5. REFERENCE GUIDE

This section provides a reference guide for an acquirer on selected topics found in MIL-STD-498, Software Development and Documentation. This section assumes a basic knowledge of system and software acquisition and software development. This reference does not attempt to provide a comprehensive coverage of these topics. It is intended to assist the acquirer in understanding the scope of the requirements for selected topics found in the standard, clarify the intention of the standard in regard to the topic, and describe the acquirer's responsibilities and considerations when applying the standard's requirements for the topic.

The topics are arranged in alphabetical order, with page footers containing the topic names, to assist the reader in locating information. Each 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.

Section 5 topics are intended as stand-alone references on an individual topic. To minimize repetitive definitions of often used acronyms, the most commonly used acronyms include:

CALS	Continuous Acquisition and Life-cycle Support
CDRL	Contract Data Requirements List
CLIN	Contract Line Item Number
COTS	Commercial-off-the-shelf
DFARS	Defense FAR Supplement
DID	Data Item Description
FAR	Federal Acquisition Regulation
NDI	Non-developmental Item
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SOW	Statement of Work

To aid users in correlating this MIL-STD-498 Application and Reference Guidebook's topics to relevant MIL-STD-498 activities and DIDs, two indexes are provided on the following pages.

- Figure 7 -- MIL-STD-498-to-Topic Index. Figure 7 provides an index from each MIL-STD-498 paragraph to applicable topics. Each activity in MIL-STD-498 is listed with its corresponding key related topics. Additional references to other related topics are provided in paragraph 5.x.4 of each relevant topic.

- Figure 8 -- Topic-to-MIL-STD-498 DIDs Index. Figure 8 provides an index between topics and the MIL-STD-498's DIDs. Each topic in this guidebook is listed along with any DID that provides requirements for information relevant to that topic.

MIL-STD-498 PARAGRAPH	KEY RELATED GUIDEBOOK TOPICS
1. SCOPE	**N/A**
1.1 Purpose	None
1.2 Application	**N/A**
1.2.1 Organizations and agreements	5.10 Contract 5.11 Contract data requirements list 5.25 In-house development 5.69 Statement of work
1.2.2 Contract-specific application	5.10 Contract 5.49 Software development environment 5.55 Software engineering environment 5.63 Software test environment
1.2.3 Tailoring	See MIL-STD-498 Overview and Tailoring Guidebook
1.2.4 Interpretation of selected terms	5.7 CASE tools 5.22 Documentation (Recording information) 5.40 Reengineering 5.43 Reusable software products 5.71 System/subsystem
1.3 Order of precedence	None
2. REFERENCED DOCUMENTS	**N/A**
3. DEFINITIONS	**N/A**
4. GENERAL REQUIREMENTS	**N/A**
4.1 Software development process	5.54 Software development process
4.2 General requirements for software development	**N/A**
4.2.1 Software development methods	5.52 Software development methods
4.2.2 Standards for software products	5.68 Standards for software products
4.2.3 Reusable software products (including 4.2.3.1, 4.2.3.2)	5.8 Commercial-off-the-shelf software products 5.18 Data rights 5.31 Licenses (Software) 5.43 Reusable software products
4.2.4 Handling of critical requirements (including 4.2.4.1 - 4.2.4.4)	5.14 Critical requirements 5.45 Safety 5.47 Security and privacy
4.2.5 Computer hardware resource utilization	5.9 Computer hardware resource utilization
4.2.6 Recording rationale	5.39 Rationale/key decisions
4.2.7 Access for acquirer review	5.2 Access for acquirer review 5.15 Data accession list 5.33 Oversight

FIGURE 7. MIL-STD-498-to-topic index.

MIL-STD-498 PARAGRAPH	KEY RELATED GUIDEBOOK TOPICS
5. DETAILED REQUIREMENTS	**N/A**
5.1 Project planning and oversight	**N/A**
5.1.1 Software development planning	5.53 Software development planning
5.1.2 CSCI test planning	5.38 Qualification testing 5.63 Software test environment
5.1.3 System test planning	5.38 Qualification testing 5.63 Software test environment
5.1.4 Software installation planning	5.27 Installation (User site(s)) 5.66 Software user manuals
5.1.5 Software transition planning	5.26 Installation (Support environment) 5.61 Software support 5.62 Software support manuals 5.64 Software transition
5.1.6 Following and updating plans	5.3 Approval by the acquirer 5.53 Software development planning
5.2 Establishing a software development environment	**N/A**
5.2.1 Software engineering environment	5.49 Software development environment 5.55 Software engineering environment
5.2.2 Software test environment	5.63 Software test environment
5.2.3 Software development library	5.51 Software development library
5.2.4 Software development files	5.50 Software development files
5.2.5 Non-deliverable software	5.49 Software development environment
5.3 System requirements analysis	**N/A**
5.3.1 Analysis of user input	5.32 Operational concept
5.3.2 Operational concept	5.32 Operational concept
5.3.3 System requirements	5.1 Acceptance by the acquirer 5.29 Interfaces 5.41 Requirements 5.71 System/subsystem 5.74 Traceability
5.4 System design	**N/A**
5.4.1 System-wide design decisions	5.5 Behavioral design 5.16 Database design 5.71 System/subsystem 5.72 System/subsystem-wide and CSCI-wide design
5.4.2 System architectural design	5.4 Architectural design 5.29 Interfaces 5.71 System/subsystem 5.74 Traceability

FIGURE 7. MIL-STD-498-to-topic index - *(continued)*.

MIL-STD-498 PARAGRAPH	KEY RELATED GUIDEBOOK TOPICS
5.5 Software requirements analysis	5.1 Acceptance by the acquirer 5.29 Interfaces 5.41 Requirements 5.74 Traceability
5.6 Software design	**N/A**
5.6.1 CSCI-wide design decisions	5.5 Behavioral design 5.16 Database design 5.72 System/subsystem-wide and CSCI-wide design
5.6.2 CSCI architectural design	5.4 Architectural design 5.29 Interfaces 5.74 Traceability
5.6.3 CSCI detailed design	5.16 Database design 5.20 Detailed design 5.29 Interfaces 5.74 Traceability
5.7 Software implementation and unit testing (including 5.7.1 - 5.7.5)	5.17 Databases 5.37 Programming languages 5.50 Software development files 5.56 Software implementation and unit testing 5.65 Software unit 5.67 Source files 5.73 Testing (Developer-internal)
5.8 Unit integration and testing (including 5.8.1 - 5.8.4)	5.50 Software development files 5.73 Testing (Developer-internal)
5.9 CSCI qualification testing (including 5.9.1 - 5.9.7)	5.38 Qualification testing 5.74 Traceability
5.10 CSCI/HWCI integration and testing (including 5.10.1 - 5.10.4)	5.50 Software development files 5.73 Testing (Developer-internal)
5.11 System qualification testing (including 5.11.1 - 5.11.7)	5.38 Qualification testing 5.74 Traceability
5.12 Preparing for software use	**N/A**
5.12.1 Preparing the executable software	5.23 Executable software
5.12.2 Preparing version descriptions for user sites	5.75 Version/revision/release
5.12.3 Preparing user manuals	5.66 Software user manuals
5.12.4 Installation at user sites	5.27 Installation (User site(s))

FIGURE 7. MIL-STD-498-to-topic index - *(continued)*.

MIL-STD-498 PARAGRAPH	KEY RELATED GUIDEBOOK TOPICS
5.13 Preparing for software transition	**N/A**
5.13.1 Preparing the executable software	5.23 Executable software
5.13.2 Preparing source files	5.67 Source files
5.13.3 Preparing version descriptions for the support site	5.75 Version/revision/release
5.13.4 Preparing the "as built" CSCI design and related information	5.64 Software transition
5.13.5 Updating the system design descriptions	5.64 Software transition
5.13.6 Preparing support manuals (including 5.13.6.1 - 5.13.6.2)	5.62 Software support manuals
5.13.7 Transition to the designated support site	5.26 Installation (Support environment) 5.61 Software support 5.64 Software transition
5.14 Software configuration management (including 5.14.1 - 5.14.5)	5.48 Software configuration management
5.15 Software product evaluation (including 5.15.1 - 5.15.3)	5.59 Software product evaluation
5.16 Software quality assurance (including 5.16.1 - 5.16.3)	5.60 Software quality assurance
5.17 Corrective action	**N/A**
5.17.1 Problem/change reports	5.35 Problem/change report
5.17.2 Corrective action system	5.12 Corrective action
5.18 Joint technical and management reviews (including 5.18.1 - 5.18.2)	5.30 Joint technical and management reviews
5.19 Other activities	**N/A**
5.19.1 Risk management	5.33 Oversight 5.44 Risk management
5.19.2 Software management indicators	5.58 Software management indicators
5.19.3 Security and privacy	5.47 Security and privacy
5.19.4 Subcontractor management	5.70 Subcontractor management
5.19.5 Interface with software IV&V agents	5.24 Independent verification and validation
5.19.6 Coordination with associate developers	None
5.19.7 Improvement of project processes	5.36 Process improvement

FIGURE 7. MIL-STD-498-to-topic index - *(continued)*.

MIL-STD-498 PARAGRAPH	KEY RELATED GUIDEBOOK TOPICS
6. NOTES	**N/A**
6.1 Intended use	None
6.2 Data requirements	5.11 Contract data requirements list
6.3 Relationship between standard and CDRL	5.11 Contract data requirements list
6.4 Delivery of tool contents	5.7 CASE tools 5.11 Contract data requirements list
6.5 Tailoring guidance	5.11 Contract data requirements list 5.42 Requirements of the standard 5.69 Statement of work
6.6 Cost/schedule reporting	5.13 Cost estimation 5.46 Schedules
6.7 Related standardization documents	5.69 Statement of work
6.8 Subject term (key word) listing	**N/A**
APPENDIX A LIST OF ACRONYMS	Appendix A - List of acronyms
APPENDIX B INTERPRETING MIL-STD-498 FOR INCORPORATION OF REUSABLE SOFTWARE PRODUCTS	5.43 Reusable software products
APPENDIX C CATEGORY AND PRIORITY CLASSIFICATIONS FOR PROBLEM REPORTING	5.34 Problem category and priority classification
APPENDIX D SOFTWARE PRODUCT EVALUATIONS	5.59 Software product evaluation
APPENDIX E CANDIDATE JOINT MANAGEMENT REVIEWS	5.30 Joint technical and management reviews
APPENDIX F CANDIDATE MANAGEMENT INDICATORS	5.58 Software management indicators
APPENDIX G GUIDANCE ON PROGRAM STRATEGIES, TAILORING, AND BUILD PLANNING	5.6 Builds 5.54 Software development process 5.57 Software life cycle processes Also see MIL-STD-498 Overview & Tailoring Guidebook
APPENDIX H GUIDANCE ON ORDERING DELIVERABLES	5.11 Contract data requirements list 5.21 Documentation (Preparing documents) 5.22 Documentation (Recording information) 5.46 Schedules
APPENDIX I CONVERSION GUIDE FROM DOD-STD-2167A AND DOD-STD-7935A	**N/A**

FIGURE 7. MIL-STD-498-to-topic index - *(continued)*.

Application and Reference Topic	MIL-STD-498 Data Item Descriptions (DIDs)																					
	COM	CPM	DBDD	FSM	IDD	IRS	OCD	SCOM	SDD	SDP	SIOM	SIP	SPS	SRS	SSDD	SSS	STD	STP	STR	STrP	SUM	SVD
5.1 Acceptance by the acquirer						•								•		•	•	•	•			
5.2 Access for acquirer review										•												
5.3 Approval by the acquirer										•												
5.4 Architectural design			•		•	•	•		•	•			•	•	•	•						
5.5 Behavioral design			•		•	•	•		•	•			•	•	•	•						
5.6 Builds									•	•					•							•
5.7 CASE tools	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.8 Commercial-off-the-shelf software products																						
5.9 Computer hardware resource utilization							•		•	•			•	•	•	•						
5.10 Contract	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.11 Contract data requirements list	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.12 Corrective action										•												•
5.13 Cost estimation																						
5.14 Critical requirements			•			•			•	•				•	•	•						
5.15 Data accession list																						
5.16 Database design			•		•	•			•	•			•	•	•	•						
5.17 Databases			•		•			•	•	•	•	•	•	•	•	•	•	•			•	
5.18 Data rights	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.19 Data standardization	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.20 Detailed design			•		•	•			•	•			•	•								
5.21 Documentation (Preparing documents)	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.22 Documentation (Recording information)	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.23 Executable software										•			•									•
5.24 Independent verification and validation										•												
5.25 In-house development										•												
5.26 Installation (Support environment)				•						•			•							•		•

FIGURE 8. Topic to MIL-STD-498 DIDs index.

Application and Reference Topic	MIL-STD-498 Data Item Descriptions (DIDs)																				
	COM	CPM	DBDD	FSM	IDD	IRS	OCD	SCOM	SDD	SDP	SIOM	SIP	SPS	SRS	SSDD	SSS	STD	STP	STR	STrP	SUM	SVD
5.27 Installation (User site(s))								•		•		•	•			•					•	•
5.28 Integrated product team																						
5.29 Interfaces			•		•	•			•	•			•	•	•	•	•	•				
5.30 Joint technical and management reviews										•												
5.31 Licenses (Software)																		•		•		
5.32 Operational concept							•			•												
5.33 Oversight						•				•			•	•		•						
5.34 Problem category and priority classifications										•												
5.35 Problem/change report										•									•			•
5.36 Process improvement										•												
5.37 Programming languages		•	•						•	•				•		•						
5.38 Qualification testing						•				•				•		•	•	•	•			
5.39 Rationale/key decisions	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.40 Reengineering									•	•					•							
5.41 Requirements			•		•	•	•		•	•			•	•	•	•	•	•				
5.42 Requirements of the standard	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.43 Reusable software products	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.44 Risk management										•												
5.45 Safety	•		•	•	•	•	•	•	•	•	•	•		•	•	•	•	•		•	•	•
5.46 Schedules										•		•						•	•	•		
5.47 Security and privacy	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
5.48 Software configuration management										•												•
5.49 Software development environment		•								•			•		•		•	•	•			•
5.50 Software development files										•							•	•	•			
5.51 Software development library										•												
5.52 Software development methods										•												

FIGURE 8. Topic to MIL-STD-498 DIDs index - *(continued)*.

Application and Reference Topic	MIL-STD-498 Data Item Descriptions (DIDs)																					
	COM	CPM	DBDD	FSM	IDD	IRS	OCD	SCOM	SDD	SDP	SIOM	SIP	SPS	SRS	SSDD	SSS	STD	STP	STR	STrP	SUM	SVD
5.53 Software development planning										•												
5.54 Software development process										•												
5.55 Software engineering environment		•								•									•			
5.56 Software implementation and unit testing										•			•									
5.57 Software life cycle processes										•												
5.58 Software management indicators										•												
5.59 Software product evaluation										•												
5.60 Software quality assurance										•												
5.61 Software support		•	•	•	•		•		•	•		•	•	•	•	•				•		•
5.62 Software support manuals		•		•						•			•							•		
5.63 Software test environment										•								•	•			
5.64 Software transition		•	•	•	•				•	•			•		•					•		•
5.65 Software unit			•		•				•	•			•	•								
5.66 Software user manuals	•							•		•	•	•		•		•				•	•	
5.67 Source files										•			•									•
5.68 Standards for software products			•		•				•	•			•		•							
5.69 Statement of work																						
5.70 Subcontractor management										•				•	•							
5.71 System/subsystem			•		•	•	•			•				•	•	•	•	•	•			
5.72 System/subsystem-wide and CSCI-wide design			•		•	•	•		•	•			•	•	•	•						
5.73 Testing (Developer-internal)										•												
5.74 Traceability			•		•	•			•	•			•	•	•	•	•	•				
5.75 Version/revision/ release	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

FIGURE 8. Topic to MIL-STD-498 DIDs index - *(continued)*.

5.1 Acceptance by the acquirer.

5.1.1 Requirements summary. MIL-STD-498 provides requirements regarding acquirer acceptance of software. The standard defines "acceptance" as "*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*." This definition is based on FAR § 46.101, April 1984. References to ***Acceptance by the acquirer*** in MIL-STD-498 are listed in Figure 9, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.1*, *3.30*, *E.3*
IRS	3
SRS	3
SSS	3

FIGURE 9. MIL-STD-498's references to ***Acceptance by the acquirer***.

The System/Subsystem Specification (SSS), the Interface Requirements Specification (IRS), and the Software Requirements Specification (SRS) DIDs use the phrase "conditions for acceptance" as criteria for determining the amount of detail to be used in defining requirements. A requirement in MIL-STD-498 is "*a characteristic that a system or CSCI must possess in order to be acceptable to the acquirer*." This means that whether the characteristic is a broad performance characteristic (e.g., the plane shall fly) or a specific one (e.g., the cursor shall blink three consecutive red WARNING messages prior to aborting), if the acquirer wants that characteristic enough to make it a condition of acceptance, it is a requirement. (See this guidebook's topic, *Requirements*, for more information.)

5.1.2 Acquirer's responsibilities and considerations. Acceptance, approval, and authentication are all contracting related government procurement terms. These terms are often loosely used as if they were interchangeable, but each has a distinct meaning in the FAR. MIL-STD-498 bases its definitions on the FAR and uses only two terms in the standard: "acceptance" and "approval." MIL-STD-498 does not use the term "authentication." Authentication is sometimes used to mean that the acquirer assumes responsibility for the correctness of a product. This concept is not part of MIL-STD-498.

The difference between "acceptance" and "approval" is the difference between taking ownership (acceptance, FAR § 46.101, April 1984) and giving the developer the affirmation that the acquirer believes that a product is "good enough" to be the basis for future work (approval). Approval is defined in the standard as "*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." (See this guidebook's topic, *Approval by the acquirer*, for more information on approval.)

Whether the "as built" system or software is acceptable is usually determined through qualification tests. Although sometimes referred to as acceptance tests, MIL-STD-498 uses the term "qualification" for these tests to avoid the implication that once testing of the requirements has taken place the acquirer owns (accepts) the system/software. Qualification tests are only one, albeit an important, condition of contractual acceptance.

MIL-STD-498 provides two levels of qualification tests: CSCI qualification tests and system/subsystem qualification tests. Both CSCI-level and system/subsystem-level qualification test plans and procedures are prepared in accordance with the Software Test Plan (STP) and the Software Test Description (STD) DIDs. The STD contains procedures traceable to the requirements found in the SSS, IRS, or SRS, depending upon the level of testing performed. (See this guidebook's topic, *Traceability*, for more information.) These test procedures and their resulting Software Test Reports (STRs) are input to the acquirer for acceptance of the software. An acquirer may want to "qualify" a CSCI, a group of CSCIs, a subsystem, etc., without accepting (taking ownership of) those products until the system has successfully passed required system-level qualification tests.

For some safety-critical, security-critical, or reliability-critical systems, such as systems with nuclear components, the acquirer may be required by policy or law to base acceptance decisions on the results of in-process IV&V and/or developer inspections, analyses, demonstrations and tests using specialized acquirer-defined acceptance criteria and processes. If needed, such specialized acceptance criteria and processes need to be defined in the SOW.

Although the contract and specifications establish the software "acceptance" criteria, an acquirer may find it necessary or expedient to accept software products that do not perform correctly or lack required capability. An acquirer may decide to accept software that contains errors, delivers less-than-required capability, or does not fully meet requirements in order to provide users with needed capabilities. Key considerations in accepting such products will be the risk of mission failure, the risk of system failure, the creation of unacceptable hazards, or the risk of compromised security, privacy, or other critical requirements. Providing deviations (allowing a developer to defer meeting requirements) and waivers (allowing a developer to omit meeting requirements) are two contractual methods to grant temporary or permanent relief from the requirements of the contract. The acquirer should check with DoD service or government agency contracting agents to determine whether other rules or FAR requirements

govern whether the software can or cannot be accepted "as is," whether warranties apply, and what provisions may be necessary to ensure that temporary relief (deviation) does not forfeit the acquirer's requirements or result in unfixed errors.

All deliverable software products are eventually accepted or rejected by the government through contractual means. To provide a stable basis for development, specifications are frequently accepted early in a project and used as contractual requirements (baseline) for future work.

Final contractual acceptance in DoD is usually accomplished through submission by the developer of a DD Form 250 with acceptance indicated by the signature of the acquirer or the acquirer's authorized representative. Figure 10 presents a sample form. Note that products can be accepted either by the acquirer or acquirer's designated representative at the source, such as the developer's site, or a specified destination, such as the support or user site. Who accepts, when they accept, and where acceptance takes place may be governed by DoD service or government agency policy or by other criteria, such as operations and test schedules, dollar thresholds, and personnel availability. The acquirer needs to understand the applicable acceptance provisions and have a process in place to implement those provisions.

Government policy regarding acquirer acceptance is stated in FAR § 46.102, April 1984, and includes making provision for inspection and other quality requirements in contracts. Such FAR provisions include: warranty clauses, when appropriate; incorporation of standards (such as MIL-STD-498) in solicitations and contracts; conducting acquirer quality assurance activities; and rejecting non-conforming products, if appropriate. Refer to FAR § 52.246-19 for the acceptance of systems that are warranted.

MATERIAL INSPECTION AND RECEIVING REPORT

Form approved OMB No.0704-0248

Public reporting burden for this collection of information is estimated to average 30 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining and the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0248), Washington, DC 20503.

PLEASE DO NOT RETURN YOUR COMPLETED FORM TO EITHER OF THESE ADDRESSES.

SEND THIS FORM IN ACCORDANCE WITH THE INSTRUCTIONS CONTAINED IN THE DFARS, APPENDIX F-401.

1. PROC. INSTRUMENT IDEN. (CONTRACT)	(ORDER)NO	6. INVOICE NO./DATE	7. PAGE	OF	8. ACCEPTANCE POINT

2. SHIPMENT NO.	3. DATE SHIPPED	4. B/L TCN	5. DISCOUNT TERMS

9. PRIME CONTRACTOR CODE		10.ADMINISTERED BY CODE	
11. SHIPPED FROM *(if other than 9)* CODE	FOB:	12. PAYMENT WILL BE MADE BY CODE	
13. SHIPPED TO CODE		14. MARKED FOR CODE	

15. ITEM NO.	16. STOCK/PART NO. DESCRIPTION *(Indicate number of shipping containers-type of container-container number)*	17. QUANTITY SHIP/REC'D*	18. UNIT	19. UNIT PRICE	20. AMOUNT

21. CONTRACT QUALITY ASSURANCE		22. RECEIVER'S USE
A. ORIGIN ( ) CQA [ ] ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents	B. DESTINATION ( ) CQA [ ] ACCEPTANCE of listed items has been made by me or under my supervision and they conform to contract, except as noted herein or on supporting documents	Quantities shown in column 17 were received in apparent good condition except as noted. DATE RECEIVED SIGNATURE OF AUTH GOVT REP TYPED NAME AND OFFICE
DATE SIGNATURE OF AUTH GOVT REP TYPED NAME AND OFFICE	DATE SIGNATURE OF AUTH GOVT REP TYPED NAME AND OFFICE	**If quantity received by the Government is the same as quantity shipped, indicate by ( ) mark, if different, enter actual quantity received below quantity shipped and encircle.*

23. CONTRACTOR USE ONLY

DD Form 250, NOV 92 *Previous edition may be used*

FIGURE 10. Sample DD Form 250.

5.1.3 Additional things to think about.

Are conditions of acceptance for systems, software, and interface implementations clear and unambiguous?
Are records of design descriptions sufficiently complete to facilitate software supportability?
Does the software development process include in-process inspection, analysis, demonstration and test activities needed to provide confidence in the acceptability of final products?
If software has been "accepted" with known errors, has the amount of retesting/regression testing to complete qualification been agreed upon?
Who will be responsible for verifying that acceptance criteria have been satisfied?
If a CDRL item is to be accepted, is the appropriate entry included in Block 7 (requestor) of the CDRL form? If a CDRL item is to be approved, is the appropriate entry included in Block 8 (approval code) of the CDRL form?
Are procedures in place to evaluate the impact of known errors on system performance, safety, security, etc.?
If deviations are granted, is the date set for remedying the deficiency agreeable to all parties (e.g., users, acquirer, developer, contracting agency, support agency, etc.)?

5.1.4 Related guidebook topics.

Approval by the acquirer
Contract
Qualification testing
Requirements
Statement of work
Subcontractor management
Traceability

5.2 Access for acquirer review.

5.2.1 Requirements summary. MIL-STD-498 provides requirements for the developer to provide the acquirer or its authorized representative access to developer and subcontractor facilities (subject to security and safety restrictions) for review of software products and activities required by the contract. Both the software engineering and test environments are included in this requirement. References to ***Access for acquirer review*** in MIL-STD-498 are listed in Figure 11, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.7
SDP	4.2.7

FIGURE 11. MIL-STD-498's references to ***Access for acquirer review***.

The developer is required to describe the approach for acquirer access in the SDP for the project. The intent of these requirements is to provide the acquirer added visibility, if needed, into the developer's efforts.

5.2.2 Acquirer's responsibilities and considerations. Acquirer visibility into developer activities is typically accomplished through review of the software products and activities required by the contract and the resulting acquirer/developer discussions regarding them. MIL-STD-498's requirements for joint technical and management reviews are intended to provide this level of visibility. These reviews might be scheduled periodically (every 6 weeks, quarterly, etc.) throughout the development with elapsed time the criteria for the visit. Or the visits might be scheduled around meeting particular objectives, such as reviewing designs or reaching agreements on issues resulting from reviews of software products, or other technical and management objectives. (See this guidebook's topic, *Joint technical and management reviews*, for more information.)

DoD service or government agency policy, however, may require added visibility through access to a developer's or subcontractor's facility based on a variety of program criteria, including dollar thresholds, criticality, or senior management or Congressional interest. When needed, access might be accomplished through:

- Permanent program office or agency representative(s) at a developer's facility with access to the project
- Members of the software support or program office team working side-by-side with the developer on the project over the course of the project

- Defense Contract Management Area Operations (DCMAO) in-plant representatives
- Acquirer access to on-line software products using compatible software development tools

Another method to provide access to developer-generated information is to include the Data Accession List (DAL) DID on contract. The DAL is a list of published data generated by the developer that can identify work products that might be needed for added acquirer insight into the project. (See this guidebook's topic, *Data accession list*, for more information.)

The amount of visibility an acquirer requires is often determined by what an acquirer needs to feel assured that the project is on track. An acquirer might feel assured with less visibility into a non-critical, low dollar project than into one to meet a critical threat or with a high dollar value. The degree of assurance may also be influenced by an acquirer's experience with the developer, the developer's track-record of successful projects in a specific technology/domain, developer (company-wide) capability assessment reports, and familiarity with project personnel.

Neither the number or frequency of visits nor the amount of lead time (notification of intent to visit) prior to visit(s) is stated in the standard. These may be made a part of the contract requirements or left to the developer to propose in the SDP, along with the details of how the acquirer's access needs will be met. When selecting and scheduling access, the acquirer should balance the worth of potential insight to be gained against potential cost/schedule impacts if the development team is diverted from critical work to confer with program office personnel or their authorized representatives.

5.2.3 Additional things to think about.

Does the SDP show a suitable schedule for joint reviews? For acquirer access to review processes? Products?
How can unauthorized requirement(s) modifications resulting from on-site discussions be prevented?
Does the DAL show updates to work-in-progress based on decisions at joint technical and management reviews?

5.2.4 Related guidebook topics.

Contract data requirements list
Data accession list
Joint technical and management reviews
Risk management

5.3 Approval by the acquirer.

5.3.1 Requirements summary. MIL-STD-498 provides requirements regarding approval by the acquirer. The standard defines "approval by the acquirer" as "*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*." This definition is based on FAR § 9.301, April 1984. References to ***Approval by the acquirer*** in MIL-STD-498 are listed in Figure 12, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.3*, 5.1.6, 5.7.1, 5.9.2, 5.11.2, 5.13.7, 5.18, 5.19.7, C.1, D.1
SDP	5.1.6, 5.7.1, 5.9.2, 5.11.2, 5.13.7, 5.18, 5.19.7

FIGURE 12. MIL-STD-498's references to ***Approval by the acquirer***.

MIL-STD-498 does not require acquirer approval for most of its software products. The standard is silent on "approval" for software products other than plans. The standard requires acquirer approval for the following:

1) Plans and updates to plans

2) Locations/dates of joint reviews

3) Use of:

- Any programming language for the deliverable software not specified in the contract
- Alternatives to the specified target computer(s) for qualification testing
- Alternatives to the specified hardware/software used to demonstrate regeneration capability for support
- Improved processes on the project (through updates to the plans)
- Alternative error categories and priority schemes for problem reporting
- Alternative software product evaluation criteria

Figure 13 lists developer's key activities related to ***Approval by the acquirer***.

Developer's key activities related to ***Approval by the acquirer***	References in MIL-STD-498
Describe the approach for obtaining acquirer approval. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses for obtaining acquirer approval.	SDP 5.1.6, 5.7.1, 5.9.2, 5.11.2, 5.13.7, 5.18, 5.19.7
Obtain acquirer approval for software development plans, software test plans, software installation plans, and software transition plans. With the exception of developer-internal scheduling and related staffing information, obtain acquirer approval for updates to plans.	5.1.6
Obtain acquirer approval to use any programming language for deliverable software not specified in the contract.	5.7.1
Obtain acquirer approval if CSCI or system qualification testing will not be conducted on the target computer system.	5.9.2, 5.11.2
Obtain acquirer approval for software and hardware not specified in the contract that is used by the developer for demonstrating that deliverable software can be regenerated and maintained.	5.13.7
Obtain acquirer approval of the locations and dates of joint technical and management reviews.	5.18
Obtain acquirer approval for any necessary and beneficial improvements to the software development process.	5.19.7
Obtain acquirer approval for an alternative problem categorization and prioritization scheme to that in Appendix C.	C.1
Obtain acquirer approval for use of software product evaluation criteria other than those found in Appendix D.	D.1

FIGURE 13. Developer's key activities related to ***Approval by the acquirer***.

5.3.2 Acquirer's responsibilities and considerations. Acceptance, approval, and authentication are all contracting related government procurement terms. These terms are often loosely used as if they were interchangeable, but each has a distinct meaning in the FAR. MIL-STD-498 bases its definitions on the FAR and uses only two terms in the standard: "acceptance" and "approval." MIL-STD-498 does not use the term "authentication." Authentication is sometimes used to mean that the acquirer assumes responsibility for the correctness of a product. This concept is not part of MIL-STD-498.

The difference between "acceptance" and "approval" is the difference between taking ownership (acceptance, FAR § 46.101, April 1984) and giving the developer the affirmation that the acquirer believes that a product is "good enough" to be the basis for future work (approval). Acceptance is defined in the standard as "*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" (FAR § 46.101, April 1984). (See this guidebook's topic, *Acceptance by the acquirer*, for more information.)

Approvals ensure that the developer provides the acquirer timely insight into plans for the project and changes to those plans. Approval of plans and changes in plans provides an acquirer visibility into the methods/procedures/tools (approach) planned for software development activities, the planned schedules, and the planned resources for performing those activities.

Approval of schedules, the dates and locations of joint activities, and other items ensures that the acquirer will be present to make necessary decisions, provide needed approvals, and resolve issues. Timely review and approval by the acquirer will prevent costly rework if changes are required. Note, however, that the standard does not require the developer to stop work pending acquirer approval. Provisions in the contract should specify the implications of untimely acquirer approvals, such as assumption of approval if no approval or disapproval is received within 30 days. An acquirer's approval does not shift responsibility from the developer to meet contractual requirements.

While the standard does not require approval for other software products, such as system or software design descriptions, the developer will usually expect acquirer feedback on the developer's design and other software products. Joint technical and management reviews are the forum in which this information is reviewed and discussed.

When deliverable software products are required, MIL-STD-498 Overview and Tailoring Guidebook, paragraph 5.4.12, provides guidance on scheduling deliverables. To the maximum extent possible, the CLIN or 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 SDP can lay out a proposed schedule that meets the constraints in the CLIN or CDRL. Final agreement on scheduling can take place during review and approval of the plan. Unrealistic scheduling of deliverable documents can result in an unsatisfactory "document driven" process, imposing a tyrannical schedule to meet contract requirements without consideration of the actual engineering work needed. This wastes the project's time and money. Appendix H of the standard also provides guidance on ordering deliverables.

5.3.3 Additional things to think about.

Are approval criteria for software plans and updates understood and agreed upon?
Are all parties aware of cost and schedule risk implications if approvals are not timely?
If a CDRL item is to be accepted, is the appropriate entry included in Block 7 (requestor) of the CDRL form? If a CDRL item is to be approved, is the appropriate entry included in Block 8 (approval code) of the CDRL form?
Are schedules for joint reviews consistent with obtaining timely approvals?
Has the acquirer approved the use of all programming languages to be used for deliverable software?

5.3.4 Related guidebook topics.

Acceptance by the acquirer
Contract
Contract data requirements list
Joint technical and management reviews
Statement of work

5.4 Architectural design.

5.4.1 Requirements summary. MIL-STD-498 provides requirements for defining and recording architectural design. Architectural design is one of the three identified elements of system/subsystem and CSCI design. MIL-STD-498 defines architecture as "*the organizational structure of a system or CSCI, identifying its components, their interfaces, and a concept of execution among them*." MIL-STD-498 calls for descriptions of the architecture at the system/subsystem and CSCI levels, provides for alternative views of that architecture, and recognizes that the information known about the system or CSCI may vary over time or that differing perspectives may be needed. References to ***Architectural design*** in MIL-STD-498 are listed in Figure 14, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.4*, 4.2.2, 4.2.3.1, 5.4.2, 5.6.2, 5.8.1, 5.10.1, 5.13.4, 5.13.5, Figure 3, Figure 6 (Items 8, 11, 25, 26), *E.4.4, E.4.6*
IDD	3
OCD	3.3, 5.3
SDD	4
SDP	4.2.2, 4.2.3.1, 5.4.2, 5.6.2, 5.13.4, 5.13.5
SPS	5.1
SRS	3.12
SSDD	4
SSS	3.12

FIGURE 14. MIL-STD-498's references to ***Architectural design***.

Architectural design and detailed design describe components from differing perspectives. If each component is considered as a "black box" with a "boundary" drawn around it, the difference between the architectural design and the detailed design of that component is whether the description provides a perspective from inside or from outside the box.

- Architectural design identifies each component (black box); describes how the components pass information between one another (interfaces); and depicts components' interaction (concept of execution) ignoring the internal workings of each component

- Detailed design describes the inside of each component (black box), i.e., how the component works internally to implement the required interface(s) and interactions

Figure 15 shows architectural design in relationship to other design elements and requirements.

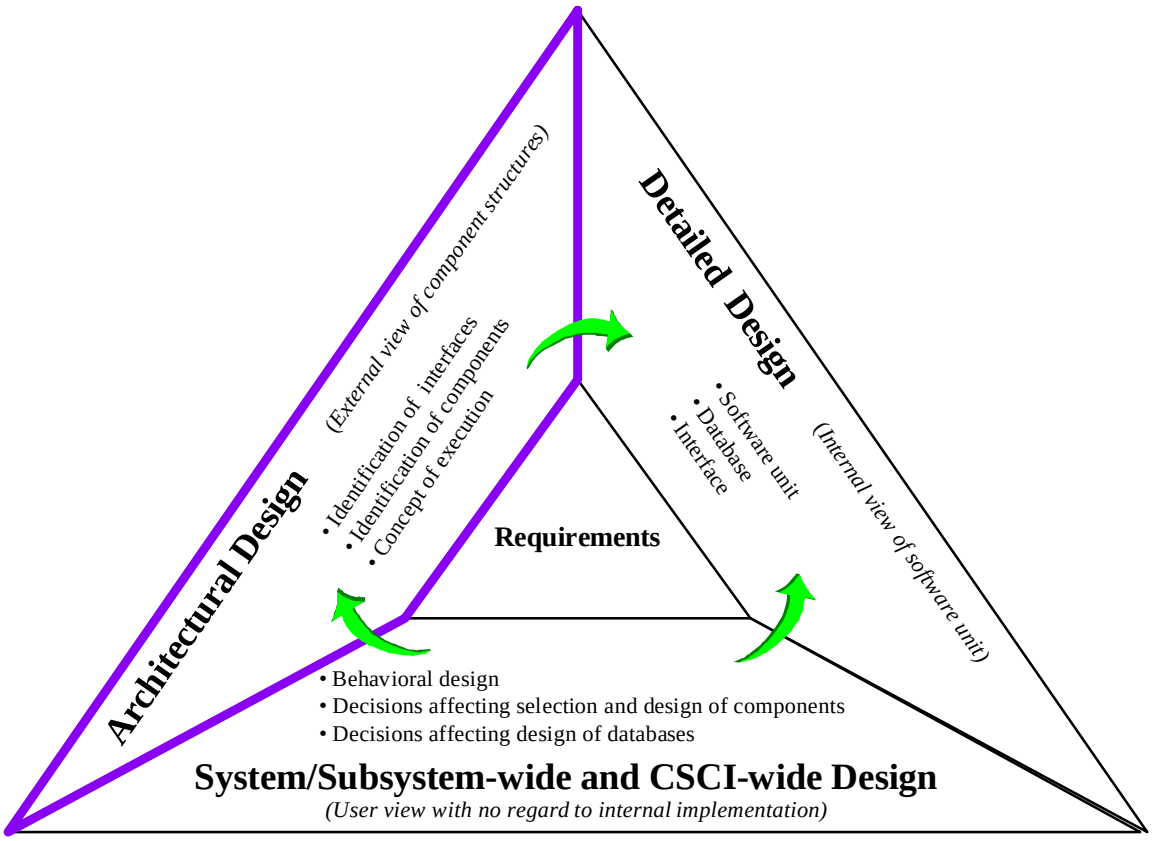


FIGURE 15. Architectural design in relationship to other design elements.

MIL-STD-498's design descriptions (System/Subsystem Design Description (SSDD) and Software Design Description (SDD)) provide requirements for a description of architectural design. The specification DIDs (System/Subsystem Specification (SSS) and Software Requirements Specification (SRS)) provide for a description of any design constraints on the architecture.

- Components are the named "pieces" of design and/or actual entities (subsystems, HWCIs, CSCIs, software units) of the system/subsystem/CSCI. In system/subsystem architectures, components consist of subsystems (or other variations), HWCIs, CSCIs, and manual operations. In CSCI architectures, components consist of software units, which may consist of other software units. MIL-STD-498's components in the design need not have a one-to-one relationship with the components that implement them.

- Interfaces are the relationships among components in which the components share, provide, or exchange data. (An interface is not a CSCI, software unit, or other system component; it is a relationship among them.) Although there may be software components or software units within components that "handle" the interface (i.e., to initiate the transfer of data, establish connectivity, or otherwise implement a relationship), in MIL-STD-498's terms, those handlers are components or units rather than interfaces.
- Concept of execution represents the dynamic relationships of the components. It can include such descriptions as flow of execution control, data flow, dynamically controlled sequencing, state transition diagrams, timing diagrams, priorities among units, handling of interrupts, etc.

The term "architectural design" is not used in database design to avoid possible conflicts with the definition if applied to databases. However, much of the information called for in the SSDD and/or SDD DIDs is also called for in some form in the Database Design Description (DBDD) DID's database-wide and detailed design sections.

MIL-STD-498 provides requirements for the developer to use the defined architecture to implement the software, to test that architecture along with other elements of the software design, and to evaluate the architecture. Evaluation criteria for architectural design are specified in Appendix D of the standard. In addition to the six basic evaluation criteria used to evaluate nearly all software products, evaluation of architectural design includes whether the architecture covers the requirements, is consistent with system/CSCI-wide design decisions, and is feasible. (See this guidebook's topic, *Software product evaluation*, for more information.) Figure 16 lists developer's key activities related to ***Architectural design***.

5.4.2 Acquirer's responsibilities and considerations. MIL-STD-498 allows many approaches for developing system/subsystem/CSCI architectures. To maintain a maximum degree of flexibility, the standard doesn't require a particular approach to be applied. This ensures that the standard remains method-independent. It does, however, provide requirements for the developer to describe the approach to be used in the SDP for the project. The developer's SDP is required to describe the methods/procedures/tools, standards, techniques, and conventions to be used in describing both the system/subsystem and CSCI architectures. This description will allow the acquirer to determine whether the level of detail, number of perspectives, and representations to be used to present the architectural design, in the acquirer's judgement, are good enough to be the basis for performing this activity. (See this guidebook's topic, *Approval by the acquirer*, for more information.)

Developer's key activities related to ***Architectural design***	References in MIL-STD-498
Describe the approach to be followed for architectural design, identifying risks/uncertainties and plans for dealing with them. Include all contractual clauses covering architectural design.	SDP 4.2.2, 4.2.3.1, 5.4.2, 5.6.2, 5.13.4, 5.13.5
Develop and apply standards for representing design. Describe or reference the standards to be followed in the SDP.	4.2.2
Describe current system or situation, identifying major system components, interconnections among them, interfaces to external systems and procedures, and inputs, outputs, data flow, and manual and automated processes.	OCD 3.3, 5.3
Participate in defining and recording the architectural design of the system (identifying the components of the system, their interfaces, and a concept of execution among them) and the traceability between the system components and system requirements. Incorporate design constraints specified in the SSS and IRS.	5.4.2 IDD 3 SSDD 4 SSS 3.12
Define and record the architectural design of each CSCI (identifying the software units comprising the CSCI, their interfaces, and a concept of execution among them) and the traceability between the software units and CSCI requirements. Incorporate design constraints specified in the SRS and IRS.	5.6.2 IDD 3 SDD 4 SRS 3.12
Identify, evaluate, and include reusable software products (that are cost-effective over the life of the system and meet requirements) in the architecture. Allocate requirements to candidate reusable CSCIs and software units.	4.2.3.1, Figure 3
Establish test cases, test procedures, and test data for conducting unit integration and testing, covering all aspects of the CSCI architectural design.	5.8.1
Participate in developing and recording test cases, test procedures, and test data for conducting CSCI/HWCI integration and testing, covering all aspects of the system-wide design. Record software-related information in SDFs.	5.10.1
Participate in updating the system design description to match the "as built" system.	5.13.5
Update the design description of each CSCI to match the "as built" software.	5.13.4 SPS 5.1
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 16. Developer's key activities related to ***Architectural design***.

To evaluate the soundness of the architectural design, the developer may need evaluations by those with expertise in related systems or software engineering disciplines such as safety, security, electronics, and logistics, as well as those knowledgeable about the system and its software. For example, security experts may be needed to evaluate the planned architectural approach, design, and implementation intended to provide security through a modular architecture. Quantitative evaluations of a selected architecture can include performance measurements via benchmark studies and analysis of computer hardware resource utilization data to provide early signs of potential sources of resource overutilization. In addition, prototype models can often validate architectural decisions regarding operational procedures, human factor considerations, interfaces, and feasibility. Discussion of system/subsystem and CSCI architecture can be the topic of one or more joint technical reviews.

