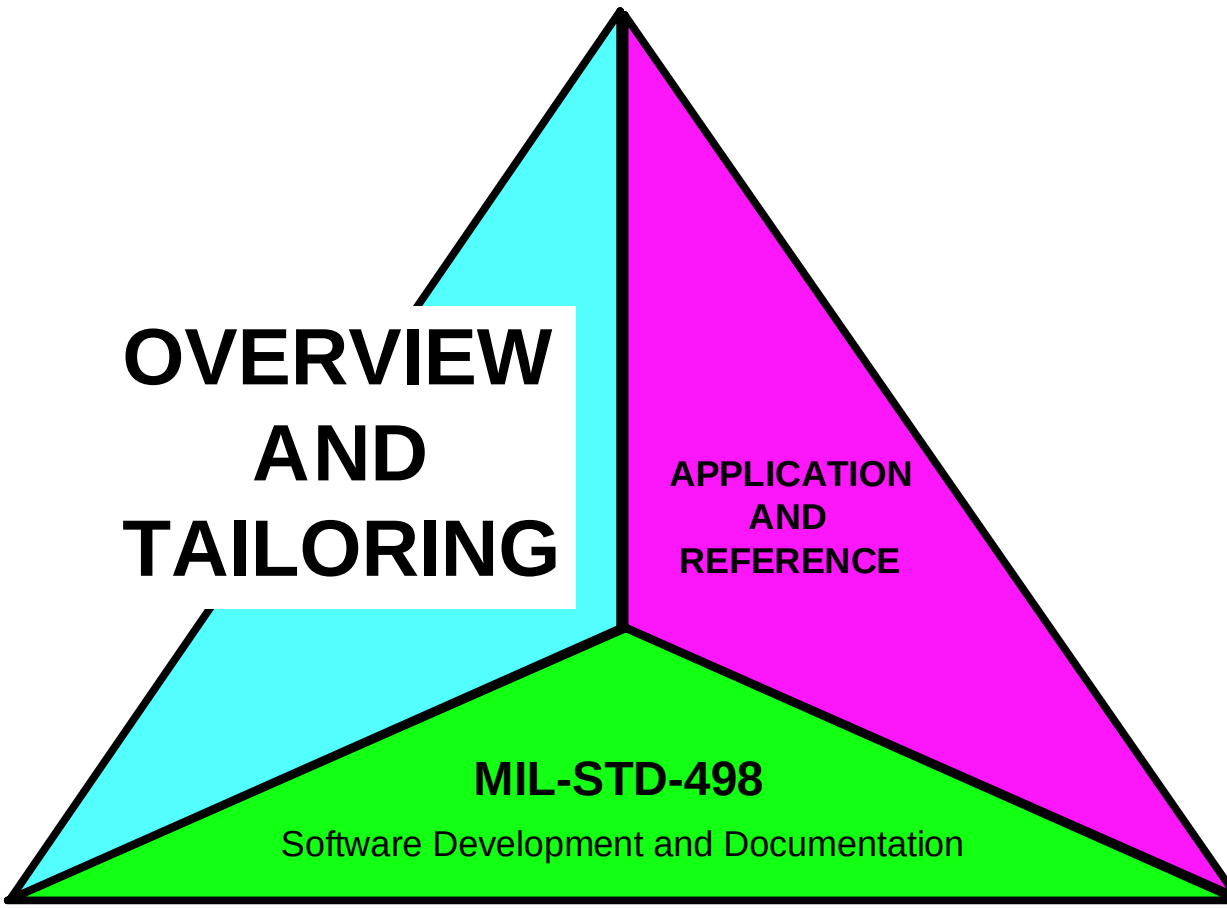
MIL-STD-498

Overview and Tailoring

GUIDEBOOK



31 JANUARY 1996

Joint Logistics Commanders
Joint Policy Coordinating Group
on Computer Resources Management

This page is intentionally blank.

CONTENTS

APPENDIXES

FIGURES

FOREWORD

1. This guidebook is available for use by all departments and agencies of the Department of Defense (DoD) or any other government agency.

2. MIL-STD-498, Software Development and Documentation, identifies a set of software development activities and defines the software products to be generated by those activities. In order to use the standard effectively, all parties involved in a software development effort need to understand the principles that underlie the standard. This guidebook is intended to help the acquirer understand and apply MIL-STD-498, but, it can be useful to developers and other parties.

3. This guidebook is the first of two guidebooks prepared by DoD for use with MIL-STD-498. The two guidebooks are:

1) MIL-STD-498 Overview and Tailoring Guidebook. The first guidebook provides the following:

- Objectives of MIL-STD-498
- Overview of the standard
- Key concepts of the standard
- Conversion guide
- Tailoring and application guidance

2) MIL-STD-498 Application and Reference Guidebook. The second guidebook provides the following:

- Topic-by-topic application of the standard including:
  - A summary of MIL-STD-498's requirements
  - Acquirer responsibilities and considerations
  - Things to think about
- Tables and indexes between the guidebooks and MIL-STD-498's requirements, activities, and Data Item Descriptions (DIDs).

4. Beneficial comments (recommendations, additions, deletions) and any pertinent data that may be of use in improving this document should be addressed to SPAWAR 10-12, 2451 Crystal Drive (CPK 5), Arlington, VA 22245-5200.

EXECUTIVE OVERVIEW

MIL-STD-498 is a standard for the software development process. It is applicable throughout the system life cycle. It establishes uniform requirements for acquiring, developing, modifying, and documenting software. It defines standard terminology and establishes activities, tasks, and products for a software development or maintenance project. It can be applied to any type of software, including application software, operating system software, the software portion of firmware, reusable software, and software employed to develop deliverable software.

The activities in the standard form a comprehensive set, sufficient for a large, complex project, but scalable and adaptable to suit the needs of small ones. The standard is meant to be tailored as needed for each project by specifying in the contract which provisions of the standard apply.

The standard is intended to be responsive to the rapidly evolving software discipline. It is independent of any particular methodology, ensuring the acquirer's right to specify the product and the developer's right to be technologically creative and innovative. In particular:

- The activities and tasks in the standard tell what to do, not how to do it. For example, the standard requires the developer to perform architectural design, but does not require use of the object-oriented design method.

- The standard does not specify or encourage any software life cycle model (Waterfall, Incremental, Evolutionary, Spiral, etc.). Instead, the standard provides the building blocks needed to carry out the life cycle model selected for a software project.

- The standard does not specify or depend on any design or programming language. It is meant to be applicable regardless of the language used.

- The standard provides flexibility regarding documentation:

    - A distinction is made between the task of generating and recording planning or engineering information, a task that is intrinsic to the development of software, and the task of generating a deliverable containing that information.

    - Information and deliverables can be in hardcopy form (using contractor or DID format), in electronic form, or in computer-aided software engineering (CASE) tools.

- The standard does not emphasize any particular software quality factor, such as reliability, maintainability, or reusability. This choice is left to each project.

- The standard can be used within an organization or contractually between two parties.

- The standard provides a performance-based distinction between requirements for a product and design of a product. A requirement is a characteristic that a system or software item is required to possess to be acceptable to the acquirer, while design is a characteristic that the acquirer is willing to leave up to the developer. This emphasis on required performance can allow a developer to provide quality products at reduced costs when unique product solutions are not required.

MIL-STD-498 specifically removes key problems found in the use of predecessor standards. MIL-STD-498:

a. Is usable with any development strategy. MIL-STD-498 has been structured to better accommodate development models other than the traditional "Waterfall" model; to avoid time-oriented dependencies and implications; to provide alternatives to formal reviews (that can force a Waterfall development model); and to explain how to apply the standard across multiple builds or iterations.

b. Is usable with any development methods. To improve compatibility with object-oriented and other methods not using functional decomposition, MIL-STD-498 allows the developer to define the design and implementation structures, and to use that structure in the documentation.

c. Is compatible with CASE tools. MIL-STD-498 recognizes that much of the significant work on a software development project does not involve writing documents. The standard: (1) separates planning and engineering activities from preparation of deliverables to make it easier to call for one without the other; (2) provides language that permits the recording of project information in forms other than traditional documents, such as CASE tools; (3) acknowledges data in CASE tools as a substitute for traditional documents; and (4) provides guidance to prevent unnecessary deliverables.

d. Provides requirements for software reuse. MIL-STD-498 recognizes that many projects incorporate or are based on reusable software. To provide clearer requirements when this is the case, the standard: (1) identifies criteria for use in evaluating reusable software, and (2) provides guidance on interpreting MIL-STD-498's activities and deliverables when applied to reusable software.

e. Supports the use of software measurement. MIL-STD-498 supports the use of software measurement. The standard requires identification and application of software management indicators and includes an annex of candidate indicators that might be used.

f. Emphasizes software supportability. MIL-STD-498 has enhanced software supportability requirements. Examples are requirements to: (1) record the rationale for key decisions made on the project; (2) identify all resources the maintenance organization will need to maintain the software; and (3) demonstrate that those resources are sufficient to maintain the software.

g. Provides a role for software in the system. MIL-STD-498 recognizes that software exists in the context of a system. It provides a role for the software developer both in the case of a hardware-software system and a software-only system.

PARTICIPANTS

The MIL-STD-498 Overview and Tailoring Guidebook was sponsored by the Joint Logistics Commanders (JLC) Joint Policy Coordinating Group on Computer Resource Management (JPCG-CRM), and was developed by Logicon under the direction of the MIL-STD-498 Harmonization Working Group (HWG). At the time this guidebook was completed, the HWG had the following membership:

Dr. Raghu Singh, SPAWAR, HWG Chairman
Norma Stopyra, SPAWAR, MIL-STD-498 Guidebook Product Manager

Paul Anderson, SPAWAR
Larry Baker, DSMC
Perry DeWeese, CODSIA
C. "Jay" Ferguson (NSA)
Robert Gagnon, OASD (ES)
Susan Gardner, FAA
Joseph Gianuzzi, SEI
Robert Hegland, USAISSC
Alan Huguley, DMA
Tim Janes, MoD, UK
Capt. Jonathan Liles, HQ AFMC
Donna McCloud, HQ DLA
LTC. Dixie McNeme, USA
Fred Moxley, DISA
Glenn Plonk, NSA
Linda Sheets, AFMC
Mary Synder, USN

Guidebook Developers:

Myrna Olson
Stuart Campbell
Dorothy Kuckuck
Jane Radatz

1. SCOPE

1.1 Purpose. The purpose of this guidebook is to:

a. Describe the objectives of MIL-STD-498
b. Provide an overview of the standard and explain its key concepts
c. Provide a conversion guide from DOD-STD-2167A and DOD-STD-7935A
d. Provide guidance on applying MIL-STD-498

1.2 Application of the guidebook. This guidebook is intended to be applied as follows.

1.2.1 Intended audience. This guidebook is addressed to the acquirer, that is, to an organization that procures software products for itself or for another organization. It can also be used by developers and other parties, but its focus is on helping the acquirer understand and apply MIL-STD-498.

1.2.2 Use on contracts. This guidebook is not intended to be included in procurements or contracts as a contractually binding document. It is meant to provide guidance only.

1.3 Organization of the guidebook. This guidebook is organized into five major sections.

a. Section 1 defines the purpose, application, and organization of the guidebook.
b. Section 2 provides information about documents mentioned in the guidebook.
c. Section 3 defines key terms used in the guidebook.
d. Section 4 describes the objectives of MIL-STD-498, provides an overview of the standard, explains its key concepts, and provides a conversion guide from DOD-STD-2167A and DOD-STD-7935A.
e. Section 5 provides guidance on applying MIL-STD-498.

2. DOCUMENTS MENTIONED IN THIS GUIDEBOOK

This section identifies documents mentioned in sections 3, 4 and 5 of this guidebook.

2.1 Government documents.

2.1.1 Specifications, standards, and handbooks. The following specifications, standards, and handbooks are mentioned in this guidebook. Unless otherwise specified, the issues of these documents are those listed in the Department of Defense Index of Specifications and Standards (DoDISS) and supplements thereto.

SPECIFICATIONS: None

STANDARDS, MILITARY

MIL-STD-498	-	Software Development and Documentation
DOD-STD-1703*	-	Software Product Standards
DOD-STD-2167A*	-	Defense System Software Development
DOD-STD-7935A*	-	DoD Automated Information Systems (AIS) Documentation Standards
DOD-STD-2168**	-	Defense System Software Quality Program

* Superseded by MIL-STD-498

** Discussed in this guidebook to aid in the transition to MIL-STD-498

HANDBOOKS, MILITARY: None

Unless otherwise indicated, copies of federal and military specifications, standards, and handbooks are available from the Defense Printing Service Detachment Office, (ATTN: Customer Service), 700 Robbins Avenue, Bldg. 4D, Philadelphia, PA 19111-5094 or by sending a request by fax to 215/697-1462.

2.1.2 Other government documents, drawings, and publications.

DoDI 5000.2	-	Defense Acquisition Management Policies and Procedures
DoDI 8120.2	-	Automated Information System Life-Cycle Management Process, Review, and Milestone Approval
DFARS 227.401	-	Defense FAR Supplement

2.2 Non-government publications. None.

3. DEFINITIONS

The definitions below are the same as those in MIL-STD-498. They are provided here to aid in using this guidebook. A list of acronyms used in this guidebook can be found in Appendix A.

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 preexisting 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.

4. OBJECTIVES, OVERVIEW, KEY CONCEPTS, AND CONVERSION

This section describes the objectives of MIL-STD-498, provides an overview of the standard, explains its key concepts, and provides a conversion guide from DOD-STD-2167A and DOD-STD-7935A.

4.1 Objectives of MIL-STD-498. MIL-STD-498 was developed with three primary objectives. These objectives are illustrated in Figure 1 and discussed in the following paragraphs.

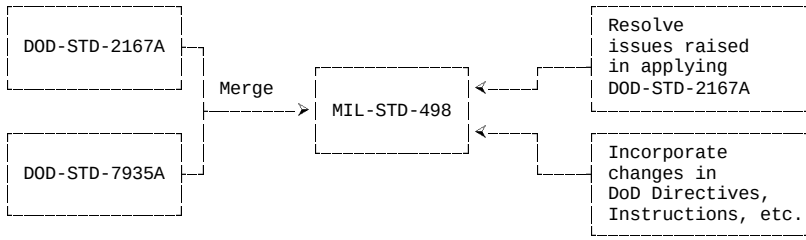


FIGURE 1. Objectives of MIL-STD-498.

4.1.1 Merge DOD-STD-2167A and DOD-STD-7935A.

a. Former standards. Before the development of MIL-STD-498, there were two primary software development and documentation standards in the DoD:

1) DOD-STD-2167A, Defense System Software Development. This standard was used for mission critical systems. It defined a software development process and had 16 Data Item Descriptions (DIDs) defining documentation.

2) DOD-STD-7935A, DoD Automated Information Systems (AIS) Documentation Standards. This standard was used for automated information systems. It defined the format and content of 11 documents and provided guidance for applying these documents.

b. Reasons to merge. As more systems began to incorporate both mission critical computer resources (MCCR) and automated information system (AIS) software and as DoD worked to streamline the acquisition processes for MCCR and AIS systems, the distinction between MCCR and AIS began to blur. For this reason, the DoD decided to merge the two standards into one that could be used for any type of software.

c. Relationship to DOD-STD-1703. One of the participants in the MIL-STD-498 project was the National Security Agency (NSA). During the project, NSA decided to adopt MIL-STD-498 in place of its software development standard DOD-STD-1703 (NS), Software Product Standards. As a result, DOD-STD-1703, though not merged with DOD-STD-2167A and DOD-STD-7935A, was superseded along with them.

4.1.2 Resolve issues identified in applying DOD-STD-2167A.

a. Issues raised. During the time that DOD-STD-2167A was in use, a number of issues were identified regarding its use. Figure 2 presents the primary issues.

ISSUES IDENTIFIED IN APPLYING DOD-STD-2167A

1) Improve compatibility with Incremental and Evolutionary development models
2) Decrease dependence on formal reviews and audits
3) Decrease emphasis on preparing documentation
4) Improve compatibility with computer-aided software engineering (CASE) tools
5) Improve links to systems engineering
6) Include the use of software management indicators
7) Improve coverage of modification, reuse, and reengineering
8) Increase emphasis on software supportability
9) Provide a clearer distinction between requirements and design
10) Improve compatibility with non-hierarchical design methods
11) Improve coverage of database development
12) Improve the criteria used for software product evaluations
13) Eliminate confusion between software quality assurance and software product evaluation
14) Extend configuration management concepts to in-process work products
15) Clarify relationships to other standards
16) Provide a means to order software via the Contract Data Requirements List (CDRL)
17) Eliminate inconsistencies and holes in the DIDs

FIGURE 2. Issues identified in applying DOD-STD-2167A.

b. Resolving the issues. These issues, gathered from conferences, workshops, articles, papers, and other sources, provided the second objective for the MIL-STD-498 effort: resolve these issues and make MIL-STD-498 compatible with current software development practices.

4.1.3 Ensure compatibility with current DoD instructions, standards, and handbooks.

a. Changing environment. Between the time that DOD-STD-2167A and DOD-STD-7935A were issued and the time that MIL-STD-498 was begun, significant changes were made in DoD directives, instructions, standards, and handbooks affecting software development. Key documents developed or undergoing change at that time were DoDI 5000.2 and DoDI 8120.2 as well as standards and handbooks on systems engineering, configuration management, and software support.

b. Ensuring compatibility. This changing environment for software development resulted in the third objective for MIL-STD-498: ensure that the merged standard would be compatible with the changes being made to these documents.

4.2 Overview of MIL-STD-498. The MIL-STD-498 package consists of the standard and 22 Data Item Descriptions (DIDs), as illustrated in Figure 3. The paragraphs that follow provide an overview of the standard and its DIDs.

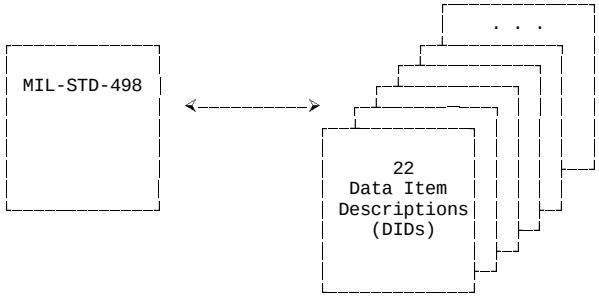


FIGURE 3. The MIL-STD-498 package.

4.2.1 Organization of MIL-STD-498. Figure 4 shows the organization of MIL-STD-498. Its requirements are presented in Sections 4 and 5 of the standard. The remainder of the standard provides details for selected requirements and associated material such as definitions and application guidance.

ORGANIZATION OF MIL-STD-498

Section 1: Scope (including purpose and applicability)
Section 2: Referenced Documents
Section 3: Definitions
Section 4: General Requirements
Section 5: Detailed Requirements
Appendixes
- Appendix A: Acronyms
- Appendix B: Interpreting MIL-STD-498 for Incorporation of Reusable Software Products
- Appendix C: Category and Priority Classifications for Problem Reporting
- Appendix D: Software Product Evaluations
- Appendix E: Candidate Joint Management Reviews
- Appendix F: Candidate Management Indicators
- Appendix G: Guidance on Program Strategies, Tailoring, and Build Planning
- Appendix H: Guidance on Ordering Deliverables
- Appendix I: Conversion Guide from DOD-ST--2167A and DOD-STD-7935A

Index to the standard and DIDs

FIGURE 4. Organization of MIL-STD-498.

