NOT MEASUREMENT SENSITIVE

MIL-STD-498
5 December 1994
(PDF version)
Superseding
DOD-STD-2167A
29 February 1988
DOD-STD-7935A
31 October 1988
DOD-STD-1703(NS)
12 February 1987

MILITARY STANDARD

SOFTWARE DEVELOPMENT AND DOCUMENTATION

AMSC NO. N7069

AREA: IPSC/MCCR

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

FOREWORD

1. This Military Standard is approved 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 SPAWAR 10-12, 2451 Crystal Drive (CPK-5), Arlington, VA 22245-5200. The comments may be submitted by letter or by using the Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document.

3. This standard merges DOD-STD-2167A and DOD-STD-7935A to define a set of activities and documentation suitable for the development of both weapon systems and Automated Information Systems. A conversion guide from these standards to MIL-STD-498 is provided in Appendix I. Other changes include improved compatibility with incremental and evolutionary development models; improved compatibility with non-hierarchical design methods; improved compatibility with computer-aided software engineering (CASE) tools; alternatives to, and more flexibility in, preparing documents; clearer requirements for incorporating reusable software; introduction of software management indicators; added emphasis on software supportability; and improved links to systems engineering. This standard supersedes DOD-STD-2167A, DOD-STD-7935A, and DOD-STD-1703 (NS).

4. This standard can be applied in any phase of the system life cycle. It can be applied to contractors, subcontractors, or Government in-house agencies performing software development. For uniformity, the term "acquirer" is used for the organization requiring the technical effort, the term "developer" for the organization performing the technical effort, and the term "contract" for the agreement between them. The term "software development" is used as an inclusive term encompassing new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products.

5. This standard is not intended to specify or discourage the use of any particular software development method. The developer is responsible for selecting software development methods that support the achievement of contract requirements.

6. This standard implements the development and documentation processes of ISO/IEC DIS 12207. It interprets all applicable clauses in MIL-Q-9858A (Quality Program Requirements) and ISO 9001 (Quality Systems) for software.

7. This standard includes all activities pertaining to software development. It invokes no other standards. It can be applied on its own or supplemented with other standards, such as those identified in Section 6. If other standards are applied, the acquirer is responsible for resolving any conflicts that arise.

8. Data Item Descriptions (DIDs) applicable to this standard are listed in Section 6. These DIDs describe the information required by this standard.

9. This standard and its Data Item Descriptions (DIDs) are meant to be tailored by the acquirer to ensure that only necessary and cost-effective requirements are imposed on software development efforts. General tailoring guidance can be found in Section 6 and in DOD-HDBK-248. Tailoring guidance specific to this standard can be found in Appendixes G and H and in guidebooks and handbooks planned for this standard.

APPENDIXES

Figure Page

1. SCOPE

1.1 Purpose. The purpose of this standard is to establish uniform requirements for software development and documentation.

1.2 Application. MIL-STD-498 is intended to be applied as follows.

1.2.1 Organizations and agreements. This standard can be applied to contractors, subcontractors, or Government in-house agencies performing software development. For uniformity, the term "acquirer" is used for the organization requiring the technical effort, "developer" for the organization performing the technical effort, "contract" for the agreement between these parties, "Statement of Work" (SOW) for the list of tasks to be performed by the developer, "Contract Data Requirements List" (CDRL) for the list of deliverable software products, and "subcontractor" for any organization tasked by the developer to perform part of the required effort. "Software development" is used as an inclusive term encompassing new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products.

1.2.2 Contract-specific application. This standard is invoked by citing it on a contract. It applies to each software product and to each type of software covered by the contract, regardless of storage medium, to the extent specified in the contract. Examples of types of software include deliverable versus non-deliverable, software designed to meet user needs versus software in the engineering and test environments, and software designed to meet one user need versus software designed to meet another. The acquirer is expected to specify the types of software to which the standard applies and to tailor the standard appropriately for each type of software. If the standard is invoked without such a statement of selective application, it will be understood to apply in its entirety to all deliverable software, with requirements concerning the software development environment applicable to the software development environment for the deliverable software. While this standard is written in terms of Computer Software Configuration Items (CSCIs), it may be applied to software not designated as a CSCI, with the term "CSCI" interpreted appropriately. Software installed in firmware is subject to all of the aforementioned provisions. This standard does not apply to the hardware element of firmware.

1.2.3 Tailoring. This standard and its Data Item Descriptions (DIDs) are meant to be tailored for each type of software to which they are applied. While tailoring is the responsibility of the acquirer, suggested tailoring may be provided by prospective and selected developers. General tailoring guidance can be found in Section 6 and in DOD-HDBK-248. Tailoring guidance specific to this standard can be found in Appendixes G and H and in guidebooks and handbooks planned for this standard.

1.2.4 Interpretation of selected terms. The following terms have a special interpretation as used in this standard.

1.2.4.1 Interpretation of "system". The following interpretations apply:

a. The term "system," as used in this standard, may mean: (1) a hardware-software system (for example, a radar system) for which this standard covers only the software portion, or (2) a software system (for example, a payroll system) for which this standard governs overall development.

b. If a system consists of subsystems, all requirements in this standard concerning systems apply to the subsystems as well. If a contract is based on alternatives to systems and subsystems, such as complex items, the requirements in this standard concerning the system and its specification apply to these alternatives and their specifications.

1.2.4.2 Interpretation of "participate" in system development. The term "participate" in paragraphs regarding system-level activities is to be interpreted as follows: If the software covered by this standard is part of a hardware-software system for which this standard covers only the software portion, the term "participate" is to be interpreted as "take part in, as described in the software development plan." If the software (possibly with its computers) is considered to constitute a system, the term "participate" is to be interpreted as "be responsible for."

1.2.4.3 Interpretation of "develop," "define," etc. Throughout this standard, requirements to "develop," "define," "establish," or "identify" information are to be interpreted to include new development, modification, reuse, reengineering, maintenance, or any other activity or combination of activities resulting in software products.

1.2.4.4 Interpretation of "record". Throughout this standard, requirements to "record" information are to be interpreted to mean "set down in a manner that can be retrieved and viewed." The result may take many forms, including, but not limited to, hand-written notes, hard-copy or electronic documents, and data recorded in computer-aided software engineering (CASE) and project management tools.

1.3 Order of precedence. In the event of conflict between the requirements of this standard and other applicable standardization documents, the acquirer is responsible for resolving the conflicts.

2. REFERENCED DOCUMENTS

This section does not apply to this standard, since no documents are referenced in Sections 3, 4, or 5. Section 6 contains a list of standardization documents that may be used with this standard.

3. DEFINITIONS

Note: In addition to the definitions provided here, Section 1 describes MIL-STD-498's interpretation or special usage of the following terms: acquirer, contract, Contract Data Requirements List, define, develop, developer, establish, identify, participate, record, software development, Statement of Work, subcontractor, subsystem, and system.

3.1 Acceptance. An action by an authorized representative of the acquirer by which the acquirer assumes ownership of software products as partial or complete performance of a contract.

3.2 Acquirer. An organization that procures software products for itself or another organization.

3.3 Approval. Written notification by an authorized representative of the acquirer that a developer's plans, design, or other aspects of the project appear to be sound and can be used as the basis for further work. Such approval does not shift responsibility from the developer to meet contractual requirements.

3.4 Architecture. The organizational structure of a system or CSCI, identifying its components, their interfaces, and a concept of execution among them.

3.5 Associate developer. An organization that is neither prime contractor nor subcontractor to the developer, but who has a development role on the same or related system or project.

3.6 Behavioral design. The design of how an overall system or CSCI will behave, from a user's point of view, in meeting its requirements, ignoring the internal implementation of the system or CSCI. This design contrasts with architectural design, which identifies the internal components of the system or CSCI, and with the detailed design of those components.

3.7 Build. (1) A version of software that meets a specified subset of the requirements that the completed software will meet. (2) The period of time during which such a version is developed. Note: The relationship of the terms "build" and "version" is up to the developer; for example, it may take several versions to reach a build, a build may be released in several parallel versions (such as to different sites), or the terms may be used as synonyms.

3.8 Computer database. See database.

3.9 Computer hardware. Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data, or producing control outputs. Such devices can perform substantial interpretation, computation, communication, control, or other logical functions.

3.10 Computer program. A combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions.

3.11 Computer software. See software.

3.12 Computer Software Configuration Item (CSCI). An aggregation of software that satisfies an end use function and is designated for separate configuration management by the acquirer. CSCIs are selected based on tradeoffs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors.

3.13 Configuration Item. An aggregation of hardware, software, or both that satisfies an end use function and is designated for separate configuration management by the acquirer.

3.14 Database. A collection of related data stored in one or more computerized files in a manner that can be accessed by users or computer programs via a database management system.

3.15 Database management system. An integrated set of computer programs that provide the capabilities needed to establish, modify, make available, and maintain the integrity of a database.

3.16 Deliverable software product. A software product that is required by the contract to be delivered to the acquirer or other designated recipient.

3.17 Design. Those characteristics of a system or CSCI that are selected by the developer in response to the requirements. Some will match the requirements; others will be elaborations of requirements, such as definitions of all error messages in response to a requirement to display error messages; others will be implementation related, such as decisions about what software units and logic to use to satisfy the requirements.

3.18 Developer. An organization that develops software products ("develops" may include new development, modification, reuse, reengineering, maintenance, or any other activity that results in software products). The developer may be a contractor or a Government agency.

3.19 Document/documentation. A collection of data, regardless of the medium on which it is recorded, that generally has permanence and can be read by humans or machines.

3.20 Evaluation. The process of determining whether an item or activity meets specified criteria.

3.21 Firmware. The combination of a hardware device and computer instructions and/or computer data that reside as read-only software on the hardware device.

3.22 Hardware Configuration Item (HWCI). An aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the acquirer.

3.23 Independent verification and validation (IV&V). Systematic evaluation of software products and activities by an agency that is not responsible for developing the product or performing the activity being evaluated. IV&V is not within the scope of this standard.

3.24 Interface. In software development, a relationship among two or more entities (such as CSCI-CSCI, CSCI-HWCI, CSCI-user, or software unit-software unit) in which the entities share, provide, or exchange data. An interface is not a CSCI, software unit, or other system component; it is a relationship among them.

3.25 Joint review. A process or meeting involving representatives of both the acquirer and the developer, during which project status, software products, and/or project issues are examined and discussed.

3.26 Non-deliverable software product. A software product that is not required by the contract to be delivered to the acquirer or other designated recipient.

3.27 Process. An organized set of activities performed for a given purpose; for example, the software development process.

3.28 Qualification testing. Testing performed to demonstrate to the acquirer that a CSCI or a system meets its specified requirements.

3.29 Reengineering. The process of examining and altering an existing system to reconstitute it in a new form. May include reverse engineering (analyzing a system and producing a representation at a higher level of abstraction, such as design from code), restructuring (transforming a system from one representation to another at the same level of abstraction), redocumentation (analyzing a system and producing user or support documentation), forward engineering (using software products derived from an existing system, together with new requirements, to produce a new system), retargeting (transforming a system to install it on a different target system), and translation (transforming source code from one language to another or from one version of a language to another).

3.30 Requirement. (1) A characteristic that a system or CSCI must possess in order to be acceptable to the acquirer. (2) A mandatory statement in this standard or another portion of the contract.

3.31 Reusable software product. A software product developed for one use but having other uses, or one developed specifically to be usable on multiple projects or in multiple roles on one project. Examples include, but are not limited to, commercial off-the-shelf software products, acquirer-furnished software products, software products in reuse libraries, and pre-existing developer software products. Each use may include all or part of the software product and may involve its modification. This term can be applied to any software product (for example, requirements, architectures, etc.), not just to software itself.

3.32 Software. Computer programs and computer databases. Note: Although some definitions of software include documentation, MIL-STD-498 limits the definition to computer programs and computer databases in accordance with Defense Federal Acquisition Regulation Supplement 227.401.

3.33 Software development. A set of activities that results in software products. Software development may include new development, modification, reuse, reengineering, maintenance, or any other activities that result in software products.

3.34 Software development file (SDF). A repository for material pertinent to the development of a particular body of software. Contents typically include (either directly or by reference) considerations, rationale, and constraints related to requirements analysis, design, and implementation; developer-internal test information; and schedule and status information.

3.35 Software development library (SDL). A controlled collection of software, documentation, other intermediate and final software products, and associated tools and procedures used to facilitate the orderly development and subsequent support of software.

3.36 Software development process. An organized set of activities performed to translate user needs into software products.

3.37 Software engineering. In general usage, a synonym for software development. As used in this standard, a subset of software development consisting of all activities except qualification testing. The standard makes this distinction for the sole purpose of giving separate names to the software engineering and software test environments.

3.38 Software engineering environment. The facilities, hardware, software, firmware, procedures, and documentation needed to perform software engineering. Elements may include but are not limited to computer-aided software engineering (CASE) tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and database management systems.

3.39 Software product. Software or associated information created, modified, or incorporated to satisfy a contract. Examples include plans, requirements, design, code, databases, test information, and manuals.

3.40 Software quality. The ability of software to satisfy its specified requirements.

3.41 Software support. The set of activities that takes place to ensure that software installed for operational use continues to perform as intended and fulfill its intended role in system operation. Software support includes software maintenance, aid to users, and related activities.

3.42 Software system. A system consisting solely of software and possibly the computer equipment on which the software operates.

3.43 Software test environment. The facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification, and possibly other, testing of software. Elements may include but are not limited to simulators, code analyzers, test case generators, and path analyzers, and may also include elements used in the software engineering environment.

3.44 Software transition. The set of activities that enables responsibility for software development to pass from one organization, usually the organization that performs initial software development, to another, usually the organization that will perform software support.

3.45 Software unit. An element in the design of a CSCI; for example, a major subdivision of a CSCI, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

3.46 Support (of software). See software support.

3.47 Transition (of software). See software transition.

3.48 Definitions of acronyms used in this standard. See Appendix A.

4. GENERAL REQUIREMENTS

4.1 Software development process. The developer shall establish a software development process consistent with contract requirements. The software development process shall include the following major activities, which may overlap, may be applied iteratively, may be applied differently to different elements of software, and need not be performed in the order listed below. Appendix G provides examples. The developer's software development process shall be described in the software development plan.

a. Project planning and oversight (section 5.1)
b. Establishing a software development environment (5.2)
c. System requirements analysis (5.3)
d. System design (5.4)
e. Software requirements analysis (5.5)
f. Software design (5.6)
g. Software implementation and unit testing (5.7)
h. Unit integration and testing (5.8)
i. CSCI qualification testing (5.9)
j. CSCI/HWCI integration and testing (5.10)
k. System qualification testing (5.11)
l. Preparing for software use (5.12)
m. Preparing for software transition (5.13)
n. Integral processes:
    1) Software configuration management (5.14)
    2) Software product evaluation (5.15)
    3) Software quality assurance (5.16)
    4) Corrective action (5.17)
    5) Joint technical and management reviews (5.18)
    6) Other activities (5.19)

4.2 General requirements for software development. The developer shall meet the following general requirements in carrying out the detailed requirements in section 5 of this standard.

4.2.1 Software development methods. The developer shall use systematic, documented methods for all software development activities. These methods shall be described in, or referenced from, the software development plan.

4.2.2 Standards for software products. The developer shall develop and apply standards for representing requirements, design, code, test cases, test procedures, and test results. These standards shall be described in, or referenced from, the software development plan.

4.2.3 Reusable software products. The developer shall meet the following requirements.

4.2.3.1 Incorporating reusable software products. The developer shall identify and evaluate reusable software products for use in fulfilling the requirements of the contract. The scope of the search and the criteria to be used for evaluation shall be as described in the software development plan. Reusable software products that meet the criteria shall be used where practical. Appendix B provides required and candidate criteria and interprets this standard for incorporation of reusable software products. Incorporated software products shall meet the data rights requirements in the contract.

