DOD-STD-1679A (Navy)
22 October 1983

SUPERSEDING
MIL-STD-1679 (Navy)
1 December 1978

MILITARY STANDARD

SOFTWARE DEVELOPMENT

AMSCN3211

AREA ECRS

DEPARTMENT OF DEFENSE

Software Development

DOD-STD-1679A (Navy)

1. This Military Standard is approved for use by the Department of the Navy and is available for use by all Departments and Agencies of the Department of Defense.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to:

   Chief of Naval Material
   ATTN: NAVMAT 08Y
   Washington, DC 20360

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

FOREWORD

1. This Standard contains requirements for the development of software which are applicable in Government contracts. A standard specifically addressing Government software is necessary because of factors concerning this software which are not common to general software, or which carry a significantly different degree of emphasis. Major factors are:

a. Criticality of performance. The combat capability of defense systems and the combat survivability of combatant units of the operating forces depend, in part, upon the effective operation of the software. Therefore, careful, persistent management must be exercised in the software development phase to ensure maximum reliability and maintainability. Special emphasis shall be placed on the accuracy and effective operation of the software.

b. Changing operational requirement. Software implements system operations and doctrine in areas susceptible to many changes of performance requirements. These changes often impact the software and need expeditious implementation. This demands that software be designed to facilitate efficient change, sometimes at the expense of technical design efficiency. Continuation of an efficient change capability over the operational life of the system also requires detailed documentation describing the software. Proposed changes and their total impact must be easily discernible and capable of being implemented by personnel not associated with the original development effort.

c. Life-cycle cost. Development and implementation of changes to the software over the operational life of the system are costly. The design of the software during development must be strongly influenced by factors which will reduce life cycle cost. Among these are various standardization requirements, such as those imposed upon program design, languages, and intersystem and intrasystem interfaces. An additional benefit of these standardization requirements is to ensure that changes developed and implemented in one system will have applicability in other systems.

2. Data Item Descriptions (DIDs) applicable to this Standard are listed in Section 6. These are essentially the same as previously used descriptions of data customarily required in connection with Government software development. The majority have been extracted from official military documentation standards. Others have been developed through the experience gained by military and commercial software developing activities.

CONTENTS

1. SCOPE

1.1 Purpose. This Standard establishes uniform minimum requirements for the development of software for the Department of Defense. Software developed under strict adherence to the provisions of this Standard will have been subjected to the highest degree of reliability and maintainability requirements feasible within the current state-of-the-art.

1.2 Application. When invoked in a specification or statement of work, these requirements shall apply to software that is developed either alone or as a portion of a system or subsystem. The contractor is responsible for invoking all the requirements of this Standard on any and all subcontractors employed by the contractor for the development of software.

2. REFERENCED DOCUMENTS

2.1 Issues of documents. None.

3. DEFINITIONS

3.1 Introduction. The definitions provided in this Section describe the terms as they are used in this Standard.

3.2 Code walk-through. (Sometimes referred to as a peer review.) A step-by-step, detailed examination of source code by a small group of qualified personnel.

3.3 Contracting activity. The contracting activity refers to that Government office, with contract and project directive administrative authority, which has prime responsibility for and authority over the software development effort.

3.4 Contractor. Contractor refers to any organization under contract or tasking agreement with the contracting activity to perform any part of the software development effort.

3.5 Design walk-through. A step-by-step detailed examination of the design of the software by a small group of qualified personnel.

3.6 Development, software. The engineering process and effort that results in software, encompassing the span of time from initiation of the contracted effort through delivery to and acceptance by the contracting activity.

3.7 Developmental baseline. The initial set of specifications and other documents formally approved by the contracting activity which define the requirements on the software being developed under contract. This baseline evolves throughout the software development cycle under the contractor's internal software configuration management and represents the definition of the software at any stage of its development. Once an element of the Developmental Baseline is formally approved or accepted by the contracting activity, that element remains a member of the Developmental Baseline but its change control comes under the purview of the contracting activity. Upon delivery to and final acceptance by the contracting activity, the Developmental Baseline becomes the Product Baseline under control of the contracting activity.

3.8 Documentation (or computer program documentation). The comprehensive written description of computer software in various formats and levels of detail that clearly define its content, composition, design, performance, testing, and use.

3.9 Error, documentation. An occurrence in the documentation which fails to reflect the operational requirements or accurately describe or support the software.

3.10 Error, intermittent. A software error which cannot be reproduced consistently when the same procedures and environment are duplicated.

3.11 Error, software. An occurrence, or lack thereof, during the execution of a program and attributable to the software that prevents satisfaction of the specified software requirements, fails to perform as designed, or performs a function not required and not desired.

3.12 Firmware. Microcode (software) which resides in computer memory that is not alterable by the computer system during program execution.

3.13 Flow chart. A graphic representation for the definition, analysis, or solution of a problem in which symbols are used to represent operations, data, flow, equipment, etc.

3.14 Graphic representations. A tool used to facilitate the translation of requirements into software design or software design into source code. They are intended to be comparable to blue prints for hardware. Examples are Program Design Languages (PDL/Ada (Ada is the trademark of the U.S. Government, Ada Joint Program Office), Structured English, Pseudo-Code, Pidgin-English, etc.), Functional Flow Diagrams, Block Diagrams, etc.

3.15 Parameters. Definable physical and logical characteristics, elements, or factors in software whose values determine the behavior of the software as it is being executed. The values assigned in the software to define these characteristics, elements, or factors remain constant throughout the system design and code execution. Sensor output limits, maximum range of defenses, maximum number of transactions handled, data storage limits, and maximum number of active devices are examples of parameters in the design and the implementation of the design by the code.

3.16 Patch. A change made to the object software after it has been assembled or compiled.

3.17 Program stop. A program stop is defined as any unspecified termination of program execution which requires that the software, or a portion thereof, be reloaded, restarted, or reinitialized.

3.18 Reduced capability mode. After experiencing the loss of one or more equipment items or equipment parts, system operation continues in a configuration which compensates for the losses but at a level below designed system capability.

3.19 Requirements walk-through. A step-by-step, detailed examination of the requirements of the software.

3.20 Software. A combination of associated computer programs and computer program data definitions required to enable the computer hardware to perform computational or control functions. Note: this definition includes firmware within its applicability. This definition of software is independent of the type of physical storage media in which the software resides. Software is categorized as follows:

3.20.1 Application. Software that implements the operational capabilities of the system.

3.20.2 Support. All software used to aid the development, testing and support of applications, systems, test and maintenance, and trainer software. Support software includes, but is not limited to:

a. Compilers, assemblers, linkage editors, builders, librarians and loaders required to generate machine code and to combine their hierarchical components into a complete computer program.

b. Debugging software.

c. Stimulation and simulation software.

d. Data extraction and data reduction software.

e. Software used for management control, software configuration management, or documentation generation and control during development.

f. Test software used in the software development.

g. Design aids, such as program design language tools, and problem statement analysis tools.

3.20.3 Systems. Software that is used to maximize the use of the computer resources at the time of use. Operating systems, executives, and data base management systems are examples of this type of software. Some operating systems run at development time; some run at use time.

3.20.4 Test and maintenance. Software provided as tools to assist in the fault diagnosis and isolation, operational readiness verification, and system alignment checkout of the system or its components. This software may be used to check out and certify the equipment and total system at installation, re-installation or after maintenance. Test and Maintenance software is also used periodically in accordance with prescribed procedures to maintain the system throughout its operational life.

3.20.5 Trainer. Software utilized to train users, operators, and maintenance personnel in the operation and maintenance of the system.

3.21 Software change proposal (SCP). A proposed change to existing software, its documentation, or its interfaces that is not a software trouble report (STR) and which would alter the approved baseline software or its attendant documentation which are under software configuration control.

3.22 Software hierarchy. Software can be hierarchically broken down into components of program, subprogram, module and unit (procedure or routine). A program refers to the totality of software in a system or one independent part of the software of a system, depending upon the complexity of any particular system. Subprogram refers to a major functional subset of a program and is made up of one or more modules. A module is an independently compilable software component. Modules are comprised of one or more units (procedures or routines).

3.23 Software trouble report (STR). A report that the software or its attendant documentation is not in conformance with the approved baseline documentation which is under software configuration control.

3.24 Use of "shall", "will", "should", and "may". "Shall" is used to express a provision that is binding; "should" and "may" are used to express nonmandatory provisions; "will" is used to express a declaration of purpose or intent.

4. GENERAL REQUIREMENTS

4.1 Software development management. The contractor shall plan and implement procedures to control the internal software development process and to provide visibility of progress and problems to contractor management and to the contracting activity. The contractor's organization shall be structured to provide positive control over development processes and resources utilized.