4.2.2 Overview of MIL-STD-498's general requirements. Figure 5 provides an overview of MIL-STD-498's general requirements, presented in Section 4 of the standard. These requirements apply across all detailed requirements presented in Section 5 of the standard.

OVERVIEW OF MIL-STD-498 GENERAL REQUIREMENTS

4.1 Establish a software development process consistent with contract requirements

4.2 General requirements for software development

- Use systematic, documented methods
- Develop and apply standards for representing requirements, design, code, and test information
- Reuse software products as appropriate:
  - Evaluate reusable software products for use in fulfilling contract requirements; incorporate those that meet the criteria in the Software Development Plan
  - Identify opportunities for developing software products for reuse; notify the acquirer of those that have cost benefits and are consistent with program objectives
- Establish and apply strategies for handling critical requirements, such as those with safety, security, or privacy implications
- Analyze and fulfill the computer hardware resource utilization requirements (such as memory reserves)
- Record rationale for key decisions for use by the support agency
- Provide the acquirer access to developer and subcontractor facilities

FIGURE 5. Overview of MIL-STD-498's general requirements.

4.2.3 Overview of MIL-STD-498's detailed requirements. Figure 6 provides an overview of MIL-STD-498's detailed requirements, presented in Section 5 of the standard. These requirements are organized into nineteen major software development activities.

OVERVIEW OF MIL-STD-498 DETAILED REQUIREMENTS

5.1 Project planning and oversight
- Plan the software development effort
- Plan for computer software configuration item (CSCI) qualification testing
- Participate in planning for system testing
- Plan for installing the software at user sites
- Plan for transitioning the software to the support agency
- Follow approved plans; conduct management reviews; get approval for updates

5.2 Establishing a software development environment
- Establish, control, and maintain:
  - A software engineering environment
  - A software test environment
  - A software development library
  - Software development files
- Use non-deliverable software only if operation/support of the deliverable software do not depend on it or if the acquirer has or can obtain the same software

5.3 System requirements analysis
- Analyze user input provided by the acquirer
- Participate in defining and recording the system operational concept
- Participate in defining and recording system requirements

5.4 System design
- Participate in defining and recording the system-wide design decisions
- Participate in defining and recording the system architectural design

5.5 Software requirements analysis
- Define and record the software requirements to be met by each CSCI

5.6 Software design
- Define and record CSCI-wide design decisions
- Define and record the architectural design of each CSCI
- Define and record the detailed design of each CSCI

FIGURE 6. Overview of MIL-STD-498's detailed requirements.

OVERVIEW OF MIL-STD-498 DETAILED REQUIREMENTS - *(continued)*

5.7 Software implementation and unit testing
- Develop and record software corresponding to each software unit
- Prepare test cases, procedures, and data for unit testing
- Perform unit testing
- Revise and retest as needed
- Analyze and record test results

5.8 Unit integration and testing
- Prepare test cases, procedures, and data for unit integration and testing
- Perform unit integration and testing
- Revise and retest as needed
- Analyze and record test results

5.9 CSCI qualification testing
- Assign responsibility to persons who did not perform detailed design or implementation
- Include testing on the target computer system or an approved alternative
- Prepare test cases, procedures, and data for CSCI qualification testing
- Dry run the test cases if testing is to be witnessed by the acquirer
- Perform CSCI qualification testing
- Revise and retest as needed
- Analyze and record test results

5.10 CSCI/HWCI integration and testing
- Prepare test cases, procedures, and data for CSCI/HWCI integration and testing
- Perform CSCI/HWCI integration and testing
- Revise and retest as needed
- Analyze and record test results

5.11 System testing
- Assign responsibility to persons who did not perform detailed design or implementation
- Include testing on the target computer system or an approved alternative
- Participate in developing and recording test cases, procedures, data for system testing
- Participate in dry run of the test cases if testing is to be witnessed by the acquirer
- Participate in performing system qualification testing
- Participate in revising and retesting as needed
- Participate in analyzing and recording test results

FIGURE 6. Overview of MIL-STD-498 detailed requirements - *(continued)*.

OVERVIEW OF MIL-STD-498 DETAILED REQUIREMENTS - *(continued)*

5.12 Preparing for software use
- Prepare the executable software for user sites
- Identify and record the exact version of software to be sent to each site
- Prepare information needed for:
  - Hands-on use of the software
  - Submitting (batch) inputs and interpreting outputs
  - Operating the software in software centers or networked environments
  - Operating the computers
- Install at user sites; provide training and other assistance as required

5.13 Preparing for software transition
- Prepare the executable software for the support site
- Prepare the source files for the support site
- Identify and record the exact version of software to be sent to the support site
- Update the CSCI design descriptions and prepare other information needed for support
- Update the system design descriptions
- Prepare information needed to:
  - Program the host and target computers
  - Program/reprogram the firmware devices
- Install the software at the support site; demonstrate that it can be regenerated from source files; provide training and other assistance as required

5.14 Software configuration management
- Identify all entities to be controlled during development: computer files, documents, other software products, and elements of the software environments
- Establish and implement procedures for controlling each entity
- Prepare/maintain records of the status of all entities under project-level or higher control
- Support acquirer-conducted configuration audits as required by the contract
- Establish and implement procedures for packaging, storage, handling, and delivery

5.15 Software product evaluation
- Perform in-process and final evaluations of software products
- Prepare records of the evaluations
- Assign responsibility to persons other than those who developed the software product

FIGURE 6. Overview of MIL-STD-498 detailed requirements - *(continued)*.

OVERVIEW OF MIL-STD-498 DETAILED REQUIREMENTS - *(continued)*

5.16 Software quality assurance
- Perform on-going evaluations to assure that:
  - Activities required by the contract or described in the software development plan (SDP) are being performed in accordance with the contract and SDP
  - Required software products exist and have undergone software product evaluation, testing, and corrective action as required by the standard and contract
- Prepare records of the evaluations
- Assign responsibility to persons other than those who developed the software product, performed the activity, or are responsible for the product or activity

5.17 Corrective action
- Prepare problem/change reports for problems found in:
  - Software products under project-level or higher control
  - Activities required by the contract or described in the software development plan
- Implement a corrective action system for handling these problems

5.18 Joint technical and management reviews
- Plan and participate in joint (acquirer/developer) technical reviews
- Plan and participate in joint management reviews

5.19 Other activities
- Identify project risks; develop/implement strategies to manage them
- Identify and apply software management indicators
- Comply with the security and privacy requirements in the contract
- Include in subcontracts all requirements necessary to ensure that software products are developed in accordance with the prime contract
- Interface with IV&V agents as specified in the contract
- Coordinate with associate developers, working groups, and interface groups as specified in the contract
- Periodically assess the processes used on the project; identify necessary improvements and propose them in updates to the SDP; implement if approved

FIGURE 6. Overview of MIL-STD-498's detailed requirements - *(continued)*.

4.2.4 Overview of the MIL-STD-498 DIDs. MIL-STD-498 provides 22 DIDs from which an appropriate set can be selected for a project. Figure 7 identifies the DIDs. Figure 8 provides an overview of each DID and the resulting software product. Section 4.4 shows the relationship of the MIL-STD-498 DIDs to those for DOD-STD-2167A and DOD-STD-7935A. Section 6.2 of MIL-STD-498 provides a cross-reference to associated paragraphs of the standard.

MIL-STD-498 DID	Resulting Software Product
Software Development Plan (SDP)	A plan for performing the software development
Software Test Plan (STP)	A plan for conducting qualification testing
Software Installation Plan (SIP)	A plan for installing the software at user sites
Software Transition Plan (STrP)	A plan for transitioning to the support agency
Operational Concept Description (OCD)	The operational concept for the system
System/Subsystem Specification (SSS)	The requirements to be met by the system
Software Requirements Specification (SRS)	The requirements to be met by a CSCI
Interface Requirements Specification (IRS)	The requirements for one or more interfaces
System/Subsystem Design Description (SSDD)	The design of the system
Software Design Description (SDD)	The design of a CSCI
Interface Design Description (IDD)	The design of one or more interfaces
Database Design Description (DBDD)	The design of a database
Software Test Description (STD)	Test cases/procedures for qualification testing
Software Test Report (STR)	Test results of qualification testing
Software Product Specification (SPS)	The executable software, the source files, and information to be used for support
Software Version Description (SVD)	A list of delivered files and related information
Software User Manual (SUM)	Instructions for hands-on users of software
Software Input/Output Manual (SIOM)	Instructions for users depending on SW operators
Software Center Operator Manual (SCOM)	Instructions for SW operators supporting users
Computer Operation Manual (COM)	Instructions for operating a computer
Computer Programming Manual (CPM)	Instructions for programming a computer
Firmware Support Manual (FSM)	Instructions for programming firmware devices

FIGURE 7. The MIL-STD-498 DIDs.

MIL-STD-498 DID*	Overview of the Resulting Software Product
Computer Operation Manual (COM) DI-IPSC-81446	Provides information needed to operate a given computer and its peripheral equipment. Focuses on the computer itself, not on particular software that will run on the computer. Intended for newly developed computers, special-purpose computers, or other computers for which commercial or other operation manuals are not readily available.
Computer Programming Manual (CPM) DI-IPSC-81447	Provides information needed by a programmer to understand how to program a given computer. Focuses on the computer itself, not on particular software that will run on the computer. Intended for newly developed computers, special-purpose computers, or other computers for which commercial or other programming manuals are not readily available.
Database Design Description (DBDD) DI-IPSC-81437	Describes the design of a database, that is, 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 (DBMS). Can also describe the software units used to access or manipulate the data. Used as the basis for implementing the database and related software units. Provides the acquirer visibility into the design and provides information needed for software support.
Firmware Support Manual (FSM) DI-IPSC-81448	Provides the information needed to program and reprogram the firmware devices of a system. Applies to read only memories (ROMs), Programmable ROMs (PROMs), Erasable PROMs (EPROMs), and other firmware devices. Describes the firmware devices and the equipment, software, and procedures needed to erase firmware devices, load software into the firmware devices, verify the load process, and mark the loaded firmware devices.
Interface Design Description (IDD) DI-IPSC-81436	Describes the interface characteristics of one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components. May describe any number of interfaces. Can be used to supplement the System/Subsystem Design Description (SSDD), Software Design Description (SDD), and Database Design Description (DBDD). With its companion Interface Requirements Specification (IRS), serves to communicate and control interface design decisions.
Interface Requirements Specification (IRS) DI-IPSC-81434	Specifies the requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components to achieve one or more interfaces among these entities. Can cover any number of interfaces. Can be used to supplement the System/Subsystem Specification (SSS) and Software Requirements Specification (SRS) as the basis for design and qualification testing of systems and CSCIs.
Operational Concept Description (OCD) DI-IPSC-81430	Describes a proposed system in terms of the user needs it will fulfill, its relationship to existing systems or procedures, and the ways it will be used. Used to obtain consensus among the acquirer, developer, support, and user agencies on the operational concept of a proposed system. Depending on its use, an OCD may focus on communicating the user's needs to the developer or the developer's ideas to the user and other interested parties.

* The DIDs are presented in alphabetical order by acronym for ease of reference.

FIGURE 8. Overview of MIL-STD-498 DIDs and the resulting software products.

MIL-STD-498 DID	Overview of the Resulting Software Product
Software Center Operator Manual (SCOM) DI-IPSC-81444	Provides personnel in a computer center or other centralized or networked software installation information on how to install and operate a software system. Developed for software systems that will be installed in a computer center or other centralized or networked software installation, with users accessing the system via terminals or personal computers or submitting and receiving inputs and outputs in batch or interactive mode. Often used with the Software Input/Output Manual (SIOM). This pair of manuals is an alternative to the Software User Manual (SUM).
Software Design Description (SDD) DI-IPSC-81435	Describes the design of a Computer Software Configuration Item (CSCI). Describes the CSCI-wide design decisions, the CSCI architectural design, and the detailed design needed to implement the software. The SDD may be supplemented by Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs). With its associated IDDs and DBDDs, is used as the basis for implementing the software. Provides the acquirer visibility into the design and provides information needed for software support.
Software Development Plan (SDP) DI-IPSC-81427	Describes a developer's plans for conducting a software development effort. Covers new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products. Provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources. Portions of the plan may be bound separately if this approach enhances their usability. Examples include plans for software configuration management and software quality assurance.
Software Input/Output Manual (SIOM) DI-IPSC-81445	Tells a user how to access, submit inputs to, and interpret output from, a batch or interactive software system that is run by personnel in a computer center or other centralized or networked software installation. Developed for software systems that will be installed in a computer center or other centralized or networked software installation, with users accessing the system via terminals or personal computers or submitting and receiving inputs and outputs in batch mode. Often used with the Software Center Operator Manual (SCOM). This pair of manuals is an alternative to the Software User Manual (SUM).
Software Installation Plan (SIP) DI-IPSC-81428	Describes plans for installing software at user sites, including preparations, user training, and conversion from existing systems. Prepared when the developer will be involved in the installation of software at user sites and the installation process will be sufficiently complex to require a documented plan. For software embedded in a hardware-software system, a fielding or deployment plan for the hardware-software system may make a separate SIP unnecessary.
Software Product Specification (SPS) DI-IPSC-81441	Provides the executable software, source files, and software support information, including the "as built" design description and the compilation, build, and modification procedures, for a Computer Software Configuration Item (CSCI). Can be used to order the executable software and/or source files for a CSCI and is the primary software support document for the CSCI.
Software Requirements Specification (SRS) DI-IPSC-81433	Specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs) referenced from the SRS. The SRS, possibly supplemented by IRSs, is used as the basis for design and qualification testing of a CSCI.

FIGURE 8. Overview of MIL-STD-498 DIDs and the resulting software products - *(continued)*.

MIL-STD-498 DID	Overview of the Resulting Software Product
System/Subsystem Design Description (SSDD) DI-IPSC-81432	Describes the system- or subsystem-wide design and the architectural design of a system or subsystem. May be supplemented by Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs). With its associated IDDs and DBDDs, is used as the basis for further system development. May be prepared as a System Design Description or Subsystem Design Description (each using the acronym SSDD).
System/Subsystem Specification (SSS) DI-IPSC-81431	Specifies the requirements for a system or subsystem and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the system or subsystem's external interfaces may be presented in the SSS or in one or more Interface Requirements Specifications (IRSs) referenced from the SSS. Possibly supplemented by IRSs, used as the basis for design and qualification testing of a system or subsystem. May be prepared as a System Specification or Subsystem Specification (each using the acronym SSS).
Software Test Description (STD) DI-IPSC-81439	Describes the test preparations, test cases, and test procedures to be used to perform qualification testing of a Computer Software Configuration Item (CSCI) or a software system or subsystem. Enables the acquirer to assess the adequacy of the qualification testing to be performed.
Software Test Plan (STP) DI-IPSC-81438	Describes plans for qualification testing of Computer Software Configuration Items (CSCIs) and software systems. Describes the software test environment to be used for the testing, identifies the tests to be performed, and provides schedules for test activities. There is usually a single STP for a project. Enables the acquirer to assess the adequacy of planning for CSCI and, if applicable, software system qualification testing.
Software Test Report (STR) DI-IPSC-81440	Provides a record of the qualification testing performed on a Computer Software Configuration Item (CSCI), a software system or subsystem, or other software-related item. Enables the acquirer to assess the testing and its results.
Software Transition Plan (STrP) DI-IPSC-81429	Identifies the hardware, software, and other resources needed for life cycle support of deliverable software and describes the developer's plans for transitioning deliverable items to the support agency. Developed if the software support concept calls for transition of responsibility from the developer to a separate support agency. May also be used by the acquirer for updating the acquirer's Computer Resources Life Cycle Management Plan.
Software User Manual (SUM) DI-IPSC-81443	Tells a hands-on software user how to install and use a Computer Software Configuration Item (CSCI), a group of related CSCIs, or a software system or subsystem. May also cover a particular aspect of software operation, such as instructions for a particular position or task. Developed for software that is run by the user and has a user interface requiring on-line user input or interpretation of displayed output. If the software is embedded in a hardware-software system, user manuals or operating procedures for that system may make separate SUMs unnecessary. An alternative to the Software Center Operator Manual (SCOM) and Software Input/Output Manual (SIOM).
Software Version Description (SVD) DI-IPSC-81442	Identifies and describes a software version consisting of one or more Computer Software Configuration Items (CSCIs). Used to release, track, and control software versions. The term "version" may be applied to the initial release of the software, to a subsequent release of that software, or to one of multiple forms of the software released at approximately the same time (for example, to different sites).

FIGURE 8. Overview of MIL-STD-498 DIDs and the resulting software products - *(continued)*.

4.3 Key concepts of MIL-STD-498. To apply MIL-STD-498 effectively, it is important to understand the key concepts embodied in the standard. The paragraphs that follow present these key concepts.

4.3.1 Players and agreements assumed by MIL-STD-498. Figure 9 illustrates the players and agreements assumed by MIL-STD-498. The paragraphs that follow explain each term.

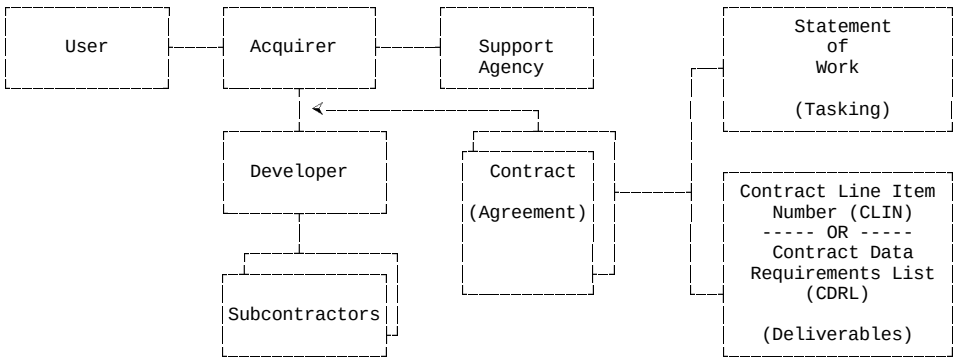


FIGURE 9. Players and agreements assumed by MIL-STD-498.

a. Players. MIL-STD-498 is written in terms of the following players.

1) Acquirer: an organization that procures software products for itself or another organization. The acquirer may be a government agency or a prime contractor.

2) Developer: an organization that develops software products ("develops" may include new development, modification, reuse, reengineering, or any other activity that results in software products). The developer may be a contractor or a government agency.

3) Subcontractor(s): organization(s) identified and tasked by the developer to perform part of the required effort. Subcontractors may be contractors or government agencies.

4) User: the organization(s) or persons within those organization(s) who will operate and/or use the software for its intended purpose.

5) Support agency: the organizations that will maintain the software and offer additional services (such as mainframe operation) to the user after the software is installed for operational use.

b. Agreements. MIL-STD-498 is written in terms of the following agreements:

1) Contract: the agreement between the acquirer and developer. Between government agencies, the contract may take the form of a Memorandum of Agreement or other form of tasking.

2) Statement of Work (SOW): that portion of the contract that lays out the tasks to be performed by the developer.

3) Contract Line Item Number (CLIN): that portion of the contract that identifies deliverable products.

4) Contract Data Requirements List (CDRL): that portion of the contract that identifies deliverable software products.

c. Project-specific interpretation. Depending upon circumstances, two or more of the players identified above may be the same; the players or agreements may have different names; or there may be other variations from the framework just described. It is important to sort out how these terms apply to each individual situation so the standard can be interpreted appropriately.