When performing software design activities, it is often difficult to determine where one element of design ends and another begins. Definition of a component's design elements (architectural, system/subsystem/CSCI-wide, and at the CSCI level, detailed design) are performed iteratively. System-wide design decisions can constrain subsystem and CSCI architectural design, and the selected system or CSCI architecture can constrain system/subsystem/CSCI-wide and detailed design.

Software component selections, interfaces, and concepts of execution are often constrained by system-wide software design decisions, such as the decision to use a pipe and filter, client/server, or distributed design. Such design decisions, i.e., "client/server architecture," "distributed architecture," etc., constrain what components are needed, what capabilities are allocated to those components, interfaces among the components, and their concept of execution. Software architecture can also be constrained by the selected design method (e.g., object oriented versus functional decomposition) and language(s) selected for use.

Multiple descriptions of the architecture might be necessary. MIL-STD-498 leaves it up to the developer to identify how many descriptions are needed to clearly represent the design and what design methods/procedures/tools are to be used to represent those architectural design descriptions.

Similar constraints occur between requirements definition and design. For example, system specification, system design, subsystem specification, subsystem design, and so on, are performed iteratively until all subsystems, their constituent subsystems, etc., are specified and designed, and HWCIs, CSCIs, and manual operations are identified and their requirements specified. Since components can consist of other components, the design of one component can provide design constraints/requirements for its constituent components. Alternatively, known sub-elements of a component, such as a reusable software unit or CSCI, can constrain

the architectural design of that component. (See this guidebook's topic, *Requirements*, for more information.)

5.4.3 Additional things to think about.

Is architectural design a topic in appropriate joint technical reviews? What level(s) of architectural design or views are considered for review?
Are there adequate assessment methods in place for evaluating the architectural design? What trade-offs have been considered? What personnel and resources are required to review the architectural design?
Are requirements, if any, for software support addressed in the architectural design?
Are requirements for safety, security, and privacy addressed in the architectural design?
Have CASE tools been considered for assisting in the development and representation of architectural design?
If CASE tools are used to develop the architectural design, are they compatible with acquirer and/or support organization capabilities?
Is it clear when and how the "as built" architectural designs are updated?
Have key decisions regarding architectural design been recorded to facilitate support and change?
Have requirements been allocated to the components identified in the architectural design? Do methods/procedures/tools support this traceability?
Do system/CSCI specifications impose unnecessary architectural design constraints?
Have architectural design constraints (e.g., developing for reuse, use of specific approved domain architectures), been identified, if needed?

5.4.4 Related guidebook topics.

Approval by the acquirer
Behavioral design
Detailed design
Interfaces
Requirements
Software product evaluation
System/subsystem-wide and CSCI-wide design

5.5 Behavioral design.

5.5.1 Requirements summary. MIL-STD-498 provides requirements regarding behavioral design. The standard defines behavioral design as "*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*." Other terms commonly associated with behavior include: human factors engineering, human-machine interface (HMI), human-computer interface (HCI), and graphical user interface (GUI). Behavioral design includes descriptions such as: design decisions regarding inputs and outputs; responses to those inputs, description of physical systems modeled, and equations/algorithms; how databases/data files will appear to the user, etc. Behavioral design is one aspect of system/subsystem-wide and CSCI-wide design. References to ***Behavioral design*** in MIL-STD-498 are listed in Figure 17, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.6*, 4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.8.1, 5.10.1, 5.13.4, 5.13.5, *Figure 2,* Figure 3, Figure 6 (Items 7, 10, 25, 26), *E.4.4, E.4.6*
DBDD	3, 4.x.b.4, 4.x.b.6, 5.x.e.6
IDD	3
OCD	3.3, 5.3, 6
SDD	3, 4.3.x.d.4, 4.3.x.d.6
SDP	4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.13.4, 5.13.5
SPS	5.1
SRS	3.12
SSDD	3, 4.3.x.d.4, 4.3.x.d.6
SSS	3.12

FIGURE 17. MIL-STD-498's references to ***Behavioral design***.

System/subsystem and CSCI "behavior" may be either part of the requirements or part of the design. If the acquirer cares enough about behavioral characteristics to make them a condition for acceptance, they are requirements. All other system and software behavior is design. Behavior provides key information on how the system/subsystem or CSCI appears to act from the user's point of view, without concerns regarding how the system/subsystem or CSCI will implement that behavior. (See this guidebook's topics, *Architectural design, Detailed design*, and *System/subsystem-wide and CSCI-wide design*, for more information.) Figure 18 lists developer's key activities related to ***Behavioral design***.

Developer's key activities related to ***Behavioral design***	References in MIL-STD-498
Describe the approach for defining and recording the system/subsystem-wide and CSCI-wide design decisions. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses in planning for following and updating the system/subsystem-wide and CSCI-wide design decisions.	SDP 4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.13.4, 5.13.5
Develop and apply standards for representing design. Describe or reference the standards to be followed in the SDP.	4.2.2
Identify, evaluate, and include applicable reusable software products that are in the software design. Allocate requirements to candidate reusable CSCIs and software units.	4.2.3.1, Figure 3
Describe current and new or modified system or situation, identifying differences associated with different states or modes of operation (regular, maintenance, training, degraded, emergency, alternative-site, wartime, peacetime). Describe one or more operational scenarios that illustrate the role of the new or modified system, its interaction with users, its interface to other systems, and all states or modes identified for the system.	OCD 3.3, 5.3, 6
Participate in defining and recording system-wide design decisions (such as colors, layouts, fonts, icons and other display elements, beeps, lights), priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply. Incorporate design constraints specified in the SSS and IRS.	5.4.1 DBDD 3 IDD 3.x IRS 3.x SSDD 3 SSS 3.12
Define and record CSCI-wide and database-wide design decisions (such as colors, layouts, fonts, icons and other display elements, beeps, lights), priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply. Incorporate design constraints specified in SRSs/IRSs.	5.6.1 DBDD 3, 4 IDD 3.x IRS 3.x SDD 4 SRS 3.12
Establish test cases, test procedures, and test data for conducting unit integration and testing, covering all aspects of the CSCI-wide design.	5.8.1
Participate in developing and recording test cases, test procedures, and test data for conducting CSCI/HWCI integration and testing, covering all aspects of the system-wide design. Record software-related information in SDFs.	5.10.1
Participate in updating the system design description to match the "as built" system.	5.13.5 SSDD
Update the design description of each CSCI to match the "as built" software.	5.13.4 SPS 5.1
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 18. Developer's key activities related to ***Behavioral design***.

5.5.2 Acquirer's responsibilities and considerations. Behavioral design describes how the system/subsystem or CSCI appears to behave from the user's point of view. End users, therefore, often have valuable inputs into the behavioral aspects of a system and its software. Early behavioral descriptions can help users define their needs, thus avoiding costly rework later. If the support organization is responsible for training new users, their personnel may have background experience to draw upon and can provide valuable insights to the project. Behavioral design may also be reviewed by others with expertise in specialized areas (e.g., safety, security, privacy, and logistics), as well.

Behavioral design includes not only screen information, keyboard functions (e.g., the function key F10 or clicking on an icon may invoke a help function), and other human-to-machine interactions (meanings of lights, WARNINGS, error codes, etc.), but also includes business rules, data behavior, and other descriptions of how the system/subsystem or CSCI will behave. Some examples of behavioral design include:

- A subject matter expert's design decisions about algorithms, mathematical rules, and mechanical, aeronautical, chemical, electronic, and other engineering rules.
- A business rule such as, "update deposits to a checking account prior to deducting withdrawals."
- A statement regarding how data is perceived to be grouped/stored, such as "the file contains the customer's full name, address, and phone number."

Behavioral design is often prototyped through use of visual displays, full-scale mock-ups, auditorial queues, and other sensory demonstrations such as virtual reality. Prototypes can be used to augment or replace the design information and often have other uses including planning, layout, work flow analysis, manufacturing considerations, and reviews. Prototype models can validate behavioral decisions regarding operational procedures, human factor considerations, feasibility, and consistency of interfaces.

The developer is required to describe the methods/procedures/tools, standards, techniques, and conventions to be used for developing and describing the system/subsystem-wide design and CSCI-wide design, including behavioral design, in the SDP for the project. This description allows the acquirer to determine whether the level of detail and the representations used appear sound and are good enough to use as a basis for performing this activity. (See this guidebook's topics, *Approval by the acquirer*, for more information.)

5.5.3 Additional things to think about.

Is behavioral design a topic in appropriate joint technical reviews? Do users participate in reviews of behavioral design?
Has the acquirer coordinated personnel (operators, maintainers, trainers, security, safety, etc.) and resources to review the developer's behavioral design?
Have the behavioral designs been updated to reflect the "as built" behavior?
Do system/CSCI specifications impose unnecessary constraints on the behavioral design?
Are proposed report layouts and input/output screen formats included in the appropriate design documents?
Are there design standardization issues to be surfaced and resolved?
Have key decisions been recorded?

5.5.4 Related guidebook topics

Approval by the acquirer
Architectural design
Detailed design
Interfaces
Joint technical and management reviews
Requirements
System/subsystem
System/subsystem-wide and CSCI-wide design

5.6 Builds.

5.6.1 Requirements summary. MIL-STD-498 provides requirements that allow software to be delivered in builds. The standard is written in terms of developing software in multiple "builds", where "build" is defined as: "*(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: Other definitions sometimes use the terms "block" and "increment" as synonyms for "build.") 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 synonymously. References to ***Builds*** in MIL-STD-498 are listed in Figure 19, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.7*, 5, *Figure 1*, *Notes (5.1 - 5.16, 5.18), G.3, G.5, G.6, Figures 9-13*
SDD	4.1.d
SDP	5
SSDD	4.1.d

FIGURE 19. MIL-STD-498's references to ***Builds***.

Each build is interpreted as software that incorporates a specified subset of the planned capabilities. The builds might be prototypes, versions offering limited capability, or versions containing full use of some capabilities but not others. The build concept in MIL-STD-498 is used to accommodate multiple program strategies for acquiring systems and software, such as the Incremental and Evolutionary strategies described in DoDI 5000.2 and DoDI 8120.2. Appendix G of MIL-STD-498 and the MIL-STD-498 Overview and Tailoring Guidebook provide guidance on planning software builds, including identifying builds and their objectives, recording tailoring decisions regarding builds, and identifying and scheduling the activities to be performed in each build.

A project involving a single build, such as one following the Grand Design or Waterfall development strategy, will accomplish all required activities in that build. For projects developed in multiple builds, some of MIL-STD-498's required activities may be performed in every build, other activities may be performed only in selected builds, and activities and software products may not be complete until several or all builds are complete. The activities performed in each build may be ongoing, overlapping, or iterative, and different software products may proceed at different paces. The developer is required to describe the approach for builds, how the software development activities will be performed in each build, the

schedules for deliverable products, and the applicable methods/procedures/tools in the SDP for the project. Figure 20, copied from Figure 1 of MIL-STD-498, shows an example of how each MIL-STD-498 activity may be applied to a project that consists of multiple builds.

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

MIL-STD-498 and the MIL-STD-498 Overview and Tailoring Guidebook provide detailed guidance on the subject of builds. MIL-STD-498's "front-end" activities, such as requirements analysis, may appear in the earlier builds but not in the later builds, while "back-end" activities, such as system qualification testing, may appear in the later builds but not in the earlier ones. Software development activities, such as software design and software implementation and unit testing, and the integral processes, such as software configuration management, software product evaluation, software quality assurance, and others, may be present in all software builds. Figures 10 and 11 of MIL-STD-498 depict one possible way of applying the standard to the builds of a project using an Incremental or Evolutionary program strategy, respectively.

Figure 13 of the standard provides an example of build planning for a project involving multiple builds.

5.6.2 Acquirer's responsibilities and considerations. A key to the successful selection and definition of the number of builds (and their schedule) for a project is to clearly define the objectives of each build. For some projects, such as those using the Grand Design or Incremental program strategy approaches, it may be possible to precisely define the number of builds and their objectives at the beginning of the project. For other projects, such as those using the Evolutionary or another similar program strategy, it may be difficult or impossible to define all builds at the start. In such cases, build planning may itself be incremental with build decisions refined as the procurement and the project proceed.

Some considerations the acquirer and the developer might use to select the number of builds and the capabilities to be provided in each include:

- Date capabilities are needed
- Whether software capabilities can be divided
- Amount of time needed to complete each build
- How well user needs are understood/known
- Availability of developer staff and other resources

The degree of participation by the acquirer in build planning is project dependent. The acquirer may select the number of builds (with input from the developer) based, for example, on fielding specific capabilities before others, supporting system test schedules, or other system development requirements, and specify the tailoring of MIL-STD-498 for each build. Or the acquirer may select an overall program strategy, but allow the developer to lay out the builds and propose tailoring for each build.

The production of deliverable software products for each build can have a significant impact on overall program cost. To reduce cost, a developer's natural work products might provide "as is" documentation for early builds and might be used as the basis for discussion at joint reviews, delaying delivery of final products necessary for operation and support until completion of the project. For example, the acquirer might receive early design information from the developer's design tools rather than hardcopy documents. A delivered build might include only user manuals, Software Version Descriptions (SVDs) and executables rather than a full suite of documents. Caution should be used, however, to ensure that information needed for eventual operation and support is not "lost." Figure 21 provides a summary of

notes in Section 5 of MIL-STD-498 that explain how to interpret each MIL-STD-498 activity for a project involving multiple builds.

MIL-STD-498 Paragraph / Activity	Interpretation for Multiple Builds
5.1 *Project planning and oversight*	Planning for each build should include: overall planning for the contract, detailed planning for the current build, and planning for future builds covered under the contract to a level of detail compatible with the information available.
5.2 *Establishing a software development environment*	Establish the environment needed to complete the build.
5.3 *System requirements analysis*	System requirements might not be fully defined until the final build. Developer planning should identify the subset of system requirements to be defined in each build and the subset to be implemented in each build. System requirements analysis for a build means defining the system requirements identified for that build.
5.4 *System design*	System design might not be fully defined until the final build. Developer planning should identify the portion of the system design to be defined in each build. System design for a build means defining the portion of the system design identified for that build.
5.5 *Software requirements analysis*	Software requirements might not be fully defined until the final build. Developer planning should identify the subset of each CSCI's requirements to be defined in each build and the subset to be implemented in each build. Software requirements analysis for a build means defining the CSCI requirements identified for that build.
5.6 *Software design*	Software design might not be fully defined until the final build. Software design in each build means the design necessary to meet the CSCI requirements to be implemented in that build.
5.7 *Software implementation and unit testing*	Software implementation and unit testing will not be completed until the final software build. Software implementation and unit testing for each build includes those software units, or parts of units, needed to meet the CSCI requirements to be implemented in that build.
5.8 *Unit integration and testing*	CSCI unit integration and testing will not be completed until the final build. CSCI unit integration and testing means integrating software developed in the current build with other software developed in that and previous builds, and testing the results.
5.9 *CSCI qualification testing*	CSCI qualification testing will not be completed until the final build for that CSCI, or possibly until later builds involving items with which the CSCI is required to interface. CSCI qualification testing means planning and performing tests of the current build of each CSCI to ensure that the CSCI requirements to be implemented in that build have been met.

FIGURE 21. Interpretation of MIL-STD-498 activities for multiple builds.

MIL-STD-498 Paragraph / Activity	Interpretation for Multiple Builds
5.10 *CSCI/HWCI integration and testing*	CSCI/HWCI integration and testing may not be complete until the final build. CSCI/HWCI integration and testing means integrating the current build of each CSCI with the current build of other CSCIs and HWCIs and testing the results to ensure that the system requirements to be implemented in that build have been met.
5.11 *System qualification testing*	Qualification testing of the complete system will not occur until the final build. System qualification testing in each build means planning and performing tests of the current build of the system to ensure that the system requirements to be implemented in that build have been met.
5.12 *Preparing for software use*	Developer planning should identify what software, if any, is to be fielded to users in each build and the extent of fielding/distribution. Preparing for software use means including in each build those activities necessary to carry out the fielding plans for that build.
5.13 *Preparing for software transition*	Developer planning should identify what software, if any, is to be transitioned to the support agency in each build. Preparing for software transition means including in each build those activities necessary to carry out the transition plans for that build.
5.14 *Software configuration management*	The software products of each build may be refinements of, or additions to, software products of previous builds. Software configuration management should take place in the context of the software products and controls in place at the start of the build.
5.15 *Software product evaluation*	The software products of each build should be evaluated in the context of the objectives established for that build. A software product that meets those objectives can be considered satisfactory even though it is missing information designated for development in later builds.
5.16 *Software quality assurance*	The activities and software products of each build should be evaluated in the context of the objectives established for that build. An activity or software product that meets those objectives can be considered satisfactory even though it is missing information designated for development in later builds.
5.18 *Joint technical and management reviews*	The types of joint reviews held and the criteria applied will depend on the objectives of each build. Software products that meet those objectives can be considered satisfactory even though they are missing information designated for development in later builds.

FIGURE 21. Interpretation of MIL-STD-498 activities for multiple builds *(continued)*.

5.6.3 Additional things to think about.

Does the build strategy make sense? Does it meet operational needs? Is it feasible?
Have key decisions regarding the selection of builds been recorded?
Is the software build strategy compatible with the system development strategy?
Is there a process to document the outcome of each build (refined requirements) and resolve out-of-scope issues?
Are all requirements allocated to at least one build?
Are lessons learned and process improvements carried forward to subsequent builds?

5.6.4 Related guidebook topics.

Access for acquirer review
Approval by the acquirer
Software development process
Software life cycle processes
Version/revision/release

5.7 CASE tools.

5.7.1 Requirements summary. MIL-STD-498 provides no requirements for using computer-aided software engineering (CASE) tools for development of software. The standard does, however, encourage use of CASE tools and provides requirements for (1) documenting the methods/procedures/tools used for developing the source and executable code, and (2) labeling and identifying deliverable information/documents when CASE tools or forms of documentation other than traditional documents are used. References to ***CASE tools*** in MIL-STD-498 are listed in Figure 22, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword-3*, *1.2.4.4*, *3.38*, 5.1.1, 5.13.1, 5.13.2, 5.13.4, *6.4, Figure 2, H.5*
All DIDs	*Block 7*, 10.1
SDP	4.2.2, 5.2, 5.13.1, 5.13.2, 5.13.4
STrP	3.3

FIGURE 22. MIL-STD-498's references to ***CASE tools***.

The developer is required to describe the methods/procedures/tools (including CASE tools) to be used to develop software products, standards for software products, and any conventions needed to understand a product (such as design representations) in the SDP for the project.

The developer is required to describe the software and hardware needed for subsequent system support, trouble-shooting, or system modifications in the Software Transition Plan (STrP). The STrP also requires identification of special licensing or implementation considerations. The Software Product Specification (SPS) DID provides requirements for describing how source and executable files are generated.

When databases, CASE tools, or other alternative documentation forms are used to provide deliverable documents, each DID provides instructions (see block 7) regarding an acquirer's need to specify on the CLIN or CDRL whether deliverable data may reside in a CASE or other automated tool rather than in the form of a traditional document. Instructions in block 10.1 provide details for handling the recording of information when such forms are used. (See this guidebook's topics, *Contract data requirements list, Documentation (Preparing documents)* and *Software transition*, for more information.) Figure 23 lists developer's key activities related to ***CASE tools***.

Developer's key activities related to ***CASE tools***	References in MIL-STD-498
Describe the approach to be followed for establishing a software development environment, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to establishing the software development environment.	SDP 5.2, 5.13.1, 5.13.2, 5.13.4
Develop and apply the standards for representing requirements, design, code, test cases, test procedures, and test results, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to representing requirements, design, code, test cases, test procedures, and test results.	4.2.2 DBDD 4 IDD 3 SDD 3 SDP 4.2.2 SSDD 3
For deliverable information in CASE repositories, include external and internal labels containing (1) a title page and (2) table of contents containing pointers to, or instructions for accessing, each paragraph, figure, table, and appendix or their equivalents and indicating tailoring information, if any (see DID paragraphs for details). Assign names or numbers to pages in such a way that desired data can be indexed and accessed.	All DIDs 10.1.c-10.1.f
Establish, control, and maintain the software development environment. Ensure that each element of the environment performs its intended function(s). Maintain the SDL and SDFs for the duration of the contract.	5.2
Prepare the executable software, including any batch files, command files, data files, or other software needed to install and operate the software on its target computer(s); and prepare the source files. Include any batch files, command files, data files, or other software needed to regenerate the executable software. Prepare the "as built" CSCI design and related information.	5.13.1, 5.13.2, 5.13.4 SPS 5.1, 5.2, 5.3
Identify and describe the software and associated documentation needed to support the deliverable software. Include CASE tools, and data in these tools, compilers, test tools, test data, simulations, emulations, utilities, configuration management tools, databases and data files, and other software.	STrP 3.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 23. Developer's key activities related to ***CASE tools***.

5.7.2 Acquirer's responsibilities and considerations. A key concept of MIL-STD-498 is that defining and recording planning and engineering information is an intrinsic part of the software development process to be performed regardless of whether deliverable documents are required. CASE and integrated CASE tools can provide a repository for storing the information regarding the project and the products that are under development. Many CASE and integrated CASE tools exist. Some are good; some are not so good. Some are capable of generating information required by MIL-STD-498, or selected portions of that information; others are not.

Appendix B.3 of MIL-STD-498 lists criteria for evaluating reusable software to be used for fulfilling the requirements of the contract. Many of these criteria can be applied when evaluating CASE tools for the software development environment. For example, Appendix B lists the following criteria that might be applied (others may apply as well):

- Ability to provide safety, security, and privacy
- Reliability/maturity, as evidenced by established track record
- Availability and quality of documentation and source code
- Likelihood that the current version will continue to be supported by the supplier
- Impact on the development/maintenance if the current version is not supported
- Warranties available

Examples of some CASE tool capabilities are: "upper-CASE" tools that focus on automation of analysis or design activities (such as capturing data element information (e.g., social security number, phone number) relevant to the project's reports) and "lower-CASE" tools that support the activities of coding (such as source code generation based on information captured in a design model), testing, debugging or maintaining the system. There are many tools that support software development, such as project management, configuration management, requirements tracking, QA, test, and reengineering tools. These may be separate tools or integrated as part of a CASE environment.

5.7.3 Additional things to think about.

Where will the CASE tool(s) be installed to support the current project effort? Future support? Are all the hardware, software and network requirements for the tool(s) to operate successfully already available in those environment(s)? If not, who will pay for the needed changes and who will be responsible for coordinating installation?
Are there any policies that require the acquirer to use specific CASE tools, programming languages, hardware, network products, operating systems or database products that might affect a potential developer's existing software engineering environment for the project?
Are the developer and the designated support agency skilled in the techniques that are automated by the CASE tool(s)? If so, is automation worthwhile?
Are the developer and the designated support agency skilled in the use of the CASE tools? If not, who will be responsible for training? How long will it take for each to "come up to speed"?
Is it absolutely clear what the CASE tool(s) will do and what they will not do?
How widely used are the CASE tools? What track record do the tools have on other successful projects?
Are all the recommended components needed to provide promised functionality included with the tool(s), or are some planned for future development by the vendor?
How long will the vendor continue to support the CASE tool(s)? What is the CASE tool(s) vendor's track record? Business condition?
Does the CASE tool(s) interface and exchange data with other CASE tools or software products? Is the structure of the repository used by the CASE tool(s) in a proprietary or open architecture?
What access does the acquirer need to information that resides in CASE tool(s)? What provisions have been made to provide this access?
How will the project information within the CASE tool(s) be stored, secured, backed up and managed?
What is the quality of the documentation provided with the product? What type of "help" capability is built into the product? Will the tool(s) generate the output in compliance with MIL-STD-498? If not, how will the information be provided?

5.7.4 Related guidebook topics.

Access for acquirer review
Commercial off-the-shelf software products
Contract data requirements list
Data rights
Documentation (Preparing documents)
Rationale/key decisions
Software development environment
Software transition

5.8 Commercial-off-the-shelf software products.

5.8.1 Requirements summary. MIL-STD-498 provides requirements regarding commercial-off-the-shelf (COTS) software products under the standard's requirements for reusable software products. See this guidebook's topic, *Reusable software products*, and Appendix B of MIL-STD-498 for information pertaining to COTS and reusable software products.

5.8.2 Acquirer's responsibilities and considerations. COTS software products used in an application may provide all or part of the capability of the system. If a single COTS product doesn't provide all the needed capabilities, the software development project might consist of integrating multiple COTS software products to fulfill the system requirements.

COTS software products are products developed by vendors and intended for the commercial marketplace. Some typical examples include:

- Engineering development, test, and management tools
- Product standards
- General purpose software products (such as operating systems, networks, and communication software)
- Software application programs
- Extensible software products (such as products that provide basic capabilities but are intended to be modified or enhanced)
- Commercial software reuse libraries
- Office automation products (word processors, spreadsheets, etc.)

COTS software products typically incorporated into a developer's software engineering and test environment include: CASE tools, editors and browsers, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, database management systems, code analyzers, test case generators, path analyzers, and others. COTS software products used in the software engineering and test environments may be deliverable or non-deliverable.

Data rights and licensing issues are key considerations for COTS software products that are deliverable. For example, a small number of user licenses may be sufficient to support the project's development, but a large number of licenses may be required for use. These costs should be factored into the "make or buy" decision for the project.

The use of COTS software products can save money during the development of the system by reducing the amount of code that needs to be designed, written, and tested. However, COTS integration and supportability problems can reduce or even reverse those savings. These and other considerations should be reviewed by the acquirer and applicable operations and support organizations for each specific acquisition program. Independent commercial assessments regarding product performance and the likelihood that a vendor will be able to support a product can provide valuable information when researching and analyzing reusable products for incorporation.

5.8.3 Additional things to think about.

Will the COTS software be supported by the vendor after it is purchased?
Will the vendor be responsible for changes needed to the COTS software for the intended life of the system?
How will changes made to the COTS software be communicated to the users?
What is the impact of upgrades to the COTS software?
Does the COTS software require other auxiliary/support software? If so, what will happen to the COTS software if the support software is changed?
Is adequate help-line/on-site support offered by the COTS vendor?
Have all data conversion issues been adequately addressed?
Have all legacy data been accounted for prior to data conversion?
Has the developer taken steps to resolve licensing issues?
How will COTS software training be handled?
Do the operations and support personnel have expertise in the specific COTS product(s) under consideration?
What are the software support organization's preferences regarding software engineering or testing products?
Are there any hardware restrictions that might impact the choice of COTS software?
Would a demonstration or prototype be helpful to gain confidence in the developer's ability to integrate COTS software?
Do developer test plans adequately cover COTS and COTS integration at the unit, CSCI, and system levels? Modified COTS?
Have safety/security/privacy issues been considered for deliverable COTS products?
Have all risks resulting from the use of COTS software products been clearly identified, including strategies needed to mitigate risks? Is there a fall-back plan if COTS doesn't work as planned? As integrated?
Who will pay future software maintenance and upgrade costs?

5.8.4 Related guidebook topics.

CASE tools
Data rights
Licenses (Software)
Reusable software products

5.9 Computer hardware resource utilization.

5.9.1 Requirements summary. MIL-STD-498 provides requirements for the developer to analyze contract requirements concerning computer hardware resource utilization, allocate computer hardware resources among the CSCIs, monitor the utilization of these resources, and reallocate or identify additional resources necessary to meet contract requirements. References to ***Computer hardware resource utilization*** in MIL-STD-498 are listed in Figure 24, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.9*, 4.2.5, 5.13.4, *F.3.h*
OCD	8.2
SDD	4.1
SDP	4.2.5, 5.13.4
SPS	5.4, 6
SRS	3.10.2
SSDD	4.1
SSS	3.10.2

FIGURE 24. MIL-STD-498's references to ***Computer hardware resource utilization***.

The developer is required to describe the methods/procedures/tools to be followed for ensuring that all computer hardware resource utilization requirements are met in the SDP for the project. The developer's plan will include a detailed description of how resource utilization is to be measured (that is, at what level -- CSCI/unit/other, with what units of measure, under what conditions). In addition, the SDP will identify how often utilization will be estimated/measured (such as, monthly, at major milestones such as system design, software design, and "as built," other), and will describe the format to be used to present the utilization data.

The developer's SDP, Software Design Description (SDD), System/Subsystem Design Description (SSDD), and Software Product Specification (SPS) describe any special considerations that affect resource utilization and the way utilization is measured, such as the use of virtual memory, overlays, multiprocessors, networks, and operating system overhead. The SDD and SPS allow flexibility to present resource utilization data in an optimum manner by providing the option to present information in each CSCI's description or in a single location, such as one SDD and SPS, that can be referenced from others.

Computer hardware resource utilization

The intent of analysis, allocation, and monitoring of computer hardware resource utilization is to minimize risks of project cost overruns and schedule slips. Such risks can arise from an inability to meet performance requirements (such as response time to inputs, output volume to auxiliary devices, processing time) and lack of provision for future growth capacity (spare memory, spare processing capacity, etc.). The standard's requirements are also meant to ensure ongoing attention to resource utilization over time, resulting in early and cost-effective problem recognition and resolution. MIL-STD-498's Appendix F includes planned and actual use of hardware resources over time as a candidate management indicator. Figure 25 lists developer's key activities related to ***Computer hardware resource utilization***.

Developer's key activities related to ***Computer hardware resource utilization***	References in MIL-STD-498
Describe the approach to be followed for computer hardware resource utilization, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to computer hardware resource utilization.	SDP 4.2.5, 5.13.4
Describe the possible impact if the proposed system exceeds the desired maximum resource utilization.	OCD 8.2
Analyze, allocate, and monitor computer hardware resource utilization to meet contract requirements.	4.2.5
Define and record the measured computer hardware resource utilization for each CSCI.	5.13.4
Define and record the system's computer hardware resource utilization requirements, including the conditions, if any, under which the resource utilization is to be measured.	SSS 3.10.2
Define and record the allocation of computer hardware resource utilization to each CSCI.	SSDD 4.1
Define and record each CSCI's computer hardware resource utilization requirements, including the conditions, if any, under which the resource utilization is to be measured.	SRS 3.10.2
Define and record the estimated CSCI computer hardware resource utilization.	SDD 4.1
Record the measured CSCI computer hardware resource utilization for the "as built" system.	SPS 5.4
Record the traceability between computer hardware resource utilization measurements and related CSCI requirements.	SPS 6
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 25. Developer's key activities related to ***Computer hardware resource utilization***.

5.9.2 Acquirer's responsibilities and considerations. The acquirer should ensure that the requirements for resource utilization are clearly specified and reflect the true needs of the user. Whoever specifies computer resource utilization requirements (whether the acquirer or the developer) should analyze all computer hardware, not just the processor and the memory, to determine what utilization requirements are appropriate for the system. The System/Subsystem Specification (SSS) and the Software Requirements Specification (SRS) DIDs provide a list of computer hardware resources to consider, such as processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity. There may also be subcategories within each hardware category that must be addressed separately, such as different types of memory, (RAM versus ROM, volatile versus non-volatile, etc.). The specifier should attempt to consider all related response time requirements for the system when determining the conditions under which resource utilization requirements are to be met. For example, if there is a one second response time requirement, the specifier should consider requiring maximum processor utilization to be measured over a one second time period. Note that some capabilities, such as initialization, shut down, reports, backup, etc., may need to be exempt from response time requirements during operation.

Additional considerations may include:

- Unique needs for each system. It may be appropriate to require processor utilization limits for a real-time system planned for a 15 year service life (such as 50% processor utilization at maximum saturation) since spare capacity reduces the risk that future enhancements or modifications to the system could be constrained. But it might not be appropriate to require spare capacity for a system that will be used "as is" for only two years.

- Changing technology. Requiring a large amount of spare processor or memory capacity may not be cost-effective if the spare capacity won't be used for years and hardware costs are rapidly declining. It may be cost-effective to wait to purchase additional capability when it is actually needed (making certain, however, that the memory needed will still be available) rather than purchase initially (especially if installation impacts are minimal), or plan for periodic hardware upgrades over time.

The specifier should clearly define and justify the system workload conditions, if any, under which resource utilization must be estimated or measured (e.g., over a specified time period, with a specified number of concurrent users, with a specified number of simulated targets, performing specified functions). Requirements that are not clearly and precisely defined (e.g., "under worst-case conditions") can be a source of conflict between the acquirer and the developer. A workload analysis or other data gathering effort may be necessary to obtain the information needed as a basis for resource utilization requirements. For complex systems, it may be necessary to define multiple measurement conditions, such as high, average, and low

workloads. When system decisions depend on computer resource utilization measurements, the software developer should ensure that plans and schedules for reporting measurements meet those needs.

Since measurement of resource utilization can be complicated and time consuming, the acquirer should only require measurements that make sense for the project based on project variables. For example, measurements might make sense when: (1) growth is expected for software requirements or performance, (2) there is a risk of resource overload, or (3) there might be cost or schedule impacts if resource utilization requirements are not met. Note, however, that measurement of computer hardware resource utilization might apply even if spare capacity is not required. Measurements of resource utilization have an intrinsic value as a tool to reduce risk of not meeting performance requirements or determining too late that needed or desired capabilities cannot be provided without rework/redesign of either or both the hardware and software.

The acquirer should review the developer's approach to resource utilization analysis, allocation, and monitoring described in the SDP to ensure that there is a clear description of how the utilization is determined, what items are measured, and how reporting will occur. When resource utilization requirements are not being met, software optimization by the developer (such as rewriting portions of high order language code using assembly language) may often improve system performance enough to meet utilization requirements. Trade-off analysis should be performed to determine if hardware changes or software changes are the most effective way to meet the requirements. The acquirer and developer might want to establish computer hardware resource utilization as a standard topic at joint technical and management reviews.

Finally, the acquirer should determine if any DoD Service-imposed policies or guidelines exist that establish reserve requirements to be imposed on the developer. If required, reserve requirements should be included in the SOW and system specification.

5.9.3 Additional things to think about.

Has a computer hardware resource utilization analysis been performed prior to selection/purchase of computer hardware?
Has the "best" operating system been selected, particularly with regard to memory usage and memory management and supportability?
Do resource utilization estimates adequately account for operating system overhead and other factors of the environment in which the software will operate?
Are peak and nominal workload definitions representative of the operational environment?
If resource utilization requirements are not being met, has a thorough trade-off analysis been performed to determine whether hardware or software changes are the most cost-effective way to meet the requirements?
Have resource utilization requirements been properly specified to prevent the developer from delivering a system that meets the specified requirements but does not meet the needs of the system's users?
Has analysis been performed to determine the capability of local area networks (LANs) and wide area networks (WANs) to support the level of transmission traffic and expected number of users? Has a growth factor been anticipated?
Is computer hardware resource utilization a topic in appropriate joint technical reviews?

5.9.4 Related guidebook topics.

Risk management

5.10 Contract.

5.10.1 Requirements summary. MIL-STD-498 can be applied to contractors, subcontractors, or government in-house agencies performing software development. When MIL-STD-498 is invoked in a two-party agreement between an acquirer and a developer, it is usually cited on a contract. MIL-STD-498 defines "contract" as an "*agreement between the acquirer and developer*." The contract's SOW will specify each type of software to which the standard applies. The CDRL package will define requirements for deliverable software products. The acquirer is expected to identify the types of software and tailor the standard appropriately for each type of software. By tailoring and invoking MIL-STD-498 on a contract, the acquirer is invoking applicable portions of sections 4 and 5, the mandatory appendixes, and the DIDs referenced from the respective sections. In turn, the standard provides requirements for the developer, as a prime contractor, to "flow down" applicable contract requirements in subcontracts. See Figure 6 of this guidebook for the relationships between the contract and the government agencies, developer, subcontractor, SOW, CLIN, and CDRL. References to ***Contract*** in MIL-STD-498 are listed in Figure 26, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 4 - 5, 1.2.1, 1.2.2, 1.2.4.1, 3.1, 3.3, 3.16, 3.18, 3.26, 3.30, 3.39*, 4.1, 4.2.3.1, 4.2.3.2, 4.2.4.4, 4.2.5, 4.2.7, *5.1 Note*, 5.1.1, 5.1.4, 5.1.5, 5.1.6, 5.2.3, 5.2.4, *5.4.1 Note*, 5.7.1, 5.12.4, 5.13.7, 5.14.1-5.14.5, 5.15.2, 5.16.1, 5.17.1, 5.17.2, 5.19.3, 5.19.5, 5.19.6, *6.2, 6.4, 6.5*, B.3, *G.6, G.6.3*
All DIDs	*Block 7*, 10.1.c, *10.1.h*
SDP	4, 5
SRS	3.11
SSS	3.16
STrP	7.d

FIGURE 26. MIL-STD-498's references to ***Contract***.

Many of MIL-STD-498's references to contract are associated with the standard's "shell requirements," imposing no requirements until the specifics are provided 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, and (3) to provide a framework for incorporating project-specific requirements in the contract. Figure 27 lists summaries of MIL-STD-498's shell requirements.

MIL-STD-498 Paragraph	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 testing is to be witnessed, dry run the test cases and procedures	Indication of whether CSCI qualification testing will be acquirer-witnessed in each build
5.11.4	If system qualification testing 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

FIGURE 27. MIL-STD-498's "shell requirements".

MIL-STD-498 Paragraph	Summary of the Requirement	To Flesh This Out, Include in the Contract:
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 27. MIL-STD-498's "shell requirements" - *(continued)*.

5.10.2 Acquirer's responsibilities and considerations. The acquirer's task in contracting involves many activities and may require anywhere from a few months to a few years to complete. Following are brief summaries of the various activities involved:

- Defining the concept. The general system concept and the funding requirements are developed. The system concept may be developed within the acquirer's organization or in another organization. The funding requirements, likewise, may be developed within the acquirer's organization or another organization.

- Getting potential developers involved. Informing potential developers of forthcoming procurements is accomplished through general informational sessions typically held annually, announcements in the Commerce Business Daily, or via face-to-face meetings between the acquirer and potential developers. Potential developers respond by performing developer-internal trade studies and preliminary designs. The preliminary designs may be used by the acquirer to shape the content of the contract and the general system architecture and specifications.

- Draft Request For Proposal (RFP). Sometimes a draft RFP is released to potential developers, typically two to four months prior to the formal RFP. The draft RFP is usually incomplete. However, the draft is of value to both the acquirer and potential developers: (1) developers are informed of the latest information that the acquirer has regarding the procurement, (2) the acquirer receives important feedback, including suggestions for tailoring MIL-STD-498 and the DIDs from the potential developers, and (3) questions and suggestions from potential bidders can lead to improvements in the formal RFP.

- Formal RFP. The formal RFP is released upon completion of the procurement package. An important element of the RFP is the model contract, whose contents are listed in Figure 28.

- Proposal. Potential developers prepare their individual responses to the formal RFP. The preparation activity usually allows the developers to submit questions regarding the procurement shortly after receipt of the RFP. Questions regarding the procurement need to be answered promptly to avoid extending the time for preparation of proposals. Typical proposal response times are 30 to 60 days, sometimes longer for very complex proposals.

- Evaluation. Proposals from the potential developers are evaluated by a source selection board or DoD service or government agency equivalent. The board usually consists mainly of personnel from the acquirer organization, but can also include user personnel. In some cases contractor support is provided, usually on a non-voting and non-competing basis. To avoid delays, the acquirer's evaluation board should be in place and ready to function upon receipt of the proposals. The evaluation process typically includes issuance of Clarification Requests (CRs) and Deficiency Reports (DRs) to individual potential developers. (CRs are for the purpose of clarifying areas of the proposal, whereas DRs require that the potential developer correct areas that would otherwise disqualify him from the competition.) During evaluation, the source selection board keeps the source selection authority or DoD service or government agency equivalent informed of their progress, usually via briefings.

- Negotiation. After completing the evaluation, potential developers are sometimes invited to individual contractor face-to-face negotiations with the acquirer. Here further clarifications are made. The acquirer may award the contract on the basis of the face-to-face meetings or the potential developers may be invited to submit a Best-and-Final-Offer (BAFO).

- BAFO. Potential developers submit their BAFOs following the face-to-face meetings.

- Contract Award. The source selection board presents the results of its evaluations, negotiations, and resultant BAFOs to the source selection authority. The presentation may include recommendation for award. The source selection authority then makes the award decision.

SECTION	TITLE
	Part I
	The Schedule
A	Solicitation/contract form
B	Supplies or services and prices/costs
C	Description/specifications/work statement
D	Packaging and marking
E	Inspection and acceptance
F	Deliveries or performance
G	Contract administration data
H	Special contract requirements
	Part II
	Contract Clauses
I	Contract clauses
	Part III
	List of Documents, Exhibits, and Other Attachments
J	List of attachments
	Part IV
	Representations and Instructions
K	Representations, certifications, and other statements of offerors or quoters
L	Instructions, conditions, and notices to offerors or quoters
M	Evaluation factors for award

FIGURE 28. Uniform contract format.

There are two basic contract categories -- fixed price and cost reimbursable. The type of contract influences the extent of the acquirer's interaction with the developer during the course of the project. The format and content of most contracts are governed by FAR § 15.406, Preparing requests for proposals (RFPs) and requests for quotations (RFQs). Others are governed by FAR § 14.201, Preparation of invitations for bids. Both have the same contract

format as shown in Figure 28. The format and content of these FARs are not modified by the applicable DFARS. FAR § 15.406 discusses each of the sections of the Uniform Contract Format. Those of most interest with regard to MIL-STD-498 are in Section C, which contains the work statement where MIL-STD-498 would be invoked. Any tailoring of the standard would also be included in the work statement. The standard itself would be listed in Section J, List of attachments. Figure 29 briefly describes variations of the two basic contract types.

Contract Type	Major Characteristics
Firm fixed-price (FFP)	The price is not subject to adjustment by reason of cost or performance. The contractor is obligated to perform the contract at the established price.
Fixed-price with economic price adjustment	The fixed price is adjusted upward or downward based upon the occurrence of contractually specified economic contingencies that are clearly outside the control of the contractor.
Fixed-price incentive (FPI)	The profit is adjusted and the final price is established by a formula based on the relationship of final negotiated cost to target cost previously established.
Firm fixed-price level of effort	A fixed price is established for a specified level of effort over a stated time frame. If the level varies beyond specified thresholds, the price is adjusted.
Cost	Reimbursement consists of allowable cost; there is no fee provision.
Cost-sharing	An agreed portion of allowable cost is reimbursed.
Cost plus fixed-fee (CPFF)	Reimbursement is based on allowable cost plus a fixed fee, adjusted by a formula based on the relationship of total allowable cost to target cost.
Cost plus incentive-fee (CPIF)	Reimbursement consists of allowable cost incurred and a fee adjusted by a formula based on the relationship of total allowable cost to target cost.
Cost plus award-fee (CPAF)	Reimbursement consists of allowable cost incurred and a two-part fee (a fixed amount and an award amount based on an evaluation of the quality of contract performance).
Time and material (T&M)	Direct labor hours expended are reimbursed at fixed hourly rates, which usually include direct labor costs, indirect expenses, and profit. Material costs are reimbursed at actual plus a material handling charge, if applicable.
Labor hour	Direct labor hours expended are reimbursed at a fixed hourly rate, usually including all cost and profit.

