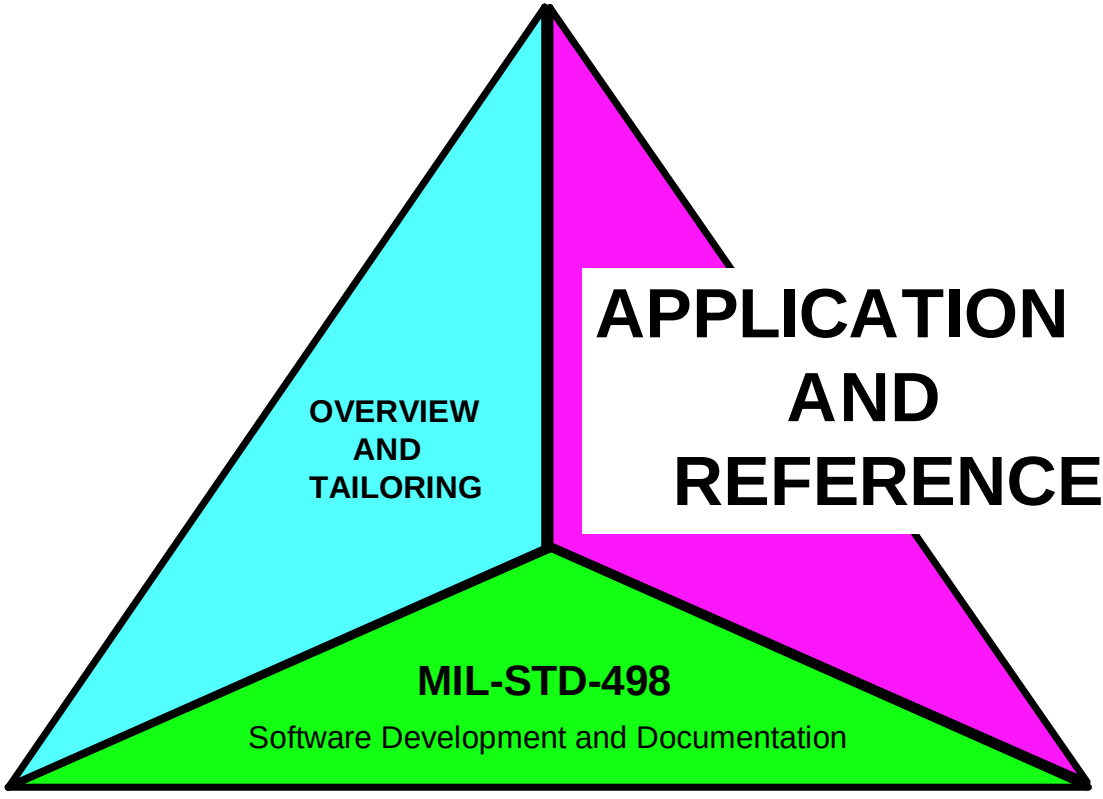
MIL-STD-498

Application and Reference

GUIDEBOOK



31 JANUARY 1996

Joint Logistics Commanders
Joint Policy Coordinating Group
on Computer Resources Management

This page is intentionally blank.

CONTENTS

APPENDIXES

MIL-STD-498 REFERENCES FIGURES

Figure Page

DEVELOPER'S KEY ACTIVITIES FIGURES

Figure Page

OTHER FIGURES

Figure Page

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 second 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 guidebook 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 Application and Reference 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
Victor Charles, ARDEC
Perry DeWeese, CODSIA
Robert Gagnon, OASD (ES)
Susan Gardner, FAA
Joseph Gianuzzi, SEI
Dung Minh Ha, MCTSSA
Robert Hegland, USAISSC
Priscilla Hopkins, DoD IG
Alan Huguley, DMA
Tim Janes, MoD, UK
Capt. Jonathan Liles, HQ AFMC
Donna McCloud, HQ DLA
LTC. Dixie McNeme, USA
Nancy Meier, NSA
Fred Moxley, DISA
Randy Paul, CIA
Glenn Plonk, NSA
Paul Shebalin, DSMC
Linda Sheets, AFMC
Mary Synder, USN

Guidebook Developers:

Myrna Olson
Stuart Campbell
Katherine Dean
Lynne Buss Ketchie
Dorothy Kuckuck
Al Peschel
Bill Stuart

This page is intentionally blank.

