DOD-HDBK-281(NAVY)
22 OCTOBER 1984

# MILITARY HANDBOOK
# GUIDANCE FOR THE USE OF
# DOD-STD-1679A

NO DELIVERABLE DOCUMENTATION
IS REQUIRED BY THIS DOCUMENT

AREA ECRS

DEPARTMENT OF DEFENSE
WASHINGTON, D.C.

Guidance for use of DOD-STD-1679A
DOD-HDBK-281 (Navy)

1. This standardization handbook was developed by the Department of Defense in accordance with established procedures.

2. This publication was approved on 22 October 1984 for printing and inclusion in the military standardization handbook series.

3. This document provides basic and fundamental information on the use and applicability of DOD-STD-1679A for software acquisition and development. It will provide valuable information and guidance to personnel involved with the acquisition of computer software, particularly software relating to mission critical systems. This handbook is not intended to be referenced in any purchase specification except for informational purposes, nor shall it supersede any specification requirements or DOD-STD-1679A.

4. Every effort has been made to reflect the latest software engineering concepts and software development practices. The Department of Defense intends to review this Handbook periodically to ensure its completeness and currency. Users of this document are encouraged to report any errors discovered and any beneficial comments (recommendations, additions, deletions) for changes to:

Chief, Naval Material Command
ATTN: NAVMAT 08Y
Washington, D.C. 20360

by using the self-addressed Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document or by letter.

# F O R E W O R D

1. DOD-STD-1679A, Software Development, is a standard which is applicable to all Department of Defense software procurements. This Handbook contains an explanation of the requirements specified by DOD-STD-1679A. The Handbook also provides guidance for writing a Statement of Work and Specification for the procurement of software. This Handbook has been written to provide an acquisition manager (AM) and his staff, involved with the procurement of software, insight into the current interpretation of the requirements of the Standard. For personnel not familiar with the processes and procedures of software development, the Handbook will also serve as an introduction to the Standard and its significance. It will provide all parties to a software development with a common interpretation of the Standard.

2. Although the Standard was intended for use by the Department of Defense to specify requirements for the development of software by contractors, the Standard should also be invoked on software developments and life cycle support conducted within the Government. Therefore, the Handbook is a useful tool to Government personnel building software as well as to those responsible for specifying, monitoring, and accepting software.

# CONTENTS

# 1. SCOPE

1.1 Purpose. This Handbook provides a common interpretation of DOD-STD-1679A, Software Development, and information to maximize its utility. This interpretation is the result of Government experience with the Standard applied over numerous software acquisitions. Based on this experience the Handbook is intended to assist acquisition managers (AMs) in avoiding the pitfalls of previous software developments. And in so doing, the Handbook should contribute to better quality software that performs better and exhibits better maintainability characteristics.

1.2 Application. This Handbook is intended for use by any activity invoking DOD-STD-1679A either on a contractor, another Government agency, or their own staff. The Handbook is not intended to be a definitive document with a cookbook approach to software development. It is, instead, intended to assist Government personnel who must deal with the many aspects and variables of software development.

## 2. REFERENCED DOCUMENTS

### 2.1 Government documents.

2.1.1 Specifications, standards, and handbooks. Unless otherwise specified, the following specifications, standards, and handbooks of the issue listed in that issue of the Department of Defense Index of Specifications and Standard (DODISS) specified in the solicitation form a part of this handbook to the extent specified herein.

a. DOD-STD-1679A Software Development

Copies of this standard required by manufacturers in connection with specific acquisition functions should be obtained from the contracting activity or as directed by the contracting officer.

# 3. DEFINITIONS

3.1 Introduction. The definitions provided in DOD-STD-1679A are applicable for use with this Handbook. Since this Handbook is intended to be a companion document to the Standard the definitions are not restated here.

3.2 Acquisition Manager. Throughout this Handbook the acronym "AM" is used for brevity. The term is used generically to represent any person or office that is involved in the contract from the side of the customer.

3.3 Handbook. The term "Handbook" (capitalization intended) refers to this document, DOD-HDBK-281. This simplification is used to improve readability.

3.4 Standard. The term "Standard" (capitalization intended) refers to DOD-STD-1679A, Software Development. This simplification is used to improve readability.

3.5 Statement of Work. A contractual document which defines the work effort required from contractors or Government support activities. The Statement of Work will invoke and reference standards, requirements, and other documents in defining the work effort.

# 4. GENERAL STATEMENTS

4.1 Acquisition Manager (AM). The AM should have an understanding of the requirements found in the Standard. The information presented in this handbook attempts to provide the most current interpretation of these requirements. While the Handbook makes every attempt to fully discuss all aspects of software development, it is impossible to set forth here the numerous alternative methodologies and techniques of software engineering. It is the goal of this Handbook to provide proven, acceptable methods of fulfilling the requirements. The words used in the Standard are broad and sweeping as they are intended to apply to any and all Department of Defense software development efforts. The AM must use technical judgement tempered by discretion when applying the requirements to their particular program and evaluating alternatives proposed by the contractor. The Handbook is designed to help the AM make the necessary extrapolation and at the same time abide by the Standard.

4.2 Management personnel. Personnel tasked with the supervision of a software development project are usually selected for their management skills but might not have the necessary background and experience in software engineering. The handbook is designed to augment their managerial skills by providing an understanding of what is being asked of the contractor by the Standard. Neither the Standard nor the Handbook can replace the skills and judgement of experienced software engineers that are required to evaluate the contractor's proposal and his performance. If experienced software engineers are not on the staff of the AM, consideration must be given to augmenting the staff from another Government activity such as the life cycle software support activity or another contractor.

# 5. DETAILED STATEMENTS

5.1 DOD-STD-1679A. Appendix A contains a paragraph by paragraph interpretation of the Standard along with supporting rationale where appropriate. The information is based upon the experience of applying MIL-STD-1679, the predecessor of the Standard, to actual software developments. The information was derived from projects through the end of 1983. There are differences, in most cases, between MIL-STD-1679 and its revision, DOD-STD-1679A, but the differences are minor and have not changed the significance of the Standard. The information presented in Appendix A is for guidance and represents the best industrial knowledge at the time of issue of this Handbook. Technical personnel and their managers should use care if, and when, a decision is made to deviate from the guidance provided. Such deviation should be well thought out and should represent changes in the form of improvements to the required methodology and not regression to older methods which have proven less effective, or for the omission of a methodology altogether. Although the disciplines discussed in Appendix A are state-of-the-art in software development technology, many contractors have not advanced to the level of sophistication presented here and in the Standard. There is, however, always room for improvement to the techniques and methodologies in Appendix A. The information in Appendix A should provide technical personnel with sufficient understanding of the intention of the Standard and of the rationale for the requirements, that a reasoned decision can be made regarding the value of deviations.

5.2 Economics. The costs of implementing the requirements of the Standard are not known; these costs are so closely intertwined with the costs of software development that it becomes difficult, if not impossible, to separate them. All of the requirements in the Standard must be performed to some degree in any software development; the additional effort required to do it right is difficult to assess. It has been established within the software industry that over the life of a system, the driving cost is the maintenance of the software after the system is delivered and becomes operational. This would suggest that any engineering which goes into the software during development and which reduces the software's maintenance costs, is cost effective over the life of the software. This is a prime objective of the Standard. Therefore, before waiving any requirements, the impact of life cycle costs must be carefully considered. Appendix B discusses in more detail the cost considerations of implementing the Standard.

5.3 Statement of Work. The procurement documents should include a Statement of Work to define that work which is to be accomplished by the contractor. It is in the Statement of Work that DOD-STD-1679A is invoked on the contractor. Appendix C contains guidance and an

illustration of a Statement of Work to develop software which can be used as a guide in preparing a specific Statement of Work.

Another important, related component of procurement documents is the Contract Data Requirements List (CDRL), which normally utilizes DD Form 1423. The CDRL is the only appropriate (and legal) means for ordering the deliverable software documentation in a contract. For each item on the CDRL, there must be a Data Item Description (DID) supplied that provides the contractor with the format and content requirements for that document. For each item ordered on the CDRL, the contract must contain, somewhere therein, a description of the work to be performed by the contractor which will be manifested in the document. The relationship between the Standard and the DIDs is DOD-STD-1679A describes the work that the contractor must perform to develop and document the software, the DIDs identify all of the documents to be delivered under the contract.

5.4 Reserve Requirements. Minimum reserve capacities are required for designated components of the computer system. The components requiring a reserve are the computer memory, input/output channels, secondary memory (disks, tape drives, bubbles) and central processor utilization. Central processor utilization is the percentage of time the processor is being used. The Standard has an appendix which defines the methods to be used in determining the reserve of each of these components. The contractor is to design the system software in such a manner that the reserve requirements are met.

5.5 Verification and validation. The phrase "verification and validation" or "V&V" has evolved into a software management cliche'. Formal definition of these terms is generally as follows:

a. Verification - Those activities which encompass the tasks and functions needed to trace requirements all the way through the evolving stages of the software development cycle to the implemented code. The iterative process of verifying the output of each stage of the software development process against the input to that stage for requirements satisfaction.

b. Validation - The final step in the software development process wherein testing is conducted to assess the software's degree of compliance with its specified requirements. (often synonymous with "acceptance test").

"Independent verification and validation" or "IV&V" is used when these activities are performed by an independent (objective) third party.

Unfortunately, the terms "V&V" and "IV&V" are often liberalized to take on various other meanings such as software quality assurance, contract or project monitoring, project auditing, project staff support, and independent software testing. Software quality assurance as used in the Standard and this Handbook refers to those activities performed by the software developer to assure that his products are of the requisite quality. The following Section discusses contract or project monitoring by the AM. This Handbook uses V&V in accordance with its formal definition.

5.6 Contract monitoring. Once a comprehensive Statement of Work invoking the Standard is in place, the most important job the AM has is to monitor the developer's execution of the contract. Depending on the size of the software development effort and the size of the AM's staff, the AM may elect to monitor the contract with its own staff or contract an independent agency, such as the life cycle software support activity or a commercial organization to assist in performing the task. This use of a third party to support the AM is sometimes mistakingly referred to as independent verification and validation (IV&V). In any case, the monitoring function will include reviewing plans submitted by the contractor; ensuring plans have been implemented; attending reviews; conducting audits throughout the development process; reviewing documentation; recommending changes; and testing or witnessing testing. If the monitoring function is contracted out, the same element of the AM's organization that is technically responsible for the software development contract should also be responsible for the monitoring contract. This facilitates coordinating the numerous interface activities between the two contracts and allows a rapid response to the actions required of all parties concerned.

Review activity:
Navy - AS, EC, OS, SH

Preparing activity:
Navy - NM
(Project ECRS - N006)

# APPENDIX A

## DOD-STD-1679A GUIDANCE

### 10 GENERAL

10.1 Scope. This Appendix provides the detailed guidance to acquisition managers for using DOD-STD-1679A, Software Development, (the Standard) when invoked on a software development contract.

10.2 Application. This Appendix is intended as a management tool to the acquisition manager (AM), and his staff,of any project which involves the development of software.

### 20 REFERENCED DOCUMENTS

DOD-STD-1679A, Software Development

### 30 DEFINITIONS

As provided in Section 3 of the Standard.

### 40 GENERAL REQUIREMENTS

40.1 General. The following Section contains detail guidance for using the Standard. The content of Section 50 is organized in the same order as the content of the Standard. The paragraph numbers in Section 50 correspond to the paragraph numbers of the Standard to facilitate referencing between documents.

### 50 DETAILED REQUIREMENTS

## 1. SCOPE

1.1 Purpose. DOD-STD-1679A, also referred to as the Standard, has basically four purposes. First, it is a contractual tool which facilitates invoking most of the state-of-the-art software engineering disciplines used to develop software in the industry today. Second, it provides requirements which will improve the software's maintainability characteristics for life cycle support. Third, the Standard provides the basic software development practices and procedures necessary to establish a minimum level of software standardization across all software development contracts which invoke it. Finally, it provides requirements which are intended to prevent many of the contractual problems encountered in earlier software developments.

1.2 Application. The Standard applies to all software that is being developed, both operational and support, regardless of the application. It is not intended that its requirements be applied retroactively to software that has been previously developed. When the software is only one element of a larger system being developed, it is necessary to integrate the software development activities into the total system development activities and not attempt to accomplish them as a stand-alone effort.

DOD-STD-1679A specifies only the minimum work required to successfully develop software. It does not contain any requirement that would limit its applicability. For example, it is independent of the applicational use of the software and the programming language used to develop the software. As a result, the Standard is applicable to all software developments, including firmware and programs with small amounts of software.

Since it is intended for contractual use, the requirements of the Standard have been limited to the scope of a software development contract. It is required that the Standard be invoked by prime contractors on any subcontract that includes software development for his prime contract. Although DOD-STD-1679A is a contractual document, it is equally applicable to non-contractual software development as well as life cycle support of software.

## 2. REFERENCED DOCUMENTS

2.1 Issues of documents. The Standard does not make reference to any other documents.

## 3. DEFINITIONS

3.1 Introduction. This is one of the most important Sections in the Standard. A complete understanding of the meaning of these terms is necessary to understand what those terms mean in the context that they are used. The Standard has used all of these terms very consistently throughout. For example, "unit" defined in Paragraph 3.22 of the Standard, is never used as a generic term. Where used, it is intended to mean the smallest, indivisible function (lowest element on the hierarchical tree). Understanding this term provides insight into the appropriate level for performing software configuration management and software quality assurance. Units also provide the basis for software integration. In the paragraphs that follow, the definitions given in the Standard are amplified where necessary for a clearer understanding.

3.2 Code walk-through. A code walk-through is an informal technical review of the source code by a small, peer group of contractor personnel. Each unit is required to be the subject of an individual walk-through. The AM should never participate or audit the conduct of a walk-through. Having the customer present defeats the informal nature of the process which is critical to the concept. The purpose of a code walk-through is the early detection of errors in the code and potential errors in the implementation of the software design. The contractor must publish walk-through procedures that specify when, how often, who participates, their roles, materials to be used, and the actual conduct. The contractor's procedures must also provide for concisely recording the results of the walk-throughs. These records will provide evidence to the AM of the satisfactory conduct of the walk-through.

3.3 Contracting activity. This is the term used to denote the owner of the contract, the customer. Throughout this handbook the term AM, acquisition manager, is used to denote not only the contracting activity, but the project manager and his project staff.

3.4 Contractor. The contractor is that organization, possibly even an in-house organizational element, tasked with the development of the software in accordance with the requirements of the Standard.

3.5 Design walk-through. The design walk-through is similar to the code walk-through in all respects except that the design of the software is being examined. Interfaces are also carefully examined during this walk-through. The discussion of conducting a walk-through provided in Paragraph 3.2 above applies.

3.6 Development, software. This is the engineering process of creating the product software. Typically it includes detailed requirements definition, high level design, detailed design, programming and debug, integration and developmental testing, software quality assurance, and software configuration management.

3.7 Developmental Baseline. This is the definitional baseline of the software to be developed and delivered and it is internal to the contractor. It is not to be confused with the Functional, Allocated, Product, or Operational Baselines that are used in other Military Standards.

3.8 Documentation. Documentation is the physical manifestation of the software. The documentation is important to the software development process to define the various aspects of the software and to serve as the baseline for configuration management. However, documentation is even more important for the life cycle support of software. The Data Item Descriptions (DIDs) listed in Section 6 of the Standard define the format and contents of the minimum set of documents required to adequately define the software in every system.

These documents are ordered contractually, using the DIDs, on the DD Form 1423, Contract Data Requirements List (CDRL). The AM should consider providing for the delivery of the project's documentation electronically as well as on hard copy.

3.9 Error, documentation. This definition is in support of Section 5.8.4 of the Standard. Typographical errors are not normally considered documentation errors unless the meaning of the phrase is affected.

3.10 Error, intermittent. This definition is in support of Section 5.10.2.10 of the Standard.

3.11 Error, software. This definition is in support of Section 5.8.4 of the Standard. It should be noted that problems may be errors of omission as well as errors of commission. Errors of omission are the most difficult to detect.

3.12 Firmware. Firmware is merely a computer program whose operational residence is in some form of read-only memory (ROM) instead of some form of read-write memory, i.e., the program cannot be modified during execution. It is developed, tested, and documented the same as any other computer program. DOD-STD-1679A is directly applicable to the development of firmware. In the software engineering process there is nothing that should be omitted just because it will reside in ROM. On the contrary, due to its primitive and unalterable final form, additional documentation and testing are required.

3.13 Flow chart. This graphic representation tool was once thought to be an effective tool in developing and maintaining software. However, flow charts lose their usefulness rapidly because they cannot be efficiently updated to remain current with the changes that take place in the software. As used in the Standard, especially in Section 5.4.17, a flow chart is a very detailed representation of the code; usually, a one-to-one correspondence between lines of code and flow chart symbols. This term is not used to denote the higher level flow diagrams that provide an overview of information.

3.14 Graphic representations. This term is applied to a variety of software engineering tools that are more state-of-the-art than flow charts. When selecting and using a tool of this type, caution should be exercised so as not to employ it in a code dependent manner, thereby incurring the same problems as in the use of flow charts.

3.15 Parameters. The maximum use of parameters in the software being developed is required by the Standard. This is an efficient software engineering practice that facilitates the modification and change of software during its life cycle.

3.16 Patch. A patch is a temporary change that can be made in the software to serve as a trial correction to an error. If the correction is unsuccessful then the patch can be removed without any residual impact. Once the trial correction is proven successful, the equivalent change is made permanently in the source code of the program and the program is recompiled or reassembled.

3.17 Program stop. This definition is in support of the use of the term throughout the Standard. Note that a stop also occurs if the computer continues to run but error recovery routines are invoked.

