuration management for system development of any type. Configuration management is an extremely critical and viable aspect of software, as well as system, development. Configuration management is a discipline applying technical and administrative direction to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, and record and report change processing and implementation status. Configuration management is the means through which integrity and continuity of a design are recorded, communicated, and controlled by program and functional managers. (MIL-STD-483A, 1985, p. 5) MIL-STD-483A covers interface control as well as specification and support documentation maintenance. It provides supplemental information to DOD-STD-480A and MIL-STD-490A on formatting and the preparation of ECPs and SCNs respectively. MIL-STD-483A is a document so integrated and comprehensive that an organization and/or contract invoking it will want to identify all appropriate paragraphs and appendices, or portions thereof, in the SOW applicable to the nature and scope of the specific program/project.

Although DOD-STD-2167A's coverage of configuration management is stipulated to be self-sufficient, FNOC

personnel should consider utilizing MIL-STD-483A in the development of any comprehensive organizational configuration management plan. A complete analysis, evaluation, and implementation of the standard, with respect to FNOC's specific configuration management requirements, is beyond the proposed scope of this research effort. However, since configuration management is considered a key part of the FNOC Software Improvement Program (SIP), an in-depth analysis of whether to utilize MIL-STD-483A as the means to a viable command configuration management plan might be performed at this level.

Worthy of note here is the value of the ANSI/IEEE STD 828-1983, Standard for Software Configuration Management Plans and IEEE P1042/D5.0, Guide to Software Configuration Management, in establishing a configuration management strategy for the command. STD 828-1983 provides an outline for documenting a Software Configuration Management Plan (SCMP). The "Guide" discusses SCM as a set of management disciplines within the context of a software engineering process rather than as a set of specific activities to be performed or as functions of an organization. It focuses on the planning aspect of SCM vice implementation. The "Guide" is divided into two parts. The first part is a discussion of issues to consider when planning SCM for a project or organization. The second part presents a series (Appendices) of sample plans illustrating different concepts and

considerations in SCMPs for different types of projects and organizations. The issues discussed in the body of the "Guide" concentrate on areas such as the process of developing a baseline, relating the elements of the SCMP to specific activities of the project's or organization's management organization, SCM activities and responsibilities, supplier and sub-contractor relationships, and records collection and retention. A major portion of text discusses interface control and the use of a Configuration Control Board (CCB). A CCB reviews proposed changes for assuring compliance with approved specifications and designs, and evaluates impacts on existing software. Each engineering change or problem report which is initiated against a formally identified configuration item is evaluated by the CCB to determine its necessity and impact. Therefore, given the proper level of authority a CCB can grant change approval. No organization should implement a SCMP without first reviewing these two documents for potential insight, impact, and alternatives that could enhance an organization's SCMP.

6. DOD-STD-2168

DOD-STD-2168 specifies requirements for a software quality program and supplements the DOD-STD-2167A requirements for software product evaluations. DOD-STD-2168 requires evaluation of software products and activities for compliance with the contract and adherence to software

planning documents. No duplication or conflicts will be created if DOD-STD-2168 is selected. (MIL-HDBK-287, 1989, p. 114) A more detailed discussion of DOD-STD-2168 and software quality assurance will take place later in Appendix E of the thesis.

7. Other References

There are several other references related to DOD-STD-2167A, but are neither invoked nor referenced by it. Their frequency of use, for FNOC requirements, is expected to be minimal and only when project situations and circumstances warrant invoking them. They are mentioned here only as a cursory note and reminder to the reader of their existence.

a. MIL-STD-482

This standard specifies requirements for preparing and maintaining configuration status accounting records. Paragraph 4.5.3 of DOD-STD-2167A briefly discusses general requirements for configuration status accounting records if required to be kept. The paragraph is meant to be self-sufficient, but is not in conflict with MIL-STD-482 if both standards are put on contract (MIL-HDBK-287, 1989, p. 114).

b. MIL-STD-882

This standard specifies requirements for Software System Safety Analysis. When warranted by the nature/environment of a specific project, performance of Software Safety Analysis is required by DOD-STD-2167A,

paragraph 4.2.3, and tailored versions of MIL-STD-882 may be used to define the type of Software Safety Analysis that needs to be performed.

c. MIL-STD-1535

MIL-STD-1535 specifies requirements of prime contractors in the development of quality assurance programs where subcontractors are concerned. DOD-STD-2167A paragraph 4.1.6 delineates general requirements in subcontractor management. MIL-STD-1535 describes the process required of the prime contractor for evaluation of the subcontractor's products. It is focused on hardware rather than software (MIL-HDBK-287, 1989, p. 114).

III. SUMMARY/RECOMMENDATIONS

DOD-STD-2167A establishes a comprehensive, uniform set of requirements for software development. The standard affords the government control over the developer's/contractor's development, testing and evaluation efforts. It provides a means for ensuring required levels of quality in software and associated documentation. DOD-STD-2167A is divided into six functional areas with eight detailed requirements/software development activities contained within each one. The framework is meant to allow tailoring to meet specific project requirements. Functional areas are grouped so that they can be tailored consistently.

Although the standard is fairly comprehensive and seems to place rigorous controls on the entire development process, it does allow a great deal of developer/contractor flexibility in meeting contract requirements. DOD-STD-2167A provides a broad framework of software development, leaving application specific details to the discretion of the developer. The standard is designed to be compatible with any software development model. It specifically avoids terminology which might suggest sequential development and instead specifies, in paragraph 4.1.1, that software development activities "may overlap and may be applied iteratively or recursively." The specific plans for conducting

these activities are proposed and documented in the Software Development Plan.

DOD-STD-2167A shows flexibility in its requirements for the use of a specific software development methodology or programming language. It is designed to be compatible with any software development methodology or any programming language. The standard supports no default methodology and states in the foreword that "the standard is not intended to specify or discourage the use of any particular software development method." The only constraint on the use of any software development method is in paragraph 4.2.1 which states that the contractor must use a systematic, well-documented software development method that supports the reviews and audits required by the contract.

The intent and design of DOD-STD-2167A is geared to meet the needs and requirements of a specific software project/development effort. Subsequent tailoring of the standard's requirements, and their associated DIDs, is also in keeping with this premise. However, a major theory behind this research effort was that there must be some version of the standard that could be tailored and applied to an entire organization's software requirements, or if not to the organization as a whole, maybe to a subset or category of the organization's software projects. As a result, some interesting findings and possible benefits were unveiled.

First of all, it is felt that no tailored version of the standard would meet the requirements of or apply to all projects within FNOC. The numbers and types of software development efforts, and their respective characteristics and requirements, are so varied that they don't lend themselves to a single tailored version of the standard that would be practical or beneficial. However, dividing development efforts into project categories appropriate to the organization does produce some useful results. Tailoring the standard according to selective criteria of a project category resulted in "skeletonized" versions of the standard and its DIDs. These "skeletonized" versions provide a project manager with a starting/reference point by automatically deleting those requirements and activities that would most likely never be needed by a project within that respective category. However, it should be noted that these versions are only tailored to a "category" of projects and that in order to produce a truly cost-effective and complete SOW, one must further refine/tailor the standard for a specific project's requirements.

Software development efforts at FNOC were divided into categories of "Major," "Intermediate," and "Minor." The criteria selected for "Major" and "Minor" categories produced distinct software project categories and consequently resulted in separate tailored versions of the standard. The major difference between them was in the

number and types of activities performed, and in the number of DIDs required to be delivered. The contents of the individual DIDs that were common to both remained very similar with only minor deletions occurring in their references to hardware/firmware specifications and safety analysis (see Appendices A and C).

The "Intermediate" category proved more difficult with a quantitatively weaker set of criteria forthcoming (at least to the level necessary to utilize the "Tailor" software package and produce a list of required deliverables applicable to the category). The questions used by the "Tailor" process were much more difficult to answer for "Intermediate" category projects. Therefore, it was determined that the only viable method to tailor the standard to a project in this category was to start with the tailored version for a "Major" project and work down deleting requirements as applicable (see Appendix B).

It should also be noted that "tailoring" should not be confined to the deletion of non-applicable requirements. The tailoring process can also include the addition of requirements and/or activities as appropriate to fulfill the needs of specific projects, regardless of the category to which they belong. The focus of this thesis dealt only with the deletion of requirements not applicable to general project category criteria. Any addition of requirements and/or activities would have to be considered and determined

on a case by case basis dependent on individual project needs and requirements.

With the passage of time and increased experience, FNOC personnel should become more adept in the use and tailoring of DOD-STD-2167A and its associated DIDs. As their knowledge and use of DOD-STD-2167A matures, it will become more widely distributed and accepted as a beneficial and practical standard. Training sessions should be planned and scheduled to expose personnel to the standard's terminology, contents, requirements, theory and steps involved in tailoring the standard, both manually and by utilizing the automated "Tailor" software package. FNOC should also look into the feasibility of setting up a "library" of tailored versions of the standard used in past development efforts. Once established, project managers could refer to this collection of already tailored standards and possibly find a version that closely matches the requirements of his/her present project. This provides the project manager with an immediately better reference point from which to begin tailoring. This should allow for only minor adjustments to be made before it is usable by a project under consideration.

APPENDIX A
MAJOR PROJECTS

This appendix represents the tailored version of DOD-STD-2167A for a FNOC "Major" project. It is basically divided into two parts. The first part represents the tailored DOD-STD-2167A standards requirements utilizing the automated tailoring software package "Tailor" and contains five reports:

1. Project Description Answers.
2. Quick-Tailor Selections.
3. Statement of Work Report.
4. Detailed Status Report.
5. Action Item List.

The second part contains the tailored set of DIDs and reviews and audits (MIL-STD-1521B) appropriate to this set of tailored standards requirements and applicable to a "Major" project. It should, however, be noted that this tailored version represents only a skeletonized package applicable to a general category of software projects and must be further tailored/refined to meet the individual requirements of a specific project within this category. Of particular note are items 23 through 32 of the Action Item List Report. These items (shell requirements), regardless of the fact that we have now moved to categories of software

projects, still remain project dependent/specific and must be specifically stated in a contract/development plan and dealt with during the refinement process for that particular project.

NOTE: The System/Segment Specification and the System/Segment Design Document are included in a "Major" project to ensure maximum coverage. Likewise, the System Requirements Review and the System Design Review are also included. They are only required if hardware is involved in the development effort. However, if the development effort only involves software then these products, reviews, and all reference to them should be tailored out for that specific project. It should also be noted that most of DOD-STD-2167A and its associated DIDs are applicable to a "Major" project. Therefore, this tailored version of the standard and the DIDs consists of those items that are non-applicable and should be eliminated because of "Major" project characteristics. In addition, if the development effort includes hardware and/or firmware then disregard those tailoring decisions that delete reference to hardware and/or firmware. This tailored version of the standard assumes that they are not a part of the development effort.

TAILOR REPORT
PROJECT DESCRIPTION ANSWERS
PROJECT: MAJOR

SCREEN 1. ACTIVITIES OVERVIEW

- | | | |
|-----|-----|--|
| YES | 1. | Help define system-level requirements and design |
| YES | 2. | Define software requirements |
| YES | 3. | Design the software |
| YES | 4. | Code the software |
| YES | 5. | Perform unit/component testing |
| YES | 6. | Perform formal qualification testing |
| YES | 7. | Support system-level testing |
| YES | 8. | Perform product evaluations |
| YES | 9. | Perform configuration management |
| YES | 10. | Prepare for software use and support |
| YES | 11. | Install software at the support site |
| YES | 12. | Conduct/support formal reviews and audits |

SCREEN 2. PROJECT CHARACTERISTICS

- | | | |
|-----|-----|--|
| YES | 1. | There may be subcontractors |
| YES | 2. | There are security issues on this project |
| NO | 3. | Some or all of the software will be implemented in firmware |
| YES | 4. | The software will have user interfaces |
| YES | 5. | The contract will specify reserve memory/timing capacities |
| YES | 6. | There will be an IV&V agent |
| YES | 7. | Contractor will be responsible for CSCI-external interfaces |
| YES | 8. | Software errors could threaten personnel or property safety |
| YES | 9. | The software will run on computers with commercial/GFE manuals |
| YES | 10. | There are significant risks to project success |

SCREEN 3. SOFTWARE DEVELOPMENT PRACTICES

- | | | |
|-----|----|--|
| YES | 1. | Require 2167A project management practices |
| YES | 2. | Require 2167A software engineering practices |
| YES | 3. | Require 2167A formal qualification testing practices |
| YES | 4. | Require 2167A software product evaluation practices |
| YES | 5. | Require 2167A configuration management practices |

TAILOR REPORT
QUICK-TAILOR SELECTIONS
PROJECT: MAJOR

SCREEN 1. PRODUCTS TO BE DEVELOPED

- YES 1. Source code
- YES 2. Software Development Plan (SDP)
- YES 3. System/Segment Design Document (SSDD)
- YES 4. Software Requirements Specification (SRS)
- YES 5. Interface Requirements Specification (IRS)
- YES 6. Software Design Document (SDD)
- YES 7. Interface Design Document (IDD)
- YES 8. Software Product Specification (SPS)
- YES 9. Software Test Plan (STP)
- YES 10. Software Test Description (STD)
- YES 11. Software Test Report (STR)
- YES 12. Computer Resources Integrated Support Document (CRISD)
- YES 13. Computer System Operator's Manual (CSOM)
- YES 14. Software User's Manual (SUM)
- NO 15. Software Programmer's Manual (SPM)
- NO 16. Firmware Support Manual (FSM)
- YES 17. Version Description Document (VDD)

SCREEN 2. REVIEWS AND AUDITS

- YES 1. System Requirements Review (SRR)
- YES 2. System Design Review (SDR)
- YES 3. Software Specification Review (SSR)
- YES 4. Preliminary Design Review (PDR)
- YES 5. Critical Design Review (CDR)
- YES 6. Test Readiness Review (TRR)
- YES 7. Functional and Physical Configuration Audits (FCA, PCA)

SCREEN 3. PROJECT MANAGEMENT REQUIREMENTS

- YES 1. Ensure subcontractor compliance with prime contract
- YES 2. Implement a corrective action process
- YES 3. Prepare problem/change reports
- YES 4. Internally coordinate products before delivery
- YES 5. Implement risk management procedures
- YES 6. Interface with the software IV&V agent
- YES 7. Establish a software development library
- YES 8. Implement security measures as specified in the contract