4.2.3.2 Developing reusable software products. During the course of the contract, the developer shall identify opportunities for developing software products for reuse and shall evaluate the benefits and costs of these opportunities. Opportunities that provide cost benefits and are compatible with program objectives shall be identified to the acquirer.

Note: In addition, the developer may be required by the contract to develop software products specifically for reuse.

4.2.4 Handling of critical requirements. The developer shall meet the following requirements.

4.2.4.1 Safety assurance. The developer shall identify as safety-critical those CSCIs or portions thereof whose failure could lead to a hazardous system state (one that could result in unintended death, injury, loss of property, or environmental harm). If there is such software, the developer shall develop a safety assurance strategy, including both tests and analyses, to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for hazardous conditions. The strategy shall include a software safety program, which shall be integrated with the system safety program if one exists. The developer shall record the strategy in the software development plan, implement the strategy, and produce evidence, as part of required software products, that the safety assurance strategy has been carried out.

4.2.4.2 Security assurance. The developer shall identify as security-critical those CSCIs or portions thereof whose failure could lead to a breach of system security. If there is such software, the developer shall develop a security assurance strategy to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for breaches of system security. The developer shall record the strategy in the software development plan, implement the strategy, and produce evidence, as part of required software products, that the security assurance strategy has been carried out.

4.2.4.3 Privacy assurance. The developer shall identify as privacy-critical those CSCIs or portions thereof whose failure could lead to a breach of system privacy. If there is such software, the developer shall develop a privacy assurance strategy to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for breaches of system privacy. The developer shall record the strategy in the software development plan, implement the strategy, and produce evidence, as part of required software products, that the privacy assurance strategy has been carried out.

4.2.4.4 Assurance of other critical requirements. If a system relies on software to satisfy other requirements deemed critical by the contract or by system specifications, the developer shall identify those CSCIs or portions thereof whose failure could lead to violation of those critical requirements; develop a strategy to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for such violations; record the strategy in the software development plan; implement the strategy; and produce evidence, as part of required software products, that the assurance strategy has been carried out.

4.2.5 Computer hardware resource utilization. The developer shall analyze contract requirements concerning computer hardware resource utilization (such as maximum allowable use of processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity). The developer shall allocate computer hardware resources among the CSCIs, monitor the utilization of these resources for the duration of the contract, and reallocate or identify the need for additional resources as necessary to meet contract requirements.

4.2.6 Recording rationale. The developer shall record rationale that will be useful to the support agency for key decisions made in specifying, designing, implementing, and testing the software. The rationale shall include trade-offs considered, analysis methods, and criteria used to make the decisions. The rationale shall be recorded in documents, code comments, or other media that will transition to the support agency. The meaning of "key decisions" and the approach for providing the rationale shall be described in the software development plan.

4.2.7 Access for acquirer review. The developer shall provide the acquirer or its authorized representative access to developer and subcontractor facilities, including the software engineering and test environments, for review of software products and activities required by the contract.

5. DETAILED REQUIREMENTS

The order of the requirements in this section is not intended to specify the order in which they must be carried out. Many of the activities may be ongoing at one time; different software products may proceed at different paces; and activities specified in early subsections may depend on input from activities in later subsections. If the software is developed in multiple builds, some activities may be performed in every build, others may be performed only in selected builds, and activities and software products may not be complete until several or all builds are accomplished. Figure 1 provides an example of how each activity may be applied in one or more builds. Non-mandatory notes throughout section 5 tell how to interpret each activity on a project involving multiple builds. A project involving a single build will accomplish all required activities in that build. Appendix G provides guidance for planning builds, determining which activities apply to each build, and scheduling these activities.

Activity	Builds			
	Build 1	Build 2	Build 3	Build 4
5.1 Project planning and oversight	x	x	x	x
5.2 Establishing a software development environment	x	x	x	x
5.3 System requirements analysis	x	x		
5.4 System design	x	x	x	
5.5 Software requirements analysis	x	x	x	x
5.6 Software design	x	x	x	x
5.7 Software implementation and unit testing	x	x	x	x
5.8 Unit integration and testing	x	x	x	x
5.9 CSCI qualification testing		x	x	x
5.10 CSCI/HWCI integration and testing		x	x	x
5.11 System qualification testing			x	x
5.12 Preparing for software use	x	x	x	x
5.13 Preparing for software transition				x
Integral processes:				
5.14 Software configuration management	x	x	x	x
5.15 Software product evaluation	x	x	x	x
5.16 Software quality assurance	x	x	x	x
5.17 Corrective action	x	x	x	x
5.18 Joint technical and management reviews	x	x	x	x
5.19 Other activities	x	x	x	x

FIGURE 1. One possible mapping of MIL-STD-498 activities to multiple builds.

5.1 Project planning and oversight. The developer shall perform project planning and oversight in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, planning for each build should be interpreted to include: a) overall planning for the contract, b) detailed planning for the current build, and c) planning for future builds covered under the contract to a level of detail compatible with the information available.

5.1.1 Software development planning. The developer shall develop and record plans for conducting the activities required by this standard and by other software-related requirements in the contract. This planning shall be consistent with system-level planning and shall include all applicable items in the Software Development Plan (SDP) DID (see 6.2).

Note 1: The wording here and throughout MIL-STD-498 is designed to: 1) Emphasize that the development and recording of planning and engineering information is an intrinsic part of the software development process, to be performed regardless of whether a deliverable is required; 2) Use the DID as a checklist of items to be covered in the planning or engineering activity; and 3) Permit representations other than traditional documents for recording the information (e.g., computer-aided software engineering (CASE) tools).

Note 2: If the CDRL specifies delivery of the information generated by this or any other paragraph, the developer is required to format, assemble, mark, copy, and distribute the deliverable in accordance with the CDRL. This task is recognized to be separate from the task of generating and recording the required information and to require additional time and effort on the part of the developer.

Note 3: The software development plan covers all activities required by this standard. Portions of the plan may be bound or maintained separately if this approach enhances the usability of the information. Examples include separate plans for software quality assurance and software configuration management.

5.1.2 CSCI test planning. The developer shall develop and record plans for conducting CSCI qualification testing. This planning shall include all applicable items in the Software Test Plan (STP) DID (see 6.2).

5.1.3 System test planning. The developer shall participate in developing and recording plans for conducting system qualification testing. For software systems, this planning shall include all applicable items in the Software Test Plan (STP) DID (see 6.2). (The intent for software systems is a single software test plan covering both CSCI and system qualification testing.)

5.1.4 Software installation planning. The developer shall develop and record plans for performing software installation and training at the user sites specified in the contract. This planning shall include all applicable items in the Software Installation Plan (SIP) DID (see 6.2).

5.1.5 Software transition planning. The developer shall identify all software development resources that will be needed by the support agency to fulfill the support concept specified in the contract. The developer shall develop and record plans identifying these resources and describing the approach to be followed for transitioning deliverable items to the support agency. This planning shall include all applicable items in the Software Transition Plan (STrP) DID (see 6.2).

5.1.6 Following and updating plans. Following acquirer approval of any of the plans in this section, the developer shall conduct the relevant activities in accordance with the plan. The developer's management shall review the software development process at intervals specified in the software development plan to assure that the process complies with the contract and adheres to the plans. With the exception of developer-internal scheduling and related staffing information, updates to plans shall be subject to acquirer approval.

5.2 Establishing a software development environment. The developer shall establish a software development environment in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, establishing the software development environment in each build should be interpreted to mean establishing the environment needed to complete that build.

5.2.1 Software engineering environment. The developer shall establish, control, and maintain a software engineering environment to perform the software engineering effort. The developer shall ensure that each element of the environment performs its intended functions.

5.2.2 Software test environment. The developer shall establish, control, and maintain a software test environment to perform qualification, and possibly other, testing of software. The developer shall ensure that each element of the environment performs its intended functions.

5.2.3 Software development library. The developer shall establish, control, and maintain a software development library (SDL) to facilitate the orderly development and subsequent support of software. The SDL may be an integral part of the software engineering and test environments. The developer shall maintain the SDL for the duration of the contract.

5.2.4 Software development files. The developer shall establish, control, and maintain a software development file (SDF) for each software unit or logically related group of software units, for each CSCI, and, as applicable, for logical groups of CSCIs, for subsystems, and for the overall system. The developer shall record information about the development of the software in appropriate SDFs and shall maintain the SDFs for the duration of the contract.

5.2.5 Non-deliverable software. The developer may use non-deliverable software in the development of deliverable software as long as the operation and support of the deliverable software after delivery to the acquirer do not depend on the non-deliverable software or provision is made to ensure that the acquirer has or can obtain the same software. The developer shall ensure that all non-deliverable software used on the project performs its intended functions.

5.3 System requirements analysis. The developer shall participate in system requirements analysis in accordance with the following requirements.

Note: If a system is developed in multiple builds, its requirements may not be fully defined until the final build. The developer's planning should identify the subset of system requirements to be defined in each build and the subset to be implemented in each build. System requirements analysis for a given build should be interpreted to mean defining the system requirements so identified for that build.

5.3.1 Analysis of user input. The developer shall participate in analyzing user input provided by the acquirer to gain an understanding of user needs. This input may take the form of need statements, surveys, problem/change reports, feedback on prototypes, interviews, or other user input or feedback.

5.3.2 Operational concept. The developer shall participate in defining and recording the operational concept for the system. The result shall include all applicable items in the Operational Concept Description (OCD) DID (see 6.2).

5.3.3 System requirements. The developer shall participate in defining and recording the requirements to be met by the system and the methods to be used to ensure that each requirement has been met. The result shall include all applicable items in the System/Subsystem Specification (SSS) DID (see 6.2). Depending on CDRL provisions, requirements concerning system interfaces may be included in the SSS or in interface requirements specifications (IRSs).

Note: If a system consists of subsystems, the activity in 5.3.3 is intended to be performed iteratively with the activities in 5.4 (System design) to define system requirements, design the system and identify its subsystems, define the requirements for those subsystems, design the subsystems and identify their components, and so on.

5.4 System design. The developer shall participate in system design in accordance with the following requirements.

Note: If a system is developed in multiple builds, its design may not be fully defined until the final build. The developer's planning should identify the portion of the system design to be defined in each build. System design for a given build should be interpreted to mean defining the portion of the system design identified for that build.

5.4.1 System-wide design decisions. The developer shall participate in defining and recording system-wide design decisions (that is, decisions about the system's behavioral design and other decisions affecting the selection and design of system components). The result shall include all applicable items in the system-wide design section of the System/Subsystem Design Description (SSDD) DID (see 6.2). Depending on CDRL provisions, design pertaining to interfaces may be included in the SSDD or in interface design descriptions (IDDs) and design pertaining to databases may be included in the SSDD or in database design descriptions (DBDDs).

Note: Design decisions remain at the discretion of the developer unless formally converted to requirements through contractual processes. The developer is responsible for fulfilling all requirements and demonstrating this fulfillment through qualification testing (see 5.9, 5.11). Design decisions act as developer-internal "requirements," to be implemented, imposed on subcontractors, if applicable, and confirmed by developer-internal testing, but their fulfillment need not be demonstrated to the acquirer.

5.4.2 System architectural design. The developer shall participate in defining and recording the architectural design of the system (identifying the components of the system, their interfaces, and a concept of execution among them) and the traceability between the system components and system requirements. The result shall include all applicable items in the architectural design and traceability sections of the System/Subsystem Design Description (SSDD) DID (see 6.2). Depending on CDRL provisions, design pertaining to interfaces may be included in the SSDD or in interface design descriptions (IDDs).

5.5 Software requirements analysis. The developer shall define and record the software requirements to be met by each CSCI, the methods to be used to ensure that each requirement has been met, and the traceability between the CSCI requirements and system requirements. The result shall include all applicable items in the Software Requirements Specification (SRS) DID (see 6.2). Depending on CDRL provisions, requirements concerning CSCI interfaces may be included in SRSs or in interface requirements specifications (IRSs).

Note: If a CSCI is developed in multiple builds, its requirements may not be fully defined until the final build. The developer's planning should identify the subset of each CSCI's requirements to be defined in each build and the subset to be implemented in each build. Software requirements analysis for a given build should be interpreted to mean defining the CSCI requirements so identified for that build.

5.6 Software design. The developer shall perform software design in accordance with the following requirements.

Note: If a CSCI is developed in multiple builds, its design may not be fully defined until the final build. Software design in each build should be interpreted to mean the design necessary to meet the CSCI requirements to be implemented in that build.

5.6.1 CSCI-wide design decisions. The developer shall define and record CSCI-wide design decisions (that is, decisions about the CSCI's behavioral design and other decisions affecting the selection and design of the software units comprising the CSCI). The result shall include all applicable items in the CSCI-wide design section of the Software Design Description (SDD) DID (see 6.2). Depending on CDRL provisions, design pertaining to interfaces may be included in SDDs or in interface design descriptions (IDDs) and design pertaining to databases may be included in SDDs or in database design descriptions (DBDDs).

5.6.2 CSCI architectural design. The developer shall define and record the architectural design of each CSCI (identifying the software units comprising the CSCI, their interfaces, and a concept of execution among them) and the traceability between the software units and the CSCI requirements. The result shall include all applicable items in the architectural design and traceability sections of the Software Design Description (SDD) DID (see 6.2). Depending on CDRL provisions, design pertaining to interfaces may be included in SDDs or in interface design descriptions (IDDs).

Note: Software units may be made up of other software units and may be organized into as many levels as are needed to represent the CSCI architecture. For example, a CSCI may be divided into three software units, each of which is divided into additional software units, and so on.

5.6.3 CSCI detailed design. The developer shall develop and record a description of each software unit. The result shall include all applicable items in the detailed design section of the Software Design Description (SDD) DID (see 6.2). Depending on CDRL provisions, design pertaining to interfaces may be included in SDDs or in interface design descriptions (IDDs) and design of software units that are databases or that access or manipulate databases may be included in SDDs or in database design descriptions (DBDDs).

5.7 Software implementation and unit testing. The developer shall perform software implementation and unit testing in accordance with the following requirements.

Note: The term "software" includes both computer programs and computer databases. The term "implementation" means converting software design into computer programs and computer databases. If a CSCI is developed in multiple builds, software implementation and unit testing of that CSCI will not be completed until the final build. Software implementation and unit testing in each build should be interpreted to include those units, or parts of units, needed to meet the CSCI requirements to be implemented in that build.

5.7.1 Software implementation. The developer shall develop and record software corresponding to each software unit in the CSCI design. This activity shall include, as applicable, coding computer instructions and data definitions, building databases, populating databases and other data files with data values, and other activities needed to implement the design. For deliverable software, the developer shall obtain acquirer approval to use any programming language not specified in the contract.

Note: Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

5.7.2 Preparing for unit testing. The developer shall establish test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data for testing the software corresponding to each software unit. The test cases shall cover all aspects of the unit's detailed design. The developer shall record this information in the appropriate software development files (SDFs).

5.7.3 Performing unit testing. The developer shall test the software corresponding to each software unit. The testing shall be in accordance with the unit test cases and procedures.

5.7.4 Revision and retesting. The developer shall make all necessary revisions to the software, perform all necessary retesting, and update the software development files (SDFs) and other software products as needed, based on the results of unit testing.

5.7.5 Analyzing and recording unit test results. The developer shall analyze the results of unit testing and shall record the test and analysis results in appropriate software development files (SDFs).

5.8 Unit integration and testing. The developer shall perform unit integration and testing in accordance with the following requirements.

Note 1: Unit integration and testing means integrating the software corresponding to two or more software units, testing the resulting software to ensure that it works together as intended, and continuing this process until all software in each CSCI is integrated and tested. The last stage of this testing is developer-internal CSCI testing. Since units may consist of other units, some unit integration and testing may take place during unit testing. The requirements in this section are not meant to duplicate those activities.