3.18 Reduced capability mode. This definition is in support of the use of the term throughout the Standard.

3.19 Requirements walk-through. The requirements walk-through is similar to the code walk-through in all respects except that the requirements of the software are being examined instead of the code. The discussion provided in Paragraph 3.2 above applies except that the AM does participate to some degree in requirements walk-throughs to ensure the proper interpretation by the contractor of the official program requirements.

3.20 Software. The important point of this definition and its categories is the broad applicability of the use of DOD-STD-1679A for developing all types of software.

3.21 Software Change Proposal (SCP). This type of software change is documented according to the SCP form DID listed in Section 6 and is processed according to the procedures contained in the contractor's Software Configuration Management Plan (SCMP).

3.22 Software hierarchy. Understanding of this definition is critical on the part of both the contractor and the AM. Defining what constitutes a program is difficult due to the vast number of possibilities. This precludes an algorithm for deciding what is a program; however, in practice, case by case, it is easily done. It should be noted that there is no correlation between the number of computers and the number of "programs". Having established the number of programs, the remainder of the hierarchical decomposition is straight-forward. The definition of module is intentionally flexible in order to be useful in programs of all sizes. Any use of these hierarchical components names in the Standard that is not in strict conformance with this definition is unintentional.

3.23 Software Trouble Report (STR). This type of software report is documented according to the STR form DID listed in Section 6 of the Standard and is processed according to the procedures contained in the contractor's Software Development Plan (SDP) and SCMP.

3.24 Use of "shall", "will", "should" and "may". These terms are defined for contractual usage in other official documents and are faithfully reproduced herein.

## 4. GENERAL REQUIREMENTS

The requirements contained in Section 4 of the Standard are, as their name implies, general in nature. There is no special reason that these requirements be specific enough to be directly enforceable. The detailed, enforceable requirements are contained in Section 5.

4.1 Software development management. This Section provides an overview requirement for the Software Development Plan (SDP). See Section 5.12 of the Standard. The contractor should be required to submit a preliminary SDP as a part of his technical proposal.

4.2 Design requirements. The important requirements contained in Section 4.2 are that the design shall not exceed requirements without prior approval and that complexity of the design is to be minimized.

4.3 Software implementation. This Section contains one of the two statements requiring the use of a DoD approved High Order Language (HOL). To authorize the contractor to use any other programming language, including assembly, requires that the AM have previously obtained a waiver from the appropriate authority. Section 4.3 is also a second statement of the requirement that all software to be delivered must be capable of being generated with support software that is or will be available to the operational software's life cycle support activity.

4.4 Software quality assurance. The contractor is required to have in-place an established software quality assurance program that satisfies the Standard's stated minimum quality assurance requirements. The contractor's Software Quality Assurance Plan (SQAP) describes how this program will be applied in developing the software under the contract. See Section 5.9 of the Standard. The contractor should be required to submit a preliminary SQAP as a part of his technical proposal. The overall quality of the software product is the responsibility of the contractor's management. This responsibility does not rest with the individual or group of individuals within the contractor's organization which has been assigned the functions of monitoring the presence or absence of software quality assurance. Such a group should not control the products or the development of the products, but should only monitor and report on the compliance of the development process to all requirements, internal and contractual.

4.5 Software configuration management. The important requirement of this Section is that the contractor's software configuration management program is to take into account the longer term requirements of life cycle software configuration management. Also included is the requirement for this software configuration management program to provide for interfacing with the AM for changes controlled by the AM. These requirements are to be described in detail in the contractor's Software Configuration Management Plan (SCMP). See Section 5.11 of the Standard. The contractor should be required to submit a preliminary SCMP as a part of his technical proposal.

4.6 Subcontractor control. The contractor is responsible for ensuring that all software development subcontractors comply with the requirements of DOD-STD-1679A. This is the second statement of the requirement that the prime contractor invoke DOD-STD-1679A on his software development subcontractors. The subcontractors must in turn invoke the Standard on his software subcontractors. The procedures for enforcing subcontractor compliance with the requirements of DOD-STD-1679A must be described in the contractually delivered Software Development Plan.

4.7 Deviations and waivers. This is merely a standard contracting requirement. Note: it requires the contractor to obtain an approved deviation or waiver from the contracting activity before the act becomes an accomplished fact.

## 5. DETAILED REQUIREMENTS

5.1 Software performance requirements. This section is the description of the work that the contractor must perform to expand and refine the requirements contained in the contract specification. The physical manifestation of this work is contained in the Program Performance Specification (PPS) (see Section 6 of the Standard). This Section provides the contractual work description that supports ordering the PPS on the CDRL.

The software performance requirements specified in the PPS are a combination of prose, algorithms, and logic equations, written in non-technical, user-understandable terminology and language. Most importantly, the PPS should be limited to stating what is required and should not contain directions on how to implement any of the requirements. The PPS serves to demonstrate an understanding of the operational requirements to be satisfied by the contractor in the performance of his contract. Hence, it is important that the contractor produce a PPS that is complete, unambiguous, testable, and accurate. In order to ensure that the PPS is complete, unambiguous,

testable, and accurate, it is necessary for the AM to have software engineers and experienced personnel from the user community participate in the reviews of the document.

The PPS reflects the transition from high level operational and contractual requirements statements to the very detailed requirements that will be satisfied by the software. The refinements and expansions result in an apparent increase in the number of requirements. This is not the case. The increase in the size of the PPS is the result of a more detailed description of the high level requirements. The contractor should begin documenting the traceability of all software requirements at this point. Ultimately the contractor should be able to demonstrate traceability from the implemented code back through the original requirement in the PPS to the contractual specifications. This ensures that all requirements have been considered and that extraneous requirements have not been introduced into the product.

The contractor, in developing the PPS, analyzes all higher level requirements and is required to address all aspects of the higher level requirements. However, if all of the requirements are not going to be implemented during the software development effort, that is, the implementation of some requirements will be deferred until some later time, the contractor still should be required to accommodate these requirements in developing the PPS. This can be accomplished by "stubbing" those particular requirements in the document. The analysis and development of these requirements will be deferred to some later date, but the current design must have a place marker for these requirements reserved for future implementation. This may require that the design provide for recognition of a process, but the design will not complete implementation of the process. That is, say the basic system specifies that the future device is to be accessed by operator activation of a button, the development effort should include the processing for the button action, but at this point should merely return a message to the operator informing him that the capability is not available. This is one method of "stubbing" the capability. Initiation of the stubbing concept can be deferred to the design phase. However, this is not recommended since the system design can change dramatically as a result of the software implementation and maturation process.

The PPS effort may require that the contractor conduct various studies, analyses, visits to end users, or surveys as he deems necessary. After the background work has been completed, individual analysts begin the documentation process by drafting the various sections of the PPS. Upon completion, the author of each section and his supervisor will conduct a Requirements Walk-through. Resources permitting, the AM should participate in these walk-throughs utilizing not only software engineers but operationally oriented personnel as well. The AM's participation should be limited to ensuring that the

contractor has correctly interpreted the operational requirements to be satisfied. The AM's participation in these walk-throughs requires the proper contractual arrangements and a close, harmonious working relationship with contractor personnel. The AM's representatives at these informal reviews must be knowledgeable in the complete requirements specified in the contract as well as knowledgeable of the expectations and preferences of the end users. The participants in the review must understand that the review is intended to ensure that both the contractor and the AM arrive at a mutual understanding of the requirements. The comments of participants, while seeking to provide some assurance of completeness, should not be binding on either party. This process is merely a series of small, segmented reviews distributed over the period designated for the preparation of the performance requirements.

The AM reviews the contract's technical requirements by reviewing the PPS. The AM may obtain support for these reviews through an independent verification & validation (IV&V) organization. It is important that the PPS be completed, baselined, and approved by the AM as early in the project as is feasible. The PPS, thus, provides the joint agreement on the performance requirements that the contracted software must ultimately satisfy.

The PPS now becomes the new requirements baseline for configuration management by the AM. It is imperative that the AM control changes to the PPS once it is approved. Otherwise, the contractor can change the implementation of the requirements. It should be specified that approval of the PPS by the AM does not release the contractor from satisfying the contractually stated performance requirements. Once the software development has been completed and the contract concluded, the PPS serves as the requirements baseline for the software for the remainder of the life cycle.

5.1.1 Supporting information for software performance requirements. The contractor is required to use a multitude of sources in defining and drafting the software performance requirements. Some of these sources are system and hardware requirements, standards, operational doctrine, and tactics which affect the software of the system. These documents must be made available to the contractor by the AM. Some documents, such as standards and doctrine, may exist prior to initiation of the software development effort. However, many will be developed simultaneously with the software. The following is a brief discussion of the nature of the documents the contractor must research:

a. System performance requirements are the high level definition of the performance expected of the system (both hardware and software). This document defines the performance required of the system without mention of the methods to accomplish the requirements. Timing and size constraints are common elements in the system performance requirements.

b. System design requirements are derived from the system performance requirements. This document provides system hardware design requirements including their capabilities and limitations. The implication of these system design requirements must be factored into the system performance requirements for an integrated approach to system requirements. This document is normally produced by a supporting (in-house) laboratory or by a contractor. If a contractor is used, that corporate entity should be excluded from competing for the software development contract because of the unfair advantage it would have.

c. Equipment design requirements are the specifications for the computer hardware (main frame and peripheral) of the system under development. Additional care must be taken if the computer hardware is developed concurrent with the software. Equipment design requirements could change during development for a number of valid reasons and the result might be that the changes are not compatible with the software as developed. Therefore, it is to the AM's advantage to receive the actual computer hardware as early in software development as possible; or, better yet, use previously developed and proven hardware.

d. Interface design requirements are higher level definitions of the interdigital processor interfaces of the system that will be contained in the Interface Design Specification (IDS) (see Section 6 of the Standard). This document defines the physical interface to an external system, including the operator. The hardware equipment specification may suffice if the system is a stand-alone system.

e. Operational standards, doctrines and tactics are normally existing documents which define the manner in which the end user is expected to operate the system or the platform on which the system will be placed. This is at a higher, broader level and not at the level that instructs the use of the system. These documents may define certain procedures for accomplishing a task which affects the implementation in the software. For some systems, these documents may not exist until after the design is complete since the design will establish the doctrine, tactics, and standards.

f. System design standards are existing or prepared documents which define the standards for designing and documenting the system. Documents describing the use of Ada as a program design language (PDL) are an example of this type of standard.

g. System readiness and maintainability requirements are documents which define or specify readiness and maintainability requirements. These requirements apply to all systems of the type being developed or all systems that must operate in a particular environment.

5.1.2 Software performance analysis. This serves as an introduction to the subsequent subparagraphs which identify areas for study by the contractor to aid in his system familiarization.

5.1.2.1 Mission areas. This section defines the mission environment in which the system is expected to operate. It is incumbent upon the AM to define, in the contract document, the mission areas to be investigated and the intent of the investigation.

5.1.2.2 Functions. The contractor must define the system functions if a document containing the system design requirements has not already done so. The system functions are normally contained in the contractual specification of requirements. In the PPS these system functions are introduced in Section 1, and are defined in detail in Section 3.

5.1.2.3 Applicable documentation. The contractor identifies all the documents which define or constrain the software performance requirements. This includes, as a minimum, all the documents described in Section 5.1.1 above. This is the appropriate area for the contractor to define the abbreviations and acronyms that are used in the PPS. Unique terms used in the PPS in conjunction with the software development process should be defined.

5.1.2.4 System description. This section serves to force the contractor to acquire a working knowledge of the system hardware and how it interfaces with the software to perform systems functions. The contractor must conduct in-depth analysis of the relationships between equipments. This analysis must include descriptions of hardware interfaces with the computer at the secondary and tertiary levels, if such equipment is normally a part of the system. For instance, a radar signal may leave the radar (tertiary level), pass through a distribution switchboard (secondary level), to a video processor (primary level) before entering the computer as data. The level definitions assume that the computer is capable of controlling each of the intermediate equipments. It is the contractor's responsibility to determine how the software will interface with the equipment to satisfy the system requirements. This will result from the various analyses and investigations previously identified. In order to develop the software, a thorough understanding of the hardware is necessary.

a. Peripheral equipment identification is the specification of each piece of equipment, by brand and model (nomenclature). The contractor must provide the physical characteristics and the method of interfacing with the other equipment, including the computer. The physical characteristics are not limited to size and weight but include such things as data transfer rates, character set use, and the electronic and logical characteristics. The method of interfacing may include intermediary equipments through which the information may pass.

b. Interface identification is usually a listing of all other systems, subsystems, processors, and computer programs with which the system under development must interface. A stand-alone system has no interface identification.

5.1.2.5 Functional description. This section marks the end of the definitional portion of the PPS and the beginning of the real specification of software requirements. This section describes each system function that is to be supported by the software, including its purpose and functional design.

a. Equipment descriptions shall be included for each equipment interfacing within the system. The description shall address the requirements imposed on the software by that equipment.

b. Flow diagrams are designed to show the interrelationships of the various equipments and subprograms. The diagrams should be in the form of line drawings with an accompanying narrative description. One set of diagrams is expected to indicate the data and control paths between the various interfacing items of equipments. Another set of diagrams should show the data and control flow within the program design. These diagrams are intended to be high level functional flow diagrams. The detail flow diagrams are developed later during the design of the software. The AM may specify additional diagrams as appropriate.

c. Intersystem interface requirements are contained in the Interface Design Specification (IDS) (see Section 6). The IDS is a document which defines the physical and logical interdigital interfaces of the system under development and with all interfacing systems and subsystems. The IDS is developed in two stages. Whenever possible, the first stage, that of defining the requirements of each interface, should be finalized and baselined prior to development of the PPS. When not possible, they must be defined during the program's detail requirements definition (the PPS). The requirements of each interface are specified in individual IDS's. This effort is supported by this Section. During the design (see Section 5.2 of the Standard) phase of the software, the second stage (the detailed design of the interface) of the IDS is developed. The IDS is critical to a system when it is a subsystem of a larger system or if the system must act

in concert with another system. This Paragraph requires that the contractor understand the intersystem interfaces thoroughly. It also requires the contractor to cooperate with the other developers in the joint development of the IDS, when so tasked. If other systems which interface with the system under development are also under development, then the contractual documents must task both contractors to cooperate in the production of the IDS. This Paragraph requires cooperation among the developing contractors. This section provides the contractual work description that supports ordering the first stage of the IDS on the CDRL.

5.1.2.6 Detailed functional requirements. This section of the Standard defines the actions to be taken by the contractor in the course of developing the major portion of the PPS. The detailed system functional requirements become the basis for the software integration and developmental testing requirements (see Section 5.8). "Input-Processing-Output" type drawings are often used to help define the software functional requirements. The system functional description becomes a collection of text, tables and figures.

a. System inputs are easily itemized in tabular format with the arguments as column headers. Each system function should have an independent input table. All variables, fields, control signals, tables or any other parameter to equations or logic statements must be unique within the function or contained in the input table.

b. The text, equations, and algorithms which describe the system functional processing to be accomplished by the software are provided herein. This section may include pictorials relating to the system functional description. This section must be accurate and complete since it defines the specific requirements the software will be designed to satisfy. It is reasonable to expect utilities and other general purpose system functions to be defined only once. It is important that the AM's reviewers of this PPS Section be well versed in the organization and plans for each of the system functions.

c. The treatment of outputs is similar to the inputs, except they include tabulations of all variables, fields, control signals, or any other resultant variable from the algorithms, equations, or logic statements which are to be used external to the system function. It is advantageous to require that the contractor automate the listing of inputs and outputs. All sources in the input tables could then be readily correlated to the destinations listed in the output tables. Discrepancies can then be easily identified and corrected.

d. Special requirements identified by the contractor are addressed in this Section. Special requirements must be generated by the system functions themselves. Any special requirements identified by the contractor should be critically evaluated to insure the requirement is not excessive.

5.1.2.7 System functional verification. This is an analysis of the inputs and outputs defined above with particular concern for the factors which relate to system capacities. This analysis will also contribute to the effort required in Paragraph 5.1.3 of the Standard.

5.1.2.8 Changeability. The contractor is required to identify those requirements which his analysis indicates will be likely to change over the life cycle of the system. The purpose of this section is to force these requirements to be designed to facilitate future change and thereby reduce life cycle costs.

5.1.2.9 Adaptive parameters. The contractor is required to identify those values in equations, algorithms, and logic statements which are not modified (remain constant) during the execution of the program. These values are to be treated symbolically as system parameters in the design and implementation of the program. This practice maximizes the program's independence from changes in its environment. A change in the system's environment (change of hardware device, upgrade of equipment, change in limits) can be accommodated simply by changing the parameter's value and recompiling the program without affecting the program's logic.

5.1.3 System resources. The contractor is required to include in this section of the PPS an estimate of the computer system resources required to satisfy the software performance requirements defined in the PPS. The ultimate contractual requirement is that, as a minimum, a 20% reserve of these resources exist at the time of delivery of the system to the contracting activity (see DOD-STD-1679A). The purpose of this requirement is to ensure that there are sufficient computer system resources available subsequent to program delivery to correct errors and to make program enhancements. It is strongly recommended that a reserve capacity of 50% be maintained. The resources to be estimated include main memory, secondary storage (e.g. bulk storage, magnetic tapes, bubble memories and disks), central processor throughput and input/output channels. If multiple units of any equipment are in the system, the resources for each must be estimated. These are estimates and will vary throughout the software development process. Estimates at the PPS level are most often extremely optimistic in relationship to the final values. The contractor must include a substantial safety margin in the estimate or else the contractor's ability to satisfy this requirement is in serious jeopardy.

5.2 Software design. This section contains the description of the work that the contractor must perform in order to create both the high and detail levels of design of all software in the system. The physical manifestation of this engineering effort is contained in the Program Design Specification (PDS) (see Section 6). The PDS is a

software engineering transformation of the performance requirements contained in the PPS into the definition of the software design. There is a one-to-one correspondence between the PPS and the PDS. This section provides the contractual work description that supports ordering the PDS on the CDRL.