QUIK-TAILOR SELECTIONS (Continued)

SCREEN 4. SOFTWARE ENGINEERING PRACTICES

- | | | |
|-----|-----|---|
| YES | 1. | Use systematic, documented development methods |
| YES | 2. | Establish a software engineering environment |
| YES | 3. | Analyze the system specification |
| NO | 4. | Perform safety analysis on requirements, design, and procedures |
| YES | 5. | Consider the use of non-developmental software |
| YES | 6. | Monitor and maintain required reserve processing capacity |
| YES | 7. | Organize CSCIs into CSCs, CSUs |
| YES | 8. | Establish software development files for CSCIs, CSCs, CSUs |
| YES | 9. | Document and implement design and coding standards |
| YES | 10. | Use the approved HOL or obtain approval to use another |

SCREEN 5. REQUIREMENTS FOR CONTRACTOR-INTERNAL TESTING

- | | | |
|-----|----|---|
| YES | 1. | Develop and document CSU test requirements, cases, and procedures |
| YES | 2. | Perform CSU testing; record results |
| YES | 3. | Develop and document CSC test requirements, cases, and procedures |
| YES | 4. | Perform CSC integration and testing; record results |

SCREEN 6. REQUIREMENTS FOR FORMAL QUALIFICATION TESTING

- | | | |
|-----|----|--|
| YES | 1. | Establish a software test environment |
| YES | 2. | Dry run FQT procedures |
| YES | 3. | Perform FQT on the target computer |
| YES | 4. | Use test personnel independent of software engineers |

SCREEN 7. REQUIREMENTS FOR SYSTEM INTEGRATION AND TESTING

- | | | |
|-----|----|---|
| YES | 1. | Support the development of system test plans and procedures |
| YES | 2. | Support system integration and testing |
| YES | 3. | Support post-test analysis and test reporting |

QUICK-TAILOR SELECTIONS (continued)

SCREEN 8. PREPARATIONS FOR SOFTWARE SUPPORT

- YES 1. Produce code regenerable and maintainable on a designated system
- YES 2. Install the code in the support environment
- YES 3. Provide training and continuing support

SCREEN 9. PRODUCT EVALUATIONS

- YES 1. Source code
- YES 2. Software Development Plan
- NO 3. System/Segment Design Document
- YES 4. Software Requirements Specification
- YES 5. Interface Requirements Specification
- YES 6. Software Design Document
- YES 7. Interface Design Document
- YES 8. Software Test Plan
- YES 9. Software Test Description
- YES 10. Software Test Report
- YES 11. CSU test requirements, cases, procedures, results
- YES 12. CSC test requirements, cases, procedures, results
- YES 13. Sample of software development files
- YES 14. Keep records of evaluations
- YES 15. Use evaluation criteria in the standard
- YES 16. Require evaluators to be independent of s/w engineers

SCREEN 10. CONFIGURATION MANAGEMENT REQUIREMENTS

- YES 1. Perform configuration identification
- YES 2. Perform configuration control
- YES 3. Perform configuration status accounting
- YES 4. Implement procedures for storage, handling, and delivery
- YES 5. Prepare ECPs and SCNs IAW DOD-STD-480, MIL-STD-481, MIL-STD-490

TAILOR REPORT
STATEMENT OF WORK REPORT
PROJECT: MAJOR

The contractor shall comply with all requirements in DOD-STD-2167A, with the following exceptions:

- 4.2.3 Delete entire paragraph.
- 4.6.4.d Delete entire paragraph.
- 4.6.4.e Delete entire paragraph.
- 5.1.4.b Delete entire paragraph.

END OF STATEMENT OF WORK REPORT

TAILOR REPORT
DETAILED STATUS REPORT
PROJECT: MAJOR

Para.	Status	Title/Description
4.	K	General Requirements
4.1	K	Software development management
4.1.1	K	Software development process
4.1.1.a	K	(System Requirements Analysis/Design)
4.1.1.b	K	(Software Requirements Analysis)
4.1.1.c	K	(Preliminary Design)
4.1.1.d	K	(Detailed Design)
4.1.1.e	K	(Coding and CSU Testing)
4.1.1.f	K	(CSC Integration and Testing)
4.1.1.g	K	(CSCI Testing)
4.1.1.h	K	(System Integration and Testing)
4.1.2	K	Formal reviews/audits
4.1.3	K	Software development planning
4.1.4	K	Risk management
4.1.5	K	Security
4.1.6	K	Subcontractor management
4.1.7	K	Interface with software IV&V agent(s)
4.1.8	K	Software development library
4.1.9	K	Corrective action process
4.1.9.a	K	(Implement a closed-loop process)
4.1.9.b	K	(Provide inputs to corrective action process)
4.1.9.c	K	(Classify problems by category and priority)
4.1.9.d	K	(Perform trend analysis)
4.1.9.e	K	(Evaluate corrective action taken)
4.1.10	K	Problem/change report
4.2	K	Software engineering
4.2.1	K	Software development methods
4.2.2	K	Software engineering environment
4.2.3	D	Safety analysis
4.2.4	K	Non-developmental software
4.2.5	K	Computer software organization
4.2.6	K	Traceability of requirements to design
4.2.7	K	High order language
4.2.8	K	Design and coding standards
4.2.9	K	Software development files
4.2.9.a	K	(Software development file contents)
4.2.9.b	K	(Software development file contents)
4.2.9.c	K	(Software development file contents)
4.2.9.d	K	(Software development file contents)
4.2.9.e	K	(Software development file contents)
4.2.10	K	Processing resource and reserve capacity
4.3	K	Formal qualification testing
4.3.1	K	Formal qualification test planning
4.3.2	K	Software test environment
4.3.3	K	Independence in FQT activities

DETAILED STATUS REPORT (continued)

- 4.3.4 K Traceability of requirements to test cases
- 4.4 K Software product evaluations
- 4.4.1 K Independence in product evaluation activities
- 4.4.2 K Final evaluations
- 4.4.3 K Software evaluation records
- 4.4.4 K Evaluation criteria
- 4.5 K Software configuration management
- 4.5.1 K Configuration identification
 - 4.5.1.a K (Identify baseline documentation)
 - 4.5.1.b K (Identify documentation and media under CM)
 - 4.5.1.c K (Identify each CSCI, CSC, and CSU)
 - 4.5.1.d K (Identify version, release, and change status)
 - 4.5.1.e K (Identify code/documentation relationship)
 - 4.5.1.f K (Identify deliverable medium contents)
- 4.5.2 K Configuration control
 - 4.5.2.a K (Establish Developmental Configuration(s))
 - 4.5.2.b K (Maintain current copies of deliverables)
 - 4.5.2.c K (Provide access to documents and code under CM)
 - 4.5.2.d K (Control change to master copy of deliverables)
- 4.5.3 K Configuration status accounting
 - 4.5.3.a K (Provide traceability of changes)
 - 4.5.3.b K (Communicate configuration status)
 - 4.5.3.c K (Ensure consistency between documents and code)
- 4.5.4 K Storage, handling, and delivery of media
- 4.5.5 K Engineering Change Proposals
- 4.6 K Transitioning to software support
- 4.6.1 K Regenerable and maintainable code
- 4.6.2 K Transition planning
- 4.6.3 K Software transition and continuing support
- 4.6.4 K Software support and operational documentation
 - 4.6.4.a K (CRISD)
 - 4.6.4.b K (CSOM)
 - 4.6.4.c K (SUM)
 - 4.6.4.d D (SPM)
 - 4.6.4.e D (FSM)
- 5. K Detailed Requirements
- 5.1 K System requirements analysis/design
 - 5.1.1 K Software development management
 - 5.1.1.1 K (Support the System Requirements Review (SRR))
 - 5.1.1.2 K (Support the System Design Review (SDR))
 - 5.1.2 K Software engineering
 - 5.1.2.1 K (Analyze preliminary system specifications)
 - 5.1.2.2 K (Allocate system requirements)
 - 5.1.2.3 K (Define preliminary engineering requirements)
 - 5.1.2.4 K (Define preliminary interface requirements)
 - 5.1.3 K (Define CSCI preliminary qualification reqmts)
 - 5.1.4 K Software product evaluations
 - 5.1.4.a K (Evaluate the SDP)
 - 5.1.4.b D (Evaluate the SSDD)
 - 5.1.4.c K (Evaluate the preliminary SRS for each CSCI)

DETAILED STATUS REPORT (continued)

- 5.1.4.d K (Evaluate the IRS)
- 5.1.5 K Configuration management
- 5.1.5.a K (Place the SDP under configuration control)
- 5.1.5.b K (Place the SSDD under configuration control)
- 5.1.5.c K (Put prelim. SSRs under configuration control)
- 5.1.5.d K (Place prelim. IRS under configuration control)
- 5.2 K Software requirements analysis
- 5.2.1 K Software development management
- 5.2.2 K Software engineering
- 5.2.2.1 K (Define engineering requirements)
- 5.2.2.2 K (Define interface requirements)
- 5.2.3 K (Define a complete set of qualification rqmts)
- 5.2.4 K Software product evaluations
- 5.2.4.a K (Evaluate the SRS for each CSCI)
- 5.2.4.b K (Evaluate the IRS)
- 5.2.5 K Configuration management
- 5.3 K Preliminary Design
- 5.3.1 K Software development management
- 5.3.2 K Software engineering
- 5.3.2.1 K (Develop preliminary design for each CSCI)
- 5.3.2.2 K (Develop prelim. design of CSCI external I/Fs)
- 5.3.2.3 K (Document other essential design information)
- 5.3.2.4 K (Establish CSC test requirements)
- 5.3.3 K (Identify qualification tests)
- 5.3.4 K Software product evaluations
- 5.3.4.a K (Evaluate the SDD for each CSCI)
- 5.3.4.b K (Evaluate the preliminary IDD)
- 5.3.4.c K (Evaluate the STP)
- 5.3.4.d K (Evaluate the CSC test requirements)
- 5.3.5 K Configuration management
- 5.3.5.1 K (Place SDDs into Developmental Configurations)
- 5.3.5.2 K (Place the STP under configuration control)
- 5.3.5.3 K (Place the IDD under configuration control)
- 5.4 K Detailed Design
- 5.4.1 K Software development management
- 5.4.2 K Software engineering
- 5.4.2.1 K (Develop detailed design for each CSCI)
- 5.4.2.2 K (Develop detailed design of CSCI external I/Fs)
- 5.4.2.3 K (Document other essential design information)
- 5.4.2.4 K (Define CSC test cases)
- 5.4.2.5 K (Define CSU test requirements and test cases)
- 5.4.3 K (Identify qualification test cases)
- 5.4.4 K Software product evaluations
- 5.4.4.a K (Evaluate the updated SDD)
- 5.4.4.b K (Evaluate the updated IDD)
- 5.4.4.c K (Evaluate CSC test cases)
- 5.4.4.d K (Evaluate CSU test requirements and test cases)
- 5.4.4.e K (Evaluate a percentage of CSU and CSC SDFs)
- 5.4.4.f K (Evaluate the STD)
- 5.4.5 K Configuration management

DETAILED STATUS REPORT (continued)

- 5.4.5.1 K (Update each Developmental Configuration)
- 5.4.5.2 K (Place updated IDD under configuration control)
- 5.4.5.3 K (Place each STD under configuration control)
- 5.5 K Coding and CSU testing
 - 5.5.1 K Software development management
 - 5.5.2 K Software engineering
 - 5.5.2.1 K (Develop CSU test procedures)
 - 5.5.2.2 K (Code and test CSUs)
 - 5.5.2.3 K (Revise documents and code based on CSU tests)
 - 5.5.2.4 K (Develop CSC test procedures)
 - 5.5.3 K Formal qualification testing
 - 5.5.4 K Software product evaluations
 - 5.5.4.a K (Evaluate the source code)
 - 5.5.4.b K (Evaluate the CSC test procedures)
 - 5.5.4.c K (Evaluate CSU test procedures and test results)
 - 5.5.4.d K (Evaluate a percentage of updated SDFs)
 - 5.5.5 K Configuration management
 - 5.5.5.1 K (Update each Developmental Configuration)
 - 5.5.5.2 K (Place source code under configuration control)
- 5.6 K CSC Integration and Testing
 - 5.6.1 K Software development management
 - 5.6.2 K Software engineering
 - 5.6.2.1 K (Conduct CSC integration and testing)
 - 5.6.2.2 K (Record CSC test results)
 - 5.6.2.3 K (Revise design documentation and code)
 - 5.6.3 K Formal qualification testing
 - 5.6.3.1 K (Develop FQT test procedures)
 - 5.6.3.2 K (Dry run FQT test procedures)
 - 5.6.4 K Software product evaluations
 - 5.6.4.a K (Evaluate CSC test results)
 - 5.6.4.b K (Evaluate updated STD)
 - 5.6.4.c K (Evaluate updated code and design documents)
 - 5.6.4.d K (Evaluate a percentage of updated SDFs)
 - 5.6.5 K Configuration management
- 5.7 K CSCI Testing
 - 5.7.1 K Software development management
 - 5.7.2 K Software engineering
 - 5.7.2.1 K (Revise documentation and code based on FQT)
 - 5.7.2.2 K (Revise the IDD based on FQT)
 - 5.7.2.3 K (Produce updated source code)
 - 5.7.2.4 K (Produce an SPS for each CSCI)
 - 5.7.3 K Formal qualification testing
 - 5.7.3.1 K (Perform formal qualification testing)
 - 5.7.3.2 K (Prepare Software Test Reports)
 - 5.7.3.3 K (Prepare updated STD for each CSCI)
 - 5.7.4 K Software produce evaluations
 - 5.7.4.a K (Evaluate STRs)
 - 5.7.4.b K (Evaluate updated code and design documents)
 - 5.7.5 K Configuration management
 - 5.7.5.1 K (Document the exact version of each CSCI)

DETAILED STATUS REPORT (continued)