4.2 Design requirements. The contractor shall determine the software design, as tasked by the contracting activity. It is the responsibility of the contractor to ensure that the proposed design satisfies all specified software requirements. The design shall completely satisfy all requirements but shall not exceed the requirements without contracting activity approval. Design complexity and the interdependencies of subprograms, modules, and units shall be minimized.

4.3 Software implementation. All software shall be coded in a high order programming language (HOL) approved for use by the Department of Defense unless a specific waiver has been previously granted by the contracting activity. The software shall be capable of being generated by government owned support software and the contractually delivered support software.

4.4 Software quality assurance. The contractor shall plan, develop, and implement procedures and practices to ensure that all requirements of the contract, including this Standard, are complied with fully. The software quality assurance program shall be part of the management reporting system during all phases of software development. The software quality assurance program is only a part of and subordinate to the total software quality program. The fulfillment of the requirements of this Standard are not intended to be the responsibility of any single contractor organization, functional unit, or person. The contractor's software quality assurance program shall, as a minimum, utilize assessments, documentation reviews, design reviews, walk-throughs, monitoring, auditing, and testing to ensure compliance with contractual requirements.

4.5 Software configuration management. The contractor shall establish and implement the disciplines of software configuration management specifically; configuration identification, configuration control, and configuration status accounting. The contractor shall be cognizant of the requirement for long-term life cycle support of the software. Accordingly, the software configuration management procedures shall ensure complete and accurate correlation between descriptive documentation and the software in order to facilitate post-delivery maintenance by software support personnel.

The contractor shall plan and implement software configuration management procedures to; identify software elements requiring control, define and control change implementation processes, track development and change status, and to ensure documentation is changed to reflect current status of the software. This Standard contains the contractor's internal software configuration management requirements. These shall complement the contractor's associated requirements for interfacing with the contracting activity when proposed engineering changes effect contracting activity controlled software. The interface shall be in accordance with appropriate contract requirements.

4.6 Subcontractor control. The contractor is responsible for assuring that all software, documentation and programming materials procured from his subcontractors conform to the prime contract requirements. Therefore, this Standard shall be invoked on all subcontractors involved in the development of software under the contract.

4.7 Deviations and waivers. All software and software documentation shall be developed and delivered in complete conformance with all the requirements of this Standard, other applicable standards, and applicable contracting activity software documentation and specifications, as contractually invoked, unless a deviation or waiver has been previously processed and approved by the contracting activity. The extent of any variance from exact conformance to all applicable requirements shall only be that which is specifically authorized by formally approved deviations or waivers.

5. DETAILED REQUIREMENTS

5.1 Software performance requirements. The contractor shall determine the detailed performance requirements of the software. The contractor shall utilize the basic descriptive requirements and design information provided by the contracting activity to determine the software performance requirements. This information may be augmented by studies, analyses, visits to the end users, and surveys as necessary. The software performance requirements are subject to the review and approval of the contracting activity. (See 6.1)

5.1.1 Supporting information for software performance requirements. The contractor shall utilize, as a minimum, the following items, when available and applicable, to determine the software performance requirements:

a. System performance requirements.

b. System design requirements.

c. Equipment design requirements.

d. Interface design requirements.

e. Operational standards, doctrine, and tactics.

f. System design standards.

g. System readiness and maintainability requirements.

5.1.2 Software performance analysis. In determining the performance requirements, the contractor shall investigate and analyze in detail all areas relating to the performance requirements of the software and the system.

5.1.2.1 Mission areas. The contractor shall investigate the system's primary and secondary mission areas, as applicable, and supporting tasks of the system.

5.1.2.2 Functions. The contractor shall define the major functions or groupings of the software necessary to meet the system performance requirements.

5.1.2.3 Applicable documentation. The contractor shall identify all documents which define or constrain the software performance requirements. Definitions of applicable terms and abbreviations not consistent with or not included in Section 3 of this Standard shall be indicated and defined by the contractor.

5.1.2.4 System description. The contractor shall analyze the relationship of all components in the system which affect the software or the software performance requirements. The contractor shall determine how the software interfaces with other components to perform required system functions.

a. Peripheral equipment identification. The contractor shall identify all equipment with which the software will interface. The contractor shall identify the physical characteristics and type of all equipment interfaces.

b. Interface identification. The contractor shall identify all other computer programs or systems with which the software will interface. (See 6.1)

5.1.2.5 Functional description. The contractor shall analyze the major functions and the functional relationships of the software with all interfaces. The contractor shall describe the performance of each function supported by the software, its purpose, and functional design.

a. Equipment description. The contractor shall identify the requirements imposed on the software by each interfacing equipment, the purpose of the equipment, and the use of options and controls.

b. Diagrams. The contractor shall generate diagrams of equipment and software relationships with internal and external data flow.

c. Intersystem interface. The contractor shall identify the interfaces with other systems and equipments and shall be cognizant of the performance requirements of all systems which will interface with the system under development. Each contractor shall be cognizant of the purpose of the interface and the data to be exchanged. Data quantity, frequency, rate, format, content, scaling requirements and conventions shall be identified. In fulfilling this assignment, the contractor may be tasked to participate with other development contractors as a team to identify the intersystem interfaces so that the performance requirements of all systems are met. If interface conflicts are uncovered that adversely affect an individual system's ability to satisfy its performance requirements, the contractors shall recommend to the contracting activity the necessary modifications to the systems or their interface to overcome the deficiency. (See 6.1)

5.1.2.6 Detailed functional requirement. The contractor shall delineate the performance of each function by detailing its narrative, logical, and mathematical descriptions.

a. Inputs. The contractor shall define all inputs (external and internal), including their source, format, method of reception, quantity, timing, range, and scaling.

b. Processing. The contractor shall generate textual and, as appropriate, mathematical descriptions of the processing requirements of each function, including functional parameters and geometric diagrams.

c. Outputs. The contractor shall define all outputs (internal and external) including their method of transmission and timing, meaning, format, quantity, destinations, range, and scaling.

d. Special requirements. The contractor shall identify all requirements imposed by higher-level constraints or by exigencies of the function.

5.1.2.7 Functional verification. The contractor shall utilize functional analysis and modeling to conduct a verification of the data flows and functional operation (timing, capacities, communication rates, etc.) prior to submission of the software performance requirements for approval.

5.1.2.8 Changeability. The contractor shall identify those requirements that are most likely to change over the lifetime of the software.

5.1.2.9 Adaptive parameters. The contractor shall identify those parameters which reflect the system environment, limits, and capacities and which can be defined symbolically and subsequently modified without requiring an altering of the logic of the software function.

5.1.3 System resources. The contractor shall define the computer memory, computer processing time, and input and output resource budgets and their projected utilization for the system. If the system under development has more than one processor the contractor shall define these resource values for each processor. (see Appendix)

5.2 Software design. The contractor shall develop the detailed software design in accordance with the detailed software performance requirements approved by the contracting activity and shall comply with other design constraints and standards as specified by the procuring agency. The requirements shall be translated into a software design in a systematic top-down method. The design shall be a hierarchical structure of identifiable programs, subprograms, modules, and units. Software shall be divided into constituent parts and then those parts broken down into their constituents. Each level of design development (or breakdown) is continued until a level is reached where there is no subservient level. Levels shall be structured so that a level does not call on the same or a higher level. The contractor shall define the assumptions, the programming approach for implementing the software and shall define the software architecture. The proposed software architecture shall be verified as to its capability to support the computational load imposed by maximum operation of all functions required to be simultaneously serviced. This verification may require extensive modeling, prototyping, or simulation, and shall in all cases be completed prior to design implementation. The software design shall be subject to review and approval by the contracting activity. Prior to submission of the detailed design to the contracting activity for review, a design walk-through shall be conducted. This design walk-through shall be accomplished by one or more technically qualified persons in conjunction with the originator or originators of the detailed design. (See 6.1)

5.2.1 Supporting information for software design. The contractor shall utilize, as a minimum, the following items, when available and applicable, to determine the software design:

a. System operational design documents.

b. Software performance requirements.

c. Interface design requirements.

d. Programming reference manuals.

e. Equipment technical manuals.

f. Specified programming standards and conventions.

g. Specified utility/support software documents.

h. System self-test (diagnostics) and maintainability requirements.

5.2.2 Software design analysis. In determining the software design, the contractor shall investigate and analyze in detail the following areas relating to the software.

5.2.2.1 Applicable documentation. All documents which constrain, define, or influence the software design shall be analyzed. The contractor shall define all design terms and abbreviations used to describe the software design.

5.2.2.2 Functional allocation. The allocation of functions to be performed by the subprograms and their modules shall be defined. All performance requirements shall be satisfied in their entirety in this allocation.