Note 2: If a CSCI is developed in multiple builds, unit integration and testing of that CSCI will not be completed until the final build. Unit integration and testing in each build should be interpreted to mean integrating software developed in the current build with other software developed in that and previous builds, and testing the results.

5.8.1 Preparing for unit integration and testing. The developer shall establish test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data for conducting unit integration and testing. The test cases shall cover all aspects of the CSCI-wide and CSCI architectural design. The developer shall record this information in the appropriate software development files (SDFs).

5.8.2 Performing unit integration and testing. The developer shall perform unit integration and testing. The testing shall be in accordance with the unit integration test cases and procedures.

5.8.3 Revision and retesting. The developer shall make all necessary revisions to the software, perform all necessary retesting, and update the software development files (SDFs) and other software products as needed, based on the results of unit integration and testing.

5.8.4 Analyzing and recording unit integration and test results. The developer shall analyze the results of unit integration and testing and shall record the test and analysis results in appropriate software development files (SDFs).

5.9 CSCI qualification testing. The developer shall perform CSCI qualification testing in accordance with the following requirements.

Note 1: CSCI qualification testing is performed to demonstrate to the acquirer that CSCI requirements have been met. It covers the CSCI requirements in software requirements specifications (SRSs) and in associated interface requirements specifications (IRSs). This testing contrasts with developer-internal CSCI testing, performed as the final stage of unit integration and testing.

Note 2: If a CSCI is developed in multiple builds, its CSCI qualification testing will not be completed until the final build for that CSCI, or possibly until later builds involving items with which the CSCI is required to interface. CSCI qualification testing in each build should be interpreted to mean planning and performing tests of the current build of each CSCI to ensure that the CSCI requirements to be implemented in that build have been met.

5.9.1 Independence in CSCI qualification testing. The person(s) responsible for qualification testing of a given CSCI shall not be the persons who performed detailed design or implementation of that CSCI. This does not preclude persons who performed detailed design or implementation of the CSCI from contributing to the process, for example, by contributing test cases that rely on knowledge of the CSCI's internal implementation.

5.9.2 Testing on the target computer system. CSCI qualification testing shall include testing on the target computer system or an alternative system approved by the acquirer.

5.9.3 Preparing for CSCI qualification testing. The developer shall define and record the test preparations, test cases, and test procedures to be used for CSCI qualification testing and the traceability between the test cases and the CSCI requirements. The result shall include all applicable items in the Software Test Description (STD) DID (see 6.2). The developer shall prepare the test data needed to carry out the test cases and provide the acquirer advance notice of the time and location of CSCI qualification testing.

5.9.4 Dry run of CSCI qualification testing. If CSCI qualification testing is to be witnessed by the acquirer, the developer shall dry run the CSCI test cases and procedures to ensure that they are complete and accurate and that the software is ready for witnessed testing. The developer shall record the results of this activity in appropriate software development files (SDFs) and shall update the CSCI test cases and procedures as appropriate.

5.9.5 Performing CSCI qualification testing. The developer shall perform CSCI qualification testing of each CSCI. The testing shall be in accordance with the CSCI test cases and procedures.

5.9.6 Revision and retesting. The developer shall make necessary revisions to the software, provide the acquirer advance notice of retesting, conduct all necessary retesting, and update the software development files (SDFs) and other software products as needed, based on the results of CSCI qualification testing.

5.9.7 Analyzing and recording CSCI qualification test results. The developer shall analyze and record the results of CSCI qualification testing. The results shall include all applicable items in the Software Test Report (STR) DID (see 6.2).

5.10 CSCI/HWCI integration and testing. The developer shall participate in CSCI/HWCI integration and testing activities in accordance with the following requirements.

Note 1: CSCI/HWCI integration and testing means integrating CSCIs with interfacing HWCIs and CSCIs, testing the resulting groupings to determine whether they work together as intended, and continuing this process until all CSCIs and HWCIs in the system are integrated and tested. The last stage of this testing is developer-internal system testing.

Note 2: If a system or CSCI is developed in multiple builds, CSCI/HWCI integration and testing may not be complete until the final build. CSCI/HWCI integration and testing in each build should be interpreted to mean integrating the current build of each CSCI with the current build of other CSCIs and HWCIs and testing the results to ensure that the system requirements to be implemented in that build have been met.

5.10.1 Preparing for CSCI/HWCI integration and testing. The developer shall participate in developing and recording test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data for conducting CSCI/HWCI integration and testing. The test cases shall cover all aspects of the system-wide and system architectural design. The developer shall record software-related information in appropriate software development files (SDFs).

5.10.2 Performing CSCI/HWCI integration and testing. The developer shall participate in CSCI/HWCI integration and testing. The testing shall be in accordance with the CSCI/HWCI integration test cases and procedures.

5.10.3 Revision and retesting. The developer shall make necessary revisions to the software, participate in all necessary retesting, and update the appropriate software development files (SDFs) and other software products as needed, based on the results of CSCI/HWCI integration and testing.

5.10.4 Analyzing and recording CSCI/HWCI integration and test results. The developer shall participate in analyzing the results of CSCI/HWCI integration and testing. Software-related analysis and test results shall be recorded in appropriate software development files (SDFs).

5.11 System qualification testing. The developer shall participate in system qualification testing in accordance with the following requirements.

Note 1: System qualification testing is performed to demonstrate to the acquirer that system requirements have been met. It covers the system requirements in the system/subsystem specifications (SSSs) and in associated interface requirements specifications (IRSs). This testing contrasts with developer-internal system testing, performed as the final stage of CSCI/HWCI integration and testing.

Note 2: If a system is developed in multiple builds, qualification testing of the completed system will not occur until the final build. System qualification testing in each build should be interpreted to mean planning and performing tests of the current build of the system to ensure that the system requirements to be implemented in that build have been met.

5.11.1 Independence in system qualification testing. The person(s) responsible for fulfilling the requirements in this section shall not be the persons who performed detailed design or implementation of software in the system. This does not preclude persons who performed detailed design or implementation of software in the system from contributing to the process, for example, by contributing test cases that rely on knowledge of the system's internal implementation.

5.11.2 Testing on the target computer system. The developer's system qualification testing shall include testing on the target computer system or an alternative system approved by the acquirer.

5.11.3 Preparing for system qualification testing. The developer shall participate in developing and recording the test preparations, test cases, and test procedures to be used for system qualification testing and the traceability between the test cases and the system requirements. For software systems, the results shall include all applicable items in the Software Test Description (STD) DID (see 6.2). The developer shall participate in preparing the test data needed to carry out the test cases and in providing the acquirer advance notice of the time and location of system qualification testing.

5.11.4 Dry run of system qualification testing. If system qualification testing is to be witnessed by the acquirer, the developer shall participate in dry running the system test cases and procedures to ensure that they are complete and accurate and that the system is ready for witnessed testing. The developer shall record the software-related results of this activity in appropriate software development files (SDFs) and shall participate in updating the system test cases and procedures as appropriate.

5.11.5 Performing system qualification testing. The developer shall participate in system qualification testing. This participation shall be in accordance with the system test cases and procedures.

5.11.6 Revision and retesting. The developer shall make necessary revisions to the software, provide the acquirer advance notice of retesting, participate in all necessary retesting, and update the software development files (SDFs) and other software products as needed, based on the results of system qualification testing.

5.11.7 Analyzing and recording system qualification test results. The developer shall participate in analyzing and recording the results of system qualification testing. For software systems, the result shall include all applicable items in the Software Test Report (STR) DID (see 6.2).

5.12 Preparing for software use. The developer shall prepare for software use in accordance with the following requirements.

Note: If software is developed in multiple builds, the developer's planning should identify what software, if any, is to be fielded to users in each build and the extent of fielding (for example, full fielding or fielding to selected evaluators only). Preparing for software use in each build should be interpreted to include those activities necessary to carry out the fielding plans for that build.

5.12.1 Preparing the executable software. The developer shall prepare the executable software for each user site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s). The result shall include all applicable items in the executable software section of the Software Product Specification (SPS) DID (see 6.2).

Note: To order only the executable software (delaying delivery of source files and associated support information to a later build), the acquirer can use the SPS DID, tailoring out all but the executable software section of that DID.

5.12.2 Preparing version descriptions for user sites. The developer shall identify and record the exact version of software prepared for each user site. The information shall include all applicable items in the Software Version Description (SVD) DID (see 6.2).

5.12.3 Preparing user manuals. The developer shall prepare user manuals in accordance with the following requirements.

Note: Few, if any, systems will need all of the manuals in this section. The intent is for the acquirer, with input from the developer, to determine which manuals are appropriate for a given system and to require the development of only those manuals. All DIDs permit substitution of commercial or other manuals that contain the required information. The manuals in this section are normally developed in parallel with software development, ready for use in CSCI testing.

5.12.3.1 Software user manuals. The developer shall identify and record information needed by hands-on users of the software (persons who will both operate the software and make use of its results). The information shall include all applicable items in the Software User Manual (SUM) DID (see 6.2).

5.12.3.2 Software input/output manuals. The developer shall identify and record information needed by persons who will submit inputs to, and receive outputs from, the software, relying on others to operate the software in a computer center or other centralized or networked software installation. The information shall include all applicable items in the Software Input/Output Manual (SIOM) DID (see 6.2).

5.12.3.3 Software center operator manuals. The developer shall identify and record information needed by persons who will operate the software in a computer center or other centralized or networked software installation, so that it can be used by others. The information shall include all applicable items in the Software Center Operator Manual (SCOM) DID (see 6.2).

5.12.3.4 Computer operation manuals. The developer shall identify and record information needed to operate the computers on which the software will run. The information shall include all applicable items in the Computer Operation Manual (COM) DID (see 6.2).

5.12.4 Installation at user sites. The developer shall:

a. Install and check out the executable software at the user sites specified in the contract.

b. Provide training to users as specified in the contract.

c. Provide other assistance to user sites as specified in the contract.

5.13 Preparing for software transition. The developer shall prepare for software transition in accordance with the following requirements.

Note: If software is developed in multiple builds, the developer's planning should identify what software, if any, is to be transitioned to the support agency in each build. Preparing for software transition in each build should be interpreted to include those activities necessary to carry out the transition plans for that build.

5.13.1 Preparing the executable software. The developer shall prepare the executable software to be transitioned to the support site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s). The result shall include all applicable items in the executable software section of the Software Product Specification (SPS) DID (see 6.2).

5.13.2 Preparing source files. The developer shall prepare the source files to be transitioned to the support site, including any batch files, command files, data files, or other files needed to regenerate the executable software. The result shall include all applicable items in the source file section of the Software Product Specification (SPS) DID (see 6.2).

5.13.3 Preparing version descriptions for the support site. The developer shall identify and record the exact version of software prepared for the support site. The information shall include all applicable items in the Software Version Description (SVD) DID (see 6.2).

5.13.4 Preparing the "as built" CSCI design and related information. The developer shall update the design description of each CSCI to match the "as built" software and shall define and record: the methods to be used to verify copies of the software, the measured computer hardware resource utilization for the CSCI, other information needed to support the software, and traceability between the CSCI's source files and software units and between the computer hardware resource utilization measurements and the CSCI requirements concerning them. The result shall include all applicable items in the qualification, software support, and traceability sections of the Software Product Specification (SPS) DID (see 6.2).

Note: In hardware development, the final product is an approved design from which hardware items can be manufactured. This design is presented in the product specification. In software development, by contrast, the final product is the software, not its design, and "manufacturing" consists of electronically duplicating the software, not recreating it from the design. The "as built" design is included in the software product specification not as the product but as information that may help the support agency understand the software in order to modify, enhance, and otherwise support it.

5.13.5 Updating the system design description. The developer shall participate in updating the system design description to match the "as built" system. The result shall include all applicable items in the System/Subsystem Design Description (SSDD) DID (see 6.2).

5.13.6 Preparing support manuals. The developer shall prepare support manuals in accordance with the following requirements.

Note: Not all systems will need the manuals in this section. The intent is for the acquirer, with input from the developer, to determine which manuals are appropriate for a given system and to require the development of only those manuals. All DIDs permit substitution of commercial or other manuals that contain the required information. The manuals in this section supplement the system/subsystem design description (SSDD) and the software product specifications (SPSs), which serve as the primary sources of information for software support. The user manuals cited in 5.12.3 are also useful to support personnel.

5.13.6.1 Computer programming manuals. The developer shall identify and record information needed to program the computers on which the software was developed or on which it will run. The information shall include all applicable items in the Computer Programming Manual (CPM) DID (see 6.2).

5.13.6.2 Firmware support manuals. The developer shall identify and record information needed to program and reprogram any firmware devices in which the software will be installed. The information shall include all applicable items in the Firmware Support Manual (FSM) DID (see 6.2).

5.13.7 Transition to the designated support site. The developer shall:

a. Install and check out the deliverable software in the support environment designated in the contract.

b. Demonstrate to the acquirer that the deliverable software can be regenerated (compiled/linked/loaded into an executable product) and maintained using commercially available, acquirer-owned, or contractually deliverable software and hardware designated in the contract or approved by the acquirer.

c. Provide training to the support agency as specified in the contract.

d. Provide other assistance to the support agency as specified in the contract.

5.14 Software configuration management. The developer shall perform software configuration management in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, the software products of each build may be refinements of, or additions to, software products of previous builds. Software configuration management in each build should be understood to take place in the context of the software products and controls in place at the start of the build.

5.14.1 Configuration identification. The developer shall participate in selecting CSCIs, as performed under system architectural design in 5.4.2, shall identify the entities to be placed under configuration control, and shall assign a project-unique identifier to each CSCI and each additional entity to be placed under configuration control. These entities shall include the software products to be developed or used under the contract and the elements of the software development environment. The identification scheme shall be at the level at which entities will actually be controlled, for example, computer files, electronic media, documents, software units, configuration items. The identification scheme shall include the version/revision/release status of each entity.

5.14.2 Configuration control. The developer shall establish and implement procedures designating the levels of control each identified entity must pass through (for example, author control, project-level control, acquirer control); the persons or groups with authority to authorize changes and to make changes at each level (for example, the programmer/analyst, the software lead, the project manager, the acquirer); and the steps to be followed to request authorization for changes, process change requests, track changes, distribute changes, and maintain past versions. Changes that affect an entity already under acquirer control shall be proposed to the acquirer in accordance with contractually established forms and procedures, if any.

Note: A number of requirements in this standard refer to "project-level or higher configuration control." If "project-level" is not a level of control selected for the project, the software development plan should state how these requirements map to the selected levels.

5.14.3 Configuration status accounting. The developer shall prepare and maintain records of the configuration status of all entities that have been placed under project-level or higher configuration control. These records shall be maintained for the life of the contract. They shall include, as applicable, the current version/revision/release of each entity, a record of changes to the entity since being placed under project-level or higher configuration control, and the status of problem/change reports affecting the entity.

5.14.4 Configuration audits. The developer shall support acquirer-conducted configuration audits as specified in the contract.

Note: These configuration audits may be called Functional Configuration Audits and Physical Configuration Audits.

5.14.5 Packaging, storage, handling, and delivery. The developer shall establish and implement procedures for the packaging, storage, handling, and delivery of deliverable software products. The developer shall maintain master copies of delivered software products for the duration of the contract.

5.15 Software product evaluation. The developer shall perform software product evaluation in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, the software products of each build should be evaluated in the context of the objectives established for that build. A software product that meets those objectives can be considered satisfactory even though it is missing information designated for development in later builds.

5.15.1 In-process and final software product evaluations. The developer shall perform in-process evaluations of the software products generated in carrying out the requirements of this standard. In addition, the developer shall perform a final evaluation of each deliverable software product before its delivery. The software products to be evaluated, criteria to be used, and definitions for those criteria are given in Appendix D.