FIGURE 29. Types of contracts.

Standards, such as MIL-STD-498, are required to be tailored for each procurement. Tailoring is the process of evaluating each requirement in a standard or DID to determine whether it is necessary for a given project and deleting those requirements that are not needed. Refer to the MIL-STD-498 Overview and Tailoring Guidebook for more information on tailoring. Tailoring for a given contract is often an incremental activity. It is important for the acquirer to involve all key system acquisition participants in the tailoring process. With each participant contributing specialized expertise, the acquirer can arrive at tailoring for the contract that addresses all concerns. Key participants include:

- Technical staff in, and available to, the program office, such as software engineering, configuration management, quality assurance, and test personnel
- Contracting office personnel
- User and support personnel
- Potential developers

The software user and software support personnel should review the tailoring decisions to avoid problems that might arise later if their needs have been neglected or tailored out. For example, the support agency might expect to receive a Software Transition Plan (STrP) from the developer if MIL-STD-498 is on contract. If the acquirer has tailored out the activity Preparing for software transition and not informed the support agency, acquirer's support agency and developer's time might be wasted searching for that plan later. The final tailoring decisions, however, subject to appropriate review, remain the responsibility of the acquirer. Figure 30 provides a table listing the levels of interest of key players in the selection/tailoring/evaluation of software products. Key players and their primary concerns and responsibilities are:

- **Program management** -- concerned with project schedules, costs, and adherence to the contract.
- **Technical/subject matter expert** -- responsible for ensuring technical integrity and functional usefulness of the products
- **User/operator** -- responsible for using or operating the system/subsystem and software, and may validate, participate in defining, or define requirements
- **Support agency** -- concerned with maintaining installed software

Software Product	Acquirer organization/functional area			
	Program management	Technical/subject matter expert	User/ operator	Support agency
Plans				
Software Development Plan (SDP)	●	●	○	◎
Software Installation Plan (SIP)	●	◎	●	○
Software Transition Plan (STrP)	●	◎	○	●
Concept/Requirements				
Operational Concept Description (OCD)	○	●	●	○
System/Subsystem Specification (SSS)	○	●	●	○
Software Requirements Specification (SRS)	○	●	●	○
Interface Requirements Specification (IRS)	○	●	●	○
Design				
System/Subsystem Design Description (SSDD)	○	●	○	●
Software Design Description (SDD)	○	●	○	●
Database Design Description (DBDD)	○	●	○	●
Interface Design Description (IDD)	○	●	○	●
Qualification Test Products				
Software Test Plan (STP)	●	●	○	○
Software Test Description (STD)	○	●	○	○
Software Test Report (STR)	○	●	○	○
User/Operator manuals				
Software User Manual (SUM)	○	●	●	○
Software Center Operator Manual (SCOM)	○	●	●	○
Software Input/Output Manual (SIOM)	○	●	●	○
Computer Operation Manual (COM)	○	●	●	○
Support manuals				
Computer Programming Manual (CPM)	○	●	○	●
Firmware Support Manual (FSM)	○	●	○	●
Software				
Software Product Specification (SPS)	○	●	○	●
Software Version Description (SVD)	○	●	●	●

● **High interest** ◎ **Medium interest** ○ **Low interest**

FIGURE 30. Levels of interest in selection/tailoring/evaluation of software products among acquirer organizations.

5.10.3 Additional things to think about.

Will it be necessary for the developer to rework software plans due to the acquirer's failure to flesh out "shell requirements"?
Has the developer considered all "shell requirements" in software planning?
Do contract provisions over-constrain development planning?
Have assumptions been documented and validated with the developer?
Does the SOW clearly identify required support or operation services and deliverables?
Is the project scope clearly stated?
Is the approach described in the SOW feasible?
What are the acquirer's responsibilities for delivering products and services to the project?
What are the developer's responsibilities for delivering products and services to the project?
What are the criteria for project completion?
Has the prime contractor flowed applicable contract requirements to subcontractors?
Does the acquirer understand the methods to be used and the degree of involvement of the acquirer organization stated in the contract and the tailored standard(s)?

5.10.4 Related guidebook topics.

Acceptance by the acquirer
Approval by the acquirer
Contract data requirements list
Integrated product team
Requirements of the standard
Statement of work
Subcontractor management

5.11 Contract data requirements list.

5.11.1 Requirements summary. MIL-STD-498 describes the Contract Data Requirements List (CDRL) as the form used to list deliverable software products to be provided by the developer. All of MIL-STD-498's DIDs are candidates to be used for requiring deliverable software products via the CDRL. References to ***Contract data requirements list*** in MIL-STD-498 are listed in Figure 31, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.1, 5.1.1 Note 2, 6.2, 6.3, 6.4, 6.5,* Figure 6, D.4.9, *Appendix H*
All DIDs	*Block 7*, 10.1.c

FIGURE 31. MIL-STD-498's references to ***Contract data requirements list***.

Special provisions have been made in MIL-STD-498 to provide for delivery of executable and source code via the CDRL in accordance with the DFARS. MIL-STD-498's Software Product Specification (SPS) provides a vehicle for this delivery (hard copy of source files is not required unless specified). Changes in the DFARS subsequent to the publication of MIL-STD-498 provide for delivery of technical data and software as a CLIN (DFARS § 227.7103-2, DFARS § 227.7203-1.b.3). Since the DFARS § 227.7203-14 defines software documentation as technical data, all of MIL-STD-498's software products can be made deliverable products via the CLIN rather than the CDRL. However, the DFARS still permits the use of a CDRL as an exhibit referenced from the SOW for use in listing deliverable software products (DFARS § 227.7203-2).

5.11.2 Acquirer's responsibilities and considerations. The CDRL is a key part of a contract referenced from the SOW. It is a collection of one or more DD Form 1423(s) specifying requirements for the deliverable software products for a project. Key information provided on the CDRL are: (1) the DID number applicable to the document, e.g., DI-IPSC-81427, Software Development Plan, (2) the tailoring, if any, applicable to the project (Block 16), (3) acceptance indicator (Block 7), (4) approval code (Block 8), (5) dates required for delivery (Blocks 12 and 13), and (6) addressees to whom the documents are to be sent and number of copies to be provided (Block 14). (See Figure 32 for a sample DD Form 1423.) Figure 6 of this guidebook places the CDRL in relationship to players and agreements assumed by MIL-STD-498.

CONTRACT DATA REQUIREMENTS LIST (1 Data Item)	Form Approved OMB NO.0704-0188

Public reporting burden for this collection of information is estimated to average 110 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503. Please DO NOT RETURN your form to either of these addresses. Send completed form to the Government issuing Contracting Officer for the Contract/PR No. listed in Block E.

A. CONTRACT LINE ITEM NO.	B. EXHIBIT	C. CATEGORY: TDP_____ TM_____ OTHER_____
D. SYSTEM/ITEM	E. CONTRACT/PR NO.	F. CONTRACTOR

1. DATA ITEM NO.	2. TITLE OF DATA ITEM	3. SUBTITLE	17. PRICE GROUP

4. AUTHORITY *(Data Acquisition Document No.)*	5. CONTRACT REFERENCE	6. REQUIRING OFFICE	18. ESTIMATED TOTAL PRICE

7. DD 250 REQ	9. DIST STATEMENT REQUIRED	10. FREQUENCY	12. DATE OF FIRST SUBMISSION	14. DISTRIBUTION			
					b. COPIES		
8. APP CODE		11. AS OF DATE	13. DATE OF SUBSEQUENT SUBMISSION	a.	DRAFT	Final Reg	Final Repro
16. REMARKS							
				15. TOTAL			

G. PREPARED BY	H. DATE	I. APPROVED BY	J. DATE

DD Form 1423-1, JUN 90 *Previous editions are obsolete* Page _____of_____Pages
1007/183

FIGURE 32. Sample DD Form 1423.

The CDRL should also specify whether deliverable data are to be delivered on paper or electronic media; are to be in electronic form (such as ASCII, CALS, or compatible with a specified word processor or other support software); may be delivered in developer format rather than in the format specified in the DID; or may reside in a CASE or other automated tool rather than in the form of a traditional document. MIL-STD-498 removes a distinction between a traditional document (paper) and electronic documents, stating that a document is "*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*." MIL-STD-498 allows the acquirer the flexibility to determine how information regarding the software development should be recorded for delivery and allows for CASE or other automated tools to act as repositories.

Prior to putting a DID on a CLIN or CDRL the acquirer should determine which documents should be delivered and when they should be delivered. Considerations include who will use the document, when that use will take place, why the document needs to be delivered, and whether the document should be delivered in the form of a traditional document. A "data call" is one method used by the acquirer to solicit this information from users and support agency personnel. Each request for data is usually supported with written rationale so that users and support agency personnel can surface their needs regarding each type of software on the project.

A key concept in MIL-STD-498 is 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. The standard specifies the software development activities to be performed and identifies the DIDs that describe the information to be recorded as a result of performing the activities. When the information is to be delivered, each DID provides both the content and format requirements for the deliverable. When the acquirer does not require delivery of the information, the contents of each DID act as a checklist of information to be recorded [*somewhere*] as a result of performing the activity. MIL-STD-498 acknowledges that putting the information into a specific paragraph structure, assembling the information, marking the document, copying and distribution are separate tasks from generating and recording the information and require additional time and effort on the part of the developer. (See this guidebook's topics, *Documentation (Preparing documents)* and *Documentation (Recording information)*, for more information.) Appendix H of MIL-STD-498 gives guidance on ordering deliverables. See the MIL-STD-498 Overview and Tailoring Guidebook for information on tailoring.

5.11.3 Additional things to think about.

How can the acquirer avoid over-specified CDRL clauses and delivery dates that overly restrict development planning or incur unnecessary costs?
Does a one-time DID need to be prepared to specify additional information requirements?
What resources will the acquirer need to determine the preferred scope and depth of needed software development activities and products for the project?
How can the acquirer ensure that developer input is considered prior to finalizing the SOW tasking and CDRL?

5.11.4 Related guidebook topics.

Acceptance by the acquirer
Approval by the acquirer
Contract
Documentation (Preparing documents)
Documentation (Recording information)
Executable software
Source files
Subcontractor management

5.12 Corrective action.

5.12.1 Requirements summary. MIL-STD-498 provides requirements for a software development process that includes corrective action. The developer is required to describe the methods/procedures/tools (approach) to be used for corrective action in the SDP for the project. The corrective action activity is required to cover each problem detected in software products under project-level or higher configuration control and each problem in activities required by the contract or described in the SDP. Problem/change reports serve as input to the corrective action system. Each problem/change report is required to describe a problem, the corrective action needed, and the actions taken to date. (See this guidebook's topic, *Problem/change report*, for more information.) References to ***Corrective action*** in MIL-STD-498 are listed in Figure 33, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.1, *Figure 1,* 5.14.3, 5.15.2, 5.16.2, 5.17, *Figure 2*, Figure 3, Appendix C, Figures 4 and 5, *F.3, Figures 9-12*
SDP	5.17

FIGURE 33. MIL-STD-498's references to ***Corrective action***.

MIL-STD-498's requirements for a corrective action system are:

- Inputs to the system are to consist of problem/change reports
- The system is to be closed-loop, ensuring that all detected problems are promptly reported and entered into the system, action is initiated on them, resolution is achieved, status is tracked, and records are maintained for the life of the contract
- Each problem in a software product is to be assigned to one or more of the problem categories in Figure 4 of MIL-STD-498; each problem in a software product or activity is to be assigned one of the priorities in Figure 5 of MIL-STD-498; and each problem in an activity is to be assigned to one or more of the activities in Figure 1 of MIL-STD-498
- Analysis is to be performed to detect trends in problems reported
- Corrective actions are to be evaluated to determine whether problems have been resolved, adverse trends have been reversed, and changes have been implemented without introducing additional problems

Figure 34 lists developer's key activities related to ***Corrective action***.

Developer's key activities related to ***Corrective action***	References in MIL-STD-498
Describe the approach to be followed for corrective action, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to corrective action.	SDP 5.15.2, 5.16.2, 5.17
Perform corrective action to include problem/change reports and a corrective action system for handling each problem detected in software products under project-level or higher configuration control and for describing and handling each problem in activities required by the contract or described in the SDP. Each problem/change report describes the problem, the corrective action needed, and the actions taken to date. Implement a corrective action system that uses problem/change reports for input, that is closed-loop, that classifies each problem by category and priority, that is analyzed to detect trends in problems reported, and that is evaluated for effectiveness.	5.17
Assign each problem in software products to one or more of the categories in Figure 4 of MIL-STD-498. Assign each problem in software products or activities to one of the priorities in Figure 5 of the standard. Assign each problem in activities to one or more of the activities in Figure 1 of the standard.	Appendix C, Figures 4 and 5
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 34. Developer's key activities related to ***Corrective action***.

5.12.2 Acquirer's Responsibilities and Considerations. MIL-STD-498's requirements for corrective action are intended to assure a disciplined process for handling problems and changes to products and processes. The standard's corrective action system may be manual or automated. Figure 35 shows an example of a corrective action activity.

MIL-STD-498's requirements for corrective action apply to software products only after they are placed under developer project-level or higher configuration control. This limitation is intended to prevent an overloading of the corrective action system with problems found during evaluations of immature products under lower-level controls, such as those found during peer reviews, walk-throughs, or evaluations of preliminary products and prototypes under author-level control. Note that products may mature early or late in a project and that when software is developed in builds, products may go through several cycles of maturity (i.e., requirement specification and design description may be mature very early while source code and test reports may not mature until late).

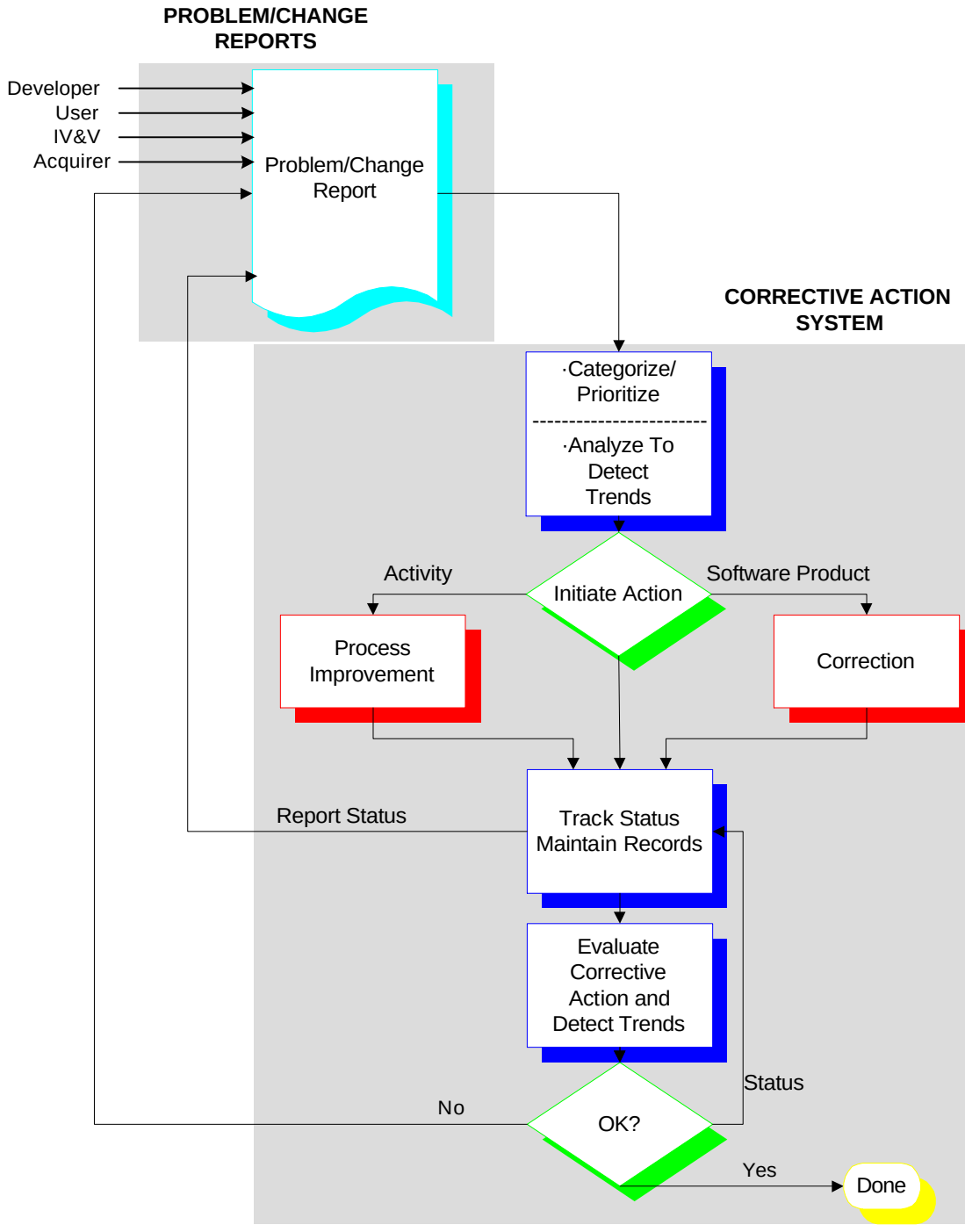


FIGURE 35. Example of a corrective action activity.

The acquirer should specify in the contract the relationship between the developer and IV&V agent(s), if any, who may provide problem/change reports to a developer. When Incremental or Evolutionary program strategies are applied, the acquirer should also determine how feedback from users and others, such as support agency personnel, should be handled. The acquirer may want to consider whether all problem/change reports should go directly to the developer or be screened by the acquirer to save project time and expense prior to submission to the developer for final evaluation and resolution.

Depending upon the size of the project, criticality, number of contractors, DoD service or government agency policy, and other factors, including availability of program office resources, the acquirer may need to establish Computer Resource Working Groups (CRWGs), Interface Control Working Groups (ICWGs), or Configuration Control Boards (CCBs) to help determine priorities for resolving problems or implementing changes.

5.12.3 Additional things to think about.

Is the developer's corrective action activity truly closed-loop?
Is the problem categorization/prioritization scheme optimal for the project?
If problem/change reports are used as a management indicator, how will the data be used?
Does the developer's corrective action "focal point" have sufficient authority and technical support to properly direct the corrective action activity, especially if IV&V agents, CRWGs, ICWGs, and CCBs are involved?
Is there a procedure for escalating problems to higher levels of management, if necessary, for timely resolution?

5.12.4 Related guidebook topics.

Independent verification and validation
Problem category and priority classification
Problem/change report
Process improvement
Software configuration management
Software quality assurance

5.13 Cost estimation.

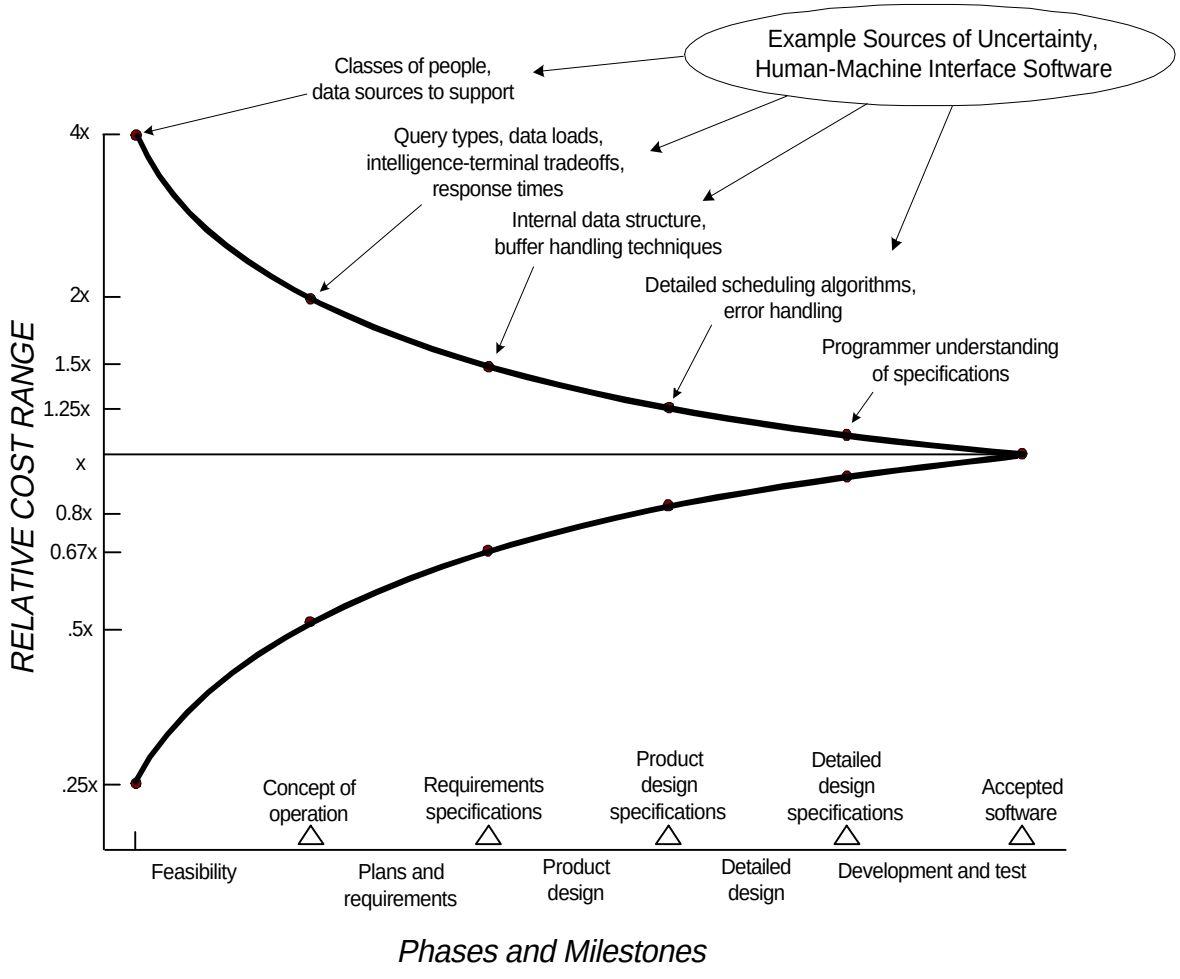
5.13.1 Requirements summary. MIL-STD-498 has no requirements regarding cost estimation.

5.13.2 Acquirer's responsibilities and considerations. Cost estimation may be performed in a variety of ways. Examples of some current cost estimating methods include comparisons between similar systems, engineering estimates, and measurement. The focus of the discussion that follows is on measurement.

Estimating costs for software development can be difficult especially when a system contains software that has (1) never been developed before (unprecedented software) and/or (2) never been developed before by the potential developer. Because these difficulties often exist on projects, cost estimates for software are often grossly underestimated (sometimes by orders of magnitude).

Primary sources of problems with cost estimation are:

- Collecting the developer-unique data needed to form the basis of cost estimation is labor intensive. Hours expended and dollars billed for software are often pooled with other costs and personnel are sensitive to measurements of their productivity.
- Tools need to be calibrated for a particular developer with developer data consisting of program costs, schedules, productivity, etc., over long periods of time.
- Computer-based software cost estimation tools are often focused primarily on estimates for new development. Tools have not supported estimates for modification, enhancement, corrections of existing software, or for reuse and reengineering projects and activities.
- Estimators need experience in developing software for a system that is similar to that being costed.
- In the early phases of a program, there are many sources of uncertainty (e.g., undefined requirements and external interfaces) that become better understood as the program progresses. Figure 36 illustrates this range of uncertainty as a function of program phase. It shows an initial wide band of uncertainty which narrows with time to the point where, eventually, software development costs can be better understood. Unfortunately, the commitment to cost must frequently be made at the time of greatest uncertainty.



* The range of uncertainty figure is based on Figure 21-1 in Software Engineering Economics, by Barry Boehm

FIGURE 36. Range of uncertainty.

Software cost estimation has two main elements. The first element is software size estimation. The second element is the developer's productivity estimate. A good cost estimate is a function of (1) the accuracy of size estimate and (2) the accuracy of the productivity estimate.

Good cost estimate = ∫(accurate size estimate * accurate productivity estimate)

Software size estimation. Software size estimates are generally based on *source lines of code (SLOC)* or on *function points*.

Estimations based on *source lines of code* are best suited to systems that have been coded before or are similar to ones the developer has previously coded. Actual source lines of code usually cannot be accurately estimated through comparisons and engineering estimates when systems are unprecedented or when the developer has had little experience developing software similar to that being estimated.

Estimations based on *function points* are generally recognized as a more front-end approach to software size estimation. Function point estimations are based on user and interface requirements of the system (usually known at the time a cost estimate is needed) rather than lines of code (not known until after the software has been coded). Key steps in function point cost estimation are:

- Identify the requirements for software

  Note: The term "requirements" in descriptions of function points includes information that in MIL-STD-498 may be system or CSCI design. Requirements in MIL-STD-498 are only those characteristics that a system or CSCI must possess to be acceptable to the acquirer. MIL-STD-498's requirements may not provide the detail necessary to support function point estimation. If requirements are minimal or performance based, the estimator may need to include elements of the developer's behavioral and architectural design for the system or CSCI when using function points for estimations.

- Create a Data Flow Diagram or other schematic for representing the user and interface requirements of the system

- Count the system's function points (external inputs, external outputs, external inquiries, internal logical files, and external interface files)

- Compute the weighted sum of function points using the appropriate complexity weighting factors

- Apply a productivity adjustment based upon evaluation of 14 general system characteristics (such as transaction rate, end-user efficiency, complexity of processing, installation ease, and distribution of processors)

Function points were initially used for transaction oriented management information systems. For application to systems that have more algorithmic content, such as military systems, the estimations are modified by adding function points for algorithms and reducing the complexity weighting for internal files.

Productivity estimate. The second element of cost estimation is the productivity estimate based on the developer's personnel, methods/procedures/tools, labor costs, and productivity experience. The developer's productivity adjustment for the project is based on understanding and calibrating the development environment for the project. Some of the key variables are:

- Personnel who will work the project
  - Skill of those who will work
  - Experience with the application area (radar, financial, database, etc.)
  - Experience with the methods/procedures/tools to be used
  - Culture of the working environment
- Methods/procedures/tools to be used for the project
  - Maturity of the methods/procedures/tools
  - Applicability of methods/procedures/tools to the application area
  - Integration of the methods/procedures/tools with other hardware/software for the project
- Cost of labor
  - Geographical location
  - Market competitive position selected
  - National and international economic factors, such as wartime and recessions
- Measured productivity experience
  - SLOC/person-month
  - Function points/person-month

An example of how varying one of the personnel factors, experience with methods/procedures/tools, can impact productivity estimates is through the choice of a programming language. One programmer may be able to code, debug and test 40 lines of COBOL per day, but that same programmer may be able to code, debug and test 100 lines of code in Ada. The reason for the increased productivity may be the programmer's understanding and experience with Ada versus COBOL. More experience with COBOL over time, may increase the programmer's productivity in COBOL. However, note, also, that another language-to-productivity variable is that the number of lines of code that are required to provide a capability varies by language. It might take 100 lines of code in one language to do what can be done in 30 in another.

Some other factors that can create instability in developing a reliable productivity estimate are: personnel turnover, new hardware and software in the software development environment, maturation (or lack thereof) of the methods/procedures/tools, size of the project, etc. Nevertheless, non-attributive individual measurement, over time, can provide a developer with averages for experience and skill categories that can be used to develop more accurate cost estimates.

Lacking historical data on which to base sound cost estimates does not necessarily mean that cost estimates for products need be abandoned. Estimations can be performed progressively. For example:

- Pilot projects (prototypes) using candidate project personnel and methods/procedures/tools can develop a "typical" subset of capabilities to determine the costs for that subset. These costs can then be used to estimate costs for the full set.

- Program strategies (Incremental, Evolutionary, etc.) can be used to develop a system in a series of builds. The first build can be used to estimate costs for subsequent builds.

MIL-STD-498 was designed to support the acquirer in obtaining sound cost estimates by allowing for alternative program strategies. The standard can be applied to prototypes and builds that will be used as a basis for further work, and allows the developer to propose in the SDP for the project the milestones that will best provide an indication of the actual progress for the project based on the methods/procedures/tools used for development.

5.13.3 Additional things to think about.

What is the developer's cost estimation methodology?
What is the developer's experience base using his cost estimation methodology?
What is the developer's measurement program and how much past data does it contain?
How are risk and "TBD" requirements reflected in the developer's estimate?
How does the schedule estimate relate to the cost estimate?
How do different contract types (e.g., fixed price, time and materials, cost plus fixed-fee, etc.) affect the overall cost for a project?
Has the acquirer defined a uniform cost estimation methodology including Work Breakdown Structure (WBS) definitions and other cost status reporting to facilitate cost estimate comparisons?

5.13.4 Related guidebook topics.

Schedules
Software development process
Software development environment
Software engineering environment
Software development methods
Software management indicators

5.14 Critical requirements.

5.14.1 Requirements summary. MIL-STD-498 provides requirements for the developer to identify as safety-critical, security-critical, privacy-critical, or "other"-critical those CSCIs or portions thereof whose failure could lead to violation of those requirements. The standard requires the developer to develop assurance strategies for critical requirements and record those strategies in the SDP; to implement the strategies; and to produce evidence that the strategies have been carried out. References to ***Critical requirements*** in MIL-STD-498 are listed in Figure 37, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.4, 5.19.3, *Figure 2*, B.3.b, Figure 5, *E.4.11*
DBDD	3
IRS	3.y
SDD	3
SDP	4.2.4
SRS	3.18
SSDD	3
SSS	3.18

FIGURE 37. MIL-STD-498's references to ***Critical requirements***.

Critical requirements are to be specified, designed, implemented, tested, evaluated, controlled, and otherwise handled in accordance with MIL-STD-498's software development requirements. (See this guidebook's topics, *Requirements* and *Requirements of the standard*, for more information.) The developer is required to: describe the methods/procedures/tools to be followed for handling all critical requirements in the SDP for the project; assign a project-unique identifier to each requirement; and trace each requirement. (See this guidebook's topics, *Safety* and *Security and privacy*, for details covering those specific types of critical requirements.) Figure 38 lists developer's key activities related to ***Critical requirements***.

Developer's key activities related to ***Critical requirements***	References in MIL-STD-498
Describe the approach to be followed for handling safety assurance, security assurance, privacy assurance, and assurance of other critical requirements. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses in planning for safety, security, and privacy assurance and assurance of other critical requirements.	SDP 4.2.4
Identify as safety-critical, security-critical, and privacy-critical CSCIs those CSCIs whose failure could lead to violation of other requirements deemed critical by the contract or system specifications; develop safety, security, privacy, and "other" assurance strategies and record them in the SDP; implement the strategies; and produce evidence that strategies have been carried out.	4.2.4
Classify a problem as Priority 1 if it jeopardizes safety, security, privacy, or other requirements designated critical.	Figure 5
Specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements.	IRS 3.y SRS 3.18 SSS 3.18
Present database-wide design decisions, including design decisions on the levels and types of availability, security, privacy, and continuity of operations to be offered by the database.	DBDD 3
Present CSCI-wide design decisions, including design decisions regarding the approach to meeting safety, security, and privacy requirements.	SDD 3 SSDD 3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19 Appendix B-G

FIGURE 38. Developer's key activities related to ***Critical requirements***.

5.14.2 Acquirer's responsibilities and considerations. The standard does not specify what is meant by "other" critical requirements. These must be defined by the acquirer, but often include such requirements as mission-related requirements, quality requirements, or support requirements essential to the system or software. Since a developer may not be sensitive to the acquirer's or users' priorities or policies for handling some critical requirements, the standard provides for assigning precedence, criticality, or weights to requirements. For example, the acquirer may require that specific types of tests or demonstrations be performed on software containing specific types of critical requirements, or may require inspections of critical software to determine that mischievous or malicious code does not exist within the software. For operational or mission essential critical requirements, DoD policy requires that the system not possess any known Priority 1 or 2 problems, as described in Figure 5 of MIL-STD-498, prior to the start of Operational Test and Evaluation (OT&E), and that Priority 3 problems be documented with appropriate impact analyses.

The standard does not impose safety, security, privacy, or other critical requirements on a system. It does have requirements regarding any safety, security, or privacy requirements that may be imposed or that may exist. The developer is required to identify as safety, security, or privacy critical CSCI's those CSCIs or portions thereof whose failure could lead to a hazardous system state (one that could result in unintended death, injury, loss of property, or environmental harm) or whose failure could lead to a breach of security and/or privacy.

When safety, security, privacy, or other critical requirements are imposed on a system or its software, the acquirer may want to hold discussions or reviews specifically targeted on specific types of requirements, such as safety or reliability. MIL-STD-498 groups all such reviews generically, calling the group, "critical requirement reviews." Other names may be used to clarify the intent of each such review to be held for the project. By holding specialized discussions, the right people can be assembled to concentrate on a key area at the right time. Although critical requirements may be addressed in separate discussions, they should not be ignored when discussions take place regarding other aspects of the system or its software.

5.14.3 Additional things to think about.

Has the acquirer specified all critical requirements?
How will the developer/acquirer team be able to avoid or resolve Priority 1 and 2 problems to proceed to OT&E?
Have critical requirements been prioritized correctly?
Have all interested parties (user, support agency, security, safety, privacy, etc.) been involved in specifying critical requirements?
Have all critical CSCIs been identified?
Are joint technical reviews scheduled for critical requirements? Have all key players been invited?

5.14.4 Related guidebook topics.

Problem category and priority classification
Requirements
Requirements of the standard
Safety
Security and privacy
Testing (Developer-internal)
Traceability

5.15 Data accession list.

5.15.1 Requirements summary. MIL-STD-498 provides no requirements relating to use of a Data Accession List (DAL).

5.15.2 Acquirer's responsibilities and considerations. The DAL is meant to provide the acquirer a means for legal access to, and delivery of, data that an acquirer does not know is needed at contract time. The DAL is not intended as a means to avoid determining up front what deliverable documentation is needed for a project. The description of DAL content and format is found in DID, DI-MGMT-81453. When placed on the CLIN or CDRL, the DAL requires the delivery of a list of data generated during the course of a development project. Included on this list (depending on contract provisions) may be information that has not been specified in the contract as deliverable, such as MIL-STD-498's DIDs contents, drawings, design notebooks, software development files, utility software, configuration management listings, memoranda, interoffice communications, letters and other communications sent to or received from subcontractors and vendors, and other internal data associated with the project.

The DAL requires the developer to list the identification number, title (describing the content), in-house release date, and data rights associated with each item on the list. If an acquirer wants delivery of data listed on the DAL that is not specified for delivery elsewhere in the contract, the developer can charge for costs such as: preparing the document or software for delivery, reproduction, shipping, handling, etc. There are no format and content requirements provided for the data in the list and the acquirer does not receive any approval rights to that data. The DAL's lack of requirements for format, content, and approval does not override requirements of the contract, such as a document specified for delivery in accordance with a DID, nor does it negate approval rights that may be required for such contract deliverables.

When a developer will be creating non-deliverable utility or other non-deliverable software for use during the development and testing of deliverable software, an acquirer might want to place the DAL DID on a CLIN or CDRL to require the developer to list that software. The acquirer can then review the listed items to determine whether any of those items that would be of benefit during operation or support should be delivered. To place the DAL on contract, the acquirer uses a CLIN or completes the CDRL form, DD Form 1423. (See this guidebook's topic, *Contract data requirements list*, for more information and a sample DD Form 1423.) The acquirer should determine when the list should be delivered (monthly, quarterly, at major milestones, etc.), if each delivery should contain all data, or if subsequent lists should contain a listing only of new data and new version/releases (to preclude the repeated and costly delivery of duplicate information).

To properly place the DAL on contract, a reference to FAR § 52.227-16 needs to be included in the contract. This clause stipulates that the acquirer may order any data first produced or specifically used in the performance of the contract at any time during contract performance or within a period of 3 years after acceptance of all deliverable items.

5.15.3 Additional things to think about.

How can the acquirer plan a budget for DAL data when the quantity of data that might be needed is not known?
Will other than restricted rights be needed for any software that might be obtained via the DAL?
Is there a need for information to support acquirer conducted activities, such as audits and other legal fact-finding activities, that is not specified on the contract as a deliverable?

5.15.4 Related guidebook topics.

Access for acquirer review
Contract data requirements list
Documentation (Recording information)
Software development files

5.16 Database design.

5.16.1 Requirements summary. MIL-STD-498 provides requirements regarding database design. The standard's software development activities cover the development of databases as well as software programs. Although other definitions of software may exclude databases, MIL-STD-498 defines "software" as "*computer programs and computer databases*." This definition encompasses developing database management systems (DBMSs), creating data definitions, building databases based on commercial, in-house, or project-unique DBMSs, as well as populating databases and other files with data. To provide added visibility to some of the database activities associated with software development, the standard makes specific references to databases and database design in the body of the standard and its software products. The standard's references to databases and data files are discussed under this guidebook's topic, *Databases*. References to ***Database design*** in MIL-STD-498 are listed in Figure 39, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.14, 3.15, 3.32, 3.45,* 4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.6.3, 5.13.4, 5.13.5, *6.2*, Figure 3, Figure 4, Figure 6, *Figures 9-12, Figure 15, Figure 17*
DBDD	All
SDD	3, *4.1.a Note*, 5, 5.x, 6
SDP	5.4.1, 5.6.1, 5.6.3
SPS	5.1, 5.3
SRS	3.12
SSDD	3, *4.1.a Note*
SSS	3.12

FIGURE 39. MIL-STD-498's references to ***Database design***.

MIL-STD-498 provides the flexibility to view a database as a system on its own, a subsystem within another system, a single CSCI, or a software unit or part of a software unit. Database development can range from development of a total database management system with all its software application data and programs to some subset in the development of a simple software program, such as defining data definitions or populating files with data. Because of this wide variety, all the requirements in the standard might be applied on a database project or selected activities applied to the project's data.

Database design

Requirements for a database are specified in the System/Subsystem Specification (SSS), the Software Requirements Specification (SRS), and possibly associated Interface Requirements Specifications (IRSs). MIL-STD-498's Database Design Description (DBDD) provides the information for design of a database using a DBMS. (The design information for a DBMS, if developed as a CSCI specifically for the project, is typically provided in the Software Design Description (SDD) and relevant Interface Design Description (IDD).)

No one standard approach (methods/procedures/tools) has been agreed upon to describe data design. The DBDD is, therefore, designed generically to accommodate development of databases in a rapidly changing environment. Since various methods often use differing terms to describe similar concepts or use similar terms differently, the DBDD identifies, as examples only, the conceptual, internal, logical, or physical levels, as possible design levels or views of data. Some methods require one or two levels of representation, while others require five or more levels or may break a representation into views to show how each "user" sees the data (each of which may have its own levels or views of the data). The DBDD provides the flexibility to describe as many levels/views as needed. The DBDD provides information regarding database-wide and detailed design including:

- Queries/other inputs the database will accept, outputs to be produced
- Database behavior in response to each query or input
- How database/data files will appear to the user
- Database management system to be used
- Levels/types of availability, security, privacy, continuity of operations
- Distributed databases including updates/maintenance, synchronization, enforcing integrity/business rules
- Backup and recovery (including data and process distribution strategies)
- New and non-standard technologies such as video and sound
- Repacking, sorting, indexing, synchronization, and consistency including automated disk management and space reclamation, optimizing strategies, storage/size, database population/legacy data
- Conceptual, internal, logical, physical or other levels/views/perspectives of design
- Characteristics of data and data assemblies from those levels/views/perspectives

To ensure that the acquirer has the opportunity to review and approve the selected methods, the developer is required to describe the approach to database design in the SDP for the project. The description in the SDP includes the methods/procedures/tools and standards selected for developing the design. The DBDD provides for recording/referencing the design conventions needed for understanding the various design representations. Figure 40 lists developer's key activities related to ***Database design***.

Developer's key activities related to ***Database design***	References in MIL-STD-498
Describe the approach to be followed for participating in system-wide design decisions and for performing CSCI-wide design decisions and database design. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses regarding the design decisions and database design.	SDP 4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.6.3
Develop and apply standards for representing design. Describe or reference the standards to be followed in the SDP.	4.2.2
Identify, evaluate, and include reusable software products in the database design. Allocate requirements to candidate reusable CSCIs and software units.	4.2.3.1, Figure 3
Define and record CSCI-wide design decisions (decisions about the CSCI's behavioral design and other decisions affecting the selection and design of the software units comprising the CSCI). Incorporate design constraints specified in the SRS.	5.6.1 DBDD 3 SSDD 3 SSS 3.12
Describe the design of the database and the software units used to access or manipulate the data. Provide traceability from each software unit to the CSCI requirements allocated to it and from each CSCI requirement to the software unit(s) to which it is allocated.	5.6.3 DBDD 4, 5, 6 SDD 5, 5.x, 6
Update the design description of each CSCI to match the "as built" software. Include all applicable items in the qualification, software support, and traceability sections of the SPS DID.	5.13.4 SPS 5.1
Participate in updating the system design description to match the "as built" system. Include all applicable items in the SSDD DID.	5.13.5
Describe procedures that must be followed to modify the CSCI, including information on databases/data files used by the CSCI and procedures for using and modifying them.	SPS 5.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19 Appendix B-G

FIGURE 40. Developer's key activities related to ***Database design***.

In addition to the design of the data, the DBDD also provides for description of detailed characteristics of data elements and data assemblies. This set of characteristics is used consistently across all of the standard's software product descriptions. This consistency allows a developer to record information in one place (such as a project database or CASE tool) and cross-reference this information from other documents to reduce redundancy and possible incompatibility when one development group on the project describes a data characteristic already described differently by another. If a data description has been published in a standard data element dictionary specified in the contract, the standard states that reference to an entry in that dictionary is preferred over including the description itself in the DBDD or IDD. By referencing the standard data description, the development team members can recognize the "standard" nature of the referenced description and avoid preparing unique descriptions when standard ones already exist for use.

5.16.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides a wide range of options for recording information about databases. Each of the design documents has room to describe data and its design. How a project selects which document or set of documents can depend on many factors including:

- Size of the system
- Use of the data/database within that system (i.e., one versus many programs that access/manipulate the data)
- Security/privacy requirements for accessing/manipulating the data
- Use of COTS software for the DBMS

For example, if data is used by two or more systems, subsystems, or CSCIs, the developer may describe the detailed data characteristics in an IDD. However, if coders, testers, or maintainers need a description of the data and the programs that access/manipulate the data/database, either the DBDD or the SDD may prove to be a better choice.