4.3.2 Accommodating Incremental and Evolutionary development.

a. Program strategies. Both DoDI 5000.2 and DoDI 8120.2 define multiple program strategies for acquiring systems and software. Included are "Grand Design," "Incremental" (also called "Preplanned Product Improvement"), and "Evolutionary" strategies. A key concept of MIL-STD-498 is accommodating each of these strategies.

b. MIL-STD-498's "build" concept. To meet this goal, MIL-STD-498 is written in terms of developing software in multiple "builds." Each build incorporates a specified subset of the planned capabilities of the software. The builds might be prototypes, versions offering partial functionality, or other partial or complete versions of the software. Figures and notes in the standard tell how to interpret the standard on projects involving multiple builds. Figure 10 shows how each major activity in MIL-STD-498 may be applied in one or more builds. Details about program strategies and build planning are given in Section 5 of this guidebook.

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

4.3.3 Alternatives to formal reviews and audits.

a. Problems with formal reviews and audits. One of the biggest distractions from "real work" on a software development project is preparing for formal reviews and audits. A common "horror story" is that everyone stops "real work" six weeks before a formal review and starts generating view-graphs for the review. The result can be tremendous expenditure of time and energy on a review that is an overly detailed snapshot of where the project was six weeks before. The developer spends hundreds of staff-hours preparing, and the acquirer is swamped by information overload.

b. MIL-STD-498's joint reviews. Instead of formal reviews and audits, MIL-STD-498 calls for more frequent, low-key joint (acquirer/developer) technical and management reviews, focusing on natural work products rather than view-graphs or other specially prepared materials. These reviews include informal discussions of status, ideas, approaches, risks, etc. The idea of these reviews is ongoing communication between the acquirer and developer with minimum time wasted. Appendix F in MIL-STD-498 suggests candidate reviews to hold.

c. Relationship to formal reviews and audits. MIL-STD-498's joint technical and management reviews are designed to provide all of the coordination needed between the developer and the acquirer. Formal reviews may be used to supplement the joint reviews if so desired, but they should focus on major issues raised at the joint reviews rather than repeating the details covered there.

4.3.4 Decreased emphasis on documentation and greater compatibility with CASE tools.

a. Document-driven software development. A frequent complaint about software development under military standards is that it is document driven. This concern is illustrated in Figure 11. Rather than focusing on the real work to be done, the developer must generate an endless series of documents, some of which have little to do with getting the job done. The developer may have to translate real work products, such as data in computer-aided software engineering (CASE) tools, into paper documents or translate work products from the format in which work is being done to an artificial format imposed by Data Item Descriptions (DIDs).

FIGURE 11.
Excessive documentation.

b. MIL-STD-498's emphasis on natural work products. A fundamental concept of MIL-STD-498 is focusing on the "real work" of a project, versus non-productive tasks that do not contribute to project goals. This focus appears in several ways:

1) Focus on natural work products versus documentation. The MIL-STD-498 activities that generate information require the developer to "define and record" information, not to "prepare a given document." Notes in the standard and a paragraph explaining the intended meaning of "record" emphasize that this wording is designed to accommodate project information in its natural form, for example, in CASE tools, rather than requiring the developer to prepare traditional documents.

2) Focus on natural work projects versus deliverables. MIL-STD-498 distinguishes between the planning and engineering work that are the "real work" of a project and the activity of preparing a deliverable. This point may seem the same as item (1) just above, but is not. Item (1) is about allowing the developer to record project information in the most natural form, for example, in a CASE tool versus a paper document. Item (2) is about not making work products, whatever their form, deliverable without good reason. In the past, acquirers often thought that the way to make the developer do the work was to require deliverables containing evidence of the work (the "homework" approach to acquisition management). But preparing information for delivery takes extra time and should not be required without good reason. MIL-STD-498 requires the work to take place, regardless of whether the results are made deliverable, and gives the acquirer access to the results in the developer's facility. When deliverables are needed, they are called for in the CDRL or in contract line item numbers (CLINs).

3) Use of the DIDs as checklists. Many of the activities in MIL-STD-498 rely on DIDs to fully define them. For example, the activity of preparing a Software Development Plan (SDP) consists of defining and recording all information called for by the Software Development Plan DID. This requirement applies regardless of whether the SDP is in the form of a document and regardless of whether the resulting plan is deliverable. The DID is used as a checklist of information to be defined in carrying out the required activity. This is a new role for DIDs. Figure 12 illustrates the role of DIDs and software products in MIL-STD-498, emphasizing the distinction made between natural work products, the form in which they reside, and their status as deliverables or non-deliverables. If the DID is used only as a checklist and the resulting software product is not deliverable, the DID is not cited in the CDRL.

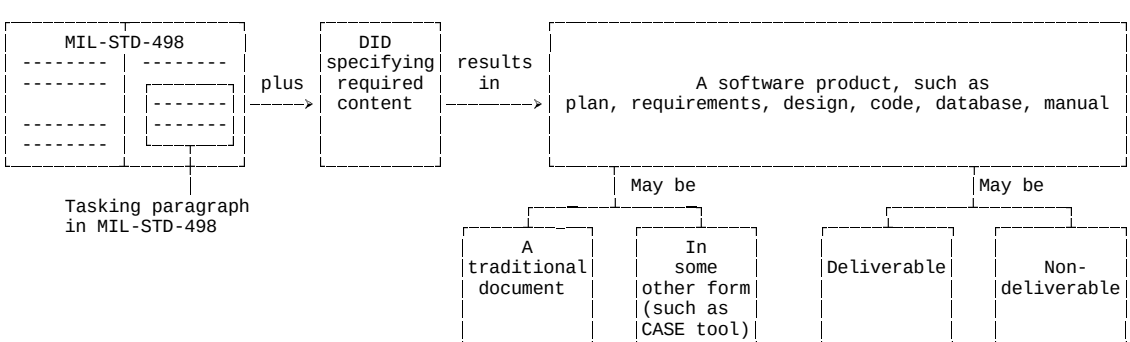


FIGURE 12. Role of DIDs and software products in MIL-STD-498.

4.3.5 Improved links to systems engineering.

a. Two types of systems. A key aspect of the merger of DOD-STD-2167A and DOD-STD-7935A was that the resulting standard had to cover two types of systems, illustrated in Figure 13: (1) hardware-software systems (such as radar systems) for which the standard covers only the software portion, and (2) software systems (such as payroll systems) for which the standard governs overall system development.

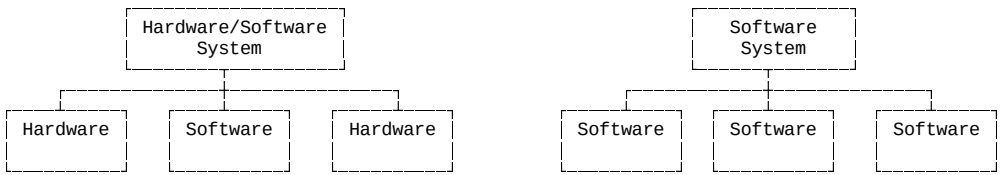


FIGURE 13. Two types of systems handled by MIL-STD-498.

b. MIL-STD-498's "participate" concept. MIL-STD-498 covers this duality by including system-level activities, requiring the developer to "participate" in them, and stating that: 1) If the software to be developed is part of a hardware-software system for which the standard covers only the software portion, the term "participate" is to be interpreted as "take part in, as described in the Software Development Plan," and 2) If the software (possibly with its computers) is considered to constitute a system, the term "participate" is to be interpreted as "be responsible for." This use of "participate" covers both cases.

4.3.6 Use of software management indicators.

a. DoD emphasis on quantitative measures. There is increasing emphasis in DoD on quantitative measurement of software progress, size, quality, and other attributes. Figure 14 illustrates this type of measurement. The idea behind these initiatives is that only by obtaining quantitative measures will it be possible to make real progress in improving the software development process.

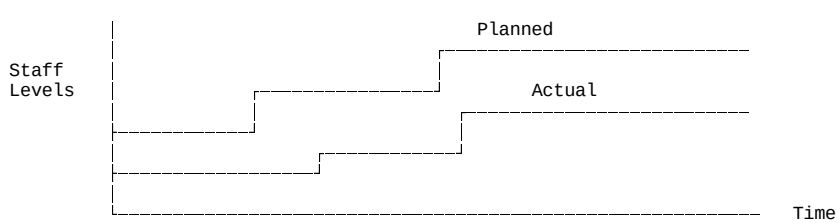


FIGURE 14. Illustration of the use of quantitative measures.

b. MIL-STD-498's software management indicators. MIL-STD-498 takes a cautious step into this area by requiring the developer to define and apply software management indicators and by providing a set of candidate management indicators to serve as a starting point. It is left to the developer to propose in the Software Development Plan the indicators to be used, the data to be collected, the approach to interpreting the data, and the reporting approach. The acquirer can provide feedback on the proposed approach when reviewing the Software Development Plan.

4.3.7 Improved coverage of modification, reuse, and reengineering.

a. DoD initiatives. There is increasing emphasis in DoD on meeting user needs by modifying, reusing, and reengineering existing software products rather than developing everything "from scratch." These initiatives apply not just to the software itself, but to associated software products such as architectures and design.

b. MIL-STD-498's support for these initiatives. Figure 15 summarizes the provisions incorporated into MIL-STD-498 to accommodate these approaches.

PROVISIONS SUPPORTING SOFTWARE MODIFICATION, REUSE, AND REENGINEERING

MIL-STD-498:

1) States that each activity in the standard may be performed by modifying, reusing, or reengineering existing items, as well as by developing something new.

2) Requires the developer to identify and evaluate reusable software products for use in fulfilling contract requirements, and to incorporate those that meet the criteria established for the project.

3) Provides criteria for evaluating software products for reuse and tells how to interpret the standard when applied to reused items.

4) Requires the developer to identify and analyze opportunities for developing software products for reuse, and to notify the acquirer of those that provide cost benefits and are compatible with program objectives.

5) Provides a definition of reengineering and its constituent activities, and provides a diagram showing how MIL-STD-498 can be applied to a reengineering project.

FIGURE 15. MIL-STD-498 provisions supporting modification, reuse, and reengineering.

4.3.8 Increased emphasis on software supportability.

a. Importance of supportability. Study after study has shown that over 75% of software's life cycle cost occurs after initial delivery. These figures emphasize the importance of making software supportable.

b. Effect of the DoD support model. In many DoD organizations, the approach to software support is to transition the software from the initial developer to a completely different organization for software support. This approach, potentially involving a complete change of players, makes supportability of the software even more critical.

c. MIL-STD-498's approach to supportability. Supportability is the driving force behind many of MIL-STD-498's requirements. Figure 16 summarizes key provisions.

PROVISIONS THAT PROMOTE SOFTWARE SUPPORTABILITY

MIL-STD-498 requires the developer to:

1) Identify all resources used or generated during the development that will be needed by the support agency.

2) Prepare a Software Transition Plan that identifies these resources and tells how deliverable items will be transitioned to the support agency.

3) Demonstrate that the delivered software can be supported given these resources.

4) Define and record the behavioral design as well the architectural and detailed designs, detailing the behavior selected to meet system and CSCI requirements.

5) Record the rationale for key decisions that may be useful to the support agency.

6) Update design descriptions after testing to reflect the "as built" software.

7) Record compilation/build procedures for those who must modify the software.

8) Prepare manuals to be used for software support.

9) Record and follow coding, design, and other standards for software products.

FIGURE 16. MIL-STD-498 provisions that promote software supportability.

4.3.9 Clearer distinction between requirements and design.

a. Traditional distinction. The traditional distinction between requirements and design has been that requirements state "what" a system or CSCI must do and design states "how" it will do it. This distinction, however, can be subjective and often leads to disagreements as to which is which.

b. MIL-STD-498's distinction. MIL-STD-498 takes a more pragmatic approach: A characteristic, whether it is "what" or "how," is a requirement if the acquirer cares enough about it to make it a condition for acceptance; a characteristic selected by the developer in response to the requirements, whether it is "what" or "how," is design. The contrast in definitions is illustrated in Figure 17.

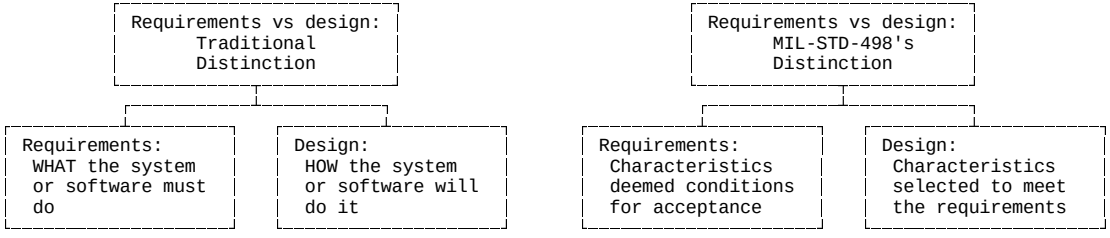


FIGURE 17. Contrasting definitions of requirements and design.

c. Implications. The implications of MIL-STD-498's distinction are the following:

1) More focused SSSs and SRSs. SSSs and SRSs should specify only those characteristics of the system and software that are conditions of acceptance. Characteristics that are left up to the developer, even if they describe "what" the system or software will do, do not belong in the SSSs and SRSs. Instead, they belong in design descriptions as behavioral design (what the system or CSCI will do), or as architectural or detailed design (how the system or CSCI will do it).

2) More focused qualification testing. Since qualification testing consists of demonstrating that all of a CSCI's or system's requirements have been met, more focused SSSs and SRSs have a similar effect on qualification testing. This testing focuses on those characteristics that have been designated conditions for acceptance without getting side-tracked into other characteristics.

3) Keeping design decisions the developer's prerogative. Design decisions made during the project, even if presented at joint reviews or included in deliverable design descriptions, remain at the discretion of the developer. They need to be covered in developer-internal testing, but they are subject to change and need not be demonstrated in qualification testing. If the acquirer decides that a design decision should be a requirement, it is necessary to take appropriate action, possibly involving contractual measures and additional cost to revise specifications, revise planning for qualification tests, and perform additional qualification tests.

4.3.10 Compatibility with non-hierarchical methods.

a. Top-down functional decomposition. Although not intended to do so, DOD-STD-2167A has been perceived to favor top-down functional decomposition over other development methods. Most frequently cited are DOD-STD-2167A's requirements regarding software architecture, in which CSCIs are decomposed into computer software components (CSCs), which are decomposed into other CSCs, which are finally decomposed into computer software units (CSUs), which are associated with physical code entities. While some users of DOD-STD-2167A have had no problem with this framework, others have found it inhibiting to their preferred methods.

b. MIL-STD-498's software architecture. MIL-STD-498 redefines software architecture by requiring simply that CSCIs be decomposed into software units, which may or may not be related to each other in a hierarchical manner. A software unit is any 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. MIL-STD-498 further acknowledges that software units in the design may or may not have a one-to-one relationship with the code and data entities that implement them or with the computer files containing those entities. This generalization of software units and separation of logical from physical entities, illustrated in Figure 18, provides more flexibility to employ a variety of software development methods.

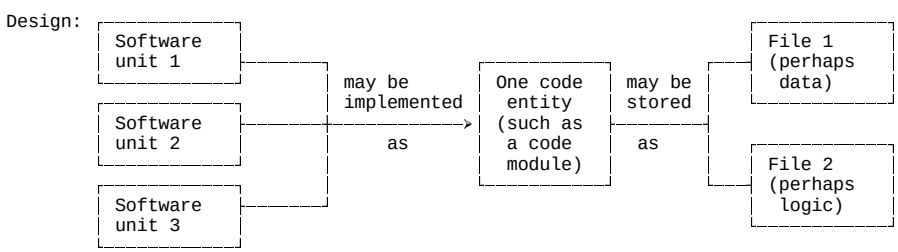


FIGURE 18. Flexibility in handling design entities, code entities, and files.

c. Methodology independence. It is MIL-STD-498's goal to neither encourage nor preclude any particular software development methodology. As shown in Figure 19, the standard lays out a set of activities that must be accomplished and defines a set of software products that must be produced, but leaves it to the developer to propose in the Software Development Plan how those activities will be performed and how the products will be produced. The acquirer has an opportunity to react to the proposed methods when reviewing the Software Development Plan.

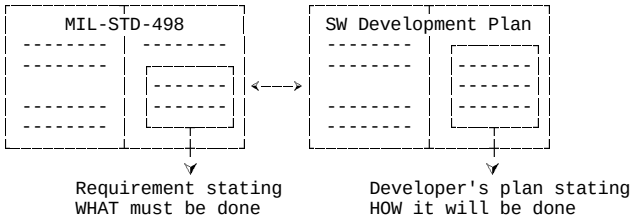


FIGURE 19. Methodology independence of MIL-STD-498.

4.3.11 Improved coverage of database development.

a. Role of databases. Databases are crucial to automated information systems. They may not be used at all in weapon systems. A significant issue that arose in merging DOD-STD-2167A and DOD-STD-7935A was making sure that database development was handled adequately.

b. MIL-STD-498's approach to databases. MIL-STD-498 takes an approach consistent with the Defense Federal Acquisition Regulation Supplement, by defining software as computer programs and computer databases. Figure 20 illustrates this relationship. Following this approach, MIL-STD-498 takes databases into account throughout the software development process. Requirements definition includes defining database requirements; software design includes database design; implementation -- purposely renamed from DOD-STD-2167A's "coding" to accommodate databases -- includes building and possibly populating databases; and testing includes ensuring that databases work as required.

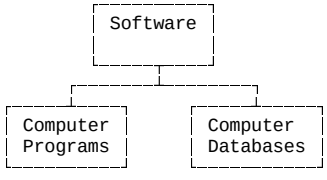


FIGURE 20. Relationship of the terms software, computer program,and computer database.

4.3.12 Improved criteria for software product evaluations.

a. Objective versus subjective criteria. A significant problem in specifying criteria for software product evaluations is that some of the most important criteria are subjective. For example, most people would agree that a requirement specification should be understandable and feasible, but there are no objective definitions for these criteria. Concern about objectivity led to the deletion of all subjective criteria from DOD-STD-2167A. The result was a set of objective but incomplete criteria that was less than satisfactory.

b. MIL-STD-498's criteria. MIL-STD-498 handles this problem by including subjective criteria such as understandability and feasibility, defining them as clearly as possible, acknowledging that they are subjective, and stating that, because of their subjectivity, the requirement is not to prove that a software product meets the criteria but to perform an evaluation using the criteria and to identify possible problems for discussion and resolution. This explicit handling of subjective criteria, illustrated in Figure 21, was accepted by reviewers and provides more meaningful software product evaluations.

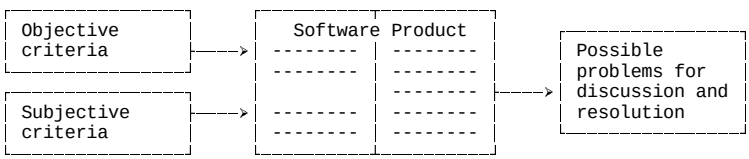


FIGURE 21. MIL-STD-498's use of objective and subjective criteria.

4.3.13 Distinguishing between software quality assurance and software product evaluation.

a. Issues addressed. MIL-STD-498 addresses two related issues:

1) Relationship of software quality assurance and software product evaluation. DOD-STD-2168, Defense System Software Quality Program, establishes requirements for software quality assurance. Although no overlap with DOD-STD-2167A's software product evaluations was intended, there were requirements in the two standards that raised questions about the standards' relationship.