1. SCOPE

1.1 Purpose. The purpose of this guidebook is to:

a. Provide guidance to an acquirer in the application of MIL-STD-498 in a contractual relationship.

b. Provide ongoing guidance for using MIL-STD-498 over the life of a project, both to evaluate project progress and perform in-house development.

c. Provide a detailed reference guide covering key topics associated with the acquisition and development of software. This reference guide provides:

1) A summary of MIL-STD-498's requirements on key topics for easy reference

2) A description of acquirer responsibilities and considerations regarding the topic

3) A list of questions an acquirer might consider in project planning or in monitoring project risks to ensure objectives and goals are met within cost and schedule constraints

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 with supporting appendixes.

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 MIL-STD-498 and this guidebook.

d. Section 4 describes and gives guidance on the intended application of the standard by an acquirer in a contractual relationship.

e. Section 5 provides a detailed reference guide covering key topics associated with the application and development of software. The topics are arranged in alphabetical order, with page footers containing the topic names, to assist the reader in locating information. Each section 5 topic description is divided into four subparagraphs:

   1) Paragraph 5.x.1 summarizes the key topic-related requirements in the standard. When specific MIL-STD-498 references exist, this section contains a figure providing an index of relevant references to the topic. Within this index, MIL-STD-498 requirements are indicated with normal text; explanatory or non-mandatory references are indicated with *italicized text*. When applicable, this section contains a second figure listing key developer activities related to the topic.

   2) Paragraph 5.x.2 discusses acquirer responsibilities and provides some issues for consideration, including lessons learned regarding the topic.

   3) Paragraph 5.x.3 lists topic-related questions. The questions are intended to provide acquirers and developers with "things to think about" over the course of the project. These questions are candidates for discussion at joint reviews and may help surface and resolve problematic issues.

   4) Paragraph 5.x.4 contains a list of topics found elsewhere in this guidebook that are related to the topic being discussed.

f. Appendix A provides a list of acronyms used in this guidebook.

g. Appendix B provides a cross-reference between topics and sources of related information.

h. Appendix C provides a list of useful internet addresses.

i. Appendix D contains tables of contents, one for each of MIL-STD-498's twenty-two DIDs.

2. APPLICABLE DOCUMENTS

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 form a part of this document to the extent specified herein. 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

DEPARTMENT OF DEFENSE STANDARDS

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

* Superseded by MIL-STD-498

** Canceled 20 March 1995

Unless otherwise indicated, copies of the above 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. The following other government documents, drawings, and publications form a part of this document to the extent specified herein.

DoDI 5000.2	-	Defense Acquisition Management Policies and Procedures
DoDI 8120.2	-	Automated Information System Life-Cycle Management Process, Review, and Milestone Approval Procedures
DFARS	-	Defense FAR Supplement
DI-MGMT-80614	-	Data Item Description for Project Folders
DI-MGMT-81453	-	Data Item Description for Data Accession List

DD Form 250	-	Material Inspection and Receiving Report
DD Form 1423	-	Contract Data Requirements List
FAR	-	Federal Acquisition Regulation
Guidebook	-	MIL-STD-498 Overview & Tailoring Guidebook

2.2 Non-government publications.

INTERNATIONAL STANDARDS

ISO/IEC 12207	-	Information technology - Software life cycle processes

3. DEFINITIONS

The definitions below are the same as those in MIL-STD-498. They are provided here as an 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 pre-existing developer software products. Each use may include all or part of the software product and may involve its modification. This term can be applied to any software product (for example, requirements, architectures, etc.), not just to software itself.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.46 Support (of software). See software support.