5.16.3 Additional things to think about.

Are the support organization's concerns or needs regarding the database development environment addressed?
Is database design a topic in appropriate joint technical reviews?
What personnel and resources are required to review the detailed design?
Have key decisions regarding database design been recorded?
Are data distribution issues addressed?
Are data integrity (ensuring that the data in the database is accurate) issues addressed?
What performance measures are evaluated for a COTS DBMS?
What resource utilization requirements affect database design and performance?
Have all legacy data been identified?
Has domain analysis been performed to ensure that all data and user programs have been identified?
Is an Interface Control Working Group (ICWG) needed to coordinate and obtain agreements regarding interfaces between systems?
Are data security and privacy issues addressed?
How many levels/views of the data design should be delivered? How many should be reviewed by others? When?
Are there data standardization issues across the enterprise that need to be surfaced/addressed?

5.16.4 Related guidebook topics.

Architectural design
Behavioral design
Commercial-off-the-shelf software products
Databases
Data standardization
Detailed design
Documentation (Recording information)
Interfaces
Rationale/key decisions
Requirements
Security and privacy
System/subsystem-wide and CSCI-wide design

5.17 Databases.

5.17.1 Requirements summary. MIL-STD-498 defines "software" as "*computer programs and computer databases*" and contains wording throughout the standard and DIDs to emphasize that the software development requirements apply to databases as well as to other software. All planning, requirement analysis, design, testing, integral process, etc., activities are intended to include databases, data files, and data. References to ***Databases*** in MIL-STD-498 are listed in Figure 41, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.14*, *3.15*, *3.32*, *3.45*, 5.7.1, 5.12.1, 5.13.1, 5.13.2, *6.4*, Figure 4, *Figures 9-12*
DBDD	All
SCOM	3.2, 3.3.c, 5.5.x.3
SDP	5.7.1, 5.12.1, 5.13.1, 5.13.2
SIOM	3.2, 3.4.a, 5.1
SIP	4.x.5.c
SPS	3.1, 3.2
SRS	3.5, 3.10.3, 3.12.a
SSS	3.5, 3.10.3
STD	4.x.y.6.a
STP	3.x.1
SUM	3.2, 3.3.c

FIGURE 41. MIL-STD-498's references to ***Databases***.

MIL-STD-498's software development requirements cover the design of databases and database management systems (DBMSs). When a DBMS is developed, the design for the DBMS itself is described in a Software Design Description (SDD). The Database Design Description (DBDD) is intended for use with a DBMS to describe the data design, storage, and data element characteristics, and provide the design descriptions of the procedures, menus, forms, etc., that manipulate or access those data stores. (See this guidebook's topic, *Database design,* for more information.) Figure 42 lists developer's key activities related to ***Databases***.

Developer's key activities related to ***Databases***	References in MIL-STD-498
Describe the approach to be followed for defining and recording databases and data files, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses in planning for defining and recording databases and data files.	SDP 5.7.1, 5.12.1, 5.13.1, 5.13.2
Specify the requirements, if any, imposed on databases and data files. Specify the requirements, if any, regarding databases or database management systems that must be used or incorporated into the software.	SRS 3.5, 3.10.3, 3.12.a SSS 3.5, 3.10.3
Implement software corresponding to each software unit, including, as applicable, building databases, populating databases and other data files with data values, and other activities needed to implement the design.	5.7.1
Identify databases and data files necessary to perform planned test activities. Include in test procedures, as applicable, the commands needed to modify database/data files.	STD 4.x.y.6.a STP 3.x.1
Use database/data file as a category for classifying problems in software products.	Figure 4
Identify all software files, including databases and data files, that must be installed for the software to operate. Describe database and data file format, content, purpose, etc.	SCOM 3.2, 3.3c, 5.5.x.3 SIOM 3.2, 3.4.a, 5.1 SUM 3.2, 3.3.c
Provide step-by-step procedures for the installation of software at user sites, including procedures for initializing databases with site-specific data.	SIP 4.x.5.c
Prepare/deliver the executable software and the source files for each user site and the support site, including any data files needed to install and operate the software on its target computer(s) and any data files needed to regenerate the executable software.	5.12.1, 5.13.1, 5.13.2 SPS 3.1, 3.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 42. Developer's key activities related to ***Databases***.

5.17.2 Acquirer's responsibilities and considerations. MIL-STD-498's numerous references to databases and data files are meant to ensure that all necessary data and data files are identified, described, developed, tested, and properly installed at user and support sites. Database references are also intended to remind the developer to consider databases as part of each software development activity. The development of databases, data files, and the data to populate those databases/data files can represent a significant part of the software development. In some cases, such as developing topographical databases, the development of data and data files can constitute the entire software development.

Testing the data in a database is an area sometimes overlooked in development. Testing with "live" data, and ensuring the integrity and accuracy of the data in the system can involve safety, security, and privacy issues as well as technical ones.

In addition to the methods/procedures/tools to be used for development of databases, the acquirer should evaluate the developer's approach to database development as described in the SDP for the project to determine whether:

- Data normalization methods/procedures, that is, the restructuring of complex data structures into simpler (flat) ones, are sufficient to ensure ease of maintenance
- Analysis activities are sufficient to surface all data needs (external and user data needs as well as internal and algorithmic data needs or business rules necessary to process the data correctly)
- Accuracy, precision, reliability, and integrity issues regarding data have been addressed

The standard's joint technical and management reviews may be used to raise and discuss issues regarding databases/data files and data as well as other issues.

5.17.3 Additional things to think about.

Are all databases and data files described in the applicable SDDs/IDDs?
Are data distribution issues for distributed, client-server, and/or networked databases being addressed?
Are database integrity (accuracy of the data) issues addressed? Are safety, security, and privacy personnel involved in database review?
Have all legacy data been accounted for prior to data conversion?
Have all data conversion issues been adequately addressed?
Do software development and support processes address configuration management controls for data?
Has domain analysis been performed to ensure that all data and user programs have been identified?
Are there data standardization issues across the system or enterprise that need to be surfaced/addressed?

5.17.4 Related guidebook topics.

Architectural design
Behavioral design
Database design
Data standardization
Detailed design
Interfaces
Security and privacy
System/subsystem-wide and CSCI-wide design

5.18 Data rights.

5.18.1 Requirements summary. MIL-STD-498 provides requirements concerning identification of data rights. "Data rights" or "rights in technical data and computer software" are terms used to describe the contractually and legally binding limitations (if any) to copy or reproduce and distribute technical data and computer software. MIL-STD-498's general requirements (Section 4) state the rights for reusable software products incorporated in the deliverable software must meet the rights specified in the contract. MIL-STD-498's Appendix B lists rights as one evaluation criterion (among others closely associated with rights, such as licenses or other fees applicable to each copy, and warranties) that might be considered when determining the suitability of reusable software for incorporation.

In addition, all of the DIDs require the developer to identify data rights limitations or restrictions on the title page of traditional documents. When information, software, or data resides in a database or other alternative form, the limitations or restrictions are to be included on external or internal labels or equivalent identification methods. For identification of data rights limitations and restrictions on deliverable executable software and source files, the Software Product Specification (SPS) requires external and internal labels for the files and media on which the files reside. References to ***Data rights*** in MIL-STD-498 are listed in Figure 43, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.3.1, B.3, Figure 3
All DIDs	10.1.c
SDP	4.2.3.1
STP	3.x.4
STrP	3.2, 3.3, 3.4

FIGURE 43. MIL-STD-498's references to ***Data rights***.

The developer is required to describe the approach to be followed for meeting all data rights requirements in the SDP for the project. The Software Test Plan (STP) requires the description of the proprietary nature, acquirer's rights, and licensing issues associated with each element of the software test environment. The Software Transition Plan (STrP) requires the identification and description of the hardware/software and associated documentation, and other documentation needed to support the deliverable software. For each element needed to support the deliverable software, the developer is required to provide information about licenses, usage, and rights. Computers, peripheral equipment, hardware simulators, CASE tools, data in those tools, compilers, test data, databases, and documentation needed for support of the deliverable software (e.g., plans, reports, studies, specifications, user/operator

manuals) are all to be identified and described. The intent of MIL-STD-498's requirements is to ensure that the developer identifies rights for each deliverable software product to avoid inadvertent rights problems.

5.18.2 Acquirer's responsibilities and considerations. In June 1995, after the publication of MIL-STD-498, DoD significantly changed its regulations on rights in technical data and computer software. The new regulations base DoD's rights in technical data and computer software on: (1) a distinction between which party funded the development, and (2) whether the technical data or computer software is deliverable to the government. Since MIL-STD-498 requires the developer to comply with the data rights specified in the contract, there is no conflict between the new DFARS rights in technical data and computer software and MIL-STD-498.

However, the DoD has clarified rights in technical data and computer software by redefining "computer software," "documentation," and "commercial computer software." The new DFARS definitions of computer software and documentation work well for rights in technical data and computer software. They do not work as well, however, in the context of the standard. The standard makes a distinction between the products that can be translated by a computer but are not readily understood by humans (software), and the descriptions of those products that can be read and understood by humans (documentation). These new definitions have a potential to cause difficulties if applied to paragraphs in the standard other than those concerned with rights in technical data and computer software.

The new DFARS definitions are as follows.

> **Computer software:** "... *computer programs, source code, source code listings, object code listings, design details, algorithms, processes, flowcharts, formulae, and related material that would enable the software to be reproduced, recreated or recompiled* ..." DFARS § 252.227-7013(a)(3) (June 1995)).
>
> **Documentation**: "... *owner's manuals, user's manuals, installation instructions, operating instructions, and other similar items* ..." (DFARS § 252.227-7013(a)(4) (June 1995)).
>
> These contrast with MIL-STD-498's definitions.
>
> **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*." (October 1988)

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*."

The DFARS definitions are to be used when interpreting those clauses that call for the identification of data rights, but should not be used when interpreting the technical specification clauses in the standard. For example, "*integrating the software corresponding to two or more software units*" means integrating the computer programs and data/databases, NOT integrating the descriptions of the computer programs and data/databases (i.e., MIL-STD-498's documentation).

DoD's rights in technical data and computer software are intended to apply to "deliverables," but are not intended to apply to software and technical data intended to support the development under the contract that are not to be delivered. Furthermore, DoD has clarified that all rights that are not granted to the government are retained by the contractor, unless the contract provides otherwise.

The revisions to the regulations on rights in technical data and computer software are based on trade-offs between the government's need to protect its ability to procure products at a reasonable cost as well as to modify, correct, or enhance the products that it buys against the developer/vendor's need for protection against piracy. The new regulations are based on the distinctions between developer and acquirer funded development or some mixture of funding and whether or not the products can be classified as "commercial computer software."

The new definition of "*commercial computer software*" is greatly expanded from the previous definition which narrowly defined commercial computer software as "*developed at private expense for the commercial marketplace and is not in the public domain*." The new definition includes software developed or regularly used for nongovernmental purposes which:

1. Has been sold, leased, or licensed to the public
2. Has been offered for sale, lease, or license to the public
3. Has not been offered for sale, lease, or licensed to the public but will be available for commercial sale, lease, or license in time to satisfy the delivery requirements of the relevant contract
4. Satisfies a criterion expressed in 1, 2, or 3 and would require only minor modification to meet the requirements of the relevant contract

The DFARS provides clauses defining the differing government rights. For commercially available computer software and documentation, DoD acquirers are to require only the license customarily provided to the public, unless there is a need for greater rights. (Note that some customary rights may be modified under DFARS § 252.211, Commercial Items.) The DFARS defines specific rights such as unlimited rights, government purpose rights, and limited rights. "Unlimited rights" means that the government can do whatever it wants with the products. For those products with less-than-unlimited rights, some restrictions apply. The DoD makes a subtle distinction between government purpose rights, which allows the government to provide the products to another contractor for "government purposes," such as a need to modify, enhance, or obtain bids, and "limited rights" which restricts the products to government use only. Figure 44 shows the relationship of rights to the source of the funding.

Funding Source	Government Rights			
	Noncommercial Technical Data	Noncommercial Computer Software & Documentation	Technical Data - Commercial Items	Commercial Computer Software
Developed Exclusively at Private Expense	Limited Rights	Restricted Rights (software) and Unlimited Rights (documentation)	Limited Rights	Customary License
Developed Exclusively at Government Expense	Unlimited Rights	Unlimited Rights	N/A	N/A
Mixed Funding	Government Purpose Rights	Government Purpose Rights	N/A	N/A

FIGURE 44. Relationship of rights to the source of funding.

The clauses applicable to rights in technical data and computer software in the DFARS, June 1995, are as follows:

- DFARS § 252.227-7014, Rights in Noncommercial Computer Software and Noncommercial Computer Software Documentation
- DFARS § 252.227-7015, Technical Data -- Commercial Items
- DFARS § 252.227-7013, Rights in Technical Data -- Noncommercial Items.

Additional provisions in DFARS § 227.7103-14, June 1995, provide requirements regarding conformity, acceptance, and warranty of technical data. The DFARS clarifies that software documentation is technical data (DFARS § 227.7203-14(a)) and references 10 U.S.C. 2320 provisions that:

- *Require contractors to furnish written assurance, at the time technical data are delivered or are made available to the Government, that the technical data are complete, accurate, and satisfy the requirements of the contract concerning such data;*

- *Provide for the establishment of remedies applicable to technical data found to be incomplete, inadequate, or not to satisfy the requirements of the contract concerning such data; and*

- *Authorize agency heads to withhold payments (or exercise such other remedies an agency head considers appropriate) during any period if the contractor does not meet the requirements of the contract pertaining to the delivery of technical data.*

Furthermore, DFARS § 227.7203-14.b.2.(i) and (ii) provide instructions regarding warranties. These DFARS sections state:

(i) Weapon systems. Computer software that is a component of a weapon system or major subsystem should be warranted as part of the weapon system warranty. Follow the procedures at 246.770.

(ii) Non-weapon systems. Approval of the chief of the contracting office must be obtained to use a computer software warranty other than a weapon system warranty. Consider the factors at FAR 46.703 in deciding whether to obtain a computer software warranty. When approval for a warranty has been obtained, the clause at 252.246-7001, Warranty of Data, and its alternates, may be appropriately modified for use with computer software or a procurement specific clause may be developed.

5.18.3 Additional things to think about.

Does the government have adequate rights to meet life cycle needs? Have users and uses been considered, such as, "Who will need to use the products? Who will support the products?"
Have rights in technical data and computer software been clarified to cover the case if software is incomplete when work stops?
What means, (e.g., escrow accounts, warranties, or maintenance contracts), might be used to protect an acquirer of commercial computer software?

5.18.4 Related guidebook topics.

Commercial-off-the-shelf software products
Contract
Licenses (Software)
Reusable software products
Software development environment
Software transition
Source files
Subcontractor management

5.19 Data standardization.

5.19.1 Requirements summary. MIL-STD-498 provides no requirements to follow DoD's data standardization program as defined in DoDD 8320.1, DoD Data Administration, DoD 8320.1-M, Data Administration Procedures, and DoD 8320.1-M-1, Data Element Standardization Procedures. It does, however, have requirements regarding data. MIL-STD-498 supports DoD's data standardization program on several levels. It supports the data element standardization program through:

- Reminding the developer that if a data description required by a DID has been published in a standard data element dictionary specified in the contract, reference to that dictionary is preferred over including the description itself
- Reminding the acquirer that if electronic formats, such as ASCII or CALS, are to be used for deliverables, the selected format should be specified on the CLIN or CDRL
- Provides requirements for describing the characteristics of data elements in the system
- Provides requirements for standard document structure, format, and content to enable CALS templates to be created for the standard's manuals and other deliverable documents
- Provides requirements for standard document content for twenty-two document types needed for software development
- Provides requirements for standard information to be recorded about software development on a project when that information is recorded in databases, CASE tools, integrated CASE tools, or other methods

References to ***Data standardization*** in MIL-STD-498 are listed in Figure 45.

All DIDs	*10.1.h*

FIGURE 45. MIL-STD-498's references to ***Data standardization***.

5.19.2 Acquirer's responsibilities and considerations. All of MIL-STD-498's design and specification DIDs (Database Design Description (DBDD), Interface Design Description (IDD), Interface Requirements Specification (IRS), Software Design Description (SDD), Software Requirements Specification (SRS), System/Subsystem Design Description (SSDD), and System/Subsystem Specification (SSS)) contain an identical list of characteristics regarding data relevant to the project. The standardization of data characteristics throughout the standard promotes data standardization on the project. When required by contract to use

DoD's standard data elements, the project can reference those standard data elements for the data in the project.

MIL-STD-498's list of data element characteristics and those of the Data Administration are nearly identical. MIL-STD-498's list of data element characteristics does not contain the requirements of DoD's standardization program for administrative information about the data element. MIL-STD-498 also contains a project-unique identifier that is not found in the DoD's standard data element program (project-unique identifier). Before submitting any data element descriptions to the DoD's Standardization Program Office for approval, the acquirer will need to supply the additional information required and reconcile any differences.

5.19.3 Additional things to think about.

Are DoD directives and instructions for data standardization (e.g., DoDD 8320.1) applicable to the project?
Have applicable standard data element characteristics been used or referenced?
Are there data standardization issues across systems/enterprise/organization that need to be surfaced/addressed?
Will other programs ever want to use this system's data? Is there enough data that data conversion costs will be substantial when this system needs to be replaced?

5.19.4 Related guidebook topics.

Database design
Databases
Interfaces

5.20 Detailed design.

5.20.1 Requirements summary. MIL-STD-498 provides requirements regarding detailed design. The standard defines design as "*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*." Detailed design is that aspect of design applicable to the software unit or group of software units. (See this guidebook's topics, *Architectural design, Behavioral design, Database design*, and *System/subsystem-wide and CSCI-wide design*, for more information regarding software design.) References to ***Detailed design*** are listed in Figure 46, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.45*, 4.2.2, 4.2.3.1, 5.6.3, 5.7.2, 5.7.3, 5.7.4, 5.8.1, 5.8.2, 5.8.3, 5.13.4 Figure 3, Figure 6, *E.4.6*
DBDD	5
IDD	3, 4
SDD	5, 6
SDP	4.2.2, 4.2.3.1, 5.6.3, 5.13.4
SPS	5.1, 5.3.c

FIGURE 46. MIL-STD-498's references to ***Detailed design***.

Detailed design covers the "nuts and bolts" of the software to be developed to implement the architectural and component-wide design. Detailed design includes: design decisions, such as algorithms needed in addition to any specified; constraints, limitations, or unusual features in the design of the software unit; programming language if other than the specified CSCI language; procedural commands (such as menu selections in a database management system for defining forms and reports); inputs, outputs, and other data elements and data element assemblies; logic to be used including, as applicable, conditions in effect within the software unit when its execution is initiated, conditions under which control is passed to other software units; response and response time to each input (including data conversion, renaming and data transfer operations); and sequence of operations and dynamically controlled sequencing during the software unit's operation, exception and error handling. Figure 47 lists developer's key activities related to ***Detailed design***.

Developer's key activities related to ***Detailed design***	References in MIL-STD-498
Describe the approach to be followed for detailed design, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses for detailed design.	SDP 4.2.2, 4.2.3.1, 5.6.3, 5.7.2, 5.7.3, 5.7.4, 5.8.1, 5.8.2, 5.13.4
Develop and apply standards for representing design. Describe or reference the standards to be followed in the SDP.	4.2.2
Define and record CSCI and database detailed design. Incorporate design constraints specified in SRSs/IRSs.	5.6.3 DBDD 5 SDD 5
Describe the interface characteristics of systems, subsystems, configuration items, manual operations, or other system components.	IDD 3
Trace from each software unit to the CSCI requirements allocated to it. Trace from each CSCI requirement to the software units to which it is allocated.	DBDD 6 SDD 6
Trace from each interfacing entity to the system or CSCI requirements addressed by the entity's interface design. Trace from each system or CSCI requirement that affects an interface to the interfacing entities that address it.	IDD 4
Identify, evaluate, and include applicable reusable software products in the software design. Allocate requirements to candidate reusable CSCIs and software units.	4.2.3.1, Figure 3
Establish test cases, test procedures, and test data covering all aspects of the unit's detailed design. Test the software corresponding to each software unit in accordance with the unit test cases and procedures. Analyze and record results of unit testing and record test and analysis results in appropriate software development files.	5.7.2, 5.7.3, 5.7.4
Establish test cases, test procedures, and test data for conducting unit integration and testing. Integrate and test the software in accordance with the unit integration test cases and procedures. Analyze and record results of unit integration testing and record test and analysis results in appropriate software development files.	5.8.1, 5.8.2, 5.8.3
Update the design description of each CSCI to match the "as built" software.	5.13.4 SPS 5.1
Describe procedures that must be followed to modify the CSCI, including design, coding, or other conventions to be followed.	SPS 5.3.c
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 47. Developer's key activities related to ***Detailed design***.

5.20.2 Acquirer's responsibilities and considerations. Software's detailed design is described through entities called software units. Descriptions of software units are intended to be made iteratively, as needed, to provide the design details of the CSCI (i.e., software units may be composed of other software units). The design can be developed top-down, bottom-up, middle-out, or in a structured or object-oriented manner. The iteration will proceed until all software units are designed to a level from which programmers can develop code. (See this guidebook's topic, *Software unit,* for more information.)

MIL-STD-498's terminology for describing the entities (i.e., software unit) was selected to accommodate object oriented methods and Ada for which the Computer Software Component (CSC) and Computer Software Unit (CSU) scheme of DOD-STD-2167A did not work well. Developers whose methods do work with a CSC/CSU structure can continue to use them, substituting the terms CSC and CSU for software unit, as appropriate.

Different design methods and different programming languages can result in detailed design that is organized along different lines. The acquirer can expect that the detailed design of a program to be developed in accordance with object oriented methods in Ada to appear different from one following structured analysis techniques in assembly language and for software units to be named differently. For example, object oriented methods will call some of the design pieces objects and classes of objects rather than software units. The developer is required to describe how the detailed design is to be organized for the project, what software units are to be called, as well as a description of the standards to be used to represent that design in the SDP for the project.

If system/subsystem-wide and CSCI-wide design can be thought of as a view from the "outside" (e.g., the user's view), detailed design can be thought of as a view from the "inside" (e.g., what a designer will need to know to design the next level of the design, or what a programmer will need to know to write the code for that software unit). Detailed design is intended to implement all of the design details provided by system/subsystem-wide and CSCI-wide design and architectural design of the CSCI. Detailed design is intended to be the transformation of higher-level design abstractions into design a programmer can understand and transform into code. Note that when performing traceability, 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.

Detailed design of the software units used for database access or manipulation may be documented in the Database Design Description (DBDD) or the Software Design Description (SDD). For example, if the database is small, or closely related to other software that will be

developed and described in an SDD, it may make more sense to provide the description of the database together with the descriptions of other software units in an SDD. Note also that interface data may be described in the SDD or in an Interface Design Description (IDD), depending upon project preferences and needs. Figure 48 shows detailed design and other design elements and their relationship to requirements.

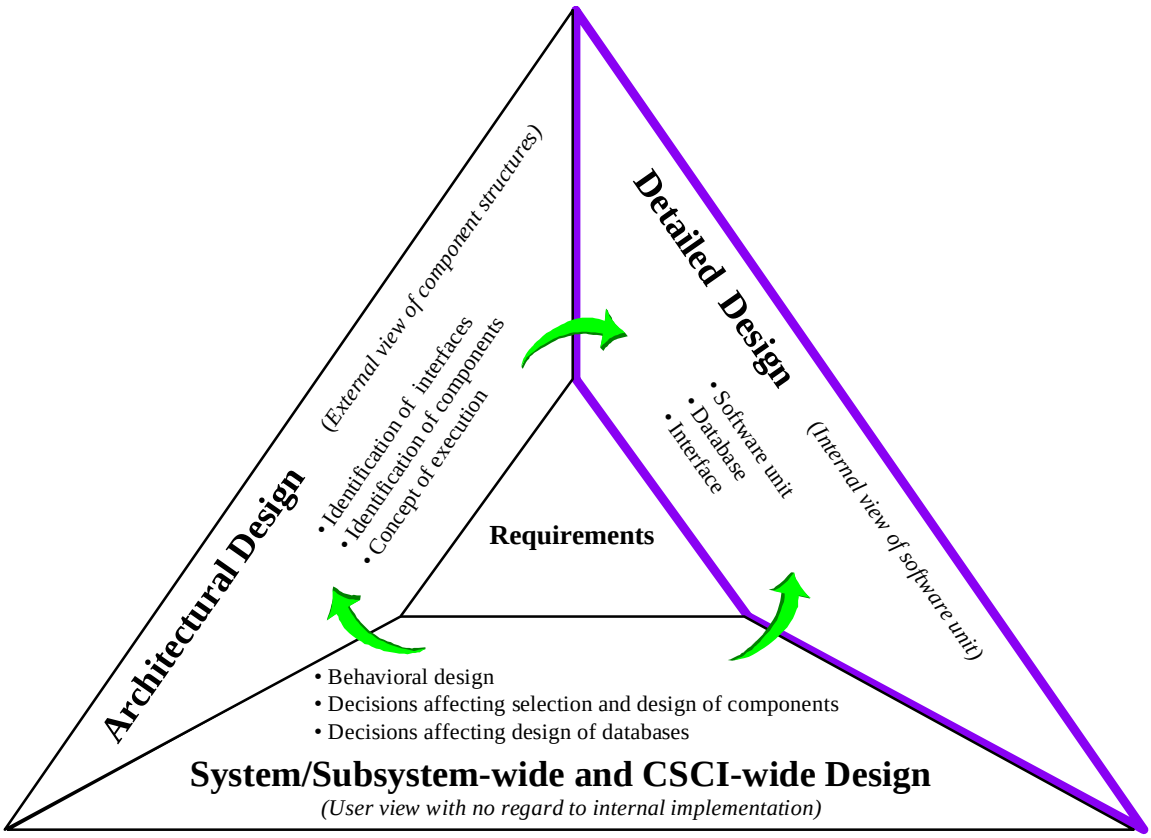


FIGURE 48. Detailed design and other design elements.

5.20.3 Additional things to think about.

Is detailed design a topic in appropriate joint technical reviews?
Have the "as built" detailed designs been updated?
Do system/CSCI specifications impose unnecessary constraints on the detailed design?
What personnel and resources, if any, are required to review the detailed design?
Have key decisions regarding detailed design been recorded?

5.20.4 Related guidebook topics

Architectural design
Behavioral design
Database design
Interfaces
Software development methods
Software unit
System/subsystem-wide and CSCI-wide design
Traceability

5.21 Documentation (Preparing documents).

5.21.1 Requirements summary. MIL-STD-498 provides requirements for preparing documents. The standard makes a distinction between preparing documents and the development and recording of planning and engineering information. A key concept in MIL-STD-498 is that development and recording of planning and engineering information (i.e., documentation) is an intrinsic part of the software development process, to be performed regardless of whether a deliverable document is required. (See this guidebook's topic, *Documentation (Recording information)*, for more information.) References to ***Documentation (Preparing documents)*** in MIL-STD-498 are listed in Figure 49, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 3-4, 1.2.3, 3.19*, *6.2, 6.4, 6.5*, D.4.4, *G.5, Appendix H, I.4, Figures 15-17*
All DIDs	*Block 7*, 10.1

FIGURE 49. MIL-STD-498's references to ***Documentation (Preparing documents)***.

A second key concept regarding documentation in MIL-STD-498 is that documentation need not be in the form of traditional documents. The standard provides the following definition:

> "*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*."

This definition opens the way for documentation to be represented in CASE tools as well as providing for delivery in alternative media, such as diskettes, CD-ROM, video, or tapes. The task of formatting, assembling, marking, copying, and distributing documents according to the CLIN or CDRL is recognized as separate from the task of generating and recording the information and requires additional time and effort on the part of the developer.

MIL-STD-498 encourages the use of automated techniques and allows the substitution of commercial or other existing documents, including other project plans, for all or part of a document if they contain the required data.

The standard's requirements for preparing deliverable documentation are found in section 10.1 of each DID. Italics have been used to highlight those portions applicable only when project information resides in a database or other alternative form. Brackets are used to indicate supplemental information not found in MIL-STD-498 to supplement the instructions when using a CLIN. The following guidance and requirements apply:

- Use of automated techniques is encouraged.

- Use diagrams, tables, and other presentation styles when data can be made more readable by using them.

- Include a title page containing, as applicable, document number, volume number, version/revision indicator, security markings or other restrictions, date, document title, name, abbreviation, any other identifier for the item to which the document applies, contract number, CDRL item number, organization for which the document has been prepared, name and address of the preparing organization, and distribution statement. *For data in a database or other alternative form, include this information on external or internal labels or by equivalent identification methods.* [If a CLIN is used, provide the number of the contract line item in lieu of the CDRL item number.]

- Include a table of contents providing the number, title, and page number of each titled paragraph, figure, table, and appendix. *For data in a database or other alternative form, include this information in external or internal table of contents containing pointers to, or instructions for accessing, each paragraph, figure, table, and appendix or their equivalent.*

- Number and label each page with a unique page number and display the document number, including version, volume, and date, as applicable. *For data in a database or other alternative form, files, screens, or other entities, assign names or numbers in such a way that desired data can be indexed and accessed.*

- For paragraphs that are tailored out, include the paragraph number and paragraph title, but state following the title "This paragraph has been tailored out." *For data in a database or other alternative form, this representation should only be placed in the table of contents.*

- Use multiple paragraphing structure within any section, paragraph, or subparagraph to enhance readability.

- Reference, when practical, standard data element dictionaries for descriptions of data. MIL-STD-498 states that referencing is the preferred method.

- Substitute commercial or other existing documents for all or part of the document if they contain the required data.

5.21.2 Acquirer's responsibilities and considerations. Different types of software may require different decisions regarding delivery of documents. Some of the types of software often found on a project include: operational, non-developmental, operating systems, diagnostic, test software in the engineering and test environments, etc. All of the standard's twenty-two documents need not be delivered for each type of software. If the SOW does not provide tailoring of the standard for each type of deliverable software, it will be understood that the standard applies in its entirety (all activities will be performed and all applicable information

will be recorded [*somewhere*]). It does not necessarily follow that all recorded information needs to be delivered.

Although detailed tailoring of each DID can be specified in the CLIN and is provided for in Block 16 of the CDRL, this type of tailoring is rarely possible at the beginning of a project. It is more common for the acquirer to know that entire documents need not be delivered (e.g., there is no need for interface information to be bound in a separate document, or that specific user manuals or support manuals are known not to be needed) than to know whether a certain "piece" of information will need to be recorded. For this reason, MIL-STD-498 provides for much of the information called for in the DIDs to be recorded "as applicable" to the project. The standard recognizes that not all projects generate all the information called for, but that, at a minimum, the developer should have considered the possibility that the information does apply and record it if "applicable." This is the standard's concept of using the DIDs as a checklist. Note that if an activity is deleted (tailored out), the DID information called for by that activity will not be defined and recorded.

When making decisions on whether or not a document should be delivered, an acquirer should consider including other key players such as: the designated support agency, user, acquirer representatives (such as IV&V agents or others who will be representing the acquirer in reviewing the work products), and those who may have special interests, such as security experts, or others who have special knowledge of how the system is intended to be used. Figure 30 in this guidebook's topic, *Contract*, provides guidance on consideration of acquirer organization participation in document selection and tailoring. Acquirer considerations for documentation decisions may include:

- What information should be deliverable?
- Should deliverable information be in the form of paper versus an electronic document?
- If in electronic form, what, if any, format (ASCII, CALS, specific word processor, support software)?
- Can the information be in contractor format and structure rather than as specified in the DID?
- Can the information reside in a CASE or other automated tool rather than in the form of a traditional document?
- If traditional documents aren't delivered, what arrangements should be made for the acquirer, or the acquirer's authorized representative, to review the information? For example, can the acquirer have access via modem to the developer's development environment? What equipment and access to information does the acquirer need? What equipment does the acquirer have? Are special

arrangements necessary for the acquirer to access information at the development site?

- How many copies of the document are needed?
- When are the documents needed? Does the developer need feedback from the acquirer? Does the acquirer have milestone decision points that require information from the developer?
- Who should receive the documents? Why do they need the information?
- Should the developer embed user manuals as a "help" capability within the system?
- Should training materials be developed?

CASE or integrated CASE environments, that record information about the project in one place and share that information, may provide the acquirer the option of packaging the information in several ways depending upon needs. While "packaging" decisions are frequently made at the beginning of the project, MIL-STD-498's concept allows the acquirer to delay these packaging decisions by emphasizing that the development and recording of information is an intrinsic part of the software development process, to be performed regardless of whether a deliverable is required. If packaging decisions are deferred, the acquirer should recognize that additional costs will be incurred to format, assemble, mark, copy, and distribute the deliverable at that later time and plans should be made to provide for this packaging in a timely manner (i.e., before developer project personnel are reassigned, or in ample time for reviewers to receive and comment on the information).

If the acquirer or the software support organization has specific word processor or other formatting needs for delivering information, these should be specified on the CLIN or CDRL; otherwise, the developer is free to choose what makes the most sense for the project.

Other considerations in determining what documentation should be delivered can include those related to management needs. Some factors that may affect whether documents should be delivered include:

- Does the acquirer need to "see" project information to make project-related decisions (i.e., do progress payments depend on certain activities having been completed, do Defense Milestone Review Board decisions depend upon the acquirer's knowledge of project progress and risk identification)?

- Does the acquirer need to maintain a set of documents to make sure the developer doesn't change something without the acquirer's consent (for example, planning documents or specifications, if such documents are not part of the contract)?

- Does the acquirer need to "see" information to feel assured that the developer is performing the work?

5.21.3 Additional things to think about.

Does the developer have the necessary experience to support the specified format, such as CALS?
Who should be involved in reviewing and approving the documentation?
Who will use and retain the deliverable documents?
What format for the deliverable documentation will be most cost-effective for oversight of the developer's progress? What will be the most cost-effective for users? For the support agency?
Has a data call been performed to obtain input regarding documentation needs of users and support personnel?

5.21.4 Related guidebook topics.

CASE tools
Contract
Contract data requirements list
Documentation (Recording information)
Software development planning

5.22 Documentation (Recording information).

5.22.1 Requirements summary. MIL-STD-498 provides requirements for recording information applicable to the project. The standard makes a distinction between (1) preparing documents and (2) development and recording of planning and engineering information. MIL-STD-498 encourages the use of automated techniques and allows the substitution of commercial or other existing documents, including other project plans, for all or part of a document if they contain the required data.

A key concept in MIL-STD-498 is that development and recording of planning and engineering information (i.e., documentation) is an intrinsic part of the software development process, to be performed regardless of whether a deliverable document is required. Another key concept in MIL-STD-498 is that documentation need not be in the form of traditional documents. (See this guidebook's topic, *Documentation (Preparing documents)*, for more information.) References to ***Documentation (Recording information)*** in MIL-STD-498 are listed in Figure 50, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.19*, 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5, 5.3.2, 5.3.3, 5.4, 5.5, 5.6, 5.9.3, 5.9.7, 5.11.3, 5.11.7, 5.12, 5.13, *6.2, 6.4, 6.5*, D.4, *H.3*
All DIDs	*Block 7*, *10.1.a, 10.1.b, 10.1.g, 10.1.h, 10.1.i*
SDP	5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5, 5.3.2, 5.3.3, 5.4, 5.5, 5.6, 5.9.3, 5.9.7, 5.11.3, 5.11.7, 5.12, 5.13.1

FIGURE 50. MIL-STD-498's references to ***Documentation (Recording information)***.

To clarify that planning and engineering information is intrinsic to the development process, the standard allows use of the DIDs as checklists of items to be covered in the planning or engineering activity. When an activity is performed, it is understood that information, if applicable to the project, is recorded [*somewhere*] as described in the referenced software product descriptions for that activity. By tailoring out (deleting) an activity, the acquirer also deletes the preparation of information that would normally result from performing that activity. For example, if a specific contract has tasked a developer only to plan and perform qualification testing by tailoring out all other activities that produce engineering information, that developer would produce information called for by the Software Test Plan (STP), the Software Test Description (STD), and the Software Test Report (STR), but would not produce the information called for in the other software product descriptions. Figure 51 lists developer's activities related to ***Documentation (Recording information)***.

Developer's key activities related to ***Documentation (Recording information)***	References in MIL-STD-498
Describe the approach to be followed for documenting the software, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to documentation.	SDP 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5, 5.3.2, 5.3.3, 5.4, 5.5, 5.6, 5.9.3, 5.9.7, 5.11.3, 5.11.7, 5.12, 5.13
Develop and record planning information. Include all information in the SDP, STP, SIP, and STrP DIDs.	5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.1.5
Develop and record engineering information. Include all information in the COM, CPM, DBDD, FSM, IRS, IDD, OCD, SDD, SPS, SRS, SSDD, SSS, STD, STR, SUM, SIOM, SCOM, and SVD DIDs.	5.3.2, 5.3.3, 5.4, 5.5, 5.6, 5.9.3, 5.9.7, 5.11.3, 5.11.7, 5.12, 5.13
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 51. Developer's key activities related to ***Documentation (Recording information)***.

5.22.2 Acquirer's responsibilities and considerations. One of the principal tasks of the acquirer is to determine the scope of the work required for a project. Basic tailoring considerations apply when determining whether a specific activity should be performed as part of the overall work to be accomplished under the contract. (See the MIL-STD-498 Overview and Tailoring Guidebook for tailoring considerations.) Figure 30 of this guidebook's topic, *Contract*, provides guidance on other acquirer organizations' participation in software product selection and tailoring.

When using MIL-STD-498, the acquirer need not be concerned if all data required by each DID is necessary. Regardless of whether or not a deliverable document is required, if an activity is performed, the developer is required to define and record information called for by the DID called out by that activity, as applicable to the project.

MIL-STD-498 defines the data resulting from performing all of the planning and software engineering activities of the standard. The data is defined in the standard's twenty-two DIDs. This information (data) is the work product that results from a software development activity. The DIDs define what information is to be defined and recorded. The developer's methods/procedures/tools (approach) for performing the activities define how the information is to be generated. The DIDs use the phrase "as applicable" throughout to indicate that not all

information is "applicable" to all projects. The developer will determine whether the information called for is or is not applicable. This is not a license to avoid documentation. The developer should be able to justify all cases of "not applicable." When information is needed as a deliverable, the acquirer should put the applicable DIDs on a CLIN or CDRL in the contract. (See this guidebook's topics, *Contract data requirements list, Documentation (Preparing documents)*, and *Oversight*, for additional information.)

5.22.3 Additional things to think about.

Who will be involved in reviewing the documentation?
Has deliverable information been contracted for to support project oversight by the acquirer, or the acquirer's authorized representatives?
If CASE tools are to be used, who will select the tools and what selection criteria will be used?
Has deliverable information needed by the user been contracted for? That needed by the support agency?
How will defined and recorded information be reviewed by the acquirer or the acquirer's authorized representative(s) if the information has not been specified as a deliverable?

5.22.4 Related guidebook topics.

CASE tools
Contract data requirements list
Data accession list
Documentation (Preparing documents)
Oversight
Software development planning

5.23 Executable software.

5.23.1 Requirements summary. MIL-STD-498 provides requirements regarding executable software. The standard's executable software requirements cover the development, testing, packaging, and delivery of executable software and the installation and check out of the executable files at both the user and support site(s). This topic covers the requirements for packaging, delivery, installation, and check out of executable software. The requirements for development of source code and generation and testing of executable files are covered under this guidebook's topic, *Software implementation and unit testing*. References to ***Executable software*** in MIL-STD-498 are listed in Figure 52, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.32*, 4.2.3.1, 5.12.1, 5.12.2, 5.12.4, 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.7, 5.14, 5.15, Appendix B, Figure 6 (Item 18), *Figures 9-12*
SDP	4.2.3.1, 5.12.1, 5.12.2, 5.12.4, 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.7, 5.14, 5.15
SPS	3.1, 3.2, 4, 5.2, 5.3
SVD	3.2, 3.3, 3.6, 3.7

FIGURE 52. MIL-STD-498's references to ***Executable software***.

MIL-STD-498's requirements regarding preparing the executable software are intended to: (1) complete developer tasks regarding preparation of executable software for delivery, and (2) transition the executable software to the user and support site(s). When executable software is to be delivered to either the user or the support site, the developer is required to prepare all files needed to install and operate the software on the target computer(s), record the exact version for each site, deliver the packaged and marked software to the specified site(s), install and check out the software at the site(s), show that the copy installed is a valid copy, and identify and record the information called for by the Software Version Description (SVD) DID.

In addition, when software is to transition to a support site(s), the standard requires the developer to install the deliverable software in the designated support environment and demonstrate that the executable files can be regenerated from the source files. Information regarding modification procedures and compilation/build procedures for creating the executable product from the source files is described and recorded in accordance with the Software Product Specification (SPS) DID. The version of the software being released, along with an inventory of the software with installation instructions and a list of possible problems and known errors, is described in accordance with the SVD DID. Figure 53 lists developer's key activities related to ***Executable software***.

Developer's key activities related to ***Executable software***	References in MIL-STD-498
Describe the approach to be followed for executable software, identifying risks/uncertainties and plans for dealing with them. Include all applicable contractual clauses relating to executable software.	SDP 4.2.3.1, 5.12.1, 5.12.2, 5.12.4, 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.7, 5.14, 5.15
Evaluate reusable software for use in fulfilling contract requirements. Incorporate reusable executable software in accordance with applicable MIL-STD-498 requirements.	4.2.3.1
Prepare the executable software to be transitioned to the user sites. Identify and record the exact version of software prepared for each user site. Deliver the packaged and marked executable software for the user sites on electronic media. Install and check out the executable software at the user sites. State the method to be used to demonstrate that the software is a valid copy of the CSCI. Provide an inventory of the exact version of the software being released, installation instructions, and a list of possible problems and known errors with the software version.	5.12.1, 5.12.2, 5.12.4, 5.13.4 SPS 3.1, 4 SVD 3.2, 3.3, 3.6, 3.7
Prepare the executable software (and the source files needed to regenerate the executable software) to be transitioned to the support site. Deliver the packaged and marked executable software for the support site on electronic media. Install and check out the deliverable software in the designated support environment. State the method to be used to demonstrate that the software is a valid copy of the CSCI. Provide compilation/build and modification procedures for creating the executable software from the source files. Demonstrate that the deliverable software can be regenerated (compiled/linked/loaded into an executable product).	5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.7 SPS 3.1, 3.2, 5.2, 5.3
Provide an inventory of the exact version of the software being released, installation instructions, and a list of possible problems and known errors with the software version.	SVD 3.2, 3.3, 3.6, 3.7
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 53. Developer's key activities related to ***Executable software***.