5.2.2.3 Program functional flow. The flow of program data and control in all required modes of program operation shall be determined. A functional description of all inputs, outputs and processing for each subprogram and their modules shall be defined.

a. Program interrupt control. The source, purpose, type, predicted rate of occurrence, and required control response for each external and internal interrupt shall be determined from the analysis.

b. Subprogram reference control. The control logic, assignment of priorities and permissible cycle times for each subprogram shall be determined from the analysis.

c. Special control features. Unique control requirements which affect the design of the control logic shall be identified.

5.2.2.4 Resource allocation and reserves. Memory storage, input and output channels, and processing time requirements for each subprogram shall be determined. Total system memory, input and output channels, and processing time reserves of at least 20 percent shall exist at the time of software acceptance by the contracting activity. (Refer to the Appendix of this Standard)

5.2.2.5 Parameterization. The contractor shall, in the design of the software, provide for the use of symbols to represent all parameters, constants, and flags which can be modified without altering the logic of the source program.

5.2.2.6 Design constraints. The constraints of the specific programming language and support software to be used, which impact the capability of the contractor to meet the performance requirements of the software or the requirements of this Standard, shall be defined by the contractor.

5.2.2.7 Data base design. During the software design, the contractor shall completely identify all data used by two or more subprograms. (See 6.1)

5.2.3 Intersystem interface. The contractor shall determine the definition of the interfaces with other systems and shall be cognizant of the design requirements of all systems which will interface with the system under development. Each contractor shall be cognizant of the purpose of the interface and the data to be exchanged. Data quantity, frequency, rate, format, content, scaling requirements, and conventions shall be developed. In fulfilling this assignment, the contractor may be tasked to participate with other development contractors as a team to design the intersystem interfaces so that the performance requirements of all systems are met. If interface conflicts are uncovered such that an individual system's ability to perform in accordance with its requirements is adversely affected, the contractors shall recommend to the contracting activity the necessary modifications to the systems or their interface to overcome the deficiency. (See 6.1)

5.3 Programming standards. The following design and coding standards shall apply to the development of the software. (See 6.1)

5.3.1 Language. All software shall be coded in a high order programming language (HOL) approved by the Department of Defense unless a specific waiver has been previously granted by the contracting activity.

5.3.2 Control structures. Programs shall be designed and shall be coded using only five basic control structures presented in figures 1A - 1E. They are: the "sequence of operations" (assignment, add, ...), "if then else" (conditional branch to one of two operations and return), "do while" (operation repeated while a condition is true), "do until" (operation repeated until a condition is true) and "case" (program control transfers to one of several statements depending on the value of the control expression).

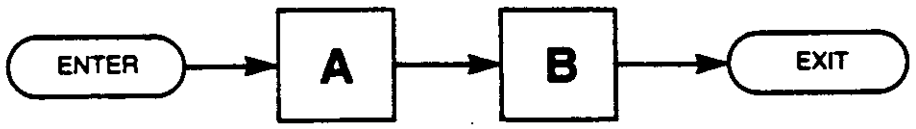
5.3.3 Included/copied segments. Included/copied segments shall only be written in a High Order Programming Language (HOL), unless a specific waiver has been previously granted by the procuring agency. Any program logic within a given structural segment shall utilize only those control structures specified in paragraph 5.3.2. When the segment contains executable statements, the entry to the segment shall be the first executable statement and the exit shall be the last executable statement.

5.3.4 Entry-Exit structure. Each unit or source code segment shall have a single entry and single exit structure. (see figure 1F)

5.3.5 Self-modification. Program self-modification of instructions during execution shall be prohibited.

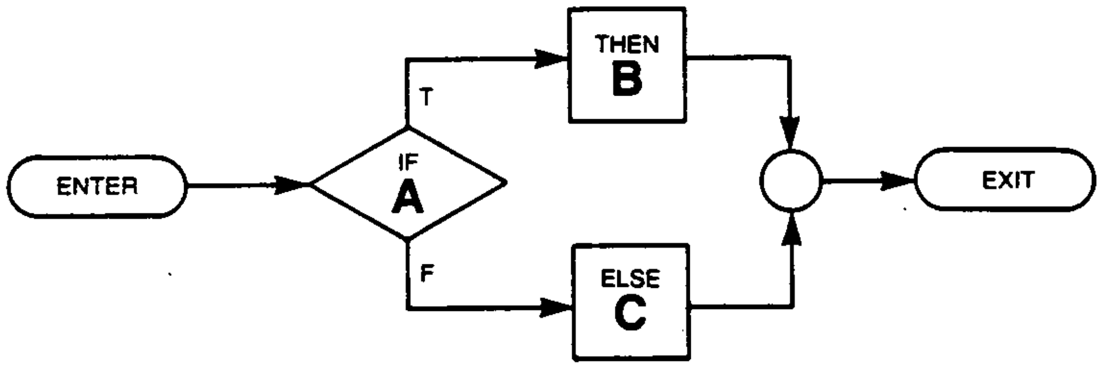
5.3.6 Size. Software units shall not exceed an average of one hundred executable source code statements per unit and shall not exceed a maximum of two hundred executable statements in any unit. Each independently executable source code statement, whether free-standing, part of a compound statement or in an included/copied segment counts as one executable statement.

FIGURE 1A. SEQUENCE: Process A, process B.



Control flows from process A to the next in sequence, process B.

FIGURE 1B. IF THEN ELSE: If condition A THEN process B, ELSE process C.



The flow of control will return to a common point after executing either process B or C. A predicates the conditional execution. If control is to skip a process pending the condition of A, then the flow chart can be modified thusly:

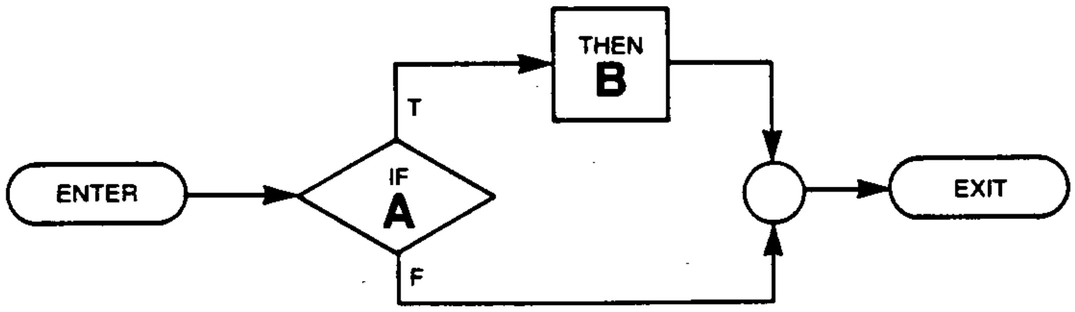
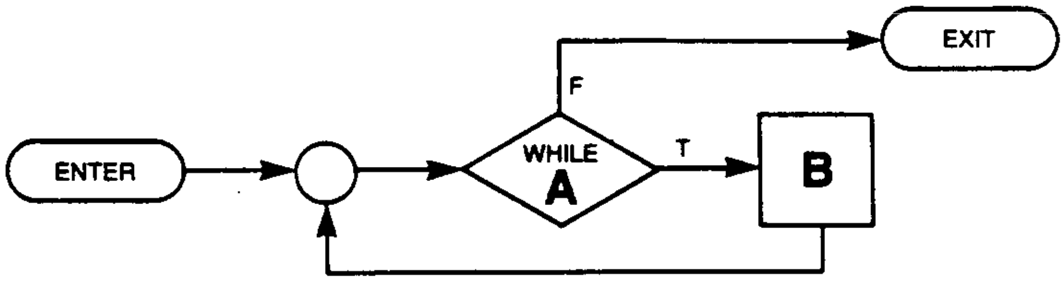


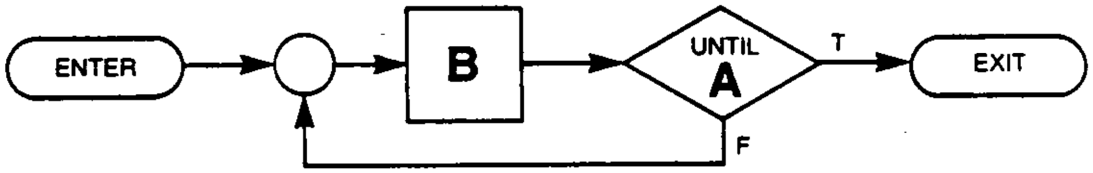
FIGURE 1. Control structures.

FIGURE 1C. DO WHILE: DO WHILE condition A, process B.