- 5.7.5.2 K (Dis-establish Developmental Configurations)
- 5.8 K System integration and testing
- 5.8.1 K Software development management
- 5.8.2 K Software engineering
- 5.8.3 K Formal qualification testing
- 5.8.3.1 K (Support development of test documentation)
- 5.8.2.3 K (Support testing activities)
- 5.8.3.3 K (Support post test analysis)
- 5.8.4 K Software product evaluations
- 5.8.5 K Configuration management
- B. k Appendix B--Requirements for coding standards
- B.10.3.1 K Presentation style
- B.10.3.2 K Naming
- B.10.3.3 K Restrictions on the implementation language
- B.10.3.4 K Use of language constructs and features
- B.10.3.5 K Complexity
- C. K Appendix C--Category & priority classification
- C.10.2.a K Classify by category--Software problem
- C.10.2.b K Classify by category--Documentation problem
- C.10.2.c K Classify by category--Design problem
- C.10.3.a K Classify by priority--PRIORITY 1
- C.10.3.b K Classify by priority--PRIORITY 2
- C.20.3.c K Classify by priority--PRIORITY 3
- C.10.e.d K Classify by priority--PRIORITY 4
- C.10.3.e K Classify by priority--PRIORITY 5
- D. K Appendix D--Evaluation criteria
- D.10.2.1 K Internal consistency
- D.10.2.2 K Understandability
- D.10.2.3 K Traceability to indicated documents
- D.10.2.4 K Consistency with indicated documents
- D.10.2.5 K Appropriate techniques
- D.10.2.6 K Appropriate allocation of sizing and timing
- D.10.2.7 K Adequate test coverage of requirements
- D.10.3 K Additional criteria
- D.10.3.1 K Adequacy of quality factors
- D.10.3.2 K Testability of requirements
- D.10.3.3 K Consistency between data definition and use
- D.10.3.4 K Adequacy of test cases and test procedures
- D.10.3.5 K Completeness of testing
- D.10.3.6 K Completeness of retesting

TAILOR REPORT
ACTION ITEM LIST
PROJECT: MAJOR

1. Ensure that the software is required to be delivered as a contract line item.
2. Tailor the Software Development Plan (SDP) DID (DI-MCCR-80030A) for your project and require delivery of the SDP as a CDRL item.
3. Tailor the System/Segment Design Document (SSDD) DID (DI-CMAN-80534) for your project and require delivery of the SSDD as a CDRL item.
4. Tailor the Software Requirements Specification (SRS) DID (DI-MCCR-80025A) for your project and require delivery of the SRSS as CDRL items.
5. Tailor the Interface Requirements Specification (IRS) DID (DI-MCCR-80026A) for your project and require delivery of the IRSS as CDRL items.
6. Tailor the Software Design Document (SDD) DID (DI-MCCR-80012A) for your project and require delivery of the SDDs as CDRL items.
7. Tailor the Interface Design Document (IDD) DID (DI-MCCR-80027A) for your project and require delivery of the IDDs as CDRL items.
8. Tailor the Software Product Specification (SPS) DID (DI-MCCR-80029A) for your project and require delivery of the SPSS as CDRL items.
9. Tailor the Software Test Plan (STP) DID (DI-MCCR-80014A) for your project and require delivery of the STP as a CDRL item.
10. Tailor the Software Test Description (STD) DID (DI-MCCR-80015A) for your project and require delivery of the STDs as CDRL items.
11. Tailor the Software Test Report (STR) DID (DI-MCCR-80017A) for your project and require delivery of the STRs as CDRL items.
12. Tailor the Computer Resources Integrated Support Document (CRISD) DID (DI-MCCR-80024A) for your project and require delivery of the CRISD as a CDRL item.

ACTION ITEM LIST (continued)

13. Tailor the Computer System Operator's Manual (CSOM) DID (DI-MCCR-80018A) for your project and require delivery of the CSOMs as CDRL items.
14. Tailor the Software User's Manual (SUM) DID (DI-MCCR-80019A) for your project and require delivery of the SUMs as CDRL items.
15. Tailor the Version Description Document (VDD) DID (DI-MCCR-80013A) for your project and require delivery of the VDDs as CDRL items.
16. Specify in the Statement of Work how the contractor is to support the System Requirements Review (SRR). Guidance on the SRR is provided in MIL-STD-1521B.
17. Specify in the Statement of Work how the contractor is to support the System Design Review (SDR). Guidance on the SDR is provided in MIL-STD-1521B.
18. Put MIL-STD-1521B on contract and tailor its Software Specification Review (SSR) requirements for your project.
19. Put MIL-STD-1521B on contract and tailor its Preliminary Design Review (PDR) requirements for your project.
20. Put MIL-STD-1521B on contract and tailor its Critical Design Review (CDR) requirements for your project.
21. Put MIL-STD-1521B on contract and tailor its Test Readiness Review (TRR) requirements for your project.
22. Specify in the Statement of Work how the contractor is to support the Functional Configuration Audit and Physical Configuration Audit (FCA/PCA). Guidance on FCA and PCA is provided in MIL-STD-1521B.
23. Specify in the Statement of Work how the contractor is to interface with the software IV&V agent.
24. Ensure that the necessary security requirements are defined in the contract.
25. Specify memory and timing requirements in the Statement of Work.
26. Specify the required high order language in the contract, if desired.

ACTION ITEM LIST (continued)

27. Specify in the Statement of Work the contractor's role in the development of system test plans, test cases, and test procedures.
28. Specify in the Statement of Work the contractor's role in performing system integration and testing.
29. Specify in the Statement of Work the contractor's role in post-test analysis and reporting for system-level testing.
30. Specify in the Statement of Work the support environment computer system on which the delivered software must be regenerable and maintainable.
31. Specify in the Statement of Work the contractor's tasks related to installation and checkout in the support environment.
32. Specify in the Statement of Work the contractor's tasks related to training of support agency personnel and continuing support after delivery of the software.

END OF ACTION ITEM LIST

DID TAILORING DECISIONS

PROJECT: MAJOR

CDRL ITEM: Software Development Plan DID (DI-MCCR-80030A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 10.2.5.2.3 - Delete reference to firmware; no software will be implemented in firmware.
- 10.2.6.1.3 - Delete reference to firmware; no software will be implemented in firmware.
- 10.2.6.1.3.2 - Delete all reference to firmware; no software will be implemented in firmware.
- 10.2.8.5 - Update noncurrent reference to DOD-STD-2167 to read "DOD-STD-2167A."

CDRL ITEM: System/Segment Design Document DID (DI-CMAN-80534)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software Requirements Specification DID (DI-MCCR-80025A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 10.1.5.7 - Delete entire paragraph; performance of safety analysis on requirements, design, and procedures is not required.

CDRL ITEM: Interface Requirements Specification DID (DI-MCCR-80012A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software Design Document DID (DI-MCCR-80012A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Interface Design Document DID (DI-MCCR-80027A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

DID TAILORING DECISIONS (continued)

CDRL ITEM: Software Product Specifications DID (DI-MCCR-80029A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software Test Plan DID (DI-MCCR-80014A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

10.1.5 - Delete reference to firmware; no software will be implemented in firmware.

10.1.5.2 - Delete reference to firmware; no software will be implemented in firmware.

CDRL ITEM: Software Test Description DID (DI-MCCR-80015A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software Test Report DID (DI-MCCR-80017A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Computer Resources Integrated Support Document DID (DI-MCCR-80024A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Computer System Operator's Manual DID (DI-MCCR-80018A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

10.1.6.2 - Delete reference to firmware; no software will be implemented in firmware.

10.1.6.3 - Delete reference to firmware; no software will be implemented in firmware.

CDRL ITEM: Software User's Manual DID (DI-MCCR-80019A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Version Description Document DID (DI-MCCR-80013A)

DID TAILORING DECISIONS (continued)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software Programmer's Manual DID (DI-MCCR-80012A)

TAILORING ENTRY: This DID will not be delivered as a CDRL ITEM. A Software Programmer's Manual is not required.

CDRL ITEM: Firmware Support Manual DID (DI-MCCR-80022A)

TAILORING ENTRY: This DID will not be delivered as a CDRL ITEM. A firmware support manual is not required since no software will be implemented in firmware.

CDRL ITEM: System/Segment Specification DID (DI-CMAN-80008A)

NOTE: This DID is invoked by MIL-STD-490A vice DOD-STD-2167A and is only cited in DOD-STD-2167A. However, its format and content preparation are to be in accordance with this DID and it can be delivered as a CDRL ITEM.

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 10.1.5.2.4 - Delete reference to safety; performance of safety analysis on requirements, design, and procedures is not required.
- 10.1.5.3.1.1 - Delete paragraph; there are no requirements for the control of toxic products or formulations known at this time.
- 10.1.5.3.6 - Delete paragraph; performance of safety analysis on requirements, designs, and procedures is not required.
- 10.1.5.3.8 - Delete paragraph; there are no system requirements for nuclear components at this time.
- 10.1.8.1.2 - Delete paragraph; there are no present requirements to analyze current and potential enemy weapon capabilities or any other threat considerations that affect system design at this time.

MIL-STD-1521B (REVIEWS AND AUDITS)
TAILORING DECISIONS
PROJECT: MAJOR

SYSTEM REQUIREMENTS REVIEW (SRR)

The specific details of how the contractor should support the System Requirements Review (SRR) are project specific depending on the scope of the project, nature of the product, and operating environment. To apply specific details to such a general software project category, such as "Major," would be inappropriate and probably inaccurate. Some general requirements the contractor/developer might be responsible for are:

1. Establishing the time, place, and agenda for the SRR in consonance with the master milestone schedule.
2. Ensure that the review takes place appropriately near the completion of the System Requirements Analysis phase. (A significant portion of the system functional requirements (System/Segment Specification DID) needs to have been established in order to ascertain the adequacy of the contractor's efforts in defining and fulfilling system and user requirements.)
3. Prepare for the review in sufficient detail consistent with the scope and magnitude of the review.
4. Provide a stenographer to record inputs to official meeting minutes, action items, conclusions, and recommended courses of action resulting from discussions.

More specifically, the contractor/developer may want to review some of the items listed in Appendix A, paragraph 10.3 of MIL-STD-1521B, as appropriate. Also, the contractor should "discuss his progress and problems in:

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

1. Risk identification and risk ranking.
2. Risk avoidance/reduction and control.
3. Significant trade-offs among stated system/segment specification requirements/constraints and resulting engineering design requirements/constraints, manufacturing methods/progress constraints and unit production cost/design-to-cost objectives.
4. Identifying computer resources of the system and partitioning the system into HWCIs and CSCIs." (MIL-STD-1521B, 1985, pp. 21/22)

Any other responsibilities of the contractor, as alluded to earlier, would be totally dependent on the specific requirements of the system under consideration. (Even the MIL-STD-1521B only delineates and expresses "general" requirements such as those mentioned above.) The "bottom-line" is to discuss and review the most economical balance of elements which meet total system requirements and is responsive to the statement of work (SOW).

SYSTEM DESIGN REVIEW (SDR)

The details of the System Design Review, much like the System Requirements Review, are also very project specific taking into account the necessary requirements of the system under review. The general requirements that the contractor/developer are responsible for under the System Requirements Review are also applicable for the System Design Review except that it should be conducted upon completion of the System Design Phase vice System Requirements Analysis. Documents that should be reviewed include the System/Segment

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

Specification, Software Requirements Specification, Interface Requirements Specification, and the System/Segment Design Document. (All reference to the Operational Concept Document should be deleted and replaced with the System/Segment Design Document. This will make the MIL-STD-1521B, with respect to the System Design Review, compatible with the DOD-STD-2167A.) The SDR shall also include a review of those items listed in paragraphs 20.3 to 20.3.14.5 of MIL-STD-1521B, as appropriate to the specific system under review (MIL-STD-1521B, 1985, pp. 24-30). However, it should be noted that if the development effort consists only of software, paragraph 20.3.14 and all its related subparagraphs should be deleted since they relate only to HWCIs.

The following reviews/audits need to be tailored, as delineated in action items 18-22 (see Action Item List Report), and prior to use with DOD-STD-2167A in order to ensure inherent incompatibilities between DOD-STD-2167A and MIL-STD-1521B have been resolved.

* NOTE: These tailoring decisions are a result of a personal analysis of the various reviews/audits in MIL-STD-1521B by the author and from guidance contained in MIL-HDBK-287, Appendix B.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

SOFTWARE SPECIFICATION REVIEW (SSR)

- Documents to be reviewed:
 1. Software Requirements Specification.
 2. Interface Requirements Specification.
- Delete reference to Operational Concept Document. This document was replaced with the System/Segment Design Document and was already reviewed in the System Design Review.
- Replace paragraph 30.2.a-g with:
 - "a. The information contained in the Software Requirements Specification(s), as tailored.
 - b. The information contained in the Interface Requirements Specification(s), as tailored."

(Items a-g relate to requirements that already form part of a project's tailored Software Requirements Specification or Interface Requirements Specification.)
- Delete paragraph 30.2.h and 30.2.i as these were covered in the System Requirements Review.

All other paragraphs and requirements apply.

PRELIMINARY DESIGN REVIEW (PDR)

- Documents to be reviewed:
 1. Software Design Document.
 2. Interface Design Document.
 3. Software Test Plan.
 4. Computer Resources Integrated Support Document.
 5. Software User's Manual.
 6. Computer Software Operator's Manual.
- Paragraph 40.1: Delete reference to the Draft Hardware Product Specification if the development effort consists of software only; Delete reference to the Computer

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

System Diagnostic Manual as it has been eliminated from DOD-STD-2167A.