In creating the design of the software, the contractor is required to utilize the "top-down" design methodology. This is the first step in the required "top-down" software development methodology that pervades the entirety of DOD-STD-1679A. The top-down design methodology is based on the process of iterative refinement. That is, the software system (program) is decomposed into its major functional components (subprograms). Each subprogram is then decomposed into the next level of major functions (modules start to be identified at this level). This process is repeated successively until the smallest, indivisible function (unit) is attained. It should be noted that the number of levels of modules is dependent on the size of the program. In very large programs there may result numerous levels of modules and a total of modules in the thousands. On the other hand, very small programs may be comprised of only units.

One result of this process is the evolution of a hierarchical tree structure of units. The higher level units typically represent the higher levels of control. The lower levels contain the units which are designed to perform the detailed execution of the requirements. The top-down methodology requires that a unit at a higher level be designed and implemented before units below it in the hierarchical tree.

The contractor is required to define the software architecture. This is the highest level description of the software elements and includes how the elements will communicate and interact with each other. The architecture is very important since it defines how the software design is to be constructed. The architecture establishes how the system will operate from a software perspective. This includes such items as how units are sequenced for execution, how files are accessed, and how interrupts are handled. The contractor must also define the assumptions made in the development of that architecture. Prior to implementation of the software, the contractor is to verify to the satisfaction of the AM that the architecture has a high probability of being able to support the software system.

The entire design of the software is documented in the PDS. If any of the performance requirements have been stubbed, it will require stubbing that same area in the design. During the development of the PDS, the design of all elements of the software are to be subjected to at least one design walk-through. Walk-throughs are conducted by one or more technically qualified persons in conjunction with the author(s). The basic concept of walk-throughs is that they are to be informal reviews by peers of the author. The major benefit of walk-

throughs is the detection of errors at the time their correction is easiest and least expensive. Since walk-throughs are to be informal peer reviews, it would be a violation of the concept, and thus defeat their purpose, for the customer to be involved in walk-throughs in any manner. It is a contractor internal quality assurance function to ensure that walk-throughs are conducted properly.

The AM should participate in a formal review of the design through a process of reviews and approvals utilizing the PDS as the vehicle. Review of the design should not be held as a single or even a two-step event, but as a series of mini-reviews which will culminate in the final approval of the entire design. The early reviews should be geared toward an understanding of the detailed design, and the last review should bring it all together. Upon final approval, the PDS should be baselined and placed under formal configuration management by the customer. The PDS then forms the design baseline for the life cycle of the software in the system. The subsequent paragraphs describe the analysis and effort required to develop the PDS.

5.2.1 Supporting information for software design. The items listed in this paragraph with which the contractor is to become familiar are in addition to those listed for the development of the PPS. This list is included to further enhance the contractor's understanding of the software he is to develop.

a. System specification or design documents are intended to provide a detailed explanation of the operational requirements in software engineering terminology. These documents should exist prior to the beginning of the software development effort. The developer of these documents should be excluded from the development of the software. This can be the same developer who developed the system design requirements documents.

b. Software performance requirements should have been delineated in the Program Performance Specification (PPS) developed under Paragraph 5.1 of the Standard. If done properly, the PPS should be the only document needed to specify the software performance requirements.

c. Interface design requirements include, as a minimum, the first stage of the Interface Design Specification (IDS) developed under Paragraph 5.1.2.5.c of the Standard. The IDS at the requirements stage specified only the requirements for the interface. At the detailed design stage, specific design information of the interface is to be developed.

d. Programming reference manuals are documents which govern the use of the programming language selected for the software development effort. These documents should be identified by the AM, and the current revisions cited at the time of contract award.

e. Equipment technical manuals are the documents which define the actual system hardware. If the hardware does not exist at this early stage of the software development process, the equipment design documents discussed in Paragraph 5.1.1.c of the Standard must also be identified here. If the hardware is under concurrent development, the software developer must be included in the configuration management process for the hardware. The impact of hardware changes on the software development effort can have a significantly adverse impact on the cost and schedule of the system acquisition.

f. Specified programming standards and conventions identify and specify the standards and conventions to be followed during software development. Note that if the contract does not specify any documents, the contractor is free to choose any set of standards and conventions, including none. The life cycle software support activity is locked into perpetuating the same programming standards and conventions used by the software developer. The contractor should be required to identify in his proposal any additional standards and conventions he intends to use. If acceptable to the AM, they should be made contractually binding.

g. Specified utility/support software documents are those which describe and define the utility or support software that the AM has required the contractor to use during the software development effort. These documents and their associated computer programs must be provided to the contractor at or immediately after contract award.

h. Processor self-tests (diagnostics) are normally a product of the developer of the hardware and are supplied as GFE to the software developer. The maintainability requirements documents are similar to those documents identified in Paragraph 5.1.1.g of the Standard.

5.2.2 Software design analysis. The remainder of Section 5.2 addresses the actual design requirements of the software development effort which are to be reflected in the PDS.

5.2.2.1 Applicable documentation. The contractor is to identify and analyze all documents which define or constrain the design of the software. Included shall be all the documents described in Paragraph 5.1.1 of the Standard and in Paragraph 5.2.1 above. This is the appropriate area for the contractor to define the abbreviations and acronyms used in the PDS. Unique terms used in the PDS in conjunction with the software development process must be defined.

5.2.2.2 Functional allocation. Functional allocation is the result of the design analysis of the performance requirements. The analysis results in system functions being allocated to the individual subprograms identified by the contractor. These subprograms will be further divided into modules through the top-down software design

process. The composite of all the subprograms must satisfy the performance requirements in their entirety. Traceability to requirements in the PPS is required in the PDS to ensure that all the performance requirements have been incorporated into the design of the software.

5.2.2.3 Program functional flow. For each subprogram, a functional description of the inputs, processing, and outputs is required in the PDS. A similar description is required at the module level. Common subroutines need only to be defined once. Considerations in scheduling tasks for execution are to be addressed in this section. Functional flow diagrams are a useful tool for augmenting the information required in this section.

a. Program interrupt control provides the analysis which results in the listing of all external and internal interrupts. Each interrupt is to be fully described as to source, purpose, type, and the required response of executive control. Predicted rates of occurrences are to be estimated and described.

b. Subprogram reference control is described in terms of the control logic required to schedule for execution the tasks within the subprograms. Priority, cycle time, and duty cycle are also to be addressed for each of the tasks.

c. Special control requirements which affect the design of the executive control logic, but which are not part of the normal operational functions, are to be analyzed and provided for in the design of the software system.

5.2.2.4 Resource allocation and reserves. This is a requirement on the design of the software that traces back to the PPS requirement discussed in Paragraph 5.1.3. The storage, processing time, and input/output channels budgets should be allocated to each subprogram and, subsequently, to each module. These allocations are to be tabularized and totaled. As was recommended for the PPS, it is strongly recommended that the PDS also maintain the reserve requirement at 50%. Estimates in the PDS can be expected to be no less optimistic that those in the PPS, consequently, a safety margin should be included in order to ensure the required reserve at the time of software acceptance. Guidance for calculating the reserves is provided in the Appendix to DOD-STD-1679A.

5.2.2.5 Parameterization. The contractor is required to provide for the use of symbols in the software design in the same manner that he was in the performance requirements (see Paragraph 5.1.2.9). Symbols should be used to define all parameters, constants, flags, etc. which can be modified without altering the logic of the source program. The intent is to be able to reference those entities by name rather than by value. This will avoid the need to reprogram the

logic each time the value of the entity is changed. To change a symbolic parameter is merely a recompilation knowing that the compiler will prevent any discrepancies in the program logic from occurring.

5.2.2.6 Design constraints. The PDS must take into account any constraints on the design due to the programming language used. This is especially important if using Ada during the early stages of the language's maturation. Design contraints may also result due to the particular software development support environment used.

5.2.2.7 Data base design. This is the design analysis that will result in the development of the Data Base Design Document (DBDD) (see Section 6). This document provides a complete detailed description of all common data items necessary to the software design. The document is based upon the PPS and is developed in consonance with the PDS. This section provides the contractual work description that supports ordering the DBD on the CDRL.

The DBD should be baselined and configuration managed by the contractor as early as possible in the software development effort. The common data base is a living entity; it is a working level detailed definition of all the various data elements which are common to the entire software system. These data elements are subject to frequent change as the software design matures. Even though these changes may be minor, they still must be managed in order to minimize their impact on all affected portions of the software. The contractor should automate the DBD and provide frequent updates of the document for the AM's use as a reference in the review of other program documentation.

5.2.3 Intersystem interface. This Paragraph (as does Paragraph 5.1.2.5.c) applies only to those systems which are required to interface with other systems or subsystems. This is the second stage in the development of the IDS. The first was requirements definition; this is detail design of the interface. This effort may require that the contractor participate with the other contractors as a team in the design of the intersystem interfaces and the development of the IDS. If during the design effort, conflicts arise that impair the system's ability to perform in accordance with its requirements, the cognizant contractor shall identify the problems to the AM and recommend the necessary modifications to the systems or to their interface. This Section provides the contractual work description that supports ordering the second stage of the IDS (as discussed in Paragraph 5.1.2.5.c. herein) on the CDRL.

5.3 Programming standards. The following paragraphs define the minimum programming standards and structured programming procedures the contractor must apply during development of the software.

5.3.1 Language. All software is required to be coded in an approved DOD high order programming language. A waiver must be obtained by the AM to allow the use of any other language, including assembly language. The high order language to be used must be specified by the AM in his Request for Proposal.

5.3.2 Control structures. In implementing the computer programs, the contractor is limited to using only the five basic control structures identified in the Standard. The allowable control structures are: sequence of operation; if then else; do while; do until; and case. There is no need to use any other control structure since it has been proven that any computer program can be implemented using only these structures. Nesting of the control structures is permissible. This requirement is the primary basis for the "structured programming" concept.

5.3.3 Included/copied segments. Included/copied statements used in developing the software are subject to all of the other requirements of Section 5.3 of DOD-STD-1679A, e.g., written in a HOL and constructed with the approved control structures. Entry to the segment shall be through the first executable statement, and the exit shall be through the last executable statement. The use of included/copied statements is often desirable during software development because the segment is programmed only once, but tested often, thereby reducing risk and cost.

5.3.4 Entry-exit structure. Each program element (subprogram, module or unit) shall have only one entry and one exit point. This requirement is another of the integral concepts of structured programming. It is a procedure for reducing the complexity of the programs.

5.3.5 Self-modification. Computer program instructions that are modified during execution of the program are extremely difficult to debug and maintain. This is an unacceptable practice which should not be tolerated under any circumstances.

5.3.6 Size. This requirement is not difficult to achieve. The intent of limiting the size of units is to force a high degree of modularity into the program. Note: There are no limits on the size of a module or subprogram.

5.3.7 Branching. This requirement was the original precept upon which structured programming was built. The objective is to force the sequential flow of program logic. There is a correlation between the number of GOTOs in a program and the program's complexity. Transfers of program control are indicative of changes in the thought processes of the person developing the logic.

5.3.8 Relocatability. The objective of requiring relocatable object load modules is to maximize machine independence of the software. The major advantages of relocatability are: the ability of the module to execute anywhere in memory after linkage with the rest of the program, and the ability of a module to reference external data and other routines without having to know where the linking loader will locate them.

5.4 Programming conventions. The programming conventions discussed below are the minimum requirement. Both the programming standards of Section 5.3 and the conventions herein should ultimately be documented in a project specific manual. They are also required to be identified in the Software Development Plan.

5.4.1 Symbolic parameters. The contractor shall ensure that the programming conventions for representing parameters is consistent with the requirements cited in Paragraphs 5.1.2.9 (Adaptive Parameters) and 5.2.2.5 (Parameterization).

5.4.2 Naming. The contractor is required to establish a uniform convention for naming the elements of the software. It is useful for readability and understanding if the convention includes in the identifier an indication of the function with which the element is associated.

5.4.3 Numerical conventions. The contractor is required to establish a uniform convention for manipulating numbers and for selecting all modes of mathematical operations. The conventions should address how the choice of numerical mode is to be made.

5.4.4 Symbolic constants and variables. The contractor is required to comply with the symbolic parameterization criteria provided in Paragraph 5.4.1 and the naming conventions of Paragraph 5.4.2. All algorithms should be carefully reviewed to ensure that any values not symbolically represented are truly independent of the total system. For example, in calculating the perimeter of a rectangle the algorithm may read, "PERIM = 2 [LENGTH + WIDTH]". Only the 2 is independent of the total system, while the variables LENGTH and WIDTH are dependent on some external factors and are properly symbolized.

5.4.5 Mixed mode expression. Mixed mode expressions are those which have two or more types of numerical representations in the same mathematical expression, consequently, they must not be used. Results of this type of computation are often unpredictable and, thus, have an adverse effect on system reliability.

5.4.6 Grouping. Grouping is the use of delimiters (parentheses, brackets, braces, etc.) to collect mathematical terms so as to clarify the order of evaluation of an expression. The required use of delimiters is important during all stages of the software development process. If grouping of terms is not used in programming the expression, it may be evaluated improperly.

5.4.7 Significant digits. The use of significant digits in the processing of mathematical expressions is an important concept. The various algorithms should be analyzed to ensure that the actual implementation will not result in a detrimental loss of significant digits or the creation of false significance.

5.4.8 Abstracts. This and the next four conventions have a single purpose: to cause software listings to be more readable and understandable by humans. This is a very important maintainability consideration. The abstract is intended to provide an overview description and history of each hierarchical component. The contractor should provide for enforcing updates to the abstracts through his software quality assurance program. Since some contractors are now using Program Design Languages (PDL), such as PDL/Ada, they are attempting to include the PDL in the abstracts in lieu of the textual descriptions. The AM must specifically authorize such a substitution.

5.4.9 Comment statement. Comments are included in the software listing to provide an explanation of what is taking place in the code. Abundant, lucid comments are invaluable for understanding the details of a computer program and invaluable to the life cycle software support activity that will be responsible for maintaining that program.

5.4.10 Indentation. Indentation of statements is used to delineate the various levels of nesting in the coding. Two to five spaces of indentation per level is typical. Indentation facilitates reading of the listing.

5.4.11 Software listings. This convention only addresses the format of the listing for purposes of delivery.

5.4.12 Cross-reference listing. The cross-reference listing is a road map through the various identifiers and units within the software. This listing enables programmers and reviewers to determine where all identifiers are set and used, and identify all external references to the identifier. This convention is a very important maintainability consideration.

5.4.13 Load Maps. Load maps identify the manner in which the various segments of code are placed together, their location within main memory, the location on secondary storage devices, and the physical size of the individual segment. As the name implies, the load map enables the programmer to determine the location of any segment of code. This mapping is important to the debugging process during development as well as to life cycle software support. The load map must be prepared for each processor and storage device, and for each mode of program operation.

5.4.14 Execution efficiency. This convention serves to formalize guidance to designers and programmers in making trade-offs during development of the software. Giving deference to readability, clarity, and maintainability is in recognition of their beneficial effects on life cycle software support.

5.4.15 Source code segment includes/copy. This is another convention that provides guidance in making efficiency trade-offs.

5.4.16 Source statement. The use of complex statements increases the risk of error. Since simple statements are not necessarily less efficient, there is little valid benefit of using compound or complex statements.

5.4.17 Flow charts. Flow charts were originally devised as a tool for aiding in the creation of the design of a computer program. Automatically generated flow charts violate this purpose since they can only be generated from completed programs. To be used as a tool for debugging or maintaining programs requires that the flow charts be kept in perfect synchronization with the code which they depict. The additional burden to revise flow charts each time the code is revised is seldom expended. The result is inaccurate flow charts, and a general distrust of the accuracy of flow charts, thus, limiting their usefulness. Finally, flow charts are an inappropriate tool for use in conjunction with DOD-STD-1679A required practice of structured programming. It should be noted that Paragraph 5.4.17 in DOD-STD-1679A only discourages contractual delivery of flow charts because it is an extremely wasteful expense. However, the higher level flow diagrams which are required by DOD-STD-1679A are useful as a software engineering tool. A Program Design Language (PDL) is often used in lieu of the detail design flow charts.

5.4.18 Design representations. These are diagrams or other graphic representations used by the developer to facilitate the structured design and code of the program. A PDL/Ada is an example. The use of such aids should be encouraged, and they should be retained for use by the life cycle software support activity after the delivery of the software. This will facilitate the post development modification of the programs. Tools of this type have superseded the use of flow charts for structured programming.

5.5 Software implementation. Software implementation entails the development of the computer programs and their attendant documentation. It should be noted that the work performed under this Section is primarily coding and debug (unit test). This Section of DOD-STD-1679A provides the contractual work description for ordering the Program Description Documents (PDDs) and the Program Package Document (PP) on the CDRL.

This Section requires that the contractor continue to use the top-down software development methodology and structured programming. An important benefit derived at this stage from the top-down technique is that the control units at the highest levels of the hierarchy will be developed and tested first and, throughout the integration process, will be tested repeatedly. The use of stubs as place markers to represent requirements/functions whose implementation has been deferred must be continued during the implementation of the programs. DOD-STD-1679A requires a code walk-through of each unit developed prior to its release to the library for configuration management. As was discussed earlier in Section 5.2, it is a violation of the walk-through concept for the contracting activity to be involved in their conduct or to even audit them. Once again, compliance is assured through the contractor's software quality assurance practices and procedures.

Upon successful completion of unit test, it is required that the unit be placed in the library and custody of the unit transferred from the programmer to the librarian. It is important that no unit of code be used by anyone other than the author until it is under formal configuration management. It is impractical to formally configuration manage a unit any earlier in its development.