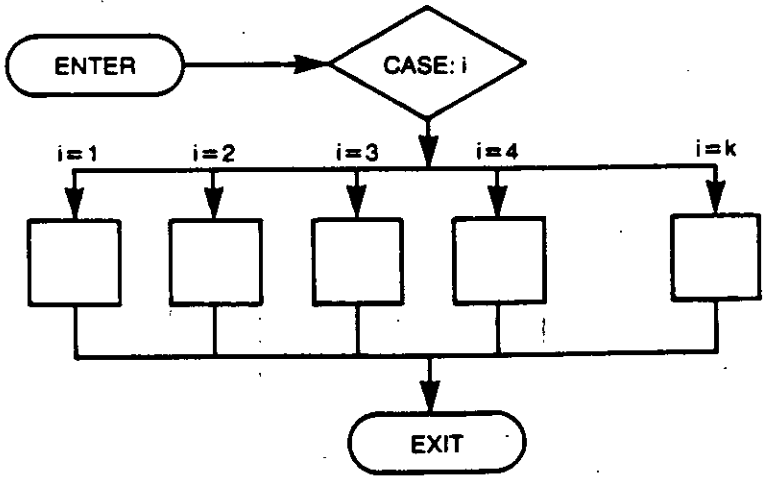
The DO WHILE structure is a loop in which the condition A is evaluated. If found to be true, then control is passed to process B and the condition A is evaluated again. If condition A is false, then control is passed out of the loop.

FIGURE 1D. DO UNTIL: DO UNTIL condition A, process B.



The DO UNTIL structure is similar to the DO WHILE -- except that the test of condition A is performed after process B has executed. Thus the DO UNTIL loop will be performed once regardless of the value of condition A.

FIGURE 1E. CASE: Based on Case conditional i, process i.



Control is passed to process k based on the value of i.

FIGURE 1. Control structures. (continued)

FIGURE 1F. Nesting of Control Structures.

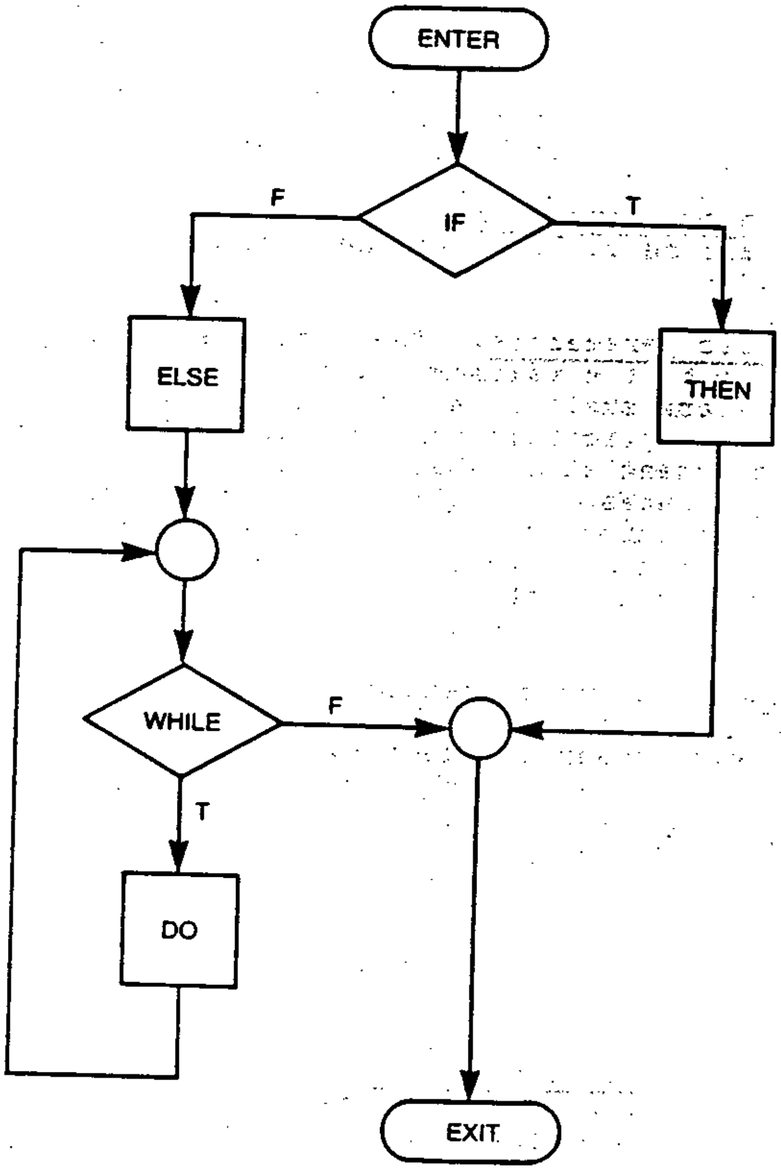


FIGURE 1. Control structures. (continued)

5.3.7 Branching. Branching statements (GOTO's) may be used only with the approval of the contracting activity. Branching statements, if approved, shall only pass control to a statement that is in the same unit. Each GOTO must pass control only forward of its point of occurrence.

5.3.8 Relocatability. Programs shall be built in the form of generic relocatable object modules, except for those segments that need to be assigned specific absolute locations in memory.

5.4 Programming conventions. The following programming conventions shall be utilized in the development of all software. (See 6.1)

5.4.1 Symbolic parameters. The software shall use symbolic parameterization to the maximum extent practical. Duplication of symbolic parameters shall be minimized through use of a common source of values. When duplication is necessary, common symbolic parameter identification nomenclature shall be used and comments will point to location of duplicates. Symbolic parameters shall be grouped at the beginning of each subprogram's data declaration area. Comments shall provide a definition and the location of all parameters. Special symbolic parametric definition features of the high level language shall be used.

5.4.2 Naming. Naming conventions shall be uniform throughout the software. Program, subprogram, module, unit, and data names shall be uniquely chosen to identify the applicable functions performed and their positions in the hierarchical logic structure relative to the other components of the software being developed. Names may be duplicatively chosen when the HOL compiler being used automatically encapsulates the names within a particular segment of code.

5.4.3 Numerical conventions. Numerical conventions shall be established by the contractor so that they are uniform throughout the software.

5.4.4 Symbolic constants and variables. Constants and variables entering into numerical computations shall follow the constraints set forth in this Standard. (see paragraph 5.4.1)

5.4.5 Mixed mode expression. The use of mixed mode numerical operations shall not be used unless a specific waiver has been previously granted by the contracting activity. When approved, the use of mixed mode numerical operations shall be completely commented in the program listing and shall be completely described in the related software descriptive documentation.

5.4.6 Grouping. Parentheses or other subexpression delimiters shall be used where necessary to clarify the order of evaluation of compound expressions.

5.4.7 Significant digits. Sufficient significant digits shall be used in calculations to yield a minimum computational error, and rounding by the programmer shall not occur until the final computational step. The degree of computational error shall be analyzed to determine if systems accuracy requirements are fulfilled.

5.4.8 Abstracts. Each hierarchical component (i.e., program, subprogram, module, and unit) shall include at the beginning of the executable code a textual description of its inputs, outputs, and function or task; and a list of other components called. In addition to general explanations, to assist understanding, precise references to the appropriate statement labels and data-names shall be included in each module and unit descriptive abstract. The descriptive abstract shall define the allowed and expected range of values for all inputs and shall define the allowed and expected range of values for all outputs. A history of the original and updating programmer names, dates and reasons for all changes, the activity or commercial company name, and the activity or company division code or billet identifier shall be included. Additionally, the abstract shall include a description of any transportability constraints.

5.4.9 Comment statement. In order to facilitate software comprehension, comment statements shall be used throughout the software. Each source statement shall be self-defined or defined by a comment phrase to a level understandable by a person knowledgeable in software but not associated with the original development effort. Logical groups of comment phrases may be included in a single comment statement.

5.4.10 Indentation. Software structural indentation shall be used in the source code listing to improve readability and clarity.

5.4.11 Software listings. For acceptance as a deliverable, the listing of compiled software shall include source language statements and comments with resulting object machine instructions interspersed appropriately (together with actual or equivalent assembler statements, if available). Relative location of instructions and operands shall be exhibited together with statement labels and identification numbers. All descriptions of referenced units and data shall be included in conjunction with this listing and arranged for convenient access.

5.4.12 Cross-reference listing. A cross-reference listing shall be produced relating each data name to the location of every statement referring to it, and relating each unit to the location of every other unit calling upon it. The list shall be exhibited as a sequential table in alphanumeric order.

5.4.13 Load maps. The contractor shall describe the format, method and location where the various components and portions thereof are loaded in the system's computers and, if applicable, stored on disks or other storage devices. This mapping shall include delineating all of the portions of the software that are to be resident in the device in question and the location and size of each portion of the software. If the system has more than one defined configuration or mode of operation for the software, the contractor shall describe this information for each configuration or mode.

5.4.14 Execution efficiency. Source code statements shall be optimized for execution efficiency. For maximum memory efficiency, common units of code should be used instead of included/copied source code blocks whenever practicable. Neither execution time nor memory efficiency shall be optimized at the expense of readability, clarity, or maintainability.