- Paragraph 40.1: Replace "Software Top Level Design Document," "Software Detailed Design Document," and "Data Base Design Document" with "Software Design Document"; all three documents have been combined and incorporated into the Software Design Document of DOD-STD-2167A.
- Paragraph 40.1: Delete reference to the Software Programmer's Manual and Firmware Support Manual as neither document is required to be delivered for this category of FNOC software development.
- Paragraphs 40.2.1.a-m, 40.3.1, 40.5.5, 40.5.7, 40.5.8, 40.5.9, 40.6.6, 40.6.8, 40.6.9, 40.9.1-3, 40.10 and all subparagraphs, 40.12 and all subparagraphs, 40.14 and all subparagraphs, 40.15 and all subparagraphs, 40.16 and all subparagraphs, and 40.19.1-3; delete these paragraphs if the development effort consists only of software. These paragraphs deal strictly in HWCIs.
- Paragraphs 40.5.1, 40.6.1, 40.6.2, 40.7.1, 40.7.2, 40.13.1, 40.13.3, 40.13.4 and all their related subparagraphs; delete all reference to hardware in these paragraphs if the development effort consists solely of software. All reference to CSCIs and software products still apply.
- Paragraph 40.8 and all subparagraphs; delete references to safety issues as safety analyses of requirements, design, and procedures are not required for this category of FNOC software development.
- Substitute "CSC" for "Top Level CSC(TLCSC)" and "Low Level CSC(LLCSC)" to make MIL-STD-1521B consistent with DOD-STD-2167A.
- Substitute "Preliminary Design" for "Top Level Design" to make MIL-STD-1521B consistent with DOD-STD-2167A.
- Replace 40.2.2.a-m with:

"40.2.2 CSCIs:

a. The preliminary design information contained in the Software Design Document(s), as tailored.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

b. The preliminary interface design information contained in the Interface Design Document(s), as tailored.

c. The contractor's plans for formal qualification testing as documented in the Software Test Plan, as tailored."

This substitution is made as items a-m (requirements) of 40.2.2 now form a part of one of these respective documents in DOD-STD-2167A.

- Replace 40.13.8-10 with:

"40.13.8 For CSCIs, review the Software Test Plan, or its equivalent as required by the contract, for completeness and technical adequacy in specifying plans for Formal Qualification Testing."

This is done because the requirements contained in these paragraphs are now incorporated into the Software Test Plan of DOD-STD-2167A.

- Substitute "CSU" for "Unit" to make MIL-STD-1521B consistent with DOD-STD-2167A.

CRITICAL DESIGN REVIEW (CDR)

- Documents to be reviewed:

1. Software Design Document.
2. Interface Design Document.
3. Software Test Description.
4. Computer Resources Integrated Support.
5. Software User's Manual.
6. Computer System Operator's Manual.

- Delete reference to the Firmware Support Manual and Software Programmer's Manual as these documents are not required to be delivered for this category of FNOC software projects.
- Paragraphs 50.1.1, 50.2.1.a-m, 50.3.1, 50.3.1.1.a-j, 50.4.a-c, 50.6.5, 50.9.1, 50.9.1-3, 50.10 and all

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

subparagraphs (except 50.10.3.e), 50.12 and all subparagraphs, 50.13.5, 50.15, 50.16; delete these paragraphs if the development effort consists only of software. These paragraphs deal strictly in HWCIs.

- Paragraphs 50.5.1, 50.5.5, 50.5.7, 50.6.1, 50.6.3, 50.7.1, 50.13.3, and 50.13.4; delete all reference to hardware if the development effort consists solely of software. All reference to CSCIs and software products still apply.
- Paragraphs 50.7.4.e, 50.8 and all subparagraphs; delete references to safety as safety analyses for requirements, design, and procedures are not required for this category of FNOC software development.
- Substitute "Software Design Document" for "Software Top Level Design Document" and "Software Detailed Design Document." These documents have been incorporated into the Software Design Document in the DOD-STD-2167A and now makes the MIL-STD-1521B compatible with DOD-STD-2167A.
- Delete all reference to "top level CSC" and "lower level CSC" and substitute "CSC." This makes MIL-STD-1521B compatible with DOD-STD-2167A.
- Delete all reference to "Unit" and substitute "CSU." This lends compatibility between MIL-STD-1521B and DOD-STD-2167A.
- Delete all reference to the Data Base Design Document as this document has been eliminated from DOD-STD-2167A. (It is now part of the Software Design Document in DOD-STD-2167A.)
- Substitute for paragraph 50.2.2.a:
 - "a. The detailed design information contained in the Software Design Document(s), as tailored.
 - b. The detailed interface design information contained in the Interface Design Document(s), as tailored.
 - c. The test plans and test case information contained in the Software Test Description, as tailored.

This substitution is made in order to make the documents, and their related information and requirements,

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

in paragraph 50.2.2 of MIL-STD-1521B compatible with those in DOD-STD-2167A.

TEST READINESS REVIEW (TRR)

- Documents to be reviewed:
 1. Software Test Description.
 2. Software User's Manual.
 3. Computer Software Operator's Manual.
 4. Computer Resources Integrated Support Manual.
- Delete all reference to the Firmware Support Manual and Software Programmer's Manual as these documents are not required to be delivered for this category of FNOC software projects.
- Delete reference to the Computer System Diagnostic Manual. This document has been eliminated from DOD-STD-2167A.
- Substitute "Software Design Document" for "Software Top Level Design Document," "Software Detailed Design Document," and "Data Base Design Document." These documents have all been incorporated into the Software Design Document of DOD-STD-2167A.
- Delete paragraph 60.2.4; Software test procedures now form a part of the Software Test Description and the requirement to review this document already exists.
- Add a requirement to review a summary of CSC testing and the Formal Qualification Testing (FQT) dry run results. This should be added because the contractor should conduct the tests documented in the Software Test Description for each CSCI to ensure that the procedures are complete and accurate and that the software is ready for FQT. (DOD-STD-2167A paragraph 5.6.3.2 refers.)

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

FUNCTIONAL CONFIGURATION AUDIT (FCA)

- Documents to be reviewed:
 1. Software Test Report.
 2. Software User's Manual.
 3. Computer Software Operator's Manual.
- Delete reference to Computer System Diagnostic Manual as it has been eliminated from DOD-STD-2167A.
- Add a requirement for the contractor to submit the final draft product specification for the configuration item(s) to be audited to the contracting agency for review prior to the Functional Configuration Audit (FCA).
- General contractor responsibilities are listed in paragraph 70.3 (and related subparagraphs) of Appendix G, MIL-STD-1521B and should be included in the SOW.
- SOW should make provisions for the contractor to provide to the FCA Team all related testing information listed in paragraph 70.4.2, Appendix G, MIL-STD-1521B.
- Provisions in the SOW should require the contractor to accomplish sufficient analysis or simulation to ensure configuration item performance where performance parameters could not be completely verified during testing.
- Delete paragraph 70.4.7 when the development effort consists of software only. This paragraph deals solely with HWCIs.
- Contractor should be required to develop a checklist which identifies documentation, hardware and computer software to be available and tasks to be accomplished at the FCA for the configuration item.
- Additional requirements, specifically related to CSCIs, should be added to the SOW as listed in paragraph 70.4.12, Appendix G, MIL-STD-1521B.
- Any other requirements should be annotated, as appropriate dependent on the specific nature or function of the project being audited.

PHYSICAL CONFIGURATION AUDIT (PCA)

- Documents to be Reviewed:
 1. Software Product Specification.
 2. Version Description Document.
 3. Software User's Manual.
 4. Computer Software Operator's Manual.
- Delete Software Programmer's Manual and Firmware Support Manual as these documents will not be delivered for this category of FNOC software projects.
- Make provisions in the SOW that require the contractor to provide all quality control records to ensure the as-coded configuration is reflected by this documentation.
- Delete reference to the Computer System Diagnostic Manual, Software Programmer's Manual and Firmware Support Manual in paragraph 80.1.4. These documents are not required to be delivered in this category of FNOC software projects.
- Contractor must provide all data pertinent to the configuration item being audited at the time of the audit. Required information should include that listed in paragraphs 80.3.1a and b and 80.3.2.1-1, 80.3.3a-e with the following exceptions:
 1. Delete 80.3.1.a(1), (4) & (5), 80.3.3.d & e, and 80.3.2.e,f,g & h if the development effort deals solely with software. These requirements address items of hardware.
 2. Delete reference to Computer System Diagnostic Manual and Firmware Support Manual in paragraph 80.3.2.i. These items have been eliminated or are not required to be delivered respectively.
- Delete 80.4.1, 80.4.2, 80.4.3, and its subparagraphs, 80.4.5, 80.4.6, 80.4.7, and 80.4.8 if the development consists strictly of software. These paragraphs strictly address HWCIs.
- Ensure all actions listed in paragraph 80.4.10, Appendix H, MIL-STD-1251B are performed for each CSCI.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

- Any other requirements, actions, or information should be included, as appropriate, dependent on the specific nature of the project being audited.

APPENDIX B

INTERMEDIATE PROJECTS

Much difficulty was encountered in trying to establish category criteria applicable to "Intermediate" projects. Because of its position between the "Major" and "Minor" project categories and the number of possible requirements combinations, it became too complex to establish definitive criterion values appropriate to an "Intermediate" project category. Every time an attempt was made to assign a value to an "Intermediate" criterion, FNOC personnel, in an effort to ensure all possible situations would be addressed, always ended up assigning a value that ultimately placed it in the "Major" category. As a result, no "skeletonized," tailored version of the standard was developed. Once projects are determined to belong in this category (because it did not meet the criteria of either the "Major" or "Minor" categories), each project will have to be judged, reviewed, analyzed, and tailored according to its individual requirements and needs. The general technique used will be to start with the requirements of a "Major" project and work down, deleting those requirements that aren't applicable to the specific project under consideration.

There are, however, some general, "if-then" guidelines that could be of some benefit to a developer/contractor when dealing with a project within this category:

1. Obviously, lines of code will fall somewhere between the values established for "Major" and "Minor" categories (greater than 2000 and less than 20,000).
2. The developer/contractor should assume that at least those requirements, DIDs and reviews that apply to "Minor" projects are also applicable to those projects in the "Intermediate" category.
3. If formal qualification and/or system-level testing is required, then the developer/contractor should review and tailor the Software Test Plan (DI-MCCR-80015A), and Software Test Report (DI-MCCR-80017A) DIDs.
4. If there are interfaces to other systems, then the developer/contractor should review and tailor the Interface Requirements Specification (DI-MCCR-80026A) and Interface Design Document (DI-MCCR-80027A) DIDs.
5. If there are hardware and/or firmware specifications involved, then the contractor should review and tailor the System/Segment Specification (DI-CMAN-80008A), System/Segment Design Document (DI-CMAN-80534) and/or Firmware Support Manual (DI-MCCR-80022A) DIDs.
6. In addition, the developer/contractor should review and specify requirements for any of the applicable reviews that may be involved due to the possible addition of any of these DIDs. These reviews are the System Requirements Review, System Design Review, and the Test Readiness Review.

These guidelines provide a reference point from which the developer/contractor may begin his efforts. To establish more detailed/tailored requirements and standards would be impractical. Therefore, one must simply consider the individual requirements, products, and environment of a specific project within this category and tailor the standard and associated DIDs accordingly.

APPENDIX C

MINOR PROJECTS

This appendix represents the tailored version of DOD-STD-2167A for a FNOC "Minor" project. In keeping with the format of Appendix A for "Major" projects, this appendix is divided into two parts. The first part represents the tailored DOD-STD-2167A standards requirements utilizing the automated tailoring software package "Tailor" and contains five reports:

1. Project Description Answers.
2. Quick-Tailor Selections.
3. Statement of Work Report.
4. Detailed Status Report.
5. Action Item List.

The second part contains the tailored set of DIDs, reviews, and audits (MIL-STD-1521B) appropriate to this set of tailored standards requirements and applicable to a "Minor" project. It should be noted that this tailored version represents only a skeletonized package applicable to a general category of software projects and must be further tailored/refined to meet the individual requirements of a specific project within this category. Of particular note are items 13 through 16 of the Action Item List Report. These items (shell requirements) remain project dependent

and must be specifically stated in a contract/development plan and dealt with during the refinement process for that particular project.

NOTE: The Physical and Functional Configuration Audits were included as requirements for a "Minor" project only as an effort to ensure maximum coverage. It is the feeling of FNOC personnel that the requirement to perform these audits will be rare where "Minor" projects are concerned and were only included so that those tailoring the standard would be aware of their existence. Therefore, the tailoring of these audits will be reserved for those projects for which they are applicable and will not form a part of this appendix. The tailoring effort of the standard, associated DIDs, and reviews for "Minor" projects will consist of identifying those items that are non-applicable because of "Minor" project characteristics.

TAILOR REPORT
PROJECT DESCRIPTION ANSWERS
PROJECT: MINOR

SCREEN 1. ACTIVITIES OVERVIEW

- | | | |
|-----|-----|--|
| NO | 1. | Help define system-level requirements and design |
| YES | 2. | Define software requirements |
| YES | 3. | Design the software |
| YES | 4. | Code the software |
| YES | 5. | Perform unit/component testing |
| NO | 6. | Perform formal qualification testing |
| NO | 7. | Support system-level testing |
| YES | 8. | Perform product evaluations |
| YES | 9. | Perform configuration management |
| YES | 10. | Prepare for software use and support |
| YES | 11. | Install software at the support site |
| YES | 12. | Conduct/support formal reviews and audits |

SCREEN 2. PROJECT CHARACTERISTICS

- | | | |
|-----|-----|--|
| NO | 1. | There may be subcontractors |
| NO | 2. | There are security issues on this project |
| NO | 3. | Some or all of the software will be implemented in firmware |
| YES | 4. | The software will have user interfaces |
| NO | 5. | The contract will specify reserve memory/timing capacities |
| NO | 6. | There will be an IV&V agent |
| NO | 7. | Contractor will be responsible for CSCI-external interfaces |
| NO | 8. | Software errors could threaten personnel or property safety |
| YES | 9. | The software will run on computers with commercial/GFE manuals |
| NO | 10. | There are significant risks to project success |

SCREEN 3. SOFTWARE DEVELOPMENT PRACTICES

- | | | |
|-----|----|--|
| YES | 1. | Require 2167A project management practices |
| YES | 2. | Require 2167A software engineering practices |
| NO | 3. | Require 2167A formal qualification testing practices |
| NO | 4. | Require 2167A software product evaluation practices |
| YES | 5. | Require 2167A configuration management practices |

TAILOR REPORT
QUIK-TAILOR SELECTIONS
PROJECT: MINOR

SCREEN 1. PRODUCTS TO BE DEVELOPED

- | | | |
|-----|-----|---|
| YES | 1. | Source code |
| YES | 2. | Software Development Plan (SDP) |
| NO | 3. | System/Segment Design Document (SSDD) |
| YES | 4. | Software Requirements Specification (SRS) |
| NO | 5. | Interface Requirements Specification (IRS) |
| YES | 6. | Software Design Document (SDD) |
| NO | 7. | Interface Design Document (IDD) |
| YES | 8. | Software Product Specification (SPS) |
| NO | 9. | Software Test Plan (STP) |
| NO | 10. | Software Test Description (STD) |
| NO | 11. | Software Test Report (STR) |
| YES | 12. | Computer Resources Integrated Support Document
(CRISD) |
| NO | 13. | Computer System Operator's Manual (CSOM) |
| YES | 14. | Software User's Manual (SUM) |
| NO | 15. | Software Programmer's Manual (SPM) |
| NO | 16. | Firmware Support Manual (FSM) |
| YES | 17. | Version Description Document (VDD) |