3.47 Transition (of software). See software transition.

4. APPLICATION GUIDE

This section describes the intended use of MIL-STD-498 by an acquirer in a contractual (acquirer-developer) relationship and gives guidance to the acquirer for using the standard under current DoD policies and procedures.

4.1 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.

4.2 Supersession of previous standards. MIL-STD-498 supersedes DOD-STD-2167A Defense System Software Development, DOD-STD-7935A DoD Automated Information Systems (AIS) Documentation Standards, and DOD-STD-1703(NS) Software Product Standards. An acquirer should take care not to cite superseded, canceled, or conflicting standards or DIDs on a contract.

4.3 Use of MIL-STD-498. MIL-STD-498 may be used on any development that involves software. The standard may be used in a two-party agreement or voluntarily for in-house development. It may be used for the development of software that is part of a hardware-software system; for software in a software-only system; for part of a system, such as a subsystem, CSCI, database, functional capability, software unit, or any part of such software; for development of specific software products associated with a phase of development, such as software requirements or design; or for partial/preliminary software products, such as "builds" or system and software architectures. The standard may be used for any type of software (reusable software, software in the engineering and test environment, database, knowledge-based software, networked/client-server, communication, avionic, etc.) in any acquisition phase (Demonstration and Validation, Engineering and Manufacturing Development, etc.) and with any program strategy (Grand Design, Incremental, etc.). MIL-STD-498 may be applied to contractors, subcontractors, or government in-house agencies performing software development. The standard is invoked by citing it on a contract. It applies to each software product and to each type of software, including firmware, covered by the contract, regardless of storage medium, to the extent specified in the contract.

4.4 Use of MIL-STD-498's Data Item Descriptions. MIL-STD-498's planning and engineering Data Item Descriptions (DIDs) are an intrinsic part of the standard. Unless an activity has been tailored out, the information specified by the DID is intended to be defined and recorded. The contents of the DIDs apply regardless of whether the information resulting from those DIDs is intended to be delivered or not. The wording throughout MIL-STD-498 is designed to: (1) emphasize that the development and recording of planning and engineering information is an intrinsic part of the software development process, to be performed regardless of whether a deliverable is required; (2) use the DID as a checklist of items to be covered in the planning or engineering activity; and (3) permit representations other than traditional documents for recording the information (e.g., CASE tools).

4.5 Use of MIL-STD-498 to develop software in "builds". 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 limited capability, or versions containing full use of some capabilities but not others. Figures and notes in the standard inform the reader how to interpret the standard on projects involving multiple builds. Figure 10 of MIL-STD-498 shows how each major activity in the standard may be applied in one or more builds. Software builds may be developed in sync with system builds where each software build matches those of the system, or multiple software builds may be performed as part of a single build for the system. This is illustrated in Figure 1.

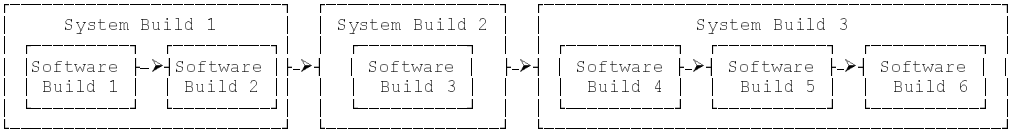


FIGURE 1. Example showing how system and software build strategies can differ.

4.6 Use of MIL-STD-498 with DoD's program strategies. DoDI 8120.2 describes three basic program strategies plus a generic strategy called "other," encompassing variations, combinations, and alternatives to the basic three. DoDI 5000.2 identifies similar strategies, called acquisition strategies. The three basic program strategies with their distinguishing characteristics are shown in Figure 2. Figures 9 through 11 of MIL-STD-498 show possible ways of applying the standard to these strategies.