5.4.15 Source code segment includes/copy. When it is not practical to utilize common software units for repetitive segments of source code, the units shall be coded only once as a structural source code block. Thereafter being referenced/utilized upon each occurrence by appropriate "includes" or "copy" features, or equivalent constructs of the source HOL compiler.

5.4.16 Source statement. A source language statement shall not be compound or complex in structure.

5.4.17 Flow charts. This Standard does not require that flow charts be a contract deliverable item.

5.4.18 Design representations. The contractor shall utilize some form of graphic representation as an aid in developing the structured design and code of the software. The type of graphic representation to be used is subject to approval, in advance of its use, by the contracting activity.

5.5 Software implementation. The contractor shall generate the software in a top-down, well-controlled manner. The highest level of control logic shall reside at the top of the hierarchy; the computational or algorithmic functions shall reside at the lower levels. Top-down implementation requires that units of code which call other units of code be designed, coded, and unit tested before the units they call. Programming shall commence with the highest levels which shall be placed under the contractor's internal software configuration management and library control and then be tested extensively before descending downward in the design to the programming of subordinate levels. This methodology will permit orderly integration of units of code into an evolving computer program. Incremental development of software will result which allows for well-controlled incremental testing where the higher control structures are tested most often.

Programming conventions, software design rules and programming standards shall be promulgated to and followed by all levels of software implementation personnel. The contractor shall ensure programmers are skilled in the use of the specified language, computer capabilities, and associated language support software programs. Standard procedures shall be developed for programmers to follow in developing the software. A code walk-through review and unit testing (debug) of each software component shall be completed prior to submission of the component for software configuration management and library control, and subsequent integration and testing. (See 6.1)

5.5.1 Software development organization. The contractor shall implement a software implementation organization that facilitates the top-down design, coding, integration, and testing of the software.

5.5.2 Resource management. The contractor shall be responsible for management of computer system resources (e.g., main memory, mass storage, processor time, input/output controller(s), and input/output channel(s)). He shall determine the original assignment of system resources through analysis and modeling. The contractor shall monitor the utilization of the assigned resources as software development progresses. A minimum reserve of 20 percent capacity shall exist in each resource area at the time of software acceptance by the contracting activity. (Refer to the Appendix of this Standard)

5.5.3 Unit test. The contractor shall conduct unit test (debug) of all software developed. The contractor shall establish procedures to ensure that every deliverable executable source code statement is executed at least one time during the unit test process.

5.5.4 Library usage and control. The contractor shall establish procedures for producing, updating and controlling source and object libraries of the software under development. All initial software and development changes shall be maintained in both source and object representations. All patches shall be maintained in maintenance and patch logs and in an electronic storage library until incorporated in the patch-free source program. Patches shall, as a minimum, be identified by: patch production date, programmer producing the patch, the software component that the patch is applicable to, the corresponding STR/SCP number or identification, the test that revealed the problem, the testing that certifies the integrity of the patch, and the problem that necessitated the patch.

5.6 Software generation. All software delivered by the contractor shall be capable of being generated from contracting activity owned support software and the contractually delivered support software. The contractor shall completely describe all procedures necessary for the generation of the software and shall provide, when used, the control or command files used to generate the software.

5.7 Software operation. Procedures for loading, initialing, and operating the software to be delivered shall be described in clear, operational terms and shall be subject to the review and approval by the contracting activity. In determining software operation procedures, the contractor shall investigate and define in detail the following areas.

5.7.1 Non-functional operation. Minimal processor and peripheral equipment requirements, equipment set-up for system operation, software set-up, special parameter entering requirements, standby/operate procedures, monitoring procedures, and recovery procedures shall be defined. (See 6.1)

5.7.2 Functional operation. Individual operator and station functions; coordinated station procedures; all human factor aspects, modes and procedures necessary for each console or station operator to perform his function in support of system operation; the function of every control button, switch, readout and display affected by or affecting the system; and all constraints imposed on operator actions shall be defined. (See 6.1)

5.8 Software testing. The contractor shall determine the scope of tests required to ensure that the software being developed meets all specified technical, operational and performance requirements, and acceptance criteria. The contractor's test planning, and subsequent test implementation, shall be consistent with the top-down development of software by providing for the top-down integration and developmental testing of the software. The contractor shall be responsible for accomplishing all development testing. Formal test planning shall include development of:

a. Test requirements.

b. Software acceptance criteria.

c. Levels of testing to verify specified performance.

d. Internal procedures for scheduling and conducting tests.

e. Detailed procedures for testing at each level.

f. Reporting procedures for test results.

All required test plans, specifications, and procedures shall be subject to review and approval by the contracting activity. The contracting activity shall be kept advised of all test schedules. Contracting activity or contractor representatives, as designated, shall be permitted to witness all tests. The contractor shall provide all supporting software necessary to conduct, control, and record tests. The contractor shall define any special support software necessary to satisfactorily test the software being developed. The contractor shall identify to the contracting activity any equipment or information to be supplied by the contracting activity that is required to support the testing program. These test requirements shall be reported early enough to allow the contracting activity to obtain and deliver them without impacting the schedule of development and testing. The contractor shall provide or ensure the availability of adequate facilities for conducting all required tests. The contracting activity shall have the option of contractually specifying the facility that should be used to conduct any portion of the software test. The contractor shall prepare test reports showing quantitative results of all tests. Such reports shall be signed by a representative of the contractor. Any formal or informal approval of the testing results by a representative of the contracting activity during the course of software implementation shall not be construed as a guarantee of the acceptance of the software to be delivered. (See 6.1)

5.8.1 <u>Software integration and developmental testing prerequisites</u>. Prior to subjecting the software to integration and developmental testing, the contractor shall ensure that, as a minimum, each unit shall have completed the following:

a. Ensure error-free compilation/assembly of the coded unit.

b. Satisfactory completion of a code walk-through.

c. Satisfactory completion of unit test.

d. Verification that the coded unit fully satisfies the design requirements, including all input and output requirements.

e. Satisfactory completion of all software quality assurance requirements.

f. Placed under the contractor's internal formal software configuration management and library control.

5.8.2 Software integration and developmental testing. Units shall be integrated individually into particular modules and then tested to determine compliance with the applicable technical, operational, performance, test, and maintenance requirements. As a minimum, developmental testing shall be performed to:

a. Ensure error-free linkage of the modules.

b. Test the software in terms of input/output performance for satisfaction of the applicable detailed performance and design requirements.

c. Ensure the capability of the software to handle properly and survive erroneous inputs.

d. Ensure the total man-machine interface.

e. Ensure proper operation including initiation, data entries via peripheral devices, software loading, restarting, and the monitoring and controlling of system operation from display consoles and other control stations as applicable.

f. Ensure the proper interfacing of all equipment specified in the software requirements specifications.

g. Ensure the capability of the software to satisfy all software performance requirements.

h. Ensure the capability of the software to be relocatable, except for those segments that need to be assigned absolute locations in memory.

5.8.3 System(s) integration test. In instances where the developed software is an element of a larger system involving the integration of two or more programs, the contractor(s) shall be required to participate in total system integration testing. System integration testing may be conducted at facilities other than the development facility, such as a land-based test site. Each contractor shall provide technical support to the integration testing as required.

5.8.4 Software trouble reporting. The contractor shall develop and implement internal procedures for handling and reporting all software or software related problems identified. In addition to the categories and priorities described below, a code shall be utilized to indicate the status of each Software Trouble Report (STR) as it progresses through the correction cycle. All STRs shall be verified for accuracy and correctness and documented on standard forms. (See 6.1)

The contractor shall maintain a complete set of software trouble report data files throughout the duration of the contract and make this information available to the contracting activity or its authorized representative upon request.

5.8.4.1 Software trouble report category. Software problems shall be classified by category as follows:

a. Software trouble (S). The software does not operate according to supporting documentation and the documentation is correct.

b. Documentation trouble (D). The software does not operate according to supporting documentation but the software operation is correct.

c. Design trouble (E). The software operates according to supporting documentation but a design deficiency exists. The design deficiency may not always result in a directly observable operational symptom but possesses the potential of creating trouble.