SCREEN 2. REVIEWS AND AUDITS

- | | | |
|-----|----|--|
| NO | 1. | System Requirements Review (SRR) |
| NO | 2. | System Design Review (SDR) |
| YES | 3. | Software Specification Review (SSR) |
| YES | 4. | Preliminary Design Review (PDR) |
| YES | 5. | Critical Design Review (CDR) |
| NO | 6. | Test Readiness Review (TRR) |
| YES | 7. | Functional and Physical Configuration Audits
(FCA, PCA) |

SCREEN 3. PROJECT MANAGEMENT REQUIREMENTS

- | | | |
|-----|----|---|
| NO | 1. | Ensure subcontractor compliance with prime
contract |
| YES | 2. | Implement a corrective action process |
| YES | 3. | Prepare problem/change reports |
| YES | 4. | Internally coordinate products before delivery |
| NO | 5. | Implement risk management procedures |
| NO | 6. | Interface with the software IV&V agent |
| YES | 7. | Establish a software development library |
| NO | 8. | Implement security measures as specified in the
contract |

QUIK-TAILOR SELECTIONS (continued)

SCREEN 4. SOFTWARE ENGINEERING PRACTICES

- | | | |
|-----|-----|---|
| YES | 1. | Use systematic, documented development methods |
| YES | 2. | Establish a software engineering environment |
| NO | 3. | Analyze the system specification |
| NO | 4. | Perform safety analysis on requirements, design, and procedures |
| YES | 5. | Consider the use of non-developmental software |
| NO | 6. | Monitor and maintain required reserve processing capacity |
| YES | 7. | Organize CSCIs into CSCs, CSUs |
| YES | 8. | Establish software development files for CSCIs, CSCs, CSUs |
| YES | 9. | Document and implement design and coding standards |
| YES | 10. | Use the approved HOL or obtain approval to use another |

SCREEN 5. REQUIREMENTS FOR CONTRACTOR-INTERNAL TESTING

- | | | |
|-----|----|---|
| YES | 1. | Develop and document CSU test requirements, cases, and procedures |
| YES | 2. | Perform CSU testing; record results |
| YES | 3. | Develop and document CSC test requirements, cases, and procedures |
| YES | 4. | Perform CSC integration and testing; record results |

SCREEN 6. REQUIREMENTS FOR FORMAL QUALIFICATION TESTING

- | | | |
|----|----|--|
| NO | 1. | Establish a software test environment |
| NO | 2. | Dry run FQT procedures |
| NO | 3. | Perform FQT on the target computer |
| NO | 4. | Use test personnel independent of software engineers |

SCREEN 7. REQUIREMENTS FOR SYSTEM INTEGRATION AND TESTING

- | | | |
|----|----|---|
| NO | 1. | Support the development of system test plans and procedures |
| NO | 2. | Support system integration and testing |
| NO | 3. | Support post-test analysis and test reporting |

QUIK-TAILOR SELECTIONS (continued)

SCREEN 8. PREPARATIONS FOR SOFTWARE SUPPORT

- YES 1. Produce code regenerable and maintainable on a designated system
- YES 2. Install the code in the support environment
- YES 3. Provide training and continuing support

SCREEN 9. PRODUCT EVALUATIONS

- YES 1. Source code
- YES 2. Software Development Plan
- NO 3. System/Segment Design Document
- YES 4. Software Requirements Specification
- NO 5. Interface Requirements Specification
- YES 6. Software Design Document
- NO 7. Interface Design Document
- NO 8. Software Test Plan
- NO 9. Software Test Description
- NO 10. Software Test Report
- NO 11. CSU test requirements, cases, procedures, results
- NO 12. CSC test requirements, cases, procedures, results
- NO 13. Sample of software development files
- NO 14. Keep records of evaluations
- NO 15. Use evaluation criteria in the standard
- NO 16. Require evaluators to be independent of s/w engineers

SCREEN 10. CONFIGURATION MANAGEMENT REQUIREMENTS

- YES 1. Perform configuration identification
- YES 2. Perform configuration control
- YES 3. Perform configuration status accounting
- YES 4. Implement procedures for storage, handling, and delivery
- YES 5. Prepare ECPs and SCNs IAW DOD-STD-480, MIL-STD-481, MIL-STD-490

TAILOR REPORT
STATEMENT OF WORK REPORT
PROJECT: MINOR

The contractor shall comply with all requirements in DOD-STD-2167A, with the following exceptions:

- 4.1.4 Delete entire paragraph.
- 4.1.5 Delete entire paragraph.
- 4.1.6 Delete entire paragraph.
- 4.1.7 Delete entire paragraph.
- 4.2.2 Delete reference to security requirements.
- 4.2.3 Delete entire paragraph.
- 4.2.6 Delete reference to IRS.
- 4.2.10 Delete entire paragraph.
- 4.3 Delete entire paragraph.
- 4.3.1 Delete entire paragraph.
- 4.3.2 Delete entire paragraph.
- 4.3.3 Delete entire paragraph.
- 4.3.4 Delete entire paragraph.
- 4.4.1 Delete entire paragraph.
- 4.4.3 Delete entire paragraph.
- 4.4.4 Delete entire paragraph.
- 4.6.4.b Delete entire paragraph.
- 4.6.4.d Delete entire paragraph.
- 4.6.4.e Delete entire paragraph.
- 5.1.1 Delete entire paragraph.
- 5.1.1.1 Delete entire paragraph.
- 5.1.1.2 Delete entire paragraph.

STATEMENT OF WORK REPORT (continued)

- 5.1.2.1 Delete entire paragraph.
- 5.1.2.2 Delete entire paragraph.
- 5.1.2.4 Delete entire paragraph.
- 5.1.4 Delete reference to evaluation criteria.
- 5.1.4.b Delete entire paragraph.
- 5.1.4.d Delete entire paragraph.
- 5.1.5.b Delete entire paragraph.
- 5.1.5.d Delete entire paragraph.
- 5.2.1 Delete reference to IRS.
- 5.2.2.2 Delete entire paragraph.
- 5.2.4 Delete reference to evaluation criteria.
- 5.2.4.b Delete entire paragraph.
- 5.2.5 Delete reference to IRS.
- 5.3.2.1 Delete reference to IRS.
- 5.3.2.2 Delete entire paragraph.
- 5.3.3 Delete entire paragraph.
- 5.3.4 Delete reference to evaluation criteria.
- 5.3.4.b Delete entire paragraph.
- 5.3.4.c Delete entire paragraph.
- 5.3.4.d Delete entire paragraph.
- 5.3.5.2 Delete entire paragraph.
- 5.3.5.3 Delete entire paragraph.
- 5.4.2.2 Delete entire paragraph.
- 5.4.3 Delete entire paragraph.

STATEMENT OF WORK REPORT (continued)

- 5.4.4 Delete reference to evaluation criteria.
- 5.4.4.b Delete entire paragraph.
- 5.4.4.c Delete entire paragraph.
- 5.4.4.d Delete entire paragraph.
- 5.4.4.e Delete entire paragraph.
- 5.4.4.f Delete entire paragraph.
- 5.4.5.2 Delete entire paragraph.
- 5.4.5.3 Delete entire paragraph.
- 5.5.4 Delete reference to evaluation criteria.
- 5.5.4.b Delete entire paragraph.
- 5.5.4.c Delete entire paragraph.
- 5.5.4.d Delete entire paragraph.
- 5.6.1 Delete entire paragraph.
- 5.6.3 Delete entire paragraph.
- 5.6.3.1 Delete entire paragraph.
- 5.6.3.2 Delete entire paragraph.
- 5.6.4 Delete reference to TRR and evaluation criteria.
- 5.6.4.a Delete entire paragraph.
- 5.6.4.b Delete entire paragraph.
- 5.6.4.d Delete entire paragraph.
- 5.7.2.2 Delete entire paragraph.
- 5.7.3 Delete entire paragraph.
- 5.7.3.1 Delete entire paragraph.
- 5.7.3.2 Delete entire paragraph.
- 5.7.3.3 Delete entire paragraph.

STATEMENT OF WORK REPORT (continued)

5.7.4 Delete reference to evaluation criteria.

5.7.4.a Delete entire paragraph.

5.8.3 Delete entire paragraph.

5.8.3.1 Delete entire paragraph.

5.8.3.2 Delete entire paragraph.

5.8.3.3 Delete entire paragraph.

5.8.4 Delete reference to evaluation criteria.

D. Delete entire paragraph.

D.10.2.1 Delete entire paragraph.

D.10.2.2 Delete entire paragraph.

D.10.2.3 Delete entire paragraph.

D.10.2.4 Delete entire paragraph.

D.10.2.5 Delete entire paragraph.

D.10.2.6 Delete entire paragraph.

D.10.2.7 Delete entire paragraph.

D.10.3 Delete entire paragraph.

D.10.3.1 Delete entire paragraph.

D.10.3.2 Delete entire paragraph.

D.10.3.3 Delete entire paragraph.

D.10.3.4 Delete entire paragraph.

D.10.3.5 Delete entire paragraph.

D.10.3.6 Delete entire paragraph.

END OF STATEMENT OF WORK REPORT

TAILOR REPORT
DETAILED STATUS REPORT
PROJECT: MINOR

Para.	Status	Title/Description
4.	K	General Requirements
4.1	K	Software development management
4.1.1	K	Software development process
4.1.1.a	K	(System Requirements Analysis/Design)
4.1.1.b	K	(Software Requirements Analysis)
4.1.1.c	K	(Preliminary Design)
4.1.1.d	K	(Detailed Design)
4.1.1.e	K	(Coding and CSU Testing)
4.1.1.f	K	(CSC Integration and Testing)
4.1.1.g	K	(CSCI Testing)
4.1.1.h	K	(System Integration and Testing)
4.1.2	K	Formal reviews/audits
4.1.3	K	Software development planning
4.1.4	D	Risk management
4.1.5	D	Security
4.1.6	D	Subcontractor management
4.1.7	D	Interface with software IV&V agent(s)
4.1.8	K	Software development library
4.1.9	K	Corrective action process
4.1.9.a	K	(Implement a closed-loop process)
4.1.9.b	K	(Provide inputs to corrective action process)
4.1.9.c	K	(Classify problems by category and priority)
4.1.9.d	K	(Perform trend analysis)
4.1.9.e	K	(Evaluate corrective action taken)
4.1.10	K	Problem/change report
4.2	K	Software engineering
4.2.1	K	Software development methods
4.2.2	R	Software engineering environment
4.2.3	D	Safety analysis
4.2.4	K	Non-developmental software
4.2.5	K	Computer software organization
4.2.6	R	Traceability of requirements to design
4.2.7	K	High order language
4.2.8	K	Design and coding standards
4.2.9	K	Software development files
4.2.9.a	K	(Software development file contents)
4.2.9.b	K	(Software development file contents)
4.2.9.c	K	(Software development file contents)
4.2.9.d	K	(Software development file contents)
4.2.9.e	K	(Software development file contents)
4.2.10	D	Processing resource and reserve capacity
4.3	D	Formal qualification testing
4.3.1	D	Formal qualification test planning
4.3.2	D	Software test environment
4.3.3	D	Independence in FQT activities

DETAILED STATUS REPORT (continued)

4.3.4	D	Traceability of requirements to test cases
4.4	K	Software product evaluations
4.4.1	D	Independence in product evaluation activities
4.4.2	K	Final evaluations
4.4.3	D	Software evaluation records
4.4.4	D	Evaluation criteria
4.5	K	Software configuration management
4.5.1	K	Configuration identification
4.5.1.a	K	(Identify baseline documentation)
4.5.1.b	K	(Identify documentation and media under CM)
4.5.1.c	K	(Identify each CSCI, CSC, and CSU)
4.5.1.d	K	(Identify version, release, and change status)
4.5.1.e	K	(Identify code/documentation relationship)
4.5.1.f	K	(Identify deliverable medium contents)
4.5.2	K	Configuration control
4.5.2.a	K	(Establish Developmental Configuration(s))
4.5.2.b	K	(Maintain current copies of deliverables)
4.5.2.c	K	(Provide access to documents and code under CM)
4.5.2.d	K	(Control change to master copy of deliverables)
4.5.3	K	Configuration status accounting
4.5.3.a	K	(Provide traceability of changes)
4.5.3.b	K	(Communicate configuration status)
4.5.3.c	K	(Ensure consistency between documents and code)
4.5.4	K	Storage, handling, and delivery of media
4.5.5	K	Engineering Change Proposals
4.6	K	Transitioning to software support
4.6.1	K	Regenerable and maintainable code
4.6.2	K	Transition planning
4.6.3	K	Software transition and continuing support
4.6.4	K	Software support and operational documentation
4.6.4.a	K	(CRISD)
4.6.4.b	D	(CSOM)
4.6.4.c	K	(SUM)
4.6.4.d	D	(SPM)
4.6.4.e	D	(FSM)
5.	K	Detailed Requirements
5.1	K	System requirements analysis/design
5.1.1	D	Software development management
5.1.1.1	D	(Support the System Requirements Review (SRR))
5.1.1.2	D	(Support the System Design Review (SDR))
5.1.2	K	Software engineering
5.1.2.1	D	(Analyze preliminary system specifications)
5.1.2.2	D	(Allocate system requirements)
5.1.2.3	K	(Define preliminary engineering requirements)
5.1.2.4	D	(Define preliminary interface requirements)
5.1.3	K	(Define CSCI preliminary qualification requirements)
5.1.4	R	Software product evaluations
5.1.4.a	K	(Evaluate the SDP)
5.1.4.b	D	(Evaluate the SSDD)

DETAILED STATUS REPORT (continued)