2) Requests to include software quality assurance in MIL-STD-498. Early drafts of MIL-STD-498 followed DOD-STD-2167A's approach of leaving software quality assurance to DOD-STD-2168. Many reviewers commented that the standard was incomplete without a software quality assurance section.

b. MIL-STD-498's approach to software quality assurance. In response to these concerns, MIL-STD-498 incorporates software quality assurance and carefully words the requirements for software quality assurance to avoid any overlap with software product evaluation. Figure 22 summarizes these requirements.

Aspect of the project	Responsibility of quality assurance
Activities required by the contract or described in the Software Development Plan	Assure they are being performed in accordance with the contract and with the plan
Software products required by the contract (whether deliverable or not)	Assure they exist and have undergone software product evaluation, testing, and corrective action as required by the contract

FIGURE 22. Avoiding overlap of SQA with other evaluation/testing activities.

c. Relationship to DOD-STD-2168. MIL-STD-498's quality assurance requirements can be used on their own. If more detail is desired, they can be replaced or supplemented by DOD-STD-2168 or another software quality assurance standard. Appendix B of this guidebook shows how MIL-STD-498 covers the requirements in DOD-STD-2168.

4.3.14 Application of configuration management to in-process work products.

a. Software configuration management in DOD-STD-2167A. DOD-STD-2167A's configuration management requirements focused on controlling deliverable code and documentation. Feedback on the standard indicated that DOD-STD-2167A's concept of a "developmental configuration" was confusing and unhelpful; that all work products, not just deliverable ones, need to be controlled during development; and that the standard ignores items that are actually controlled (such as computer files).

b. MIL-STD-498's approach to configuration management. MIL-STD-498 eliminates the concept of the developmental configuration and requires the developer to: 1) identify all of the entities to be controlled (such as computer files, documents, CSCIs, etc.), 2) establish the levels of control each entity must pass through (from author control up to acquirer control), and 3) establish procedures for controlling and tracking these entities and changes to them. This approach focuses on in-process work products as well as deliverables and provides the levels of control needed during software development.

c. Relationship to configuration management standards. MIL-STD-498's requirements for software configuration management can be used on their own to provide configuration management during software development. If more detail is desired, they can be replaced or supplemented by a separate configuration management standard.

4.3.15 Relationship to other related standards.

a. Tiering of standards. Tiering of standards, that is, having one standard invoke other standards, is discouraged by the DoD Acquisition Streamlining initiative as leading to excessive, sometimes unintended requirements. The problem is illustrated in Figure 23.

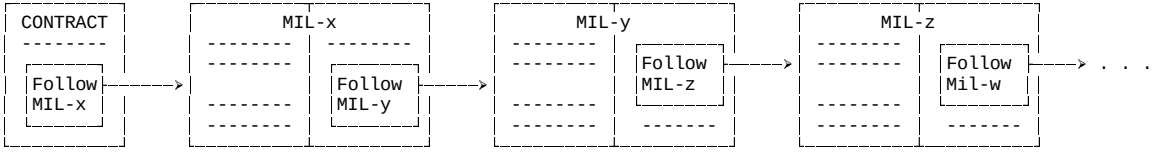


FIGURE 23. Tiering of standards.

b. MIL-STD-498's approach. MIL-STD-498 invokes no other standards. It takes the approach of covering all relevant topics at least in summary form, and provides, for information only, a table of related standards that the acquirer may wish to consider as alternatives or supplements to the requirements in MIL-STD-498. Included in the table are both DoD and commercial standards. Because the listed standards are from various sources and are not coordinated, a set selected from the table may be inconsistent. The acquirer is responsible for reconciling any inconsistencies in the selected standards.

4.3.16 Ordering the software via the CDRL (DD Form 1423).

a. Differing methods. Different DoD services and government agencies disagree on how to order delivery of the software itself. Some order delivery of the software via the CDRL (DD Form 1423), using a DID; others order delivery of the software as a contract line item number (CLIN). As a result of the controversy, DOD-STD-2167A was issued without a DID for ordering software. This decision forced DoD services and government agencies who wished to order software via the CDRL to find or prepare a DID for this purpose.

b. The MIL-STD-498 Software Product Specification. Research into this question led to Defense Federal Acquisition Regulation Supplement clause 227.403-77, which states that, with certain exceptions, any computer program or computer database to be acquired under a contract is to be listed on the CDRL. Based on this research, provisions were included in MIL-STD-498's Software Product Specification DID to permit this DID to be used to order both executable code and source files. MIL-STD-498 thus provides for both methods of ordering software -- via CDRL or CLIN. The reader should follow DoD service or government agency policy or practice in deciding which method to use.

4.3.17 Tailoring.

a. Importance of tailoring. MIL-STD-498 and its DIDs are designed to be tailored for each project and for each type of software being developed on a project. Each project is to review the standard and DIDs and select only those requirements that will be cost-effective for the project.

b. Relationship to "builds." Tailoring goes hand in hand with multiple builds because different requirements of the standard apply to different builds. For example, the requirements for qualification testing may not apply to early prototypes, but do apply to the finished product. Section 5 provides guidance on how to plan the builds for a project and tailor MIL-STD-498 accordingly.

4.3.18 The importance of plans.

a. MIL-STD-498's plans. MIL-STD-498 requires 4 types of plans, described in Figure 24.

Title	Purpose
Software Development Plan (SDP)	Describes the developer's approach to carrying out the software development project. Includes planning for software quality assurance and software configuration management, which can be published separately
Software Test Plan (STP)	Describes the developer's approach to software qualification testing
Software Installation Plan (SIP)	Describes the developer's approach to installing the software at user sites
Software Transition Plan (STrP)	Describes the developer's approach to transitioning the software to the support agency

FIGURE 24. The MIL-STD-498 plans.

b. Role of plans in MIL-STD-498. MIL-STD-498 tells "what" the developer must do, but not "how" it is to be done. The developer proposes the "how" via the plans. This philosophy makes the plans, especially the Software Development Plan, key documents on the project. The acquirer has the right to review and approve (or disapprove) the plans, and the developer must adhere to the plans once approved. To accommodate changes during the project, MIL-STD-498 requires the developer to submit all updates except those involving developer-internal schedules and related staffing information for approval.

4.4 Converting to MIL-STD-498 from DOD-STD-2167A and DOD-STD-7935A. This section is designed to ease the transition from DOD-STD-2167A and DOD-STD-7935A to MIL-STD-498. It identifies key terminology changes and maps the DIDs of the preceding standards to and from the MIL-STD-498 DIDs.

a. Mapping of key terms. Figure 25 identifies selected terms in MIL-STD-498 and gives their counterparts in DOD-STD-2167A and DOD-STD-7935A.

b. Mapping of DIDs. Figure 26 identifies the DOD-STD-7935A DIDs and tells which MIL-STD-498 DIDs contain their contents. Figure 27 provides a similar mapping from the DOD-STD-2167A DIDs to the MIL-STD-498 DIDs. Figure 28 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.

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 25. Mapping of key terms.

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 26. 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 27. 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 28. Mapping of MIL-STD-498 DIDs to DOD-STD-2167A and DOD-STD-7935A DIDs.

5. TAILORING AND APPLICATION GUIDANCE

This section provides guidance to an acquirer on tailoring and applying MIL-STD-498. This section assumes a basic knowledge of system and software acquisition and does not attempt to describe these processes.

5.1 Approval of MIL-STD-498. MIL-STD-498 was issued 5 December 1994. Its approval extends for a two-year period, at the end of which time a commercial standard is expected to be ready to use in its place. Readers are advised to check with the Departmental Standardization Office (DepSO) Standards Improvement Executive (SIE) of their DoD service or government agency or the DoD Standards Office for current policy and guidance regarding the standard.

5.2 Ways of applying MIL-STD-498. As shown in Figure 29 and described below, MIL-STD-498 can be used in four basic ways. The applicability of each method depends upon DoD service or government agency policy and practice. The acquirer is advised to check with the Software Acquisition Executive for the DoD service or government agency for which the project is being developed to determine the current status of pertinent policy.

Four basic ways of applying MIL-STD-498			
Cite as a requirement in the RFP and the contract	Cite as a requirement in the RFP; require an SDP; put the resulting SDP on contract	Cite as guidance in the RFP; require an SDP; put the resulting SDP on contract	Cite as a requirement in the contract as a result of the winning bidder having proposed it

FIGURE 29. Ways of applying MIL-STD-498.

a. Requirement in RFP and contract. One way of using MIL-STD-498 is to cite it as a required standard, appropriately tailored, in the Request for Proposal (RFP) and in the contract. This is the traditional use of military standards.

b. Requirement in the RFP, relying on SDP. Another way to use MIL-STD-498 is to cite it as a required standard in the RFP, to require a Software Development Plan (SDP) in the proposal, and to put the SDP on contract. A possible drawback of this method is that changing the SDP during the project may require a contract modification.

c. Guidance in the RFP, relying on SDP. A third way to use MIL-STD-498 is to cite it as guidance in the RFP, require an SDP based on the guidance, and put the SDP on contract. The drawback cited above also applies here.

d. Developer-proposed requirement. DoD encourages bidders to propose alternative standards to those cited in the RFP. If a bidder proposes to follow MIL-STD-498 and that bidder is selected, the standard is placed on contract, constituting the fourth way of using the standard.

5.3 Information about tailoring. Regardless of the manner in which MIL-STD-498 is used, it is intended to be tailored for each type of software to which it is applied. This section provides guidance on tailoring.

5.3.1 What is tailoring? The tailoring process is illustrated in Figure 30. It consists of:

a. Evaluating each requirement in a standard or DID to determine whether it is necessary and cost effective for a given project, for a given phase or build within a project, and for particular software on the project

b. For standards, deleting, modifying, or adding to the requirements, as appropriate

c. For DIDs, deleting unneeded requirements or modifying requirements in a manner that does not increase required workload (This more restrictive tailoring for DIDs is based on U.S. law, specifically the Paperwork Reduction Act)

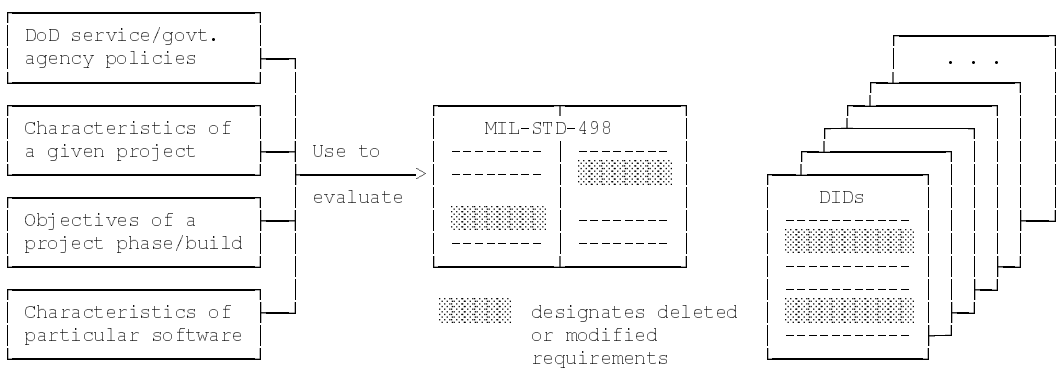


FIGURE 30. Overview of the tailoring process.

5.3.2 Why tailor?

a. DoD directed. DoDI 5000.2 directs that all acquisitions "contain only those requirements that are essential and cost-effective." This direction is the basis for tailoring.

b. Cost avoidance. Tailoring avoids unneeded activities, software products, controls, and practices. It can also eliminate duplicative requirements that may be invoked when multiple standards are on contract. These measures result in cost avoidance.

c. Shorter schedules. By avoiding unnecessary requirements, projects can be performed more quickly and their products delivered and fielded sooner.

d. Balance between near-term and long-term benefits. In making tailoring decisions, it is important to perform a risk analysis to balance near-term savings of cost and schedule against long-term risks. Sample trade-offs include the following:

1) Products needed for software support. Tailoring out software products needed for software support can save time and money in the initial development, but may have severe negative effects on the long-term cost of supporting the software.

2) Tests and evaluations. Tailoring out software product evaluations, software testing, and related activities can save time and money in the short term, but can result in reduced quality and expensive and time-consuming rework if software products are delivered before they are ready.

5.3.3 When is tailoring performed? Tailoring of MIL-STD-498 and its DIDs should be viewed as an incremental process, as illustrated in Figure 31 and described below.

a. During preparation of the draft RFP. Initial tailoring is often performed during preparation of a draft Request for Proposal (RFP). The acquirer assesses the project and its objectives and determines what requirements of MIL-STD-498 and its DIDs apply to each type of software. The draft RFP requests feedback on the tailoring as well as on other aspects of the procurement.

b. In response to the draft RFP. Prospective bidders often include proposed revisions to the tailoring in their comments on the draft RFP. This tailoring provides each prospective bidder's view as to the most cost-effective manner to perform the project given project requirements and the particular methods and tools used by that bidder.

c. During preparation of the final RFP. The tailoring in the final RFP is often revised from the draft based on input from the prospective bidders. This tailoring reflects the set of requirements (or guidance) on which bidders will base their proposals.

d. In proposals. Bidders' proposals may contain additional or revised tailoring if permitted by the RFP. In addition, bidders are often encouraged to suggest alternative standards and practices in place of military standards cited as requirements or guidance in the RFP. The amount and type of tailoring submitted in this step will depend upon how the procurement is worded.

e. In Best and Final Offers. Depending upon the procurement, the acquirer and bidders may perform additional tailoring during Best and Final Offers.

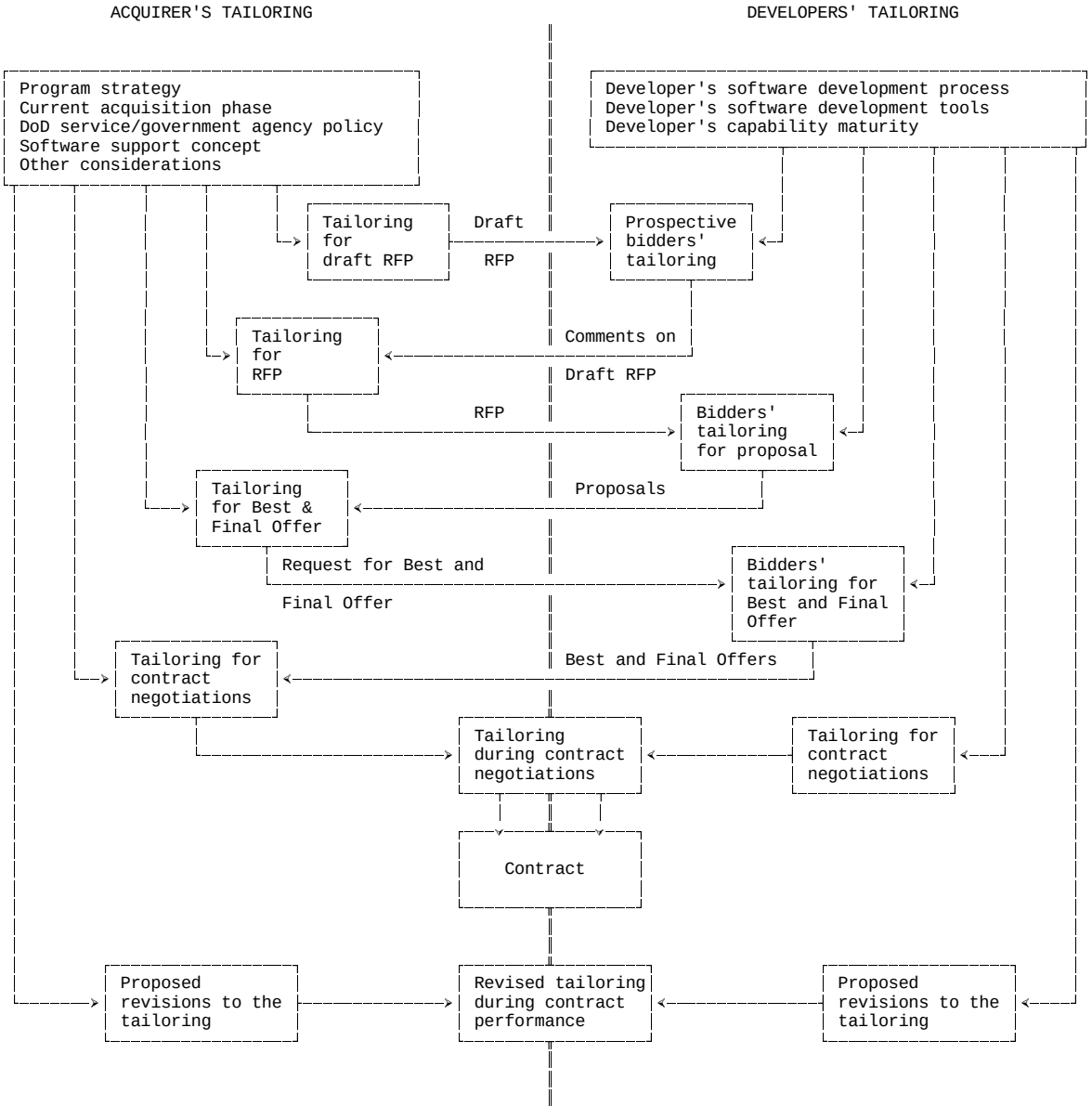


FIGURE 31. Tailoring as a shared, incremental process.

f. During contract negotiations. Additional tailoring may occur as part of contract negotiations with the winning bidder.

g. Throughout the project. As the project proceeds, both the acquirer and the developer will learn more about the job to be done and the activities and software products appropriate to that job. It is expected that this ongoing learning experience will be reflected in revised tailoring of both the standard and DIDs. Reexamination is especially appropriate at significant project markers such as the start of a new build.

5.3.4 Who performs tailoring?

a. Team effort. The preceding paragraphs show that both the acquirer and the developer play an important part in tailoring. The acquirer should also solicit participation from other interested parties, such as those listed below. This team effort is illustrated in Figure 32.

1) Technical staff available to the acquirer, such as systems and software engineering, configuration management, quality assurance, and test personnel
2) Contracting and contract administration personnel
3) User personnel
4) Support personnel

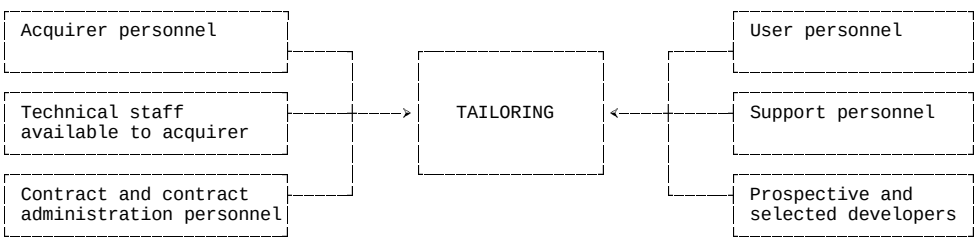


FIGURE 32. Contributors to tailoring.

b. Benefits of the team approach. The team approach has significant benefits. It ensures that each organization affected by the contract has an opportunity to state its needs, and it takes advantage of the combined expertise of all participants.

c. Responsibility. The final decisions about tailoring remain the responsibility of the acquirer.

5.4 Tailoring and applying MIL-STD-498. This section provides guidance on tailoring and applying MIL-STD-498. It lays out a process to be followed and provides considerations for each step.

5.4.1 Step 1: Determining the context of the software development. The first step in tailoring and applying MIL-STD-498 is determining the context of the software development. The following guidelines apply.

a. System context. Software may be developed as part of a larger, hardware-software system, such as a radar system, or it may constitute a system by itself, such as a payroll system. Determine, as a starting point, which of these contexts applies.

b. Familiarization with the system. Become familiar with the system's concept and status. For example:

1) What is the purpose of the system? Who is the user? Do descriptions exist?
2) How far along in planning or development is the system?
3) What decisions, if any, have been made about software's role in the system?