5.15.2 Software product evaluation records. The developer shall prepare and maintain records of each software product evaluation. These records shall be maintained for the life of the contract. Problems in software products under project-level or higher configuration control shall be handled as described in 5.17 (Corrective action).

5.15.3 Independence in software product evaluation. The persons responsible for evaluating a software product shall not be the persons who developed the product. This does not preclude the persons who developed the software product from taking part in the evaluation (for example, as participants in a walk-through of the product).

5.16 Software quality assurance. The developer shall perform software quality assurance in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, the activities and software products of each build should be evaluated in the context of the objectives established for that build. An activity or software product that meets those objectives can be considered satisfactory even though it is missing aspects designated for later builds. Planning for software quality assurance is included in software development planning (see 5.1.1).

5.16.1 Software quality assurance evaluations. The developer shall conduct on-going evaluations of software development activities and the resulting software products to:

a. Assure that each activity required by the contract or described in the software development plan is being performed in accordance with the contract and with the software development plan.

b. Assure that each software product required by this standard or by other contract provisions exists and has undergone software product evaluations, testing, and corrective action as required by this standard and by other contract provisions.

5.16.2 Software quality assurance records. The developer shall prepare and maintain records of each software quality assurance activity. These records shall be maintained for the life of the contract. Problems in software products under project-level or higher configuration control and problems in activities required by the contract or described in the software development plan shall be handled as described in 5.17 (Corrective action).

5.16.3 Independence in software quality assurance. The persons responsible for conducting software quality assurance evaluations shall not be the persons who developed the software product, performed the activity, or are responsible for the software product or activity. This does not preclude such persons from taking part in these evaluations. The persons responsible for assuring compliance with the contract shall have the resources, responsibility, authority, and organizational freedom to permit objective software quality assurance evaluations and to initiate and verify corrective actions.

5.17 Corrective action. The developer shall perform corrective action in accordance with the following requirements.

5.17.1 Problem/change reports. The developer shall prepare a problem/change report to describe each problem detected in software products under project-level or higher configuration control and each problem in activities required by the contract or described in the software development plan. The problem/change report shall describe the problem, the corrective action needed, and the actions taken to date. These reports shall serve as input to the corrective action system.

5.17.2 Corrective action system. The developer shall implement a corrective action system for handling each problem detected in software products under project-level or higher configuration control and each problem in activities required by the contract or described in the software development plan. The system shall meet the following requirements:

a. Inputs to the system shall consist of problem/change reports.

b. The system shall be closed-loop, ensuring that all detected problems are promptly reported and entered into the system, action is initiated on them, resolution is achieved, status is tracked, and records of the problems are maintained for the life of the contract.

c. Each problem shall be classified by category and priority, using the categories and priorities in Appendix C or approved alternatives.

d. Analysis shall be performed to detect trends in the problems reported.

e. Corrective actions shall be evaluated to determine whether problems have been resolved, adverse trends have been reversed, and changes have been correctly implemented without introducing additional problems.

5.18 Joint technical and management reviews. The developer shall plan and take part in joint (acquirer/developer) technical and management reviews in accordance with the following requirements.

Note: If a system or CSCI is developed in multiple builds, the types of joint reviews held and the criteria applied will depend on the objectives of each build. Software products that meet those objectives can be considered satisfactory even though they are missing information designated for development in later builds.

5.18.1 Joint technical reviews. The developer shall plan and take part in joint technical reviews at locations and dates proposed by the developer and approved by the acquirer. These reviews shall be attended by persons with technical knowledge of the software products to be reviewed. The reviews shall focus on in-process and final software products, rather than materials generated especially for the review. The reviews shall have the following objectives:

a. Review evolving software products, using as criteria the software product evaluation criteria in Appendix D; review and demonstrate proposed technical solutions; provide insight and obtain feedback on the technical effort; surface and resolve technical issues.

b. Review project status; surface near- and long-term risks regarding technical, cost, and schedule issues.

c. Arrive at agreed-upon mitigation strategies for identified risks, within the authority of those present.

d. Identify risks and issues to be raised at joint management reviews.

e. Ensure on-going communication between acquirer and developer technical personnel.

5.18.2 Joint management reviews. The developer shall plan and take part in joint management reviews at locations and dates proposed by the developer and approved by the acquirer. These reviews shall be attended by persons with authority to make cost and schedule decisions and shall have the following objectives. Examples of such reviews are identified in Appendix E.

a. Keep management informed about project status, directions being taken, technical agreements reached, and overall status of evolving software products.

b. Resolve issues that could not be resolved at joint technical reviews.

c. Arrive at agreed-upon mitigation strategies for near- and long-term risks that could not be resolved at joint technical reviews.

d. Identify and resolve management-level issues and risks not raised at joint technical reviews.

e. Obtain commitments and acquirer approvals needed for timely accomplishment of the project.

5.19 Other activities. The developer shall perform the following activities.

5.19.1 Risk management. The developer shall perform risk management throughout the software development process. The developer shall identify, analyze, and prioritize the areas of the software development project that involve potential technical, cost, or schedule risks; develop strategies for managing those risks; record the risks and strategies in the software development plan; and implement the strategies in accordance with the plan.

5.19.2 Software management indicators. The developer shall use software management indicators to aid in managing the software development process and communicating its status to the acquirer. The developer shall identify and define a set of software management indicators, including the data to be collected, the methods to be used to interpret and apply the data, and the planned reporting mechanism. The developer shall record this information in the software development plan and shall collect, interpret, apply, and report on those indicators as described in the plan. Candidate indicators are given in Appendix F.

5.19.3 Security and privacy. The developer shall meet the security and privacy requirements specified in the contract. These requirements may affect the software development effort, the resulting software products, or both.

5.19.4 Subcontractor management. If subcontractors are used, the developer shall include in subcontracts all contractual requirements necessary to ensure that software products are developed in accordance with prime contract requirements.

5.19.5 Interface with software IV&V agents. The developer shall interface with the software Independent Verification and Validation (IV&V) agent(s) as specified in the contract.

5.19.6 Coordination with associate developers. The developer shall coordinate with associate developers, working groups, and interface groups as specified in the contract.

5.19.7 Improvement of project processes. The developer shall periodically assess the processes used on the project to determine their suitability and effectiveness. Based on these assessments, the developer shall identify any necessary and beneficial improvements to the process, shall identify these improvements to the acquirer in the form of proposed updates to the software development plan and, if approved, shall implement the improvements on the project.

6. NOTES

(This section contains information of a general or explanatory nature that may be helpful, but is not mandatory.)

6.1 Intended use. This standard contains requirements for the development and documentation of software. Its application is described in 1.2.

6.2 Data requirements. The following Data Item Descriptions (DIDs) must be listed, as applicable, on the Contract Data Requirements List (DD Form 1423) when this standard is applied on a contract, in order to obtain the data, except where DOD FAR Supplement 227.405-70 exempts the requirement for a DD Form 1423.

Reference Para	DID Number	DID Title
5.1.1	DI-IPSC-81427	Software Development Plan (SDP)
5.1.2, 5.1.3	DI-IPSC-81438	Software Test Plan (STP)
5.1.4	DI-IPSC-81428	Software Installation Plan (SIP)
5.1.5	DI-IPSC-81429	Software Transition Plan (STrP)
5.3.2	DI-IPSC-81430	Operational Concept Description (OCD)
5.3.3	DI-IPSC-81431	System/Subsystem Specification (SSS)
5.3.3, 5.5	DI-IPSC-81434	Interface Requirements Specification (IRS)
5.4.1, 5.4.2, 5.13.5	DI-IPSC-81432	System/Subsystem Design Description (SSDD)
5.4.1, 5.4.2, 5.6.1, 5.6.2, 5.6.3	DI-IPSC-81436	Interface Design Description (IDD)
5.5	DI-IPSC-81433	Software Requirements Specification (SRS)
5.6.1, 5.6.2, 5.6.3	DI-IPSC-81435	Software Design Description (SDD)
5.4.1, 5.6.1, 5.6.3	DI-IPSC-81437	Database Design Description (DBDD)
5.9.3, 5.11.3	DI-IPSC-81439	Software Test Description (STD)
5.9.7, 5.11.7	DI-IPSC-81440	Software Test Report (STR)
5.12.1, 5.13.1, 5.13.2, 5.13.4	DI-IPSC-81441	Software Product Specification (SPS)
5.12.2, 5.13.3	DI-IPSC-81442	Software Version Description (SVD)
5.12.3.1	DI-IPSC-81443	Software User Manual (SUM)
5.12.3.2	DI-IPSC-81445	Software Input/Output Manual (SIOM)
5.12.3.3	DI-IPSC-81444	Software Center Operator Manual (SCOM)
5.12.3.4	DI-IPSC-81446	Computer Operation Manual (COM)
5.13.6.1	DI-IPSC-81447	Computer Programming Manual (CPM)
5.13.6.2	DI-IPSC-81448	Firmware Support Manual (FSM)

The above DIDs were those cleared as of the date of this standard. The current issue of DOD 5010.12, Acquisition Management Systems and Data Requirements Control List (AMSDL), must be researched to ensure that only current, cleared DIDs are cited on the Form 1423.

6.3 Relationship between standard and CDRL. If the CDRL calls for a DID different from the one named in corresponding paragraph(s) of this standard, all references to the DID in the standard should be interpreted to mean the one in the CDRL.

6.4 Delivery of tool contents. Depending on contract provisions, the developer may be permitted to satisfy CDRL requirements by delivering: 1) a repository or database containing the information specified in the cited DID; 2) a means of accessing that repository or database, such as a CASE tool, if not already available to the recipients designated on the CDRL; and 3) a hard-copy or electronically stored table of contents, specifying how and where to access the information required in each paragraph of the DID.

6.5 Tailoring guidance. This standard and its Data Item Descriptions (DIDs) are applied at the discretion of the acquirer. In each application, the standard and DIDs should be tailored to the specific requirements of a particular program, program phase, or contractual structure. Care should be taken to eliminate tasks that add unnecessary costs and data that do not add value to the process or the product. Tailoring for the standard takes the form of deletion of activities, alteration of activities to more explicitly reflect the application to a particular effort, or addition of activities to satisfy program requirements. This tailoring is specified in the Statement of Work. Tailoring for the DIDs consists of deleting requirements for unneeded information and making other changes, such as combining two documents under one cover, that do not increase the required workload. DID tailoring for deliverables is specified in Block 16 of the CDRL.

6.6 Cost/schedule reporting. Developer cost/schedule reports should be prepared at the CSCI level. The cost reports should indicate budgeted versus actual expenditures and should conform to the Work Breakdown Structure (WBS) applicable to the development effort. These reports should also indicate to the acquirer planned, actual, and predicted progress.

6.7 Related standardization documents. Figure 2 identifies a set of standardization documents related to software development. These and other standardization documents may be imposed or quoted in the Statement of Work to supplement the requirements in MIL-STD-498. MIL-STD-498 does not invoke these documents. The acquirer should use caution to ensure that supplemental standards are appropriate to the project and that any conflicts among these standards or with MIL-STD-498 are identified and resolved.

6.8 Subject term (key word) listing. The following list of key words may be used to catalog or characterize key topics in this standard.

Builds/incremental development
Computer software configuration item
Database
Joint technical/management reviews
Operational concept
Reusable software
Risk management
Security/privacy
Software
Software configuration management
Software development
Software documentation
Software implementation
Software management indicators
Software product evaluation
Software quality assurance
Software requirements analysis
Software safety
Software support
Software testing
Software unit
Tailoring

Topic and MIL-STD-498 Paragraph	Related Standardization Documents (Determine latest version before use)
Behavioral design (5.4.1, 5.6.1)	MIL-STD-1801, User Computer Interface MIL-HDBK-761, Human Engineering Guidelines for Management Information Systems
Computer security (4.2.4.2)	DOD-5200.28 STD, DoD Trusted Computer System Evaluation Criteria
Configuration management (5.14)	ANSI/IEEE Std 828, Standard for Software Configuration Management Plans ANSI/IEEE Std 1042, Guide to Software Configuration Management MIL-STD-973, Configuration Management MIL-HDBK-61 Guidelines for Configuration Management
Continuous acquisition and life-cycle support (CALS)	MIL-STD-1840, Automated Interchange of Technical Information MIL-STD-1556, Government-Industry Data Exchange Program MIL-HDBK-59, Continuous Acquisition and Life-Cycle Support Program Implementation Guide MIL-HDBK-800, Documentation Streamlining MIL-D-28000, Digital Representation for Communication of Product Data: IGES Application Subset and IGES Application Protocols MIL-M-28001, Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange of Text MIL-R-28002, Requirements for Raster Graphics Representation in Binary Format MIL-D-28003, Digital Representation for Communication of Illustration Data: CGM Application Profile
Joint technical and management reviews (5.18, App. E)	ANSI/IEEE Std 1028, Standard for Software Reviews and Audits MIL-STD-499, Engineering Management MIL-STD-1521, Technical Reviews and Audits for Systems, Equipments, and Computer Software (audit portion superseded by MIL-STD-973)
Programming languages (5.7.1)	FIPS-PUB-119, Ada (Also issued as ANSI/ISO/IEC 8652; formerly ANSI/MIL-STD-1815, Ada Programming Language)
Software design (5.4, 5.6)	ANSI/IEEE Std 1016, Recommended Practice for Software Design Descriptions IEEE Std 1016.1, Guide for Software Design Descriptions IEEE/ANSI Std 990, Recommended Practice for Ada as a Program Design Language
Software development environment (5.2)	IEEE Std 1209, Recommended Practice for the Evaluation and Selection of CASE Tools DOD-STD-1467 (AR), Software Support Environment MIL-HDBK-782 (AR), Software Support Environment Acquisition
Software development planning (5.1.1)	ANSI/IEEE Std 1058.1, Standard for Software Project Management Plans
Software development process (4.1, App. G)	ISO/IEC 12207 (when issued), Software Life-Cycle Processes ANSI/IEEE Std 1074, Standard for Developing Software Life Cycle Processes MIL-STD-1803 (USAF), Software Development Integrity Program Guidebook on MIL-STD-498 (when issued) MIL-HDBK-498 (when issued)

Note: MIL-STD-498 does not invoke any of these documents.

FIGURE 2. Related standardization documents.

Topic and MIL-STD-498 Paragraph	Related Standardization Documents (Determine latest version before use)
Software management indicators (5.19.2, App. F)	ISO/IEC 9126, Quality Characteristics and Guidelines for Their Use ANSI/IEEE Std 982.2, Guide: Use of Standard Measures to Produce Reliable Software IEEE Std 1045, Standard for Software Productivity Metrics IEEE Std 1061, Standard for Software Quality Metrics Methodology
Software problem categories/priorities (Appendix C)	IEEE Std 1044, Standard Classification for Software Anomalies
Software product evaluation (5.15)	ANSI/IEEE Std 1012, Standard for Software Verification and Validation Plans IEEE Std 1059, Guide for Verification and Validation Plans
Software quality assurance (5.16)	ISO 9001, Quality System - Model for Quality Assurance in Design/Development, Production, Installation, and Servicing ISO 9000-3, Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software ANSI/IEEE Std 730, Standard for Software Quality Assurance Plans IEEE Std 1298/A3563.1, Software Quality Management System DOD-STD-2168, Defense System Software Quality Program MIL-HDBK-286, A Guide for DOD-STD-2168
Software requirements (5.3.3, 5.5)	ANSI/IEEE Std 830, Recommended Practice for Software Requirements Specifications MIL-STD-490, Specification Practices
Software safety (4.2.4.1)	MIL-STD-882, System Safety Program Requirements MIL-HDBK-272, Safety Design and Evaluation Criteria for Nuclear Weapons Systems IEEE Std 1228, Standard for Software Safety Plans
Software support (all paragraphs)	IEEE Std 1219, Standard for Software Maintenance MIL-HDBK-347, Mission-Critical Computer Resources Software Support
Software testing (5.1.2, 5.1.3, 5.7 - 5.11)	ANSI/IEEE Std 829, Standard for Software Test Documentation ANSI/IEEE Std 1008, Standard for Software Unit Testing ANSI/IEEE Std 1012, Standard for Software Verification and Validation Plans IEEE Std 1059, Guide for Verification and Validation Plans
Software user documentation (5.12.3)	ANSI/IEEE Std 1063, Standard for Software User Documentation
Systems engineering (5.1.3, 5.3, 5.4, 5.10, 5.11)	MIL-STD-499, Engineering Management MIL-HDBK-805, Microcomputer Software and Hardware Guidelines
Tailoring (1.2.3, 6.5, App. G, H	DOD-HDBK-248, Guide for Application and Tailoring of Requirements for Defense Materiel Acquisitions MIL-HDBK-498 (when issued)
Training (5.12.4, 5.13.7)	MIL-STD-1379, Military Training Programs
Work breakdown structure (6.6)	MIL-STD-881, Work Breakdown Structures for Defense Materiel Items

Note: MIL-STD-498 does not invoke any of these documents.