5.1.4.c	K	(Evaluate the preliminary SRS for each CSCI)
5.1.4.d	D	(Evaluate the IRS)
5.1.5	K	Configuration management
5.1.5.a	K	(Place the SDP under configuration control)
5.1.5.b	D	(Place the SSDD under configuration control)
5.1.5.c	K	(Put prelim. SRSs under configuration control)
5.1.5.d	D	(Place prelim. IRS under configuration control)
5.2	K	Software requirements analysis
5.2.1	R	Software development management
5.2.2	K	Software engineering
5.2.2.1	K	(Define engineering requirements)
5.2.2.2	D	(Define interface requirements)
5.2.3	K	(Define a complete set of qualification rqmts)
5.2.4	R	Software product evaluations
5.2.4.a	K	(Evaluate the SRS for each CSCI)
5.2.4.b	D	(Evaluate the IRS)
5.2.5	R	Configuration management
5.3	K	Preliminary Design
5.3.1	K	Software development management
5.3.2	K	Software engineering
5.3.2.1	R	(Develop preliminary design for each CSCI)
5.3.2.2	D	(Develop prelim. design of CSCI external I/Fs)
4.3.2.3	K	(Document other essential design information)
5.3.2.4	K	(Establish CSC test requirements)
5.3.3	D	(Identify qualification tests)
5.3.4	R	Software product evaluations
5.3.4.a	K	(Evaluate the SDD for each CSCI)
5.3.4.b	D	(Evaluate the preliminary IDD)
5.3.4.d	D	(Evaluate the STP)
5.3.4.d	D	(Evaluate the CSC test requirements)
5.3.5	K	Configuration management
5.3.5.1	K	(Place SDDs into Developmental Configurations)
5.3.5.2	D	(Place the STP under configuration control)
5.3.5.3	D	(Place the IDD under configuration control)
5.4	K	Detailed Design
5.4.1	K	Software development management
5.4.2	K	Software engineering
5.4.2.1	K	(Develop detailed design for each CSCI)
5.4.2.2	D	(Develop detailed design of CSCI external I/Fs)
5.4.2.3	K	(Document other essential design information)
5.4.2.4	K	(Define CSC test cases)
5.4.2.5	K	(Define CSU test requirements and test cases)
5.4.3	D	(Identify qualification test cases)
5.4.4	R	Software product evaluations
5.4.4.a	K	(Evaluate the updated SDD)
5.4.4.b	D	(Evaluate the updated IDD)
5.4.4.c	D	(Evaluate CSC test cases)
5.4.4.d	D	(Evaluate CSU test requirements and test cases)
5.4.4.e	D	(Evaluate a percentage of CSU and CSC SDFs)
5.4.4.f	D	(Evaluate the STD)

DETAILED STATUS REPORT (continued)

5.4.5	K	Configuration management
5.4.5.1	K	(Update each Developmental Configuration)
5.4.5.2	D	(Place updated IDD under configuration control)
5.4.5.3	D	(Place each STD under configuration control)
5.5	K	Coding and CSU testing
5.5.1	K	Software development management
5.5.2	K	Software engineering
5.5.2.1	K	(Develop CSU test procedures)
5.5.2.2	K	(Code and test CSUs)
5.5.2.3	K	(Revise documents and code based on CSU tests)
5.5.2.4	K	(Develop CSC test procedures)
5.5.3	K	Formal qualification testing
5.5.4	R	Software product evaluations
5.5.4.a	K	(Evaluate the source code)
5.5.4.b	D	(Evaluate the CSC test procedures)
5.5.4.c	D	(Evaluate CSU test procedures and test results)
5.5.4.d	D	(Evaluate a percentage of updated SDFs)
5.5.5	K	Configuration management
5.5.5.1	K	(Update each Developmental Configuration)
5.5.5.2	K	(Place source code under configuration control)
5.6	K	CSC Integration and Testing
5.6.1	D	Software development management
5.6.2	K	Software engineering
5.6.2.1	K	(Conduct CSC integration and testing)
5.6.2.2	K	(Record CSC test results)
5.6.2.3	K	(Revise design documentation and code)
5.6.3	D	Formal qualification testing
5.6.3.1	D	(Develop FQT test procedures)
5.6.3.2	D	(Dry run FQT test procedures)
5.6.4	R	Software product evaluations
5.6.4.a	D	(Evaluate CSC test results)
5.6.4.b	D	(Evaluate updated STD)
5.6.4.c	K	(Evaluate updated code and design documents)
5.6.4.d	D	(Evaluate a percentage of updated SDFs)
5.6.5	K	Configuration management
5.7	K	CSCI Testing
5.7.1	K	Software development management
5.7.2	K	Software engineering
5.7.2.1	K	(Revise documentation and code based on FQT)
5.7.2.2	D	(Revise the IDD based on FQT)
5.7.2.3	K	(Produce updated source code)
5.7.2.4	K	(Produce an SPS for each CSCI)
5.7.3	D	Formal qualification testing
5.7.3.1	D	(Perform formal qualification testing)
5.7.3.2	D	(Prepare Software Test Reports)
5.7.3.3	D	(Prepare updated STD for each CSCI)
5.7.4	R	Software product evaluations
5.7.4.a	D	(Evaluate STRs)
5.7.4.b	K	(Evaluate updated code and design documents)
5.7.5	K	Configuration management

DETAILED STATUS REPORT (continued)

5.7.5.1	K	(Document the exact version of each CSCI)
5.7.5.2	K	(Dis-establish Developmental Configurations)
5.8	K	System integration and testing
5.8.1	K	Software development management
5.8.2	K	Software engineering
5.8.3	D	Formal qualification testing
5.8.3.1	D	(Support development of test documentation)
5.8.3.2	D	(Support testing activities)
5.8.3.3	D	(Support post test analysis)
5.8.4	R	Software product evaluations
5.8.5	K	Configuration management
B.	K	Appendix B--Requirements for coding standards
B.10.3.1	K	Presentation style
B.10.3.2	K	Naming
B.10.3.3	K	Restrictions on the implementation language
B.10.3.4	K	Use of language constructs and features
B.10.3.5	K	Complexity
C.	K	Appendix C--Category & priority classification
C.10.2.a	K	Classify by category--Software problem
C.10.2.b	K	Classify by category--Documentation problem
C.10.2.c	K	Classify by category--Design problem
C.10.3.a	K	Classify by priority--PRIORITY 1
C.10.3.b	K	Classify by priority--PRIORITY 2
C.10.3.c	K	Classify by priority--PRIORITY 3
C.10.3.d	K	Classify by priority--PRIORITY 4
C.10.3.e	K	Classify by priority--PRIORITY 5
D.	D	Appendix D--Evaluation criteria
D.10.2.1	D	Internal consistency
D.10.2.2	D	Understandability
D.10.2.3	D	Traceability to indicated documents
D.10.2.4	D	Consistency with indicated documents
D.10.2.5	D	Appropriate techniques
D.10.2.6	D	Appropriate allocation of sizing and timing
D.10.2.7	D	Adequate test coverage of requirements
D.10.3	D	Additional criteria
D.10.3.1	D	Adequacy of quality factors
D.10.3.2	D	Testability of requirements
D.10.3.3	D	Consistency between data definition and use
D.10.3.4	D	Adequacy of test cases and test procedures
D.10.3.5	D	Completeness of testing
D.10.3.6	D	Completeness of retesting

TAILOR REPORT
ACTION ITEM LIST
PROJECT: MINOR

1. Ensure that the software is required to be delivered as a contract line item.
2. Tailor the Software Development Plan (SDP) DID (DI-MCCR-80030A) for your project and require delivery of the SDP as a CDRL item.
3. Tailor the Software Requirements Specification (SRS) DID (DI-MCCR-80025A) for your project and require delivery of the SRSS as CDRL items.
4. Tailor the Software Design Document (SDD) DID (DI-MCCR-80012A) for your project and require delivery of the SDDs as CDRL items.
5. Tailor the Software Product Specification (SPS) DID (DI-MCCR-80029A) for your project and require delivery of the SPSS as CDRL items.
6. Tailor the Computer Resources Integrated Support Document (CRISD) DID (DI-MCCR-80024A) for your project and require delivery of the CRISD as a CDRL item.
7. Tailor the Software User's Manual (SUM) DID (DI-MCCR-80019A) for your project and require delivery of the SUMs as CDRL items.
8. Tailor the Version Description Document (VDD) DID (DI-MCCR-80013A) for your project and require delivery of the VDDs as CDRL items.
9. Put MIL-STD-1521B on contract and tailor its Software Specification Review (SSR) requirements for your project.
10. Put MIL-STD-1521B on contract and tailor its Preliminary Design Review (PDR) requirements for your project.
11. Put MIL-STD-1521B on contract and tailor its Critical Design Review (CDR) requirements for your project.
12. Specify in the Statement of Work how the contractor is to support the Functional Configuration Audit and Physical Configuration Audit (FCA/PCA). Guidance on FCA and PCA is provided in MIL-STD-1521B.

ACTION ITEM LIST (continued)

13. Specify the required high order language in the contract, if desired.
14. Specify in the Statement of Work the support environment computer system on which the delivered software must be regenerable and maintainable.
15. Specify in the Statement of Work the contractor's tasks related to installation and checkout in the support environment.
16. Specify in the Statement of Work the contractor's tasks related to training of support agency personnel and continuing support after delivery of the software.

END OF ACTION ITEM LIST

DID TAILORING DECISIONS
PROJECT: MINOR

CDRL ITEM: Software Development Plan DID (DI-MCCR-80030A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 10.2.5.1.1 - Delete reference to security issues; there are no security issues associated with "Minor" projects.
- 10.2.5.2.3 - Delete reference to firmware; no software will be implemented in firmware.
- 10.2.5.3 - Delete paragraph; there are no significant risks to project success where "Minor" projects are concerned.
- 10.2.5.4 - Delete paragraph; there are no security issues associated with "Minor" projects.
- 10.2.5.5 - Delete paragraph; there are no associate contractors or subcontractors involved in "Minor" projects.
- 10.2.5.6 - Delete paragraph; there will be no IV&V agents for "Minor" projects.
- 10.2.6.1.3 - Delete reference to firmware; no software will be implemented in firmware.
- 10.2.6.1.3.1 - Delete reference to classified processing and security issues; there are no security issues associated with "Minor" projects.
- 10.2.6.1.3.2 - Delete reference to interfacing equipment and firmware; there will be no interfacing with other equipment and no software will be implemented in firmware where "Minor" projects are concerned.
- 10.2.7 - Delete paragraph and all subsequent subparagraphs; no formal qualification testing is required for "Minor" projects.
- 10.2.8 - Delete paragraph and all subsequent subparagraphs; DOD-STD-2167A software product evaluation practices are not

DID TAILORING DECISIONS (continued)

required. Contractor/developer software product evaluation practices will suffice.

- 10.2.8.3 - Delete paragraph; there are no subcontractors involved in "Minor" projects.
- 10.2.8.4 - Delete paragraph; no software product evaluation records are required to be kept.
- 10.2.8.5,
10.2.8.5.1 - Delete paragraphs; contractors/developer software product evaluation practices will suffice.
- 10.2.9.5 - Delete paragraph; this tailored version assumes that no configuration audits will be required. If so, then reincorporate paragraph.

CDRL ITEM: Software Requirements Specification DID
(DI-MCCR-80025A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 7.3 - Delete reference to System/Segment Specification as this DID will not be associated with "Minor" projects.
- 10.1.5 - Delete reference to System/Segment Specification as this DID will not be associated with "Minor" projects.
- 10.1.5.1 - Delete paragraph; there will be no interfaces to other systems where "Minor" projects are concerned.
- 10.1.5.4.b - Delete subparagraph; there will be no interfaces to other systems where "Minor" projects are concerned.
- 10.1.5.5.1 - Delete reference to safety limits, there are no safety issues associated with "Minor" projects.
- 10.1.5.6 - Delete paragraph; there are no sizing and timing requirements associated with "Minor" projects.

DID TAILORING DECISIONS (continued)

- 10.1.5.7 - Delete paragraph; there are no safety requirements associated with "Minor" projects.
- 10.1.5.8 - Delete paragraph; there are no security requirements associated with "Minor" projects.
- 10.1.5.12 - Delete reference to System/Segment Specification as this DID will not be associated with "Minor" projects.
- 10.1.6.2.d - Delete reference to system level testing as this level of testing will not be performed for "Minor" projects.

CDRL ITEM: Software Design Document DID (DI-MCCR-80012A)

TAILORING ENTRY: All DID paragraphs apply with the following exceptions:

- 3.2 - Delete reference to Physical Configuration Audit as FNOC personnel do not feel that these audits will be required with regard to "Minor" projects. If this applies to particular project then reincorporate reference.
- 10.1.5.1 - Delete reference to external interfaces; there will be no interfaces to other systems where "Minor" projects are concerned.
- 10.1.5.1.3 - Delete paragraph; there are no memory or timing capacity requirements for "Minor" projects.
- 10.1.6.1.2.2.h - Delete reference to timing relationships as these are not required for "Minor" projects.
- 10.1.7.6 - Delete paragraphs; there are no interfaces to other systems where "Minor" projects are concerned.

CDRL ITEM: Software Product Specification DID (DI-MCCR-80029A)

DID TAILORING DECISIONS (continued)

TAILORING ENTRY: All DID paragraphs apply with the following exception:

- 10.1.5.4 - Delete paragraph; it is not required that "Minor" projects specify reserve memory or timing capacities.

CDRL ITEM: Computer Resources Integrated Support Document DID (DI-MCCR-80024A)

TAILORING ENTRY: No entry--All DID paragraphs apply.

CDRL ITEM: Software User's Manual DID (DI-MCCR-80019A).

TAILORING ENTRY: All DID paragraphs apply with the following exception:

- 10.1.5 - Delete reference to the Computer System Operator's Manual DID as this is not required to be delivered for "Minor" projects.

CDRL ITEM: Version Description Document DID (DI-MCCR-80013A)

TAILORING ENTRY: All DID paragraphs apply with the following exception:

- 10.2.5.6 - Delete reference to interfaces with other systems; there will be no interfaces with other systems where "Minor" projects are concerned.

The following DIDs are not required to be delivered as CDRL ITEMS for "Minor" projects:

1. System/Segment Specification DID (DI-CMAN-80008A).
2. Software Test Plan DID (DI-MCCR-80014A).
3. Software Test Description DID (DI-MCCR-80015A).
4. Software Test Report DID (DI-MCCR-80017A).
5. Computer System Operator's Manual DID (DI-MCCR-80018A).
6. Software Programmer's Manual DID (DI-MCCR-80021A).