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 2. Key features of three DoD program strategies.

a. Grand Design. The "Grand Design" strategy (not named in DoDI 5000.2 but treated as one strategy) is essentially a "once-through, do-each-step-once" strategy. Simplistically: determine user needs, define requirements, design the system, implement the system, test, fix, and deliver. This is the "traditional" strategy. It is particularly applicable to systems that are precedented (that is, where similar systems of this nature have been previously developed).

b. Incremental. The "Incremental" strategy (called "Preplanned Product Improvement" in DoDI 5000.2) determines user needs and defines the system requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the system is complete. This strategy is similar to the "Grand Design" with the primary distinction being that only part of the intended capabilities will be present in each increment. Sometimes these increments are called "builds." An Incremental strategy is particularly applicable when capabilities are required quickly and resources to complete the entire system must be funded over time.

c. Evolutionary. The "Evolutionary" strategy (called "Evolutionary" in both DoDI 5000.2 and DoDI 8120.2) 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, but are then refined in each succeeding build. This strategy differs from the others in that requirements are not totally defined at the start of the project, but "evolve" as both the developer and the user become more knowledgeable of the needs. This strategy is particularly applicable when technology is changing rapidly, where the threat is only partially identified, or when requirements are not clear to the users.

4.7 Use of MIL-STD-498 within system acquisition phases. DoDI 5000.2, Defense Acquisition Management Policies and Procedures, and DoDI 8120.2, Automated Information System (AIS) Life-Cycle Management (LCM) Process, Review and Milestone Approval Procedures, describe system acquisition phases (also known as system life cycle phases). MIL-STD-498 may be used within any phase when tailored appropriately for the objectives of each respective phase. Specific software product descriptions are often defined and recorded in certain phases of a program strategy. For example, when using the Grand Design program strategy, the operational concept description may be developed in Phase 0, system and subsystem requirements and design in Phase I, with full software implementation in Phase II. When the acquirer's justification for proceeding with development depends upon successful completion of objectives and criteria for a phase, MIL-STD-498's software products, such as plans, operational concept descriptions, design descriptions, and prototypes, can provide critical information identifying possible risks that could impact cost and schedule.

Milestone	Phase 0	Phase I	Phase II	Phase III	Phase IV
DoDI 5000.2 Phase Name	Concept Exploration and Definition	Demonstration and Validation	Engineering and Manufacturing Development	Production and Deployment	Operations and Support
DoDI 8120.2 Phase Name	Concept Exploration and Definition	Demonstration and Validation	Development	Production and Deployment	Operations and Support

FIGURE 3. DoDI 5000.2 and DoDI 8120.2 acquisition phases.

a. Concept Exploration and Definition. Most of MIL-STD-498's activities may not apply in this phase. The acquirer's efforts focus on identifying the concept for a new or modified system based on the new or modified aspects of user needs, threats, missions, objectives, environments, interfaces, personnel, or other factors. Little may be known about the eventual software product. However, it is possible that software will be developed to affirm the feasibility of the concepts proposed. Software developed as part of a feasibility study could eventually form the basis of the software developed for the final system. The acquirer should determine whether such software will be developed, and, if developed, whether that software will be reused later or thrown away. If the software is intended for reuse, the acquirer should apply MIL-STD-498 to avoid the all-too-frequent case where software that no one understands, but that "works" (most of the time), becomes permanently embedded in the system.

b. Demonstration and Validation. During this phase of system acquisition, the acquirer's efforts focus on determining the best product design. The acquirer should consider that a strong possibility exists that software will be developed during prototyping of critical components. The acquirer should determine whether such software will be developed, and, if so, whether any software developed is "throw-away" software or software that will be reused later. If the software is intended for reuse, the acquirer should apply MIL-STD-498.

c. Development/Engineering and Manufacturing Development. MIL-STD-498 is most frequently applied to the development of software during this phase of development. During this phase, the most promising design approach from the Demonstration and Validation Phase is translated into a stable, producible design. Among other considerations, the acquirer will apply MIL-STD-498 taking into consideration whether system level requirements/design activities have been completed and some software products developed. If software has already been developed that is reusable, Figure 3 of MIL-STD-498 explains how the activities of the standard apply to that software.

d. Production and Deployment. The activities of this phase are concerned with manufacturing the system, conducting tests to confirm and monitor performance and quality, and verifying the correction of deficiencies. For software, production means copying the executable software, not recoding the software from design documents. If enhancements are made to the software, MIL-STD-498 should be applied to those enhancements.