FIGURE 2. Related standardization documents - continued.

APPENDIX A

LIST OF ACRONYMS

A.1 Scope. This appendix provides a list of acronyms used in this standard, with their associated meanings. This appendix is not a mandatory part of the standard. The information provided is intended for guidance only.

A.2 Applicable documents. This section is not applicable to this appendix.

A.3 Acronyms.

CASE	Computer-Aided Software Engineering
CDRL	Contract Data Requirements List
COM	Computer Operation Manual
CPM	Computer Programming Manual
CSCI	Computer Software Configuration Item
DBDD	Database Design Description
DID	Data Item Description
DoD	Department of Defense
FSM	Firmware Support Manual
HWCI	Hardware Configuration Item
IDD	Interface Design Description
IRS	Interface Requirements Specification
IV&V	Independent Verification and Validation
OCD	Operational Concept Description
SCOM	Software Center Operator Manual
SDD	Software Design Description
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SIOM	Software Input/Output Manual
SIP	Software Installation Plan
SOW	Statement of Work
SPS	Software Product Specification
SRS	Software Requirements Specification
SSDD	System/Subsystem Design Description
SSS	System/Subsystem Specification
STD	Software Test Description
STP	Software Test Plan
STR	Software Test Report
STrP	Software Transition Plan
SUM	Software User Manual
SVD	Software Version Description
SW	Software
WBS	Work Breakdown Structure

APPENDIX B

INTERPRETING MIL-STD-498 FOR INCORPORATION OF REUSABLE SOFTWARE PRODUCTS

B.1 Scope. This appendix interprets MIL-STD-498 when applied to the incorporation of reusable software products. This appendix is a mandatory part of this standard, subject to tailoring by the acquirer.

B.2 Applicable documents. This section is not applicable to this appendix.

B.3 Evaluating reusable software products. The developer shall specify in the software development plan the criteria to be used for evaluating reusable software products for use in fulfilling the requirements of the contract. General criteria shall be the software product's ability to meet specified requirements and to be cost-effective over the life of the system. Non-mandatory examples of specific criteria include, but are not limited to:

a. Ability to provide required capabilities and meet required constraints
b. Ability to provide required safety, security, and privacy
c. Reliability/maturity, as evidenced by established track record
d. Testability
e. Interoperability with other system and system-external elements
f. Fielding issues, including:
    1) Restrictions on copying/distributing the software or documentation
    2) License or other fees applicable to each copy
g. Maintainability, including:
    1) Likelihood the software product will need to be changed
    2) Feasibility of accomplishing that change
    3) Availability and quality of documentation and source files
    4) Likelihood that the current version will continue to be supported by the supplier
    5) Impact on the system if the current version is not supported
    6) The acquirer's data rights to the software product
    7) Warranties available
h. Short- and long-term cost impacts of using the software product
i. Technical, cost, and schedule risks and tradeoffs in using the software product

B.4 Interpreting MIL-STD-498 activities for reusable software products. The following rules apply in interpreting this standard:

a. Any requirement that calls for development of a software product may be met by a reusable software product that fulfills the requirement and meets the criteria established in the software development plan. The reusable software product may be used as-is or modified and may be used to satisfy part or all of the requirement. For example, a requirement may be met by using an existing plan, specification, or design.

b. When the reusable software product to be incorporated is the software itself, some of the requirements in this standard require special interpretation. Figure 3 provides this interpretation. Key issues are whether the software will be modified, whether unmodified software constitutes an entire CSCI or only one or more software units, and whether unmodified software has a positive performance record (no firm criteria exist for making this determination). The figure is presented in a conditional manner: If an activity in the left column is required for a given type of software, the figure tells how to interpret the activity for reusable software of that type.

If this MIL-STD-498 activity is required:	Interpret the activity as follows for each type of existing, reusable software:				
	For CSCIs to be used unmodified		For software units to be used unmodified		For software units being modified for/ during project
	Positive performance record	No or poor performance record	Positive performance record	No or poor performance record	
5.1 Project planning and oversight	Include the activities in this figure in project plans				
5.2 Establishing software devel environment	Establish and apply a software test environment, software development library, and software development files as appropriate to perform the activities in this figure				Apply full requirements
5.3 System requirements analysis	Consider software's capabilities in defining the operational concept & system requirements				
	Use test/ performance records to confirm ability to meet needs	Test to confirm ability to meet needs	Use test/ performance records to confirm ability to meet needs	Test to confirm ability to meet needs	Use tests or records to determine potential to meet needs
5.4.1 System-wide design	Consider the software's capabilities and characteristics in designing system behavior and in making other system-wide design decisions				
5.4.2 System architectural design	Include the CSCI in the system architecture; allocate system requirements to it		Consider the unit's capabilities and characteristics in designating CSCIs and allocating system requirements to them		
5.5 Software requirements analysis	Specify the project-specific requirements the CSCI must meet; verify via records or retest that the CSCI can meet them		Consider the unit's capabilities and characteristics in specifying the requirements for the CSCI of which it is a part		
5.6.1 CSCI-wide design	No requirement: the CSCI-wide design decisions have already been made (recording the "as built" design is under 5.13)		Consider the unit's capabilities and characteristics in designing CSCI behavior and making other CSCI-wide design decisions		
5.6.2 CSCI architectural design	No requirement: the CSCI's architecture is already defined (recording the "as built" design is under 5.13)		Include the unit in the CSCI architecture and allocate CSCI requirements to it		
5.6.3 CSCI detailed design	No requirement: the CSCI's detailed design is already defined (recording the "as built" design is under 5.13)		No requirement: the unit is already designed (recording the "as built" design is under 5.13)		Modify the unit's design as needed
5.7.1 Software implementation	No requirement: the software for the CSCI's units is already implemented		No requirement: the software for the unit is already implemented		Modify the software for the unit
5.7.2-5.7.5 Unit testing	No requirement: the CSCI's units are already tested	Perform selectively if in question and units are accessible	No requirement: the unit is already tested	Perform this testing	

FIGURE 3. Interpreting MIL-STD-498 for incorporation of reusable software.

If this MIL-STD-498 activity is required:	Interpret the activity as follows for each type of existing, reusable software:				
	For CSCIs to be used unmodified		For software units to be used unmodified		For software units being modified for/ during project
	Positive performance record	No or poor performance record	Positive performance record	No or poor performance record	
5.8 Unit integration and testing	No requirement: the CSCI's units are already integrated	Perform selectively if in question and units are accessible	Perform except where integration is already tested/proven	Perform this testing	
5.9 CSCI qualification testing	No requirement: CSCI is already tested & proven	Perform this testing	Include the unit in CSCI qualification testing		
5.10 CSCI/HWCI integration and testing	Perform, except where integration is already tested/proven	Include the CSCI in CSCI/HWCI integration and testing	Include the unit in CSCI/HWCI integration and testing		
5.11 System qualification testing	Include the CSCI in system qualification testing		Include the unit in system qualification testing		
5.12 Preparing for software use	Include the software for the CSCI or unit in the executable software; include in version descriptions; handle any license issues; cover use of the CSCI or unit, as appropriate, via existing, new, or revised user/operator manuals; install the CSCI or unit as part of the overall system; include its use, as appropriate, in the training offered				
5.13 Preparing for software transition	Include the software for the CSCI or unit in the executable software; prepare source files for the CSCI or unit, if available; include in version descriptions; handle any license issues; prepare or provide "as built" design descriptions for software whose design is known; install the CSCI or unit at the support site; demonstrate regenerability if source is available; include in the training offered				
5.14 Software configuration management	Apply to all software products prepared, modified, or used in incorporating this software				
5.15 Software product evaluation	Apply to all software products prepared or modified in incorporating this software; for software products used unchanged, apply unless a positive performance record or evidence of past evaluations indicates that such an evaluation would be duplicative				
5.16 Software quality assurance	Apply to all activities performed and all software products prepared, modified, or used in incorporating this software				
5.17 Corrective action	Apply to all activities performed and all software products prepared or modified in incorporating this software				
5.18 Joint reviews	Cover the software products prepared or modified in incorporating this software				
5.19 Other activities	Apply the full requirements of this section				

FIGURE 3. Interpreting MIL-STD-498 for incorporation of reusable software - continued.

APPENDIX C

CATEGORY AND PRIORITY CLASSIFICATIONS FOR PROBLEM REPORTING

C.1 Scope. This appendix contains requirements for a category and priority classification scheme to be applied to each problem submitted to the corrective action system. This appendix is a mandatory part of the standard, subject to the following conditions: 1) these requirements may be tailored by the acquirer, and 2) the developer may use alternate category and priority schemes if approved by the acquirer.

C.2 Applicable documents. This section is not applicable to this appendix.

C.3 Classification by category. The developer shall:

a. Assign each problem in software products to one or more of the categories in Figure 4.

b. Assign each problem in activities to one or more of the categories in Figure 1 (shown at the start of Section 5).

C.4 Classification by priority. The developer shall assign each problem in software products or activities to one of the priorities in Figure 5.

Category	Applies to problems in:
a. Plans	One of the plans developed for the project
b. Concept	The operational concept
c. Requirements	The system or software requirements
d. Design	The design of the system or software
e. Code	The software code
f. Database/data file	A database or data file
g. Test information	Test plans, test descriptions, or test reports
h. Manuals	The user, operator, or support manuals
i. Other	Other software products

FIGURE 4. Categories to be used for classifying problems in software products.

Priority	Applies if a problem could:
1	a. Prevent the accomplishment of an operational or mission essential capability b. Jeopardize safety, security, or other requirement designated "critical"
2	a. Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known
3	a. Adversely affect the accomplishment of an operational or mission essential capability but a work-around solution is known b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known
4	a. Result in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capability b. Result in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of those responsibilities
5	Any other effect

FIGURE 5. Priorities to be used for classifying problems.

APPENDIX D

SOFTWARE PRODUCT EVALUATIONS

D.1 Scope. This appendix identifies the software products that are to undergo software product evaluations, identifies the criteria to be used for each evaluation, and contains a default set of definitions for the evaluation criteria. This appendix is a mandatory part of the standard, subject to the following conditions: 1) these requirements may be tailored by the acquirer, 2) the developer may use alternate criteria or definitions if approved by the acquirer, and 3) if the development of a given software product has been tailored out of the standard, the requirement to evaluate that product does not apply.

D.2 Applicable documents. This section is not applicable to this appendix.

D.3 Required evaluations. Figure 6 identifies the software products that are to undergo software product evaluations and states the criteria to be applied to each one. Each software product and criterion is labelled for purposes of identification and tailoring. For convenience, they may be treated as subparagraphs of this paragraph (referring to the first criterion, for example, as D.3.1.a). The software products are expressed in lower case letters to convey generic products, not necessarily in the form of hard-copy documents. Evaluations of system-level products are to be interpreted as participation in these evaluations. Some of the criteria are subjective. Because of this, there is no requirement to prove that the criteria have been met; the requirement is to perform the evaluations using these criteria and to identify possible problems for discussion and resolution.

D.4 Criteria definitions. The following paragraphs provide definitions for the criteria in Figure 6 that may not be self-explanatory. The criteria are listed in alphabetical order, matching as closely as possible the wording used in Figure 6.

D.4.1 Accurately describes (an item). This criterion, applied to user/operator/programmer instructions and to the "as built" design and version descriptions, means that the instructions or descriptions are correct depictions of the software or other item described.

D.4.2 Adequate test cases, procedures, data, results. Test cases are adequate if they cover all applicable requirements or design decisions and specify the inputs to be used, the expected results, and the criteria to be used for evaluating those results. Test procedures are adequate if they specify the steps to be followed in carrying out each test case. Test data are adequate if they enable the execution of the planned test cases and test procedures. Test or dry run results are adequate if they describe the results of all test cases and show that all criteria have been met, possibly after revision and retesting.

D.4.3 Consistent with indicated product(s). This criterion means that: (1) no statement or representation in one software product contradicts a statement or representation in the other software products, (2) a given term, acronym, or abbreviation means the same thing in all of the software products, and (3) a given item or concept is referred to by the same name or description in all of the software products.

D.4.4 Contains all applicable information in (a specified DID). This criterion uses the DIDs to specify the required content of software products, regardless of whether a deliverable document has been ordered. Allowances are to be made for the applicability of each DID topic. The formatting specified in the DID (required paragraphing and numbering) are not relevant to this evaluation.

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets CDRL, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
1. Software development plan (5.1.1)	a. SDP DID	b.	c.	d.	e.	f. (Up-dates)	g. Covers all activities/deliverables in SOW and CDRL h. Consistent with other project plans i. Presents a sound approach to the development
2. Software test plan (5.1.2, 5.1.3)	a. STP DID	b.	c.	d.	e.	f.	g. Covers all software-related qualification activities in the SOW h. Covers all requirements for the items under test i. Consistent with other project plans j. Presents a sound approach to the testing
3. Software installation plan (5.1.4)	a. SIP DID	b.	c.	d.	e.	f.	g. Covers all user site installation activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the installation
4. Software transition plan (5.1.5)	a. STrP DID	b.	c.	d.	e.	f.	g. Covers all transition-related activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the transition
5. Operational concept (5.3.2)	a. OCD DID	b.	c.	d.	e.	f.	g. Feasible
6. System requirements (5.3.3)	a. SSS, IRS DIDs	b.	c.	d.	e.	f.	g. Covers the operational concept h. Feasible i. Testable
7. System-wide design decisions (5.4.1)	a. SSDD, IDD, DBDD DIDs	b.	c.	d.	e.	f.	g. Consistent with system requirements h. Feasible
8. System architectural design (5.4.2)	a. SSDD, IDD DIDs	b.	c.	d.	e.	f.	g. Covers the system requirements h. Consistent with the system-wide design decisions i. Feasible
9. CSCI requirements (5.5)	a. SRS, IRS DIDs	b.	c.	d.	e.	f.	g. Covers system requirements allocated to the CSCI h. Feasible i. Testable

FIGURE 6. Software products and associated evaluation criteria.