DID TAILORING DECISIONS (continued)

7. Firmware Support Manual DID (DI-MCCR-80022A).
8. Interface Requirements Specification DID (DI-MCCR-80026A).
9. Interface Design Document DID (DI-MCCR-80027A).
10. System/Segment Design Document DID (DI-CMAN-80534).

MIL-STD-1521B (REVIEWS AND AUDITS)
TAILORING DECISIONS
PROJECT: MINOR

NOTE: These tailoring decisions are a result of a personal analysis of the various reviews/audits in MIL-STD-1521B by the author and from guidance contained in MIL-HDBK-287, Appendix B.

SOFTWARE SPECIFICATION REVIEW (SSR)

- Documents to be reviewed:
 1. Software Requirements Specification
- Paragraph 30.1: Delete references to the Interface Requirements Specification and Operational Concept Document (now replaced by the System/Segment Design Document). These DIDs are not required deliverables for a "Minor" project and therefore are not applicable.
- Replace paragraph 30.2.A-C with: "a. The information contained in the Software Requirements Specification, as tailored."
- Paragraph 30.2.b: Delete reference to timing and storage requirements. It is not required for "Minor" projects to specify reserve memory or timing capacities.
- Paragraph 30.2.d: Delete reference to external interfaces as "Minor" projects will not interface with other systems.

All other paragraphs and requirements apply.

PRELIMINARY DESIGN REVIEW (PDR)

- Documents to be reviewed:
 1. Software Design Document.
 2. Computer Resources Integrated Support Document.
 3. Software User's Manual.
- Paragraph 40.1: Delete references to Interface Design Document, Software Test Plan and Computer Software User's Manual as these DIDs are not required deliverables for "Minor" projects.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

- Paragraph 40.1: Delete reference to the Draft Hardware Product Specification and HWCi Test Plan as "Minor" projects do not include hardware specifications and/or changes.
- Paragraph 40.1: Replace "Software Top Level Design Document," "Software Detailed Design Document," and "Database Design Document" with "Software Design Document." These three documents are now combined and incorporated into the Software Design Document of DOD-STD-2167A.
- Paragraphs 40.2.1.a-ae, 40.3.1, 40.5.5, 40.5.7, 40.5.8, 40.5.9, 40.6.2, 40.6.6, 40.6.8, 40.6.9, 40.9.1-3, 40.10 and all subparagraphs, 40.12 and all subparagraphs, 40.14 and all subparagraphs, 40.15 and all subparagraphs, 40.16 and all subparagraphs, and 40.19.1-3: Delete these paragraphs as "Minor" projects do not include HWCIs, firmware and/or hardware specifications/changes.
- Paragraphs 40.5.1, 40.6.1, 40.7.1, 40.7.2, 40.13.3, and 40.13.4: Delete all reference to HWCIs, hardware development and interface documents as these are not included in "Minor" projects.
- Paragraph 40.8 and all subparagraphs: Delete this paragraph and all related subparagraphs as "Minor" projects do not include safety requirements.
- All applicable paragraphs: substitute "CSC" for "Top Level CSC (TLCSC)" and "Low Level CSC (LLCSC)" to make MIL-STD-1521B consistent with DOD-STD-2167A.
- All applicable paragraphs: Substitute "CSU" for "unit" to make MIL-STD-1521B consistent with DOD-STD-2167A.
- Paragraph 40.2.2.a: Delete reference to Interface Requirement Specification(s) as "Minor" projects have no interfaces to other systems.
- Paragraph 40.2.2.b: Delete paragraph. There are no reserve memory/timing capacity requirements on "Minor" projects.
- Paragraph 40.2.2.e: Delete paragraph. There are no security requirements/issues associated with "Minor" projects.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

- Paragraph 40.2.2.m, 40.2.3.m: Delete reference to firmware. Software will not be associated or implemented in firmware for "Minor" projects.
- Paragraph 40.2.2.n: Delete reference to Computer Software Operator's Manual and Computer System Diagnostic Manual as these will neither be delivered in "Minor" projects and no longer exists respectively.
- Paragraph 40.3.2.a, 40.3.2.b: Delete reference to external interfaces since no "Minor" projects will interface with other systems.
- Paragraph 40.3.2.f: Delete paragraph. There are no requirements for system and/or nuclear safety where "Minor" projects are concerned.
- Paragraph 40.4: Delete paragraph. There are no requirements for HWCI design compliance with electromagnetic compatibility/interference requirements where "Minor" projects are concerned.
- Paragraph 4.5.11: Delete paragraph. There are no subcontractors associated with "Minor" projects.
- Paragraph 40.6.11: Delete reference to HWCIs. There are no HWCIs associated with "Minor" projects.
- Paragraph 40.7.3. Delete paragraph. System/Segment Specification and Interface Requirements Specification(s) are not delivered where "Minor" projects are concerned.
- Paragraph 40.13.1: Delete reference to System/Segment Specification, HWCI development and Interface Requirements Specifications. These are not associated/applicable to "Minor" projects.
- Paragraph 40.18, 40.18.1, 40.18.2: Delete paragraphs. These deal with hardware/physical equipment and are not applicable to "Minor" projects.

CRITICAL DESIGN REVIEW (CDR)

- Documents to be reviewed:
 1. Software Design Document.
 2. Computer Resources Integrated Support Document.
 3. Software User's Manual.
- Paragraph 50.1: Delete reference to Draft Hardware Product Specification, Interface Design Document, Hardware Development Specification, Computer Software Operator's Manual, Software Programmer's Manual and Firmware Support Manual. These items are not required deliverables for "Minor" projects.
- Paragraph 50.1: Delete reference to the Computer System Diagnostic Manual as this document has been eliminated for DOD-STD-2167A.
- All applicable paragraphs: Substitute "Software Design Document" for "Software Top Level Design Document," "Software Detailed Design Document" and "Data Base Design Document." These documents have been combined and incorporated into the Software Design Document of DOD-STD-2167A.
- All applicable paragraphs: Delete all reference to "Top Level CSC" and "Lower Level CSC" and substitute "CSC." This makes MIL-STD-1521B compatible with DOD-STD-2167A.
- All applicable paragraphs: Delete all reference to "unit" and substitute "CSU." This too lends compatibility between MIL-STD-1521B and DOD-STD-2167A.
- Paragraphs 50.1.1, 50.2.1.a-m, 50.3.1, 50.3.1.1.a-j, 50.4.a-c, 50.6.5, 50.9.1-3, 50.10 and all subparagraphs (except 50.10.3.e), 50.12 and all subparagraphs, 50.13.5, 50.13.6, 50.15 and all subparagraphs, 50.16 and all subparagraphs, and 50.18 and all subparagraphs: Delete paragraphs. Paragraphs deal strictly in firmware and HWCIs, and there are no firmware and/or HWCIs specifications/changes associated with "Minor" projects.
- Paragraph 50.1.2.2.a: Delete paragraph. Sizing and timing capacities/data are not required for "Minor" projects.

MIL-STD-1521B (REVIEWS AND AUDITS) (continued)

- Paragraph 50.2.2.a: Delete reference to Interface Design Document. This document is not a required deliverable for "Minor" projects.
- Paragraph 50.2.2.c,e,f: Delete paragraphs. These items are not required deliverables for a "Minor" project.
- Paragraph 50.3.2.c: Delete reference to timing, sizing and storage requirements. There are no requirements to specify reserve memory and/or timing capacities in "Minor" projects.
- Paragraphs 50.5.1, 50.5.5, 50.5.7, 50.5.8, 50.6.1, 50.6.3, 50.7.1, 50.7.4, 50.13.3, and 50.13.4: Delete all reference to HWCIs, hardware specifications and/or external interface specifications. These items will not be included/involved in "Minor" projects.
- Paragraphs 50.7.4.e and 50.8 and all subparagraphs: Delete paragraphs as there are no safety issues/requirements in "Minor" projects.

APPENDIX D

SOFTWARE MAINTENANCE

DOD-STD-2167A does not cover the vital and increasingly important area of software maintenance. The only remote reference to software maintenance in DOD-STD-2167A comes from paragraph 4.6.1, General Requirements, where it includes maintenance as part of some general software support concept/environment to be documented in deliverables such as the Computer Resources Integrated Support Document (CRISD), Computer System Operator's Manual (CSOM), and the Software User's Manual (SUM). It states, "The contractor shall provide to the contracting agency deliverable code that can be regenerated and maintained using commercially available, government owned, or contractually deliverable support software and hardware that has been identified by the contracting agency." (DOD-STD-2167A, 1988, p. 118) It does, however, recommend several actions to be performed that can be related to increased software maintainability:

1. Include the user and support communities in requirements definitions, formal reviews and audits, and other appropriate activities throughout the software development process.
2. Require early and repeated deliveries of user and support documentation to allow time for adequate review and revision.
3. Work closely with the designated support agency to determine what elements of the contractor's software

need to be designated as deliverables in order to support the software. (MIL-HDBK-287, 1989, p. 13)

These actions are not adequate to maintain the usability and reliability of a software product.

Software maintenance is defined as the "modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment." (ANSI/IEEE 729, 1983) Maintainability, on the other hand, is "the ease with which a software system can be corrected when errors or deficiencies occur, and can be expanded or contracted to satisfy new requirements." (Schneidewind, 1987, p. 303) These definitions are conventional. Changes are not confined to the post-delivery phase, but are made during all life cycle phases. Modifications to software must be managed effectively during the entire life of the software. Software maintenance represents 60-70% of the total cost of software which runs into the tens of billions of dollars each year (NBS Special Publication 500-130, 1985, pp. 5-6). Software maintenance is a process that starts when determining user requirements and never ends. It is a process of change management and is not only concerned with changes to the software, but includes changes to the associated documentation as well. (Schneidewind, 1989, p. 6) Strong effective management of the entire process is needed to control the effects and associated costs of software maintenance. The

level of discipline invoked in the software maintenance process, correlates to the quality of resultant software.

While software systems vary in function, type, and size, many of the functions performed under software maintenance are universal in scope. (NBS Special Publication 500-106, 1983, p. 2) Although no standards exist for maintenance, management guides are available from the National Bureau of Standards, which provide methodologies and procedures for conducting an effective maintenance program. Suggested guidelines to follow are:

1. Develop a software maintenance plan/policy. A software maintenance policy should employ standards which describe, in broad terms, the responsibilities, authorities, functions, and operations of the software maintenance organization. It must specifically address the need and justification for changes, the responsibility for making the changes, the change controls and procedures, and the use of modern programming practices, techniques and tools. It should describe management's role in software maintenance and define the process and procedures for controlling changes to the software after a baseline has been established. (NBS Special Publication 500-106, 1983, pp. 51-52)
2. Ensure that design documentation is available to maintainers at design time, and that documentation guidance is in place that supports and results in complete and accurate documentation. Too often, the maintainer receives little, no, conflicting or incorrect communication from those who have previously handled the system. Thus, the problems of software maintenance begin simply with a breakdown in communication between those involved with ensuring that the system does what it is supposed to do. On many occasions the only source of information available to the maintainer may be the documentation and the code. Thus, good documentation is the only reliable means for good communication (NBS Special Publication 500-106, 1983, p. 15). As a minimum the following should be standard documentation for supporting maintenance: requirements specification,

design specification, program listing, test plan, and test results (Schneidewind, 1989, p. 15).

3. Ensure that guidance exists to assist in the determination of when to rebuild/redesign a system vice maintaining it. A major concern of managers and software engineers is how to determine whether a system is hopelessly flawed or whether it can be successfully maintained. While there are no absolute rules on when to rebuild rather than maintain the existing system, some of the factors/characteristics to consider in weighing a decision of this type can be found in Table 4 of the NBS Special Publication 500-106. These characteristics are meant to be general "rules of thumb" which can assist a manager in understanding the problems involved with maintaining an existing system and in deciding whether or not it has outlived its usefulness to the organization (NBS Special Publication 500-106, 1983, pp. 20-25).
4. Develop formal policies for controlling software changes. Regardless of the type of software maintenance (perfective, adaptive, or corrective), the key to controlling software maintenance is to organize it as a visible, discrete function and, to the extent possible, plan for it. Software changes must be managed and controlled. (NBS Special Publication 500-106, 1983, p. 26) All change requests should be formal and written, and changes should be limited only to those requests that have been reviewed and approved. Changes should be approved only if their benefits outweigh the cost of making them. No change should be implemented without careful consideration of its ramifications. All maintenance/changes should be scheduled, and documentation and coding standards enforced. Most importantly, to the extent possible, plan for preventive maintenance.
5. Ensure that the source code employed in changes supports maintainability. Many techniques and aids exist to assist the system developer, but there has been little emphasis on aids for the maintainer. However, since the processes which occur in the maintenance phase are similar to those of the development phase, there is considerable overlap in the applicability of the development aids in the maintenance environment (NBS Special Publication 500-106, 1983, p. 31). These ideas/techniques can improve the maintainability of the system and can therefore make future maintenance efforts easier. Some of these techniques are:

- A. Use a single high level language.
- B. Use standard coding conventions (variable names, structures, formats, etc.).
- C. Use modular structures.
- D. Use standard data definitions.
- E. Use meaningful comments in the code.
- F. Use only standard compiler options.

The point is to not only develop systems with maintenance in mind, but maintain them with future maintenance in mind.