Special provisions have been made to provide for delivery of executable software via the SPS. (See this guidebook's topic, *Contract data requirements list*, for more information and a sample DD Form 1423.) MIL-STD-498's SPS provides for delivery of both source and executable files (hard copy is not required unless specified). Although it is not mentioned in MIL-STD-498 as an alternative, the acquirer may order delivery of the executable files as a CLIN. These files (including any batch files, command files, data files, or other files needed to install and operate the software on its target computers) are to be provided on electronic media selected by the developer unless the acquirer includes a requirement in the contract specifying the deliverable media.

MIL-STD-498 not only provides requirements for the development of software products, such as executable software, but also provides requirements for the evaluation of those products. Criteria to be used in evaluation of executable files (and other software products) are found in Appendix D of the standard. Executable file evaluation criteria, in addition to the six generic criteria, include such items as whether the executable software has been formatted, marked, and packaged in accordance with contract/CLIN/CDRL requirements, whether all files needed for execution are present, and whether the version on the delivered media exactly matches the version that passed testing and is accurately labelled. (See this guidebook's topic, *Software product evaluation*, for more information.)

When reusable executable code is considered for incorporation, Appendix B of the standard provides mandatory and candidate criteria for evaluation of that software. Mandatory criteria are the software product's ability to meet specified requirements and its cost-effectiveness over the life of the system. Other candidate criteria included are: reliability/maturity of the reusable software, ability to provide required safety/security/privacy, testability, interoperability, licensing and data rights issues, maintainability, technical/cost/schedule risks, and the availability and quality of documentation and source files. In addition, Figure 3 in MIL-STD-498 provides self-tailoring of the standard for any reusable software products. The figure specifies how to interpret MIL-STD-498's activities based on performance record, place in the architecture, and whether it has been modified. For example, the table shows that the requirements for testing reusable executable files vary depending on whether there is a positive performance record for the software and whether the software is a unit or a CSCI.

5.23.2 Acquirer's responsibilities and considerations. MIL-STD-498's requirements for executable software are intended to ensure that the executable software, whether developed or reused, is prepared, controlled, evaluated, and installed following a well-defined process so that both user and support sites have the software products needed. When reusable software is to be used, the acquirer may want to verify that the developer's evaluation of executable files includes source code data rights considerations.

If there is the possibility that COTS software products or NDI source code will need to be modified at some future time in the system's life, the developer's plans should include obtaining the appropriate licenses and data rights for the software that provide information needed for modification. (See this guidebook's topics, *Data rights* and *Licenses (Software)*, for more information.)

When software is intended to transition to a support site after development and installation, personnel other than the developer will be responsible for modifying, enhancing, or otherwise changing the software. To ensure supportability, the developer is required to demonstrate that

the software can be regenerated (compiled/linked/loaded into an executable product). The demonstration can be used to validate the compilation/build procedures and software modification procedures described in the SPS as well as verify that the necessary hardware and software elements are available in the support environment. Although the standard does not require the developer to perform a virus scan on deliverable software, virus protection requirements are often worthwhile when COTS and other deliverable software is being introduced into an existing operational environment.

A key consideration and responsibility of the acquirer is to determine whether to allow delivery of software with known errors. Depending on the software support concept, scheduling issues, and other contractual or fielding issues, it may be cost-effective or necessary for the acquirer to field such a system. MIL-STD-498 allows delivery of software with known errors and requires those errors to be documented in the SVD along with other possible problems with the software version being delivered.

Some DoD services or government agencies have established thresholds for accepting software with errors, such as a specific total number of errors or limitations based on the category and priority classifications in MIL-STD-498's Figures 4 and 5. (See this guidebook's topics, *Acceptance by the acquirer* and *Contract*, for more information.) The acquirer should check whether any such thresholds, DoD service or government agency policies, or other contractual requirements regarding errors and problems (such as warranty provisions) apply. The acquirer may need to consider different types of software separately, such as COTS/NDI software versus software developed for the project. If there are no contract requirements concerning the acceptability of deliverable software containing errors, and if the developer's description of the planned approach for delivery of software to user sites and the support site does not address the subject, it will be in the best interests of both parties to establish such thresholds. Joint reviews provide a forum for discussion so agreement between the acquirer and developer can be reached early in the project.

5.23.3 Additional things to think about.

Are data rights and licensing issues resolved for COTS and/or NDI being used in the system?
Are COTS/NDI supportability provisions agreed to by all interested parties (user, acquirer, support agency, developer, vendor)?
Will the executable software be delivered on media that the acquirer can use? Media that the support agency can use in the support environment?
Is source code available for COTS/NDI that has been or might be modified?
Are accurate compilation/build procedures and software modification procedures for the support agency described in the SPS?
Is the developer's readiness to demonstrate the ability to regenerate executable software being addressed? Are the resources available in the support environment to perform the regeneration procedures?
Has agreement been reached on the issue of accepting software with known errors?
Are adequate configuration management controls in place to ensure the integrity between source files and related executables?

5.23.4 Related guidebook topics.

Acceptance by the acquirer
Contract
Contract data requirements list
Data rights
Licenses (Software)
Software implementation and unit testing
Software product evaluation
Source files

5.24 Independent verification and validation.

5.24.1 Requirements summary. MIL-STD-498 provides requirements for the developer to interface with the software Independent Verification and Validation (IV&V) agent(s), if any, as specified in the contract. This is one of the standard's "shell" requirements that impose no tasking on the developer unless "fleshed out" in the contract. The standard defines independent verification and validation as:

> "*Systematic evaluation of software products and activities by an agency that is not responsible for developing the product or performing the activity being evaluated*."

Requirements for performing IV&V are not within the scope of MIL-STD-498. If an IV&V agent is specified in the contract and requirements are provided for the developer to work with that agent, the developer is required to describe the approach to be followed for interfacing with the software IV&V agent(s) in the SDP for the project and to cover all contractual clauses regarding that interface. References to ***Independent verification and validation*** in MIL-STD-498 are listed in Figure 54, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.23*, 5.19.5, *Figures 9-12*
SDP	5.19.5

FIGURE 54. MIL-STD-498's references to ***Independent verification and validation***.

5.24.2 Acquirer's responsibilities and considerations. Hiring an IV&V agent or performing IV&V through program or support agency personnel is one option available to the acquirer for gaining assurance that a system is being developed that will meet user needs. IV&V can be performed on any project for which the acquirer needs extra assurance. Software whose failure to perform as required could jeopardize safety, security, or other critical requirements, and result in unacceptable circumstances, such as unintended nuclear detonation, or other hazardous conditions detrimental to humans and the environment, is a prime candidate for IV&V. Other factors can also affect the acquirer's need for additional assurance, such as the developer's performance record, risk of failure to the project or enterprise, and cost.

IV&V is intended to provide another set of "expert eyes" who can evaluate the development of the software. IV&V agents are concerned first and foremost with determining whether the requirements of the system/subsystem or CSCI have been implemented, and, if implemented, that they have been implemented correctly. IV&V agents may also trace requirements through

design to code, independently derive algorithms and design to ensure that alternatives have been considered, and independently perform tests on products. IV&V is not intended as a "document review process" focused on the quality of the information recorded, although that is sometimes one by-product of the activity.

5.24.3 Additional things to think about.

Is an IV&V agent warranted on the project?
Would funds allocated to the IV&V agent be better spent in providing the developer with additional funds?
If in-house or support agency personnel are used to perform IV&V, do they have the required subject matter expertise? Are personnel available? Can they be dedicated to the project?
Has the IV&V agent a proven record for this type of project?
Should safety, security, and other critical requirements undergo IV&V, even if the whole system will not undergo IV&V?
Should the developer and IV&V agent(s) interface directly or through the acquirer?

5.24.4 Related guidebook topics.

Process improvement
Qualification testing
Requirements
Software product evaluation
Software quality assurance
Testing (Developer-internal)
Traceability

5.25 In-house development.

5.25.1 Requirements summary. MIL-STD-498 provides no requirements differentiating between in-house development and contractual development. MIL-STD-498 can be applied to government in-house agencies performing software development as well as contractors or subcontractors. When applied to in-house projects, the agency or organization that procures software products for itself or another organization is the acquirer. The organization that develops software products is the developer ("develops" may include new development, modification, reuse, reengineering, or any other activity that results in software products).

5.25.2 Acquirer's responsibilities and considerations. MIL-STD-498's general requirements state that the developer is to establish a software development process consistent with contract requirements. When in-house development takes place, the term "contract" in MIL-STD-498 may be interpreted as Memorandum of Agreement, Memorandum of Understanding, list of tasks to be performed, or some other formal or even informal understanding of the work to be performed by one group for another. For in-house development, as for contracted, the acquirer should evaluate the in-house developer's SDP to determine the in-house developer's capability to perform the tasks and meet the schedules.

When MIL-STD-498 is applied to an in-house project, the different roles played need to be carefully defined to avoid confusion. For example, an "acquirer" might also be a "user;" the "developer" might also be a "user;" the "support agency" might be the "acquirer" for this project, but might be the "developer" on another. One organization may perform several roles on a single project or perform differing roles on different projects. Personnel involved as part of the acquisition team may be transferred to the development team; organizations may be disbanded, transferred, or reorganized so that the "acquirer" and "developer," once separated by organizational boundaries now reside together. Documenting the agencies/organizations involved and their respective roles and responsibilities can help ensure that changes that occur over time do not needlessly disrupt the development or raise project risks.

5.25.3 Additional things to think about.

When development is in-house, what is the acquirer's leverage to avoid cost and schedule overruns?
Are risk evaluations available for in-house development?
Are in-house methods/procedures/tools mature? Have they been used successfully for development of similar software?
Are acquirer, developer, user, and support roles clearly established?
Are there provisions supporting appropriate player roles irrespective of rank or privilege considerations?

5.25.4 Related guidebook topics.

Contract
Risk management
Software development process
Statement of work

5.26 Installation (Support environment).

5.26.1 Requirements summary. MIL-STD-498 provides requirements for the developer to plan and perform software installation in the support environment specified in the contract. The software to be installed might be newly developed, modified, reusable, reengineered, or any other type of software called for under the contract. The software to be installed may also include support software and other software needed to use, regenerate, or modify that software. References to ***Installation (Support environment)*** in MIL-STD-498 are listed in Figure 55, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.1.5, 5.13, 5.14.5, *Figure 3, Figure 6, E.4.1, E.4.10*
FSM	3.x.6
SDP	5.1.5, 5.13, 5.14.5
SPS	3, 5
STrP	All
SVD	3

FIGURE 55. MIL-STD-498's references to ***Installation (Support environment)***.

MIL-STD-498 provides requirements for the developer to prepare the executable and source files for installation, describe installation procedures in version descriptions, develop and record information specified in the Software Product Specification (SPS) DID, and, if applicable, the Firmware Support Manual (FSM) DID, and record key decisions in documents that will transition to the support environment. As part of transitioning the software to the support environment, the developer is tasked to install the deliverable software in the support environment designated in the contract and to demonstrate to the acquirer that the deliverable software can be regenerated and maintained using available, acquirer-owned, or contractually deliverable software and hardware designated in the contract or approved by the acquirer.

If the support site is also a user site the requirements associated with preparing for software use and installation at user site(s) may need to be performed for the support site(s) as well. (See this guidebook's topic, *Installation (User site(s))*, for more information.) Figure 56 lists developer's key activities related to ***Installation (Support environment)***.

Developer's key activities related to ***Installation (Support environment)***	References in MIL-STD-498
Describe the approach to be followed for installation in the support environment identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to installation in the support environment.	SDP 5.1.5, 5.13, 5.14.5
Develop and record plans for performing software installation at the support site(s) specified in the contact. Verify that the plan is consistent with other project plans and presents a sound approach to the installation.	5.1.5, Figure 6 STrP
Install and check out the deliverable software on its target computer(s) at the support site(s) designated in the contract.	5.12.4
Prepare the executable software for each support site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s).	5.13.1 SPS 3.1
Prepare the source files to be transitioned to the support site, including any batch files, command files, data files, or other files needed to regenerate the executable software.	5.13.2 SPS 3.2
Identify and record the exact version of software prepared for the support site.	5.13.3 SVD 3
Define and record the methods to be used to verify copies of the software.	5.13.4 SPS 3.1, 3.2
Identify and record information needed to program and reprogram any firmware devices in which the software will be installed.	5.13.6 FSM 3.x.6
Install and check out the deliverable software in the support environment designated in the contract. Demonstrate to the acquirer that the deliverable software can be regenerated and maintained using commercially available, acquirer-owned, or contractually deliverable software and hardware.	5.13.7 SPS 5.1, 5.2, 5.3, 5.4, 5.5
Establish and implement procedures for the packaging, storage, handling, and delivery of deliverable products.	5.14.5 SPS 3.3
Conduct joint software supportability technical and management reviews, if selected as reviews.	E.4.10
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 56. Developer's key activities related to ***Installation (Support environment)***.

5.26.2 Acquirer's responsibilities and considerations. A key decision that the acquirer should make early is the support strategy for the software. This decision involves determining who should be responsible for modifying, enhancing, correcting and otherwise supporting the software. (See this guidebook's topics, *Software support* and *Software transition*, for more information.) Options include support by the developer, no support, and transitioning to a government support agency or another contractor.

When the decision is to transfer support to another, the Software Transition Plan (STrP) DID requires the developer to describe the resources, procedures, and training needed for support; anticipated areas of change; and plans for transition. The description of resources is to include identification of those responsible for providing each hardware and software item. For a variety of reasons, there may be a mixture of responsibility. The acquirer and/or the support agency may have provided items to the developer; the support agency may already have items; the developer may be using company-owned items not intended for delivery, etc. Providing all necessary resources for support in a timely manner needs careful attention, coordination, and planning by the acquirer, developer, and designated support agency.

The acquirer will also need to determine how the designated support environment personnel and developer personnel will interface during installation. Acquirer planning might include such items as:

- Who from the support environment will be responsible for coordinating with the developer on each installation activity?
- After installation, who (developer or support agency) will track and manage any pre-existing problems or bugs in the software?
- Who will be responsible for resolving the problems and/or bugs after installation?
- Is there a specific date by which all responsibilities must be transitioned? Has adequate time been allocated for transition planning and coordination?
- Is training required in conjunction with installation? If so, who should perform that training?
- What happens if the installation is unsuccessful? Will unsuccessful installation affect users at user site(s)? In the field? Will unsuccessful installation affect planned training?

Throughout the planning, development, installation, and transition activities, the developer will be responsible for communicating, reviewing, and resolving open issues regarding installation in the support environment with the acquirer. The forum for this discussion is a joint technical or management review. Appendix E of the standard identifies software supportability reviews as one of the candidate reviews that might be held on a project.

5.26.3 Additional things to think about.

Who is responsible for verifying that the software and hardware used for acceptance testing are identical to that delivered to support site(s)?
Who is responsible for verifying that the executable software can be regenerated in the support environment?
Is the criteria for successful installation in the support environment clearly defined?
Does the software transition plan identify risks (including the risk that facilities, hardware, and software to support installation may not be available at installation time) and does it identify activity dependencies?
Have there been undocumented changes to the hardware or software that would affect the planned support environment?
Who will be responsible for approving changes to the installation plan?
Is new technology planned for the support environment? Has adequate consideration been given to the learning curve necessary for that technology? What if the technology cannot be implemented in the support environment?
Have software license issues been resolved for COTS software to be transitioned to the support environment?
Has the expected system performance been benchmarked?
What are the acceptance criteria for electronic documentation? How will its installation be performed?
Are there backup plans for installation failure or delay? How will recovery occur if the installation is postponed?
Does the developer have the needed transition assumptions and data to complete planning? Is transfer of configuration management responsibilities explicitly addressed in planning?

5.26.4 Related guidebook topics.

Acceptance by the acquirer
Documentation (Preparing documents)
Documentation (Recording information)
Installation (User site(s))
Schedules
Software support
Software transition
Version/revision/release

5.27 Installation (User site(s)).

5.27.1 Requirements summary. MIL-STD-498 provides requirements for the developer to plan and perform software installation at the user site(s) specified in the contract. The software to be installed might be newly developed, modified, reusable, reengineered, or other software called for under the contract, such as software needed to monitor, reinstall, or run that software in the user environment. References to ***Installation (User site(s))*** in MIL-STD-498 are listed in Figure 57, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.1.4, 5.12, Figure 3, Figure 6, *E.4.1, E.4.9, Figure 14*
SCOM	4
SDP	5.1.4, 5.12, 5.14.5
SIP	All
SPS	3.1, 3.3
SSS	3.16
SUM	4.1.3
SVD	3

FIGURE 57. MIL-STD-498's references to ***Installation (User site(s))***.

MIL-STD-498 also provides requirements for the developer to prepare the executable and source files for installation and describe installation procedures in accordance with the Software Version Description (SVD) DID, the Software Product Specification (SPS) DID, and applicable user manuals. Note that a user site may also be a support site and that the tasks of transitioning software to the support environment may need to be coordinated with the task of installing the software for use. Figure 58 lists developer's key activities related to ***Installation (User site(s))***.

Developer's key activities related to ***Installation (User site(s))***	References in MIL-STD-498
Describe the approach to be followed for installation at the user site(s) identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to installation at the user site(s).	SDP 5.1.4, 5.12, 5.14.5
Develop and record plans for performing software installation at the user site(s) specified in the contact. Verify that the plan is consistent with other project plans and presents a sound approach to the installation.	5.1.4, Figure 6 SIP
Establish and implement procedures for the packaging, storage, handling, and delivery of deliverable products.	5.14.5 SPS 3.3
Prepare the executable software for each user site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s).	5.12.1 SPS 3.1
Identify and record the exact version of software prepared for each user site.	5.12.2 SVD 3
Prepare installation procedures for the user site(s) and record in applicable user and operator manuals.	5.12.3.1, 5.12.3.3 SCOM 4 SSS 3.16 SUM 4.1.3
Install and check out the deliverable software on its target computer(s) at the user site(s) designated in the contract.	5.12.4
Conduct joint software usability technical and management reviews, if selected as reviews.	E.4.9
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 58. Developer's key activities related to ***Installation (User site(s))***.

5.27.2 Acquirer's responsibilities and considerations. A key activity that the acquirer will need to perform is identify the user site(s) at which the software will be installed, identify users with whom developer personnel will interface during installation, and ensure that installation risks have been identified and plans have been made for dealing with those risks. The Software Installation Plan (SIP) DID provides details of the information to be defined and recorded regarding installation.

To complete installation plans, the acquirer or user will need to provide the developer with information such as:

- Who from the user site will be responsible for coordinating with the developer on each installation activity?

- After installation, who (developer, user, acquirer, support agency) will track and manage any pre-existing problems or bugs in the software?
- Who will be responsible for resolving the problems and/or bugs after installation?
- Is training required in conjunction with installation? If so, who will perform that training?
- Will the old system and the new one need to exist in parallel to ensure that the necessary capability can be provided? If so, what are the criteria for deciding to terminate the old system?

Throughout planning and development, the developer will be responsible for communicating, reviewing, and resolving open issues regarding installation at user sites with the acquirer. The forum for this discussion is the standard's joint technical and management reviews. Appendix E of MIL-STD-498 identifies software plan and usability reviews as candidate reviews that might be held on a project to cover these topics.

5.27.3 Additional things to think about.

Who is responsible for verifying that the software and hardware used for acceptance testing are identical to that delivered to the user site(s)?
Is the criteria for successful installation at the user site(s) clearly defined?
Will users need a help line? If so, who will be responsible for it?
Does the software installation plan identify risks (including the risk that the user may not have facilities, hardware, and software to support installation at the user site)?
If qualification testing was not performed on the target computer, who is responsible for ensuring that the software will "work" on the target computer?
Should the software be accepted after or before installation at the user site(s)? If accepted before installation, are there any warranties?
Once installation begins, how can loss of continuity of operation be avoided?
Are there activity dependencies? Does the schedule rely too heavily on any one individual?
Are there any conditions that would cause the user to stop the installation at a user site? Are there contingency plans for installations which are stopped? Who makes the decision to stop installation?
Are there backup plans for installation failure or delay? How will recovery occur if the installation is postponed?
Who will be affected by unsuccessful installation? Will unsuccessful installation affect planned training?
Who will be responsible for approving changes to the installation plan?
Has the expected system performance been benchmarked on the target computer(s)? In the target system?

5.27.4 Related guidebook topics.

Acceptance by the acquirer
Documentation (Recording information)
Installation (Support environment)
Software support
Source files
Version/revision/release

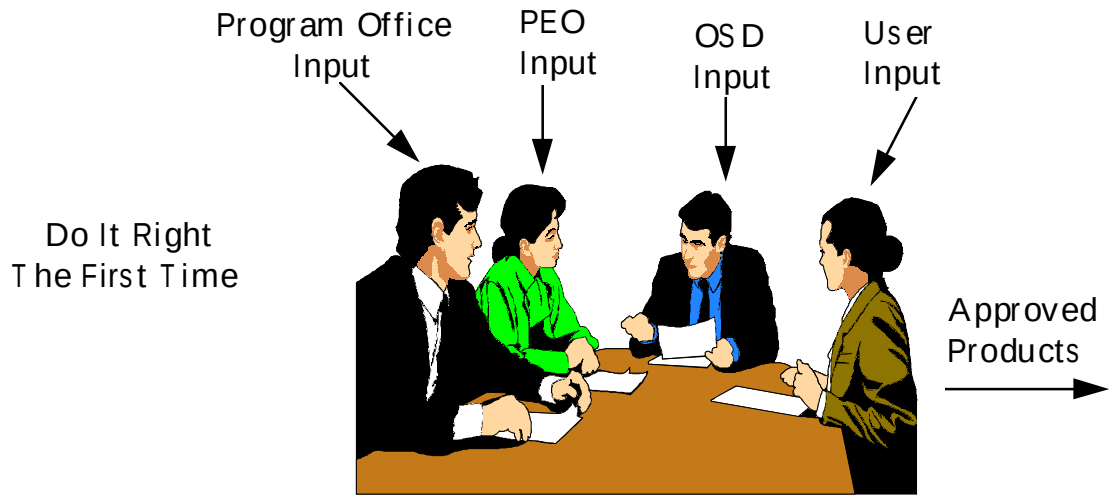
5.28 Integrated product team.

5.28.1 Requirements summary. MIL-STD-498 provides no requirements relating to Integrated Product Teams (IPTs).

5.28.2 Acquirer's responsibilities and considerations. DoD's acquisition policy identifies two types of DoD IPTs: (1) the program IPT composed of interdisciplinary, cross functional members (e.g., program management, engineering, manufacturing, test, logistics, financial management, procurement, and contract administration), including customers (users) and developers; and (2) the acquisition management IPT that includes members from differing levels of DoD's acquisition management (e.g., program office, responsible DoD organization such as Service testing agency or logistics agency, Component Acquisition Executive (CAE) or Program Executive Officer (PEO) staff, and Office of the Secretary of Defense (OSD) operational and developmental test offices).

DoD's concept of IPTs is intended to avoid a sequential development and approval process thereby streamlining acquisition and preventing delays and rework. A non-sequential process has the potential to decrease the time to develop acquisition-related products. Figure 59 exemplifies the difference between IPT and non-IPT product development.

Program IPTs are structured to ensure that adequate cross functional resources are available to perform tasks, such as, define requirements, perform risk management, prepare life cycle plans, and review and evaluate products. Acquisition management IPTs are structured to allow the development of acceptable acquisition-related products the first time through rather than causing costly rework. For example, a test strategy acquisition management IPT could be formed to develop a Test and Evaluation Master Plan (TEMP) for government testing of a system. The test strategy IPT could include test representatives from the program office, DoD Service testing agency, CAE or PEO staff, and OSD operational and developmental test offices. The varying levels of management can contribute information about criteria for an acceptable TEMP from the view of key players at each level. The goal would be to develop a TEMP that is acceptable to all "the first time." Rework can be avoided by including members from various staff and line levels. Rather than using a sequential process wherein a TEMP developed at a lower-level is forwarded for review at higher-levels, then modified substantially or even rejected at those levels and sent back down, program office staff and higher-level managers will work together.



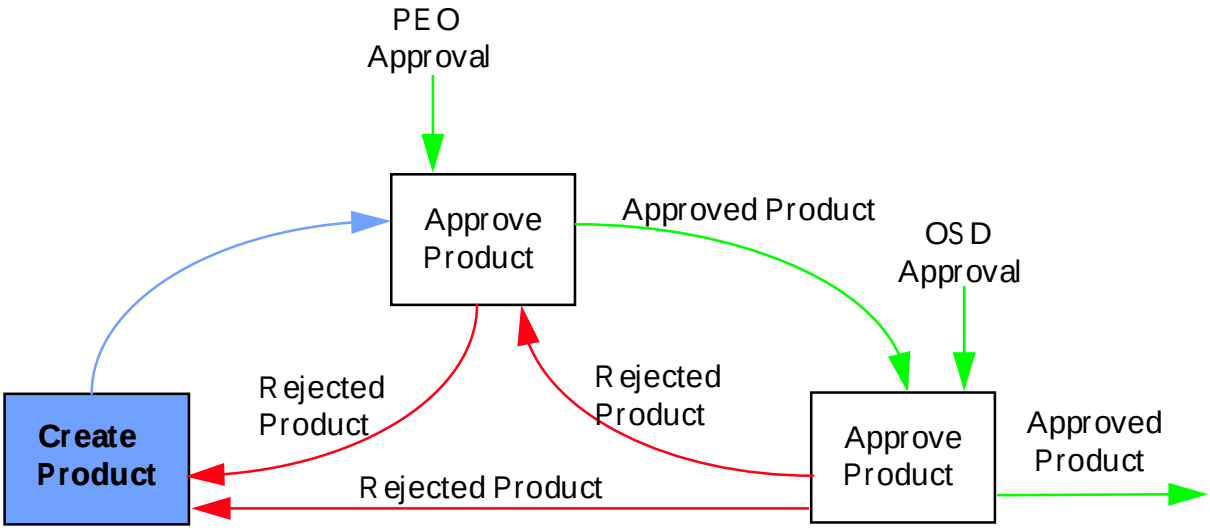


FIGURE 59. IPT versus non-IPT product development.

The concept of IPTs for program and acquisition management is most applicable to acquisition category (ACAT) ID programs. However, the use of the concept for programs in all acquisition categories is encouraged by DoD acquisition policy. The DoD's goal is to move away from a pattern of strictly hierarchical decision-making in the acquisition of systems to a process where decisions are facilitated by integrated product teams to reduce inefficient rework.

5.28.3 Additional things to think about.

Are IPTs aware of the software program strategy for the project (i.e., Grand Design, Incremental, Evolutionary)?
Do the IPTs cover all needed functional areas, including software?
Are joint technical and management reviews coordinated with IPTs so that information and decision making is effective and efficient?
Are responsibilities and roles clear regarding IPTs?

5.28.4 Related guidebook topics.

Contract

Joint technical and management reviews

Oversight

5.29 Interfaces.

5.29.1 Requirements summary. MIL-STD-498 defines interface as "*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. References to ***Interfaces*** in MIL-STD-498 are listed in Figure 60, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.24*, 4.2.2, 4.2.3.1, 5.3.3, 5.4 - 5.11, 5.13.4, 5.13.5, *6.2*, Figure 6 (Items 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 25, 26), *Figures 9-12, Figures 15-17*
DBDD	3, 5.x
IDD	All
IRS	All
SDD	3, 4.3, 5, 6
SDP	4.2.2, 4.2.3.1, 5.3.3, 5.4 - 5.11, 5.13.4, 5.13.5
SPS	5.1
SRS	3.3, 3.4, 5
SSDD	3, 4.3
SSS	3.3, 3.4, 5
STD	5
STP	6

FIGURE 60. MIL-STD-498's references to ***Interfaces***.

The standard requires the following information, as applicable, about interfaces:

- Interface identification and diagrams
- Priority assigned to the interface by interfacing entities
- Type of interface (such as real-time data transfer or storage and retrieval of data)
- Characteristics of individual data elements that interfacing entities send, receive, store, access, etc.
- Characteristics of data element assemblies (such as records, messages, files, displays, and reports) that interfacing entities send, receive, store, access, etc.
  - Characteristics of individual data elements and their assemblies can include: data type, format, size, units of measure, range, accuracy, frequency of transmission, and security constraints. Some or all of these characteristics may be provided separately as a data dictionary.

- Communication methods
- Protocol characteristics
- Physical characteristics

Figure 61 lists developer's key activities related to ***Interfaces***.

Developer's key activities related to ***Interfaces***	References in MIL-STD-498
Describe the approach to be followed for specifying interface requirements and for designing, implementing, and testing interfaces, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to interfaces.	SDP 5.3.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.13.4, 5.13.5
Participate in defining and recording system/subsystem interface requirements. Define and record traceability between system-level and subsystem-level requirements.	5.3.3 IRS SSS 3.3, 3.4, 5
Participate in defining and recording system interface design. Define and record traceability between system design and system requirements.	5.4 DBDD 3, 5.x IDD SSDD 3, 4.3, 5
Define and record CSCI interface requirements. Define and record traceability between CSCI requirements and system requirements.	5.5 IRS SRS 3.3, 3.4, 5
Define and record CSCI interface design. Define and record traceability between CSCI design and CSCI requirements.	5.6 DBDD 3, 5.x IDD SDD 3, 4.3, 5, 6
Implement the interface design. Perform applicable developer internal testing.	5.7, 5.8, 5.10
Perform CSCI qualification testing and participate in system qualification testing. Define and record traceability between interface requirements and tests/test cases.	5.9, 5.11 STD 5 STP 6
Update interface design descriptions to match the "as built" software.	5.13.4, 5.13.5 SPS 5.1
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 61. Developer's key activities related to ***Interfaces***.

Descriptions of an interface(s) may be developed during system requirements analysis, system design, software requirements analysis, software design, and software implementation. When a new system, subsystem, or CSCI is to be developed that must interface with the existing systems, subsystems, or CSCIs, interface descriptions may already exist.

Interface requirements are described in one or more System/Subsystem Specifications (SSSs), Interface Requirements Specifications (IRSs), and/or Software Requirements Specifications (SRSs). Interface design information is described in one or more System/Subsystem Design Descriptions (SSDDs), Interface Design Descriptions (IDDs), Software Design Descriptions (SDDs), or Database Design Descriptions (DBDDs). (Note that the IRS and IDD DIDs may be reused at any level, system/subsystem and CSCI, to specify and describe interfaces.) Interface design descriptions will also include or reference any conventions or notations needed to understand the design. (See this guidebook's topics, *Contract data requirements list, Documentation (Preparing documents),* and *Documentation (Recording information)*, for more details on defining and recording information.)

5.29.2 Acquirer's responsibilities and considerations. There are many types of interfaces within a system. Examples of hardware interfaces are connections between mechanical, electrical, hydraulic, and magnetic components. In software, information and control data (including messages, signals, interrupts) provide connectivity between components. Rules regarding how connection is established, what services are supplied, data characteristics, and other timing and sizing considerations, constitute a software interface description. Other common categories of interfaces combine hardware, software, and people; these types of interfaces are known as human-machine interfaces (HMIs), human-computer interfaces (HCIs), and graphical user interfaces (GUIs). (See this guidebook's topic, *Behavioral design*, for more information regarding interfaces.)

Other factors enter into determining how interfaces are described and documented. These factors include:

- Whether there is an existing or planned system, subsystem, or CSCI that the system must interface with that is not under the developer's control
- Whether the acquirer needs to specify data characteristics, such as characteristics needed to support data standardization within a specific domain
- Whether a single developer is responsible for producing the system/subsystem/CSCI or the development effort is shared between a contractor team, subcontractors, and/or associate contractors

To provide a consistent set of characteristics for describing data elements, each of MIL-STD-498's requirements and design DIDs provide a common list of required information. This does not mean that descriptions at each level for every interface need to include all the information identified in each DID. Higher-level descriptions (such as those in the SSSs and SSDDs) may often contain incomplete or sketchy descriptions and the detail provided may vary between levels and between differing interfaces. Alternatively, some system/subsystem interfaces may need to be described in great detail early in the project (for example, to provide a description of an existing system with which the system must interface), while development of yet other interface descriptions may need to wait until hardware has been developed.

When requirements have been specified in detail (for example, when interfacing with an existing system/subsystem/CSCI), a separate interface design description might not be needed if the design information simply duplicates those requirements. MIL-STD-498 also allows one document to reference another in satisfying information requirements. For example, interface information found in user manuals, standard data element dictionaries, or established standards (such as the common serial interface standard RS-232) may be referenced to provide needed information. Caution should be used to ensure that whoever needs the information has or can readily obtain the necessary information.

When multiple developers are involved on a project, a potential problem area exists if different developers are responsible for development on opposite sides of an interface. Without proper coordination and communication, changes made on one side of the interface can easily cause problems on the other side. One way to avoid this type of problem is through the use of special groups of developer personnel who represent both sides of an interface so that requirements and design can be agreed upon. These groups are sometimes referred to as Interface Control Working Groups (ICWGs). The progress and results of ICWG efforts can be included as a part of joint technical review meetings.

5.29.3 Additional things to think about.

If two or more developers have responsibilities for a particular interface, are adequate measures in place to promote their coordination?
Is any one group responsible for control of interfaces and defining their documentation or specifications? Who "owns" the interface?
Are interfaces a topic in appropriate joint technical reviews?
Does the SDP make it clear how interfaces are described? In which software products?
At which level(s) are interfaces tested?
Is all data defined in the interfaces including messages, interrupts, signals? Have all control aspects been defined including communication and protocol characteristics?
Is there adequate software participation in system interface activities?
Do demonstrations and prototypes cover interface capabilities?

5.29.4 Related guidebook topics.

Architectural design
Behavioral design
Contract data requirements list
Database design
Detailed design
Documentation (Preparing documents)
Documentation (Recording information)
Requirements
Requirements of the standard
System/subsystem-wide and CSCI-wide design

5.30 Joint technical and management reviews.

5.30.1 Requirements summary. Planning and taking part in joint technical and management reviews with the acquirer is one of MIL-STD-498's software development activities. It is considered one of the integral processes that cut across the other activities in the standard. Joint technical and management reviews include informal discussions of status, ideas, approaches, risks, etc. Objectives of these reviews include providing the acquirer visibility into the project status, directions being taken, technical issues to be resolved, overall status of evolving software products, and obtaining commitments and acquirer approvals needed for timely accomplishment of the project. These reviews provide the acquirer and the developer insight into the needs and concerns each has regarding the project. References to ***Joint technical and management reviews*** in MIL-STD-498 are listed in Figure 62, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.25*, 4.1, *Figure 1*, 5.18, *Figure 2*, Figure 3, *Figures 9-12, G.6.3, H.3*
SDP	5.18

FIGURE 62. MIL-STD-498's references to ***Joint technical and management reviews***.

The intent of these reviews is to provide ongoing communication between the acquirer and developer with minimum wasted time and effort for both. The standard assumes that:

- Joint technical reviews will be attended by developer and acquirer personnel technically qualified to present and discuss the technical aspects of the project. For example, participants in a technical review of the software architecture of an object-oriented project should assume that the review will be attended by developer personnel capable of clearly presenting the issues that need to be resolved and by acquirer personnel who have an understanding of that methodology.

- Prior to conducting any joint management review, the acquirer has reviewed the products in advance and determined that one or more technical reviews have been held to resolve issues. This leaves the joint management review as a forum to resolve open issues and reach agreement as to whether the project is progressing as planned.

For both joint technical and management reviews, the standard requires the developer to plan and get acquirer approval for the time and place, and plan the technical matter to be discussed. For joint management reviews, it is intended that both acquirer and developer have in attendance personnel with the authority to make technical, cost, and schedule decisions. Figure 63 shows how the review activities interrelate.

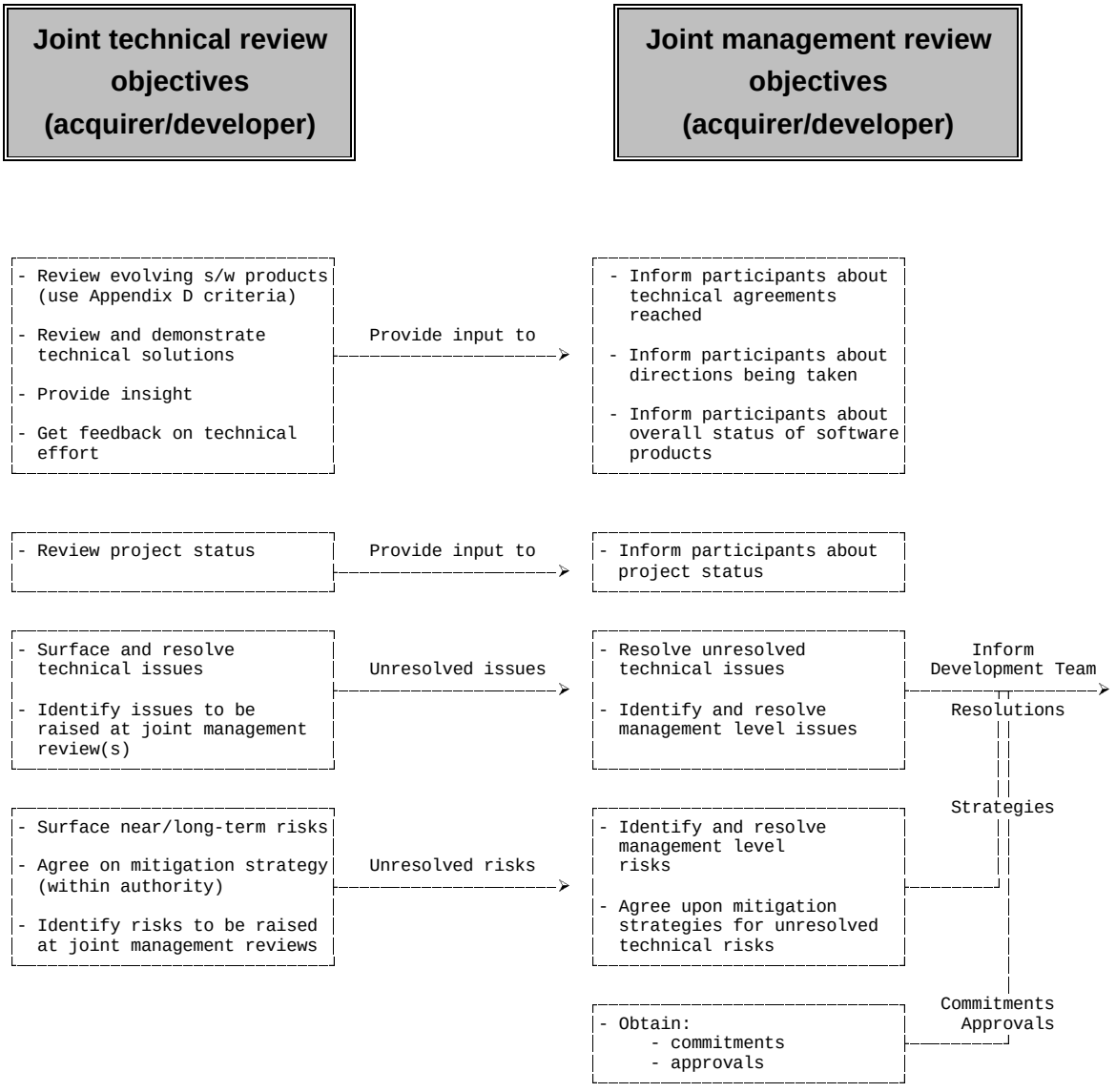


FIGURE 63. Joint review objectives.

5.30.2 Acquirer's responsibilities and considerations. MIL-STD-498's joint technical and management reviews are designed to provide all of the coordination needed between the developer and the acquirer. Joint reviews may include associate developers, subcontractors, and acquirer representatives/agents. To differentiate technical from management meetings, technical reviews are sometimes called by other names, such as, Technical Interface Working Group, Technical Interchange Meetings, or other name that clarifies the intent of the working group. Reviews might be scheduled periodically throughout the development, with elapsed time the criteria for the visit, or they might be based upon particular objectives, such as refining requirements and qualification test procedures. Reviews may focus on a development aspect of a single CSCI or multiple CSCIs, or may focus on development aspects that cut across the system (e.g., supportability).

Instead of formal reviews, MIL-STD-498 calls for more frequent, low-key joint (acquirer/developer) technical and management reviews, focusing on natural work products (such as IDEF models or prototypes) rather than presentation materials (such as view-graphs, transparencies) or other specially prepared materials. View-graphs and other specially prepared materials are not, however, prohibited.

Regardless of the scheduling of joint reviews or their proposed agenda, the technical reviews are intended to precede the corresponding management sessions. Management reviews are intended to be concerned with summary information of progress, schedules, and costs, to deal with resolutions of unresolved technical issues, and to reach decisions on whether milestones are sufficiently complete to warrant proceeding.

If a system or CSCI is developed in multiple builds, the types of joint reviews held and the criteria applied will depend on the objectives of each build. For example, in an evolutionary development it may be necessary to discuss and agree upon additional requirements for a subsystem at the same time that the behavioral design of a CSCI for another subsystem is discussed. When reviewing software products, it is important to remember that software products that meet the objectives of a particular build can be considered satisfactory even though they are missing information designated for development in later builds.

Appendix E of MIL-STD-498 lists eleven candidate joint management reviews. The management reviews listed are candidates intended to be used by the developer to structure an approach to joint technical and management reviews for a project. Figure 64 lists the objectives of the eleven candidate management reviews.

Candidate Review	Objective:
Software plan reviews	Identify and resolve open issues regarding one or more of the following: - software development plan - software test plan - software installation plan - software transition plan
Operational concept reviews	Identify and resolve open issues regarding the operational concept for a software system
System/subsystem requirements reviews	Identify and resolve open issues regarding the specified software requirements for a system or subsystem
System/subsystem design reviews	Identify and resolve open issues regarding one or more of the following: - system-wide or subsystem-wide design decisions - architectural design of a software system or subsystem
Software requirements reviews	Identify and resolve open issues regarding the specified requirements for a CSCI
Software design reviews	Identify and resolve open issues regarding one or more of the following: - CSCI-wide design decisions - architectural design of a CSCI - detailed design of a CSCI or portion of a CSCI (such as a database)
Test readiness reviews	Identify and resolve open issues regarding one or more of the following: - status of software test environment - test cases and test procedures to be used for CSCI or system qualification testing - status of the software to be tested
Test results reviews	Identify and resolve open issues regarding the results of CSCI or system qualification testing
Software usability reviews	Identify and resolve open issues regarding one or more of the following: - readiness of the software for installation at user sites - user and operator manuals - software version descriptions - status of installation preparations and activities
Software supportability reviews	Identify and resolve open issues regarding one or more of the following: - readiness of the software for transition to the support agency - software product specifications - software support manuals - software version descriptions - status of transition preparations and activities, including transition of the software development environment
Critical requirement reviews	Identify and resolve issues regarding the handling of critical requirements, such as safety, security, and privacy

FIGURE 64. Candidate management reviews and objectives.