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets CDRL, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
10. CSCI-wide design decisions (5.6.1)	a. SDD, IDD, DBDD DIDs	b.	c.	d.	e.	f.	g. Consistent with CSCI requirements h. Feasible
11. CSCI architectural design (5.6.2)	a. SDD, IDD DIDs	b.	c.	d.	e.	f.	g. Covers CSCI requirements h. Consistent with CSCI-wide design decisions i. Feasible
12. CSCI detailed design (5.6.3)	a. SDD, IDD, DBDD DIDs	b.	c.	d.	e.	f.	g. Covers CSCI requirements allocated to each unit h. Consistent with CSCI-wide design decisions
13. Implemented software (5.7.1)	N/A	b.	c.	d.	e.	f.	g. Covers the CSCI detailed design
14. CSCI qualification test descriptions (5.9.3)	a. STD DID	b.	c.	d.	e.	f.	g. Covers all CSCI requirements
15. CSCI qualification test results (5.9.7)	a. STR DID	b.	c.	d.	e.	f.	g. Covers all planned CSCI qualification test cases h. Shows evidence that the CSCI meets its requirements
16. System qualification test descriptions (5.11.3)	a. STD DID	b.	c.	d.	e.	f.	g. Covers all system requirements
17. System qualification test results (5.11.7)	a. STR DID	b.	c.	d.	e.	f.	g. Covers all planned system qualification test cases h. Shows evidence the system meets its requirements
18. Executable software (5.12.1, 5.13.1)	N/A	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All software necessary for execution is present i. Version exactly matches version that passed testing j. Deliverable media accurately labelled

FIGURE 6. Software products and associated evaluation criteria - continued.

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets CDRL, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
19. Software version descriptions (5.12.2, 5.13.3)	a. SVD DID	b.	c.	d.	e.	f.	g. Accurately identifies the version of each software component (file, unit, CSCI, etc.) delivered h. Accurately identifies the changes incorporated
20. Software user manuals (5.12.3.1)	a. SUM DID	b.	c.	d.	e.	f.	g. Accurately describes software installation and use to the intended audience of this manual
21. Software input/ output manuals (5.12.3.2)	a. SIOM DID	b.	c.	d.	e.	f.	g. Accurately describes software input/output to the intended audience of this manual
22. Software center operator manuals (5.12.3.3)	a. SCOM DID	b.	c.	d.	e.	f.	g. Accurately describes software installation and operation to the intended audience of this manual
23. Computer operation manuals (5.12.3.4)	a. COM DID	b.	c.	d.	e.	f.	g. Accurately describes the operational characteristics of the computer
24. Source files (5.13.2)	a. SPS DID	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All required software is present i. Version exactly matches version that passed testing j. Deliverable media accurately labelled
25. "As built" CSCI design and related information (5.13.4)	a. SPS DID	b.	c.	d.	e.	f.	g. Accurately describes the "as built" design of the CSCI h. Accurately describes compilation/build procedures i. Accurately describes modification procedures j. Source files cover all units in the CSCI design k. Measured resource utilization meets CSCI requirements
26. "As built" system design (5.13.5)	a. SSDD DID	b.	c.	d.	e.	f.	g. Accurately describes the "as built" system design

FIGURE 6. Software products and associated evaluation criteria - continued.

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets CDRL, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
27. Computer programming manuals (5.13.6.1)	a. CPM DID	b.	c.	d.	e.	f.	g. Accurately describes the programming features of the computer
28. Firmware support manuals (5.13.6.2)	a. FSM DID	b.	c.	d.	e.	f.	g. Accurately describes firmware programming features
29. Sampling of software development files (5.7.2, 5.7.3, 5.8.1, 5.8.4, 5.9.4, 5.10.1, 5.10.4, 5.11.4)	N/A	b.	N/A	d.	e.	f.	g. Contents are current with the ongoing effort h. Adequate unit test cases/procedures/data/results i. Adequate unit integration test cases/procedures/ data/results j. Adequate CSCI qualification dry run results k. Adequate CSCI/HWCI integration test cases/ procedures/data/results l. Adequate system qualification dry run results

FIGURE 6. Software products and associated evaluation criteria - continued.

D.4.5 Covers (a given set of items). A software product "covers" a given set of items if every item in the set has been dealt with in the software product. For example, a plan covers the SOW if every provision in the SOW is dealt with in the plan; a design covers a set of requirements if every requirement has been dealt with in the design; a test plan covers a set of requirements if every requirement is the subject of one or more tests. "Covers" corresponds to the downward traceability (for example, from requirements to design) in the requirement, design, and test planning/description DIDs.

D.4.6 Feasible. This criterion means that, in the knowledge and experience of the evaluator, a given concept, set of requirements, design, test, etc. violates no known principles or lessons learned that would render it impossible to carry out.

D.4.7 Follows software development plan. This criterion means that the software product shows evidence of having been developed in accordance with the approach described in the software development plan. Examples include following design and coding standards described in the plan. For the software development plan itself, this criterion applies to updates to the initial plan.

D.4.8 Internally consistent. This criterion means that: (1) no two statements or representations in a software product contradict one another, (2) a given term, acronym, or abbreviation means the same thing throughout the software product, and (3) a given item or concept is referred to by the same name or description throughout the software product.

D.4.9 Meets CDRL, if applicable. This criterion applies if the software product being evaluated is specified in the CDRL and has been formatted for delivery at the time of evaluation. It focuses on the format, markings, and other provisions specified in the CDRL, rather than on content, covered by other criteria.

D.4.10 Meets SOW, if applicable. This criterion means that the software product fulfills any Statement of Work provisions regarding it. For example, the Statement of Work may place constraints on the operational concept or the design.

D.4.11 Presents a sound approach. This criterion means that, based on the knowledge and experience of the evaluator, a given plan represents a reasonable way to carry out the required activities.

D.4.12 Shows evidence that (an item under test) meets its requirements. This criterion means that recorded test results show that the item under test either passed all tests the first time or was revised and retested until the tests were passed.

D.4.13 Testable. A requirement or set of requirements is considered to be testable if an objective and feasible test can be designed to determine whether each requirement has been met.

D.4.14 Understandable. This criterion means "understandable by the intended audience." For example, software products intended for programmer-to-programmer communication need not be understandable by non-programmers. A product that correctly identifies its audience (based on information in Block 3 of the corresponding DID) and is considered understandable to that audience meets this criterion.

APPENDIX E

CANDIDATE JOINT MANAGEMENT REVIEWS

E.1 Scope. This appendix describes a candidate set of joint management reviews that might be held during a software development project. This appendix is not a mandatory part of this standard. The information provided is intended for guidance only.

E.2 Applicable documents. This section is not applicable to this appendix.

E.3 Assumptions. This appendix makes the following assumptions:

a. The acquirer has reviewed the subject products in advance, and one or more joint technical reviews have been held to resolve issues, leaving the joint management review as a forum to resolve open issues and reach agreement as to the acceptability of each product.

b. Any of the reviews may be conducted incrementally, dealing at each review with a subset of the listed items or a subset of the system or CSCI(s) being reviewed.

E.4 Candidate reviews. Given below is a set of candidate joint management reviews that might be held during a software development project. There is no intent to require these reviews or to preclude alternatives or combinations of these reviews. The objectives supplement those given in 5.18.2.

E.4.1 Software plan reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The software development plan
b. The software test plan
c. The software installation plan
d. The software transition plan

E.4.2 Operational concept reviews. These reviews are held to resolve open issues regarding the operational concept for a software system.

E.4.3 System/subsystem requirements reviews. These reviews are held to resolve open issues regarding the specified requirements for a software system or subsystem.

E.4.4 System/subsystem design reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The system- or subsystem-wide design decisions
b. The architectural design of a software system or subsystem

E.4.5 Software requirements reviews. These reviews are held to resolve open issues regarding the specified requirements for a CSCI.

E.4.6 Software design reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The CSCI-wide design decisions
b. The architectural design of a CSCI
c. The detailed design of a CSCI or portion thereof (such as a database)

E.4.7 Test readiness reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The status of the software test environment
b. The test cases and test procedures to be used for CSCI qualification testing or system qualification testing
c. The status of the software to be tested

E.4.8 Test results reviews. These reviews are held to resolve open issues regarding the results of CSCI qualification testing or system qualification testing.

E.4.9 Software usability reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The readiness of the software for installation at user sites
b. The user and operator manuals
c. The software version descriptions
d. The status of installation preparations and activities

E.4.10 Software supportability reviews. These reviews are held to resolve open issues regarding one or more of the following:

a. The readiness of the software for transition to the support agency
b. The software product specifications
c. The software support manuals
d. The software version descriptions
e. The status of transition preparations and activities, including transition of the software development environment, if applicable

E.4.11 Critical requirement reviews. These reviews are held to resolve open issues regarding the handling of critical requirements, such as those for safety, security, and privacy.

APPENDIX F

CANDIDATE MANAGEMENT INDICATORS

F.1 Scope. This appendix identifies a set of management indicators that might be used on a software development project. This appendix is not a mandatory part of this standard. The information provided is intended for guidance only.

F.2 Applicable documents. This section is not applicable to this appendix.

F.3 Candidate indicators. Given below is a set of candidate management indicators that might be used on a software development project. There is no intent to impose these indicators or to preclude others.

a. Requirements volatility: total number of requirements and requirement changes over time.

b. Software size: planned and actual number of units, lines of code, or other size measurement over time.

c. Software staffing: planned and actual staffing levels over time.

d. Software complexity: complexity of each software unit.

e. Software progress: planned and actual number of software units designed, implemented, unit tested, and integrated over time.

f. Problem/change report status: total number, number closed, number opened in the current reporting period, age, priority.

g. Build release content: planned and actual number of software units released in each build.

h. Computer hardware resource utilization: planned and actual use of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity) over time.

i. Milestone performance: planned and actual dates of key project milestones.

j. Scrap/rework: amount of resources expended to replace or revise software products after they are placed under project-level or higher configuration control.

k. Effect of reuse: a breakout of each of the indicators above for reused versus new software products.

APPENDIX G

GUIDANCE ON PROGRAM STRATEGIES, TAILORING, AND BUILD PLANNING

G.1 Scope. This appendix identifies three of the program strategies used by DoD and shows how MIL-STD-498 can be applied under each of these strategies and on a project involving reengineering. This appendix is not a mandatory part of the standard. The information provided is intended for guidance only.

G.2 Applicable documents. Documents cited in this appendix are as follows:

a. DODI 5000.2, Defense Acquisition Management Policies and Procedures

b. DODI 8120.2, Automated Information System Life-Cycle Management Process, Review, and Milestone Approval

G.3 Candidate program strategies. DODI 8120.2 describes three basic program strategies plus a generic strategy called "other," encompassing variations, combinations, and alternatives to the three. DODI 5000.2 identifies similar strategies, called acquisition strategies. The three basic strategies are summarized below and in Figure 7.

a. Grand design. The "grand design" strategy (not named in DODI 5000.2 but treated as one strategy) is essentially a "once-through, do-each-step-once" strategy. Simplistically: determine user needs, define requirements, design the system, implement the system, test, fix, and deliver.

b. Incremental. The "incremental" strategy (called "Preplanned Product Improvement" in DODI 5000.2) determines user needs and defines the system requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the system is complete.

c. Evolutionary. The "evolutionary" strategy (called "evolutionary" in both DOD Instructions) also develops a system in builds, but differs from the incremental strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, user needs and system requirements are partially defined up front, then are refined in each succeeding build.

Program Strategy	Define All Requirements First?	Multiple Development Cycles?	Field Interim Software?
Grand Design	Yes	No	No
Incremental (Preplanned Product Improvement)	Yes	Yes	Maybe
Evolutionary	No	Yes	Yes

FIGURE 7. Key features of three DOD program strategies.

G.4 Selecting an appropriate program strategy. The program strategy is selected by the acquirer, but may be proposed by prospective or selected developers. Figure 8 illustrates a risk analysis approach for selecting an appropriate strategy. The approach consists of listing risk items (negatives) and opportunity items (positives) for each strategy; assigning each item a risk or opportunity level of High, Medium, or Low; and making a decision on which strategy to use based on a trade-off among the risks and opportunities. The fill-ins shown are sample considerations only. An actual analysis may use others. The "DECISION" entry on the bottom line shows which strategy was selected.

G.5 Relationship of MIL-STD-498 to program strategies. The program strategy usually applies to the overall system. The software within the system may be acquired under the same strategy or under a different one, such as requiring that all software be finalized in the first build of the system. Figures 9, 10, and 11 show how MIL-STD-498 might be applied under each of the program strategies identified in G.3. Figure 12 shows how MIL-STD-498 might be applied on a reengineering project. All four figures are, by necessity, simplified. For example, they show MIL-STD-498 activities in sequence when they might actually be ongoing, overlapping, or iterative; they show each software product as a single entity, without depicting early drafts or updates; and they represent each software product by the name of the corresponding DID, when the actual software product is the information called for by the DID, not necessarily in the form of a hard-copy document.

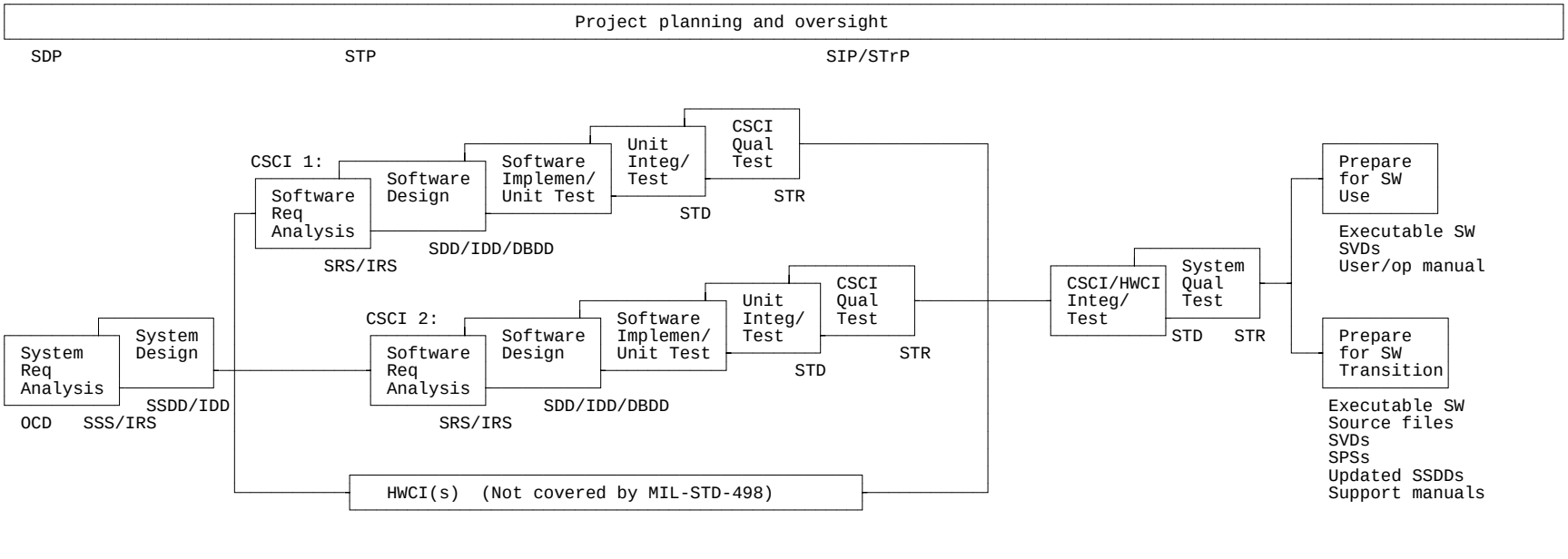
G.6 Planning software builds and tailoring MIL-STD-498. Planning the software builds on a project and tailoring MIL-STD-498 for each build may be accomplished in several ways. The acquirer might, for example, select an overall program strategy and tailor the standard for the overall contract, leaving it to the developer to lay out the software builds and propose the tailoring for each build. Alternatively, the acquirer might lay out the software builds and specify the tailoring for each as part of the contract. The approach selected will be project-dependent. The paragraphs below provide guidelines for planning the builds and tailoring the standard without attempting to divide these activities between the acquirer and developer.