(NBS Special Publication 500-130, 1985, p. 25)

- 6. Ensure that testing standards and procedures are established to verify and validate the correctness of changes. Testing is a critical component of software maintenance. Testing is done to find errors, not to prove that errors do not exist. Regression testing and system testing should be performed in addition to any unit, component and integration testing. Regression testing will detect the introduction of any "ripple effects" in the system and verify that the maintenance modifications have preserved the functionality of the system. System testing verifies that the system produces the same results and specifications. Overall, testing helps to provide assurance that the activities of software maintenance have been performed correctly.
- 7. Ensure that an appropriate set of quality metrics has been established. A quality metric is a quantitative measure of the degree to which software possesses a given attribute that affects its quality. In order to manage software change it is desirable to measure the effects of change. This is accomplished with quality metrics. (Schneidewind, 1989, p. 10) Basically, they aid in understanding how software changes affect the overall software system from a maintainability standpoint.
- 8. Establish policy to determine the feasibility and appropriateness of reusing software/existing code. Research has shown that the quality of software deteriorates when the only elements of the software that are reused are the source and object code. By effectively reusing the requirements, specifications,

design, documentation, test data, and other elements on which the code is based, the quality of the software can be maintained, or even enhanced during repeated modifications which occur after implementation. (NBS Special Publication 500-130, 1985, p. 9) Even if the code itself is not elegant and possibly not reusable, a study of the specifications and identification of the most frequently used components could reveal a set of generic classes of algorithms and functions which could be usable in future systems. (Schneidewind, 1989, p. 30)

Prevalent throughout this discussion on software maintenance were three recurring principles:

1. In order to maintain control over the software process it is important that software maintenance be anticipated and planned for.
2. Software maintenance must be performed in a structured and controlled manner.
3. Systems must not only be developed with maintenance in mind, they must be maintained with maintainability in mind.

APPENDIX E

SOFTWARE QUALITY ASSURANCE

Quality Assurance is a planned and systematic pattern of actions necessary to provide adequate confidence that the item or product conforms to established technical requirements (ANSI/IEEE STD 730, 1984, p. 9). Software Quality Assurance should be fully integrated with the activities and procedures of any software development project. Any software quality program should be implemented in accordance with a carefully prepared and documented "plan"; one that was reviewed and approved by the division head/representative of each unit of the organization having responsibilities defined within the plan. The plan should be a comprehensive document covering everything from defining responsibility for Software Quality Assurance through delineating the evaluation criteria governing each phase and product of the software development effort. It should even expound upon general management's role in the software quality program.

DOD-STD-2167A addresses Software Quality Assurance in requirements for software product evaluations (Paragraphs 5.x.4). Detailed requirements of DOD-STD-2167A call for the evaluation of specific deliverables (DIDs) at the completion of each software development activity within a project.

Each product/deliverable is to be evaluated against criteria specified in Figures 4-10 of the standard, as applicable to the project at hand. Default definitions for the criteria are specified in Appendix D of DOD-STD-2167A. When discrepancies are encountered, a problem/change report will be initiated (in accordance with DOD-STD-480A/MIL-STD-481A) and shall serve as input to the corrective action process. Therefore, as a result of these evaluation activities, the contractor/developer will have made an effort to ensure that the deliverable item/product is acceptable in terms of its ability to satisfy project requirements.

DOD-STD-2168, Defense System Software Quality Program, supplements DOD-STD-2167A. DOD-STD-2168 specifies requirements for a software quality program, and not only requires the evaluation of software products but also requires the evaluation of all software development activities and processes for compliance and adherence to a project's planning document. The requirements of the standard are documented by its own Data Item Description (DID), the Software Quality Program Plan (SQPP)--DI-QCIC-80572. The SQPP provides a vehicle for communicating information relevant to DOD-STD-2168 requirements.

DOD-STD-2168 and its associated DID are very comprehensive for evaluating all of the functional areas and software development activities of a software project. DOD-STD-2168, used in conjunction with DOD-STD-2167A, is more

comprehensive than the set of requirements stipulated by the IEEE Standard for software quality assurance plans (ANSI/IEEE STD 730). (That is not to say that an organization should not review this standard when establishing a software quality program.) The DOD-STD-2168 requires that the SQPP be placed under configuration and version description control prior to implementation. The standard's software quality program includes the evaluation of all software documentation, software qualification procedures, configuration management and corrective action procedures, the software development library, deliverable elements of the software engineering and test environments, and participation in all formal reviews and audits.

Although DOD-STD-2168 and its associated DID are comprehensive in their coverage of the various functional areas of a software development effort, the contents of the individual paragraphs within both are very general in the scope of their requirements and specifications. Both the standard and the DID, like DOD-STD-2167A, are designed and required to be tailored for each contract/development project. Almost all paragraphs of both the standard and the DID apply to both "Major" and "Minor" project categories at FNOC. Paragraph 1.2.2 of DOD-STD-2168 should be deleted for all "Minor" projects (no "Minor" projects have a requirement for software to be implemented in firmware) and for those "Major" projects that do not involve hardware/firmware

changes. Likewise, all reference to hardware/firmware in paragraph 5.6 should also be deleted. Paragraph 5.8 and all of its subparagraphs should be deleted with respect to all "Minor" projects as there are no requirements for subcontractor involvement in FNOC "Minor" software development efforts. All other paragraphs of DOD-STD-2168 apply. All paragraphs of the Software Quality Program Plan DID, DI-QCIC-80572, apply to both software project categories. Application of the standard and the DID, as with DOD-STD-2167A, will have to be evaluated on a case by case basis for FNOC "Intermediate" projects.

There will be no duplication or conflicts created if DOD-STD-2168 is implemented in conjunction with DOD-STD-2167A. It is the opinion of this author that DOD-STD-2168 be incorporated and utilized in every software development project regardless of its category.

APPENDIX F

SOFTWARE METHODOLOGY: PROTOTYPING

DOD-STD-2167A is designed with the flexibility to be compatible with any software development methodology. It doesn't endorse the use of any explicit default methodology. It adheres to the idea that the contractor/developer should be responsible for selecting the software development method that best supports the achievement of contract/project requirements. The only constraint imposed by DOD-STD-2167A is that the software development method must be systematic, well documented and support formal reviews and audits required by the contract/project.

Prototyping is a systems development methodology that is gaining rapid acceptance by an increasing number of businesses. Prototyping is an implementation-oriented design approach where emphasis is placed on constructing a working model (prototype) of a system as quickly as possible. Design by prototyping consolidates the definition, design, and at least part of the construction phases of the system development life cycle. Prototypes are iteratively developed and refined to meet users' requirements and, in some cases, are gradually transformed into the final system. Systems analysts, utilizing non-procedural/fourth-generation languages, can use prototyping to significantly

decrease the time required to develop an information system.
(Whitten, Bentley, Ho, 1986, pp. 166-168)

Standards are very important to a prototyping team. Without them, everything is a personal product without availability to other team members. Communication between team members requires adhering to a productive set of standards. (Boar, 1984, p. 70)

Prototyping standards need to be developed within an organizational context designed to meet a particular environment. However, there are several "generic" guidelines/principles that could be followed by any software development project team that chooses prototyping as its software development methodology:

1. Ensure that the contractor/developer has established guidelines/criteria for determining whether an application is a suitable project for prototyping or whether the use of prespecification would be more appropriate. The selection of good candidate systems for prototyping is a project sensitive one. All applications are not good candidates. All candidacy factors need to be considered before reaching a prudent conclusion. Suitability requires a balanced evaluation of a number of factors: system structure, logic structure, user characteristics, project management, application constraints, and environmental assumptions (Boar, 1984, p. 63). A good starting point is that good candidates for prototyping are those applications that are on-line/transaction processing oriented with many data elements and record relationships but few algorithmic processes. (Boar, 1984, p. 64)
2. Ensure that the contractor/developer still follows a Life Cycle approach to support the prototyping methodology. Prototyping should not be used as a shortcut to systems development. The prototyper still has to build a working model with all of the problems inherent in building any computer application. The prototyper must build a working model consisting of

some of the proposed system's required elements, records, screens, reports, and programs. To accomplish this the prototyper still needs guidelines and rules to permit efficient development like any other application developer. Although the prototype life cycle may be somewhat different from the conventional SDLC (due to its consolidation of the definition and design phases with the construction phase), even prototyping can solve the wrong problems just as any of the other more traditional design methodologies. "A life cycle (for prototyping) is necessary to insure the delivery of a malleable prototype with sufficient functionality and completeness to be representative of the ultimate system. A prototype without these attributes will result in the same thrashing and churning that is characteristic of traditional analysis methods and the anticipated benefits will be lost." (Boar, 1984, p. 60)

3. Ensure that the contractor/developer does not try to directly implement the prototype as the final system. In some cases, through successive iterations, a prototype can become the final system. However, for the most part, prototypes are "models" representing the final system. Unless that which has been prototyped is trivial or a minor extension to an existing application, there is much left to be done. The purpose of the prototype stage is to quickly establish and iteratively refine the user's basic requirements. The purpose of an actual system is to implement those requirements in the context of many other requirements. Prototyping does not deliver a magical solution to eliminating all the necessary steps and considerations necessary to create a system that will work in a production environment. Prototyping can address in an excellent manner the issues of data, functionality, and user/machine interface but does not address performance issues, sizing issues, documentation issues, or training issues. (Boar, 1984, pp. 102-103) The entire issue of whether you can directly implement a prototype is one of honestly admitting the realities of development. A prototype, by intent and design, is not an operational system. Attempting to implement such a production system will only result in new problems as the ignored issues surface. (Boar, 1984, p. 104)
4. Ensure prototyping is performed by small teams. Prototyping cannot be performed by large teams. The optimum size of a prototyping team is two people with perhaps a third member doing supplementary support

functions for multiple projects. (Boar, 1984, p. 109) A small team is necessary in order to keep overhead and miscommunication to a minimum. Communication problems grow exponentially while marginal productivity declines as team size grows. In addition, to achieve conceptual unity and maintain integrity in implementation, the number of creators must also be kept at a minimum (Boar, 1984, p. 109). A twosome will almost instinctively maintain product unity and will best be able to integrate the next piece with completed work.

5. Ensure that proper project management procedures are in place for prototyping. Prototyping is a subprocess of the greater development activity and as such comes under the control of the project management function. Prototyping does not alter the requirement to apply sound management practices to an overall project. It does, however, require some modifications. The prototyping phase requires a reasonable amount of looseness and flexibility in order to encourage the innovative nature and quickness characteristic of this phase. Formalities and bureaucracies would defeat the entire thrust of the prototyping process, discourage change, and retard the rapidity of the process. The prototyping process will self-correct itself during iteration if a major mistake is made.
6. Ensure that prototyping software exhibits those requirements that optimize the productivity of the prototyper. Most significant of these requirements are:
 - a. data dictionary driven.
 - b. structurally encourages component engineering .
 - c. enables "cutting and pasting" of new components from existing components.
 - d. provides an interactive prototyper workbench.
 - e. permits declarative specification rather than procedural specification.
 - f. automatically generates application documentation.
(Boar, 1984, p. 115)

Prototyping is a very comprehensive subject that to fully explain and decipher would require the concentration

of a separate research effort. However, an extensive discussion on prototyping is not the focal point of this thesis. Prototyping is a high productivity strategy for solving some real world problems of defining business system requirements. Its success is anchored in a simple yet elegant formula: start small, gain acceptance, evolve (Boar, 1984, p. 207).

LIST OF REFERENCES

- ANSI/IEEE Standard 729, An American National Standard IEEE Standard Glossary of Software Engineering Terminology, 1984.
- ANSI/IEEE Standard 730, IEEE Standard for Software Quality Assurance Plans, 1984.
- Boar, B.H., Application Prototyping: A Requirements Definition for the 80s, John Wiley & Sons, Inc., 1984.
- Braverman, P.H., "Yes, Folks, Standards Are a Many Splendored Thing," Computer, 12, July 1979.
- Department of Defense, DOD-STD-480A, Configuration Control--Engineering Changes, Deviations and Waivers, 12 April 1978.
- Department of Defense, DOD-STD-2167A, Defense System Software Development, 29 February 1988.
- Department of Defense, MIL-STD-481A, Configuration Control--Engineering Changes, Deviations and Waivers (Short Form), 18 October 1972.
- Department of Defense, MIL-STD-490A, Specification Practices, 4 June 1985.
- Department of Defense, MIL-STD-499A, Engineering Management, 1 May 1974.
- Department of Navy, MIL-HDBK-287, A Tailoring Guide for DOD-STD-2167A, Defense System Software Development, 21 June 1989.
- Fleet Numerical Oceanography Center, Change of Command Brochure, 1989.
- Fleet Numerical Oceanography Center, 1986 Master Plan, 1986.
- Fleet Numerical Oceanography Center, Software Improvement Program Macroplan 1.0, April 1988.
- Newport, J.P., "A Growing Gap in Software," Fortune, 28 April 1986.

- Peters, L., "A Conceptual Basis for Software Design Standards," Proceedings of the Software Engineering Standards Application Workshop, Silver Springs, MD: IEEE Computer Society Press, 1981.
- Pressman, R.S., Software Engineering: A Beginner's Guide, McGraw-Hill, Inc., 1988.
- Poston, R.M., "Software Standards," IEEE Software, 1, January 1984.
- Schneidewind, N.F., "Software Maintenance: The Need for Standardization," Proceedings of the IEEE, April 1989.
- Schneidewind, N.F., "The State of Software Maintenance," IEEE Transactions on Software Engineering, Vol. SE-13, No. 3, March 1987.
- Tausworthe, R.C., Standardized Development of Computer Software: Part II, Standards, California Institute of Technology, 1978.
- Telephone conversation between David S. Maibor, David Maibor Associates, Inc., Needham Heights, MA, and the author, 17 May 1989.
- Telephone conversation between Jane Radatz, Logicon, Inc. Strategic and Information Systems, San Diego, CA, and the author, 17 May 1989.
- United States Air Force, MIL-STD-483A, Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs, 4 June 1985.
- United States Air Force, MIL-STD-1521B, Technical Reviews and Audits for Systems, Equipments, and Computer Software, 4 June 1985.
- U.S. Department of Commerce, National Bureau of Standards, Executive Guide to Software Maintenance, NBS Special Publication 500-130, October 1985.
- U.S. Department of Commerce, National Bureau of Standards, Guidance on Software Maintenance, NBS Special Publication 500-106, December 1983.
- Whitten, J.L., Bentley, L.D., and Ho, T.I.M., Systems Analysis and Design Methods, Times Mirror/Mosby College Publishing, 1986.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Professor Barry Frew, Code 54Fw Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	2
4. Professor Tarek Abdel-Hamid, Code 54Ah Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
5. Mrs. Jeanne L. Frew, Code 008 Fleet Numerical Oceanography Center Monterey, California 93940	2
6. Curricular Officer, Code 37 Computer Technology Programs Naval Postgraduate School Monterey, California 93943-5000	1
7. LT William T. Livings, USN 1557 Willimantic Drive Virginia Beach, Virginia 23456	1