e. Operations and Support. The objectives of this phase are to ensure that the system continues to provide the capabilities required to meet the identified mission need and to identify shortcomings or deficiencies that must be corrected to improve performance. Studies have shown that 60-80% of life cycle costs occur in the operations and support phase. The activities of MIL-STD-498 can be applied in this phase when making enhancements and when correcting deficiencies. When applying the standard to making enhancements and correcting deficiencies in the software, the standard should be interpreted as for a new build or a reengineering effort. Software products, such as requirements, design, code, test procedures, plans, and manuals may already exist (or reside in databases or software development tools). Notes in the standard interpret the application to builds. Figure 12 of the standard shows one possible way of applying MIL-STD-498 to a reengineering project. For software developed under DOD-STD-2167A, DOD-STD-7935A, or DOD-STD-1703(NS), the acquirer should determine whether

DoD service or government agency policies exist that govern whether MIL-STD-498 or predecessor standards apply.

4.8 Application of MIL-STD-498 to development of systems. MIL-STD-498 can be applied to any system that contains software. The term "system," as used in the standard, may mean: (1) a hardware-software system (for example, a radar system) for which this standard covers only the software portion, or (2) a software-only system (for example, a payroll system) for which the standard governs overall development.

a. Hardware-software systems. MIL-STD-498 was developed for application to systems that contain both hardware and software, such as communications systems, radar systems, flight control systems, and engine control systems. For those systems, the standard covers only the software portion. The standard assumes that the contract (Statement of Work (SOW) or other agreement) covers the remaining portions. To accommodate this relationship, the standard tasks the software developer to "participate in" system development activities. When used in this manner, "participate in" means that the developer is to "*take part in, as described in the software development plan*," performing those activities.

b. Software only systems. MIL-STD-498 was developed for application to development of software-only systems, such as financial packages and networked software (possibly with the commercial computers on which that software will run). For those systems, the standard covers the entire system. When a system consists only of software, the term "participate" means that the developer is to "*be responsible for*" performing those activities.

4.9 Application of MIL-STD-498 to development of parts of systems. MIL-STD-498 can be applied to the development of part of a system, such as a subsystem, CSCI, database, functional capability, software unit, or any part(s) or aggregates of such software. If a development is based on alternative breakdowns of the system, the standard and its documentation applies to these alternatives as well.

a. Subsystem(s). If a system consists of subsystems, all requirements in the standard concerning systems apply to the subsystems as well.

b. CSCI(s). If a development is for a single CSCI or group of CSCIs, the requirements of the standard can be applied. The standard should be tailored to remove those system requirements that are not applicable to that development. The acquirer should determine whether it is desirable for the CSCI's developer to "participate in" system level activities (such as contributing to definition of the

system architecture and being present at integration testing activities to correct or "tweak" the software) or whether the CSCI is "stand-alone" and no "participation in" system level activities by the CSCI's developer is needed.

c. Database(s). The standard can be applied to the development of database management systems, to the development of software based on database management systems, or to the population of databases in such systems. Database development may be considered a system/subsystem development project, or a CSCI development project. A database may also be a software unit, or part of a software unit of a CSCI that is being developed. Application of the standard to databases, therefore, can vary from application of the standard as for a system/subsystem, to application as a CSCI, or as one part of a larger development project.

d. Capability(ies). The standard can be applied to the development of capability(ies) intended to cut across multiple subsystems, CSCIs, software units, or their aggregates. This case often exists when enhancements are required for existing systems. Enhancements may be considered as a software build or a reengineering project. Notes throughout the standard interpret the standard's application to builds. Figure 12 of MIL-STD-498 shows one possible way of applying the standard to a reengineering project.

e. Software unit(s) or part(s) or aggregates of such software. The standard can be applied to any software designated for development. It may be applied to a single software unit; to part(s) or aggregates of software units; or to only the design, or the implementation, or the testing of those units, parts, or aggregates. Selected activities may be applied to some of the software and not to others.

f. Storage media. The standard can be applied to software regardless of the media on which the software has been stored (disk, tape, card, ROM, RAM). MIL-STD-498 applies to software intended to be placed in firmware, such as a read-only type of storage medium (PROM) or other type of firmware.