5.4.2 Step 2: Determining the program strategy for the system. The second step in tailoring and applying MIL-STD-498 is determining the program strategy being used to acquire the system. If selecting this strategy is someone else's responsibility, the task here is to participate in the selection or, if already completed, to find out what program strategy has been selected. If determining the strategy is your responsibility, the task here is to actually select it. The following guidelines apply.

a. Basic 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 shown in Figure 33 and summarized below.

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 33. Key features of three DoD program strategies.

1) 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.

2) 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.

3) 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 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.

b. Approach for selecting the program strategy. The program strategy is selected by the acquirer, but may be proposed by prospective or selected developers. Figure 35 shows 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 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.

5.4.3 Step 3: Selecting a strategy for acquiring the software. The third step in tailoring and applying MIL-STD-498 is selecting a strategy for acquiring the software. The following guidelines apply.

a. Relationship to system strategy. The software may be acquired under the same strategy as the system or under a different one, such as requiring that all software be finalized in the first build of the system. This possibility is illustrated in Figure 34.

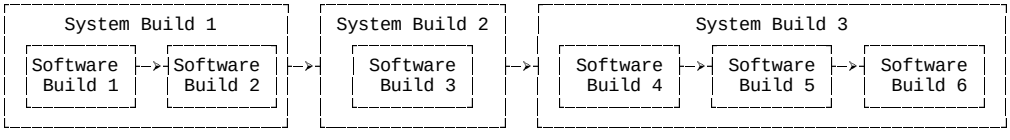


FIGURE 34. Example showing how system and software strategies can differ.

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 35. Sample risk analysis for determining the appropriate program strategy.

b. Examples to illustrate software strategies. Figures 37, 38, and 39 show how MIL-STD-498 might be applied under each of the program strategies described above. Figure 40 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, and they show each software product as a single entity, without depicting early drafts or updates.

c. An approach for selecting the software strategy. The software strategy is selected by the acquirer, but may be proposed by prospective or selected developers. Figure 35, presented on the preceding page, can be used for this selection. Paragraph 5.4.2.b describes how to use the form.

5.4.4 Step 4: Selecting the software support concept. Step 4 in tailoring and applying MIL-STD-498 is selecting the software support concept. The following guidelines apply.

a. Importance of the support concept. The support concept affects many planning and tailoring decisions. It should be treated as an important consideration in project planning.

b. Key issues. Key issues to be considered in determining the support concept are illustrated in Figure 36 and described below.

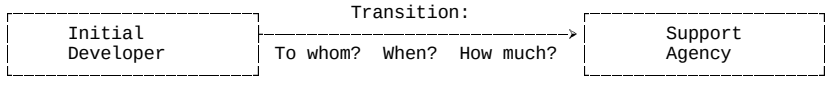
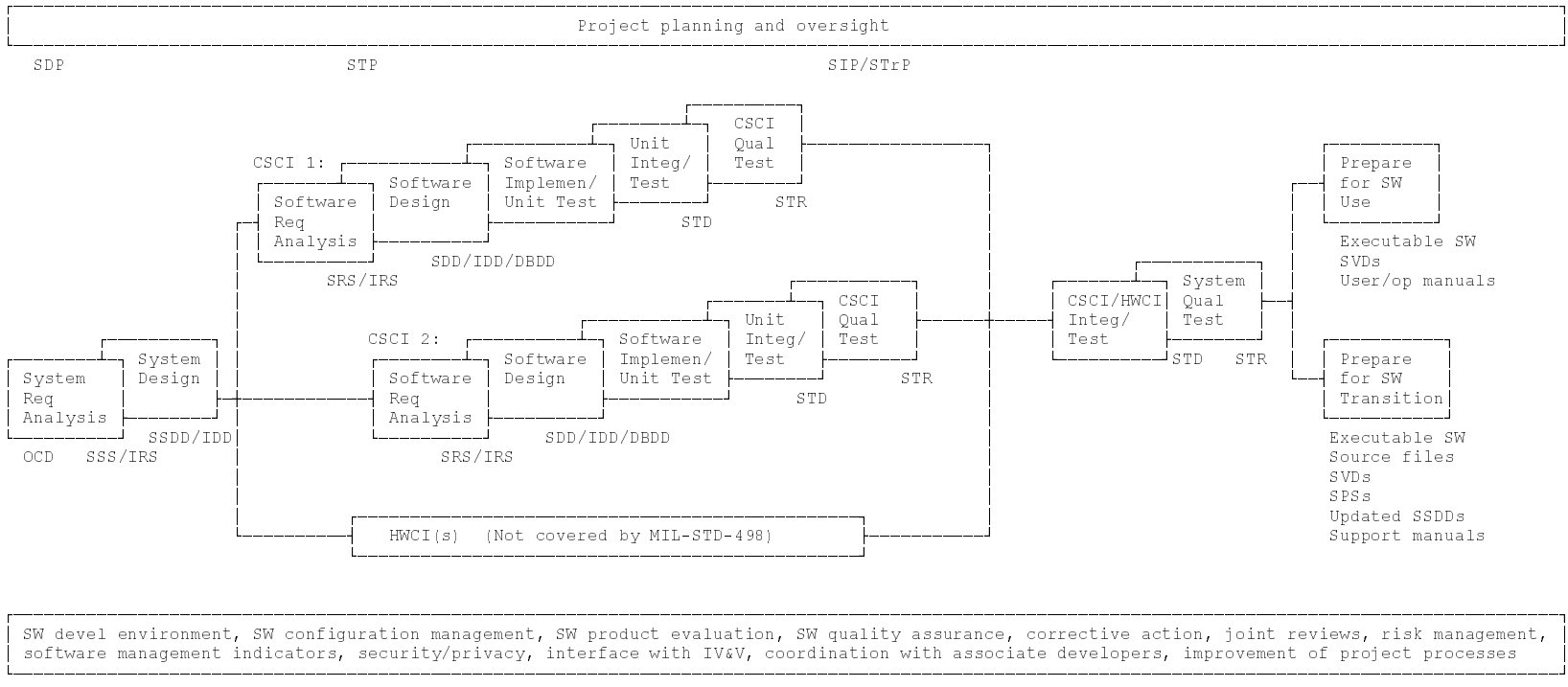


FIGURE 36. Key issues addressed by the support concept.

1) Who will support the software -- will the developer support the software, or will the software transition to another organization, such as a government support agency? Will the developer be completely out of the picture, so that all expertise, experience, and information must be captured and turned over to another agency or can some continuity of expertise be assumed?

2) If transition is to occur, when -- gradually over time? At the very end?

3) Is the software expected to change after delivery? How much support will be needed? What is the relationship to the Integrated Logistics Support Plan?



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

FIGURE 37. One possible way of applying MIL-STD-498 to the Grand Design program strategy.

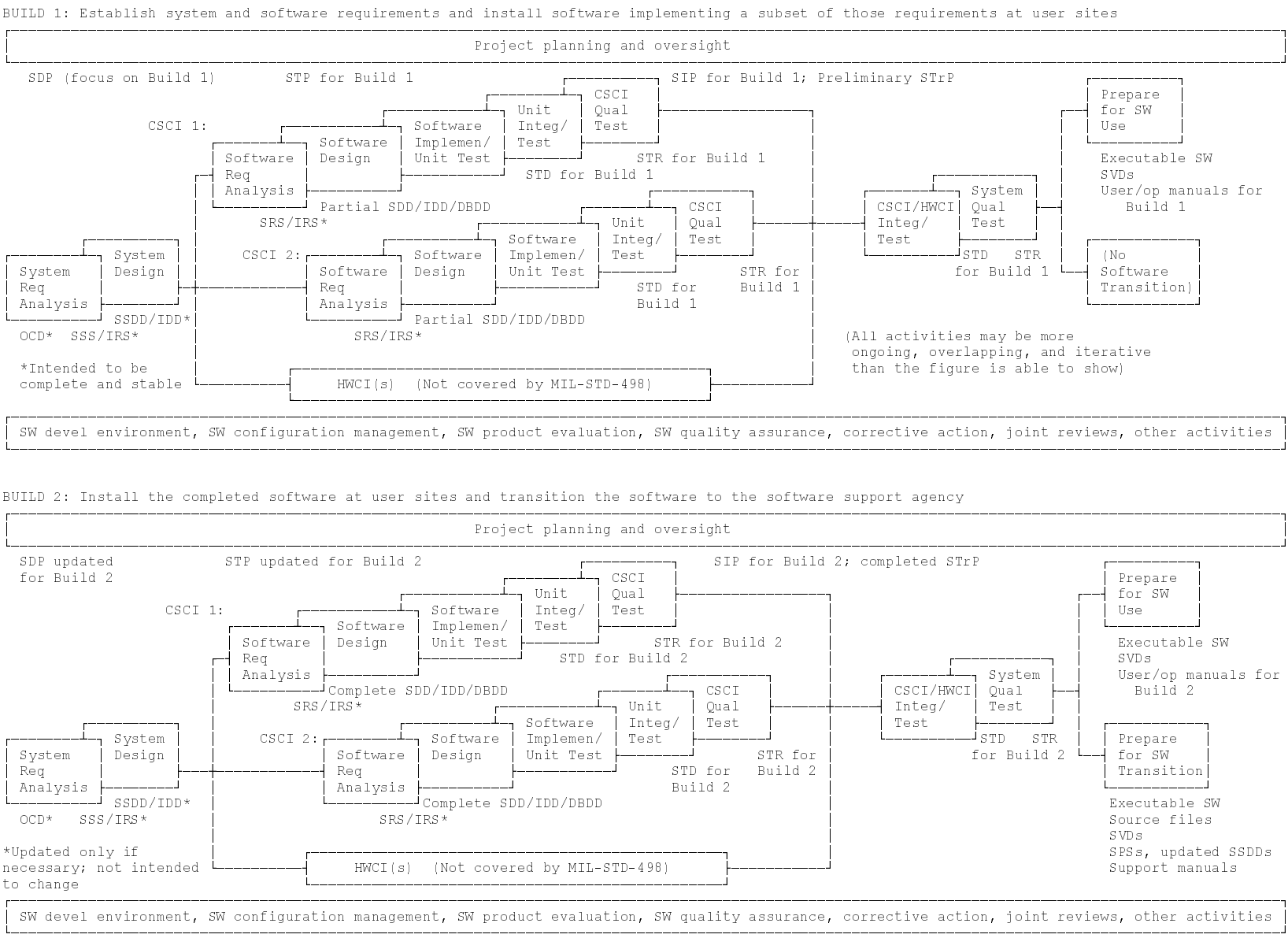


FIGURE 38. One possible way of applying MIL-STD-498 to the Incremental program strategy.

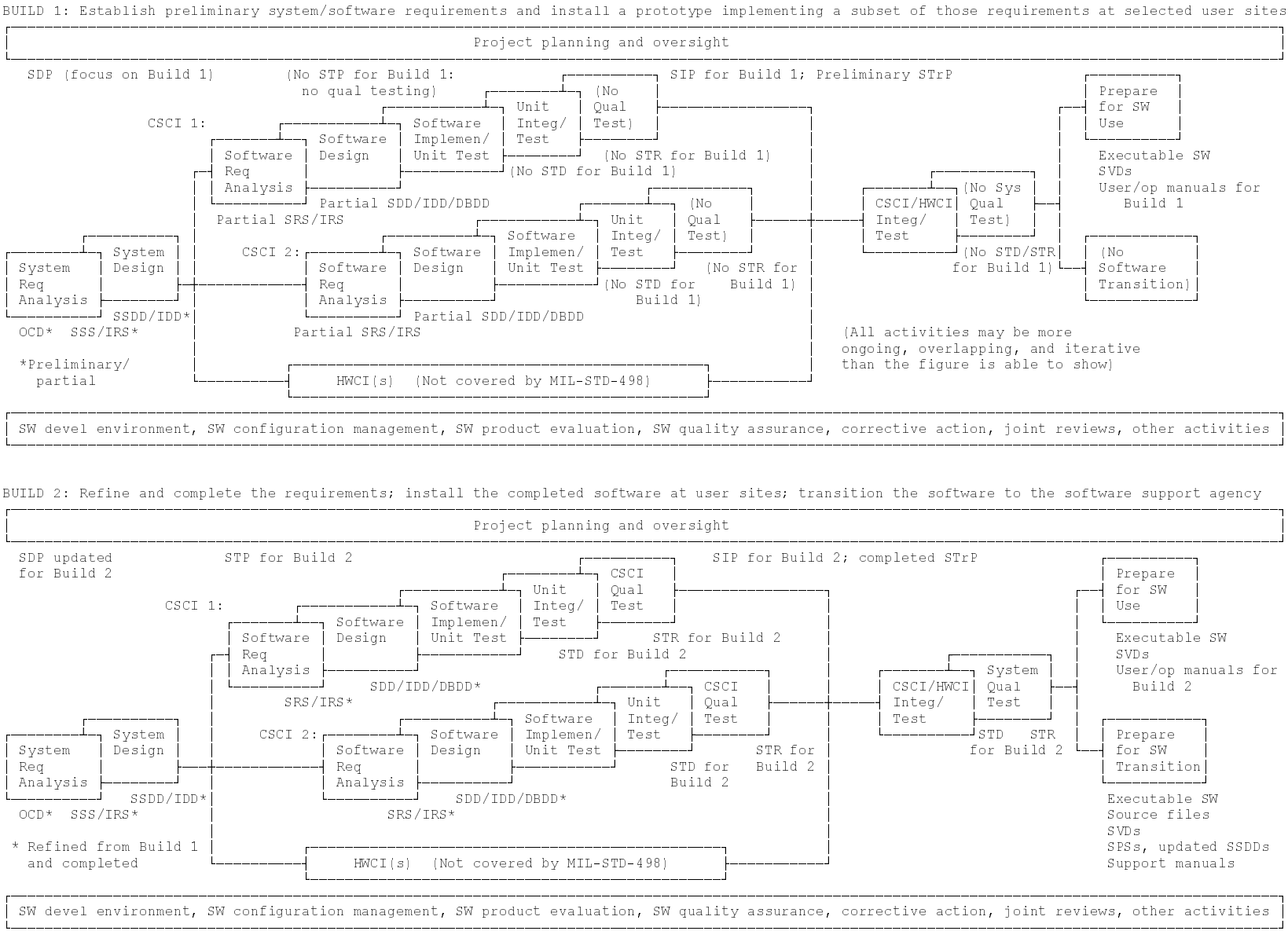


FIGURE 39. One possible way of applying MIL-STD-498 to the Evolutionary program strategy.

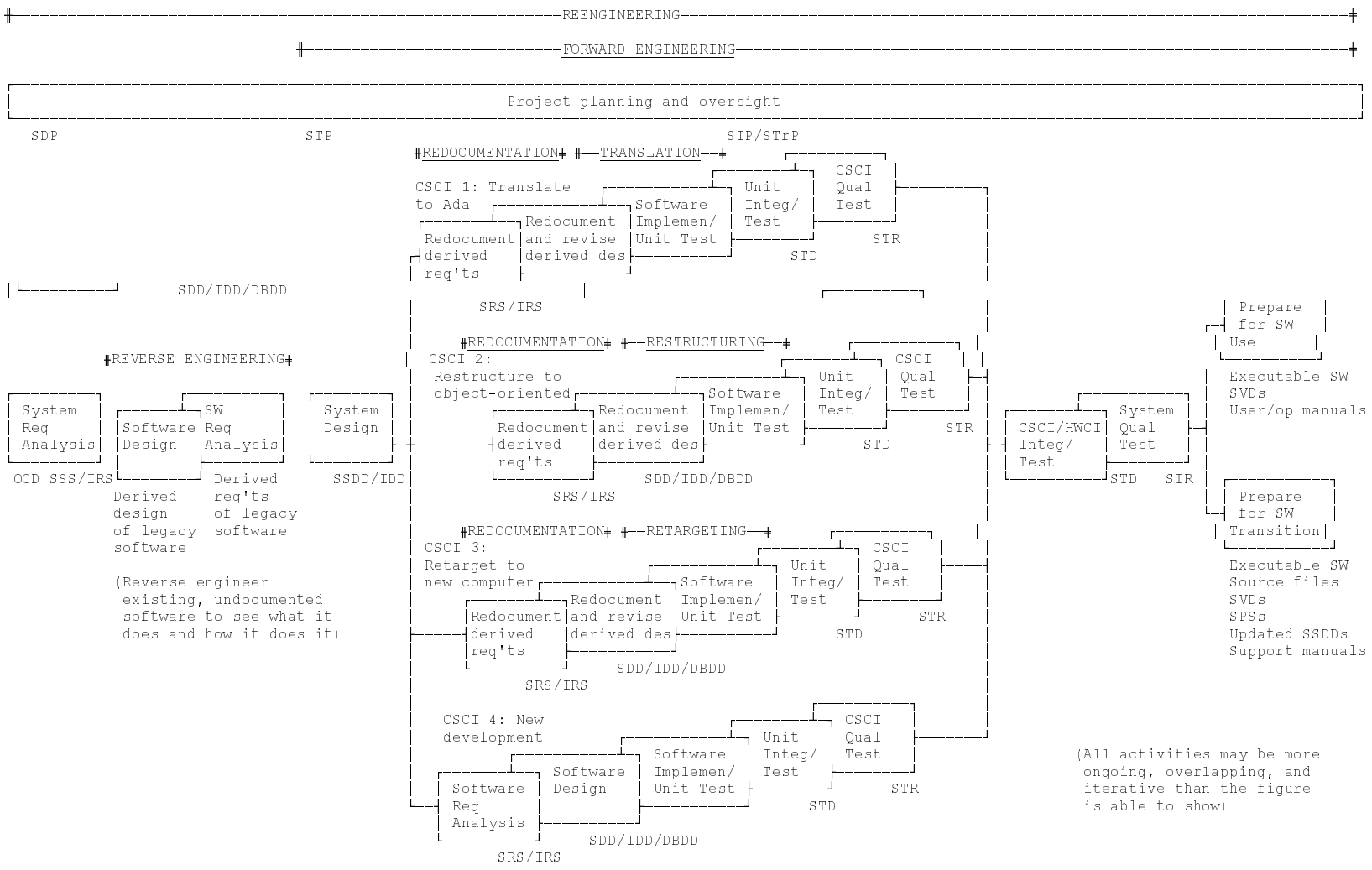


FIGURE 40. One possible way of applying MIL-STD-498 to a reengineering project.

c. Implications of the support concept. If the developer will not be involved in software support, a key concern is transferring the developer's knowledge to the support agency. This transfer of knowledge should not be underestimated. The developer accumulates a tremendous amount of knowledge about the software and the software engineering and test environments. Transferring that knowledge to another agency requires significant time and effort on the part of both the developer and the support agency. Key considerations include the following:

1) Software Transition Plan. If the developer and the support agency are different agencies, you will probably want to require a Software Transition Plan (STrP), coordinate that plan with the support agency, and budget resources to carry out the training and other transition activities it describes. If the developer and support agency are the same, such a plan will not be needed.

2) Design information. A survey of support agencies during the development of MIL-STD-498 revealed significant differences in the way different agencies view design information. Some agencies want every bit of design information they can get; others do not use it, preferring to rely on the code. Check with your support agency to determine how much and what type of design information they want. Requiring delivery of more than is needed, or requiring early design information when only the "as built" design is needed, can waste development resources. If the developer will continue to be involved during support, this transfer of design information will not be necessary.