5.5.1 Software development organization. This is additional reenforcement of the requirement to use the top-down software development methodology and structured programming. There is no particular organizational structure which the contractor must follow. The evaluation of the contractor's proposed organization will have to be made on a case-by-case basis.

5.5.2 Resource management. The contractor is responsible for management of the computer resources as discussed in Paragraphs 5.1.3 and 5.2.2.4. It is during software implementation that the computer resource reserves are most likely to be eroded. Consequently, continuous monitoring of the reserves available is very important during this phase.

5.5.3 Unit test. Unit test (debug) is to be completed prior to placing the unit into the library under configuration management for subsequent integration with other units and performance testing as required in Section 5.8. This informal testing is the initial de-

bugging of the code by the programmer. The purpose of debug is to detect and correct as many programming errors as possible before integrating the unit into the program. During unit test it is required that each source code statement that is to be delivered be executed at least once. This requirement is not to be confused with complete path testing or exhaustive testing, neither of which is practical.

5.5.4 Library usage and control. Libraries are collections of documentation, code, and data which provide the total representation of the software at any stage of its life cycle. Strict control of the use and changes to items in the library is mandatory. This responsibility is normally given to a specific individual commonly referred to as the project's configuration manager. The configuration manager is tasked with the enforcement of the practices and procedures for control of the library as specified in the Software Configuration Management Plan. This includes an accurate account of all changes to the documentation under his control.

It is reasonable to require the contractor to have an automated configuration management system. The automated configuration management system can be used to keep track of the Software Change Proposals (SCP) and Software Trouble Reports (STR), as well as the patches which the Standard requires to be kept in electronic media. The automated configuration management system also facilitates the functions of configuration management status accounting and reporting. It is an extremely powerful management tool.

5.6 Software generation. This requirement is for the purpose of ensuring that the vital set of support software necessary for life cycle software support is available and that the life cycle software support activity has the instructions necessary to use it. It is strongly recommended that the AM specify in the Request for Proposal and in the Statement of Work, the delivery of all of the support software needed for life cycle software support of the software under development. The AM must specify the use of a software development support environment that is compatible with the delivered software's life cycle support environment.

5.7 Software operation. This Section of the Standard does not contribute to the software engineering process. This Section only serves as a contractual work description to support ordering the Operator's Manual (OM) and the System Operator's Manual (SOM) on the CDRL. This is a very important consideration since it represents the user documentation for the software under development.

5.7.1 Non-functional operation. The software system procedures for system configuration, set-up, initialization, and operation are to be defined by the contractor. They are documented in the OM.

5.7.2 Functional operation. The contractor defines in the SOM the specific procedures to be executed by system users during all modes of system operation, included degraded modes, and all man-machine interactions.

5.8 Software testing. Section 5.8 of the Standard prescribes software performance testing requirements. This is the testing that commences upon completion of unit test (described in Section 5.5) and the initiation of integrating the units into the program. Described is the engineering effort that is expended in preparing the software for delivery. A widely accepted, time proven rule of thumb states that 50% of the total software development effort ultimately will be expended during these integration and test activities. Although it is improper for the AM to dictate organizational restrictions on the contractor, he should, however, encourage the use of software testers that are independent of the engineers who developed the software. This promotes objectivity in testing and helps to ensure testing against requirements instead of against the "as-built" interpretation of the requirements. Section 5.8 of the Standard provides the contractual work description for ordering the Computer Program Test Plan, Computer Program Test Specification, Computer Program Test Procedures, Computer Program Test Report, and Software Trouble Reports on the CDRL.

The contractor is required to provide all of the support software necessary to conduct, control, record and analyze the testing. Software designed to provide the external stimuli to the operational system during testing is an example of this type of support software. Any software used in performance testing should be formally tested, documented, and placed under configuration control. The AM should ensure that this software and its documentation is cited as deliverables in the contract. It is critical that the AM review and approve all test documentation. This review, and subsequent approval by the AM, is especially important since it will be used as the basis for the testing conducted to determine the software's acceptance.

Performance testing must be conducted in a controlled environment. It is the AM's responsibility to be involved in the decisions regarding all environments to be used for performance testing of the program. The contractor must identify in his proposal any equipment, facilities, or information to be supplied by the AM in support of testing. The required equipment could include non-computer related hardware to provide external stimuli or a mobile platform to carry the system. The contractor must advise the AM sufficiently in advance to ensure the availability of the requisite equipment and information to prevent adverse impact on the schedule of testing.

It needs to be emphasized that testing is not software quality assurance. Testing can only establish the absence of quality. Quality is designed and built into the software, it cannot be tested in. Section 5.9 discusses the quality assurance of software.

5.8.1 Software integration and developmental testing prerequisites. Each software unit is required to satisfy several requirements before its integration and developmental testing can begin. The requirements discussed below are considered to be the minimum prerequisites for initiation of this testing. The entire set of units comprising the program do not have to have satisfied these requirements before testing of the first unit may begin. The top-down software development method calls for the incremental integration and test of the finished units as they gradually evolve toward the complete program.

a. The unit must have been compiled without errors. Error-free compiles are a prerequisite for unit test. Therefore, the contractor will normally satisfy this requirement in the course of routine software implementation.

b. A code walk-through in accordance with approved procedures must have been completed. Satisfactory completion of a code walk-through includes, as a minimum, the correction of the errors and deficiencies noted during the actual walk-through.

c. The code must have completed unit test satisfactorily. The completion of unit test includes the correction and retest of all errors discovered during the unit test.

d. Procedures for verifying that a unit satisfies its requirements must be contained in the Software Development Plan.

e. The Software Quality Assurance Plan must contain the specific requirements to be satisfied by each unit before it is acceptable for turn-over to the library.

f. Having satisfied all of the requirements in a. through e. the unit is to be turned over to the custody of the contractor's configuration manager and placed in the library for configuration control. The procedures for this turn-over are to be found in the Software Configuration Management Plan. This turn-over must be completed prior to integration and developmental testing of the unit.

5.8.2 Software integration and developmental testing. The Standard discusses the integration of units into modules for purposes for developmental testing. Subsequently, many modules evolve as the stubs are replaced by the completed units. This process continues,

as the subprograms emerge, until the program has been totally integrated. The developmental testing described in this Section requires this entire top-down process.

Developmental testing is conducted top-down according to the approved test documentation that was based on the requirements contained in the PPS. The test specifications and procedures, as a minimum, must include provisions for testing those items enumerated in Section 5.8.2 of the Standard. This testing is intended to ultimately prepare the software for its acceptance testing. The contractor must conduct all testing in accordance with the approved test procedures. This is necessary in order to ensure that all requirements have been tested and to be able to repeat tests exactly. Discrepancies cannot be corrected unless the error condition can be duplicated. All requirements stubbed in the software must also be stubbed in the test documentation. This eliminates the confusion of attempting to test for a requirement which has not been implemented. The witnessing of tests and the review of test reports are important contractor and AM quality assurance requirements.

5.8.3 System(s) integration test. This test is required only when the software system under development is a part of a larger system and the software system under development is scheduled to be integrated with the elements of the larger system. The AM must coordinate this testing with the other contractors and all may be required to participate in and provide technical support for the system integration testing. System integration testing may be conducted at a site other than the software development contractor's facility, such as the contracting activity's test site. If the software development contractor is to conduct the system integration testing, the technical support for the other systems should be required of the respective contractors. The system integration test is intended to ensure that the requirements of the larger system, which are performed or affected by the software system under development, are satisfied. In addition, the interface between the larger system and the software system under development is to be completely tested. The IDS is a critical document since it defines the requirements that the software system under development is to satisfy in interfacing with the larger system. The system integration test will require a complete set of computer program test documentation from the Test Plan through to the Test Reports.

5.8.4 Software trouble reporting. Once a software unit is accepted into the library and placed under configuration management, it is required that Software Trouble Reports (STR) be utilized for documenting and processing all errors discovered in the unit. These procedures are to be contained in the contractor's Software Configuration Management Plan. Since the use of the STR is required for controlling changes to software after it has been placed under configuration control, it is normal for a large number of STRs to be

generated during developmental testing. It is important that the STR procedures be strictly enforced because they contain the information necessary for duplicating the problem so that it can be corrected. The Software Configuration Management Plan provides the mandatory procedures for the status accounting and reporting of STRs. Invaluable management and progress information can be derived from the STR status accounting file. Consequently, the AM should have access to this data throughout the course of the software development contract.

5.8.4.1 Software Trouble Report category. There are three categories of STRs. Earlier versions of the STR form (see Section 6) contained four categories. The category, Logic (L), has been subsumed by the category, Design (E). Standardization of the categories (and priorities) of STRs is important for the purposes of fault analysis by the life cycle software support activity.

a. Software trouble (S). This category is for those errors (bugs) in the software. The best time to detect (and correct) this type of error is during the code walk-through or unit test.

b. Documentation trouble (D). This category is for documentation errors. The documentation troubles are normally discovered during walk-throughs or reviews of the document. Typographical errors are not considered documentation errors unless the meaning of the phrase is changed.

c. Design trouble (E). This category of error occurs when the documentation is correct and the software conforms to the documentation but the software fails to satisfy a requirement. This category of error is the most difficult and costly to detect and correct. Design deficiencies which are not discovered until relatively late in the integration and test process are very expensive and time consuming to correct. This is why design walk-throughs are so important. Design corrections by their very nature usually have pervasive impact on all of the software.

5.8.4.2 Software Trouble Report priority. The priority assigned to an error is intended to indicate the importance of correcting the problem based on its impact on system operation. The number of STRs in the various categories and according to priority provides insight into the maturation of the evolving software. Standardization of STR priorities is very beneficial to the life cycle support of the software.

a. Priority 1. Problems that result in program stops, hazards to personnel or equipment, or loss of a major function must be given the maximum amount of attention. During integration and test, problems of this type have serious impact on the cost and progress of the

software development effort. After acceptance and employment of the system, these errors can jeopardize human life or prevent the system from performing its mission.

b. Priority 2. The key distinction between priority 1 and priority 2 errors is the difference between "prevents" and "adversely affects...with no alternate work-around".

c. Priority 3. The key distinction between priority 2 and priority 3 errors is the availability of an acceptable "work-around solution", restarting or reloading the program are not acceptable solution.

d. Priority 4. Priority 4 errors do not significantly impede system operation and only present a nuisance to a system operator.

e. Priority 5. All errors of lesser significance fall into this priority group. All documentation errors fall into this category.

5.8.4.3 Software Trouble Report disposition. The proper management of STRs is important since they represent errors to be corrected prior to the software's acceptance and because errors represent impediments to the software development effort. All STRs have cost and schedule impact on the project. Procedures must be established to prioritize and resolve STRs.

5.9 Software quality assurance. The software quality requirements are embedded throughout all of the Standard. Section 5.9 is intended to identify minimum requirements on the contractor's internal software quality assurance program. How the contractor intends to apply his software quality assurance program to this particular development effort is to be contained in his Software Quality Assurance Plan (SQAP) (see Section 6.1 of the Standard). This Section provides the contractual work description that supports ordering the SQAP on the CDRL. The AM is limited to monitoring contractual compliance. Consequently, it is very important when reviewing, and subsequently approving, the contractor's SQAP to ensure the desired level of software quality assurance will be applied to software development and that adequate enforcement mechanisms are provided. The contractor's software quality assurance program must also contain procedures for monitoring the timely detection, correction, and reporting of discrepancies discovered during all facets of the software development process.

The Standard does not place any organization structure requirements on the contractor. Also, the Standard attempts to avoid any inference of the existence of a software quality assurance organization. It is not even required that the same group perform all of the software quality assurance functions. The intent is that the contractor have an established software quality assurance program and that

qualified personnel be assigned to perform each of the functions therein. Those who are assigned the software quality assurance functions should not to be in the same management chain as the software development organization. The problems, errors and discrepancies discovered during the software quality assurance activities are only reported to the project manager for resolution. They, SQA personnel, monitor the effort with the objective of discovering discrepancies at the earliest possible stage of development. They must be careful not to indulge in a subjective critique of a persons work. A well documented software quality assurance program and SQAP are vital ingredients to establishing and maintaining good working relationships between software development personnel and those with software quality assurance responsibilities.

5.9.1 Reporting level. In order to maximize objectivity, the AM should demand that the contractor personnel assessing internal compliance with the software quality assurance program have an organizational reporting channel that is independent of the contractor's project manager.

5.9.2 Participation in audits. The contractor's internal software quality assurance program is required to contain procedures for the conduct of software quality audits throughout the total software development. These procedures must define the required participation in the audit. The SQAP must tailor these procedures to the specific software system that is being developed.

5.9.3 Requirements reviews. The contractor's internal software quality assurance program is required to contain procedures related to the conduct of software requirements reviews during development of the PPS. It must also define the required participation by all parties to a review. The SQAP must tailor these procedures to the specific software system that is being developed.

5.9.4 Design reviews. The contractor's internal software quality assurance program is required to contain procedures for the conduct of design reviews during development of the software design. It must also define the required participation in a review. The SQAP must tailor these procedures to the specific software system that is being developed.

5.9.5 Software design. The contractor's internal software quality assurance program is required to contain procedures for monitoring the development of the design of the software. It must also define the required participation in these activities. The SQAP must tailor these procedures to the specific software system that is being developed.

5.9.6 Program code. The software quality assurance effort includes routinely reviewing the programming activities during the development of the software. The objective remains the early detection of discrepancies. The Software Quality Assurance effort is also concerned with the code's compliance with established programming standards and conventions. Code audits are tools available for use in assessing the quality of the code.

5.9.7 Tests. Software quality assurance personnel do not conduct tests. The software quality assurance requirement is to witness tests to ensure that the tests are being conducted in accordance with the approved test procedures, that all errors detected are properly documented on an STR, and that accurate, complete test reports are submitted. Software quality assurance procedures must also include monitoring the maintenance and control of the test environment and the test support software.

5.9.8 Deliverable items. The quality assurance program is required to have procedures in effect during the contract which will cause each deliverable to be subjected to review by the software quality assurance personnel. This review is a final check of the item's readiness for delivery.

5.9.9 Reporting. The quality assurance program shall have procedures to record and report all discrepancies noted as a result of the software quality assurance activities. These discrepancy reports should be considered project action items. The items should remain open until quality assurance personnel have determined that the appropriate action has been taken. This procedure is internal to the contractor and the AM should not be involved. The AM should be advised of the status of these action items at delivery of each program item.

5.9.10 Software trouble reporting. The software quality assurance program shall have procedures to monitor the processing of STRs. Audits shall be conducted to verify the proper functioning of the processing and status accounting of STRs.

5.10 Software acceptance. The Standard contains four acceptance criteria that are applicable to all users. Each AM using DOD-STD-1679A must augment these criteria with additional criteria that are specific to the program, such as satisfying the requirements of the PPS.

5.10.1 Reserve requirements for software acceptance. The AM must include in the contract specific reserve requirements and a description of how they will be measured during the process of software acceptance. Where possible, it is recommended that the reserve requirements be increased to 50% for both memory and central processor throughput. The Appendix to DOD-STD-1679A contains guidance in measuring the reserves available.

5.10.2 Stress Test requirements for software acceptance. The software stress test is intended to be an integral part of the software acceptance process. It must be completed successfully prior to final acceptance of the software. This is a unique test apart from all other developmental testing, consequently, it will require its own complete set of documentation; Plan, Specification, Procedures, and Report. The stress test has two objectives: First, it is an attempt to demonstrate that the software will run for an extended period in an operational environment with simultaneous missions being performed, without serious errors. This is different than testing to satisfy specific functional requirements in a laboratory environment. Second, it must demonstrate that the software will not stop functioning entirely just because it has exceeded its designed capacities. In other words, the software must have been designed and programmed for the eventuality that the capacities might be exceeded and will logically handle the condition without causing the program to fail and the system to shut down. This is particularly critical in real-time systems where a potential hazard to personnel or equipment exists should the system shut down. It is strongly recommended that the Stress Test be conducted by an independent third party. The software development contractor should be a last resort for conducting the Stress Test since it would be difficult for him to remain totally objective in the face of his conflict of interests.

5.10.2.1 Stress Test environment. Defining the environment for conducting the Stress Test is extremely critical to the validity of the test. It is normally impractical to conduct the test in the ultimate user environment due to a lack of availability of some of the elements of the system. It is also often impractical due to the inability to generate the necessary inputs and conditions that will exceed the software's capacities. During the integrated Stress Test, all digital interfaces with the computer program being tested must be present or simulated. A fully integrated test facility should have the same computer and display equipment as in the eventual user environment. The specification of the environment must be included in the original contract.

5.10.2.2 Software to be tested. This requirement is included to preclude the situation whereby a special version of the operational software would be created especially for the Stress Test. Normally, the computer programs will be placed under configuration management by the AM prior to commencement of the Stress Test. The objective of the test is to ensure that the program to be delivered can withstand the stress conditions. Section 5.6 of the Standard defines the software generation requirement applicable to the software to be used for the Stress Test.

5.10.2.3 Stress Test documentation. The important aspect of this requirement is that when completing the CDRL the AM must ensure that the delivery date for all software documentation predate the conduct of the Stress Test. This also includes the Stress Test documentation discussed in Paragraph 5.10.2 above.

5.10.2.4 Stress Test software operation. This paragraph requires continuous operation of the program during the Stress Test. Consequently, any program stop (regardless of the reason) or invocation of an auto-recovery routine prior to completion of the test constitutes a failure of the test. Auto-recovery routines are designed to execute when a program experiences a failure (stop). Consequently, their invocation is indication that a failure has occurred. After a failure of the Stress Test, it must be rerun in its entirety. One of the objectives is to establish that the software can perform in an operational environment for long periods of time (much longer than the Stress Test duration). Therefore, as a minimum, it must be able to run continuously through a complete Stress Test without a program stop. This process is repeated until the test duration requirements have been satisfied.