5.30.3 Additional things to think about.

Are reviews scheduled in time to meet overall system objectives and acquisition needs?
Are one or more technical reviews scheduled prior to a management review?
Are hardware, interface, and other issues from system-level working groups being addressed?
Are unresolved software/hardware/interface issues identified for resolution elsewhere if needed (such as at Interface Control Working Group or Computer Resource Working Group meetings)?
Are technical, cost, and schedule risks being surfaced?
Are realistic schedule modifications and cost estimates agreed upon?
Have action items been tracked to resolution?
Who is responsible for meeting minutes? Where are they kept? What is the level of detail? When are they distributed? Who is on distribution?
Are joint software reviews coordinated with joint reviews for the system?
Does the acquirer have the resources and schedule to support the joint reviews?

5.30.4 Related guidebook topics.

Access for acquirer review
Data accession list
Documentation (Recording information)
Oversight
Risk management
Schedules
Software management indicators
Software product evaluation

5.31 Licenses (Software).

5.31.1 Requirements summary. MIL-STD-498 provides requirements for identifying data rights (referred to in the DFARS of June 1995 as "rights in technical data and computer software") and reminds the developer to consider, as one possible criterion, license or other fees when evaluating reusable software for incorporation. (See this guidebook's topic, *Data rights*, for more information). References to ***Licenses (Software)*** in MIL-STD-498 are listed in Figure 65, with non-mandatory references indicated with italicized text.

MIL-STD-498	B.3.f, Figure 3
STP	3.x.4
STrP	3.2.f, 3.3.f, 3.4.e

FIGURE 65. MIL-STD-498's references to ***Licenses (Software)***.

MIL-STD-498's requirements regarding licenses are intended to alert the user and support agency of restrictions that may apply to the software and/or any related documentation. Software license agreements may apply to the software itself, to proprietary coding techniques, to confidential information, and/or to other trade secrets used in the development of software. A license agreement may apply to shareware, freeware, and to products derived through the use of such software, such as run-time licenses for database management systems, knowledge based systems, and other programs incorporated into the project software. Figure 66 lists developer's key activities related to ***Licenses (Software)***.

Developer's key activities related to ***Licenses (Software)***	References in MIL-STD-498
Identify the proprietary nature, acquirer's rights, and licensing issues associated with each element of the software test environment.	STP 3.x.4
Identify and describe the hardware, software, and any documentation needed to support the deliverable software, including information about vendor/manufactur support, licensing and data rights.	STrP 3.2.f, 3.3.f, 3.4.e

FIGURE 66. Developer's key activities related to ***Licenses (Software)***.

5.31.2 Acquirer's responsibilities and considerations. DFARS rights in technical data and computer software are DoD regulations that stipulate the terms and conditions that the government makes when offering to purchase technical data and computer software from a vendor. In contrast, software license agreements are the terms and conditions a vendor makes to potential buyers as a condition of sale of software products in the general marketplace.

The DFARS § 227.7202-3, June 1995, Rights in commercial computer software or commercial computer software documentation, states:

> *(a) The Government shall have only the rights specified in the license under which the commercial computer software or commercial computer software documentation was obtained.*
>
> *(b) If the Government has a need for rights not conveyed under the license customarily provided to the public, the Government must negotiate with the contractor to determine if there are acceptable terms for transferring such rights. The specific rights granted to the Government shall be enumerated in the contract license agreement or an addendum thereto.*

It is important to note that the government may require rights beyond those stated in the license. If the government requires rights in technical data and computer software beyond those provided in the license agreement, these must be negotiated between the vendor and buyer. When considering reusable software, including commercial-off-the-shelf software, the developer must be certain that the rights specified in the contract have been provided.

5.31.3 Additional things to think about.

Will license restrictions limit the acquirer's ability to support the system over its full life cycle?
What is the status of the license if the original developer goes out of business?
If licenses are transferable, is the developer authorized to make the transfer or must the original vendor provide permission?
Is the payment of royalties included in the licensing agreement?
Do license restrictions apply to the software only, software produced by the software, software in which this software (or parts of the software) is included, documentation about the software, or some other combination?
Will a separate license be required for each operational site?
Have all costs associated with COTS software been identified (e.g., initial purchase, upgrades, maintenance, special/additional hardware costs)?
Can the number of run-time licenses needed be determined at the onset of the project, or will that numb need to be negotiated at production time (possibly at a higher cost)?

5.31.4 Related guidebook topics.

Commercial-off-the-shelf software products
Data rights

5.32 Operational concept.

5.32.1 Requirements summary. MIL-STD-498 provides requirements for defining and recording an operational concept for the system. The system's operational concept describes a proposed system in terms of the user needs that it will fulfill, its relationship to existing systems or procedures, and the ways it will be used. (See this guidebook's topic, *System/subsystem*, for more information.) References to ***Operational concept*** in MIL-STD-498 are listed in Figure 67, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.3.2, *6.2*, Figure 3, Figure 4, Figure 6, *E.4.2*, *Figures 9-12*, *Figures 15-17*
OCD	All
SDP	5.3.2

FIGURE 67. MIL-STD-498's references to ***Operational concept***.

MIL-STD-498's Operational Concept Description (OCD) provides a checklist of information to be defined and recorded about the current system, if any, and the concept for the new or modified system. The operational concept description also includes: the justification for the change, such as new or modified aspects of user needs, threats, missions, objectives, environments, interfaces, personnel or other factors; deficiencies or limitations in the current system or situation that make it unable to respond to these factors; support concept for both the old and new/modified system; description of the users or involved personnel; operational scenarios; impacts to operations, organizations, and any impacts that would be experienced during the development by the user, acquirer, developer, and support agency(ies); and advantages, disadvantages/limitations, alternatives, and trade-offs considered.

The operational concept is intended to: (1) provide the developer with information needed to understand how to meet user needs by describing how the user intends to use the system; (2) provide the acquirer with the developer's understanding of how the user intends to use the system; (3) provide the support agency(ies) with information that can help define the support requirements for long-lead items (such as facilities and personnel), validate the support concept, and help support agency personnel understand the system when modifications and enhancements are needed; and (4) provide the user with confirmation that the acquirer has provided the developer with a correct interpretation of user needs and priorities for the system. Figure 68 lists developer's key activities related to ***Operational concept***.

Developer's key activities related to ***Operational concept***	References in MIL-STD-498
Describe the approach to be followed for defining and recording the operational concept for the system, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the operational concept.	SDP 5.3.2
Participate in analyzing user input, such as need statements, surveys, problem/change reports, feedback on prototypes, interviews, etc., provided by the acquirer to gain an understanding of user needs.	5.3.1
Participate in defining and recording the operational concept for the system. Include all applicable items in the OCD DID.	5.3.2 OCD
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 68. Developer's key activities related to ***Operational concept***.

5.32.2 Acquirer's responsibilities and considerations. A defined operational concept may result from a need to modify or update a current system, or may be developed in response to a newly identified threat, mission, or user need created by new technology or modes of operation. The operational concept description may be used to: (1) obtain consensus among the acquirer, developer, support, and user agencies on the operational and support concept for the proposed system, and (2) help users understand and adapt to operational and support changes needed as a result of modifications to a currently operating system undergoing reengineering, addition of new capabilities, and/or other modifications. It may also be helpful for those responsible for preparing training materials and user manuals.

The operational concept for a system may be defined before the requirements of the system are defined, or may be defined after those requirements. The operational concept is often prepared before requirements are defined when the acquirer is familiar with a similar system or is familiar with the manual operations the new system is to implement. When an operational concept description is prepared prior to requirements definition, it should be modified/updated periodically as requirements and design progress and users, acquirers, and developers become more knowledgeable about how the system will work. When the system to be developed implements new technologies with which the acquirer has little, if any, operational experience, the acquirer may ask the developer to prepare an operational concept following system requirements definition. In this case, the developer would be providing the acquirer and user the developer's vision of how the system would be used. In essence, the developer would be asking the question, "How about this?"

The operational concept can be defined by the acquirer and/or the user, by a contractor hired by the acquirer specifically to develop that concept, or by the developer of the system. An effort should be made to include all users and support agents in validating the concept to ensure that needed provisions and contingencies have been identified.

Note that DoD 5000.2-M, Defense Acquisition Management Documentation and Reports, February 1991, requires the acquirer to prepare an Operational Requirements Document (ORD). The ORD and MIL-STD-498's OCD contents overlap but have requirements for different information in some areas. The ORD, in contrast with the OCD, is not intended to be prepared by a developer, is not in DID format, and is not suitable for placing on contract unless a one-time DID is prepared for that purpose.

5.32.3 Additional things to think about.

Is the operational concept current?
Has the operational concept description been provided to all parties who might have need of it, such as users, acquirer, and developer's system and software specifiers, designers, and testers?
Has the operational concept description been reviewed by engineering specialists, such as those concerned with safety, security, privacy, supportability, logistics, and reliability?
Is the developer using the operational concept to determine system/subsystem/CSCI requirements and design?
Have the ORD and OCD, if both exist, been developed or coordinated for consistency?
If the developer defines the operational concept, are concepts defined that would limit the acquirer to only that developer, hinder the ability to adapt to changes in technology, or unduly constrain the support of the system?
Have alternative concepts been explored to the extent desired? Is the operational concept feasible?

5.32.4 Related guidebook topics.

Documentation (Recording information)
Requirements
Risk Management
System/subsystem

5.33 Oversight.

5.33.1 Requirements summary. MIL-STD-498 provides no requirements relating to oversight. The reason for this is two-fold: (1) MIL-STD-498 is a technical document and is not primarily concerned with general management, personnel, and other tasks associated with the non-technical management aspects of a project, and (2) MIL-STD-498 is not a "how to" standard, but instead focuses on "what" is to be accomplished, leaving the "how to" to the developer's project manager(s).

The standard does, however, provide requirements for activities that support the developer's management and monitoring of the project. These activities are: planning, developer testing (unit and all levels of integration), dry run of qualification testing, software configuration management, software product evaluation, software quality assurance, corrective action, risk management, software management indicators, subcontractor management, and improvement of project processes. (See this guidebook's related topics on these subjects for additional information.)

Other tasking in the standard intended to support the acquirer's monitoring responsibilities for the project include: access to developer and subcontractor facilities, approvals, qualification tests and reports, joint technical and management reviews, interface with software IV&V agents, and coordination with associate developers.

5.33.2 Acquirer's responsibilities and considerations. An acquirer has two primary responsibilities: (1) the acquirer is responsible to the DoD service or government agency funding a system or software development for cost-effective expenditure of resources allocated to the project, and (2) the acquirer is responsible to the user for providing a quality product that meets user needs within project cost and schedule constraints. To exercise these responsibilities appropriately, an acquirer needs insight into the developer's processes for developing, modifying, or otherwise supporting the system/software and the preliminary or revised/modified and final products relating to that system/software.

Oversight

The degree of insight into a project an acquirer needs to carry out these monitoring responsibilities may depend on such factors as:

- Project size
- Criticality of the system/software
- Program strategy
- Contract type
- Support concept
- Longevity of the system/software
- Development costs
- Past experience with a developer
- Capability assessment reports
- Familiarity with project personnel
- Senior management or Congressional interest
- Prior software acquisition management experience
- Developer's understanding of the problem domain
- Developer's track-record of successful projects in the specific technology/domain

DoDI 5000.2 requires that the DoD's program manager report periodically on the status of the project at identified milestones to show that key criteria established for a phase of the project have been met. One way an acquirer gains the information needed to determine that the criteria have been met is through insight into processes used and products developed over the course of the project. Key information to provide such insight includes:

- Approach (methods/procedures/tools) to be used for performing the required activities
- Risks associated with each of the activities and plans for dealing with them
- Information and products resulting from performance of required activities
- Results from evaluation of products and processes, including management indicators
- Other ongoing cost and schedule information

Project plans, such as the SDP, contain information intended to provide visibility into the approach to be used to perform activities, risks associated with those activities, and plans for dealing with them. These plans are defined by the developer and associated subcontractors, if any, coordinated with other plans for the system, and approved by the acquirer. MIL-STD-498's plans provide insight for both the developer and acquirer. The insight can help determine whether the project is on track (e.g., methods/procedures/tools selected are performing as planned; schedules are adhered to; and products are produced in accordance with requirements). The intent of MIL-STD-498's planning requirements is the establishment of a software development process by which products can be developed that meet user needs within the acquirer's cost and schedule constraints for the project.

Specifications, design descriptions, test procedures and reports, and manuals, provide visibility into the technical progress being made on the project. Since these are technical descriptions of the product and its qualification procedures, the acquirer may need specialists to assist in the evaluations. The acquirer may want evaluations from those in related engineering disciplines, such as safety, security, maintainability, logistics, training, etc., as well as users and others with knowledge of the task(s) or mission that needs to be accomplished. Such evaluations support the acquirer's two primary responsibilities; identify risk areas; and help both the developer and acquirer prepare for joint technical and management reviews. The evaluation criteria listed in Appendix D of the standard (or approved alternative) can be used by both the developer and the acquirer when performing evaluations. When both acquirer and developer use the same set of criteria, discussions can be more productive.

In addition, an acquirer's project monitoring responsibilities may include involvement in and coordination among different working groups associated with one or more systems under development or planned for the future (e.g., Computer Resource Working Group (CRWG), Test Evaluation Working Group (TEWG), Interface Control Working Group (ICWG), etc.). These working groups are frequently focal points for raising and resolving issues in specific areas and for gaining information from resources normally not associated with a specific procurement.

MIL-STD-498's specifications (System/Subsystem Specification (SSS), Interface Requirements Specification (IRS), Software Requirements Specification (SRS), Software Product Specification (SPS)) contain requirements for the system and its software. MIL-STD-498 defines requirements as those characteristics that a system or CSCI must possess to be acceptable to the acquirer. An acquirer frequently makes such specifications a part of a contract or other agreement between the acquirer and the developer. Changes to specifications that are part of the contract are subject to contractual modification rules and procedures. If specifications are not on contract, the acquirer and developer will usually establish procedures for formalizing and controlling changes to the requirements. A set of agreed upon requirements is often referred to as a "baseline." In both cases, the requirements, along with the developer's plans, form the basis for an acquirer's monitoring activities.

5.33.3 Additional things to think about.

Is there a clear understanding between the acquirer and developer on who will grant approvals to proceed and who will provide other needed feedback on the developing products?
Is the developer silent or vague in implementing standards or acceptance criteria for products or project processes?
Is there acquirer and developer agreement on specifications, design constraints, special development methods (e.g., methods for safety or security critical software) and qualification test plans and results?
Does the acquirer provide timely input to working groups (CRWG, TEWG, ICWG) and get timely information from those groups applicable to the project?
Are acquirer plans for software coordinated with an acquirer's system management plans?
Is there sufficient reporting or disclosure of risk items for timely management action?
Does the developer consider and trade-off alternatives for planning, risk abatement, corrective action or replanning efforts? Are these alternatives presented and discussed to help the acquirer maintain confidence that project planning/replanning is sound?
Does the acquirer monitor progress against approved plans?
Does the acquirer encourage use of software management indicators for monitoring developers? Does the prime contractor use management indicators to monitor subcontracted software activities?

5.33.4 Related guidebook topics.

Approval by the acquirer
Corrective action
Independent verification and validation
Integrated Product Team
Joint technical and management reviews
Process improvement
Qualification testing
Risk management
Software configuration management
Software development planning
Software management indicators
Software product evaluation
Software quality assurance
Subcontractor management
Testing (Developer-internal)

5.34 Problem category and priority classifications.

5.34.1 Requirements summary. MIL-STD-498 provides requirements for assigning a category and priority classification for each problem submitted to the corrective action system. These categories and priorities are listed in Figures 4 and 5 of MIL-STD-498 and provide one method for grouping problems for use in identifying, tracking, and resolving them. References to ***Problem category and priority classifications*** in MIL-STD-498 are listed in Figure 69, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.17.2.c, *Figure 2*, Appendix C, Figures 4-5, *F.3.f*
SDP	5.17

FIGURE 69. MIL-STD-498's references to ***Problem category and priority classifications***.

MIL-STD-498 allows the developer to use alternate category and priority schemes if approved by the acquirer. Problem category and priority classifications may be used informally but are intended for use as part of the developer's closed-loop corrective action system. (See this guidebook's topics, *Corrective action* and *Problem/change report*, for more information). Figure 70 lists developer's key activities related to ***Problem category and priority classifications***.

Developer's key activities related to ***Problem category and priority classifications***	References in MIL-STD-498
Describe the approach to be followed for problem category and priority classification, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to problem category and priority classification.	SDP 5.17
Classify each problem by category and priority, using the categories and priorities in Appendix C or approved alternatives.	5.17.2.c, Appendix C, Figures 4-5
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 70. Developer's key activities related to ***Problem category and priority classifications***.

5.34.2 Acquirer's responsibilities and considerations. Priority classification is used as an indication of the severity of the problems from the user's/acquirer's viewpoint. The category classifications provide useful "buckets" in which to group problems. Priorities 1 and 2 focus on cataclysmic problems, such as the mission cannot be accomplished or is adversely affected and no work-around solution is known; the system/subsystem or CSCI is unsafe, not secure, or

unable to perform required critical requirements; or the problem causes technical, cost, or schedule risks, or life cycle support is adversely affected and no work-around solution is known.

MIL-STD-498's software problem category and priority classifications provide a developer with a default method of identifying the sources of problems and the severity of those problems. They do not provide criteria for scheduling correction of those problems, i.e., not all Priority 1's must be corrected before Priority 2's, before Priority 3's, etc.

An acquirer may use problem category and priority classifications to judge whether a system/subsystem or CSCI is ready for a particular acquirer event. For example, a milestone decision to move into production with a system might be predicated upon the absence of Priority 1, 2, and 3 problems; the decision to field an early build might depend upon there being no Priority 1 problems that would jeopardize safety, security, or other requirement designated "critical" for that build; or a version under refinement during maintenance might not be released until all Priority 1, 2, 3, and 4 problems are resolved and corrected.

DoD's policy regarding readiness for system Operational Test and Evaluation (OT&E) states that:

> Software maturity must be demonstrated prior to the start of the dedicated OT&E. At the operational test readiness review that precedes the OT&E, the system must not possess any known Priority I or II problems that impact the OT&E so as to constitute a deficiency relative to a critical operational issue and Priority III problems must be documented with appropriate impact analyses completed.

Figure 71 shows a comparison between DOD-STD-2167A's requirements and MIL-STD-498's. The following differences should be noted:

Priority 1: MIL-STD-498 adds the criterion "jeopardizes security, privacy, and other critical requirements" to DOD-STD-2167A's "jeopardizes safety" criterion.

Priority 2: MIL-STD-498 adds the criterion that the problem is Priority 2 if it adversely affects technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known.

Priority 3: MIL-STD-498 adds the criterion that the problem is Priority 3 if it adversely affects technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known.

Priority 2 and 3 classifications add criteria related to technical, cost or schedule risks. These criteria were added to ensure that systems known to be too expensive or to take too long to produce, or whose technologies are too fragile are identified as problem systems.

DOD-STD-2167A	MIL-STD-498	Analysis
Priority 1: A software problem that does one of the following:	**Priority 1 applies if the problem could:**	
1. Prevents the accomplishment of an operational or mission essential capability specified by baselined requirements	a. Prevent the accomplishment of an operational or mission essential capability	No change: 498 summarizes DOD-STD-2167A 1 & 2
2. Prevents the operator's accomplishment of an operational or mission essential capability specified by baselined requirements		
3. Jeopardizes personnel safety	b. Jeopardize safety, security, or other requirement designated "critical"	498 adds security and other critical requirements
Priority 2: A software problem that does one of the following:	**Priority 2 applies if the problem could:**	
1. Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which no alternative work-around solution is known	a. Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known	No change: 498 summarizes DOD-STD-2167A 1 & 2
2. Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which no alternative work-around solution is known		
	b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known	498 adds this criterion for Priority 2
Priority 3: A software problem that does one of the following:	**Priority 3 applies if the problem could:**	
1. Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which an alternative work-around solution is known	a. Adversely affect the accomplishment of an operational or mission essential capability but a work-around solution is known	No change: 498 summarizes DOD-STD-2167A 1 & 2
2. Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which an alternative work-around solution is known		
	b. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known	498 adds this criterion for Priority 3

FIGURE 71. Comparison of problem category and priority classifications between DOD-STD-2167A and MIL-STD-498.

5.34.3 Additional things to think about.

Are operational, mission essential, and other critical requirements clearly identified?
Are too many problems classified as operational or mission essential?
Are the alternative work-around solutions for problems assigned priority 3 documented? Are the solutions acceptable?
Are technical, cost, and schedule problems identified?
Is the developer able to identify system production cost impacts of software problems, if any? Other impacts on the system?

5.34.4 Related guidebook topics.

Corrective action
Problem/change report
Qualification testing
Software configuration management
Software management indicators
Software product evaluation
Software quality assurance
Testing (Developer-internal)

5.35 Problem/change report.

5.35.1 Requirements summary. MIL-STD-498 provides requirements for the developer to prepare a problem/change report to describe each problem detected in software products under project-level or higher configuration control and each problem in activities required by the contract or described in the SDP. The term, "problem/change report," is a generic term, not necessarily the name of the actual report that is filed. Actual reports may be called Software Problem Reports, Trouble Reports, Program Trouble Reports, or other names selected by a project or organization. References to ***Problem/change report*** in MIL-STD-498 are listed in Figure 72, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.3.1, 5.14.3, 5.15.2, 5.16.2, 5.17, Figures 4-5, *F.3.f*
SDP	5.17
STR	3.1.b, 4.x.2.y
SVD	3.3, 3.7

FIGURE 72. MIL-STD-498's references to ***Problem/change report***.

The problem/change report describes the problem, the corrective action needed, and the actions taken to date. It serves as input to the corrective action system. Submission of problem/change reports is not limited strictly to the developer. Others, such as users, support agency(ies), IV&V agents, the program office, and other authorized acquirer representatives, may submit problem/change reports. The problem/change report is the principal interface mechanism between problem identifiers and the corrective action system.

In addition to serving as the principal input to the corrective action system, problem/change reports serve as an important part of record keeping. The developer is required to:

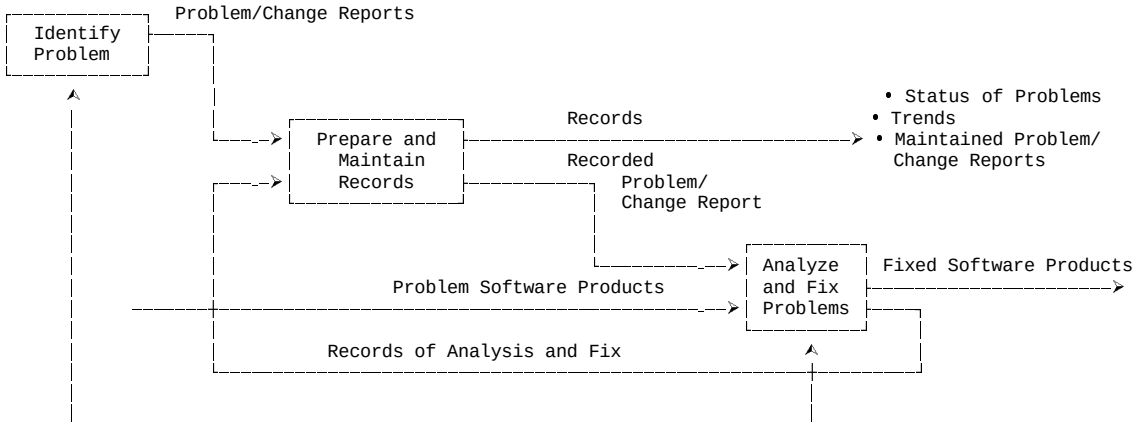
- Maintain records (including problem/change reports) of the configuration status of all entities under project-level or higher configuration control
- Prepare and maintain records (including problem/change reports) of each software product evaluation
- Prepare and maintain records (including problem/change reports) of each software quality assurance activity
- List all changes and associated problem/change reports incorporated in new software versions

Figure 73 lists developer's key activities related to ***Problem/change report***.

Developer's key activities related to ***Problem/change report***	References in MIL-STD-498
Describe the approach to be followed for problem/change reports, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to problem/change reports.	SDP 5.17
Participate in analyzing user input provided by the acquirer, including problem/change reports, to gain understanding of user's needs.	5.3.1
Identify any remaining deficiencies, limitations, or constraints detected during testing (problem/change reports may be used for reporting deficiencies). Identify test cases in which problems occurred, referencing associated problem/change reports.	STR 3.1.b, 4.x.2.y
List all changes, and associated problem/change reports, incorporated into the software version since the previous version, and possible problems or known errors at the time of release.	SVD 3.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 73. Developer's key activities related to ***Problem/change report***.

5.35.2 Acquirer's responsibilities and considerations. Problem/change reports document problems with both software products and software development activities. Problem/change reports are used for problems in software products after the product comes under project-level or higher configuration control. Problem/change reports can serve as sources of information to support generation of software management indicators. They may be submitted on software development activities at any time. Problem/change reports are primarily for developer internal use, whereas Engineering Change Proposals (ECPs) or their equivalents are used to initiate contractual action and involve both the developer's and acquirer's contracting organizations. Although problem/change reports are not formal deliverables, the acquirer can obtain copies of them via the Data Accession List or may schedule a review of them at the developer's facility. Problem/change reports are used to initiate a corrective action within the development environment. The corrective action system provides the administrative records of the action taken. Figure 74 shows that the corrective action system prevents undisciplined development actions ensuring that problems and proposed changes are identified, recorded, analyzed, and corrected, as appropriate; the status of problems and changes is tracked; and actions taken to date are identified, maintained, and reported.



INITIATORS

Development Team Members, such as:
- Specifiers of Requirements
- Designers
- Coders
- Testers

Others, such as:
- Users
- Evaluators
- IV&V Agents
- Management
- Associate Contractors
- Subcontractors

FIXERS

Development Team Members, such as:
- Specifiers of Requirements
- Designers
- Coders
- Testers

Others, such as:
- Management
- Associate Contractors
- Subcontractors

FIGURE 74. Corrective action system interfaces.

Problem/change reports are for internal developer use, therefore, no DID is required. The developer can define content and format. Although the problem/change report content and format is developer-defined, the SDP identifies candidate topics a problem/change report might contain, as follows:

- analysis time
- analyst assigned
- approval of solution
- category and priority
- correction date
- correction time
- corrector
- date assigned
- date completed
- description
- description of solution implemented
- document affected
- follow-up actions
- impacts
- origination date
- originator
- problem name
- problem number
- problem status
- project name
- recommended solution
- software element or document affected
- version where corrected

5.35.3 Additional things to think about.

Are problem/change reports generated promptly by the developer?
Does each problem/change report clearly describe the problem, corrective action needed, and accurately record actions taken to date? Is all necessary information included in the problem/change report?
Does the developer encourage submission of problem/change reports by users and others?
Is problem/change report information analyzed and acted upon?

5.35.4 Related guidebook topics.

Access for acquirer review
Corrective action
Independent verification and validation
Process improvement
Software configuration management
Software development planning
Software management indicators
Software quality assurance
Version/revision/release

5.36 Process improvement.

5.36.1 Requirements summary. MIL-STD-498 provides requirements for the developer to periodically assess the software development process used on the project to determine suitability and effectiveness. Based on these assessments, the developer is to identify necessary and beneficial improvements to the acquirer in the form of proposed updates to the SDP. If approved, the developer implements the improvements on the project. References to ***Process improvement*** in MIL-STD-498 are listed in Figure 75, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.27*, 5.19.7, *Figures 9-12*
SDP	5.19.7

FIGURE 75. MIL-STD-498's references to ***Process improvement***.

The developer is required to describe the approach to be followed for process improvement activities in the SDP for the project, including applicable risks/uncertainties, and plans for dealing with them. If additional requirements regarding process improvement have been added to the SOW or contract, the description is to cover all contractual clauses. Process improvement is an ongoing activity throughout the software development process. Figure 76 lists developer's key activities related to ***Process improvement***.

Developer's key activities related to ***Process improvement***	References in MIL-STD-498
Describe the approach to be followed for improvement of project processes. Cover all contractual clauses regarding process improvement.	SDP 5.19.7
Periodically assess the processes used on the project to determine their suitability and effectiveness. Identify any necessary and beneficial improvements, identify these to the acquirer in the form of proposed updates to the SDP and, if approved, implement them on the project.	5.19.7
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 76. Developer's key activities related to ***Process improvement***.

5.36.2 Acquirer's responsibilities and considerations. MIL-STD-498's focus is on improvements to a specific project covered by a contract. The intent of MIL-STD-498's process improvement requirement is to ensure that improvements needed to ensure project success, to provide cost/time savings, or to accrue other benefits for the project are identified and proposed. Although a project's process improvement activities may provide recommendations for improving a developer's overall process, that is not its purpose. The standard's process improvement activities are project focused. Recommended project process improvements may result from assessments of key process areas in a Capability Maturity Model/Software Capability Evaluation, or other process improvement or quality management activity performed by the developer's organization. While MIL-STD-498's process improvement activities are intended for the project, a no-cost benefit may be gained by other projects when lessons learned on one project benefit another.

A developer's process improvement activities often include:

- Establishing a focal point within the developer's project team to organize the process improvement activity and keep it on track
- Creating the means to identify processes needing improvement
- Providing the right people to create solutions to the problem
- Assessing the improvement activities

Process improvement recommendations are most often made by the developer's project personnel. Project personnel are close to the process, know what isn't working, know what needs improvement, and often know how to make the improvements. Process improvement forms, if readily available to project personnel, can provide input to the process improvement activity. Other MIL-STD-498 requirements that support process improvement are:

- Problem reports/corrective action reports
- Joint technical and management reviews
- Software quality assurance
- IV&V
- Software management indicator trends

5.36.3 Additional things to think about.

Is the developer so immersed in "getting the job done" that the requirement to look for ways to improve is ignored?
Are developer processes that involve the project's software development process and interfaces with other parts of the company properly addressed?
Does the acquirer have sufficient visibility into the project's software development process and process improvement activities?
Is there a quantifiable benefit (cost/time saving, quality improvement, product functionality) to the project from the proposed improvement?

5.36.4 Related guidebook topics.

Corrective action
Joint technical and management reviews
Problem/change report
Software development process
Software management indicators
Software quality assurance

5.37 Programming languages.

5.37.1 Requirements summary. MIL-STD-498 provides no requirements to use a particular programming language. It does, however, provide requirements regarding the use of any programming language. MIL-STD-498 also provides requirements for the developer to obtain approval to use any programming language not specified in the contract. References to ***Programming languages*** in MIL-STD-498 are listed in Figure 77, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.29,* 4.2.2, 5.7.1, 5.13.6, *6.2*, *Figure 2*, Figure 6 (Item 27)
CPM	All
DBDD	5.x.c
SDD	5.x
SDP	4.2.2, 5.7.1, 5.7.2, 5.13.6
SRS	3.12.b
SSS	3.12.b

FIGURE 77. MIL-STD-498's references to ***Programming languages***.

MIL-STD-498 provides requirements for the developer to develop and apply standards for representing code. The developer's SDP for the project contains or references these standards. Coding standards are to include standards for: format, header comments, naming conventions and any restrictions on the use of programming language constructs or features, such as renaming, go to's, etc. (See this guidebook's topic, *Standards for software products,* for more information.)

Note that the standard also provides requirements for developing a Computer Programming Manual (CPM) when special-purpose computers are used, or when other computers, for which commercial or other programming manuals are not readily available, are developed or used. The CPM is not a manual about programming languages, such as MIL-STD-1815A, Ada (also referenced as ANSI-1815A-1983, ISO 8652-1987, or FIPS PUB 119, Federal Information Processing Standard), but rather a manual focusing on the instruction set architecture used to program the computer itself. Development of the CPM is covered under this guidebook's topic, *Software support manuals*. Figure 78 lists developer's key activities related to ***Programming languages***.

Developer's key activities related to ***Programming languages***	References in MIL-STD-498
Describe the approach to be followed for programming languages, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to programming languages.	SDP 4.2.2, 5.7.1, 5.13.6
Develop and apply standards for representing code. Describe these standards in the software development plan.	4.2.2
Develop and record software corresponding to each software unit in the CSCI design, include coding computer instructions and data definitions, building databases, populating databases or other data files with data values, and other activities needed to implement the design. Obtain acquirer approval to use any programming language not specified in the contract.	5.7.1
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 78. Developer's key activities related to ***Programming languages***.

5.37.2 Acquirer's responsibilities and considerations. MIL-STD-498 has no requirements specifying the programming language in which source code is to be developed. MIL-STD-498 is usable with any programming language. MIL-STD-498 provides requirements for the developer to obtain approval to use any language not specified in the contract. Current DoD policy mandates the Ada programming language for new starts and requires Ada for upgrades if more than 30% of the code is to be modified.

The specification of a source language can be a difficult choice for system enhancements, reengineering efforts, and major system upgrades where existing code is other than Ada. Different DoD services or government agencies may have different policies/guidelines regarding this issue. The acquirer and developer may need to evaluate trade-offs such as:

- Programming new source code using the same language as existing software to maintain consistency of languages versus implementing new source code in Ada, thereby moving to the mandated language but creating a "mixed bag" of languages

- Programming new source code in Ada and translating existing code to Ada avoiding the "mixed bag" versus the costs and risks of reengineering software that works "as is"

5.37.3 Additional things to think about.

Are DoD directives and instructions regarding the use of a specific programming language applicable to the project?
Will waivers be needed for any specified programming languages? Have criteria for waivers been identified?
Has approval been obtained for each programming language planned for use? Has support agency input been considered as a factor in the approval process?
Do the developer and subcontractors have experience with the compiler(s) and programming tools (debuggers, etc.) chosen for the project?
Has a trade-off study been conducted for reengineering projects to determine if it is cost-effective to translate legacy software?

5.37.4 Related guidebook topics.

Computer hardware resource utilization
Detailed design
Executable software
Software implementation and unit testing
Software support manuals
Source files
Standards for software products

5.38 Qualification testing.

5.38.1 Requirements summary. MIL-STD-498 provides requirements for the developer to perform CSCI qualification testing and participate in system qualification testing. The standard defines qualification testing as "*testing performed to demonstrate to the acquirer that a CSCI or a system meets its specified requirements*." Requirements are those characteristics of a system/subsystem or CSCI that are conditions for its acceptance by the acquirer. MIL-STD-498 has two activities focused on testing those requirements: (1) CSCI qualification testing and (2) system qualification testing. To cover systems containing both hardware and software (such as radar systems) and software-only systems (such as payroll systems), the standard includes systems-level activities and requires the developer to "participate in" them. For systems involving hardware and software development, "participate in" is interpreted to mean "*take part in, as described in the software development plan*." For software systems, "participate in" means "*be responsible for*." References to ***Qualification testing*** in MIL-STD-498 are listed in Figure 79, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.28, 3.43,* 4.1, *Figure 1*, 5.1.2, 5.1.3, 5.2.2, 5.3.3, *5.4.1 Note*, 5.5, 5.9, 5.11, Figure 3, B.4, Figure 4, Figure 6 (Items 2, 14, 15, 16, 17), D.4.2, D.4.12, *E.4.7, E.4.8, Figures 9-13*
IRS	4
SDP	5.1.2, 5.1.3, 5.2.2, 5.3.3, 5.5, 5.9, 5.11
SRS	4
SSS	4
STD	All
STP	All
STR	All

FIGURE 79. MIL-STD-498's references to ***Qualification testing***.

Requirements for CSCIs are usually specified in Software Requirements Specification(s) (SRSs) and/or Interface Requirements Specification(s) (IRSs); system/subsystem requirements are specified in a System/Subsystem Specification (SSS) and/or an IRS(s). Although subsystem qualification testing is not identified as an activity, like subsystem requirements and design activities, system level testing activities may also be applied to subsystems. Figure 80 lists developer's key activities related to ***Qualification testing*** (CSCI and system).

Developer's key activities related to ***Qualification testing***	References in MIL-STD-498
Describe the approach to be followed for qualification testing, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to qualification testing.	SDP 5.1.2, 5.1.3, 5.2.2, 5.3.3, 5.5, 5.9, 5.11
Develop and apply standards for representing test cases, test procedures, and test results. Describe or reference the standards to be followed in the SDP.	4.2.2 SDP 4.2.2
Develop and record plans for conducting CSCI qualification testing and "participating in" system qualification testing. Include all applicable items in the Software Test Plan (STP) DID.	5.1.2, 5.1.3 STP
Establish, control, and maintain a software test environment to perform qualification, and possibly other, testing of software. Ensure that each element of the environment performs its intended function.	5.2.2
Define a set of qualification methods for each requirement identified in the SSS, IRS, and SRS to be used to ensure that the requirement has been met.	5.3.3, 5.5 IRS 4 SRS 4 SSS 4
Prepare for qualification testing. Define and record: (1) test preparations; (2) detailed descriptions of each test case including requirements addressed, prerequisite conditions, inputs, expected results, criteria for evaluating results, test procedures, and assumptions and constraints; and (3) traceability between test cases and requirements. Prepare test data. Provide the acquirer advance notice of the time and location of qualification testing.	5.9.3, 5.11.3 STD
Identify person(s) responsible for qualification testing who did not perform detailed design or implementation of the software.	5.9.1, 5.11.1
If the acquirer is to witness qualification testing, perform a dry run of the test cases and procedures. Record the results of this activity in appropriate SDF(s) and update the test cases and procedures as appropriate.	5.9.4, 5.11.4 STD
Perform qualification testing on the target computer system or an alternative system approved by the acquirer in accordance with the test cases and procedures.	5.9.2, 5.9.5, 5.11.2, 5.11.5 STD
Make necessary revisions to the software, provide the acquirer advance notice of retesting, conduct all necessary retesting, and update the SDFs and other software products as needed, based on the results of qualification testing.	5.9.6, 5.11.6 STD
Analyze and record the results of qualification testing. Include all applicable items in the Software Test Report (STR) DID.	5.9.7, 5.11.7 STR
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 80. Developer's key activities related to ***Qualification testing***.

The standard provides a set of information to be defined and recorded regarding system/subsystem and CSCI qualification. This set of information is found in the standard's Software Test Plan (STP), Software Test Description (STD), and Software Test Report (STR) DIDs. Figure 81 presents a summary of this information.

Qualification Testing Document	Summary
Software Test Plan (STP)	Describes plans for qualification testing of CSCIs and software systems. (1) Describes the software test environment including test sites, software items, hardware and firmware items, proprietary nature of test environment items, installation/check out/control of test items, participating organizations, personnel, and orientation/training required; (2) identifies the tests to be performed including test objectives, test level (CSCI or system), test classes, conditions, and sequencing, data recording/reduction/analysis; (3) provides schedules for test activities, and (4) contains requirement traceability between CSCI/software system requirements and the test(s) that address them.
Software Test Description (STD)	Describes the test preparations, test cases, test procedures, and requirement traceability to be used to perform qualification testing of a CSCI or a software system or subsystem. Preparations include providing specific hardware to be used, switch settings, step-by-step start-up instructions, specific software and associated storage media, software loading instructions, and any other pre-test personnel actions, preparations, and procedures necessary to perform the test. Test cases include prerequisite conditions, test inputs, expected test results, and criteria for evaluating results. Test procedures consist of operator inputs and actions, expected result and evaluation criteria for each step, actions to follow in the event of a program stop or indicated error, and procedures to be used to reduce and analyze test results. Traceability is mapped between CSCI/software system requirements and test cases that address them.
Software Test Report (STR)	Provides for a record of the qualification testing performed on a CSCI, software system, or subsystem. Provides (1) an overview of the test results including overall assessment, impact of test environment, recommended improvements; (2) detailed test results including for each test a summary, problems encountered, test case description, and deviations from test cases/procedures; and (3) a test log with dates/times/locations of tests, hardware/software configurations, and a detailed log of test activities, personnel, and witnesses.
Software Development File	Contains a record of the "dry run" of a qualification test.

FIGURE 81. Overview of required qualification testing information.

If a system contains reusable software, the standard provides requirements in Figure 3, interpreting the standard's testing requirements for any reusable software incorporated into the system. Appendix B of the standard provides interpretations based on whether the software is being used "as is" or modified, whether the software is a CSCI or software unit, and whether the software has a positive performance record or no or poor performance record. Problems

found during qualification testing can be reported in accordance with procedures for the corrective action system described in the SDP. When problems found in qualification testing are not with the software itself but with test plans, descriptions, and reports, a special problem category labeled "test information" has been provided for classifying those problems (see MIL-STD-498, Figure 4).

5.38.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for developer CSCI and system qualification testing. Both CSCI and system qualification testing include planning for qualification testing, preparing the software and hardware for qualification testing, defining and recording qualification test cases and procedures, tracing requirements to qualification test cases, performing dry run tests, performing the actual tests, analyzing and recording test results, and revising and retesting when necessary, (i.e., revising and retesting may occur during and after dry run testing, or during and after qualification testing, as needed).

Although requirements are provided for qualification testing at CSCI and higher levels, qualification testing need not be required for each level. The acquirer may elect to require qualification testing of each CSCI, or may defer qualification to a group of CSCIs, a subsystem, or wait to qualify the system as a whole. For example, if a system consists of several CSCIs, each may be qualified separately. Alternatively, the acquirer may decide that it makes more sense to delay qualification to the subsystem level. Other alternatives might be to qualify the CSCI in conjunction with other CSCI(s) and/or HWCI(s), or delay qualification until all system elements have been integrated and tested. There are many options. Whether to qualify a component at a lower-level or not is frequently contract dependent. That is, payment to a developer may depend on successful accomplishment of the qualification testing to meet contractual obligations.

Delaying qualification testing to higher levels does not mean that thorough testing will not be performed at lower levels. Qualification testing is only that testing needed to demonstrate to the acquirer that requirements have been met. Developer-internal testing is performed at all levels of development: unit, unit integration, CSCI, CSCI integration, subsystem, and system. Developer-internal testing is performed to test all aspects of the software, including, but not limited to, system-wide, CSCI-wide, and architectural design. The results of developer-internal testing are recorded in the appropriate SDF(s). (See this guidebook's topics, *Software development files, Software implementation and unit testing*, and *Testing (Developer-internal)*, for more information.) Figure 82 shows a progression of developer-internal design and related tests to be performed.