3) Information in software development files. Under MIL-STD-498, the test cases, test procedures, test data, and test results for developer-internal testing are recorded in software development files, not in deliverable software products. Some software support agencies want this and other information in the software development files. If this is the case on your project, discuss with your Data Manager the best way to achieve delivery of the desired information, possibly by using the Data Accession List. If the developer will continue to be involved during support, this delivery will not be necessary.

4) Support manuals. MIL-STD-498 offers DIDs for two manuals intended for the support agency. These are the Computer Programming Manual (CPM) and the Firmware Support Manual (FSM). If the developer will not be involved in support, it may be appropriate to order these manuals. If the developer will continue to be involved, delivery of these manuals may not be necessary.

5.4.5 Step 5: Identifying types of software on the project. Step 5 in tailoring and applying MIL-STD-498 is identifying the types of software involved on the project. The following guidelines apply.

a. Software types. A typical project involves several types of software, each of which may require different build planning and tailoring decisions. Figure 41 illustrates the concept of dividing software into types. The paragraphs below discuss possible types that might be used.

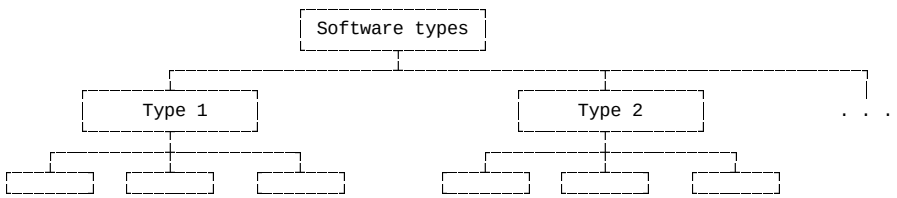


FIGURE 41. Dividing software into types.

1) Deliverable versus non-deliverable software. One basic distinction is between software that must be delivered versus software that need not be delivered. It is often appropriate to apply different tailoring to these two types of software.

2) Software with different uses. The software on a project is often intended for significantly different uses. Examples include software that will be used in the operational environment, training software, simulation software, and software in the software engineering and test environments. These differences can lead to different tailoring decisions.

3) Precedented versus unprecedented. A third distinction is between software that is precedented, that is, similar to software that has been developed before, and software that is unprecedented, involving more uncertainty and technical risk. These distinctions can also lead to different tailoring decisions.

Note regarding development status: For purposes of tailoring, it is not necessary to establish types based on whether the software will be newly developed, modified, reused as-is, etc. Appendix B of MIL-STD-498 interprets the standard to accommodate these variations, with no tailoring required on the part of the acquirer.

b. Incremental definition of types. The different types of software on a project are often identified only as planning, tailoring, and project execution proceed. Begin by identifying some preliminary types, and refine the distinctions as the need arises.

5.4.6 Step 6: Defining the software builds. Step 6 in tailoring and applying MIL-STD-498 is defining the software builds. The following guidelines apply.

a. Who should define the builds? Defining the software builds on a project and tailoring MIL-STD-498 for each build may be accomplished in several ways. The acquirer might select an overall software 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 that follow provide guidelines for planning the builds and tailoring the standard without attempting to divide these activities between the acquirer and developer.

b. Key issues. Defining the builds on a project involves two basic issues, identified in Figure 42 and described below.

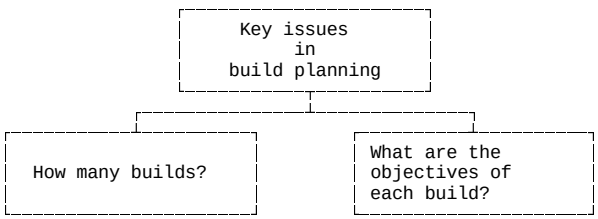


FIGURE 42. Key issues in build planning.

1) Deciding how many builds. For the Grand Design program strategy, there will be a single build. For the Incremental, Evolutionary, and variations of these strategies, there may be two, three, or more. The specific circumstances of the project will determine the best number to use. Considerations include:

   a) A reasonable division of the software into increments
   b) How soon and how often builds need to be fielded
   c) Reasonable allowance of time to accomplish each build
   d) Number of iterations needed to fully understand user needs
   e) Availability of staff and other resources to apply to each build
   f) Coordination with the program strategy for the system

2) Defining the objectives of each build. The key characteristics of each build are the objectives established for it. The more clearly these objectives are defined, the better basis they will form for tailoring and for monitoring the work.

c. Example. The top part of Figure 44 illustrates the defining of builds. 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.

d. Worksheets for build planning and tailoring. Appendix C of this guidebook provides sample worksheets to be used for build planning and tailoring. These worksheets are laid out like the example in Figure 44. Create your own worksheets from the samples, then begin by filling out the top part of the worksheets with the build objectives. If the objectives are lengthy, point from the worksheets to an attachment describing the objectives of each build. The following considerations apply.

1) Different types of software. Build planning can be completely different for different types of software on a project. One type of software may be developed over three or four builds; another type may need to be completed in a single build. If this is the case on your project, decide whether to:

a) Prepare a separate set of worksheets for each type of software as shown in Figure 43, or
b) Accommodate the different types on a single worksheet

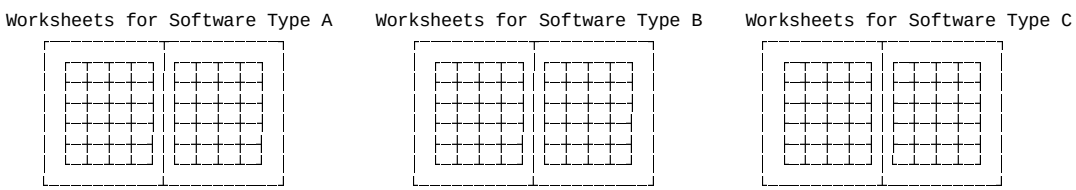


FIGURE 43. Separate worksheets for each type of software.

2) Incremental build planning. It may not be possible to make final decisions about the number of builds or the objectives of each build during initial planning. These decisions can be refined as the procurement and the project proceed.

3) Deferred build planning. To provide objectives for the overall contract, without planning the individual builds, ignore the columns for Builds 2 through 4 and express one set of objectives for the contract. The developer can then propose the detailed build planning.

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 required 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 required for Build 1 testing	Yes: As required for Build 2 testing	Yes: Set up fully for Build 3 qualification testing	Yes: Update as needed for Build 4 qualification testing

FIGURE 44. Example of build planning for a MIL-STD-498 project.

5.4.7 Step 7: Tailoring MIL-STD-498 for each build. Step 7 in tailoring and applying MIL-STD-498 is tailoring the standard, that is, identifying the MIL-STD-498 activities to be performed in each build. The following guidelines apply.

a. Completing the worksheets for the standard. This step consists of filling out the lower portion of each of the MIL-STD-498 worksheets that you established in the preceding step. For each such worksheet, proceed through the entries in the worksheet, read the actual text in the standard, decide whether the activity applies in each build, decide the degree to which it applies, and record these decisions in the worksheet. This process is illustrated in Figure 45.

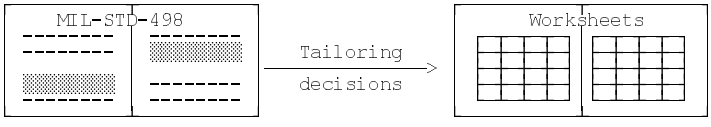


FIGURE 45. Using the worksheets to record tailoring decisions.

b. Example. The lower part of Figure 44 illustrates the results of this step. Listed on the left are 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.

c. Accommodating differences in different builds. As the example shows, 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. 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 depend upon the objectives established for each build.

d. Different types of software. Different types of software will call for different tailoring decisions. Activities that apply to critical operational software will be different from those applied to non-deliverable software in the development environment. Reflect these differences in the worksheets.

e. Activities that result in software products. Many of the activities in MIL-STD-498 involve the preparation of software products. Examples of such products include plans, specifications, manuals, and the software itself. Figure 8 (in Section 4) provides an overview of MIL-STD-498 software products. This figure may prove helpful in deciding whether to require the associated activities. Note: Delivery of the resulting software products is a later decision.

f. A tailoring process. The key question in each tailoring decision is: does the activity in question support the objectives of the current build, a future build, or the software support concept? Figure 46 illustrates a process for performing the tailoring. This process helps to ensure that the right questions are asked about each MIL-STD-498 requirement.

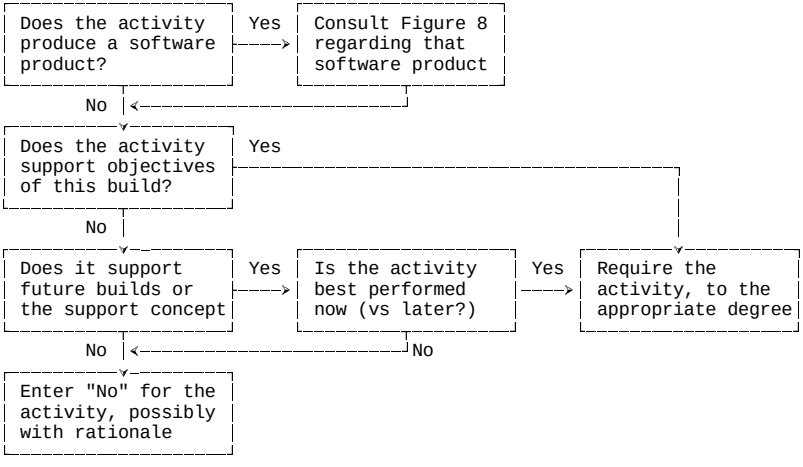


FIGURE 46. Process for tailoring the standard.

g. Incremental tailoring. It may not be possible to make final tailoring decisions during initial planning. These decisions can be refined as the procurement and the project proceed.

h. Deferred tailoring. To provide overall tailoring for the contract, without tailoring for the individual builds, ignore the columns for Builds 2 through 4 and express one set of objectives for the contract. The developer can then propose the detailed tailoring for each build.

5.4.8 Step 8: Tailoring the DIDs as activity checklists. Step 8 in tailoring and applying MIL-STD-498 is tailoring the DIDs as activity checklists. The following guidelines apply.

a. DIDs as activity checklists. A unique feature of MIL-STD-498 is that it uses the DIDs to fully define many of the activities in the standard. The DIDs specify the information that is to be defined and recorded in carrying out the activities. This use of the DIDs applies regardless of whether the information will be deliverable.

b. Tailoring the DIDs for this purpose. It may not be necessary to have the developer prepare all information called for in the DIDs for required software products. Part of tailoring the standard therefore consists of reviewing each relevant DID and tailoring out any portions not needed for the project, for a given build, or for given types of software. Figure 47 illustrates the relationship of the tailoring for the standard and the DIDs.

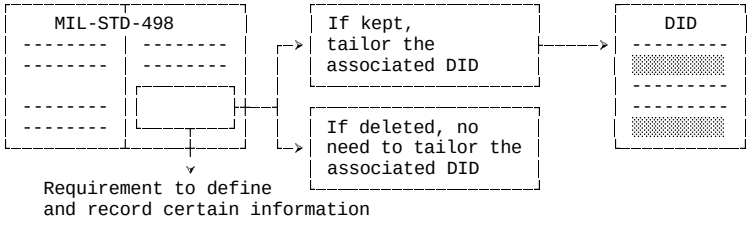


FIGURE 47. Relationship of tailoring decisions for the standard and DIDS.

c. Tailoring worksheets for the DIDs. Appendix C of this guidebook contains a sample worksheet for the DIDs. This sample is intended to be used as a model for creating worksheets for each DID that needs to be tailored. These worksheets can be used in the same manner as the worksheet for the standard, to indicate which requirements apply in each build and to what extent.

d. Types of software. As with tailoring of the standard, tailoring of the DIDs will differ for different types of software. The worksheets should reflect these differences.

e. Incremental or deferred tailoring. In many cases, it will not be possible to know in advance which paragraphs of the DIDs do and do not apply. The tailoring may need to be an ongoing process, discussed at joint reviews as the project proceeds. To provide overall tailoring for the contract, without tailoring for the individual builds, ignore the columns for Builds 2 through 4 and express one set of objectives for the contract. The developer can then propose the detailed tailoring for each build.

5.4.9 Step 9: Recording tailoring decisions in the Statement of Work. Step 9 in tailoring and applying MIL-STD-498 is recording the tailoring decisions of Steps 7 and 8 in the Statement of Work. The following guidelines apply.

a. Tailoring by exception. One method of recording the tailoring decisions in the Statement of Work is tailoring by exception. This method consists of requiring the fulfillment of all requirements of the standard and DIDs with certain, itemized exceptions. Figure 48 provides an example. Note that the DID paragraphs should have "10.2" appended to the front of the numbers given in the tailoring worksheets.

b. Tailoring by inclusion. An alternative method is tailoring by inclusion. This method consists of listing the paragraph numbers of the requirements in the standard and DIDs that must be fulfilled, with any special considerations noted.

In Build 3 of the operational software, the developer shall comply with all requirements of MIL-STD-498 and the associated DIDs with the following exceptions:

a) 4.2.4.4 Delete entire paragraph

b) 5.1.1 In preparing the Software Development Plan, ignore SDP DID paragraphs pertaining to requirements tailored out of the standard.

c) 5.1.4 Delete "and training"
In preparing the Software Installation Plan, ignore SIP DID paragraphs pertaining to training.

d) 5.2.4 Add "SDFs shall be subject to periodic review by the acquirer."

e) 5.3.2 In preparing the Operational Concept Description, ignore the following OCD DID paragraphs:

1) 10.2.3.3 Description of current system or situation
2) 10.2.4.4 Changes considered but not included

f) 5.12.4 Change "other assistance" to "on-site assistance for one month"

FIGURE 48. Sample language for specifying tailoring in the Statement of Work.

5.4.10 Step 10: Clarifying "shell requirements". Step 10 in tailoring and applying MIL-STD-498 is clarifying MIL-STD-498's "shell requirements." The following guidelines apply.

a. MIL-STD-498's "shell requirements". MIL-STD-498 contains a number of requirements whose specifics must be provided in the contract. These requirements are termed "shell requirements" because they are empty shells, often imposing no requirement at all, until fleshed out in the contract. "Shell requirements" serve a three-fold purpose: 1) to remind the acquirer to consider the issue raised in each requirement; 2) to make the standard self-tailoring (since most impose no work without additional words in the contract); and 3) to provide a framework for incorporating project-specific requirements in the contract.

b. Identification of the "shell requirements". Figure 49 identifies MIL-STD-498's shell requirements, provides a summary of each, and tells what information should be provided in the contract for the requirement to be complete.

498 Para	Summary of the Requirement	To Flesh This Out, Include in the Contract:
4.2.3.1	Reused software products shall comply with the data rights in the contract	Data rights for each type of deliverable software
4.2.4.4	Develop and implement a strategy for assuring fulfillment of requirements deemed critical by the contract or system specification	Identification of "critical" requirements
4.2.5	Analyze and implement contract requirements regarding computer hardware resource utilization	Requirements concerning computer hardware resource utilization (for example, memory reserves)
5.1.4	Develop a plan for performing software installation and training at user sites specified in the contract	Identification of user sites at which the developer is to install software and/or train users in each build
5.1.5	Identify software development resources needed by the support agency to fulfill the support concept specified in the contract	Description of the software support concept, including transition aspects in each build
5.3.1	Analyze user input provided by the acquirer	Description of the type and amount of user input to be provided for analysis in each build
5.7.1	For deliverable software, obtain acquirer approval to use any programming language not specified in the contract	Requirements concerning the programming language(s) to be used in each build
5.9.4	If CSCI qualification is to be witnessed, dry run the test cases and procedures	Indication of whether CSCI qualification testing will be acquirer-witnessed in each build

FIGURE 49. MIL-STD-498's "shell requirements".

498 Para	Summary of the Requirement	To Flesh This Out, Include in the Contract:
5.11.4	If system qualification is to be witnessed, dry run the test cases and procedures	Indication of whether system qualification testing will be acquirer-witnessed in each build
5.12.1	Prepare the executable software for each user site	Identification of user sites at which the developer is to install software and/or train users in each build
5.12.4.a	Install and check out the software at user sites specified in the contract	Identification of user sites at which the developer is to install software and/or train users in each build
5.12.4.b	Provide training to users as specified in the contract	Description of user training required in each build
5.12.4.c	Provide other assistance to user sites as specified in the contract	Description of additional assistance to users required in each build
5.13.1	Prepare the executable software for the support site	Identification of the support site
5.13.2	Prepare the source files for the support site	Identification of the support site
5.13.7.a	Install and check out the software in the support environment designated in the contract	Identification of the support site/environment
5.13.7.b	Demonstrate that the deliverable software can be regenerated and maintained using ... software and hardware designated in the contract or approved by the acquirer	Requirements or restrictions, if any, on the software and hardware that are to be assumed in the support environment
5.13.7.c	Provide training to the support agency as specified in the contract	Description of training to be provided to the support agency in each build
5.13.7.d	Provide other assistance to the support agency as specified in the contract	Description of additional assistance to be provided to the support agency in each build
5.14.2	Process changes to acquirer-controlled entities in accordance with contractually established forms and procedures, if any	Descriptions of forms and procedures, if any, to be used for processing changes to acquirer-controlled software products
5.14.4	Support acquirer-conducted configuration audits as specified in the contract	Description of acquirer-conducted configuration audits, if any, to be held in each build
5.19.3	Meet the security and privacy requirements specified in the contract	Security and privacy requirements, if any, to be followed on the project (as opposed to requirements on the system, which will be in specifications)
5.19.5	Interface with IV&V agent(s) as specified in the contract	Requirements, if any, for interfacing with IV&V agents in each build
5.19.6	Coordinate with associate developers, working groups, and interface groups as specified in the contract	Requirements, if any, for coordinating with associate developers, working groups, and interface groups in each build

FIGURE 49. MIL-STD-498's "shell requirements" - *(continued)*.

5.4.11 Step 11: Selecting deliverables. Step 11 in tailoring and applying MIL-STD-498 is selecting the software products to be designated deliverable. The following guidelines apply.

a. MIL-STD-498's philosophy regarding 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. Software products should be designated as deliverables 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.

b. Deciding which software products to make deliverable. Step 7 results in decisions regarding which software products the developer should prepare. Figure 8 in Section 4 was offered as an aid in making these decisions. That figure, together with the DIDs themselves, may also be helpful in deciding which software products to make deliverable. Figure 50 illustrates the distinction between these two decisions. The key question in the current step is whether it is enough for each software product to be generated and available for on-site review, or whether you also want to require the developer to transform the recorded information into deliverable form and deliver it. Paragraph 5.4.4.c describes the impact of the support concept on these decisions.