G.6.1 Identifying builds and their objectives. The first step in software build planning is to lay out a series of one or more builds and to identify the objectives of each build. The top part of Figure 13 illustrates such planning. In the example, the system/subsystem specification (SSS) already exists and fulfillment of its requirements is divided into four builds, two of which will be prototypes delivered to a selected set of users, and two of which will actually be fielded. A further objective of Build 4 is transitioning the software to the designated support agency. An actual project would expand on these objectives.

G.6.2 Identifying the MIL-STD-498 activities to be performed in each build. The next step in build planning is identifying which MIL-STD-498 activities apply in each build and determining the extent to which they apply. The lower part of Figure 13 shows the start of such planning. Listed on the left are the paragraphs of MIL-STD-498. The worksheet entries indicate in which builds each activity is to be performed and include any notes regarding the nature of each activity in each build. For example, the figure shows that each build will include software development planning (5.1.1), but that the nature of that planning changes in each build. Some activities will not apply at all in a given build, some will apply identically in all builds, and some will apply differently in different builds. Since some aspects of the project, such as number and type of CSCIs, may not have been identified at the time the worksheet is being filled out, completion of the worksheet may itself be incremental. The following guidelines apply:

Grand Design		Incremental		Evolutionary	
Risk Item (Reasons against this strategy)	Risk Level	Risk Item (Reasons against this strategy)	Risk Level	Risk Item (Reasons against this strategy)	Risk Level
- Requirements are not well understood	H	- Requirements are not well understood	H		
- System too large to do all at once	M	- User prefers all capabilities at first delivery	M	- User prefers all capabilities at first delivery	M
- Rapid changes in mission technology anticipated--may change the requirements	H	- Rapid changes in mission technology are expected--may change the requirements	H		
- Limited staff or budget available now	M				
Opportunity Item (Reasons to use this strategy)	Opp. Level	Opportunity Item (Reasons to use this strategy)	Opp. Level	Opportunity Item (Reasons to use this strategy)	Opp. Level
- User prefers all capabilities at first delivery	M	- Early capability is needed	H	- Early capability is needed	H
- User prefers to phase out old system all at once	L	- System breaks naturally into increments	M	- System breaks naturally into increments	M
		- Funding/staffing will be incremental	H	- Funding/staffing will be incremental	H
				- User feedback and monitoring of technology changes is needed to understand full requirements	H
				DECISION: USE THIS STRATEGY	

FIGURE 8. Sample risk analysis for determining the appropriate program strategy.



Note: All activities may be more ongoing, overlapping, and iterative than the figure is able to show.

FIGURE 9. One possible way of applying MIL-STD-498 to the Grand Design program strategy.

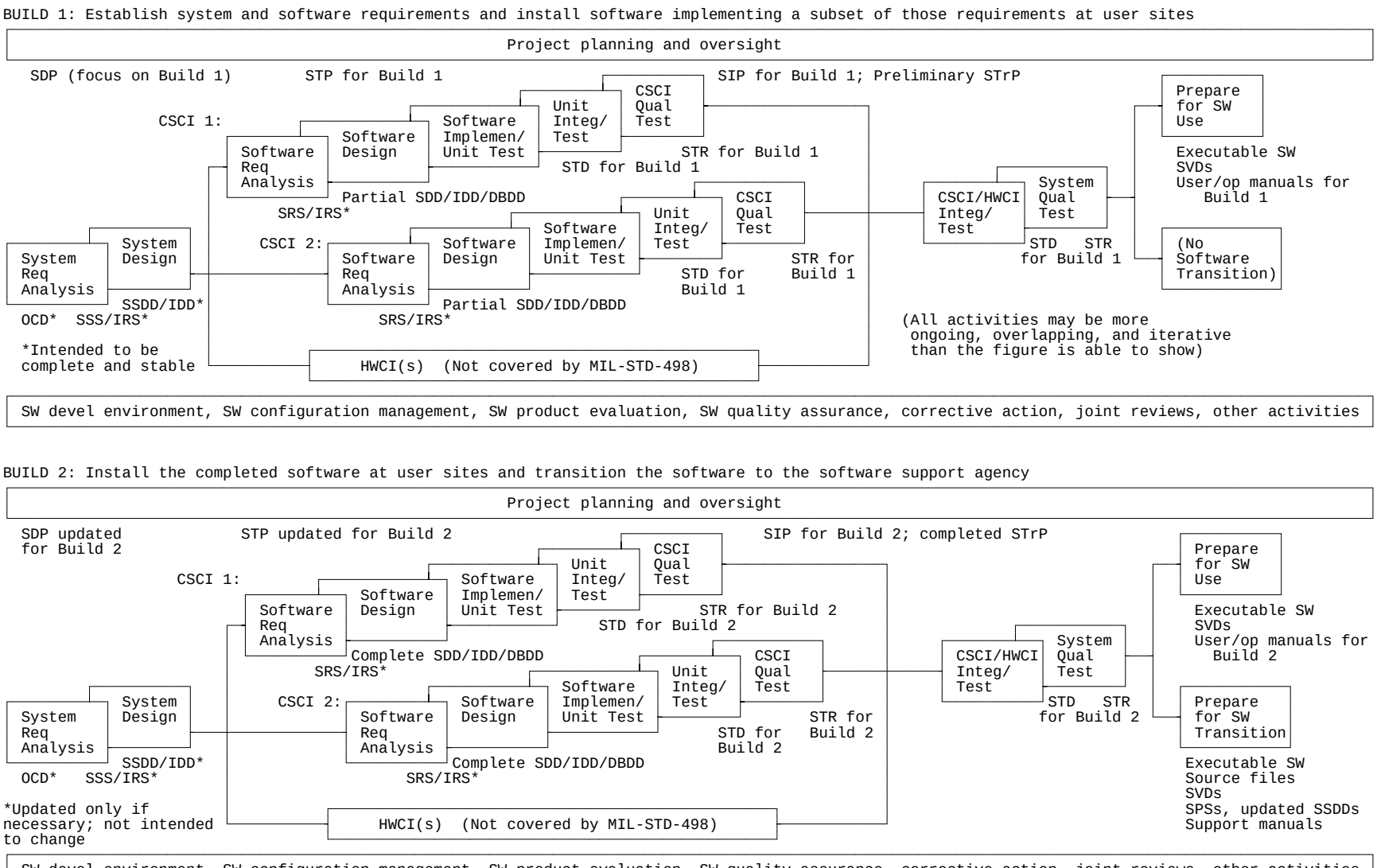


FIGURE 10. One possible way of applying MIL-STD-498 to the Incremental program strategy.

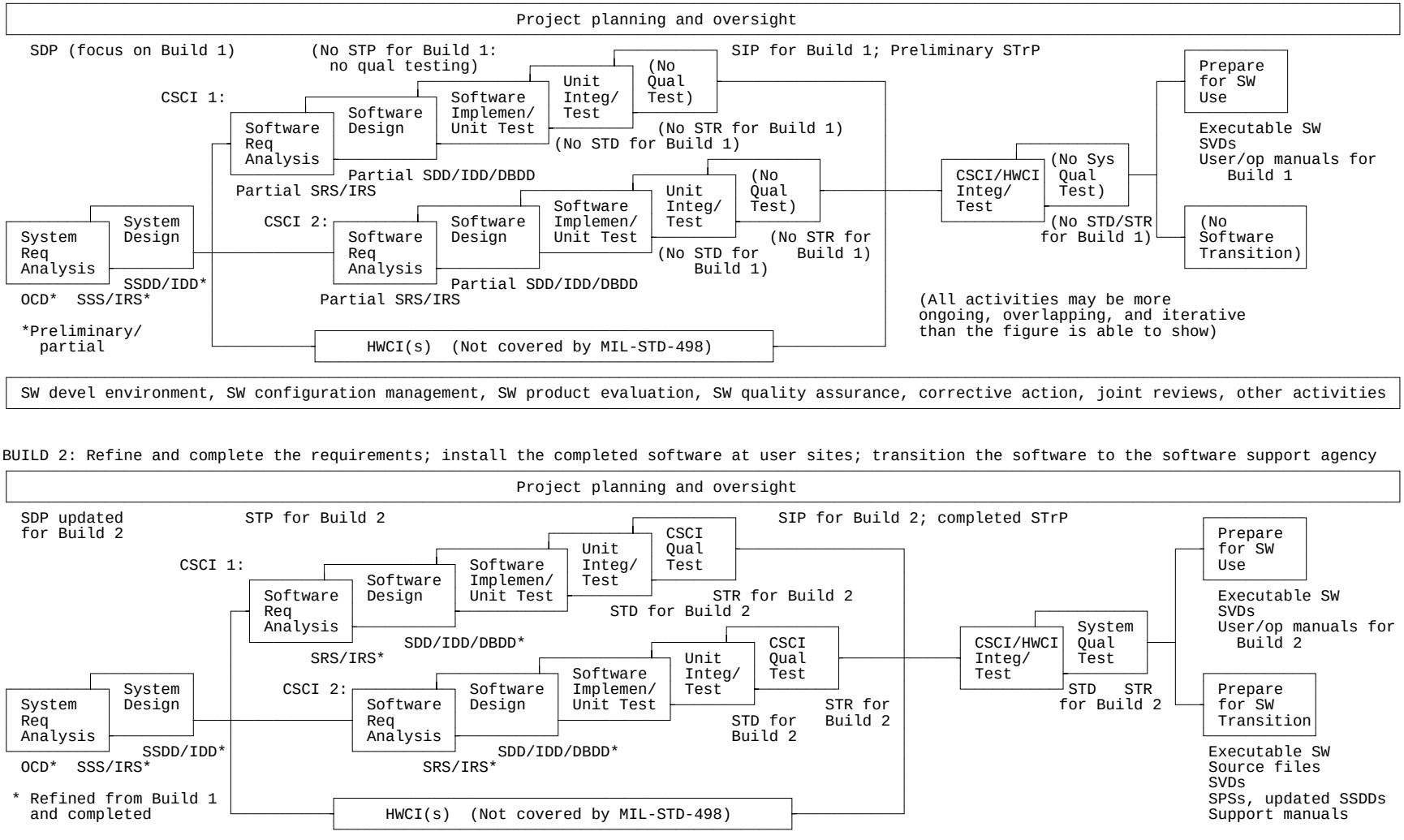


FIGURE 11. One possible way of applying MIL-STD-498 to the Evolutionary program strategy.

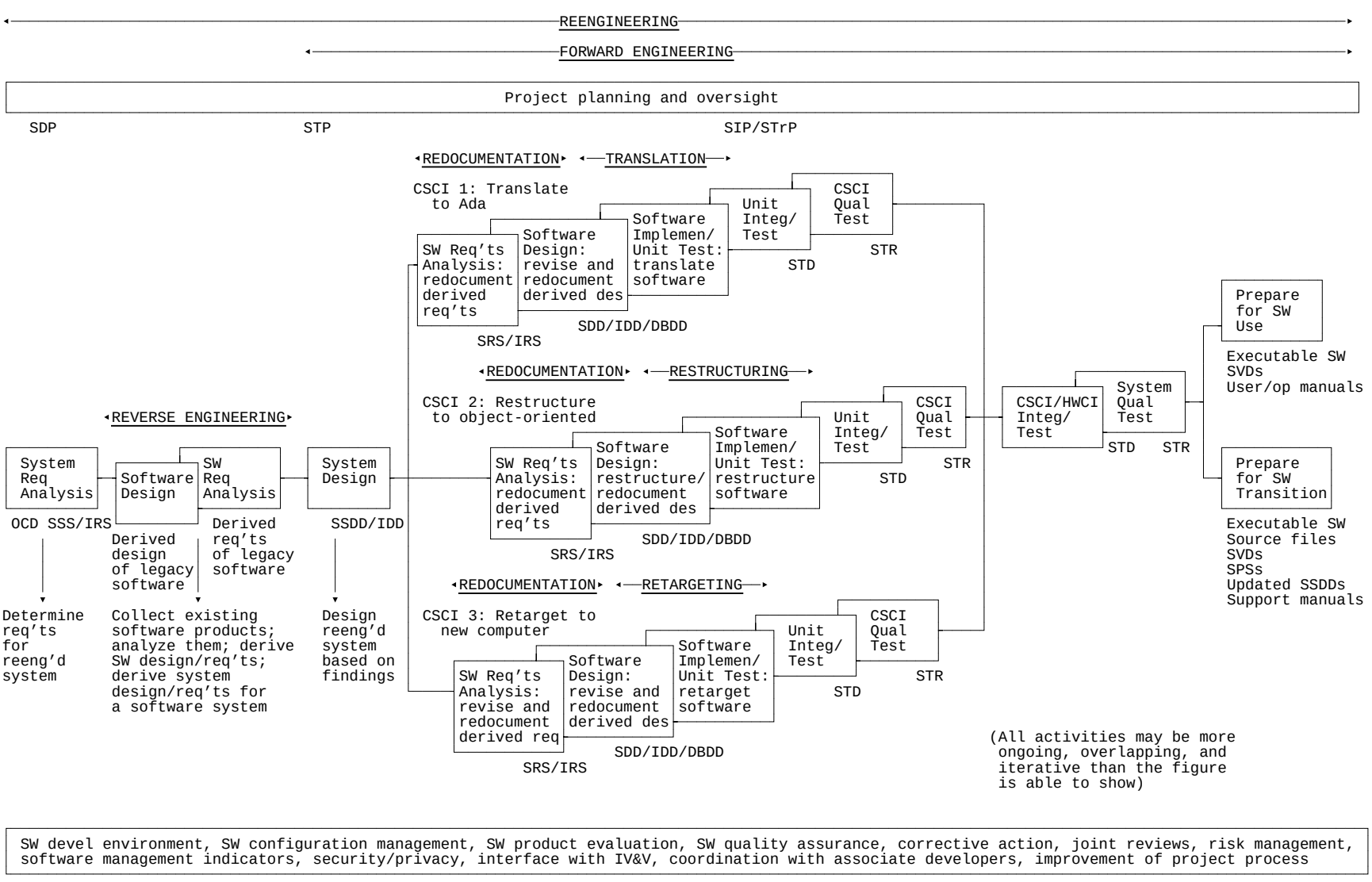


FIGURE 12. One possible way of applying MIL-STD-498 to a reengineering project.

BUILD PLANNING WORKSHEET FOR MIL-STD-498		Build			
		1	2	3	4
1. Identify at right the objectives of each build 2. Indicate below which activities are to be accomplished during the development of each build. Add clarifying notes as needed.		Deliver to selected users an operational prototype that meets the following system-level requirements: SSS-1, SSS-5, ... SSS-1250	Deliver to selected users an operational prototype that meets the requirements of Build 1 plus: SSS-2, SSS-3, SSS-15, ..., SSS-1249	Deliver to all users a tested system that meets the requirements of Builds 1 and 2 plus: SSS-4, SSS-7, SSS-10, ..., SSS-1248	Deliver to all users a tested system that meets all system-level requirements; transition to designated support agency
Para	Activity				
5.1	PROJECT PLANNING AND OVERSIGHT				
5.1.1	Plan the software development effort	Yes: Plan Build 1 in detail; Builds 2-4 in general	Yes: Plan Build 2 in detail; Builds 3-4 in general	Yes: Plan Build 3 in detail; Build 4 in general	Yes: Plan Build 4 in detail
5.1.2	Plan for CSCI qualification testing	No: No CSCI qual testing in this build	No: No CSCI qual testing in this build	Yes: Plan for CSCI qual testing in this build	Yes: Update for CSCI qual testing in this build
5.1.3	Plan for system qualification testing	No: No system qual testing in this build	No: No system qual testing in this build	Yes: Plan for system qual testing in this build	Yes: Update for system qual testing in this build
5.1.4	Plan for installing software at user sites	No: Let users install on their own	No: Let users install on their own	Yes: Plan to install at user sites	Yes: Update as needed for installation of Bld 4
5.1.5	Plan for transitioning software to the support agency	Yes: Very preliminary planning only	Yes: Update preliminary plans	Yes: Update preliminary plans	Yes: Finalize transition planning
5.1.6	Follow plans; perform management review	Yes: For those plans that are in effect	Yes: For those plans that are in effect	Yes: For those plans that are in effect	Yes: For those plans that are in effect
5.2	ESTABLISHING A SOFTWARE DEVEL ENVIRONMENT				
5.2.1	Establish a software engineering environment	Yes: As needed for Build 1	Yes: Update as needed for Build 2	Yes: Update as needed for Build 3	Yes: Update as needed for Build 4
5.2.2	Establish a software test environment	Yes: As needed for Build 1 testing	Yes: As needed for Build 2 testing	Yes: Set up fully for Build 3 qualification testing	Yes: Update as needed for Build 4 qualification testing

FIGURE 13. Example of build planning for a MIL-STD-498 project.

a. Different decisions will apply to different types of software on a project. These differences can be shown within the entries of one worksheet or by using different worksheets for different types of software on a project.

b. If early builds are devoted to experimentation, developing "throw-away" software to arrive at a system concept or system requirements, it may be appropriate to forgo certain formalities, such as coding standards, that will be imposed later on the "real" software. If the early software will be used later, such formalities may be appropriate from the start. These decisions are project-dependent.