5.10.2.5 Stress Test duration. For purposes of the Stress Test, all programs can be divided into two categories. Those that are designed to operate continuously and those designed to execute for a prescribed duration. Some systems are intended to operate continuously for days or months. The arbitrary period of 25 hours was established to provide a measure of confidence that continuous operating programs will not experience serious errors over extended periods of time. A period of 25 hours was felt to be sufficiently close to the point of diminishing return that a longer test duration, although desirable, would be less cost effective. Because of the complexity of some systems the AM may require longer test duration just to satisfy all of the test requirements. The testing of both categories of programs shall be continuous. If a Stress Test is stopped for any reason whatsoever it must be restarted and run for its complete duration.

5.10.2.6 Stress Test input data. The Stress Test inputs shall be defined in the Stress Test documentation. The inputs to the program during the Stress Test must span the full range of all possible system inputs to test normal software operation. In addition, attempts to input values outside the designed range and at possible singularity values (such as 0) should be made in order to test the software's ability to deal with an illegal input and continue to perform other operations which may be occurring simultaneously. A particular function may satisfactorily pass a test that is always performed in the same sequence. Therefore, the function must be tested in several random sequences to preclude misleading test results. For complex systems, this must include different combinations of functions per-

forming simultaneously during different periods of the test. And, since most systems interface with human operators who are subject to making errors, illegal inputs must be inserted to ensure that they will not stop the program. The Test Procedures shall be designed to accommodate these variations while ensuring compliance with the test requirements.

5.10.2.7 Stress Test performance. Notice that the Standard requires "at least three distinct stress periods and the total time spent stressing the system shall represent at least one-third of the total length of the test." This will provide alternating periods of normal and maximum loading of the system. This is intended to uncover any errors that result from cumulative effects over extended periods of operation. It is also intended that functions be exercised repeatedly during the test, not just once. During the periods of stress, the objective is to overload any and all software capacities in the attempt to expose the presence of any serious errors. The pass/fail criteria for the Stress Test is contained in Paragraph 5.10.2.10.

5.10.2.8 Reduced capability software stress testing. This testing should be conducted after all other Stress Test criteria have been satisfied, especially if the reduced capability software must be loaded after the failure occurs, e.g., failure of a memory board or one of several CPUs. The objectives of this test are the same as for the full capability software, however, the performance requirements in the reduced mode may be quite different. It is necessary for the AM to specifically include provisions for this level of test in the contract, including the proper test documentation.

5.10.2.9 Test and maintenance support software stress testing. The assumption is made that a user, during normal operations, would have the option to select the period to conduct maintenance and would not choose to do so during periods of high system utilization. But on the other hand, performance requirements would not have specified "on-line" maintenance if there were not a requirement for the system to continue to perform nominal functions during the maintenance periods. Therefore, test and maintenance support software must be tested while the system is performing a variety of routine functions at a nominal level of activity. It is necessary for the AM to specifically include provisions for this testing in the contract, including the requirement for proper test documentation.

5.10.2.10 Errors during test. This paragraph contains the pass/fail criteria for the Stress Test. The thinking is that if the software can't run for the duration of the Stress Test without a Priority One or Priority Two error, the software is not mature enough to be employed in an operational environment. It also provides di-

rection for dealing with serious errors that cannot be repeated at will. The AM has the ultimate responsibility in accepting software with an intermittent error. Patches are not permitted during the test because it changes the software from that point on in the test. There is no way of knowing how a change would have affected the test had the change been in the software from the beginning. Thus, there is the requirement to run the Stress Test from start to finish without a serious error.

5.10.2.11 Stress Test limitations. It is desirable to have as few errors in the final version of the software as possible. But in addition, a relatively small number of errors is usually an indication of mature, high quality software. If during the course of conducting the Stress Test the number of errors exceeds the limits set forth in the Standard, it suggests that the software is not ready for delivery but requires more developmental testing. The Stress Test is costly to conduct and therefore should not be used as an extension of developmental testing. If the contractor is having difficulty meeting the error and patch limitations, the AM should consider delaying the Stress Test until such time as the software has stabilized.

5.10.3 Error limits for software. The software error priorities in this Section are based upon the criteria defined in Paragraph 5.8.4.2 of the Standard. The limits are based upon the total number of executable instruction words developed by the contractor. Buffers, data, files and other non-instruction words are not counted. An error is resolved when the STR documenting the error is officially closed. This requires not only the design, development and implementation of the correction, but also the successful testing of the correction. Intermittent errors known to exist count as an unresolved error.

5.10.4 Patch limits for software acceptance. The limitation on patches is one of the more difficult to achieve. Due to the capabilities provided in current software development facilities, there is a strong tendency for AMs to demand patch-free software deliveries. While this is clearly most desirable, other important considerations may render a patch-free delivery impractical. A major draw-back is the potential loss of configuration management of the software, STRs, and corrections during the final days of acceptance testing. This is compounded by the need to regression test the software that had previously completed a portion of the testing because of changes to a particular item. It would be a questionable decision not to regression test the corrections to errors discovered during the Acceptance and Stress Tests. Caution is recommended when deciding to lower the Standard's patch limits for a specific program. If a high number of patches are required during the final days of acceptance, it is an indication that the software is not sufficiently mature, needs more testing, and is not ready for delivery.

5.11 Software configuration management. Similar to the software quality assurance requirements, this Section establishes minimum requirements on the contractor's internal software configuration management program. It is very important for the contractor's software configuration management program to be integrated into his system configuration management program when hardware is being developed in conjunction with the software. How the contractor plans to apply his software configuration management program to this particular development effort is contained in his Software Configuration Management Plan (SCMP) (see Section 6.1 of the Standard). This Section provides the contractual work description that supports ordering the SCMP on the CDRL.

It must be stressed that these software configuration management requirements are internal to the contractor and address the working level of activities within the project. The contractor's developmental baseline is the foundation for these requirements and the SCMP. Consequently, this area of configuration management is independent of that addressed by MIL-STD-483, "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs" and DOD-STD-480A, "Configuration Control - Engineering Changes, Deviations, and Waivers". These two Standards are at the higher, contractual level in that they address baselines that are controlled by the AM and changes that affect the contract.

The automated configuration management system discussed in Paragraph 5.5.4 is crucial to the configuration management of software. The information contained in the automated system at the time of software delivery is extremely valuable for the life cycle support of software. It is mandatory that this information be transferred to the life cycle software support activity upon conclusion of the contract. If this information is dependent on the automated system, the AM must provide for the acquisition of the automated configuration management system.

5.11.1 Software configuration identification. The SCMP is required to contain procedures for the identification of all software and documentation items. The SCMP defines all baselines to be used during the development as well as numbering/labeling scheme to be employed.

5.11.1.1 Developmental Baseline. The SCMP must identify this baseline in terms of the documents that constitute its definition. The Developmental Baseline starts out composed of all the contractual specifications that contain requirements on the software. During the software development this baseline evolves as newly created software documentation is incorporated. The SCMP must identify these items and cite when and under what conditions these items will be incorpo-

rated into the baseline. The Developmental Baseline will evolve into the baseline upon which all software configuration management will depend for the life of the contract.

5.11.1.2 Documentation identification. In the SCMP the contractor will propose the numbering/labeling scheme that he intends to use in the software development effort. It is important that, when reviewing and approving the SCMP, the AM ensure that the scheme is compatible with other related contracted activities and with the life cycle software support activity's scheme.

5.11.2 Software configuration control. Software configuration control may be the single most critical discipline within the software development effort. The loss of configuration control has been the cause for many of the so called "software disasters". The SCMP must contain meticulous procedures for controlling the products of the software development effort as well as all subsequent changes. Some of the items that must be managed are: documentation, external interfaces, internal interfaces, data bases, source code, object code, patches (patches to the patches), SCPs, STRs, and plans. The SCMP must define when each item within the project is to placed under formal, whether contractor or AM, configuration management. It must identify all configuration management prerequisites to be satisfied by each item before it can be accepted under configuration control. Importantly, the SCMP must define the procedures for proposing, submitting, processing, approving/disapproving, implementing, and closing-out all changes to items under formal configuration control. The automated configuration management system is an invaluable tool for implementing software configuration management.

5.11.2.1 Software changes. Software changes are those candidate modifications to items under software configuration management. Software, both instructions and data, is completely defined in terms of its documentation, requirements, design, implementation, data bases, interfaces, test, and user. Therefore, to make changes to software requires that changes be made to its documentation. The procedure involves proposing a change to the applicable documents (baseline), via a Software Change Proposal (SCP). Once the proposed change to the baseline is approved and incorporated, then and only then is the software changed to regain conformance with its official definition.

If an SCP has contractual cost, schedule, or AM controlled baseline impact, the SCP shall be used as the basis for an Engineering Change Proposal (ECP), Class I. The ECP is defined in DOD-STD-480A.

Note: STRs denote software errors whereby the software does not conform to its official definition (documentation). The status accounting portion of the software configuration management program is used for tracking STRs for the detection and correction of errors in software that has been placed under formal contractor configuration management.

5.11.2.2 Documentation changes. Since the documentation contains the definition of the software, it is important that all documents remain accurate and current at all times. This requires that changes to documents be distributed promptly and no later than when the software is changed. The SCMP is required to contain procedures to ensure currency and accuracy of the documentation.

5.11.2.3 Software configuration control board (SCCB). This board is internal to the contractor and is responsible for processing proposed changes to the Developmental Baseline and making recommendations to the project manager as to the disposition of the change. A common misunderstanding is that SCCBs can approve changes. Since SCCBs do not have the final fiscal responsibility or project accountability, they must defer decision to the person who does, the project manager. The SCMP is required to define the procedures under which the SCCB manages the Developmental Baseline and for the proper processing of SCPs. The SCCB often serves as a filter for higher level boards, such as the system CCB or the AM's CCB.

5.11.3 Software configuration status accounting. The contractor should have an automated status accounting system (a dedicated word processor can be used effectively in the unlikely event that he does not have an automated configuration management system). The level of change activity associated with developing software is too great for manual efforts to be cost effective. Status accounting is a vital input to effective management of the software development process.

5.12 Software management planning. Paragraphs 5.12, 5.12.1, and 5.12.2 of the Standard are not intended to impose requirements on how the contractor manages the project. The purpose of these paragraphs is to provide the contractual work description that supports ordering the Software Development Plan (SDP) on the CDRL. The SDP is the management document that defines how the contractor intends to manage the software development effort. The AM's approval of this Plan constitutes his concurrence with the contractor on all aspects of the software development project.

5.12.3 Software status reviews. This Section and its subsections provide the contractual work description to support ordering the various management reports and briefings that are a natural component of all contracted efforts.

5.12.4 Software documentation reviews. The requirement of this Section is not an additional burden but rather a re-statement of the general practice of reviewing all deliverables. The reviews are to be scheduled in two phases. The first is a working level review wherein the AM can provide mid-course corrections. The second is a formal review conducted as an adjunct to acceptance of the item for delivery. The contractor is required to provide draft copies of the review material to the AM sufficiently in advance of the review to allow evaluation of the materials. It is incumbent on the AM to be prepared for the reviews on schedule in order not to give the contractor cause for delay in meeting related schedules that are impacted by the results of the reviews.

5.12.5 Special software reviews. This Section merely provides the AM the option of scheduling other program reviews as he deems necessary. It does identify a special review of the developmental test program that the AM must require.

5.12.6 Inspections and audits. This is a standard requirement in most contracts. It merely preserves the right of the AM to physically inspect and audit the contractor software development activities.

## 6. MISCELLANEOUS

6.1 Contract data requirements. This Section is contractually non-mandatory. It is included to indicate the Data Item Descriptions (DID) that come under the cognizance of DOD-STD-1679A. This set of DIDs represents those documents which constitute the minimum requirement for adequately documenting the developed software. The first six DIDs listed document the software per se. The next four document the software testing. DIDs listed 11 and 12 are for user documentation, followed by the DIDs for the management plans. The final two DIDs listed are for the software change forms.

In the case of the software, each of the six documents contains a stand-alone definition of the baseline and, thus, provides the proper vehicle for software configuration management. Each of the six documents also represents a software engineering activity that is integral to software development and cannot be done away with. It would be a tragic case of false economy to fail to order (or to subsequently delete) any of these documents on the contract.

The test documentation is essential for both development and life cycle support of the software. First, it provides the contractor's approach to testing. Second, it provides the means to trace the functions tested to the requirements to be satisfied. And third, it

documents the result. With this set of documents the software's original performance can be validated throughout its life cycle. It too cannot be eliminated in an attempt to reduce costs.

The inclusion of documentation for operating and using the software is mandatory.

The three management plans serve two purposes. They force the software developer to do his homework and adequately prepare for the undertaking. Secondly, they provide the vehicle for the AM to influence the contractor's management of the software development. The AM's approval of these plans authorizes the developer to execute them. The AM must approve any subsequent changes to the plans.

The STR and SCP forms provide vehicles for standardizing the format and content of that information which is invaluable in developing the historical input to the software's life cycle support. Since STRs and SCPs are essential ingredients in software development, including their attendant work description in the Standard minimizes their delivery cost. Formatting the information on the forms for purpose of delivery is an insignificant accommodation when using automated systems to support the development of software.

Certain of these documents may be combined. This is permitted when doing so will reduces costs. The following documents are permitted to be combined:

a. Program Design Specification, Program Description Document, and Data Base Design Document.

b. Computer Program Test Specification and Computer Program Test Procedures.

c. Operator's Manual and System Operator's Manual.

d. Software Development Plan, Software Configuration Management Plan, and Software Quality Assurance Plan.

In addition, if there are no interdigital interfaces in the Program, an Interface Design Specification is not required. Any other deletions or combinations require a waiver.

Although the documents listed in Section 6.1 of the Standard comprise the minimum set which must be acquired for any Program, it may be necessary or desirable, especially in the case of large systems, to acquire additional documentation above this minimum requirement.

# APPENDIX B

## FISCAL CONSIDERATIONS

### 10. GENERAL

10.1 Scope. This Appendix discusses the fiscal considerations of using DOD-STD-1679A (the Standard) on a contract that includes software development.

10.2 Application. This Appendix provides information that should be read and considered by the AM before allowing a deviation from the Standard or waiving any of its requirements.

### 20. REFERENCED DOCUMENTS

DOD-STD-1679A, Software Development

### 30. DEFINITIONS

As provided in the Standard.

### 40. GENERAL REQUIREMENTS

40.1 General. There is a common misconception among some AMs and contractors that requiring the use of the Standard will drive up the cost of developing software. While actual cost figures from software development projects, which either did or did not implement the Standard, are not generally available, the rationale presented in Section 50 should dispel this misconception.

### 50. DETAILED REQUIREMENTS

50.1 Minimum requirements. The Standard specifies only the minimum set of requirements required for any software development effort. This minimum set was the end result of contributions from literally hundreds of software engineers and months of editing and reducing. It is unlikely, therefore, that a software developer will have devised a new way to develop software by using less than this minimum set of engineering practices and disciplines. Since only the minimum requirements are specified, there should be no additional costs beyond those required to do a satisfactory software development.

50.2 Work description. The Standard only describes work to be performed under the contract to satisfy the minimum set of requirements. There are no deliverables, including documents, embedded in the mandatory sections of the Standard. Consequently, the AM will

not pay for any deliverable item that he does not properly order elsewhere in the contract, e.g., CDRLs. The AM completely determines the costs associated with the deliverables when he specifies them throughout his contract.

50.3 Software engineering disciplines. The Standard specifies a number of state-of-the-art software engineering disciplines that are to be used during software development. These disciplines, such as high level languages and structured programming, required by the Standard, were created to reduce costs, improve maintainability, and make programmers more productive with fewer errors. These and the other disciplines required by the Standard have withstood the test of time and are still in use and recommended for use by leading software developers. The test of time would have filtered out any discipline that was not cost effective. These are proven techniques and methodologies which have been determined to be cost effective and contribute to better quality software. These disciplines will help to control the cost of software development as well as improve the quality. Therefore, the contractor should already be using these disciplines or something very similar and there should be no additional cost associated with implementing them. If the contractor suggests that there are start-up costs involved in implementing some of the disciplines in the Standard, then there is good reason to believe that this particular contractor has never used some of these disciplines and it casts some doubts about his technical competence to perform on a software development contract.

50.4 Software quality assurance. When addressing the cost associated with the Standard's software quality assurance requirements, there are two areas that must be kept in proper perspective. First, only consider the cost of the quality assurance procedure, not the software engineering cost. For example, code walk-throughs are required by the Standard. The Standard does not state how often they are conducted or who is required to participate. These are software engineering procedures contained in the SDP and, accordingly, are software engineering costs. The software quality assurance role of ensuring that they are conducted as specified is defined in the SQAP and, accordingly, is a software quality assurance cost. Thus, the contractor determines the software quality assurance costs associated with that role in code walk-throughs. The same is true for all of the software quality assurance requirements in the Standard. Secondly, a proper perspective must be maintained in how software quality assurance is to be charged to the contract. It should be charged exactly the same as for hardware quality assurance, with certain overhead items being charged indirectly. The AM should never tolerate an attempt to bill the total cost of software quality assurance directly to his contract. If these items are kept in proper perspective, then the true, and reasonable, software quality assurance costs can be identified.

To reiterate, if the contractor suggests that there are start-up costs involved in implementing some of the software quality assurance practices or procedures required in the Standard, then there is good reason to believe that this particular contractor has never used these disciplines and it casts some doubts about his technical competence to perform on a software development contract.