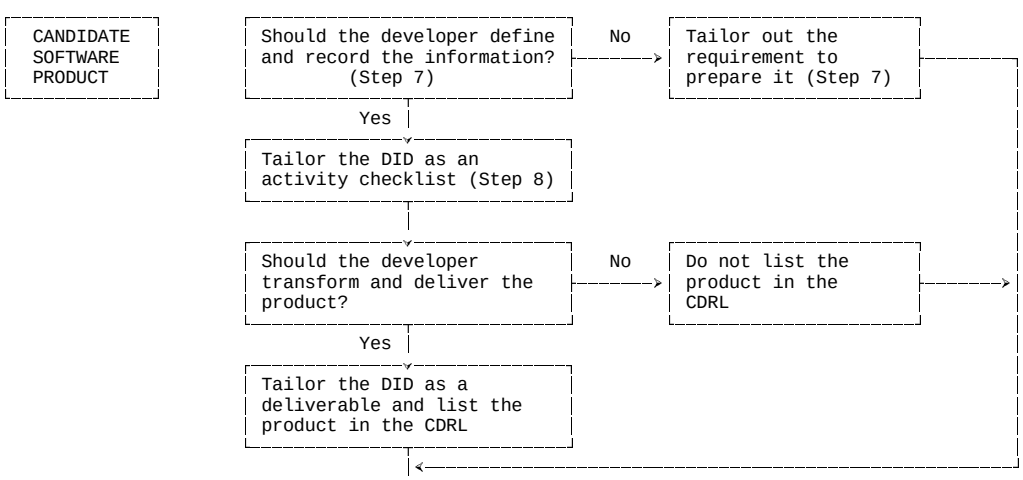


FIGURE 50. Distinguishing between preparation and delivery of software products.

5.4.12 Step 12: Tailoring the DIDs for deliverables. Step 12 in tailoring and applying MIL-STD-498 is tailoring the DIDs for those software products designated as deliverables. The following guidelines apply.

a. Possible differences in tailoring. In some instances, you may make different tailoring decisions when tailoring the DIDs for use in defining deliverables than you did in using the DIDs as activity checklists (Step 8). For example, you may wish to make only a subset of the information deliverable.

b. Indicating the differences. If there are differences in these two types of tailoring, mark them on the same or different copies of the tailoring worksheets, whichever is handier. This tailoring will be transferred to the CDRL in a later step.

c. Selecting between alternative "homes" for deliverable information. MIL-STD-498 and its DIDs provide alternative locations for certain information in deliverables. Figure 51 identifies the topics in question and the alternatives offered. If you wish to remove this flexibility and require that information be included in a given deliverable, specify this choice on the CDRL. To retain the flexibility, take no action. The developer can then place the information where it seems most appropriate and reference it from the other deliverables.

Topic	Possible Locations	MIL-STD-498 Paragraph
Requirements concerning system interfaces	SSS or IRSs	5.3.3
System-wide design pertaining to interfaces	SSDD or IDDs	5.4.1
System-wide design pertaining to databases	SSDD or DBDDs	5.4.1
System architectural design pertaining to interfaces	SSDD or IDDs	5.4.2
Requirements concerning CSCI interfaces	SRSs or IRSs	5.5
CSCI-wide design pertaining to interfaces	SDDs or IDDs	5.6.1
CSCI-wide design pertaining to databases	SDDs or DBDDs	5.6.1
CSCI architectural design pertaining to interfaces	SDDs or IDDs	5.6.2
CSCI detailed design pertaining to interfaces	SDDs or IDDs	5.6.3
CSCI detailed design of software units that are databases or that access or manipulate databases	SDDs or DBDDs	5.6.3

FIGURE 51. Alternative "homes" for selected information.

5.4.13 Step 13: Selecting the format of deliverables. Step 13 in tailoring and applying MIL-STD-498 is selecting the format of deliverables. The following guidelines apply.

a. Multiple formats possible. Traditional deliverables take the form of paper documents following DID formats as specified in block 16 of the CDRL. This form works well for some deliverables, but it is not the only form, and alternatives should be considered. One variation is word processing files. This saves paper, but still requires the developer to follow the DID format. 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 CASE tools. These variations can minimize the time spent transforming actual work products into deliverables. Be sure, however, that you and other recipients will be able to use the software products in the delivered format and that trade-offs, such as additional time needed to find desired information, are considered in your decision.

b. Combining (consolidating) deliverables. On some projects, it may be desirable to combine deliverables. Examples include a Consolidated Software Plan, combining two or more of the plans; a Consolidated Software Requirements Document; a Consolidated Software Design Document; a Consolidated Software Test Document; a Consolidated Software User/Operator Manual; and a Completely Consolidated Software Development Record, combining all of the software products. Figure 52 illustrates this concept.

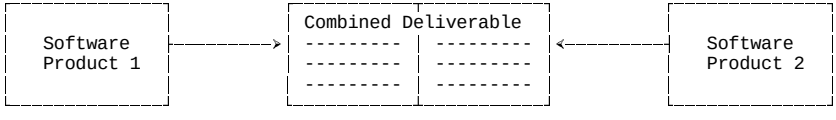


FIGURE 52. Combining software products into a single deliverable.

1) Not "combining DIDs". It is important to note that this discussion is about combining deliverables, not about combining DIDs. Combining DIDs is a more difficult undertaking, resulting in a new DID that would need to be coordinated and approved by your Data Manager. In combining deliverables, the DIDs themselves are not combined; it is the resulting deliverables that are combined.

2) Considerations in combining deliverables. In deciding whether to combine deliverables, it is important to consider the compatibility of the deliverables, for example, the timing of their preparation, their intended audiences, and their frequency of update. Combining incompatible deliverables may cause problems. An advantage of consolidation is fewer documents to control, track, and handle.

3) Sample wording. Figure 53 provides sample wording that can be placed in Block 16 of the CDRL to combine two deliverables. For variations, such as combining three or more deliverables, the sample wording can be adapted as needed.

To require that two software products (X and Y) be combined and delivered as a single deliverable:

a. Make a CDRL Item (say A001) for X. Insert its normal DID number in Block 4, with a reference to Block 16. Make a CDRL Item (say A002) for Y. Insert its normal DID number in Block 4, with a reference to Block 16.

b. Insert instructions for the consolidation into Block 16 of both CDRL Items (or put in one and reference from the other). For example:

"Combine CDRL Items A001 and A002 into one document. Paragraph numbers, titles, and content shall be as follows:

1. Scope: Provide the information required by Section 1 of the X and Y DIDs in the paragraph structure required by the X DID.

2. Referenced documents: Provide the information required by Section 2 of the X and Y DIDs in the paragraph structure required by the X DID.

3. (Title of X): Provide the information required by Sections 3 through N of the X DID (up to but not including the Notes section). Use paragraph numbering that retains the basic structure of the X DID.

4. (Title of Y): Provide the information required by Sections 3 through N of the Y DID (up to but not including the Notes section). Use paragraph numbering that retains the basic structure of the Y DID.

5. Notes: Combine the information required by the Notes sections of the X and Y DIDs.

Appendixes: Combine the information required by the Appendixes sections of the X and Y DIDs.

FIGURE 53. Sample CDRL wording for combining deliverables.

5.4.14 Step 14: Scheduling deliverables. Step 14 in tailoring and applying MIL-STD-498 is scheduling the selected deliverables. The following guidelines apply.

a. The "tyranny of the CDRL". 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. Common mistakes are:

1) Treating 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.

2) Treating all 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.

3) Imposing a predetermined sequence on MIL-STD-498 software products. While sequential delivery is the easiest to picture, it is often not the most effective approach. Figure 54 provides a simple illustration of the difference between sequential delivery and incremental, overlapping delivery of software products.

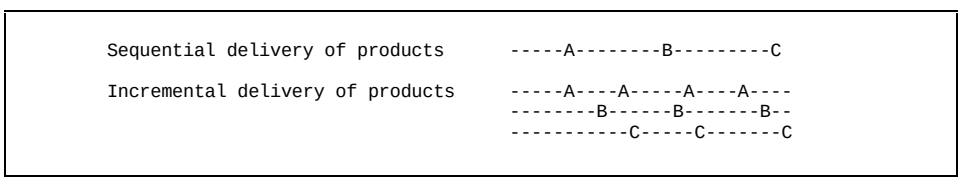


FIGURE 54. Sequential versus incremental/overlapping delivery of software products.

4) Unrealistic dates for early deliverables. Impatient to see results, the acquirer often schedules due dates for plans and requirements so soon that they cannot possibly be done well or with the consideration necessary for project success.

b. Retaining flexibility. As with build planning, the acquirer may set forth general milestones and have the developer provide specifics or may provide specific schedules. To the maximum extent possible, the CDRL should leave 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 can lay out a proposed schedule that meets the constraints in the CDRL. Final agreement on scheduling can take place during review and approval of the plan.

5.4.15 Step 15: Preparing the CDRL (DD Form 1423). Step 15 in tailoring and applying MIL-STD-498 is preparing the CDRL (DD Form 1423). The following guidelines apply.

a. Block 4: Authority. Block 4 of the CDRL is used to hold the DID number. If the DID has been tailored or if contractor format is acceptable, append "/T" to the DID number.

b. Block 5: Contract reference. Block 5 identifies the paragraph number(s) in the Statement of Work, standard, specification, or other documents that task the developer to generate the information called for by the CDRL Item. An entry such as "MIL-STD-498 5.5" can be used in this block.

c. Block 10: Frequency. Block 10 specifies the required frequency of delivery. To request developer-proposed scheduling, enter "See Block 16" and place the request there.

d. Blocks 12 and 13: Date of First and Subsequent Submission. Blocks 12 and 13 specify, respectively, the required dates of first and subsequent submissions of the deliverable. To request developer-proposed scheduling, enter "See Block 16" and place the request there.

e. Block 16: Remarks. Block 16 is used to convey information not contained in the other blocks. Figure 55 provides examples. Note that whenever DID paragraphs are cited, "10.2" needs to be appended to the front of the paragraph number shown in the DID and in the tailoring worksheets.

CANDIDATE ENTRIES IN BLOCK 16 OF THE CDRL

1) Tailoring decisions for deliverables, such as

   "The following DID paragraphs do not apply: 10.2.5.3, 10.2.5.5"

2) Formatting decisions, such as:

   "Contractor format acceptable"

3) Decisions about delivery media, such as

   "Deliver on PC-compatible 3-1/2" diskettes"

4) Clarifications of the frequency and due dates in Blocks 10, 12, and 13, such as:

   "Frequency and due dates to be determined based on the approach described in the Software Development Plan"

FIGURE 55. Examples of information that can be included in Block 16.

5.4.16 Step 16: Including SDP preparation in Instructions to Bidders. Step 16 in tailoring and applying MIL-STD-498 is including in Instructions to Bidders a requirement to include a Software Development Plan in the proposal. The following guidelines apply.

a. Requesting the SDP in Instructions to Bidders. The Instructions to Bidders in a Request for Proposal identify all of the elements that must be included in bidders' proposals and provide instructions, constraints, criteria, and other information to bidders. It is recommended that among these instructions the acquirer include a requirement to include a draft SDP in the proposal.

b. Role of the SDP. MIL-STD-498 tells what the developer is to do over the course of a contract. Its requirements leave significant leeway for how the developer will carry out the contract. The SDP provides this information.

c. Relationship to SDP on the CDRL. Including SDP preparation in the Instructions to Bidders is not the same as including the SDP in the CDRL. Including the SDP in the CDRL will result in delivery of an SDP after the contract is awarded. To receive an SDP with bidders' proposals, it is necessary to call for it in the Instructions to Bidders. The SDP in the CDRL will then be an update to the one in the winning proposal. This relationship is illustrated in Figure 56.

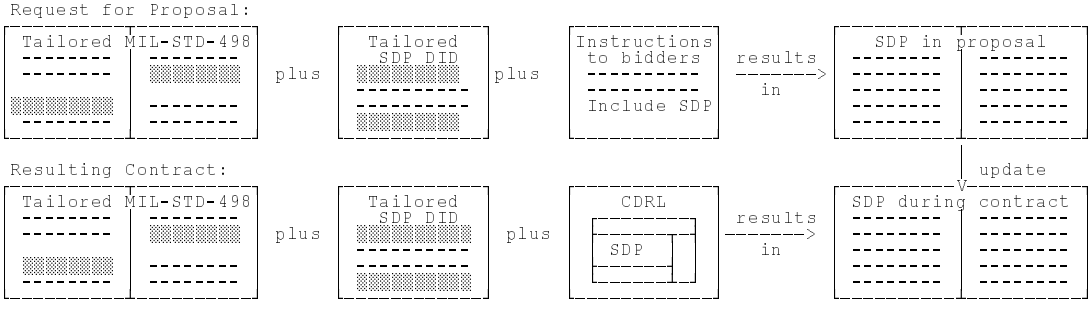


FIGURE 56. Requiring an SDP in proposals as well as after contract award.

d. Requiring adherence to the SDP. There are two ways to require adherence to the SDP. One is to put paragraph 5.1.6 of MIL-STD-498 on contract. This paragraph requires adherence to the SDP. The second method is to include in the Statement of Work a requirement similar to paragraph 5.1.6 of MIL-STD-498, for example: "The developer shall adhere to the SDP. With the exception of developer-internal scheduling and related staffing information, updates to the plan shall be subject to acquirer approval."

5.4.17 Step 17: Evaluating Software Development Plans. Step 17 in tailoring and applying MIL-STD-498 is evaluating the Software Development Plans that are submitted in proposals. The following guidelines apply.

a. Conformance to RFP requirements. A basic criterion in evaluating an SDP is whether the proposed approach fulfills all RFP requirements. Does it perform all tasks required in the Statement of Work; does it produce all deliverables required in the CDRL; does it follow all standards imposed on the project; does it meet all contractual constraints?

b. Feasibility of approach. A second criterion is the feasibility of the proposed approach. Based on the experience and knowledge of the evaluators, does the approach make sense; does it appear to be based on experience versus unproven theory; does it appear to attack the problem in a reasonable way; are risks identified and dealt with?

c. Feasibility of schedules. A third criterion is the feasibility of the schedules proposed. Based on the experience and knowledge of the evaluators, do the schedules make sense; do they allow enough time to do the work; do the scheduled activities and subactivities relate to one another in reasonable ways?

d. Adequacy of visibility. A fourth criterion is the adequacy of visibility provided by the bidder's approach. Key ways of achieving visibility are listed in Figure 57. Does the SDP lay out a combination of these approaches that will provide the acquirer adequate visibility into the evolving software?

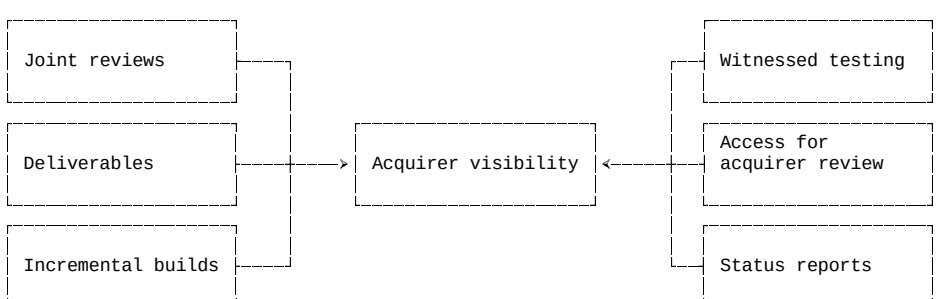


FIGURE 57. Ways of achieving visibility into the evolving software.

e. Concern for supportability. A fifth criterion is concern for supportability. Is the approach consistent with the support concept in the RFP? Is adequate concern shown for providing the support agency, if different from the developer, the materials and training needed to support the software? If the developer will perform the support, are adequate software products being prepared in a manner usable for this future support?

5.4.18 Step 18: Monitoring contract performance. Step 18 in tailoring and applying MIL-STD-498 is monitoring the resulting contract. Figure 57, on the preceding page, identifies some of the methods available to perform this monitoring. The following guidelines apply.

a. Joint technical and management reviews. A key method for achieving visibility is joint technical and management reviews. Joint technical reviews are attended by persons with technical knowledge of the software products to be reviewed. They focus on the technical status of in-process and final software products. Joint management reviews, generally held less frequently, are attended by persons with authority to make cost and schedule decisions. These reviews resolve issues that could not be resolved at the joint technical reviews. To make the most of both types of reviews, the acquirer should assign knowledgeable people, prepare in advance, and ensure that the reviews retain MIL-STD-498's intent of being useful discussions of real work products. Figure 58 illustrates the benefits of this approach. As in all acquirer-developer meetings, care should be taken to avoid agreements or direction that exceed or otherwise violate the contract.

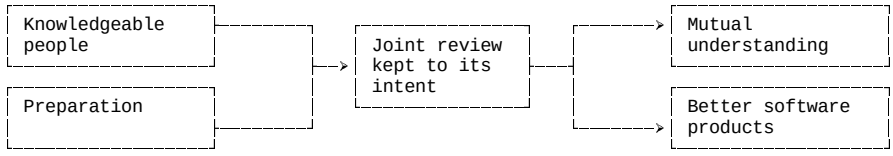


FIGURE 58. Ingredients for successful joint reviews.

b. Evaluating deliverables. A second method for achieving visibility is evaluating software products designated as deliverables. To make the most of these evaluations, assign knowledgeable people and provide feedback as quickly as possible.

c. Evaluating incremental builds. Incremental builds may take the form of early demonstrations, prototypes, or partial capabilities. Such builds can provide better visibility into the evolving software than written descriptions. To make the most of these evaluations, get the user involved, provide feedback as quickly as possible, and establish clear guidelines about the handling of "nifty" but out-of-scope suggestions that can derail a project.

d. Witnessing qualification testing. Witnessing qualification testing can provide a "warm feeling" about the software, but occurs late in the project. Do not rely on this method alone. When used, assign knowledgeable people and evaluate the test cases and procedures in advance to ensure that the right tests are being performed.

e. Access for acquirer review. MIL-STD-498 guarantees the acquirer access to the developer and subcontractor facilities to review software products and activities required by the contract. Be sure to arrange such access in advance to minimize disruption to the work being performed.

f. Status reports. Regular status reports are a standard contract monitoring technique. On a MIL-STD-498 project, these status reports might cover cost and schedule status, recent accomplishments, current values for and interpretation of software management indicators, identification of current and upcoming risks and associated risk management approaches, and identification of current and anticipated issues and problems. These reports can provide important visibility into the project.

g. Formal reviews and audits. MIL-STD-498 requires no formal reviews or audits. Such reviews and audits can be used to supplement MIL-STD-498's joint technical and management reviews if so desired. If used, they should focus on key issues raised at the joint technical and management reviews rather than repeating the detail covered there.

6. NOTES

This guidebook contains no notes.

This page is provided for notes that you may wish to make in using this guidebook.

APPENDIX A
LIST OF ACRONYMS

This appendix provides a list of acronyms used in this guidebook, with their associated meanings.

AIS	Automated Information Systems
CASE	Computer-Aided Software Engineering
CDRL	Contract Data Requirements List
CLIN	Contract Line Item Number
COM	Computer Operation Manual
CPM	Computer Programming Manual
CRM	Computer Resource Management
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DBDD	Database Design Description
DBMS	Database Management System
DepSO	Departmental Standardization Office
DFARS	Defense FAR Supplement
DID	Data Item Description
DoD	Department of Defense
DoDISS	Department of Defense Index of Specifications and Standards
EPROM	Erasable Programmable Read Only Memory
FAR	Federal Acquisition Regulation
FSM	Firmware Support Manual
HWCI	Hardware Configuration Item
HWG	Harmonization Working Group
IDD	Interface Design Description
IRS	Interface Requirements Specification
IV&V	Independent Verification and Validation

JLC	Joint Logistic Commanders
JPCG	Joint Policy Coordinating Group
MCCR	Mission Critical Computer Resources
NS	National Security
NSA	National Security Agency
OCD	Operational Concept Description
PROM	Programmable Read Only Memory
RFP	Request for Proposal
ROM	Read Only Memory
SCOM	Software Center Operator Manual
SDD	Software Design Description
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SIE	Standards Improvement Executive
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

APPENDIX B
MAPPING BETWEEN DOD-STD-2168 AND MIL-STD-498

Figure 59 presents a mapping between the requirements of DOD-STD-2168 and those of MIL-STD-498. It is intended to show that all key provisions of DOD-STD-2168 have been incorporated into MIL-STD-498.