G.6.3 Recording tailoring decisions. Tailoring decisions made by the acquirer before the project begins are specified in the Statement of Work. Tailoring proposed by the developer may be communicated via feedback on draft solicitations, proposals written in response to solicitations, the software development plan, joint reviews during the project, or by other means of communication. Refinements to the tailoring decisions may be ongoing as the project proceeds. Those involving contractual changes should be handled accordingly.

G.6.4 Scheduling the selected activities in each build. Another important step in build planning is scheduling the activities in each build. As with tailoring, the acquirer may set forth general milestones and have the developer provide specifics or may provide specific schedules. The following guidelines apply:

a. A common mistake is to treat all CSCIs as though they must be developed in "lock-step," reaching key milestones at the same time. Allowing CSCIs to be on different schedules can result in more optimum development.

b. A similar mistake is to treat software units as though they must be developed in "lock-step," all designed by a certain date, implemented by a certain date, etc. Flexibility in the scheduling of software units can also be effective.

c. The activities in MIL-STD-498 need not be performed sequentially. Several may be taking place at one time, and an activity may be performed continually or intermittently throughout a build or over multiple builds. The activities in each build should be laid out in the manner that best suits the work to be done.

APPENDIX H

GUIDANCE ON ORDERING DELIVERABLES

H.1 Scope. This appendix provides guidance to the acquirer on the deliverables to be required on a software development project. This appendix is not a mandatory part of this standard. The information provided is intended for guidance only.

H.2 Applicable documents. This section is not applicable to this appendix.

H.3 Ordering deliverables. MIL-STD-498 has been worded to differentiate between the planning/engineering activities that make up a software development project and the generation of deliverables. A key objective of this wording is to eliminate the notion that the acquirer must order a given deliverable in order to have planning or engineering work take place. Under MIL-STD-498, the planning and engineering work takes place regardless of which deliverables are ordered, unless a given activity is tailored out of the standard. In addition, joint technical reviews have been included to review the results of that work in its natural form, without the generation of deliverables. Deliverables should be ordered only when there is a genuine need to have planning or engineering information transformed into a deliverable, recognizing that this transformation requires time and effort that would otherwise be spent on the engineering effort. Block 3 of each DID provides information helpful in deciding whether the corresponding deliverable should be ordered.

H.4 Scheduling deliverables. MIL-STD-498 has been structured to support a variety of program strategies and to provide the developer flexibility in laying out a software development process that will best suit the work to be done. All of this flexibility can be canceled by rigid scheduling of deliverables on the CDRL. If the CDRL lays out a strict "waterfall" sequence of deliverables, little room is left to propose innovative development processes. If the CDRL forces all CSCIs into lock-step with each other, little room is left to develop the CSCIs in an optimum order. To the maximum extent possible, the CDRL should avoid such pre-determination, leaving the door open for incremental delivery of software products, staggered development of CSCIs, and other variations to optimize the software development effort. The developer’s software development plan will lay out a proposed schedule that meets the constraints in the CDRL. Final agreement on scheduling can take place at that time.

H.5 Format of deliverables. Traditional deliverables take the form of paper documents exactly following DID formats. While this form works well for some deliverables, it is not the only form, and alternatives should be considered. One variation from paper documents is word processing files containing those documents. This format saves paper, but still requires the developer to format the information as required by the DID. Another variation is specifying that a paper or word processor document is to include all DID contents but may be in the developer’s format. Yet another variation is allowing deliverables to take forms that are not traditional documents at all, such as data in computer-aided software engineering (CASE) tools. These variations in required format can be specified on the CDRL, minimizing the time spent transforming actual work products into deliverables.

H.6 Tailoring the DIDs. Tailoring the DIDs consists of deleting requirements for unneeded information and making other changes that do not increase the required workload, such as combining two documents under one cover. DID tailoring for deliverables is specified in Block 16 of the CDRL.

APPENDIX I

CONVERSION GUIDE FROM DOD-STD-2167A AND DOD-STD-7935A

I.1 Scope. This appendix provides a conversion guide from DOD-STD-2167A and DOD-STD-7935A, the two standards that were merged to form MIL-STD-498. It maps key terms from each of these standards to their counterparts in MIL-STD-498 and shows the relationship of the DIDs required by these standards to their counterparts in MIL-STD-498. This appendix is not a mandatory part of the standard. The information provided is intended for guidance only.

I.2 Applicable documents. This appendix references the following standards, both of which are superseded by this standard:

a. DOD-STD-2167A, Defense System Software Development, 29 February 1988

b. DOD-STD-7935A, DoD Automated Information System Documentation Standards, 31 October 1988

I.3 Mapping of key terms. Figure 14 identifies selected terms in MIL-STD-498 and states their counterparts in DOD-STD-2167A and DOD-STD-7935A.

MIL-STD-498 Term	DOD-STD-2167A Counterpart	DOD-STD-7935A Counterpart
Acquirer	Contracting agency	User Group (no distinction made between acquirer and user roles)
Developer	Contractor (covers Government agency or contractor)	Development Group (covers Government agency or contractor)
Implementation	Coding	Development, production, coding, database generation
Installation (at user sites)	Deployment	Deployment, implementation, installation
Software unit	Computer software component and computer software unit	Software unit
Computer Software Configuration Item (CSCI)	Computer Software Configuration Item (CSCI)	Program, computer program
Software support	Software support	Software maintenance
Software system (consisting of software and possibly computers)	System (No specific term used to distinguish this type of system)	Automated Information System, system
Hardware-software system (where the hardware may be other than computers)	System (No specific term used to distinguish this type of system)	(This type of system not covered by DOD-STD-7935A)

FIGURE 14. Mapping of key terms.

I.4 Mapping of DIDs. Figure 15 identifies the DOD-STD-7935A DIDs and tells which MIL-STD-498 DIDs contain their contents. Figure 16 provides a similar mapping from the DOD-STD-2167A DIDs to the MIL-STD-498 DIDs. Figure 17 provides the reverse mapping, identifying the MIL-STD-498 DIDs and telling which DOD-STD-2167A and/or DOD-STD-7935A DIDs formed the basis for each.

DOD-STD-7935A DID	Incorporated into These MIL-STD-498 DIDs
Functional Description (FD)	System concepts into Operational Concept Description (OCD) System requirements into System/Subsystem Specification (SSS) Development planning into Software Development Plan (SDP)
System/Subsystem Specification (SS)	System requirements into System/Subsystem Specification (SSS) System design into System/Subsystem Design Description (SSDD)
Software Unit Specification (US)	Requirement information into Software Requirements Specification (SRS) and Interface Requirements Specification (IRS) Design information into Software Design Description (SDD) and Interface Design Description (IDD)
Database Specification (DS)	Database Design Description (DBDD)
Test Plan (PT)	High-level planning into Software Test Plan (STP) Detailed planning into Software Test Description (STD)
Test Analysis Report (RT)	Software Test Report (STR)
Users Manual (UM)	Software Input/Output Manual (SIOM)
End User Manual (EM)	Software User Manual (SUM)
Computer Operation Manual (OM)	Software Center Operator Manual (SCOM)
Maintenance Manual (MM)	Planning information into Software Transition Plan (STrP) Software description into Software Design Description (SDD) Maintenance procedures into Software Product Specification (SPS)
Implementation Procedures (IP)	Software Installation Plan (SIP)

FIGURE 15. Mapping of DOD-STD-7935A DIDs to MIL-STD-498 DIDs.

DOD-STD-2167A DID	Incorporated into These MIL-STD-498 DIDs
Software Development Plan (SDP)	Software Development Plan (SDP)
System/Segment Specification (SSS)	System/Subsystem Specification (SSS)
System/Segment Design Document (SSDD)	Operational concept into Operational Concept Description (OCD) System design into System/Subsystem Design Description (SSDD)
Software Requirements Specification (SRS)	Software Requirements Specification (SRS)
Interface Requirements Specification (IRS)	Interface Requirements Specification (IRS)
Software Design Document (SDD)	Software Design Description (SDD)
Interface Design Document (IDD)	Interface Design Description (IDD)
Software Test Plan (STP)	Software Test Plan (STP)
Software Test Description (STD)	Software Test Description (STD)
Software Test Report (STR)	Software Test Report (STR)
Computer System Operator's Manual (CSOM)	Computer Operation Manual (COM)
Software User's Manual (SUM)	Software User Manual (SUM)
Computer Resources Integrated Support Document (CRISD)	Planning information into Software Transition Plan (STrP) Modification procedures into Software Product Specification (SPS)
Software Product Specification (SPS)	Software Product Specification (SPS)
Software Programmer's Manual (SPM)	Computer Programming Manual (CPM)
Firmware Support Manual (FSM)	Firmware Support Manual (FSM)
Version Description Document (VDD)	Software Version Description (SVD)

FIGURE 16. Mapping of DOD-STD-2167A DIDs to MIL-STD-498 DIDs.

MIL-STD-498 DID	DOD-STD-2167A and DOD-STD-7935A Source DIDs
Software Development Plan (SDP)	2167A Software Development Plan (SDP) 7935A Functional Description (FD), section 7
Software Installation Plan (SIP)	7935A Implementation Procedures (IP)
Software Transition Plan (STrP)	2167A Comp Res Integ Sup Doc (CRISD) - planning info 7935A Maintenance Manual (MM) - planning info
Operational Concept Description (OCD)	2167A System/Segment Design Doc (SSDD), section 3 7935A Functional Description (FD), section 2
System/Subsystem Specification (SSS)	2167A System/Segment Specification (SSS) 7935A Functional Description (FD) - system req't info 7935A System/Subsystem Spec (SS) - system req't info
System/Subsystem Design Description (SSDD)	2167A System/Segment Design Document (SSDD) 7935A System/Subsystem Spec - system design info
Software Requirements Specification (SRS)	2167A Software Requirements Specification (SRS) 7935A Software Unit Specification (US) - req't info
Interface Requirements Specification (IRS)	2167A Interface Requirements Specification (IRS) 7935A SW Unit Specification (US) - interface req't info
Software Design Description (SDD)	2167A Software Design Document (SDD) 7935A Software Unit Specification (US) - design info 7935A Maintenance Manual (MM) - "as built" design info
Interface Design Description (IDD)	2167A Interface Design Document (IDD) 7935A SW Unit Specification (US) - interface design info
Database Design Description (DBDD)	7935A Database Specification (DS)
Software Test Plan (STP)	2167A Software Test Plan (STP) 7935A Test Plan (PT) - high-level information
Software Test Description (STD)	2167A Software Test Description (STD) 7935A Test Plan (PT) - detailed information
Software Test Report (STR)	2167A Software Test Report (STR) 7935A Test Analysis Report (RT)
Software Product Specification (SPS)	2167A Software Product Specification (SPS) 2167A CRISD - modification procedures 7935A MM - maintenance procedures
Software Version Description (SVD)	2167A Version Description Document (VDD)
Software User Manual (SUM)	2167A Software User's Manual (SUM) 7935A End User Manual (EM)
Software Center Operator Manual (SCOM)	7935A Computer Operation Manual (OM)
Software Input/Output Manual (SIOM)	7935A Users Manual (UM)
Computer Operation Manual (COM)	2167A Computer System Operator's Manual (CSOM)
Computer Programming Manual (CPM)	2167A Software Programmer's Manual (SPM)
Firmware Support Manual (FSM)	2167A Firmware Support Manual (FSM)

FIGURE 17. Mapping of MIL-STD-498 DIDs to DOD-STD-2167A and DOD-STD-7935A DIDs.

INDEX

This index covers both MIL-STD-498 and its DIDs. Paragraphs in the DIDs are indicated by the DID acronym followed by paragraph numbers; an overall DID is indicated by the DID acronym alone; paragraphs and figures in the standard have no preceding acronym. DID references that begin with "10.1" refer to paragraphs in Block 10, section 10.1 of the DID (General Instructions). All other DID references that do not cite a Block number refer to paragraphs in Block 10, section 10.2 of the DID (Content Requirements). Entries in **bold** indicate primary sources of information about a topic. The entry "et al" indicates that a topic (such as "software") appears too frequently to cite all references.

Custodians:
Army - SC
Navy - EC
Air Force - 10
DISA - DC
DLA - DH

Preparing Activity:
Navy - EC

(Project IPSC-0230)

Review Activities:
OSD - SO, IR, NT
Army - AR, CR, MI, AV
Navy - AS, SH, SA, TD, OM, MC
Air Force - 02, 06, 11, 13, 17, 19
DMA - MP
DNA - DS
NSA - NS

Civil Agency coordinating activities:
NASA - NA
DOT - FAA, USCG
COM - NIST
CIA

Other DoD activities:
DSMC
SEI

Agencies other than US Government:
DND Canada
DoD Germany
MoD UK
US Industry - CODSIA

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

INSTRUCTIONS

1. **The preparing activity must complete blocks 1, 2, 3, and 8. In block 1, both the document number and revision letter should be given.**
2. **The submitter of this form must complete blocks 4, 5, 6, and 7.**
3. **The preparing activity must provide a reply within 30 days from receipt of the form.**

NOTE: This form may not be used to request copies of documents, nor to request waivers, or clarification of 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.

I RECOMMEND A CHANGE:	**1. DOCUMENT NUMBER** MIL-STD-498	**2. DOCUMENT DATE (*YYMMDD*)** 941205

3. DOCUMENT TITLE

Software Development and Documentation

4. NATURE OF CHANGE *(Identify paragraph number and include proposed rewrite, if possible. Attach extra sheets as needed.)*

5. REASON FOR RECOMMENDATION

6. SUBMITTER

a. NAME *(Last, First, Middle Initial)*	**b. ORGANIZATION**	
c. ADDRESS *(Include Zip Code)*	**d. TELEPHONE** *(Include Area Code)* **(1) Commercial** **(2) AUTOVON** *(if applicable)*	**7. DATED SUBMITTED** *(YYMMDD)*

8. PREPARING ACTIVITY

a. NAME	**b. TELEPHONE** *(Include Area Code)*
Space & Naval Warfare Systems Command	**(1) Commercial** **(703)602-4491** **(2) AUTOVON** **332-4491**
c. ADDRESS *(Include Zip Code)* **SPAWAR 10-12** **2451 Crystal Drive (CPK-5)** **Arlington, VA 22245-5200**	**IF YOU DO NOT RECEIVE A REPLY WITHIN 45 DAYS, CONTACT:** Defense Quality and Standardization Office 5203 Leesburg Pike, Suite 1403, Falls Church, VA 22041-3466 Telephone (703)756-2340 AUTOVON 289-2340