4.10 Application of MIL-STD-498 to development of different types of software. A typical software development project might involve development of several different types of software. Types of software that might be found on a project are: deliverable and non-deliverable; precedented ("tried and true") and unprecedented ("cutting-edge technology"); pre-existing developer software, commercial-off-the-shelf (COTS) software, and acquirer-furnished software (Government-off-the-shelf (GOTS)); software in the software engineering and test environments; and operational, training, and simulation software. Software in a project may fit into one or more categories. For example, simulation software might be categorized as: (1) deliverable, (2) unprecedented and (3) simulation software. The different types of software on a project are often identified only as planning, tailoring, and project execution proceed. The acquirer and developer may need to clarify the tasks and software products applicable for each type of software over time. Questions of this nature can be raised and resolved during joint technical and management reviews. Notice that for purposes of using the standard, it is not necessary to make allowances based on whether the software will be newly developed, modified, reused without modification, etc. Appendix B of MIL-STD-498 interprets and tailors the standard to accommodate newly developed, modified, and reused software.

4.10.1 Application to non-deliverable and deliverable software. MIL-STD-498 makes a clear distinction between these two types of software. MIL-STD-498 states that without specific tailoring, the standard is understood to be invoked in its entirety for all deliverable software. In general, MIL-STD-498's requirements do not apply to non-deliverable software (for example project management software used on the project). However, the standard does have limited requirements for non-deliverable software used in the software engineering and test environments (see 4.10.4).

4.10.2 Application to unprecedented and precedented software. MIL-STD-498 makes no distinction between its application to software that is on the cutting-edge of technology (unprecedented), such as real-time satellite telemetry software to be coded in Ada, and software that is "tried and true" (precedented), such as a financial package to be coded in COBOL. MIL-STD-498 can be applied equally to either type. Cost, schedule, and performance risk for an unprecedented software development may be orders of magnitude greater than that for a precedented software development, but the activities that apply and products that are developed may be identical for both.

4.10.3 Application to reusable software. MIL-STD-498 defines requirements for incorporating reusable software products for use in fulfilling contractual requirements and for developing reusable software. The standard uses the term "reusable software" as an umbrella term that covers software products that are developed for one use but have other uses on other projects or multiple roles on a specific project. Some examples of reusable software

include: COTS software, acquirer-furnished software, software in reuse libraries, and pre-existing developer software. Each project use may include all or part of the reusable software product and may involve modifying the reusable software products. Figure 4 of this guidebook summarizes how MIL-STD-498's application to types of reusable software was determined.

APPLICATION TO REUSABLE SOFTWARE IN MIL-STD-498

MIL-STD-498 interprets each activity in the standard for existing, reusable software and specifies how the standard is to be interpreted for each application (see MIL-STD-498 Figure 3). Default application of the standard to reusable software requires determining into which category the reusable software falls, based on:

1) Performance record, i.e., positive versus no or poor performance record
2) Place in the architecture, i.e., CSCI versus software unit
3) How used, i.e., modified versus used "as is"

NOTE: No tailoring of the standard is required to accommodate using reusable software falling into these categories.

FIGURE 4. Application to reusable software in MIL-STD-498.

4.10.4 Application to software in the software engineering and test environments. MIL-STD-498 establishes default applications for two types of software. Reusable software is one such default application, described above. The other default is the standard's application to non-deliverable software in the software engineering and software test environments. These defaults are intended to tailor the standard (limit the extent of, or interpret, the application) for that type of software. Application to non-deliverable software in the software engineering and test environments is limited to: (1) establishing the non-deliverable software for the environment, (2) controlling the non-deliverable software, (3) maintaining the non-deliverable software, and (4) ensuring the non-deliverable software performs its intended functions. (The requirements to establish, control, maintain, and ensure apply to the non-deliverable hardware elements of the software development environment as well.) When the software development environment is required to be delivered, the requirements of the standard are understood to apply in their entirety to the deliverable software unless tailored. (Note that application to deliverable COTS software in the software development environment, as one form of reusable software is, by default, limited as described in Appendix B of MIL-STD-498.)