5.8.4.2 Software trouble report priority. Software errors are prioritized by severity as follows:

a. Priority 1 - An error which prevents the accomplishment of an operational or mission essential function in accordance with official requirements (e.g., causes a program stop), which prevents the operator's accomplishment of an operational or mission essential function, or which jeopardizes personnel safety.

b. Priority 2 - An error which adversely affects the accomplishment of an operational or mission essential function in accordance with official requirements so as to degrade performance and for which no alternative work-around solution exists; or which adversely affects the operator's accomplishment of an operational or mission essential function so as to degrade performance and for which no alternative work-around solution exists. (Reloading or restarting the software is not an acceptable work-around solution.)

c. Priority 3 - An error which adversely affects the accomplishment of an operational or mission essential function in accordance with official requirements so as to degrade performance and for which there is a reasonable alternative work-around solution; or which adversely affects the operator's accomplishment of an operational or mission essential function so as to degrade performance and for which there is a reasonable alternative work-around solution. (Reloading or restarting the software is not an acceptable work-around solution.)

d. Priority 4 - An error which is an operator inconvenience or annoyance and does not effect a required operational or mission essential function.

e. Priority 5 - All other errors.

5.8.4.3 Software trouble report disposition. The contractor shall determine the initial status of each STR when it is reported and shall monitor and record any and all changes of the status of each STR. When all appropriate action concerning an STR has been completed, the contractor shall determine and record the final disposition of the STR.

5.9 Software quality assurance. The contractor shall implement software quality assurance policies, practices and procedures to verify that in each stage of the development that the product software will meet the specified software requirements as they have been approved by the contracting activity. The contractor shall implement software quality assurance procedures to ensure that the accuracy, correctness and performance of the product software are validated, to verify the accuracy and conformance of software documentation to the requirements of this Standard and to ensure that all procedures incumbent on the contractor are properly and completely followed. The procedures shall be open to review by the contracting activity or its authorized representative. The implementation and functioning of the procedures shall also be open to inspection by the contracting activity or its authorized representative. The contractor's organization responsible for software quality assurance shall include provisions for addressing all of the subject areas discussed below. (See 6.1)

5.9.1 Reporting level. The managers of the organization responsible for software quality assurance shall have corporate reporting responsibility that is independent of the manager of the organization responsible for developing the software under contract so as to assure an objective evaluation of conformity and progress.

5.9.2 Participation in audits. The organization responsible for software quality assurance shall implement procedures for independent software quality audits which measure system conformance with technical and management requirements and standards. These audits shall take place throughout the development phase, starting with requirements definition and ending with test, delivery and acceptance.

5.9.3 Requirements reviews. The organization responsible for software quality assurance shall participate in requirements reviews and requirements walk-throughs of the detailed software performance requirements utilizing procedures to assure completeness, consistency, and accuracy of presented materials and to assure timely and correct completion of action assignments.

5.9.4 Design reviews. The organization responsible for software quality assurance shall participate in design reviews and design walk-throughs utilizing procedures to assure completeness, consistency, and accuracy of presented materials and to assure timely and correct completion of action assignments.

5.9.5 Software design. The organization responsible for software quality assurance shall ensure that the detailed design of the software is examined for complete compliance with all software requirements specified by the contracting activity.

5.9.6 Program code. The organization responsible for software quality assurance shall ensure that all code to be delivered is examined for complete compliance with the detailed software design and to the specified programming conventions and standards.

5.9.7 Tests. The organization responsible for software quality assurance shall witness tests to assure conformance with approved test procedures. Software quality assurance activities shall ensure the adequacy and accuracy of record-keeping, maintenance and control of test materials, and conflict/discrepancy resolution.

5.9.8 Deliverable items. The organization responsible for software quality assurance shall provide quality assurance procedures and shall monitor conformance to all procedures to assure contractual correctness of all deliverable items.

5.9.9 Reporting. The organization responsible for software quality assurance shall utilize both interdepartmental and intradepartmental reporting chains to assure prompt reporting of the results of software quality related activities. Software quality assurance shall follow up any noted discrepancy/action assignment to assure timely and complete correction of the problem.

5.9.10 Software trouble reporting. The organization responsible for software quality assurance shall ensure that internal procedures for reporting, correcting, and analyzing software related problems/failures are adequate and effective.

5.10 Software acceptance. In addition to any criteria specified by the contracting activity, software acceptance shall be predicated upon satisfaction of system reserve requirements, successful completion of the stress test, the priority and number of unresolved software and documentation errors, and the number of existing patch words.

5.10.1 Reserve requirements for software acceptance. Total system utilization of memory, input and output channels, and processing time shall be examined to ensure that a reserve of at least 20 percent shall exist at the time of software acceptance by the contracting activity. (Refer to the Appendix of this Standard)

5.10.2 Stress test requirements for software acceptance. Prior to software acceptance, the software shall have successfully completed the stress test. This test is intended to exercise all of the functions of the software for a previously approved period of time in order to demonstrate that the software is reasonably free of serious or numerous errors. Under this test, the software is to be stressed to the limits of its designed capacities and beyond in order to ensure that degradation at the point of saturation is not catastrophic.

5.10.2.1 Stress test environment. The stress test shall be conducted in an environment approved by the contracting activity. Normally, the stress test shall be conducted in the ultimate user environment for which the system and software were designed. If the ultimate user environment cannot be used for the stress portion of the stress test, the alternate test site for the stress portion shall be a fully integrated facility equipped with the same hardware found in the ultimate user environment. The remainder of the stress test requirements must still be met in the ultimate user environment including the test length as specified below.

5.10.2.2 Software to be tested. The software to be used in this test shall be the currently approved baseline version which is under the contractor's internal formal software configuration control. It shall have been generated with support software which has been previously approved by the contracting activity.

5.10.2.3 Stress test documentation. All software documentation deliverables for the software being tested shall be available in their final form to the testing activity at the time the stress test is conducted.

5.10.2.4 Stress test software operation. At the start of this test the software under test shall be loaded, initialized and started and shall not be stopped until scheduled test completion. Any stop of the program under test prior to scheduled completion shall constitute failure to meet the stress test requirements. For software with auto-recovery capabilities, any interruption in program execution which invokes the auto-recovery feature shall be treated as a program stop. When testing an integrated system, subsystems shall operate simultaneously throughout this test as they are intended to during normal usage operations.

5.10.2.5 Stress test duration. The length of time for this test will vary depending on the complexity of the software and the mission of the system under test. For software which is of such complexity that all of the testing requirements and the intent of this Standard cannot be satisfied within the minimum periods prescribed below, the length of time for this test shall be extended as determined by the contracting activity or the requirement, if any, specified in the software requirements specification or a higher level document. Initial system setup time to establish normal operating conditions shall not be included as part of the testing period.

a. For systems that are designed to operate continuously or for more than one day at a time when the system is placed into operation, the minimum length of time for this test shall be 25 continuous hours.

b. For systems that are not designed to operate continuously or for more than one day at a time, the minimum length of time for this test shall be the length of time required to fulfill the system's mission(s) including any pre-mission or post-mission periods, or the length of time it takes to complete the test requirements of this Standard, whichever is longer. The testing period shall be continuous.

5.10.2.6 Stress test input data. The software under test shall be subjected to normal and abnormal input data in such a way as to exercise all functions. The test shall be designed to exercise in random order, as specified in approved test procedures, variations of all modes of operation following scenarios of typically normal, as well as abnormal, system operation which demonstrate compliance with operational requirements. Legal and illegal inputs shall be made to test software integrity.

5.10.2.7 Stress test performance. For certain periods during the test, as prescribed by the contracting activity, the software shall be required to operate at saturation levels which stress the software's capabilities in terms of response times and data handling capacity. Attempts shall be made to exceed every data rate and data volume capacity. Such stress shall be generated through inputs from manual or automated interfaces. There shall be at least three distinct stress periods and the total time spent on stressing the system shall represent at least one-third of the total length of the test. Methods of stressing the system shall include, but are not limited to:

a. Provision for more information to be processed than the processor is designed to accommodate.

b. Saturation of the data transfer capabilities by requiring more data to be transferred in and out of memory, peripherals, subsystems, and inter-facing systems than the system was designed to accommodate.

c. Exceeding assigned storage area capacities, e.g., buffers, tables and scratch areas.

5.10.2.8 Reduced capability software stress testing. For systems designed to operate in a reduced capability mode(s) due to hardware failure(s), each possible reduced capability mode shall be validated by causing actual physical degradation of the hardware (where this can be safely accomplished), e.g., turn off power to a piece of equipment. Correct processing of the failure shall be validated including the capability to return to a normal mode of operation. Scenarios while operating in the reduced capability mode(s) shall demonstrate compliance with formal specifications. The duration of system operation in the reduced capability mode shall be determined by the contracting activity.

5.10.2.9 Test and maintenance support software stress testing. On-line hardware maintenance support software shall be demonstrated during test periods of nominal loading when such a capability exists. For test and maintenance support software designed to operate only when the operational system is off-line, the demonstration of the test and maintenance support software will not be included as part of the test of the operational system software, but as a separate demonstration to the satisfaction of the procuring agency.