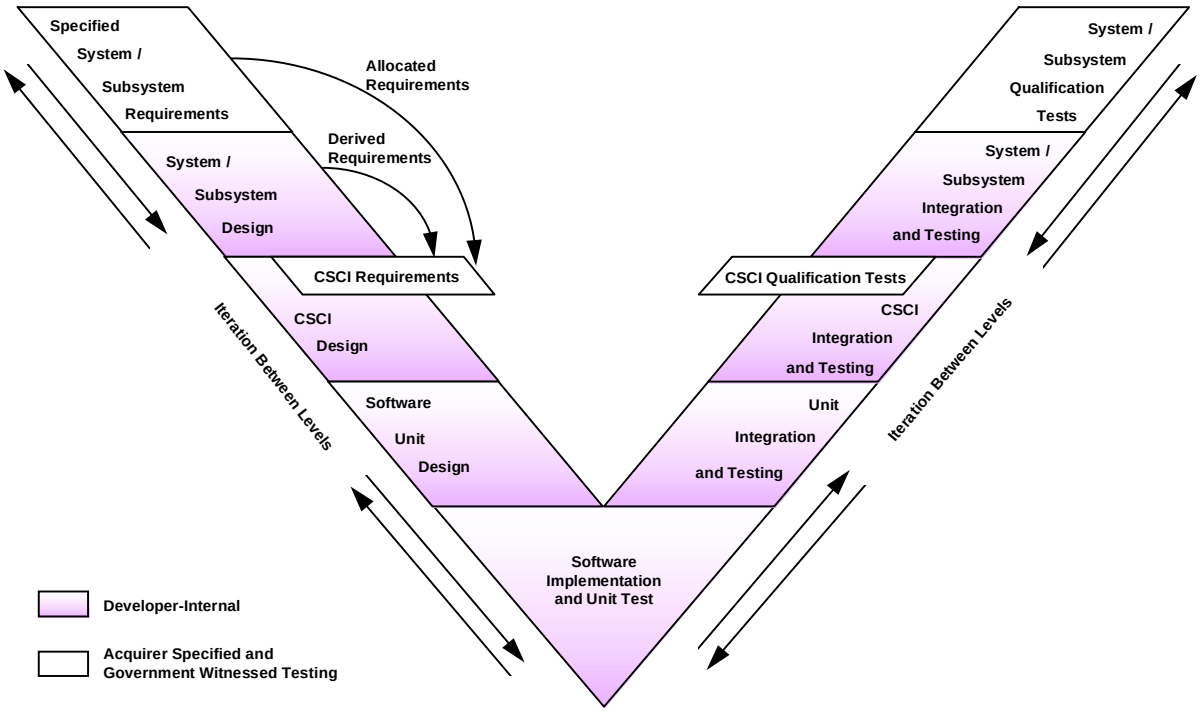


FIGURE 82. Developer-internal design and related tests.

The standard provides three DIDs that describe the information to be defined and recorded for the qualification testing effort. Each may be used to define and record information applicable to the level of qualification testing required. For example, there may be an STP, STD, and STR for qualification of a single CSCI; if more than one CSCI needs to be qualified, there may be one STP covering all CSCIs, each CSCI with its own separate STD and STR; if a system consists of subsystems, there may be an STP covering one or more subsystems, each subsystem, group of subsystems, or CSCI with its own STD(s) and STR(s), and so on.

MIL-STD-498 provides requirements for independence in qualification testing. This does not mean that the developer must have an independent test organization to perform this testing. Although the standard prohibits those who performed detailed design or implementation of the software undergoing testing from being responsible for qualification testing activities, the standard does not preclude those who performed detailed design or implementation of the software from contributing to the qualification testing process, for example by contributing test cases that rely on knowledge of the software's internal implementation.

The standard's requirements for independence in qualification testing are intended to ensure that those responsible for developing the products undergoing qualification testing are not responsible for demonstrating that the products tested provide the required characteristics. The requirement for independence exists not so much from a concern that those who have performed the activity or developed the product will qualify systems that do not provide the required characteristics, but rather from the knowledge that a second perspective may be needed to "see" that a requirement has been implemented correctly. If those responsible for the detailed design and implementation of the requirement have misinterpreted the requirement from the beginning, they are not likely to "see" that that interpretation is incorrect when developing test cases and procedures for the requirement. (See this guidebook's topics, *Software product evaluation* and *Software quality assurance*, for more information regarding independence.)

The standard provides a list of qualification methods from which to choose. Each requirement is to be qualified by one or more of the methods described. The methods are: demonstration, test, analysis, inspection, and/or special qualification methods. Special qualification methods include the use of special tools, techniques, procedures, facilities, acceptance limits, use of standard samples, preproduction or periodic production samples, pilot models, or pilot lots. The SSS, IRS, and SRS provide the flexibility to present the correlation of requirement to qualification method in a way that makes sense for the project, such as in a table, by annotating each requirement with the qualification method(s) to be used wherever the requirement is recorded, or some other way depending upon recording methods used (e.g., CASE tools or databases).

Discussion of qualification testing plans and other aspects of qualification testing can be held during joint technical and management reviews. The standard provides a candidate set of management reviews which can be used. The developer can propose the use of these reviews or other reviews to provide the acquirer insight into the project's qualification testing aspects. MIL-STD-498's candidate reviews provide three reviews intended to focus on qualification testing: (1) software planning review(s), which may cover one or more of the planning descriptions, such as the software test plan, (2) test readiness reviews to cover the status of the software test environment, test cases and procedures, and the software to be tested, and (3) test results reviews to cover one or more levels of qualification testing, or one or more groups of items being qualified.

Even though thorough developer-internal testing and a dry run of the qualification testing have been performed, it is still possible for errors to occur during qualification testing. This situation is most likely to occur if simulators have been used for developer-internal and dry run testing, while qualification testing is to take place on the target computer, or when hardware or other

equipment has been delayed and is available for the first time during qualification. The developer's test planning should identify such risks so that time can be allocated for retesting, if needed. Figure 83 shows qualification testing activities.

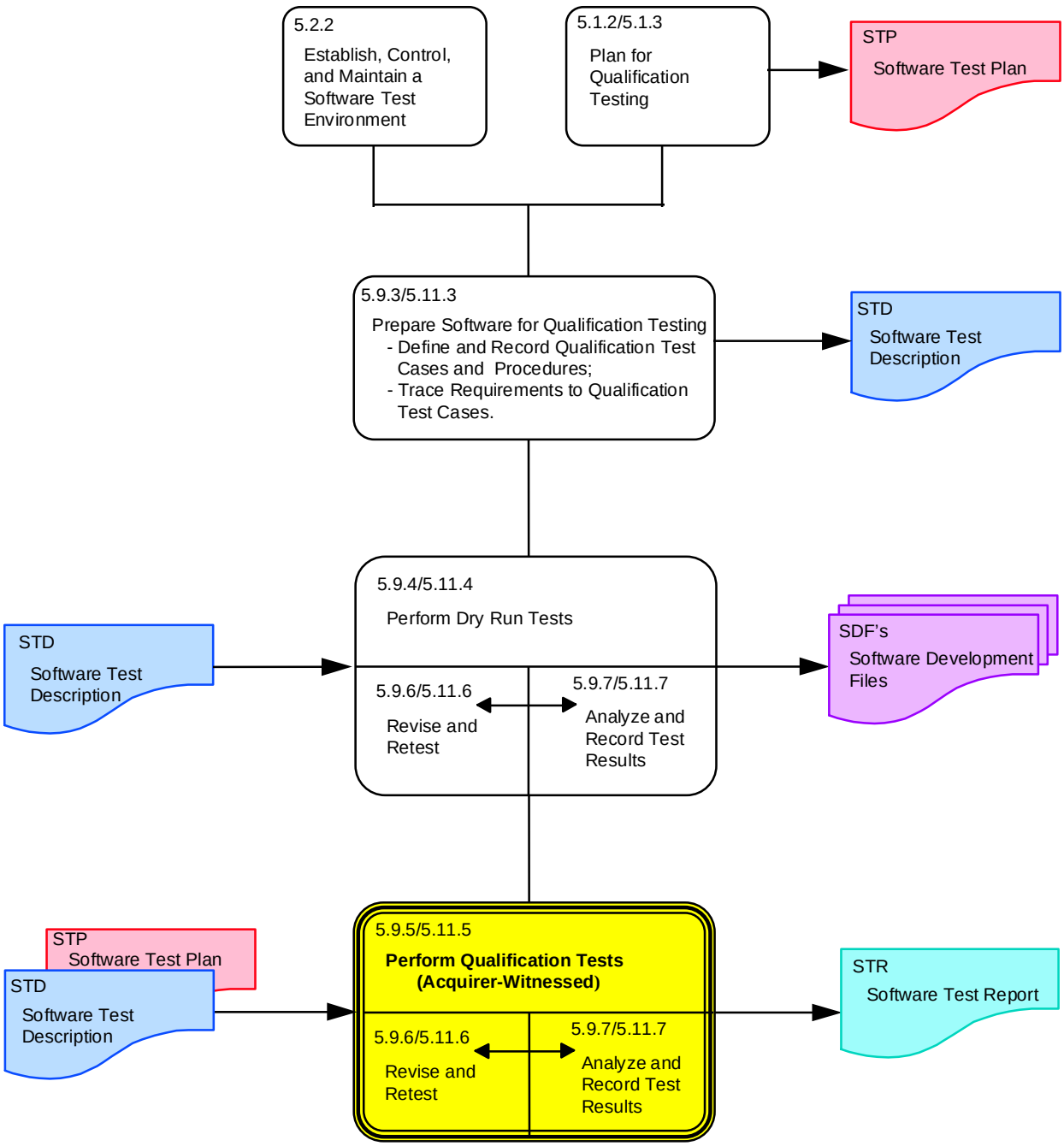


FIGURE 83. Qualification testing activities.

5.38.3 Additional things to think about.

Will software, computers, and other necessary hardware be ready in time for qualification testing? If not, have alternatives been identified and approved?
Has acquirer furnished equipment been identified? Have those responsible for providing acquirer furnished equipment been notified of the planned dates for qualification testing? Can equipment be provided in time for qualification preparations to be complete?
Do developer-internal test results indicate that the software functions correctly, doesn't stop or break, and that errors are reduced to an acceptable level?
Has the acquirer identified program personnel or their authorized representatives for witnessing qualification testing?
Have funds been planned and allocated for the acquirer's participation in qualification testing?
Are other acquirer testing activities, such as Operational Test and Evaluation (OT&E), dependent upon qualification test results?
Have schedules for qualification testing been coordinated with acquirer testing far enough in advance so that the acquirer and the developer can make the necessary plans? Are schedules realistic? What fall-back or contingency plans are there for operational testing if the software can't be qualified?
Are interoperability tests necessary for qualification of the system? If so, has testing time been reserved far enough in advance (sometimes as much as a year) to meet other project deadlines?
If the software is qualified on an alternative computer, who will be responsible for ensuring that the software works correctly on the target system when ready?
If patches are allowed, when (if ever) will the software be recompiled to eliminate those patches? Who will be responsible for making corrections to the patched software? Who will determine how many tests need to be rerun if the software is recompiled after correcting the patches?
If software is revised and retested during qualification, how will the determination be made of how much retesting constitutes "all necessary"?
Has the amount of retesting/regression testing to complete qualification been agreed upon if software has been "accepted" with known errors? Are costs and schedules for retesting provided?

5.38.4 Related guidebook topics.

Acceptance by the acquirer
Independent verification and validation
Joint technical and management reviews
Requirements
Reusable software products
Software development files
Software implementation and unit testing
Software product evaluation
Software quality assurance
System/subsystem
Testing (Developer-internal)
Traceability

5.39 Rationale/key decisions.

5.39.1 Requirements summary. MIL-STD-498 provides requirements for the developer to record the rationale for key decisions that will be useful to the support agency. Key decisions recorded are to cover decisions made in specifying, designing, implementing, and testing the software. Included are the trade-offs considered, analysis methods, and criteria used in making the key decisions. References to ***Rationale/key decisions*** in MIL-STD-438 are listed in Figure 84, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.34*, 4.2.6, 5.2.4
All DIDs	*Notes*
SDP	4.2.6, 5.2.4

FIGURE 84. MIL-STD-498's references to ***Rationale/key decisions***.

The developer is required to define what is "key" for the project, state where those decisions will be recorded, and describe the methods/procedures/tools (approach) for providing the rationale in the project's SDP. The meaning of "key decisions" is open for developer interpretation. What may be a key decision on one project may not be on another. Figure 85 lists developer's key activities related to ***Rationale/key decisions***.

Developer's key activities related to ***Rationale/key decisions***	References in MIL-STD-498
Describe the approach to be followed for defining and recording the rationale for key decisions, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the rationale for key decisions.	SDP 4.2.6, 5.2.4
Record rationale for key decisions made in specifying, designing, implementing, and testing the software. Include trade-offs considered, analysis methods, and criteria used to make the key decisions. Record rationale in documents, code comments, or other media that will transition to the support agency. Describe the meaning of "key decisions" and the approach for providing the rationale in the SDP.	4.2.6
Record information about the development of the software in appropriate SDFs and maintain them for the duration of the contract.	5.2.4
Provide general information that aids in understanding the information found in the software product, including rationale.	All DIDs Notes
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 85. Developer's key activities related to ***Rationale/key decisions***.

5.39.2 Acquirer's responsibilities and considerations. Recording rationale for key decisions is an activity designed to provide the support agency with information needed to help support personnel understand the software. During software development, key decisions are made in acquirer-developer meetings, between team members, or by individuals. Key decisions, if recorded at all, are often placed in meeting minutes, team member's notes, software products that are not specified as deliverables, or in deliverables that may not transition to the support agency, such as trade studies. To ensure that rationale regarding key decisions is transferred to the support agency, the standard provides requirements for key decisions to be recorded in documents, code comments, and other media that will transition to the support agency.

The scope and detail to be recorded in deliverables regarding the rationale for key decisions need to be agreed upon between the developer and the acquirer early in the project. Many decisions are made by the development team each day -- recording all of them would be expensive and non-value added. It is important to determine what decisions are the "key" ones. Too many recorded rationale could result in information overload and hinder, rather than help, the support agency understand the software. A "key criterion," therefore, in determining what decisions should be regarded as key, might be, "What is likely to be of interest to the support agency or might keep the support agency from having to 're-invent the wheel' if a modification is needed?"

Using this criterion might clarify, for example, that a "key decision" is the choice of a particular optimizing algorithm whose selection was based on a specific operating system version. If the support agency is to understand the impact of updating the operating system version, it needs to know the impact on the software if such a change is made. If changing the algorithm so that it will work with the new version involves changing a significant number of lines of code and retesting all of the software, the support agency would know that the changes would result in substantial cost. If costs are too high, the support agency might determine that changing the operating system is not worth the expense and risk. In this case, knowing whether or not the algorithm depended upon the particular operating system version would be "key."

Other key decisions might be those that influence the overall structure of the software; those based on user preferences rather than technical merit; or those based on cost constraints or schedule constraints. For example, a key decision might be to select one particular COTS software product rather than another because the licensing agreement allows the acquirer to make run-time copies without an additional fee. The selected COTS package might not be the "best choice" technically, but might be the "best choice" given the cost constraints for the project. Another key decision might be that a data-driven design approach was selected not because it was the "best choice" for the software being developed, but because that was what the team was familiar with and could implement the most quickly.

Comments in the code are one way of recording key decisions. The number of comments versus quality of code comments, however, is often an issue. Every line of code need not have a comment regarding a "key decision." The usefulness of the information, such as the conditions that exist when the instruction is executed, the reason an algorithm was selected (such as why one search algorithm was selected rather than another), or the rationale for selecting a particular record or field length, is more important than the sheer number of comments.

5.39.3 Additional things to think about.

Has the developer defined the criteria to be used to determine what decisions are "key"?
Are key decisions being recorded along with other information as the development proceeds rather than at the end?
Are key decisions being recorded in deliverable products as described in the SDP?
Have documents that are to be transitioned to the support agency been identified?
Do software development environment methods/procedures/tools facilitate recording key decisions?

5.39.4 Related guidebook topics.

Contract data requirements list
Data accession list
Documentation (Recording information)
Software support

5.40 Reengineering.

5.40.1 Requirements summary. MIL-STD-498 defines reengineering as "*the process of examining and altering an existing system to reconstitute it in a new form*." Reengineering encompasses one or more of the following:

- **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;
- **Translation:** transforming source code from one language to another or from one version of a language to another.

References to ***Reengineering*** in MIL-STD-498 are listed in Figure 86, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 4, 1.2.1, 1.2.4.3, 3.29, 3.33, G.5, Figure 12*
SDD	4.1.d
SSDD	4.1.d

FIGURE 86. MIL-STD-498's references to ***Reengineering***.

A developer's only mandatory requirement is to identify components whose development status/type is "to be reengineered." The other MIL-STD-498 references are guidance material to emphasize that MIL-STD-498 may be applied to any software development project including reengineering projects. Software development, as used in the standard, encompasses new development, modification, reuse, reengineering, maintenance and all activities resulting in software products.

MIL-STD-498 requires that a developer tasked with a reengineering project describe the approach to be used for performing applicable activities in the SDP for the project and develop and record the necessary information about the system/software. Figure 12 of the standard shows possible ways of applying MIL-STD-498 to a reengineering project.

When software is reused or reengineered, the developer is to identify that software and its development status/type when recording the design. (See this guidebook's topic, *Reusable software products*, for more information.)

5.40.2 Acquirer's responsibilities and considerations. Reengineering of systems and software, like business process reengineering or business reengineering, is often a fundamental rethinking and redesign of something to achieve improvements in cost, quality, service, and speed, or to transform existing software so it can be understood, controlled, and reused.

Reengineering is one method a DoD service or government agency may select to update a needed capability, add new capabilities to existing capabilities, migrate to a new platform or processor, or otherwise revise, renew, and restore capabilities. To make decisions on whether to reengineer or start anew, the acquirer may need to perform trade-off studies to understand the existing system, its operational concept and mission, and any embedded rules or capabilities.

Reengineering and reuse are closely related. Products identified for reuse (legacy systems) can be reengineered to meet higher standards, can be restructured to account for different platforms, or can be redocumented to meet differing needs for details of implementation. Reengineering takes advantage of all or part of existing products.

Software reengineering can aid in efforts to improve performance or operations by identifying undocumented rules or capabilities embedded in a system(s) that may need to be revised. Conversely, operational concepts or rules of operation may need to be revised (sometimes referred to as Business Process Reengineering (BPR)) when underlying concepts or rules are ineffective.

Prior to making a decision whether or not to reengineer, the acquirer should have a clear understanding of the expected benefits of reengineering a system and quantifiable measurements for determining whether these benefits have been achieved. When multiple systems are considered for reengineering, the acquirer might want to develop a prioritized list of those systems along with system names, platforms on which they run, associated operating systems, vendor name, computer languages, data associated with those systems (such as reports input to and/or output by the system), and databases the product supports. This list will help ensure that all processes, products, and data are considered in the trade-off studies. Other considerations for making a reengineering decision include:

- Effectiveness of current systems
- Criticality of the systems
- Objectives of the organization(s) requiring the system
- Potential developer's reengineering capabilities
- Capabilities of the recommended reengineering tools in relationship to the target environment

The acquirer may want to designate a reengineering team to coordinate the activities, such as training and planning for cut-over to the new system, needed by the users to ensure a transition to the reengineered processes or systems.

5.40.3 Additional things to think about.

What is the developer's rationale for recommending reengineering of the system? Was business process redesign considered? Was the feasibility of redeveloping the system analyzed as an alternative strategy?
Is the scope of the reengineering activity clearly defined?
Will the developer be able to acquire or use all the reengineering tools required to perform the reengineering?
What training will be required for users? What training will be required for support of the reengineered system within the support organization?
Should system(s) be analyzed with analysis methods and tools prior to reengineering to gauge the potential success of the reengineering effort?
Can reengineering activities be performed in an incremental manner? Can reengineering start with lower risk elements?
How will the reengineered system be moved from the reengineering environment to the target environment?
How will the reengineered system be tested to ensure all requirements were captured?
Are there any COTS software products in the system(s) that could cause a legal issue if reengineered?
Does the developer have the environment needed to use selected reengineering tools?
Will the reengineered system still interact successfully with other systems and their respective environments?
What will be used as a repository for the reengineered source code, documentation and/or design information?
How will the reengineered system be documented?
Is reengineering of the business processes required prior to reengineering the system(s)?
What are the targeted performance measurements related to processing time and expected cost savings?
How will the existing system be measured to establish baseline performance and how will the reengineered system be measured to gauge results?
What software will be required to support the execution and subsequent maintenance of the reengineered software?

5.40.4 Related guidebook topics.

Commercial-off-the-shelf software products **Requirements**
Documentation (Recording information) **Reusable software products**
Joint technical and management reviews

5.41 Requirements.

5.41.1 Requirements summary. MIL-STD-498 provides requirements regarding requirements of a system, subsystem or CSCI. The standard defines "requirements" as: (1) a characteristic that a system or CSCI must possess to be acceptable to the acquirer, discussed in this topic, and (2) a mandatory statement in the standard or another portion of the contract, which is discussed in this guidebook's topic, *Requirements of the standard*. References to ***Requirements*** in MIL-STD-498 are listed in Figure 87, with non-mandatory references indicated with italicized text. (Requirements for a system or CSCI are referenced throughout the standard and its DIDs. Only key references are listed below.)

MIL-STD-498	*3.30*, 5.3, 5.5, 5.6.2, *5.7 Note, 5.9 Notes 1-2*, 5.9.3, *Figure 2*, Figure 3, Figure 4, Figure 6, *Figures 9-12*, *Figures 15-17*
DBDD	3, 6
IDD	4
IRS	3, 4, 5
OCD	3, 4, 5, 6
SDD	3, 6
SDP	5.3, 5.5
SPS	3, 6
SRS	3, 4, 5
SSDD	3, 5
SSS	3, 4, 5
STD	4.x.y.1, 5
STP	6

FIGURE 87. MIL-STD-498's references to ***Requirements***.

To cover systems containing both hardware and software (such as radar systems) and software-only systems (such as payroll systems), the standard includes systems-level activities and requires the developer to "participate in" them. For systems involving hardware and software development, "participate in" is interpreted to mean "*take part in, as described in the software development plan*." For software-only systems, "participate in" means "*be responsible for*."

MIL-STD-498 provides requirements for defining requirements, tracing requirements, evaluating, controlling and testing requirements, and transforming requirements into

executable software that meets user needs. Except for the planning documents and user manuals, the standard's software product descriptions define and record the technical information necessary to make the transformation. Figure 88 lists developer's key activities related to ***Requirements***.

Developer's key activities related to ***Requirements***	References in MIL-STD-498
Describe the approach to be followed for requirements analysis, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to requirements.	SDP 5.3, 5.5
Develop and apply standards for representing requirements. Describe or reference the standards to be followed in the SDP.	4.2.2 SDP 4.2.2
Participate in analysis of user input. Analyze user inputs such as need statements, surveys, problem/change reports, feedback on prototypes, interviews, or other user input or feedback. Participate in defining and recording the operational concept of the system, including all applicable items in the OCD DID, the requirements to be met by the system and the methods to be used to ensure requirements have been met, including all applicable items in the SSS DID.	5.3 IRS 3, 4, 5 OCD 3, 4, 5, 6 SSS 3, 4, 5
Define and record system/subsystem-wide and CSCI-wide design decisions that describe how it will behave, from a user's point of view, in meeting its requirements.	DBDD 3 SDD 3 SSDD 3
Define and record the software requirements and the methods to be used to ensure each requirement has been met.	5.5, 5.6.2, 5.9.3 IRS 3, 4 SRS 3, 4 STD 4.x.y.1
Trace between CSCI requirements and system requirements, between software units and CSCI requirements, between test cases and requirements, between source files and software units, and between computer hardware utilization measurements and CSCI measurements.	DBDD 6 IDD 4 IRS 5 SDD 6 SPS 6 SRS 5 SSDD 5 STD 5 STP 6
Assign a project-unique identifier to each requirement. State requirements in such a way that an objective test can be defined for it. Annotate each requirement with associated qualification methods.	IRS 3 SRS 3 SSS 3
Perform corrective action. Include "requirements" as one of the categories used in classifying problems in software products.	Figure 4
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 88. Developer's key activities related to ***Requirements***.

MIL-STD-498 has two activities for defining and recording requirements that a system or CSCI must possess to be acceptable to the acquirer: (1) system requirements analysis and (2) software requirements analysis. The standard has two comparable activities for demonstrating that requirements have been met: (1) system qualification testing and (2) CSCI qualification testing. (See this guidebook's topic, *Qualification testing*, for more information.)

System requirements analysis. System requirements analysis consists of: (1) analyzing user input provided by the acquirer to gain an understanding of user needs, (2) defining and recording the operational concept for the system, and (3) defining and recording the requirements to be met by the system and the methods to be used to ensure that each requirement has been met. The result of this activity includes defining and recording all applicable items in the System/Subsystem Specification (SSS) and the Operational Concept Description (OCD) DIDs. (Depending upon CLIN, SOW, or CDRL provisions, interface requirements may be included in an SSS or an Interface Requirements Specification (IRS).) System requirements are those characteristics of the system that are conditions for its acceptance.

System requirements analysis is often performed iteratively with system design activities; each iteration provides requirements and design data for additional levels until CSCIs, HWCIs, and manual operations are completely identified. While the OCD is frequently developed prior to the system requirements analysis activities, it can also be prepared in conjunction with system/subsystem requirements or after those requirements are established to provide a clear picture of the operational concept for the system.

Software requirements analysis. Software requirements analysis consists of defining and recording: (1) the software requirements to be met by each CSCI, (2) the methods to be used to assure that each requirement has been met, and (3) the traceability between the CSCI requirements and system requirements. The result of this activity includes all applicable items in the Software Requirements Specification (SRS). CSCI requirements are system requirements allocated to a particular CSCI, software capabilities required to satisfy those requirements, derived requirements resulting from system/subsystem requirements/design activities, and may include system/subsystem design that has been converted to CSCI requirements for subcontracts or other purposes. (Depending upon CLIN, SOW, or CDRL provisions, interface requirements may be included in an SRS or an IRS.)

MIL-STD-498's specifications (SSS, IRS, SRS, Software Product Specification (SPS)) contain the characteristics of the system/software that an acquirer has made conditions of acceptance, i.e., the requirements. An acquirer frequently makes such specifications a part of a contract or other agreement between the acquirer and the developer. Changes to specifications that are

part of the contract are subject to contractual modification rules and procedures. If the specifications are not on contract, the acquirer and developer will usually establish procedures for formalizing and controlling changes to the requirements. A set of agreed upon requirements is often referred to as a "baseline." Changes to requirements are reflected in updates to specifications (SSS, IRS, SRS, SPS).

If a system or CSCI is developed in multiple builds, its requirements may not be fully defined until the final build. Some requirements may be detailed, others relatively general. It is to be understood that the development activities performed and any associated reviews apply to the subset of requirements defined for the particular build, not to the entire system or CSCI.

Section 3 of both the SSS and SRS DIDs identify the following eighteen categories of requirements to be addressed in the analysis activities:

- Required states and modes
- Capabilities
- External interfaces
- Internal interfaces
- Internal data representation
- Adaptation
- Safety
- Security and privacy
- Environment
- Computer resources
- Quality factors
- Design and implementation constraints
- Personnel-related
- Training-related
- Logistics-related
- Other
- Packaging
- Precedence and criticality

5.41.2 Acquirer's responsibilities and considerations. A major acquirer responsibility is to ensure that requirements are specified that will meet user needs within cost and schedule constraints. The standard makes a clear distinction between requirements and design. MIL-STD-498 removes the need to determine the difference between what and how as the distinction. Instead, the standard separates requirements from design based on the following rule:

> A characteristic 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 is design.

When requirements are viewed as characteristics needed for acceptance, more flexibility to design the "best" solution may be provided. For example, the acquirer might have a requirement for a system interface consisting of a data communications channel capable of handling 10 megabits of data per second. If no other requirement were specified, the developer would not be constrained to a specific design or implementation, but would be free to analyze, design, and propose the "best" solution available, taking into consideration any other requirements (such as maintainability) that might constrain the design. However, the acquirer might have specified an ACME XYZ-123 interface channel to interface with an existing system, eliminating any possible "better," cheaper, faster solution. In either case, the result is a "requirement" for acceptance. In MIL-STD-498, the acquirer can specify performance requirements without specifying the design solution or provide the design solution, whichever is needed.

The segregation of requirements and design characteristics is reinforced through the content layout of the standard's software products. Characteristics that are conditions of acceptance are specified in one or more of the following: SSS, IRS, SRS, and SPS. Characteristics selected by the developer in response to requirements are included in one or more of the following: System/Subsystem Design Description (SSDD), Interface Design Description (IDD), Software Design Description (SDD), or Database Design Description (DBDD).

As part of system and software design activities, design decisions for each architectural component can impose constraints on the components' constituent components and associated characteristics. These constraints may be called "derived" requirements. The SRS and IRS DIDs explain that derived requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.

There is no requirement to trace design characteristics to qualification tests. Design decisions made during the project, even if presented at joint reviews or included in deliverable design descriptions and approved by the acquirer, remain at the discretion of the developer. They need to be covered in developer-internal testing, but are subject to change without acquirer notification or approval and need not be demonstrated during qualification testing. Figure 89 provides developer-internal design and testing activities and their relationships to acquirer requirements and acquirer witnessed qualification tests.

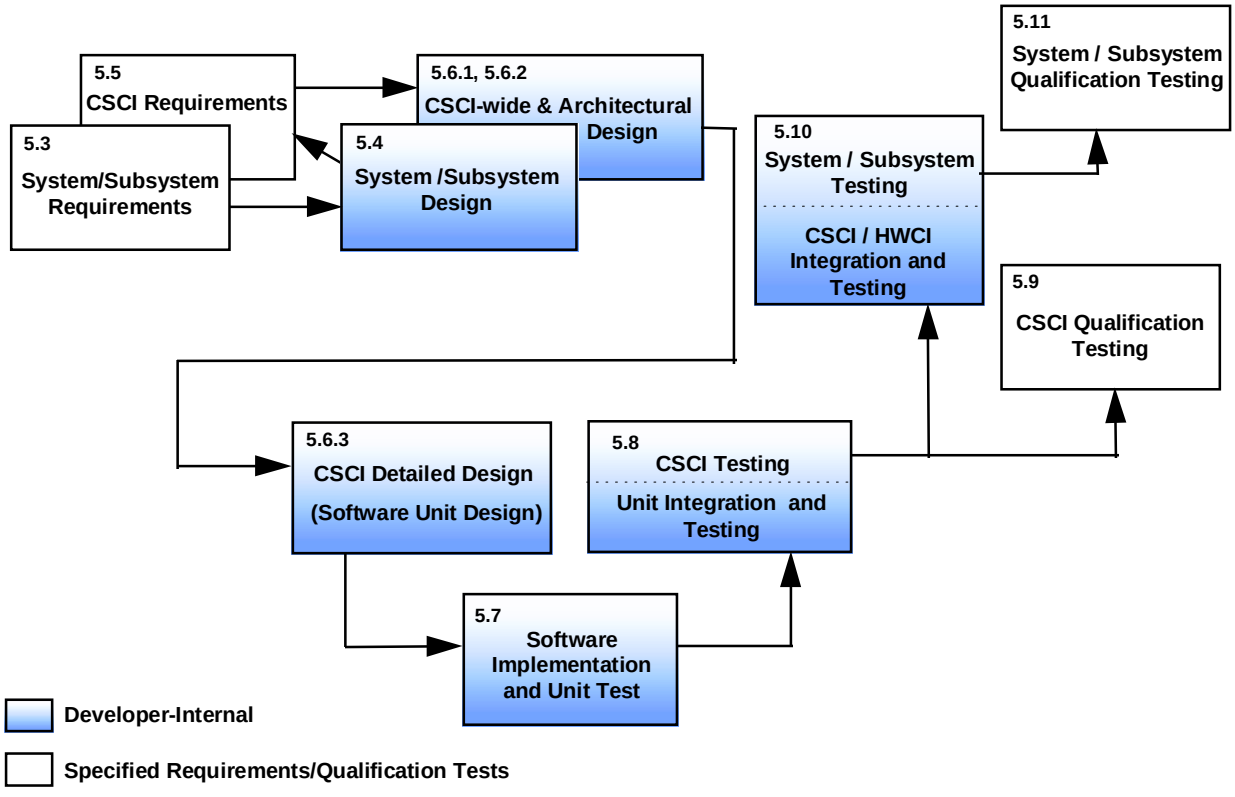


FIGURE 89. Requirements, design, and test relationships.

The SSS and SRS DIDs are similarly structured to aid in tracing requirements. Information called for in one paragraph of the SSS will be called for in the comparable paragraph in the SRS. Although this does not completely cure the one-to-many, many-to-one, and many-to-many traceability problem between system and software requirements, it can help. The developer is to provide both upward and downward traceability between CSCI requirements and system/subsystem requirements. CSCI requirements are also to be traced upward and downward between the requirements and tests in the Software Test Plan (STP) and between the CSCI requirements and test cases in the Software Test Description (STD). (Refer to this guidebook's topic, *Traceability*, for more information regarding requirements traceability.)

5.41.3 Additional things to think about.

Does the schedule allow for a thorough requirements analysis?
For systems involving numerous subsystems or multiple subcontractors, have all of the requirements been allocated? Are all requirements traceable?
Are all requirements testable? Are the requirements quantifiable? Can it be objectively determined that a requirement has been satisfied?
Do requirements impose unnecessary design constraints on the developer?
Are developer's tools used for requirements analysis compatible with the development process and environment, and if so should arrangements be made to transition them to the support agency?
Are procedures for handling changes to requirements clearly defined?
If an Evolutionary program strategy is being followed, is the effort to revise and augment requirements well defined?
When appropriate, are proven design constraints, such as architectures, included in specifications?
Is there a need (such as when multiple contractors are involved) for interfaces to be documented and delivered separately from the SSS(s) and SRS(s)?
Have all interested parties (e.g., security, support, logistics, etc.) been included in providing inputs to requirements definition (OCD, SSS, SRSs)?
Are all specified requirements really conditions of acceptance?
Have key decisions regarding requirements been recorded?
Is there a need for more than one developer to independently define requirements?

5.41.4 Related guidebook topics.

Acceptance by the acquirer
Critical requirements
Databases
Interfaces
Joint technical and management reviews
Operational concept
Qualification testing
Rationale/key decisions
Requirements of the standard
Safety
Security and privacy
System/subsystem
Testing (Developer-internal)
Traceability

5.42 Requirements of the standard.

5.42.1 Requirements summary. MIL-STD-498 provides requirements regarding software development and documentation. The standard defines "requirements" as: (1) a characteristic that a system or CSCI must possess to be acceptable to the acquirer, which is discussed in this guidebook's topic *Requirements*, and (2) a mandatory statement in the standard or another portion of the contract, which is discussed in this topic.

MIL-STD-498 itself is the second type of requirement. Together with its twenty-two DIDs, MIL-STD-498 contains a comprehensive set of requirements for software development and documentation. The standard provides general requirements in section 4, such as "use systematic, documented methods; standardize representations for information; and reuse applicable existing software products." The standard provides detailed requirements in section 5. The detailed requirements are organized into major software development activities supported by mandatory Appendixes B, C, and D. The appendixes provide requirements pertaining to incorporating reusable software products, classifications for problem reporting, and software product evaluations, respectively. The detailed requirements activities reference DIDs containing the information applicable to the project generated as a result of performing the activity.

5.42.2 Acquirer's responsibilities and considerations. MIL-STD-498's requirements are invoked by citing the standard on a contract. (See this guidebook's topics, *Contract* and *Statement of work*, for more details.) The standard's requirements and its DIDs are intended to be tailored. Tailoring is the responsibility of the acquirer, but tailoring suggestions may be proposed by developers. Tailoring guidelines are provided in the standard (Section 6 and appendixes G and H) and in the MIL-STD-498 Overview and Tailoring Guidebook.

MIL-STD-498 contains requirements whose specifics must be provided in the contract. These requirements, known as "shell requirements," impose no requirement until details are provided in the contract. Shell requirements serve to (1) make the standard self-tailoring, (2) provide a framework for incorporating project-specific requirements in the contract, and (3) remind the acquirer to consider the issue raised. Examples of shell requirements are:

- Comply with the data rights in the contract for reused software products
- Develop a plan for performing software installation and training at user sites specified in the contract
- Interface with IV&V agent(s) specified in the contract

A complete list of shell requirements is presented in Figure 27 or this guidebook's topic, *Contract*.

5.42.3 Additional things to think about.

Have the requirements of the standard been tailored to meet project needs?
Have all parties concerned in using, supporting, and acquiring the system provided input to tailoring the standard?

5.42.4 Related guidebook topics.

Acceptance by the acquirer
Contract
Requirements
Reusable software products
Statement of work
Subcontractor management

5.43 Reusable software products.

5.43.1 Requirements summary. MIL-STD-498 provides requirements for using reusable software products. The standard defines a reusable software product as "*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*." Software development is used as an inclusive term encompassing new development, modifications, reuse, reengineering, maintenance, and all other activities resulting in software products. Both developing for reuse and using reusable software products fall under the development requirements of the standard. References to ***Reusable software products*** in MIL-STD-498 are listed in Figure 90, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 3-4, 1.2.1, 1.2.4.3, 3.12, 3.31, 3.33,* 4.2.3.1, 4.2.3.2, B.3, Figure 3, *Figures 9-12*
All DIDs	*10.1.i*
SDD	4.1.d
SDP	4.2.3.1, 4.2.3.2
SSDD	4.1.d

FIGURE 90. MIL-STD-498's references to ***Reusable software products***.

MIL-STD-498 provides requirements for the developer to:

- Specify in the SDP the criteria to be used for evaluating reusable software products for use in fulfilling the requirements of the contract
- Identify, evaluate, and reuse software products that meet the evaluation criteria
- Ensure that incorporated software products meet the data rights requirements in the contract
- Identify opportunities for developing software products for reuse and evaluate the benefits and costs of these opportunities
- Inform the acquirer of reuse opportunities that provide cost benefits and are compatible with program objectives

Appendix B of the standard interprets MIL-STD-498 when the standard is applied to the incorporation of reusable software products. Each of the standard's activities is interpreted for reusable software products based on:

- Whether or not the reuse candidate has a record of good performance

- Whether or not the reuse candidate is proposed to be used as a complete CSCI or a software unit (part of a CSCI)

- Whether or not the reuse candidate is to be used "as is" or modified

For reusable system/software components that are identified, evaluated, and selected for incorporation into the system/software, identifying information such as name, version, documentation references, and location where the reusable software is stored is to be provided. Figure 91 lists developer's key activities related to ***Reusable software products***.

Developer's key activities related to ***Reusable software products***	References in MIL-STD-498
Describe the approach to be followed for identifying and evaluating reusable software products and/or developing software for reuse, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to reusable software products.	SDP 4.2.3
Identify and evaluate reusable software products for use in fulfilling contract requirements. Identify opportunities for developing software products for reuse if justified by cost/benefit evaluations.	4.2.3, Appendix B
Identify and describe reusable software units and components as well as software to be developed for reuse.	SDD 4.1.d SSDD 4.1.d

FIGURE 91. Developer's key activities related to ***Reusable software products***.

5.43.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for the use of reusable software whenever practical, that is, whenever reusable software products meet system and software requirements and are cost-effective over the life of the system. Examples of reusable software include:

- COTS software products. COTS software is existing software that is commercially available and intended for the marketplace.

- Acquirer-furnished software. Acquirer-furnished software is existing software provided by the acquirer.

- Reuse library software. Reuse library software is existing software in reuse libraries that has been provided by the acquirer, developer, or a third party (vendor).

- Pre-existing developer software. Pre-existing developer software is existing software developed by the developer for one project but used on another.

Other terms sometimes used as synonyms for reusable software are non-developmental item (NDI) and non-developmental software (NDS).

Reusable software products

The term "reusable software product" can be applied to any software product that is needed for a project, not only to executable software. Reuse of a software product may include all or part of the software product and may involve its modification. The following illustrates how system/subsystem-wide and CSCI-wide, architectural, and detailed design may be revised and reused:

- System/subsystem-wide and CSCI-wide design. (The user's perspective of how a system, subsystem, or CSCI behaves.) The behavioral design of an application may be reused when the application is reengineered to run in a distributed computing environment.
- Architectural design. (The representations of the system, subsystem, or software components depicting interfaces and concept of execution.) The software architecture for a simulation model may be reused when the model is migrated to a different host computer.
- Detailed design. (The description of how each component works internally.) The detailed design of an inventory system may be reused as part of an upgrade that includes new capabilities.

Other examples of reusable software products are: software development plans, requirements, data element descriptions, test plans, and test data.

MIL-STD-498 requires that the developer's criteria for selecting software products for reuse include the product's ability to meet the specified requirements and be cost-effective over the life of the system. The standard lists, as examples of additional candidate criteria that may be used, the following:

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

The developer is required to describe the criteria to be used for evaluating reusable software products in the SDP for the project. These criteria are intended not as inhibitors to software reuse, but as protection for the acquirer. For example, some acquirers may not be aware of the fact that much of today's reusable software is not self-contained, but relies on auxiliary or support software. A potential problem with reusable software of this type is illustrated by application program interface (API) software. Often, the design of APIs requires other software libraries (such as dynamic link libraries (DLLs)) for support. If these libraries are upgraded or modified in any way, old applications may not work with new applications installed using the newer upgraded libraries.

MIL-STD-498 provides requirements for the developer to state the scope of the search to be made for reusable software in the SDP for the project. The acquirer may also want to indicate in the contract specific vendors or products to be included in the search. In addition, the acquirer may want the developer to develop software for reuse. If so, the acquirer may require the developer to search for opportunities for reuse across systems that have requirements in the same domain. While development costs for software intended for reuse may be higher than for software intended for a single application, the cost savings across multiple systems can be significant.

5.43.3 Additional things to think about.

Is there a software repository available of reusable government-furnished software? Does the developer have access to it?
Are risks associated with reusable software products clearly defined?
How are software changes to the reusable components determined across multiple projects?
Who is responsible for maintaining software developed for reuse?
What is the developer's process for incorporating reusable software, including fallback provisions if the reusable software doesn't work as planned?
How many buyers of the COTS packages are there? Are there enough users to warrant continued life cycle support by the supplier? Have the risks of using a COTS product that was developed for a limited market been assessed?
When should the developer evaluate opportunities for reuse development? At the beginning of the project? As an ongoing effort with each build? During all activities?
How can the acquirer determine a dollar value for identifying candidates for reuse?

5.43.4 Related guidebook topics.

Commercial-off-the-shelf software products
Data rights
Documentation (Recording information)
Licenses (Software)
Reengineering

5.44 Risk management.

5.44.1 Requirements summary. MIL-STD-498 provides requirements for the developer to perform risk management. The developer is to identify applicable risks/uncertainties for each activity and product for the project and develop plans for dealing with them. The standard's risk management requirements are based on DoDI 5000.2. DoDI 5000.2 requires the acquirer to assess project risk throughout the acquisition of a system. MIL-STD-498's risk management requirements support this policy by requiring the developer to address the software's technical, cost, and schedule risks and surface those risks for discussion in joint reviews. References to ***Risk management*** in MIL-STD-498 are listed in Figure 92, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.18.1, 5.18.2, 5.19.1, B.3, Figure 5, *G.4, Figures 8-12*
SDP	4, 5, 5.19.1

FIGURE 92. MIL-STD-498's references to ***Risk management***.