50.5 Life cycle costs. At the beginning of every new software system there is a tendency to become completely absorbed with the cost to develop that software. But more significant than development costs are the software's life cycle costs. Life cycle costs consists of development costs plus life cycle support or maintenance costs. It might take two to three years to develop the software but the software is likely to be in use, and require support, for something in excess of 20 years. During that time the software must remain viable and responsive to changing operational environments and requirements. Best estimates available conservatively indicate that development represents 25% of total life cycle costs while the support or maintenance phase represents the remaining 75%. Therefore, any actions taken during development to reduce life cycle support costs will provide a three fold return to the customer.

50.6 Summary. Because the Standard only specifies the minimum set of requirements, there is little chance an AM will ever pay for anything he does not need: work, documentation, deliverables, software quality assurance, software configuration management, or state-of-the-art software engineering. Any statement suggesting that the use of the Standard would result in software development costing more than it otherwise would, must be challenged for validity. All of the Standard's requirements are intended to improve maintainability of the software for the remainder of its life cycle. It is all too easy to fall into the trap of waiving requirements to save money during development which will ultimately cost the customer money during the life cycle support phase. AMs must maintain their integrity and the integrity of the software system by implementing all of the Standard's requirements.

# APPENDIX C

## PROCUREMENT PACKAGE GUIDANCE

10. GENERAL. This Appendix provides guidance in the preparation of a Statement of Work (SOW) which is part of a solicitation for proposals leading to a contract that involves software development. The SOW invokes DOD-STD-1679A in the contract. The paragraph numbers contained within the Sections of this Appendix correspond to those that are used in the SOW. Items enclosed in brackets require that the contracting activity to provide the information discussed. Section 1 of the SOW is entitled "Scope". This section briefly describes the background, tasking, and facilities for the work to be performed under the software development contract.

1.1 Background. [A background description shall be included to provide the contractor with an overview and indication of the scope of the software development portion of the contract.]

1.1.1 Operational environment. [Describe the operational environment in which the new system is to be installed. It should include a top level description of the operational missions, especially those missions which the new system will perform or support.]

1.1.1.1 System description. [Describe the system hardware which is to be exercised and controlled by the software. Known, future enhancements should be identified and included here.]

1.1.1.2 Requirements description. [Identify all the existing documents which specify the new system's requirements. This identification should be a narrative description and should provide an overview of the requirements documents, rather than specifically naming the documents. The documents are specified formally in Section 2 of the SOW.]

1.1.1.3 Statement of Work (SOW) description. [Provide an overview of the SOW by briefly describing the major tasks of the software development effort. Each task should be described in a short sentence.]

1.1.1.4 Description of supporting contractor(s). [Provide an overview of all other contracts that will be interfacing with the software development and their relationship to this contract. Examples of such contracts include the manufacture of hardware, hardware integration, independent verification and validation effort, and the

system integration. Each contract should be a separate paragraph and numbered accordingly. If the contract has already been awarded, name the successful vendor.]

1.1.2 Contracting activity. [Identify the Government office that is to be designated the "contracting activity" as defined in Paragraph 3.3 of the Standard. The designated organization should have both administrative and technical responsibility for the contract.]

1.2 Tasking. [This Section introduces the major tasks with a brief, overview description. Each task should also be given a short name description.]

1.3 Facilities. [Identify any constraints, conditions, or restrictions to be placed on the contractor regarding the facilities to be used for performing on this contract. It might be desirous to develop the software utilizing a specific software development environment. The AM might prefer the use of their own facilities to reduce development costs. Another constraint might be that all of the contractor's technical effort must be performed in one general locale, e.g., within approximately one hour's travel time of the AM's primary facility for this contract. Care should be exercised to allow for exceptions to the contract in order to provide for work to be performed at a system integrated test site or the ultimate operational environment. This might include stress testing or final acceptance testing.]

1.3.1 Life cycle software support environment. The [identify the life cycle software support environment] will be used for life cycle software support of the software developed or delivered under this contract. The contractor shall verify and warrant the supportability within this life cycle software support environment of all software developed or delivered under this contract. [Describe herein, or reference, the current life cycle software support environment. Identify or list each item (facilities, equipment, software, documentation) by nomenclature, specification number, drawing number, revision, release level, publication dates, or other appropriate identifiers. It is important here and in the following paragraphs to identify those resources that are not available or will have restrictions on their access or use, in order to avoid future contract negotiation or source selection problems, and avoid wasted efforts and unnecessary problems for the designated life cycle software support activity.]

1.3.1.1 Contracting activity furnished resources. The following life cycle software support items are required for use by the contractor, and are available for delivery to the contractor. The contractor shall integrate the required items into the software development support environment. The use of other support items as substitutes, or the non-use of contracting activity furnished resources, shall require approval by the contracting activity prior to implemen-

tation or use. The contractor shall identify the proposed use (or non-use) of the contracting activity furnished resources and the proposed alternatives. The contractor shall also identify the costs and benefits expected to accrue to the contracting activity through the contractor's proposed use (or non-use) of these resources.

[Identify each item (facilities, equipment, software, documentation) by nomenclature, specification number, drawing number, part number, revision, release level, publication date, or other appropriate identifiers.]

1.3.1.2 Contracting activity identified resources. The following commercially available items and procedures are required to be used by the contractor as part of the software development support environment. The contractor shall accomplish the necessary contractual arrangements and other management agreements to obtain and integrate these items into the software development support environment.

[Identify each item (facilities, equipment, software, documentation) by nomenclature, specification number, drawing number, part number, revision, release level, publication date, or other appropriate identifiers.]

1.3.1.3 Identification of requirements for additional contracting activity resources. The contractor shall identify requirements for contracting activity furnished resources in addition to those identified in Paragraph 1.3.1.1 above. The contractor shall identify the costs and benefits expected to accrue to the contracting activity through the contractor's use of these additional resources. The requirement on the part of the contractor to have access to or use of any additional items of the contracting activity's resources is subject to approval by the contracting activity prior to implementation or use. For those additional items that are approved for implementation or use, the contracting activity retains the option to furnish the additional resources or to require, through this contract, the contractor to obtain them.

1.3.1.4 Disclosure or use of contracting activity furnished documentation and computer software. The following limitations or restrictions apply to the resources identified in Paragraph 1.3.1.1 above. The contractor shall ensure that its employees and any subcontractors or vendors comply with these restrictions and limitations. The contractor shall protect all contracting activity furnished resources from unauthorized disclosure or release.

[Identify each item (facilities, equipment, software, documentation) by nomenclature, specification number, drawing number, part number, revision, release level, publication date, or other appropriate identifiers. For each item, identify the type of limitation or restriction and include herein or reference the documents or agreements that limit or restrict the disclosure or use of that item.]

1.3.2 Access of the contracting activity to contractor facilities. The contractor shall provide and arrange access to the contractor and subcontractor facilities to personnel designated by the contracting activity for purposes of assessing and monitoring the progress of the contracted effort. These personnel may be Government employees or agents of the contracting activity employed to assist in the management of the contracted effort. The contractor shall provide technical support and assistance as necessary to assist these personnel in accessing the necessary information and in developing assessments of the contract progress.

1.3.3 Access of the contractor to the designated Life Cycle Software Support Center. [Identify here the limits and restrictions on contractor access, clearances, coordination requirements, notification lead times and arrangements. Include provisions and limitations for both physical and electronic (remote) access. State conditions under which changes in arrangements are to be processed or under which denial of access may be invoked.]

1.3.4 Access of the contractor to supporting contractor activities. The contractor shall ensure access and coordination arrangements with the supporting contractors identified in Paragraph 1.2.3 of this Statement of Work as necessary to perform the tasks required by this Statement of Work.

1.4 Visits to organizational units. [Separate and specific provision shall be made for a procedure which will permit the contractor to apply for visits to organizational units which are to receive the system being developed. If the operational environment is located at a remote site or on a mobile platform, the AM should maintain the option to provide transportation to the site.]

20. Section 2.0, Applicable Documents. The specifications, Contract Data Requirements List (CDRL), Delivery Schedule, military specifications and standards, Data Item Descriptions (DIDs), contracting activity approved portions of the contractor's proposal, contracting activity approved planning documents, and other documents, to the extent that they are referenced in this Statement of Work, further define the work to be performed under this contract. In particular, the Delivery Schedule and the CDRL define the requisite Periods of Performance and delivery dates applicable to all Statement of Work defined tasks and their products.

[All documents invoked in Section 3 of this Statement of Work must be listed in this section by document number and title. DoD and Departmental instructions shall not be cited in the Statement of Work to control contractor effort. Any document listed in this section must be invoked and selectively tailored to meet minimal needs in Section 3.]

2.1 Government documents. The following documents of the issue in effect on date of invitation for bids or request for proposal, form a part of this Statement of Work to the extent described herein.

SPECIFICATIONS:

Federal

Military

Other Government activity

[Insert number, title, and date of all applicable System, System Segment, or Performance Specifications].

STANDARDS:

Federal

Military

Other Government activity

DRAWINGS:

List by number and revision. (Where detailed drawings referred to in a document are listed on an assembly drawing, it is only necessary to list the assembly drawing.)

OTHER PUBLICATIONS:
Manuals
Handbooks
Technical Orders
Bulletins
Regulations,etc.

2.2 Non-government documents. The following documents form a part of this Statement of Work to the extent described herein. Unless otherwise indicated, the issue in effect on date of invitation for bids or request for proposal shall apply.

SPECIFICATIONS:
STANDARDS:
DRAWINGS:
OTHER PUBLICATIONS:

30. REQUIREMENTS. [This Section describes in detail the requirements, conditions, and constraints which the contract is to place on the contractor. This is the detailed description from which the contractor must develop the software product. If there is an error in the requirements stated in this section, chances are good that the error will be built into the software product. And if there is an omission in this Section, the contractor will use his discretion during development. The contractor's choice of actions will be biased by a desire to reduce his costs and not necessarily in the best interests of building a better software product. It is this section of the SOW which is so critical to developing a quality software product. Quality software starts with complete requirements. The discussion of the detailed requirements of a SOW will be presented in what follows by way of a model SOW for a hypothetical System development which includes software development. It should be noted that this SOW is intended only to serve as a tool for providing guidance and illustration on the proper use of DOD-STD-1679A in contracts that involve software development. No portion of this illustration should be used directly on a real contract without careful consideration in each instance.]

## 3. REQUIREMENTS

3.0 Requirements. The contractor shall perform the following required tasks. The contractor shall ensure that all software and associated computer resources covered by this Statement of Work are designed, developed, integrated, tested, documented, installed, and supported according to the requirements of this Statement of Work.

3.1 Management planning. The contractor shall develop the following plans, perform the following reviews and monitor the use of computer resources as required in Paragraph 5.12 and Subparagraphs of DOD-STD-1679A. The contractor shall coordinate and perform the tasks as described below.

3.1.1 Management plans. The contractor shall develop the management plans necessary to implement and coordinate the efforts required by this Statement of Work. The contractor shall provide the necessary qualified personnel, facilities, materials, and services required to develop, implement and maintain the planning documents listed below. The contractor shall ensure that all comments and requirements identified by the contracting activity are incorporated

into the plans and the contractor's planning process. The contractor shall implement the plans as approved by the contracting activity. The contractor shall maintain the currency of these documents after approval of the contracting activity and shall submit all changes for contracting activity approval prior to implementation or use.

3.1.1.1 Software Development Plan. The contractor shall prepare a Software Development Plan (Contract Data Requirements List, DD Form 1423, Sequence No._____) in accordance with DI-A-2176A and DOD-STD-1679A, and submit for contracting activity review and approval on or before the date specified in the Delivery Schedule. The contractor shall formulate the Plan to apply to the development of all software as defined in Paragraph 3.20 of DOD-STD-1679A. The contractor shall describe in the Plan the efforts and techniques necessary to satisfy the requirements of this Statement of Work. The contractor shall include the necessary references to, and ensure compatibility with, the Software Development Support Environment Plan.

3.1.1.2 Software Quality Assurance Plan. The contractor shall prepare a Software Quality Assurance Plan (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-R-2174A and DOD-STD-1679A, and submit for contracting activity review and approval on or before the date specified in the Delivery Schedule. The contractor shall describe the organization, policies and procedures the contractor intends to use to ensure that the delivered software complies with the requirements of the Statement of Work. These procedures shall include the means by which the contractor's organization will monitor the contractor's compliance with the development and engineering practices described in the Software Development Plan. This plan shall identify and document all tools, techniques, methodologies, records, standards, practices and procedures to be employed in the implementation and management of the quality assurance program. Each item shall be defined and then its use shall be described for satisfying or augmenting the requirements of the quality assurance program. The plan shall also identify methods to be used for the detection, reporting, analysis, and correction of software errors.

3.1.1.3 Software Configuration Management Plan. The contractor shall prepare a Software Configuration Management Plan (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2035B and DOD-STD-1679A, and submit for contracting activity review and approval on or before the date specified in the Delivery Schedule. The contractor shall describe the contractor's internal policies for conducting software configuration management throughout the contract effort, including transfer of the software support to the designated life cycle software support activity. The contractor shall describe all configuration management and interface control agreements with subcontractors and vendors.

3.1.1.4 Computer Program Test Plan. The contractor shall prepare a Computer Program Test Plan (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-T-2142A and DOD-STD-1679A, and submit for contracting activity review and approval on or before the date specified in the Delivery Schedule. The contractor shall prepare a separate test plan for each type of performance and acceptance testing included in this Statement of Work, to include software development and integration testing, system testing, operational testing, and acceptance testing. The contractor shall ensure that the test plans provide, for each program to be developed, at least one test specification as defined in Paragraph 3.2.2.1 of this Statement of Work.

3.1.2 Reviews. The contractor shall conduct the necessary reviews and audits to properly monitor and assess the progress of the software development and to review and assess the acceptability of the software.

3.1.2.1 Software documentation reviews. For all deliverable documents under this contract, the contractor shall conduct software documentation reviews in accordance with Paragraph 5.12.4 of DOD-STD-1679A. The contractor shall conduct joint contractor and contracting activity detailed reviews of the preliminary versions of all deliverable documents on or before the date specified in the Delivery Schedule. The contractor shall conduct joint contractor and contracting activity detailed reviews of the proposed final versions of all deliverable documents on or before the date specified in the Delivery Schedule. These reviews may or may not generate input from the contracting activity for the contractor to consider or take action upon. The final version of the documents shall take into account all contracting activity input provided during the review of the preliminary and final drafts of the documents. The contractor shall establish the time, place and agenda for each review. The contractor is encouraged to provide incremental delivery and segmented documents to facilitate preparation for the document review. The reviews will provide the contracting activity with an oversight of the contractor's intended performance and progress of technical activities on this contract. They will also provide a forum for resolution of any open issues prior to approval of the document by the contracting activity. The contractor shall publish and distribute minutes no later than seven calendar days after each review.

3.1.2.1.1 Approval of software documents. The contractor shall not be relieved of any responsibility for the proper design and implementation of the software as a result of the reviews. Comments by contracting activity participants at reviews (both oral and written) shall be considered as advisory or suggestive alternatives and not as direction. Contracting activity approval and direction will be pro-

vided by the Contracting Officer. Upon approval and acceptance of the documents by the contracting activity, the contractor shall maintain and use the documents throughout the life of the contract.

3.1.2.1.2 Processing of changes to software documents. The contractor shall provide recommended revisions or changes to the software documentation, together with supporting rationale. All proposed changes to the documents subsequent to approval of the final version shall require contracting activity approval prior to their implementation or use. The contractor shall provide any changes or revisions sufficiently in advance of the required implementation or use to allow contracting activity review, evaluation, and approval. The contractor shall publish and distribute all approved changes.

3.1.2.2 Software status reviews. The contractor shall conduct [insert the frequency desired by the contracting activity] status reviews in accordance with Paragraph 5.12.3 of DOD-STD-1679A. The contractor shall include in the presentation material a current organizational chart that incorporates the names of all key personnel and any key personnel changes. The contractor shall also present the status of all open action items as well as those resolved since the last review. The contractor shall publish and distribute minutes no later than seven calendar days after each review.

3.1.2.3 Independent software reviews. Routine verification and validation of all software to be developed, as well as various inspections and audits of this contracted effort, may be conducted by the contracting activity with assistance from an independent third party organization. The contracting activity may utilize the services of its internal organizations or the services of another contractor. The contractor shall support these efforts as directed by the contracting activity. The contracting activity designated third party shall have access to, but not necessarily limited to: software development data (including engineering design or analysis documentation), Software Trouble Reports, test documentation, status accounting information, and the software engineering practices and procedures. The access may be controlled by the contractor to the extent necessary to protect proprietary data and preclude access to other information not related to this contracted effort.

3.1.3 Computer resources monitoring. The contractor shall provide for the continuing capability to: a) define and monitor sizing and timing requirements; b) forecast and plan for resource availability and utilization; and c) control resource allocation and use. The contractor shall define computer memory, processing time, and input and output resource budgets and their projected utilization for the system in accordance with Paragraph 5.1.3 of DOD-STD-1679A. The contractor shall plan options for the resource allocations in all modes of operation to meet the performance requirements contained in

the system specifications. The target computer system(s) shall have at least a 20% total system reserve of the computer resources of storage (main and secondary) and processor throughput at the time of program acceptance by the contracting activity.

3.2 Documentation. The contractor shall develop and deliver the documentation as specified in the following paragraphs. The contractor shall ensure that the effort allocated to these tasks shall be only that which would not be incurred if the documentation were eliminated. All documentation shall be processed and be subject to contracting activity approval in accordance with Paragraph 3.1.2.1 and Subparagraphs of this Statement of Work.