5.10.2.10 Errors during test. Occurrence of a software error of a priority one or two severity during the software stress test shall require correcting the error and repeating the test in its entirety. Occurrence of an intermittent software error of a priority one or two severity shall require that the particular operation(s) be repeated several times to the satisfaction of the contracting activity, and that the full stress test then be repeated in its entirety. There shall be no patching to correct errors while the stress test is in progress. The only other patches that may be in the software during the software stress test are those which are considered to be required for testing purposes and they shall be specifically set forth in the test procedures. Caution should be exercised that these patches do not interfere with the validity of the software stress test results.

5.10.2.11 Stress test limitations. Specified in Sections 5.10.3 and 5.10.4 are error and patch limits relevant to the software under-going the stress test. These limits shall be satisfied prior to the commencement of the test. In addition, errors detected during the test which cause this criteria to be exceeded shall invalidate the test; but an individual test may continue to run to its planned completion in order to uncover any additional errors. In those tests where the error limits are exceeded, the test shall be rerun in its entirety. In those tests where the patch limits are exceeded, consequently requiring source level changes and recompilation or assemblage, the test shall be rerun in its entirety.

5.10.3 Error limits for software acceptance. The following error limits shall apply at the time of software acceptance by the contracting activity:

a. The number of unresolved software errors (excluding documentation errors) shall not exceed the following:

Severity	Limits
Priority 1 and 2 (high)	Zero
Priority 3 (medium)	One per 70K of machine instruction words or fraction thereof.
Priority 4 and 5 (low)	One per 35K of machine instruction words or fraction thereof.

b. Intermittent software errors shall be included in the count of software errors and receive no special consideration.

c. The number of unresolved technical errors in all of the deliverable documentation shall not exceed the sum of three, plus one for every 25K of machine instructions or fraction thereof. For example, a program having 300K machine instructions: 3 + 12 = 15 allowable documentation errors.

5.10.4 Patch limits for software acceptance. The following patch limits shall apply at the time of software acceptance by the contracting activity:

a. The total number of patch words in the software shall not exceed 0.005 times the total machine instruction words in the software.

b. Patches which correct software errors are permitted only if they are automatically applied to the software during the load process.

c. All patches shall be documented.

5.11 <u>Software configuration management</u>. The contractor shall develop and implement internal policies, practices, and procedures to ensure the positive identification, control and status accounting of the software and all items of deliverable software documentation during all phases of the development effort. The contractor shall ensure that such procedures are integrated with the configuration management procedures addressing the total system. (See 6.1) Procedures shall provide:

a. Positive identification of all software components and associated documentation.

b. Timely, comprehensive and accurate processing, reporting, and recording of proposed changes to components under software configuration control.

c. Comprehensive implementation of approved changes and dissemination of corrected documentation and software changes.

d. Accurate recording and reporting on the status of all proposed changes and change resolution.

e. Verification and implementation of identification, change control, and status accounting of the descriptive documentation and software materials.

5.11.1 <u>Software configuration identification</u>.

5.11.1.1 <u>Developmental baseline</u>. Internal baselines shall be established by the contractor to ensure orderly transition from one software development phase to the next. Internal baselines shall be established whenever it is necessary to define internal departure points for future changes in performance, design, and related technical requirements.

5.11.1.2 <u>Documentation identification</u>. The contractor shall establish and implement titling, labeling, numbering, and cataloging procedures for all computer software documentation and software materials which satisfy the following criteria:

a. Denotes the component to which it applies.

b. Describes the purpose of the document.

c. Defines the baseline which it is a part of, or in support of.

d. Denotes the serial, edition and change status of the document.

e. Indicates the compilation date for each delivered component.

f. Sequence numbering of all source records in a module shall be structured so that future changes to any component can be properly noted.

g. Provides visual and machine readable identification for all delivered software media (disks, tapes, listings, etc.) which permits direct correlation with delivered documentation and specifies and identifies their content.

5.11.2 Software configuration control. The contractor shall establish and implement procedures for the internal formal control of all documents, software materials and the development support library. These procedures shall include bringing each component of the software under configuration control. These procedures shall also include the establishment and functioning of a software configuration control board and the methods and formats for submission and acting on software change proposals and software trouble reports. (See 6.1)

5.11.2.1 Software changes. During software development, the contractor shall establish and maintain internal procedures to process proposed changes, Software Change Proposals (SCP), to the software (including descriptive documentation) which has been placed under internal software configuration control. SCPs to software (including descriptive documentation) which is under software configuration control by the contracting activity shall be submitted to the contracting activity. SCPs which have contractual cost, interface, or schedule impact shall be attached to a DD Form 1692 (Engineering Change Proposal, page 1) completed and numbered in accordance with appropriate instructions and submitted to the contracting activity. (See 6.1)

5.11.2.2 Documentation changes. Procedures for controlling the preparation and dissemination of changes to documentation to reflect approved and implemented Engineering Change Proposals (ECP), Class I, and SCPs shall be developed. Such procedures shall be designed to insure the simultaneous promulgation of the descriptive documentation and software materials.

5.11.2.3 Software configuration control boards (SCCB). All contractor defined baselines, plus approved changes from those baselines shall be under the formal control of a internally established SCCB. The SCCB shall consider all proposed changes to the baseline and take appropriate action on each proposal.

5.11.3 Software configuration status accounting. The contractor shall establish procedures to enable the generation of periodic status reports on all software components under configuration management. These procedures shall identify all SCPs and STRs in review, and in the current stage of implementation. These procedures shall confirm the incorporation of approved software configuration changes. These procedures shall also identify all disapproved and deferred SCPs and STRs. The contractor shall implement procedures which reconcile the software configuration status accounting reports and the status of the software with the approved baseline(s) and its approved changes.

5.12 Software management planning. The contractor shall determine and implement a management system for the development effort which is acceptable to the contracting activity. The management of the development shall emphasize efficiency and economy. Clear lines of authority and responsibility shall be established. The management system shall provide for the coordination of all facets of the development under a master schedule of events and milestones. The detailed performance requirements, the software architecture and the detailed software design will be subject to review by the contracting activity at scheduled milestones in the software development cycle. Milestone dates shall be established for demonstrations of evolving software capabilities. Such demonstrations are intended to provide the necessary visibility for project management and meaningful output for product validation. The management system shall provide a capability to monitor the progress of the development by means of regular status reports, reviews and audits. The management system, including planning and procedural guidance for the development effort, shall be compiled in an overall plan for visibility, formality, control and coordination of the development. (See 6.1)

5.12.1 Software management organization. The contractor may use an internal organization of his own choice, subject only to the requirements from this Standard which are invoked by the contracting activity. The contractor shall designate an overall manager for the development effort. The functions of design, implementation, test, software quality assurance, and software configuration management shall be given organizational visibility. The relationship of all support functions, both full-time and part-time, required to support the development effort shall be clearly defined. The responsibilities of all subcontractors, if used, shall be clearly visible to the contracting activity.

5.12.2 Computer resource requirements. The contractor shall determine his resource requirements in the three areas of personnel, facilities, and equipment. Planning shall be completed early enough to permit orderly acquisition, installation, and training (if applicable) of resources on an optimum schedule to prevent delay and to avoid dead-time. Reusability, permanency, or length of project and convenience of location shall be weighed. The contracting activity may direct the use of specific facilities. Planning shall be responsive to schedule changes. The contractor shall avoid sharp fluctuations in personnel requirements by judicious shifting of personnel as development tasks change.

The contractor shall consider the cost-effectiveness of commercial equipment to assist in the development where appropriate. The possibility of continuing use of the equipment by the contracting activity during the operational support phase of the software life cycle shall be a consideration. The contractor shall implement a system of management monitoring of utilization in the areas of personnel, facilities, and equipment considering both quantity and cost. Actual utilization rates shall be compared to predicted rates at least monthly. The contracting activity may specify more frequent comparison. Variations shall be expeditiously investigated and corrective action initiated. Personnel stability and productivity shall be measured regularly.

5.12.3 Software status reviews. Status reviews may be requested by the contracting activity at regular intervals during the development effort. The contractor shall be able to provide at these reviews the current status, progress, and problems occurring in the development effort within the purview of the contractor.

5.12.3.1 Software status review subjects. The contractor shall address the following subjects, as appropriate to the stage of the development effort, in each status review:

a. Organization changes, managerial personnel changes

b. Design status

c. Development schedule status (milestone prognosis)

d. Coding status

e. Software Trouble Report (STR) status

f. Software Change Proposal (SCP) status

g. Integration schedule status

h. Testing status

i. Deliverables

j. Progress on previous problems

k. New action items/problems

l. Delinquencies: governmental, outside contractor, subcontractor, and internal

m. Manpower utilization

n. Facilities utilization

o. Computer system resource utilization (see 5.5.2)

p. Financial summary.