4.10.5 Application to operational, training, operating system, simulation, and other software. MIL-STD-498 can be applied to software used for performing any capability. It can be applied to operating systems, simulators, databases, training, financial, knowledge-based, artificial intelligence, weapon systems, command and control systems, communications, or avionics software, software in nuclear power plants, or any other software.

4.11 Ways of applying MIL-STD-498. As shown in Figure 5 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 5. Ways of applying MIL-STD-498.

a. Requirement in RFP and contract. One way of applying 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 apply 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 apply 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 applying the standard.

4.12 Players and agreements assumed by MIL-STD-498. Figure 6 illustrates the players and agreements assumed by MIL-STD-498. The paragraphs that follow explain each term.

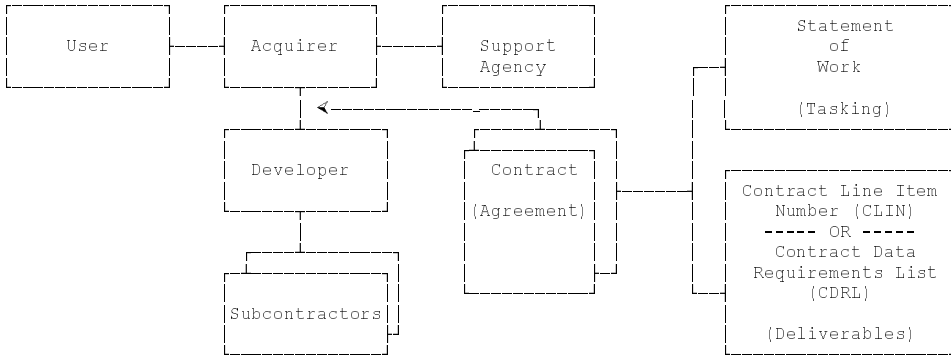


FIGURE 6. Players and agreements assumed by MIL-STD-498.

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, maintenance, 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 organization(s) 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.

4.13 Acquirer's role in MIL-STD-498. MIL-STD-498 is written to provide a contractual basis of agreement between an acquirer and a developer. It is written in contractual language to provide the wording required for a contract or other agreement. DoD policy requires the acquirer to tailor (that is, delete, modify, or add, if necessary) the requirements in the standard to meet the objectives for each type of software required. The acquirer should consider both the system and software objectives, as well as support and user requirements when performing this tailoring. The MIL-STD-498 Overview and Tailoring Guidebook gives additional information regarding tailoring.

4.14 Relationship to DoD acquisition policy and instructions. 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 acquisition. Key documents developed or undergoing change at that time were DoDI 5000.2 and DoDI 8120.2 as well as standards and handbooks on government reviews and audits, government configuration management, and work breakdown structures (WBSs). These changes are still ongoing. While the acquisition environment in which MIL-STD-498 is applied and the way the standard is applied (see 4.11) is subject to on-going change, the basic principles of software development (i.e., determine needs, define

requirements, design, implement, test, control, evaluate, etc.) specified in the standard apply regardless of current acquisition policy.

4.15 Relationship to other standards. MIL-STD-498 invokes no other standards. It takes the approach of covering all relevant topics, at least in summary form. Listed in MIL-STD-498's Figure 2 are related standards that the acquirer may wish to consider as alternatives or supplements to the requirements in MIL-STD-498. Included in the list 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 incomplete, inconsistent, or may conflict. In addition, many of the listed standards have been canceled, are in the process of being canceled, or are being converted to commercial or international standards, guidebooks, or other "guidance only" documents. The acquirer is responsible for reconciling any inconsistencies in the selected standards and for determining their status.

This page intentionally blank.