3.2.1 Program documentation.

3.2.1.1 Program Performance Specification. For each program to be developed, the contractor shall prepare a preliminary draft and a final version of the Program Performance Specification (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2136B and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.1.2 Program Design Specification. For each Program Performance Specification, the contractor shall prepare a preliminary draft and a final version of the Program Design Specification (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2138A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.1.3 Interface Design Specification (interim). The contractor shall prepare a preliminary draft and final version of an interim Interface Design Specification (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2135A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule. Each interface identified as a result of the efforts required by Paragraph 3.5 of this Statement of Work shall comprise a separate volume in this single document, with each volume having its own group of sections.

3.2.1.4 Interface Design Specification (final). Utilizing the interim Interface Design Specification, the contractor shall prepare a preliminary draft and a final version of the Interface Design Specification (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2135A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.1.5 Data Base Design Document. The contractor shall prepare a preliminary draft and a final version of the Data Base Design Document (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2140A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule. Each data base identified as a result of the efforts required by Paragraph 3.6 of this Statement of Work shall comprise a separate volume in this single document, with each volume having its own group of sections.

3.2.1.6 Program Description Document. For each program, the contractor shall prepare a preliminary draft and a final version of the Program Description Document (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2139A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.1.7 Program Package Document. For each program to be delivered, the contractor shall prepare a Program Package Document (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2141A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule. The Program Package Document shall be expanded to include a load map for each program as required in Paragraph 5.4.13 of DOD-STD-1679A. The physical form of delivery for the source and object programs specified in Sections 3 and 4 of DID DI-E-2141A shall be: [Insert the specific delivery media and formatting instructions and constraints required by the contracting activity. Include requirements for psuedo, source, and object code media and format instructions.]. Use of other formats or types of media for delivery shall require contracting activity approval prior to implementation or use.

3.2.2 Test documentation.

3.2.2.1 Test specifications. For every related series of tests required to be conducted by the Computer Program Test Plans, the contractor shall prepare a preliminary draft and a final version of the Computer Program Test Specification (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2143A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.2.2 Test procedures. For each test specified to be conducted in the Computer Program Test Specifications, the contractor shall prepare a preliminary draft and a final version of the Computer Program Test Procedures (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-T-2144A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.2.3 Test reporting. For every test conducted in accordance with the approved Computer Program Test Procedures, the contractor shall prepare a Computer Program Test Report (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-T-2156A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.3 Software change documentation.

3.2.3.1 Software trouble reporting. For each error discovered during, and subsequent to, integration and developmental testing of the software, the contractor shall prepare a Software Trouble Report (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-E-2178A and DOD-STD-1679A. The Software Trouble Reports shall be processed in accordance with the requirements contained in the approved Software Development Plan and Software Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.3 of this Statement of Work.

3.2.3.2 Software Change Proposals. For each proposed change to software or documentation that has been placed under contractor internal or contracting activity configuration control, the contractor shall prepare a Software Change Proposal (Contract Data Requirements LIst, DD Form 1423, Sequence No. _____), in accordance with DI-E-2177A and DOD-STD-1679A. Software Change Proposals shall be processed in accordance with the requirements contained in the approved Software Development Plan and Software Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.3 of this Statement of Work.

3.2.4 User documentation.

3.2.4.1 Non-functional software operation. For each program, the contractor shall prepare a preliminary draft and a final version of the Operator's Manual (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-M-2145A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.4.2 Functional software operation. For each program, the contractor shall prepare a preliminary draft and a final version of the System Operator's Manual (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-M-2148A and DOD-STD-1679A and submit for contracting activity review on or before the date specified in the Delivery Schedule.

3.2.5 Management review documentation.

3.2.5.1 Conference minutes. For each management review required by Paragraph 3.1.2 of this Statement of Work, the contractor shall document the results of the reviews in Conference Minutes, (Contract Data Requirements List, DD Form 1423, Sequence No. _____), in accordance with DI-[insert proper DID identifier].

3.2.5.2 Progress reports. The contractor shall document the [insert the period of performance to be covered by each report] progress of the efforts required under this Statement of Work in Computer Software Project Status Reports (Contract Data Requirements List, DD Form 1423, Sequence No. _____) in accordance with DI-[insert proper DID identifier].

3.3 Software development. The contractor shall develop all software in accordance with the provisions of DOD-STD-1679A and this Statement of Work as described in the following tasks. The contractor shall implement, execute and maintain the contracting activity approved Software Development Plan developed under Paragraph 3.1.1.1 of this Statement of Work. This Software Development Plan shall be applied to all software developed or used in the performance of this contract. The contractor shall provide the necessary qualified personnel, facilities, materials and services (supplemented by Government Furnished Equipment (GFE) and Government Furnished Information (GFI) as listed in the contract) required to develop the software. The contractor shall use the high order language [insert language name] as defined in [insert language definitional reference] in designing and implementing all software developed for delivery under this contract. The contractor shall ensure that the translation of requirements into design and implemented software is compatible and traceable.

3.4 Software development support environment implementation. The contractor shall implement, use, and maintain a software development support environment that meets the requirements of DOD-STD-1679A. The contractor shall obtain contracting activity approval of the software development support environment prior to its implementation or use in developing software to be delivered under this contract. All software to be delivered or used for the performance of this contract shall be developed and maintained within this environment. All operations and support requirements identified by the contracting activity shall be reconciled with the software development support environment. The environment shall provide a full range of engineering support services and shall address, as a minimum, the requirements specified in the following paragraphs:

3.4.1 Interface with the life cycle software support environment. The contractor shall identify the interfaces of the software development support environment with the life cycle software support environment designated in Paragraph 1.3.1. The contractor shall identify all items in the software development support environment that are not contained in the designated life cycle software support environ-

ment and recommend the disposition of these items. The software development support environment shall satisfy all specified requirements and shall be fully compatible with the designated life cycle software support environment.

3.4.2 Software development support environment commonality. The contractor shall design the software development support environment to make maximum use of the existing capabilities and resources of the designated life cycle software support environment. Unless otherwise approved by the contracting activity, hardware and software development items such as compilers, linkers, emulators, and microprocessor development stations, shall be used that are the same or functionally equivalent to those currently owned or licensed by the designated life cycle software support activity.

3.4.3 Software development support environment contents. The software development support environment shall be designed to provide, as a minimum, a set of defined user and system interfaces, a set of software support tools, and a central library for the storage of the software and all information associated with the development and support of the software over its life cycle. The software development support environment shall provide for storage of software both in a source form and in a form that has been compiled for a host computer or a particular operational computer. The software development support environment shall include a control language which presents an interface to the user and to the information in the central library. The software support tools shall include tools for software development, maintenance, test, and configuration control. The software development support environment shall support the functions of project management, documentation, and release control. All contracting activity specified data bases, tools, interfaces, and procedures shall be integrated into the environment.

3.4.4 Software development support environment operation. Procedures and controls for access, generation, and change of all items in the software development support environment shall be established. As a minimum, the DOD-STD-1679A requirements for library usage and control, software generation, software operation, and software trouble reporting shall be included and shall be applied to all items in the software development support environment. All differences between the software development support environment and the designated life cycle software support environment shall be described. Procedures to be used to ensure the compatibility of operations between the two environments shall also be described.

3.5 Software performance requirements. The contractor shall analyze and completely define the detailed performance requirements for all software as required by Paragraph 5.1 and its Subparagraphs of DOD-STD-1679A. This analysis and the resulting requirements shall further detail the software performance requirements that are specified in Section 2 of this Statement of Work. This analysis shall be

performed in accordance with the contracting activity approved tools, techniques and methodology that were presented in the Software Development Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work. The contractor shall evaluate the interfaces within the system and within and among the software. The contractor shall include all interfaces required for system or software operation, calibration, data measurement and collection, test, acceptance, verification and support. The contractor shall define or refine each digital interface, including communication protocols and formats. The contractor shall ensure that the performance and interface requirements are directly and clearly traceable to the system level requirements documents and any related requirements analyses or design tradeoffs that result in the allocation of performance requirements to the software and associated computer resources. The contractor shall document the results of these efforts for programs and interfaces to be developed in Program Performance Specifications and Interface Design Specification developed under Paragraphs 3.2.1.1 and 3.2.1.3 of this Statement of Work.

3.6 Software design. The contractor shall develop the detailed design of the programs, their interfaces, and their data bases in accordance with Paragraph 5.2 and its Subparagraphs of DOD-STD-1679A. The contractor shall use the program design language PDL [insert the name of the required PDL] for this effort. For each program and interface that results from the efforts of Paragraph 3.5 above, the contractor shall create a design that completely satisfies the requirements contained in the Program Performance Specifications and Interface Design Specification that were developed under Paragraphs 3.2.1.1 and 3.2.1.3 of this Statement of Work. The contractor shall ensure that all software design information is directly and clearly traceable to the requirements in the Program Performance Specifications and Interface Design Specification. For each program, the contractor shall analyze and complete the identification of all program data that will be used by two or more subprograms. This design effort shall be performed in accordance with the contracting activity approved tools, techniques, and methodology that were presented in the Software Development Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work. The contractor shall document the results of these efforts in the Program Design Specifications, Interface Design Specification, and Data Base Design Document developed under Paragraphs 3.2.1.2, 3.2.1.4, and 3.2.1.5 of this Statement of Work.

3.7 Software implementation. The contractor shall implement all software in accordance with Paragraphs 5.3, 5.4, and 5.5 and Subparagraphs of DOD-STD-1679A and this Statement of Work. All software engineering practices, procedures, standards and conventions used shall strictly conform to DOD-STD-1679A and shall be defined in the contracting activity approved Software Development Plan that was de-

veloped under Paragraphs 3.1.1.1 of this Statement of Work. Any software codes, instructions or equipment functions associated with any computer equipment that the manufacturer or supplier has designated as unassigned, reserved, or restricted shall not be used.

3.7.1 Subprogram description. For each subprogram, the contractor shall create a detailed design that completely satisfies the requirements contained in that program's Program Design Specification, Interface Design Specification and Data Base Design Document that were developed under Paragraphs 3.2.1.2, 3.2.1.4, and 3.2.1.5 of this Statement of Work. This detailed design effort shall be performed in accordance with DOD-STD-1679A and the contracting activity approved methodology that was presented in the Software Development Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work. The contractor shall document the results of these efforts in the Program Description Documents and Program Package Documents developed under Paragraphs 3.2.1.6 and 3.2.1.7 of this Statement of Work.

3.7.2 Programming. The contractor shall provide the necessary qualified personnel, facilities, materials and services required to program and debug the software as defined in the approved design documentation. This effort shall be performed in accordance with DOD-STD-1679A and the contracting activity approved methodology that was presented in the Software Development Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work. The contractor shall program all software in the specified high order language.

3.7.3 Program package description. The contractor shall prepare each program for delivery in accordance with Paragraphs 5.4.11, 5.4.12, and 5.4.13 of DOD-STD-1679A. The contractor shall document these efforts in the Program Package Document developed under Paragraph 3.2.1.7 of this Statement of Work.

3.7.4 Commercially available or privately developed software. The contractor shall deliver all commercially available or privately developed software in accordance with DOD-STD-1467 and the Software Development Plan developed under Paragraph 3.1.1.1 of this Statement of Work. When previously approved by the contracting activity as an alternative, documentation of commercially available or existing privately developed software shall be delivered in accordance with [insert proper DID identifier] (Contract Data Requirements List, DD Form 1423, Sequence No. _____) and this Statement of Work.

3.7.5 Transition of software support responsibility. The contractor shall plan and implement the efforts necessary to transition responsibility for the support of all software developed under this contract to the designated life cycle software support activity. The contractor shall assist the designated life cycle software support activity as necessary in accomplishing the transition. The efforts shall address, as a minimum, the tasks described in the following paragraphs.

3.7.5.1 Additions to the life cycle software support activity. The contractor shall identify all software and procedures that must be added to the life cycle software support environment to properly support the operational software throughout its life cycle. The contractor shall describe how the resulting life cycle software support environment will be used to support the operational software.

3.7.5.2 Implementation of additions to the life cycle software support activity. The contractor shall plan for the transfer of support software and documentation to the life cycle software support activity. This effort shall be designed to ensure a phased transfer without loss or degradation of the support required for the operational software or to other tasks performed by the life cycle software support activity. The requirement to use any of the life cycle software support activity's resources during this transfer phase shall be identified. Sufficient lead-time to ensure the completion of the transfer prior to the planned assumption of operational software support responsibilities by the life cycle software support activity shall be planned. The contractor shall assist the life cycle software support activity as necessary in the transition of software support responsibility from the contractor to the life cycle software support activity.

3.7.5.3 Access to the software development support environment. The contractor shall provide access to personnel designated by the contracting activity, prior to the transition effort, for purposes of familiarization and training. The intent of this requirement is to prepare life cycle software support activity personnel to assume their software support responsibilities. The contractor shall provide technical support and assistance necessary to assist these personnel in understanding the operation of the software development support environment and how its operation compares to that of the life cycle software support environment.

3.7.5.4 Operation of the life cycle software support environment. The contractor shall verify that the added support software properly executes in the life cycle software support environment and that the software is capable of being evaluated, generated, installed, integrated, tested, and modified using the environment. The contractor shall verify that the operational software produces identical results when executed in the operational environment, in the contractor's software development support environment, and in the designated life cycle software support environment.

3.7.5.5 Technical support to the life cycle software support activity. The contractor shall provide technical assistance and analysis required by the life cycle software support activity to resolve any problems encountered accepting software support responsibility and in operating the life cycle software support environment. The contractor shall provide necessary technical assistance and engineer-

ing services to assist the life cycle software support activity in properly developing and implementing changes to the operational software. This service shall be provided during the transition period and for [insert the required period] after acceptance of the operational software support responsibility by the life cycle software support activity.

3.8 Software integration and development testing. The contractor shall conduct all software integration and development testing in accordance with the requirements of Paragraph 5.8 and Subparagraphs of DOD-STD-1679A and the contracting activity approved Software Development Plan and Computer Program Test Plans that were developed under Paragraphs 3.1.1.1 and 3.1.1.4 of this Statement of Work. The contractor shall report and correct all errors discovered in the course of the testing. The contractor shall document the errors and proposed changes in Software Trouble Reports and Software Change Proposals required under Paragraph 3.2.3 of this Statement of Work.

3.8.1 Software test requirements. Software testing is not to be structured to demonstrate software capabilities as implemented, but, rather, structured to demonstrate satisfaction of the approved performance requirements, and to ensure that the software does not implement performance requirements that are not specified. The contractor shall design and develop all software test documentation exclusively to prove the satisfaction of the performance requirements contained in the approved Program Performance Specifications and the approved Interface Design Specifications that were developed under Paragraphs 3.2.1.1 and 3.2.1.4 of this Statement of Work. No other source of performance requirements, information, or data shall be used in developing the test requirements to be included in the software test documentation.

3.8.2 Software test conduct. All software testing shall be strictly conducted in accordance with the contracting activity approved Computer Program Test Plans, Computer Program Test Specifications, and Computer Program Test Procedures developed under Paragraphs 3.1.1.4 and 3.2.2 of this Statement of Work. Test Reports and Software Trouble Reports shall be prepared in accordance with Paragraphs 3.2.2.3 and 3.2.3.1 of this Statement of Work. The conduct of the software integration and developmental testing shall conform to the requirements of the Delivery Schedule.

3.9 System integration. (When applicable) The contractor shall integrate the software as required by [insert the specification number and title of the Type A System Specification, or equivalent]. The contractor shall ensure that the software performs as required to fulfill all requirements of the system specification. The contractor shall correct any incompatibilities among the programs and their interfaces with the other elements of the system.

3.10 System test and evaluation. (When applicable) The contractor shall support [or conduct] the system test and evaluation program as defined in the system specification, and the contracting activity approved Computer Program Test Plans, Computer Program Test Specifications, and Computer Program Test Procedures that were developed under Paragraphs 3.1.1.4 and 3.2.2 of this Statement of Work. The contractor shall prepare Test Reports and Software Trouble Reports in accordance with Paragraphs 3.2.2.3 and 3.2.3.1 of this Statement of Work. The conduct of the system testing shall conform to the requirements of the Delivery Schedule. This effort includes the detailed planning, conduct, resources, and support of the tests necessary to validate that the system performance requirements applicable to the software have been implemented and achieved, and that the implemented software is adequate for operational use.

3.11 Operational test and evaluation. (When applicable) The contractor shall support the operational test and evaluation program as defined in the system specification and the contracting activity approved Computer Program Test Plans, Computer Program Test Specifications, and Computer Program Test Procedures that were developed under Paragraphs 3.1.1.4 and 3.2.2 of this Statement of Work. The contractor shall prepare Test Reports and Software Trouble Reports in accordance with Paragraphs 3.2.2.3 and 3.2.3.1 of this Statement of Work. The contractor shall provide a continuing availability of personnel, equipment maintenance, software support, operators and other services, plus all specified documentation and supplies as referenced in the system specification and the contracting activity approved contractual test documentation.

3.12 Software acceptance. The contractor shall perform the efforts necessary to satisfy the software acceptance requirements of Paragraph 5.10 and Subparagraphs of DOD-STD-1679A. These requirements are amplified in the following paragraphs, and are the minimum criteria to be satisfied prior to successful completion of the contract. The contractor shall integrate and implement these tasks as necessary with the other tasks in this Statement of Work to ensure the proper phasing and completion of the acceptance efforts.

3.12.1 Software acceptance testing. Acceptance of the software to be delivered requires the successful completion of acceptance testing in the four areas listed in the following paragraphs. The contractor shall develop and deliver a complete set of test documentation, as described in Paragraph 3.2.2 and Subparagraphs of this Statement of Work, for each of these tests. The contractor shall furnish the qualified personnel, facilities, resources and services to plan, develop, implement, and support as necessary this acceptance testing. The contractor shall coordinate with and assist the contracting activity as necessary in the scheduling and conduct of the tests. The contractor shall provide necessary support to assist the contracting activity in the evaluation of the test results. The contractor shall resolve any discrepancies or action items that result.