5.12.3.2 Software status review subject items. Within each subject area, the contractor shall cover the following items, as applicable:

a. The schedule updated to the end of this reporting period.

b. Major difficulties encountered and plans to overcome them, including: Tasks/units that are currently behind schedule (or have anticipated schedule changes), their effects on completion of the project, and steps being taken to remedy schedule delays.

c. Other information which defines cause and effect of significant changes on the contract schedule.

d. Problems which actually or potentially will cause deviation from contract requirements.

e. Summary of meetings and conferences held during the reporting period, including action items with due dates for both the contractor and the contracting activity. Current status of action items shall be included until reported closed.

5.12.4 Software documentation reviews. Documentation and software materials, as specified, shall be scheduled for detailed review prior to approval or acceptance. The purpose of the review shall be to:

a. Verify that the subject documentation and software materials comply completely and accurately with the software performance and design requirements of higher level documentation and software materials and all other standards and constraints imposed by the contracting activity.

b. Verify the accuracy and completeness of the documentation and software materials by checking for all components, their correct cross-reference, and editorial accuracy.

c. Verify that software documentation is being developed concurrently with the development of the item which it documents.

The review shall be in two stages: a preliminary working-level review followed by a formal (or critical) review after changes resulting from the preliminary review have been entered. Reviews shall be scheduled by the contractor, with the concurrence of the contracting activity, and in accordance with milestones in the software development management plan. The contracting activity may designate other activities to participate in the review. The contractor shall distribute drafts of review documentation and software materials to each designated activity in advance of the scheduled preliminary review. The critical review for the acceptance or approval of the documentation and software materials shall expeditiously follow the distribution of the corrected version.

5.12.5 Special software reviews. Special reviews may be scheduled by the contracting activity at major milestones or events in the development effort not covered by documentation reviews or status reviews. A special review of the test program as developed shall be conducted. The contractor shall furnish the same support for special reviews as the status reviews.

5.12.6 Inspections and audits. The contracting activity may employ a physical inspection to determine the contractor's conformance with contractual requirements. As a minimum, areas of interest shall include documentation controls, deliverable data items, government-imposed standards, and the following:

a. Facilities. The development and test facilities may be inspected for contractual conformity at any time during the life of a software development contract.

b. Software configuration management. Contractor conformance with the approved software configuration management requirements may be audited through examination of records and attendance at software configuration control board meetings.

c. Internal standards. The contracting activity may audit the contractor's conformance with internal standards of software development and control.

d. Software quality assurance. The contracting activity may audit and inspect the contractor's conformance with the approved software quality assurance requirements.

6. MISCELLANEOUS

6.1 Contract data requirements. The following list of Data Item Descriptions shall be utilized if the contracting activity desires to order data that is generated from having invoked pertinent work tasks that are established within this Standard. Such data must be specified on the Contract Data Requirements List, DD Form 1423.

Paragraph no.	Data requirements title	Applicable DID no.
5.1.2.4b, 5.1.2.5c/5.2.3	Interface Design Specification (IDS)	DI-E-2135A
5.1	Program Performance Specification (PPS)	DI-E-2136B
5.2	Program Design Specification (PDS)	DI-E-2138A
5.3/5.4/5.5	Program Description Document (PDD)	DI-S-2139A
5.2.2.7	Data Base Design Document (DBD)	DI-S-2140A
5.3/5.4/5.5	Program Package Document	DI-S-2141A
5.8	Computer Program Test Plan	DI-T-2142A
5.8	Computer Program Test Specification	DI-E-2143A
5.8	Computer Program Test Procedures	DI-T-2144A
5.8	Computer Program Test Report	DI-T-2156A
5.7	Operator's Manual (OM)	DI-M-2145A
5.7	System Operator's Manual (SOM)	DI-M-2148A
5.9	Software Quality Assurance Plan	DI-R-2174A

5.11	Software Configuration Management Plan (SCMP)	DI-E-2035B
5.12	Software Development Plan	DI-A-2176A
5.11.2.1	Software Change Proposal (SCP)	DI-E-2177A
5.8.4/5.11.2	Computer Software Trouble Report (STR)	DI-E-2178A

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

Preparing activity:
Navy - NM
(Project ECRS - N003)

APPENDIX

RESOURCE CAPACITY MEASURING

10. GENERAL

10.1 Scope. This appendix provides guidance to be used in measuring the reserve capacity requirements contained in sections 5.2.2.4, 5.5.2, and 5.10.1 of DOD-STD-1679A.

10.2 Application. The degree to which this Appendix applies to any specific contract is the option of the contracting activity.

20. REFERENCED DOCUMENTS

Not applicable

30. DEFINITIONS

30.1 Main memory. That component of a computer system from which stored programs are executed and within which data, manipulated by such programs or involved in input/output operations, is stored.

30.2 Secondary storage. That component of a computer system that is used as an auxiliary to main memory. Typical types of secondary storage are bulk store, magnetic tapes, and disks.

40. GENERAL REQUIREMENTS

40.1 The applicable portions of this Appendix may be used for measuring the computer system reserve capacities of Main Memory, Secondary Storage, Throughput, Number of Input/Output Channels, and Input/Output Channel Throughput. The intent of requiring a reserve in the capacities of the computer system is to provide physical resources for the correction of discrepancies or for growth in the future. In situations wherein these instructions cannot be followed, the intent shall be satisfied.

50. DETAIL REQUIREMENTS

50.1 Main memory. The reserve capacity shall be measured at peak main memory loading of the computer system during performance of its operational mission. Peak main memory loading shall include all software required for the successful execution of the systems operational mission. The reserve capacity need not be physically installed in the computer system, but no cabinet, chassis, or backplane modifications shall be required for its installation.

50.2 Secondary storage. The reserve capacity shall be measured at peak secondary storage loading of the computer system, with all secondary storage information included, during the performance of the systems operational mission. When secondary storage consists of multiple units and/or types of units, the reserve capacity shall apply to the sum total of all secondary storage reserve capacity. The reserve capacity need not be physically installed in the computer system, but no cabinet, chassis, or backplane modifications shall be required for its installation.

50.3 Throughput. The reserve capacity of central processor throughput shall be expressed as a percentage of available capacity at full operational loading over a specific period of time, as determined by the characteristics of the operational mission. In the case of cyclic applications, this period would be the time between availability of successive input data buffers.

50.4 Number of input/output channels. The reserve capacity shall be a percentage of those channels available. The reserve channels need not be physically installed in the computer system, but no cabinet, chassis, or backplane modifications shall be required for their installation.

50.5 Input/output channel throughput. The reserve capacity shall be as expressed in section 50.3 above.

INSTRUCTIONS: In a continuing effort to make our standardization documents better, the DoD provides this form for use in submitting comments and suggestions for improvements. All users of military standardization documents are invited to provide suggestions. This form may be detached, folded along the lines indicated, taped along the loose edge *(DO NOT STAPLE)*, and mailed. In block 5, be as specific as possible about particular problem areas such as wording which required interpretation, was too rigid, restrictive, loose, ambiguous, or was incompatible, and give proposed wording changes which would alleviate the problems. Enter in block 6 any remarks not related to a specific paragraph of the document. If block 7 is filled out, an acknowledgement will be mailed to you within 30 days to let you know that your comments were received and are being considered.

NOTE: This form may not be used to request copies of documents, nor to request waivers, deviations, or clarification of specification requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document*(s)* or to amend contractual requirements.

(Fold along this line)

(Fold along this line)

DEPARTMENT OF THE NAVY

Chief of Naval Material
ATTN: NAVMAT 08Y
Washington, DC 20360

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 12503 WASHINGTON D. C.

POSTAGE WILL BE PAID BY THE DEPARTMENT OF THE NAVY

Chief of Naval Material
ATTN: NAVMAT 08Y
Washington, DC 20360

(TO DETACH THIS FORM, CUT ALONG THIS LINE.)

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

(See Instructions – Reverse Side)

1. DOCUMENT NUMBER DOD-STD-1679A(NAVY)	2. DOCUMENT TITLE SOFTWARE DEVELOPMENT
3a. NAME OF SUBMITTING ORGANIZATION	4. TYPE OF ORGANIZATION *(Mark one)* ☐ VENDOR ☐ USER ☐ MANUFACTURER ☐ OTHER *(Specify)*: ______
b. ADDRESS *(Street, City, State, ZIP Code)*	

5. PROBLEM AREAS

a. Paragraph Number and Wording:

b. Recommended Wording:

c. Reason/Rationale for Recommendation:

6. REMARKS

7a. NAME OF SUBMITTER *(Last, First, MI)* – Optional	b. WORK TELEPHONE NUMBER *(Include Area Code)* – Optional
c. MAILING ADDRESS *(Street, City, State, ZIP Code)* – Optional	8. DATE OF SUBMISSION *(YYMMDD)*

DD FORM 1426 82 MAR

PREVIOUS EDITION IS OBSOLETE.