DOD-STD-2168 Paragraph	MIL-STD-498 Paragraph(s)
4.1 Objective of the software quality program	MIL-STD-498 does not include 2168's statement of objective, but does contain the requirements necessary to meet that objective.
4.2 Responsibility for the software quality program	5.16.3 specifies 498's requirements on who may perform and who may be responsible for software quality assurance; the Software Development Plan (SDP) DID, which covers planning for all 498 activities, including SQA, requires the developer's approach to meeting this requirement.
4.3 Documentation for the software quality program	5.1.1 requires the developer to perform software development planning, covering all applicable items in the SDP DID. The SDP DID calls for the planning associated with all activities in MIL-STD-498, including that for SQA; as in 2168, this planning can reference documented SQA procedures; all aspects of the SDP are subject to acquirer approval.
4.4 Software quality program planning	5.1.1 requires the developer to perform software development planning, covering all applicable items in the SDP DID. The SDP DID calls for the planning associated with all activities in MIL-STD-498, including that for SQA; all software products, including the SDP, are required by 5.14 to be placed under developer configuration control.
4.5 Software quality program implementation	5.1.6 requires the developer to follow the Software Development Plan, which includes SQA planning; the overall structure and wording of 498 are designed to achieve the integration of SQA activities required by this 2168 paragraph.
4.6 Software quality evaluations	5.16.1 specifies the SQA evaluations required by 498. A key element of MIL-STD-498's approach is removing the perceived overlap between Software Product Evaluation and Software Quality Assurance. MIL-STD-498 does this by stating that SQA of software products consists of assuring that each required product exists and has undergone software product evaluations, testing, and corrective action, as required. This statement is meant to preclude overlapping evaluations.
4.7 Software quality records	5.16.2 requires the developer to record the results of SQA activities and to prepare problem/change reports as appropriate for submission to the corrective action system; the SDP DID calls for the developer to describe planned contents. There are no minimum contents specified.
4.8 Software corrective action	5.16.2 requires the developer to record the results of SQA activities and to prepare problem/change reports as appropriate for submission to the corrective action system; 5.16.1.a covers evaluation of all required activities, including the corrective action system; 5.17.2.b requires the developer to track the status of problems and resolution.

FIGURE 59. Mapping between DOD-STD-2168 and MIL-STD-498.

DOD-STD-2168 Paragraph	MIL-STD-498 Paragraph(s)
4.9 Certification	5.16.2 requires the developer to record the results of SQA activities and to prepare problem/change reports as appropriate for submission to the corrective action system; these reports, together with the corrective action records, software test reports, and software product evaluation records required by 498, can constitute the required evidence.
4.10 Management review of the software quality program	5.1.6 requires management review of the software development process at intervals specified in the SDP to assure that it complies with the contract and adheres to the plans; this requirement includes management review of all MIL-STD-498 activities, including SQA.
4.11 Access for contracting agency review	4.2.7 requires the developer to provide access to the acquirer or its authorized representative to developer and subcontractor facilities for review of software products and activities required by the contract.
5.1 Evaluation of software	5.15 requires all software products in 498 to undergo software product evaluations; 5.16.1.b requires the developer to assure that all software products, including the software itself, exist and have undergone software product evaluation, testing, and corrective action as required by the standard and other contract provisions; the criteria in 2168 are covered in Appendix D, criteria for software product evaluations.
5.2 Evaluation of software documentation 5.2.1 Evaluation of software plans 5.2.2 Evaluation of other software documentation	5.15 requires all software products in 498 to undergo software product evaluations; 5.16.1.b requires the developer to assure that all software products, including the SDP and other software documentation, exist and have undergone software product evaluation, testing, and corrective action as required by the standard and other contract provisions; the criteria in 2168 are covered in Appendix D, criteria for software product evaluations; 498 uses the term "software products" rather than "documentation" to acknowledge representations of information other than traditional hard-copy documents.
5.3 Evaluation of the processes used in software development 5.3.1 Evaluation of software management 5.3.2 Evaluation of software engineering 5.3.3 Evaluation of software qualification 5.3.4 Evaluation of software configuration management 5.3.5 Evaluation of software corrective actions 5.3.6 Evaluation of documentation and media distribution 5.3.7 Evaluation of storage, handling, and delivery 5.3.8 Evaluation of other processes used in software development	5.16.1.a requires SQA evaluations of all activities required by the contract or described in the Software Development Plan; the standard does not attempt to enumerate these activities as is done in 2168; 498 uses the term "activities" rather than processes to be consistent with the terminology used in the rest of the standard, but the intent is the same.

FIGURE 59. Mapping between DOD-STD-2168 and MIL-STD-498 - *(continued)*.

DOD-STD-2168 Paragraph	MIL-STD-498 Paragraph(s)
5.4 Evaluation of the software development library	5.2.3 requires the developer to establish, control, and maintain a software development library; 5.16.1.a requires evaluation of this activity to assure that it adheres to the Software Development Plan and is being performed in accordance with 498 and with other contract provisions; it was deemed inappropriate for the SQA section to impose additional requirements on the library.
5.5 Evaluation of non-developmental software	5.15 requires all software products in 498 (including non-developmental) to undergo software product evaluations; 5.14 requires all software products to be placed under developer configuration management; Appendix B interprets these requirements for reusable software (which includes non-developmental software); 4.2.3.1 requires reusable software that is incorporated into software products to comply with the data rights in the contract; 5.16.1 requires assurance that these requirements have been met.
5.6 Evaluation of non-deliverable software	5.2.5 requires the developer to ensure that non-deliverable software performs its intended functions; 5.14 requires developer configuration management of software products developed and used under the contract (including non-deliverables); 5.16.1 requires assurance that these requirements have been met.
5.7 Evaluation of deliverable elements of the software engineering and test environments	5.2.1 and 5.2.2 require that the environments be controlled and that the developer ensure that each element of the environment performs its intended functions; 5.1.5 and the Software Transition Plan DID require disclosure of data rights for deliverable items; all deliverable items must comply with contract data rights provisions without special mention; 5.16.1 requires assurance these requirements have been met.
5.8 Evaluation of subcontractor management	5.19.4 requires the developer to include in subcontracts all contractual requirements necessary to ensure that prime contract requirements have been met; 5.15 requires all software products in 498 (including subcontracted ones) to undergo software product evaluations; 4.2.7 requires the developer to provide the acquirer or its authorized representative access to developer and subcontractor facilities for review of software products and activities required by the contract; 5.16.1 requires assurance that these requirements have been met.
5.9 Evaluations associated with acceptance inspection and preparation for delivery	5.15 requires final evaluation of deliverable software products before delivery; 5.17.2 requires a closed-loop corrective action system; 5.16.1 requires assurance that these requirements have been met; together these requirements ensure that deliverable products are available, ready for acquirer inspection, and incorporate all acquirer-approved changes scheduled for inclusion.
5.10 Participation in formal reviews and audits	5.18 calls for joint technical reviews and joint management reviews in place of formal reviews; however, if the contract calls for formal reviews, 5.16.1 requires assurance that all contractual requirements and all software development planning associated with those requirements have been complied with. 498 itself contains no requirements regarding the role of the developer at or following formal reviews.

FIGURE 59. Mapping between DOD-STD-2168 and MIL-STD-498 - *(continued)*.

APPENDIX C
SAMPLE BUILD PLANNING AND TAILORING WORKSHEETS

This appendix provides sample worksheets for planning a MIL-STD-498 project and tailoring MIL-STD-498 and its Data Item Descriptions (DIDs). Included are a worksheet for the standard and for one of the DIDs. These worksheets are intended to be used as examples for setting up on-line or hard-copy worksheets suited to each project.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build			
		1	2	3	4
System:					
Type of SW:					
Prepared by:					
Date:					
1. Identify at right the objectives of each build 2. Indicate below which MIL-STD-498 requirements apply during the development of each build. Add clarifying notes as needed. (The requirement summaries are by nature incomplete; see the standard itself for the full wording of each requirement)					
Para	Requirement Summary				
4	General requirements				
4.1	Establish a software development process (EXAMPLE)	Yes: Establish a general process for the project and specifics for Build 1	Yes: Refine general process; establish specifics for Build 2	Yes: Refine general process; establish specifics for Build 3	Yes: Establish specifics for Build 4
4.1	Establish a software development process				
4.2	General requirements for software development				
4.2.1	Use systematic, documented methods for software development activities				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
4.2.2	Develop and apply standards for representing requirements, design, code, and test info				
4.2.3	Reusable software products				
4.2.3.1* (See end of table)	Identify and evaluate reusable software products; use those that meet established criteria				
4.2.3.2	Identify and analyze opportunities for developing software products for reuse				
4.2.4	Handling of critical requirements				
4.2.4.1	Identify safety-critical SW; develop and follow a strategy for assuring that software				
4.2.4.2	Identify security-critical SW; develop and follow a strategy for assuring that software				
4.2.4.3	Identify privacy-critical SW; develop and follow a strategy for assuring that software				
4.2.4.4* (See end of table)	Identify other critical SW; develop and follow a strategy for assuring that software				
4.2.5* (See end of table)	Analyze and fulfill contract requirements on computer hardware resource utilization				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued).*

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
4.2.6	Record rationale for key decisions, for use by the support agency				
4.2.7	Provide the acquirer access to developer and subcontractor facilities				
5	Detailed requirements				
5.1	Project planning and oversight				
5.1.1	Develop and record plans for conducting the software development effort				
5.1.2	Develop and record plans for conducting CSCI qualification testing				
5.1.3	Participate in developing and recording plans for system qualification testing				
5.1.4* (See end of table)	Develop and record plans for performing SW installation and training at user sites				
5.1.5* (See end of table)	Develop and record plans for transitioning deliverable SW to the support agency				
5.1.6	Follow approved plans; perform management reviews; obtain approval for updates				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.2	Establishing a software development environment				
5.2.1* (See end of table)	Establish, control, and maintain a software engineering environment				
5.2.2	Establish, control, and maintain a software test environment				
5.2.3	Establish, control, and maintain a software development library				
5.2.4	Establish, control, and maintain software development files				
5.2.5	Use non-deliverable SW only if operation/support of deliv-erable SW do not depend on it				
5.3	System requirements analysis				
5.3.1* (See end of table)	Participate in analyzing user input provided by the acquirer				
5.3.2	Participate in defining and recording the operational concept for the system				
5.3.3	Participate in defining and recording the system requirements				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	2	3	4
5.4	System design				
5.4.1	Participate in defining and recording system-wide design decisions				
5.4.2	Participate in defining and recording the architectural design of the system				
5.5	Define and record the software requirements to be met by each CSCI				
5.6	Software design				
5.6.1	Define and record CSCI-wide design decisions				
5.6.2	Define and record the architectural design of each CSCI				
5.6.3	Define and record the detailed design of each software unit in the CSCIs				
5.7	Software implementation and unit testing				
5.7.1* (See end of table)	Develop and record software; use acquirer approved language for deliverable SW				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.7.2	Establish test cases, test procedures, and test data for unit testing				
5.7.3	Perform unit testing in accordance with the test cases and procedures				
5.7.4	Make all necessary revisions; retest as necessary; update SDFs and other SW products				
5.7.5	Analyze and record the results of unit testing				
5.8	Unit integration and testing				
5.8.1	Establish test cases, test procedures, and test data for unit integration and testing				
5.8.2	Perform unit integration and testing in accordance with the test cases and procedures				
5.8.3	Make all necessary revisions; retest as necessary; update SDFs and other SW products				
5.8.4	Analyze and record the results of unit integration and testing				
5.9	CSCI qualification testing				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.91	Those responsible shall not have done detailed design or implementation of the CSCI				
5.9.2	Include testing on the target computer system or an approved alternative				
5.9.3	Define and record test cases, test procedures, and test data for CSCI qualification testing				
5.9.4* (See end of table)	If the CSCI qualification testing is to be witnessed, dry run the CSCI test cases				
5.9.5	Perform CSCI qualification testing in accordance with the test cases and procedures				
5.9.6	Make all necessary revisions; retest as necessary; update SDFs and other SW products				
5.9.7	Analyze and record the results of CSCI qualification testing				
5.10	CSCI/HWCI integration and testing				
5.10.1	Establish test cases, test procedures, and test data for CSCI/HWCI integ and testing				
5.10.2	Perform CSCI/HWCI integ and testing in accordance with the test cases and procedures				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.10.3	Make all necessary revisions; retest as necessary; update SDFs and other SW products				
5.10.4	Analyze and record the results of CSCI/HWCI integration and testing				
5.11	System qualification testing				
5.11.1	Those responsible shall not have done detailed design or implementation of the SW				
5.11.2	Include testing on the target computer system or an approved alternative				
5.11.3	Participate in defining and recording test cases, test procedures, and test data				
5.11.4* (See end of table)	If the testing is to be witnessed, participate in dry run of the system test cases				
5.11.5	Participate in performing system qual testing in accordance with cases/procedures				
5.11.6	Make all necessary revisions; participate in retest; update SDFs and other SW products				
5.11.7	Participate in analyzing and recording the results of system qualification testing				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued).*

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.12	Preparing for software use				
5.12.1* (See end of table)	Prepare the executable software for each user site				
5.12.2	Prepare software version descriptions for each user site				
5.12.3	Preparing user manuals				
5.12.3.1	Identify and record information needed by hands-on users of the software				
5.12.3.2	Identify and record information needed by software users relying on computer operators				
5.12.3.3	Identify and record information needed by computer operators running the software				
5.12.3.4	Identify and record information needed for hands-on operation of the computer				
5.12.4* (See end of table)	Install and check out the software at user sites; provide training and other assistance				
5.13	Preparing for software transition				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued).*

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.13.1* (See end of table)	Prepare the executable software for transition to the support site				
5.13.2* (See end of table)	Prepare the source files for transition to the support site				
5.13.3	Prepare software version descriptions for SW transitioned to the support site				
5.13.4	Update the CSCI design descriptions and prepare other info needed for SW support				
5.13.5	Update the system design description to match the "as built" system				
5.13.6	Preparing support manuals				
5.13.6.1	Identify and record information needed to program the computer(s)				
5.13.6.2	Identify and record information needed to program the firmware devices				
5.13.7* (See end of table)	Install, check out, and demo the SW at the support site; provide training, other aid				
5.14	Software configuration management				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.14.1	Identify each entity (file, document, etc.) to be placed under configuration control				
5.14.2* (See end of table)	Establish and implement procedures for controlling each entity and changes to it				
5.14.3	Prepare and maintain records of the status of entities and proposed changes to them				
5.14.4* (See end of table)	Support acquirer-performed configuration audits as specified in the contract				
5.14.5	Establish and implement procedures for packaging, storage, handling, and delivery				
5.15	Software product evaluation				
5.15.1	Perform in-process and final evaluations of software products				
5.15.2	Prepare and maintain records of each software product evaluation				
5.15.3	Those responsible for a given evaluation shall not have developed the SW product				
5.16	Software quality assurance				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.16.1	Conduct on-going evaluations of SW products and activities to assure they adhere to plans and comply with the contract				
5.16.2	Prepare and maintain records of the SQA evaluations				
5.16.3	Those responsible shall not have developed the product or performed the activity (plus other conditions)				
5.17	Corrective action				
5.17.1	Prepare problem/change reports for problems in SW products and activities				
5.17.2	Implement a corrective action system for problems in SW products and activities				
5.18	Joint technical and management reviews				
5.18.1	Plan and take part in joint technical reviews of in-process & final software products				
5.18.2	Plan and take part in joint management reviews to resolve issues				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued).*

Build Planning/Tailoring Worksheet For MIL-STD-498		Build 1	Build 2	Build 3	Build 4
5.19	Other activities				
5.19.1	Perform risk management throughout the software development process				
5.19.2	Identify, define, and apply software management indicators				
5.19.3* (See end of table)	Meet the security and privacy requirements in the contract				
5.19.4	Include in subcontracts all requirements needed to ensure software products will meet the prime contract				
5.19.5* (See end of table)	Interface with software indepen-dent verification and validation agents				
5.19.6* (See end of table)	Coordinate with associate developers, working groups, and interface groups				
5.19.7	Periodically assess the processes used; identify and propose improvements				
App. B	Interpreting MIL-STD-498 for incorporation of reusable software products				

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued).*

Build Planning/Tailoring Worksheet For MIL-STD-498		Build			
		1	2	3	4
B.3	Specify criteria for evaluating reusable SW products; include meeting requirements, cost effectiveness over system life				
B.4	Apply Figure 3 in interpreting MIL-STD-498 requirements for reusable software products				
App. C	Category and priority classifications for problem reporting				
C.3	Assign problems to the categories specified or to approved alternatives				
C.4	Assign problems to the priorities specified or to approved alternatives				
App. D	Software product evaluations				
D.3	Perform SW product evaluations using the criteria in Fig.6 or approved alternatives				

* These requirements depend upon additional contract provisions. Figure 49 identifies the information to be provided in the contract.

FIGURE 60. Sample build planning and tailoring worksheet for MIL-STD-498 - *(continued)*.

Build Planning/Tailoring Worksheet For SOFTWARE REQUIREMENTS SPECIFICATION (SRS) DID		Build 1	2	3	4
System:					
What software:					
Prepared by:					
Date:					
1. Identify at right the objectives of each build as they pertain to this DID. 2. Indicate below which DID paragraphs apply during the development of each build. Add clarifying notes as needed. Only paragraph titles are given; see the DID for full wording of each paragraph.					
Para*	Paragraph Title				
1.	Scope				
1.1	Identification				
1.2	System overview				
1.3	Document overview				
2.	Referenced documents				

*Note: Append 10.2. before each of these numbers when referring to the DID paragraphs.

FIGURE 61. Sample build planning and tailoring worksheet for a DID.

Build Planning/Tailoring Worksheet for the SOFTWARE REQUIREMENTS SPECIFICATION (SRS) DID		Build			
		1	2	3	4
3.	Requirements				
3.1	Required states and modes				
3.2	CSCI capability requirements				
3.2.x	(CSCI capability)				
3.3	CSCI external interface requirements				
3.3.1	Interface identification and diagrams				
3.3.x	(Project-unique identifier of interface)				
3.4	CSCI internal interface requirements				
3.5	CSCI internal data requirements				
3.6	Adaptation requirements				

FIGURE 61. Sample build planning and tailoring worksheet for a DID - *(continued)*.

Build Planning/Tailoring Worksheet for the SOFTWARE REQUIREMENTS SPECIFICATION (SRS) DID		Build 1	2	3	4
3.7	Safety requirements				
3.8	Security and privacy requirements				
3.9	CSCI environment requirements				
3.10	Computer resource requirements				
3.10.1	Computer hardware requirements				
3.10.2	Computer hardware resource utilization requirements				
3.10.3	Computer software requirements				
3.10.4	Computer communications requirements				
3.11	Software quality factors				
3.12	Design and implementation constraints				

FIGURE 61. Sample build planning and tailoring worksheet for a DID - *(continued)*.

Build Planning/Tailoring Worksheet for the SOFTWARE REQUIREMENTS SPECIFICATION (SRS) DID		Build			
		1	2	3	4
3.13	Personnel-related requirements				
3.14	Training-related requirements				
3.15	Logistics-related requirements				
3.16	Other requirements				
3.17	Packaging requirements				
3.18	Precedence and criticality of requirements				
4.	Qualification provisions				
5.	Requirements traceability				
6.	Notes				
A.	Appendixes				

FIGURE 61. Sample build planning and tailoring worksheet for a DID - *(continued)*.