3.12.1.1 Stress test. The contractor shall support the Stress Test of the software as required in Paragraph 5.10.2 an Subparagraphs of DOD-STD-1679A, at [insert the location(s) to be used for the stress testing]. The contractor shall design the stress test specification to include typical operational environments and the full range of operational modes and situations, plus any other situations expected to be encountered during system operation, that are considered necessary to fulfill the stress test requirements of DOD-STD-1679A. The contractor shall provide software support, operator support, documentation, and other services during the Stress Test and evaluation periods. The contractor shall design the test to utilize procedural documentation that is to be delivered under the contract and that has been approved by the contracting activity. Stress testing of support software developed for use only as part of the host computer system in the life cycle software support environment is not required.

3.12.1.2 Reduced capability software testing. The contractor shall identify and recommend for contracting activity approval, the reduced capability modes that can safely be accomplished. The contractor shall design this portion of the acceptance test to evaluate all reduced capability modes or combinations of modes designated by the contracting activity. The contractor shall ensure that the test is compatible with all operational performance requirements included in the approved system specification or other operational and support concept documents that are referenced in Section 2 of this Statement of Work. The contractor shall provide software support, operator support, documentation, and other services required during reduced capability software testing.

3.12.1.3 Test and maintenance software testing. The contractor shall design this portion of the acceptance test to exercise this software in as near a normal operating environment and situation as possible. The contractor shall ensure that these tests are designed to realistically track with the normal testing and maintenance procedures to be employed by the operating and support activities. The contractor shall identify and recommend for contracting activity approval, a representative set of test and maintenance situations, based on the collective judgements of the most likely system failure and recovery modes to be encountered. The contractor shall include as a minimum, the representative failures causing or leading to the reduced capability modes identified in the reduced capability software acceptance testing. The contractor shall provide software support, operator support, documentation, and other services required during test and maintenance software testing.

3.12.1.4 Performance testing. The contractor shall identify and develop a performance related subset of those tests used during developmental testing that can be used to validate the ability of the software to satisfy its performance requirements. The contractor

shall design the test to use, where possible, previous test approaches and documentation. The final scope of the tests shall require prior contracting activity approval and shall be dependent on the results achieved during development testing, the type and extent of problems encountered during that testing, and the nature and extent of corrective actions required as a result of that testing.

3.12.2 Supportability and compatibility testing. The contractor shall conduct supportability and compatibility testing necessary to fulfill the requirements identified in Paragraph 3.4 of this Statement of Work. In addition to the other requirements specified in the contract, final acceptance of the software development effort shall be predicated on: (a) A demonstrated compatibility of the contractor's software development support environment with the designated life cycle software support environment and (b) the demonstrated capability of the life cycle software support environment to perform life cycle software support for the operational software items of the system. The satisfaction of the compatibility requirements shall depend on the existence of the following conditions:

3.12.2.1 Operating capability. All software delivered by the contractor shall be capable of being evaluated, generated, installed, integrated, tested, and modified at the designated life cycle software support activity utilizing only the existing life cycle software support environment and support software that has been delivered by the contractor to the contracting activity.

3.12.2.2 Functional compatibility. All operations or functions that are accomplished in the contractor's software development support environment and have been identified by the contracting activity as required for use in the life cycle software support environment, shall be capable of being accomplished in the life cycle software support environment.

3.12.2.3 Equivalence of operations. All operations or functions that affect the configuration of the operational software shall produce identical results when performed in the contractor's software development support environment and in the life cycle software support environment.

3.12.3 Review of the test results. All testing is subject to final contracting activity reviews prior to acceptance of the test effort. The objective of the reviews is to substantiate that sufficient evidence has been obtained to determine that the software installed in the operational system will satisfy its specified requirements. The contractor shall schedule the reviews and submit schedules to the contracting activity sufficiently in advance of the required dates to allow for review and approval by the contracting activity. The contractor shall phase and segment the reviews as necessary to permit correction of any actions resulting from the reviews without disrupting the overall contract schedules. At least

one review shall be scheduled at the completion of all scheduled testing. The contractor shall make available all necessary test documentation, specifically the draft test reports, sufficiently in advance of the reviews to permit prior contracting activity review, evaluation and coordination on the contents of the documentation.

3.12.4 Software acceptance audits. The contracting activity may conduct periodic audits to determine the conformance of the delivered products and the contractor's efforts with the contract requirements. The contractor shall ensure the participation of the appropriate contractor personnel, subcontractors, and vendors as required by the contracting activity. The contractor shall provide the necessary resources, facilities, documentation, and services to allow the contracting activity to effectively perform the audits. The contractor shall accomplish all follow-on efforts required to correct and verify correction of any discrepancies that are identified during the audits. In addition to the audits required by Paragraph 5.12.6 of DOD-STD-1679A, the contracting activity may conduct, as a minimum, audits of the following items to determine their acceptability.

3.12.4.1 Computer resources reserves. The contractor shall provide a current status of the total system utilization of computer memory, processing time, and input/output channels. The contractor shall show the current utilization for all modes of operation as required by Paragraph 3.1.3 of this Statement of Work. The contractor shall define and measure the reserve capacity as specified in the Appendix of DOD-STD-1679A. The contractor shall provide any added technical documentation necessary to assist the contracting activity in determining that the computer resource capacity reserve requirements have been met.

3.12.4.2 Patches. The contractor shall provide current configuration status accounting records and a complete, documented history of all patches sufficient for the contracting activity to determine that the patch limits required by Paragraph 5.10.4 of DOD-STD-1679A have not been exceeded.

3.12.4.3 Errors. The error limits specified in Paragraph 5.10.3 of DOD-STD-1679A shall be used to determine the acceptability of the delivered software and its documentation. The contractor shall provide complete records for each item of software and documentation, including test reports, Software Trouble Reports, Software Change Proposals, configuration status accounting records, and the records of all previous reviews, inspections, and audits. The contractor shall provide an indexed, summary record showing all discrepancies or action items and their resolution status. The contractor shall provide an analysis of the unresolved software and documentation errors sufficient for the contracting activity to determine that the error limits have not been exceeded.

3.12.4.4 Status accounting records. The contracting activity may conduct independent audits of the contractor's configuration management system and records prior to conducting the acceptance audits listed above in order to determine the accuracy and currency of the information provided. All discrepancies noted during these audits shall be corrected by the contractor before the remaining acceptance audits are completed.

3.13 User documentation. The contractor shall define and document the procedures for system users and life cycle software support activity personnel to follow in loading, utilizing, initiating, and operating all software developed or delivered under this contract. The contractor shall ensure that the software generation and operation requirements of Paragraphs 5.6 and 5.7 of DOD-STD-1679A are included. The contractor shall ensure that all procedures for the operation of the life cycle software support environment completely describe all methods necessary to evaluate, generate, install, integrate, test, modify and operate the operational software.

3.14 Software configuration management. The contractor shall implement and maintain a software configuration management program in accordance with Paragraph 5.11 and Subparagraphs of DOD-STD-1679A and the contracting activity approved Software Configuration Management Plan that was developed under Paragraph 3.1.1.3 of this Statement of Work. The contractor shall apply this program to all software developed under this Statement of Work or used for the performance of this contract. The contractor shall reconcile and integrate the software configuration management requirements with the configuration management requirements addressing the total contracted effort.

3.14.1 Software configuration definition. The software documentation to be used to define the contractor's internal baselines shall be as defined in the contracting activity approved Software Development Plan and Software Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.3 of this Statement of Work. For all software that is to be developed as part of the requirements of this Statement of Work, all software documentation developed under this contract shall be used to define the contractor's internal developmental baseline. For contracting activity designated or furnished software, the set of documentation defined by the contracting activity shall be used for formal definition and baselining efforts. For commercially available or existing privately developed software, the set of documentation, as approved by the contracting activity and supplemented as necessary by the contractor, shall be used as reference documents for the contractor's configuration definition purposes.

3.14.2 Software changes. The contractor shall develop and implement methods of software change control that recognize and are tailored to both operational and support software as defined in the contracting activity approved Software Development Plan, and Software

Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.3 of this Statement of Work. In addition to the requirements of DOD-STD-1679A, the following requirements apply to the software from sources indicated:

3.14.2.1 Software that is to be developed. All software changes shall additionally identify the impact of the change on the software development support environment and the designated life cycle software support environment.

3.14.2.2 Software designated or furnished by the contracting activity. The contractor shall not make changes or modifications to any item of this software without prior approval by the contracting activity. The contractor shall include in the software configuration management program the procedures necessary to prevent unauthorized changes to this software.

3.14.2.3 Software that is commercially available. The contractor shall identify and resolve with subcontractors or vendors, any deficiencies or other changes that are required to this software. The contractor shall not make any change to this software without prior approval of the contracting activity. Any such changes shall be fully identified and documented. The contractor shall identify to the contracting activity any changes submitted or released by subcontractors or vendors that are not recommended for incorporation, with an assessment of the effects of non-incorporation. The contracting activity may designate subcontractor or vendor identified changes for incorporation and the contractor shall implement all changes designated by the contracting activity into this software.

3.14.2.4 Software that is privately developed. All changes proposed by the contractor to this software shall additionally identify the impact of the change on the operational software, the software development support environment, and the designated life cycle software support environment. The contractor shall identify and resolve with subcontractors, vendors, or other sources any deficiencies, conflicts or inconsistancies of this software with the operational system, the software development support environment, or the designated life cycle software support environment.

3.14.3 Software status accounting. The contractor shall develop and implement status accounting procedures that recognize and are tailored to both operational and support software as defined in the contracting activity approved Software Development Plan and Software Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.3 of this Statement of Work. The contractor shall ensure that the software development support environment is properly integrated into, and used as a part of, the status accounting procedures.

3.15 Software quality assurance. The contractor shall implement and maintain a software quality assurance program in accordance with Paragraph 5.9 and Subparagraphs of DOD-STD-1679A and the contracting activity approved Software Quality Assurance Plan that was developed under Paragraph 3.1.1.2 of this Statement of Work. The contractor shall apply this program to all software developed or delivered under this Statement of Work or used for the performance of this contract. The contractor shall ensure that the quality of all software, documentation and programming materials procured from subcontractors conform to the contract requirements. The contractor shall include in the software quality assurance program, as a minimum, the requirements of Paragraph 5.9 and Subparagraphs of DOD-STD-1679A and the tasks detailed in the following paragraphs.

3.15.1 Software quality assurance program. The contractor shall provide a software quality assurance program that, as a minimum, utilizes assessments, documentation reviews, design reviews, monitoring, and auditing to ensure compliance with contractual requirements. The program shall be applied to, but not be limited to, the following: software requirements; software design; software engineering standards, practices and procedures; software implementation; software documentation; software testing; software library controls; software configuration management; corrective action; and subcontractor performance.

3.15.2 Software quality assurance management. Software quality assurance management shall have sufficient, well defined responsibility and authority, and the organizational freedom to identify and evaluate quality problems and to initiate, recommend or provide solutions. The contractor shall regularly review the status and adequacy of the quality assurance program and adjust the program as necessary to ensure that its requirements and those of this Statement of Work will be satisfied. The contractor shall ensure that the organizational group responsible for this program's implementation has, as a minimum, corporate reporting responsibility external to and independent of the software developing or engineering group.

3.15.3 Reviews and audits. The contractor shall establish procedures for the preparation and execution of independent software quality assessment reviews and audits. The procedures shall include provisions for assessing the software's and its associated documentation's conformance with standards and technical or contractual requirements and for monitoring traceability of the original contractual performance requirements throughout the software development.

3.15.4 Reporting. The contractor shall document the results of all software quality assessment activities in established formats and promptly submit these results to the proper authority. These reports shall be available for review by the contracting activity. Failure to report and process discovered discrepancies as required by the

contracting activity approved Software Quality Assurance Plan and Software Development Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.2 of this Statement of Work will be considered as non-conformance to contractual requirements.

3.15.5 Detailed requirements. The contractor's software quality assurance program shall contain provisions for satisfaction of the detailed requirements set forth in the paragraphs below. These requirements shall be described and included in the contracting activity approved Software Quality Assurance Plan that was developed under Paragraph 3.1.1.2 of this Statement of Work.

3.15.5.1 Requirements assessments. The contractor shall conduct assessments of the software requirements process to ensure that the detailed software requirements are being determined and documented in accordance with the requirements of Paragraph 3.5 of this Statement of Work.

3.15.5.2 Software design. The contractor shall conduct assessments of the software design process to ensure that the software design is being developed and documented in accordance with the requirements of Paragraph 3.6 of this Statement of Work.

3.15.5.2.1 Architecture examination. The contractor shall ensure that the architecture of the software is examined for implementability, supportability, and the capability to support the computational work load. This examination shall be completed satisfactorily prior to the commencement of any program implementation.

3.15.5.2.2 Design walk-through. The contractor shall ensure the compliance to the practice of conducting internal design walk-throughs. A design walk-through of each portion of the software design must be completed prior to that portion of the design being implemented.

3.15.5.2.3 Computer system resources. The contractor shall monitor periodically the availability of computer system resources such as memory, processor time, and input/ output capacity as required in Paragraph 3.1.3 of this Statement of Work. The availability of these resources as contrasted against allocated budgets shall be reported as part of the contractor's status reporting system.

3.15.5.2.4 Program functional flow. The contractor shall independently assess the various types of graphical representations used to depict the flow of program data and control in all required modes of program operation for compliance with established standards.

3.15.5.3 Software engineering standards, practices, and procedures. The contractor shall monitor the utilization of a comprehensive set of software engineering standards, practices, and procedures as defined in the contracting activity approved Software Development

Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work. The program shall include assessments to ensure that these standards, practices, and procedures are being followed.

3.15.5.4 Software implementation. The contractor shall develop methods to ensure that all source code is audited to ensure compliance with the established software engineering standards, practices, procedures, and the programming standards and conventions as defined in Paragraph 3.7 and Subparagraphs of this Statement of Work and in the contracting activity approved Software Development Plan that was developed under Paragraph 3.1.1.1 of this Statement of Work.

3.15.5.5 Software Development Support Environment. The contractor shall conduct periodic audits of the software development process and procedures to ensure that the requirements for implementing the software development support environment and ensuring supportability of the software are capable of being satisfied. The contractor shall conduct audits to ensure that the software development support environment requirements defined in Paragraph 3.4 of this Statement of Work are being satisfied.

3.15.5.6 Software operation. The contractor shall conduct reviews to ensure that the procedures for the operation of the software and operation of the life cycle software support environment have been determined, described and documented in terms understandable to end user personnel, and are approved by the contracting activity prior to software delivery.

3.15.5.7 Software test reviews. The contractor shall ensure that reviews of all software test documentation are conducted and audit the test activities to ensure conformance with the: contract, software requirements specifications, established software engineering standards, practices, and procedures, and the specified software acceptance criteria. The contractor shall assess the documentation in three phases: 1) verify the Computer Program Test Plan(s) against this Statement of Work, the program performance requirements defined in Paragraph 3.5 of this Statement of Work, and the acceptance requirements defined in Paragraph 3.12 of this Statement of Work, 2) verify the Computer Program Test Specifications against the Computer Program Test Plans, and 3) verify the Computer Program Test Procedures against the Computer Program Test Specifications that are developed under Paragraph 3.2.2 and Subparagraphs of this Statement of Work.

3.15.5.8 Software test activities. The contractor shall provide procedures for the witnessing of the conduct of software development and integration testing. The contractor shall monitor the system development and operational testing activities. Included in the witnessing of the tests is assurance that the approved test proce-

dures are being followed, that accurate records of the tests are being kept, that all discrepancies discovered during the tests are being properly documented, and the certification of the associated test reports are submitted.

3.15.5.9 Deliverable items. The contractor shall monitor conformance with all software quality procedures to ensure contractual correctness of all deliverable items prior to their delivery.

3.15.5.10 Subcontractor software quality assessment audits. The contractor shall conduct periodic audits of subcontractors' software quality assurance program, plan, and execution thereof to ensure compliance with all contractual requirements.

3.15.5.11 Software acceptance. In addition to the performance, functional or usability criteria specified by this Statement of Work, software acceptance shall also be predicated upon satisfaction of the contractually specified computer system reserve requirements, successful completion of the software stress test, the limitations for the unresolved software and documentation errors, and the number of existing patch words, as required in Paragraphs 5.10 and Subparagraphs of DOD-STD-1679A. Satisfactory completion of the software development effort shall depend on the satisfactory implementation of a support capability for the operational software in the designated life cycle software support activity.

3.15.5.12 Software configuration management. The contractor shall conduct formal audits of the software configuration management functions to ensure compliance with the requirements of the contracting activity approved Software Development Plan and Software Configuration Management Plan that were developed under Paragraphs 3.1.1.1 and 3.1.1.4 of this Statement of Work. The contractor shall ensure the effective software configuration management of the developmental baseline from the time of contract award until final acceptance of the software and its associated documentation by the contracting activity. The contractor shall ensure that procedures for identifying and controlling software changes and maintaining the software change history are applied to all software to be developed or delivered under this contract and that these changes are properly integrated with the software development support environment. The contractor shall implement procedures to ensure the independent auditing of the status accounting system to assess effectiveness in tracking Software Trouble Reports. The contractor shall ensure that software configuration management procedures are integrated with the configuration management procedures addressing the total system acquisition.

3.15.5.13 Corrective action. The contractor shall establish a system for reporting, correcting, and analyzing problems or failures occurring during all phases of software development. The contractor shall provide for the conduct of periodic audits of this system to ensure its continuous effectiveness. The contractor shall perform periodic analyses of all Software Trouble Reports for the purpose of identifying trends which may disclose generic problem areas. The trend analysis shall include the study of the causes, magnitude of impact, frequency of occurrence, and preventive measures.



U.S. GOVERNMENT PRINTING OFFICE: 1988 - 505-038/20078