The standard requires the developer to perform on-going risk management to:

- Identify, analyze, and prioritize the areas of the software development project that involve potential technical, cost, or schedule risks
- Develop strategies for managing those risks
- Record the risks and strategies in the SDP
- Implement the strategies in accordance with the plan

The intent of MIL-STD-498's risk management requirements is to identify and address risk conditions before they become problems. Figure 93 lists developer's key activities related to ***Risk management***.

Other activities covered by the standard that support risk management are:

- Evaluations of software products using the criteria in Appendix D
- Management indicators such as those described in Appendix F
- Quality assurance evaluations of processes
- Problem reporting, analysis, tracking and handling
- Project assessments of the processes used

Developer's key activities related to ***Risk management***	References in MIL-STD-498
Describe the approach to be followed for risk management, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to risk management.	SDP 5.19.1
Identify, analyze, and prioritize areas that involve potential technical, cost, or schedule risks; develop strategies for managing those risks; record risks and strategies in the SDP; and implement the strategies in accordance with the plan.	5.19.1
Identify applicable risks/uncertainties and plans for dealing with them for each topic addressed in the SDP	SDP 4, 5
Consider as a criterion for evaluating reusable software products the technical, cost, and schedule risks and trade offs in using the software product.	B.3
Classify problems as priority 2 problems if they adversely affect technical, cost, or schedule risk to the project. [See Figure 5 in Appendix C of the standard]	Figure 5
During joint technical reviews: • Surface near- and long-term risks regarding technical, cost, and schedule issues • Agree upon mitigation strategies for identified risks, within the authority of those present • Identify risks and issues that are to be raised at joint management reviews	5.18.1
During joint management reviews: • Inform management of project status with respect to risk management issues • Agree upon mitigation strategies for near- and long-term risks that could not be resolved at joint technical reviews • Identify and resolve management-level issues and risks that were not raised at joint technical reviews	5.18.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 93. Developer's key activities related to ***Risk management***.

5.44.2 Acquirer's responsibilities and considerations. Software typically presents a significant source of project risk. Risk management can help reduce or control individual drivers of risk, and often includes some combination of avoidance, control, prevention, assumption (accept the risk), and transfer (from one phase to another).

The developer's risk management activities required by MIL-STD-498 are one element of the acquirer's overall risk management program. The DoD's risk management program, as described in DoDI 5000.2, part 5, section B, consists of planning, identification, assessment, analysis, and reduction techniques, including:

- A structured and documented risk assessment and analysis process with user participation to identify risks early in the program and to provide proactive, look-ahead assessment and review

- Clearly defined criteria for elements leading to the risk assessment events

- Assessment of the contractor's managerial, development, and manufacturing capabilities and processes

DoD services or government agencies may have further policies regarding risk management requirements to assist the evaluation of program risk. For example, there may be DoD service or government agency policies specifying:

- Risk criteria/drivers to be periodically assessed

- Specific contents to be included in risk management reports

- Joint acquirer-developer risk assessment teams to be formed

- Software management indicators to be used for risk management

Other activities the acquirer might consider to reduce the technical, cost, and schedule risks for a software development project include:

- Implement **risk element tracking**. Tracking of highest risk elements increases awareness of program risks. It supports risk reduction or avoidance by determining whether or not risks are being dealt with. (This guidebook's topic, *Joint technical and management reviews*, provides additional information.)

- Rely on **multiple cost and schedule estimates**. The use of multiple, independent sources of estimates can reduce cost and schedule risk. Discrepancies between various estimates may indicate incorrect assumptions or errors made during the estimate.

- Implement **cost/schedule/performance tracking**. Tracking supports risk mitigation and reduction by early identification of cost overruns, schedule slippage, and performance shortfalls. Tracking allows the developer and acquirer to address cost, schedule, and performance problems early, thus reducing overall program risk. (This guidebook's topics, *Joint technical and management reviews* and *Schedules*, provide additional information.)

- Award development contracts to **competing contractors** when time and funding permit. Multiple awards to competing contractors, one or more who will be selected for production, can reduce the risk associated with relying on a single developer. A design competition between contractor teams can result in definition and resolution of high risk items during design. Sustaining the competition into the production phase allows the acquirer the ability to award the follow-on production contract to the contractor providing the most cost-effective production capability.

- Structure contracts with **performance incentives and award fees**. Performance incentive and award fee contracts can provide motivation for improved contractor performance, reducing cost and schedule risk for the system. However, this strategy requires spending time and money to prepare for regularly scheduled presentations to the award fee board to substantiate fee awards. These resources could otherwise have been applied to the engineering effort.

- Perform **software development capability evaluations**. Identify risks associated with candidate software developers by assessing how consistently the developer's processes/practices are used and how mature those processes/practices are.

- Select an appropriate **program strategy** for the project. MIL-STD-498 provides options for various program strategies that can affect risk. The Incremental strategy, also called Pre-planned Product Improvement in DoDI 5000.2, and the Evolutionary strategy both allow the acquirer to spread risk by developing the system in a series of builds. The Evolutionary strategy differs from the Incremental strategy by acknowledging that requirements are not fully understood up-front, are further defined with each build, and can reduce the risk of building a system that users don't want, can't use, or meets no known needs. In any strategy, prototyping can involve users early and serves as a vehicle for better understanding user requirements to reduce performance risks. Appendix G of MIL-STD-498 provides additional information on program strategies.

- Use a **design-to-cost** development approach if the project's performance requirements are flexible. Design-to-cost reduces risk by constraining the developer to develop a system within a fixed cost constraint, at the expense of performance, if necessary.

- Perform **make-or-buy** tradeoff studies to determine if COTS software solutions offer a lower risk approach than custom software solutions. Buying COTS software can reduce cost and schedule risk but can increase performance risk if COTS software cannot be modified. (This guidebook's topics, *Commercial-off-the-shelf software products*, *Data rights*, and *Licenses (Software)*, provide additional information.)

- Evaluate opportunities for **reuse and reengineering**. Cost, schedule and performance risks can be reduced by incorporating modified, unmodified, or reengineered reusable software products into a system. (This guidebook's topics, *Reengineering* and *Reusable software products*, provide additional information.)

- Define the **software support concept** early in the project to reduce the risks that the software might not be supportable. Optimally, the acquirer should determine the support concept before program initiation. Early support agency involvement can reduce cost, schedule and supportability risk by providing for the delivery of necessary elements of the software development environment (SDE) and necessary documentation that the support agency needs. (This guidebook's topic, *Software support*, provides additional information.)

- Require the use of an **open system architecture**. An open system architecture can reduce cost risk by increasing interoperability, portability, and information sharing across platforms, thus reducing future software procurement costs when platforms or operating systems change.

- Acquire all **data rights** needed to sufficiently maintain and modify the system. Obtaining the data rights reduces the supportability risk raised by relying on a single developer by preserving the option for the acquirer's maintenance of the software. (This guidebook's topic, *Data rights*, provides additional information and trade-offs.)

5.44.3 Additional things to think about.

Does the developer have the capability and capacity to implement a sufficiently mature risk management process? Is the developer following his defined process?
Are risks surfaced early enough? Is the acquirer made aware of them?
Is the developer using acquirer inputs to help identify operational and support risks?
Is the developer using inputs from the user, support, IV&V, certification, and other agencies to identify risks?
Has the developer implemented a risk management process that helps define risk mitigation strategies and courses of action, and tracks risk items to resolution?
Are the acquirer and developer coordinating on risk management?
Can a team environment be created in which the developer is willing to surface risks early without causing the acquirer undue anxiety?

5.44.4 Related guidebook topics.

Commercial-off-the-shelf software products
Data rights
Joint technical and management reviews
Licenses (Software)
Oversight
Reengineering
Reusable software products
Schedules
Security and privacy
Software support

5.45 Safety.

5.45.1 Requirements summary. MIL-STD-498 provides requirements for the developer to identify as safety-critical those CSCIs or portions thereof whose failure could lead to a hazardous system state (one that could result in unintended death, injury, loss of property, or environmental harm). The term "failure" includes a failure to perform correctly, out of specification performance, and may include intended operations where performance is incorrect in the system context. For such software, the developer is required to develop a safety assurance strategy, including both tests and analyses, to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for hazardous conditions. The safety assurance strategy includes a software safety program, which is to be integrated with the system safety program if one exists. The developer is required to record the strategy in the SDP for the project, implement the strategy, and produce evidence, as part of the required software products, that the safety strategy has been carried out. MIL-STD-498 lists safety as one of the items that may be addressed by a "critical requirement review." References to ***Safety*** in MIL-STD-498 are listed in Figure 94, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.4.1, *Figure 2,* B.3, Figure 5, *E.4.11*	SIOM	4, 6
COM	3	SIP	4.x.5, 5.x.2
DBDD	3, 5.x	SRS	3.3.x, 3.7, 3.18
FSM	3.x.6	SSDD	3, 4.3.x,
IDD	3.x	SSS	3.3.x, 3.7, 3.18
IRS	3.x, 3.y	STD	3, 4
OCD	3.3, 5.3	STP	4.2.x.y
SCOM	4, 5	STrP	3.1
SDD	3, 4.3.x	SUM	4, 5
SDP	4.2.4.1	SVD	3.6

FIGURE 94. MIL-STD-498's references to ***Safety***.

MIL-STD-498 provides requirements for assigning a unique identifier to each system/subsystem/CSCI requirement and for tracking those requirements. The unique identifier is intended to support safety strategies by providing a method for identifying requirements concerned with safety. The standard's requirement to prioritize/weight each requirement, if needed, can also be used in support of the safety strategy for the project. (See

this guidebook's topics, *Critical requirements, Problem category and priority classifications,* and *Traceability,* for more information.)

Figure 95 lists developer's key activities related to ***Safety***. These fall into two general categories: safety requirements relating to the specification, design, and implementation of the software itself; and safety requirements relating to the usage and installation of the software and the system, including warnings and cautions.

Developer's key activities related to ***Safety***	References in MIL-STD-498
Describe the approach to be followed for safety identifying risks/uncertainties and plans for dealing with them. Describe the approach for developing a software safety program and integrating it with the overall system safety program, if one exists. Cover all contractual clauses pertaining to safety.	SDP 4.2.4.1
Identify as safety-critical CSCIs those CSCIs, or portions thereof, whose failure could lead to a hazardous system state; develop a safety assurance strategy, including both tests and analyses, and record it in the SDP; implement the strategy; and produce evidence that the strategy has been carried out. Develop a software safety program and integrate it with the overall system safety program, if one exists.	4.2.4.1
Resolve open issues regarding safety at critical requirement reviews, if held.	E.4.11
Describe the current system or situation and the new or modified system, including provisions for safety.	OCD 3.3, 5.3
Assign priority 1 to each problem in software or activities if it jeopardizes safety.	Figure 5
Describe safety considerations that are part of the characteristics of communication methods that the system, CSCI, or interfacing entity(ies) will use for the interface.	DBDD 5.x IDD 3.x IRS 3.x SDD 4.3.x SRS 3.3.x SSDD 4.3.x SSS 3.3.x
Specify system and CSCI requirements concerned with preventing or minimizing unintended hazards to personnel, property, and the physical environment. Include system and CSCI requirements for nuclear components, if any, including requirements for component design, prevention of inadvertent detonation, and compliance with nuclear safety rules.	SRS 3.7 SSS 3.7

FIGURE 95. Developer's key activities related to ***Safety***.

Developer's key activities related to ***Safety***	References in MIL-STD-498
Specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements.	IRS 3.y SRS 3.18 SSS 3.18
Group requirements for each critical requirement by type (e.g., safety, security, privacy, etc.) together. Place safety related design decisions that respond to safety critical requirements in separate subparagraphs.	DBDD 3 SDD 3 SSDD 3
Identify safety considerations associated with system qualification testing or CSCI qualification testing.	STP 4.2.x.y
Consider the ability to provide the required safety requirements as a criterion to be used for evaluating reusable software products.	B.3.b
Include safety precautions, marked by CAUTION or WARNING, throughout test preparation and test descriptions, in computer system operation manuals, in firmware installation and repair procedures, in procedures for installation and setup of software by the software center operator, in description of software center runs, in software user manuals, in user terminal processing procedures, in software access procedures, in the processing reference guide, and in installation procedures for both software center operations staff and software users.	COM 3 FSM 3.x.6 SCOM 4, 5 SIOM 4, 6 SIP 4.x.5, 5.x.2 STD 3, 4 SUM 4, 5
Describe the facilities needed to support the deliverable software, including building features to support safety requirements (smoke alarms, safety glass, etc.).	STrP 3.1
Describe safety precautions regarding the installation.	SVD 3.6
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 95. Developer's key activities related to ***Safety*** - *(continued)*.

5.45.2 Acquirer's responsibilities and considerations. MIL-STD-498's major activities relating to safety consist of developing a safety assurance strategy to avoid unintended death, injury, loss of property, or environmental harm; developing safety precautions, marked by CAUTION or WARNING, for installation, testing, and operation of the system; and compliance with nuclear safety if the system has a nuclear component. Prior to contractor selection and contract award, the acquirer's principal concern regarding safety may be that the selected developer has adequate experience and personnel with expertise in the area of software safety. Following contract award, the acquirer's principal concern may be to carefully monitor the developer's safety performance throughout contract performance. Other acquirer considerations may include:

- Have the appropriate regulations and standards regarding safety been included in the SOW?

- Have unusual operating conditions that can have impact on safety been explored with the users? For example, the system may need to be operated by hearing-impaired or sight-impaired personnel under emergency conditions.

- Have special circumstances or operational scenarios in which the software might be required to operate been identified by the users and conveyed to the developer? For example, electronic equipment may be required to operate in a hostile electronic warfare jamming environment; the system may need to protect personnel operating in unexpectedly harsh environmental conditions; or the system may be required to operate in a hostile enemy environment, the characteristics of which are not generally known due to security reasons.

- If COTS software is to be used, have safety considerations been addressed? If COTS software introduces safety risks, have trade-off analyses been performed to determine whether to:

  - accept the safety risk and use the COTS software as is
  - accept the expense to change the COTS software
  - develop new software that will mitigate safety risks

- Have "Lessons Learned" databases regarding safety issues been reviewed in order to avoid problems that have occurred in other programs? (Each Service has a "Lessons Learned" database that is available to Government Program Management Offices.)

- Is it worthwhile to keep a "Hazard Log" database, which records, reports, and tracks hazards until they are closed, to ensure that any residual system safety risk is properly documented and made available to the users and the support agency? For example, after incorporation of safety hazard reduction/elimination measures, residual hazard risk may still exist. If so, trade-off analyses may need to be repeated to determine if further risk reduction measures should be taken.

When safety considerations apply, a software safety review may be one type of critical requirement review held during system development to discuss safety issues, resolve those issues, or identify additional work required, such as further trade-off studies.

5.45.3 Additional things to think about.

Have all important user/operator safety requirements been identified and provided to the developer?
Are safety issues covered in key software development processes? Is the safety assurance process commensurate with safety risk?
Has safety been considered in COTS evaluation and trade-offs?
Have trade-offs between safety, reliability, security, etc., been performed?
Have the appropriate standards (e.g., MIL-STD-882) regarding software safety been tailored and included in the SOW?
Have unusual operating conditions that have a potential to impact safety been explored among all parties?
Has the developer passed software safety requirements down to the subcontractors?
Have the acquirer and the developer properly accounted for software safety in costs and schedules?
Has the acquirer been made aware of any potential increased technical risk caused by software safety requirements?
Are appropriate safety warnings and cautions included in the user, test, and installation procedures?
Does the overall system safety program include software safety requirements?
Does software safety design handle unintended capabilities the software may perform as well as intended ones?
Has residual safety risk been adequately quantified and documented at the completion of software development?
Have programming standards, conventions, guidelines, and techniques that relate to safety been specified (e.g., fill empty memory instead of leaving blank)?
Do safety-specific test cases and procedures cover requirements for safety?
Has the acquirer defined the meaning of "acceptable level of safety risk" for the developer?
Have safety personnel been included in the list of participants at specific reviews? Should they be automatically included in each review?
Have safety requirements for the development, production, and support facilities been addressed?

5.45.4 Related guidebook topics.

Commercial-off-the-shelf software products
Critical requirements
Independent verification and validation
Installation (Support environment)
Installation (User site(s))
Operational concept
Problem category and priority classifications
Qualification testing
Requirements of the standard
Statement of work
Subcontractor management
Testing (Developer-internal)
Traceability

5.46 Schedules.

5.46.1 Requirements summary. MIL-STD-498 provides requirements for the developer to establish a software development process. The software development process activities may overlap, may be scheduled iteratively, and/or may be scheduled differently for different elements of the software. Software development, qualification testing, installation, and transition schedules are found in four plans: Software Development Plan (SDP), Software Installation Plan (SIP), Software Test Plan (STP), and Software Transition Plan (STrP). References to ***Schedules*** in MIL-STD-498 are listed in Figure 96, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.34,* 5.1, 5.18.1, 5.18.2, 5.19.1, *6.6,* B.3.i, *Figure 5, F.3, G.6.4*
SDP	3, 5.1, 5.18.1, 5.18.2, 5.19.1, 6, 7.2
SIP	3.1, 4.x.1, 5.x.1
STP	5
STrP	5, 7

FIGURE 96. MIL-STD-498's references to ***Schedules***.

Software Development Plan. The SDP provides for schedule(s) identifying the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity. The SDP also contains an activity network, depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restriction on the project.

Other SDP information relating to scheduling includes the developer's staff-loading; scheduling of acquirer-furnished equipment, software, services, documentation, data, and facilities; and scheduling of any other resources. Updates to the SDP are subject to acquirer approval with the exception of developer-internal scheduling and related staffing information.

Software Installation Plan. Software installation schedules depict the activities and events required to support the software delivery to the user and possibly to a support organization. The schedules often include meetings; preparation of deliverable items; packaging, shipment, installation, and check out of operational software; and training of user personnel. The SIP contains an overall schedule of the installation process as well as site-specific detailed schedules of each installation task for software user sites. The schedule depicts the tasks in chronological order including beginning and ending dates for each task. Supporting narrative is to be provided with the schedules, as necessary.

Software Test Plan. The STP requires (1) an overall testing schedule listing or depicting the sites at which testing will be scheduled and the time frames during which the testing will be conducted and (2) a chronological schedule (for each test site) depicting on-site test periods and periods assigned to major portions of the testing; pre-test on-site setup period; schedules for collection of database/data file values, input values, and other operational data needed for testing; test conduct and retest periods; and test results preparation, review, and approval of the STR.

Software Transition Plan. Software transition schedules depict the activities and events required to support the software delivery and transition to a support organization. These activities and events often include meetings; preparation of deliverable items; packaging, shipment, installation, and check out of the software support environment; packaging, shipment, installation, and check out of operational software; and training of support personnel.

Figure 97 lists developer's key activities related to ***Schedules***.

Developer's key activities related to ***Schedules***	References in MIL-STD-498
Describe the approach to be followed for schedules, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to schedules.	SDP 5.1, 5.19.1, 6, 7.2
Prepare schedules as part of project planning for software development, CSCI qualification testing, system qualification testing, software installation, and software transition. Schedule developer's management reviews of the software development process at periodic intervals. Obtain acquirer approvals for updates to plans with the exception of developer-internal schedules and related staffing information.	5.1
Classify software problems according to whether or not they adversely impact schedule risk to the project or to life cycle support of the system.	Figure 5
Propose dates for joint technical and management reviews. Obtain acquirer approval for these dates.	5.18
Perform risk management throughout the software development process to identify, analyze, and prioritize the areas of the software development project that involve potential schedule risk.	5.19.1

FIGURE 97. Developer's key activities related to ***Schedules***.

Developer's key activities related to ***Schedules***	References in MIL-STD-498
Use and report software management indicators to aid in managing the software development process. Consider management indicators based on meeting planned and actual schedules, such as: requirements volatility over time; software size, planned and actual over time; software progress, planned and actual over time; problem/change report status; computer hardware resource utilization, planned and actual over time; milestone performance, planned and actual dates.	5.19.2, F.3
Develop and record installation schedules.	SIP 3.1, 4.x.1, 5.x.1
Develop and record schedules for conducting qualification testing.	STP 5
Develop and record schedules for training support personnel and for transition activities.	STrP 5, 7
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 97. Developer's key activities related to ***Schedules*** - *(continued)*.

5.46.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for developing plans and estimated schedules for the project. The plans provide the basis for both the developer and acquirer to monitor the project's progress against those plans and estimated schedules. Schedules can provide a basis for joint review discussions and risk management activities. If the acquirer needs information to support major milestone decisions, the acquirer's review of the SDP should determine whether the planned software development approach will provide the information when needed.

The acquirer is to approve developer schedules and updates to those schedules with the exception of developer internal schedules and related staffing information. When considering the proposed dates for joint technical reviews, the acquirer and developer should keep in mind system review schedules, IPT schedules, and other system or software working group schedules (such as Interface Control Working Group (ICWG) and Computer Resources Working Group (CRWG)), and coordinate those schedules, if possible. Coordination of these schedules can help provide timely transfer of needed information.

Many planning, tracking and analysis tools exist to help plan and track schedules. These include:

- Multi-tiered tools that integrate schedules
- Critical path analysis tools

- Project estimating models that use project software activity schedules with respect to benchmarks and that support cost versus schedule trade-offs
- Software management indicator tools that display software schedule parameters
- Interactive and other software process models that capture historical data to help managers perform schedule estimation on new projects

5.46.3 Additional things to think about.

Is the developer's flexibility in laying out the software development process and schedule reduced by rigidly scheduling deliverables in the contract/CLIN/CDRL?
Do the schedules in the plans mesh with one another?
Is there a system-level development schedule? Are software schedules synchronized with system-level schedules?
Do software review schedules support system-level and milestone reviews?
Are there any conflicts between schedule items (such as events, resources) on the various schedules?
Do the schedules accurately reflect tailoring of MIL-STD-498? Do schedules accurately reflect the selected acquisition strategy?
Do the schedules accurately reflect the progress of the project?
Are appropriate subcontractor, IV&V, and associate developer activities included in the project's schedules?
Are schedule risk drivers monitored?
Are schedules validated with models for cost/schedule?
Is the project using any schedule-related software management indicators (such as software progress or milestone performance)?
Have indicators been clearly defined to track project progress and provide early identification of project progress?
Are long lead-time items and relationships shown on the schedule(s)?
Do plans include schedules for action item closeout, process improvement activities, process and product evaluations, scheduling and prioritization?
Does the developer have sufficient personnel resources to support the schedule?

5.46.4 Related guidebook topics.

Joint technical and management reviews
Risk management
Software development planning
Software transition

5.47 Security and privacy.

5.47.1 Requirements summary. MIL-STD-498 provides requirements for the developer to identify as security-critical and/or privacy-critical those CSCIs whose failure could lead to a breach of system security and/or privacy. The developer is required to develop security assurance and/or privacy assurance strategies so that the potential for breaches of system security and/or privacy are minimized or eliminated.

Security and/or privacy strategies are described in the project's SDP. However, security and/or privacy strategies may be recorded separately and referenced from the SDP for the project, if needed, when the developer has well-defined, documented security and/or privacy procedures in place. The standard requires the developer to implement the security and/or privacy strategy(ies) applicable to the project and produce evidence that those strategies have been carried out. References to ***Security and privacy*** in MIL-STD-498 are listed in Figure 98, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.4.2, 4.2.4.3, 5.19.3, B.3.b, *Figure 2,* Figure 5, *E.4.11, Figures 9-12*	SIOM	3.2, 3.4, 3.6, 4.2.1, 4.3.1, 4.3.2
All DIDs	10.1.c, 1.3	SIP	3.7, 4.x.2
COM	3.2.4	SRS	3.3.x, 3.8, 3.18
DBDD	3, 4.x, 5.x	SSDD	3, 4.3.x
FSM	3.x.5	SSS	3.3.x, 3.8, 3.18
IDD	3.x	STD	3, 4
IRS	3.x, 3.y	STP	3.x.1-3.x.3, 4.2.x.y
OCD	3.3, 5.3	STrP	3.1-3.5
SCOM	3.2, 3.4, 3.6, 5.5.x	SUM	3.2, 3.6, 4.1.2
SDD	3, 4.3.x	SVD	3.1, 3.2, 3.6
SDP	3, 4.2.4.2, 4.2.4.3, 5.19.3, 7.2.b		

FIGURE 98. MIL-STD-498's references to ***Security and privacy***.

Figure 99 lists developer's key activities related to ***Security and privacy***.

Developer's key activities related to ***Security and privacy***	References in MIL-STD-498
Describe the approach to be followed for security and/or privacy, identifying risks/uncertainties and plans for dealing with them. Describe secure areas of developer facilities. Cover all contractual clauses pertaining to security and/or privacy.	SDP 3, 4.2.4.2, 4.2.4.3, 5.19.3, 7.2.b
Identify as security-critical and/or privacy-critical CSCIs those CSCIs, or portions thereof, whose failure could lead to a breach of security and/or privacy, develop security and/or privacy assurance strategies and record them in the SDP, implement the strategies, and produce evidence the strategies have been carried out.	4.2.4.2, 4.2.4.3
Meet security and/or privacy requirements specified in the contract and consider these throughout the software development process.	5.19.3, Figures 9-12
Use specified criteria for evaluating reusable software products such as "ability to provide required security and/or privacy."	B.3
Resolve open issues regarding security and/or privacy at critical requirement reviews, if held.	E.4.11
For each software product, describe any security and/or privacy considerations associated with its use or distribution.	All DIDs 10.1.c, 1.3
Describe the current system or situation and the new or modified system, including provisions for security and/or privacy in emergencies.	OCD 3.3, 5.3
Assign priority 1 to each problem in software or activities if it jeopardizes security and/or privacy.	Figure 5
Describe security and/or privacy constraints for required or designed characteristics of both individual data elements and data element assemblies; describe security/privacy considerations, such as encryption, user authentication, compartmentalization, and auditing.	DBDD 4.x, 5.x IDD 3.x IRS 3.x SDD 4.3.x SRS 3.3.x SSDD 4.3.x SSS 3.3.x
Specify the system and CSCI requirements concerned with maintaining security and/or privacy, and be able to trace and audit them.	SRS 3.8, 5 SSS 3.8, 5
Specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements.	IRS 3.y SRS 3.18 SSS 3.18
Group requirements for each critical requirement by type (e.g., safety, security, privacy, etc.) together. Security and/or privacy related information should be provided in separate paragraphs.	DBDD 3 SDD 3 SSDD 3

FIGURE 99. Developer's key activities related to ***Security and privacy***.

Developer's key activities related to ***Security and privacy***	References in MIL-STD-498
Identify any security and/or privacy issues associated with the software items, hardware and firmware items, and any other materials needed for testing at the test site(s) and for each test.	STP 3.x.1, 3.x.2, 3.x.3, 4.2.x.y
Consider security and/or privacy throughout test preparation/descriptions.	STD 3, 4
Include security and/or privacy considerations associated with ensuring continuity of operations in the event of emergencies.	COM 3.2.4
Describe the procedures, including security and/or privacy measures, to be used for programming and reprogramming the firmware device.	FSM 3.x.5
Provide an overview of the security and/or privacy considerations associated with the software.	SCOM 3.6 SIOM 3.6 SUM 3.6
Include security and/or privacy considerations when identifying and describing all software files that must be installed for the software to operate; the organization and operation of the software from the operator's point of view including reports and other outputs; and run management information.	SCOM 3.2, 3.4, 5.5.x
Include security and/or privacy considerations when identifying software files that the user must request to access the software described in the SIOM; in describing the conditions to be observed in preparing each type or class of input to the software; and for each type or class of output.	SIOM 3.2, 3.4, 3.6, 4.2.1, 4.3.1, 4.3.2
Include security and/or privacy considerations in identifying all software files that must be installed for the software to operate, and those pertinent to the storage and marking of output reports and other media.	SUM 3.2, 4.1.2
Present an overview of system security and/or privacy considerations. Identify security classifications for software to be installed.	SIP 3.7, 4.x.2
Describe the facilities needed to support the deliverable software, including building features to support security and/or privacy requirements (TEMPEST shielding, vaults, etc.).	STrP 3.1
Identify and describe the hardware and software documentation and any additional documentation needed to support the deliverable software, including security and/or privacy considerations/limitations. Identify security clearances needed by support personnel.	STrP 3.2, 3.3, 3.4, 3.5
List physical media (listings, tapes, disks, etc.) and associated documentation and computer files that make up the software version being released; include applicable security and/or privacy considerations.	SVD 3.1, 3.2
Describe security and/or privacy precautions regarding the installation.	SVD 3.6
Meet general requirements and perform integral processes of the standard.	4, 5.14, 5.19, Appendix B-G

FIGURE 99. Developer's key activities related to ***Security and privacy*** - *(continued)*.

5.47.2 Acquirer's responsibilities and considerations. Security requirements protect information that, if disclosed, would reveal performance capabilities/limitations to the detriment of the *nation*. Privacy requirements protect information that, if disclosed, would result in embarrassment, harm, unfairness, or inconvenience to the *individual.*

For many systems, broad general security and/or privacy requirements can be expected to flow down to the acquirer and the developer from DoD and other government sources via the contract. Potential sources of security and/or privacy requirements are the President, Congress, National Security Council (NSC), National Security Agency (NSA), CIA, DoD, and possibly others.

Many factors may need to be considered by the acquirer or his agent in specifying security and/or privacy requirements, including:

- COMSEC
- Computer Security Act of 1987
- configuration management
- control of compromising emanations
- covert channel analysis
- department policy requirements (e.g., C2 level minimum)
- discretionary/mandatory access control
- discretionary/mandatory security
- eavesdropping, authentication, masquerading using wireless communication
- encryption
- extended background investigations
- "hacker" penetration/disruption
- life cycle/operational assurance
- marking
- multilevel security
- need-to-know
- object reuse
- "Orange Book"
- overzealous monitoring of personnel
- Privacy Act of 1974
- protection of personal health/financial/other sensitive information
- secure development environment
- secure operating system
- secure operational environment
- security clearances
- security engineering assessments
- Sensitive Compartmented Information Facilities (SCIFs)
- System Threat Assessment Report (STAR)
- system security policy
- system accreditation
- TEMPEST
- trusted facility
- trusted software
- unification/referencing of government databases
- vaults
- verifiable digital signatures

MIL-STD-498, its appendixes, and associated DIDs define security and privacy activities that the developer must perform in response to specified requirements. MIL-STD-498 provides requirements for system-associated activities for security (see 4.2.4.2) and privacy (see 4.2.4.3). MIL-STD-498 paragraph 5.19.3 provides requirements for security and privacy that affect the software development effort, the resulting software products, or both.

5.47.3 Additional things to think about.

Have all applicable security and/or privacy requirements been identified and specified? Has system security risk been assessed?
Has the developer passed down security and/or privacy requirements, if applicable, to subcontractor(s)?
Are security and privacy personnel represented at applicable joint reviews concerning these topics?
Have the acquirer and developer accounted for security and/or privacy in costs and schedules?
Does the developer have sufficient facilities and personnel with required security clearances to support the security and/or privacy strategy?
Has the acquirer made provisions for proper and timely access by the developer to facilities, data, government furnished materials/equipment, etc., needed by the project, but covered by acquirer security and/or privacy regulations?

5.47.4 Related guidebook topics.

Architectural design
Critical requirements
Independent verification and validation
Software development environment
System/subsystem

5.48 Software configuration management.

5.48.1 Requirements summary. MIL-STD-498 provides requirements regarding software configuration management (SCM). SCM is considered one of the integral processes that support the other activities in the standard. The developer's approach, described in the project's SDP, is to address all applicable contract clauses for SCM including:

- Configuration identification
- Configuration control
- Configuration status accounting
- Configuration audits
- Packaging, storage, handling, and delivery

References to ***Software configuration management*** in MIL-STD-498 are listed in Figure 100, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.12*, 4.1, *Figure 1, 5.1.1 Note 3*, 5.14, *Figure 2*, Figure 3, *Figures 9-12*
SDP	5.14
SVD	3

FIGURE 100. MIL-STD-498's references to ***Software configuration management***.

SCM activities apply to all software products prepared, modified, and/or used to develop software products as well as to the products under development, modification, reengineering, or reuse. If a system/subsystem or CSCI is developed in multiple builds, SCM in each build is to be understood to take place in the context of the software products and controls in place at the start of the build. Figure 101 lists developer's key activities related to ***Software configuration management***.

Developer's key activities related to ***Software configuration management***	References in MIL-STD-498
Describe the approach to be followed for software configuration management, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software configuration management.	SDP 5.14
Participate in selecting CSCIs during system (architectural) design. Identify entities to be placed under configuration control. Assign a project-unique identifier to each CSCI and each additional entity to be placed under configuration control, including software products to be developed or used and the elements of the software development environment. Use an identification scheme that identifies entities at the level of control and include version/revision/release status.	5.14.1
Establish and implement procedures designating levels of control each identified entity must pass through, the persons or groups with authority to authorize changes and to make changes at each level, and the steps to be followed to request authorization for changes, process change requests, track changes, distribute changes, and maintain past versions. Propose to the acquirer, in accordance with contractually established forms and procedures, changes that affect an entity already under acquirer control.	5.14.2
Prepare and maintain records of configuration status of all entities that have been placed under project-level or higher configuration control. Maintain configuration status records for the life of the contract. Include, as applicable, version/revision/release, changes since being placed under project-level or higher configuration control, and status of associated problem/change reports.	5.14.3
Support acquirer-conducted configuration audits as specified in the contract.	5.14.4
Establish and implement procedures for packaging, storage, handling, and delivery of deliverable software products. Maintain master copies of delivered software products for the duration of the contract.	5.14.5
Prepare a version description for the system.	SVD 3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 101. Developer's key activities related to ***Software configuration management***.

5.48.2 Acquirer's responsibilities and considerations. MIL-STD-498's SCM requirements task the developer to "keep track of" everything during the course of the development. MIL-STD-498's SCM is an activity, not an organization. SCM may be performed by members of the development team, individuals within a project tasked with that responsibility, a separate organization, or other arrangement suitable for the project.

The standard requires the developer to establish levels of control for all work products. Some examples of possible levels of control and of things the developer might identify and control are:

Author control:

- Engineering data -- notes, records, work-in-progress (i.e., data specified in DIDs associated with particular development activities)
- Software development files

Project control:

- Source code files, data files, installation software
- Information in DIDs agreed upon by the project to be correct
- Reuse libraries
- Evaluation records

Organizational control:

- General purpose software -- operating systems, database management systems, e-mail, word processors, spreadsheets
- Engineering and development tools -- CASE tools, editors, compilers, debuggers, SCM tools, test software
- Computer system administrative tools and products -- diagnostic software, network managers, archives, backups
- Evaluation records

Acquirer control:

- Specifications

Some key goals of MIL-STD-498's SCM requirements are to ensure that the developer: keeps track of all software and software product descriptions associated with the project; implements only authorized changes to requirements; and knows what software and associated products match a specific set of requirements or changes to those requirements.

To implement changes to requirements, the acquirer and developer must agree upon what those changes are. When requirements have been defined and recorded as specifications and those specifications have been placed on contract, changes are implemented through contract modifications. When specifications have not been made a part of the contract, the

acquirer and developer will need to provide a means for controlling and making changes to requirements. These means can be as informal as a phone call or hand-shake, or as formal as documents signed by authorized acquirer and developer representatives. The standard does not provide contractual forms or notices concerning changes in requirements, such as Engineering Change Proposals (ECPs), Engineering Change Notices (ECNs), or notification to users of changes in a particular version of the software. Although the standard does provide a reminder in the form of two "shell" requirements to support acquirer configuration management activities for (1) proposing changes to acquirer controlled entities, and (2) supporting configuration audits, these activities may not apply to all projects. (See this guidebook's topics, *Contract* and *Requirements of the standard*, for additional information on "shell" requirements.)

All work products (including computerized files, the software products that constitute the development environment, and hardware), not just deliverables, are to be identified and controlled during the development and under MIL-STD-498's developer software configuration management activity. The physically controlled items can include: computer files, magnetic media (tapes, diskettes, video cassettes), paper documents, books, manuals, and drawings.

The standard leaves it up to the developer to describe what software configuration management records will be produced, when they will be produced, the level of detail of information that will be contained in each record and who is responsible for performing these activities.

5.48.3 Additional things to think about.

When software products are placed under project-level or higher configuration control, are problems handled via the corrective action system?
Are the delivery media compatible with that required or known to exist at the installation site(s)?
Are needed agreements with user, support personnel, and (possibly) certification agents in place if configuration audits are to be conducted?
Have the costs to conduct configuration audits (both in time and dollars) and to bring closure to any needed corrective actions been budgeted by both acquirer and developer?
Are mature SCM tools and procedures planned for use on the project?

5.48.4 Related guidebook topics.

Contract
Corrective action
Requirements of the standard
Software development environment
Software development files
Software development library
Software development process
Version/revision/release

5.49 Software development environment.

5.49.1 Requirements summary. MIL-STD-498 provides requirements for the establishment of a software development environment (SDE) as one of the standard's software development activities. References to ***Software development environment*** in MIL-STD-498 are listed in Figure 102, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.2, 3.37, 3.38, 3.43*, 4.1, 4.2.7, *Figure 1*, 5.2, 5.14.1,y *Figure 2*, Figure 3, *E.4.7, E.4.10, Figures 9-13*
CPM	3
SDP	5.2
STD	3
STR	3.2

FIGURE 102. MIL-STD-498's references to ***Software development environment***.

Providing a software development environment for the project involves establishing, controlling, and maintaining a software engineering environment (SEE), a software test environment (STE), a software development library (SDL), and software development files (SDFs). Establishing, controlling, and maintaining the SDE is required regardless of whether the environment consists of deliverable elements, non-deliverable elements, or some mixture of both. Both the SEE and STE consist of the facilities, hardware, software, firmware, procedures, and documentation needed to perform the software development effort. The elements of the SDE are used as follows:

- STE -- to perform qualification, and possibly other, testing of software
- SEE -- to perform all other software engineering efforts
- SDL -- to facilitate the orderly development and subsequent support of software
- SDFs -- to provide a repository for recording information about the development of the software.

MIL-STD-498's requirements for an SDE permit the use of non-deliverable software. The developer is to ensure that all non-deliverable software performs its intended functions. Figure 103 lists developer's key activities related to ***Software development environment***.

Developer's key activities related to ***Software development environment***	References in MIL-STD-498
Describe the approach to be followed for the software development environment, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software development environment.	SDP 5.2
Establish, control, and maintain the software development environment (SEE, STE, SDL, and SDFs). Ensure that each element of the SEE and STE, and all non-deliverable software performs its intended function(s). Maintain the SDL and the SDFs for the duration of the contract.	5.2
Assign a project-unique identifier to each software development environment element and place the element under configuration control.	5.14.1
Provide access for acquirer review of developer and subcontractor facilities including software engineering and test environments for activities required by the contract.	4.2.7
Describe the software test environment at each intended test site. Provide an assessment of the manner in which the test environment may be different from the operational environment and the effect of this difference on the test results.	STP 3 STR 3.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 103. Developer's key activities related to ***Software development environment***.

5.49.2 Acquirer's responsibilities and considerations. The developer is required to describe the methods/procedures/tools (approach) for establishing the software development environment in the SDP for the project. The developer's selection of the elements that make up the SEE and STE, and the approach used for SDFs and the SDL, should be consistent with the software development methods that are used on the project. Therefore, a project would probably not establish an SEE that incorporated design tools for both object oriented development and structured analysis and design methodologies. The software development tools and the software development methods chosen by the developer should complement one another and are key to the success of the developer's software development process. The developer's use of immature, unproven tools or tools that the developer has no prior experience with can pose a major cost and schedule risk.

The software development environment may vary from project to project depending on the complexity of the project, the developer's experience, company-wide procedures and tools, and other factors, such as program and build strategy. If a project consists of multiple builds, the developer may need to modify or update the software development environment for each new build (i.e., early builds may be tested using simulators while later builds may require the target computer). Figure 104 lists examples of the typical contents of the SDE.

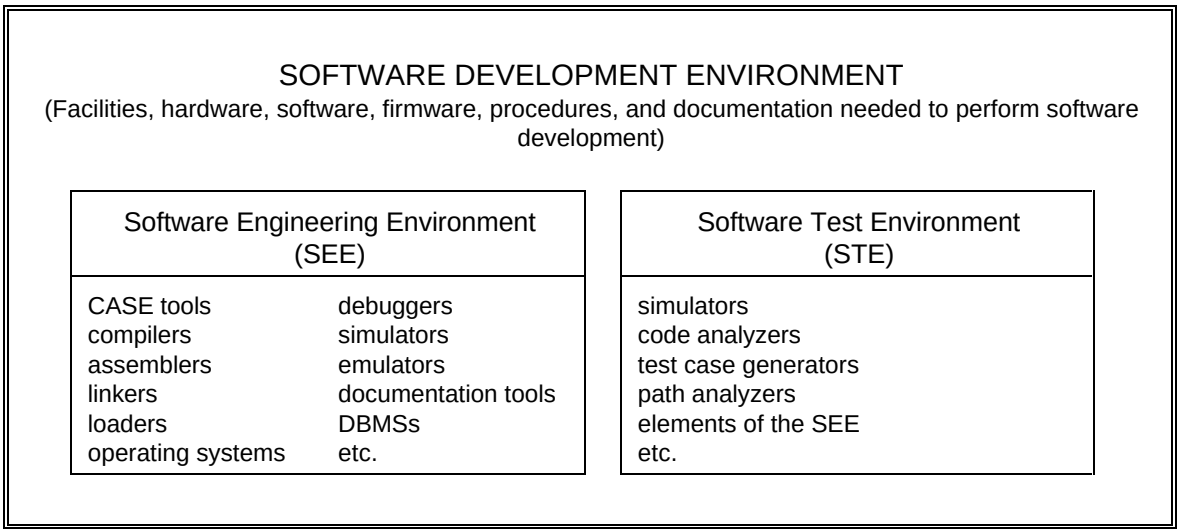


FIGURE 104. Typical elements of the software development environment.

MIL-STD-498 provides a default application of the standard to the software development environment. The default application is that the standard's requirements apply to all deliverable software, including the deliverable software in the software development environment. An acquirer may want to consider limiting deliverable documentation for deliverable SDE software, by tailoring the standard, to the Software Version Descriptions (SVDs), Software Product Specifications (SPSs) and user manuals. If highly complex relationships exist between products, a System/Subsystem Design Description (SSDD) that relates the various tools to one another might also be useful.

The developer's environment may include non-deliverable software as long as the operation and support of the deliverable software after delivery to the acquirer do not depend on the non-deliverable software or provision is made to ensure that the acquirer has or can obtain the same software. This requirement is intended to ensure that the user and support agency can obtain the software needed for use and support. (The SDE is usually known as the Software Support Environment after transition to the support environment or when a new SDE is established for support.)

If the acquirer is considering specifying requirements for the content of the software development environment, trade-offs should be made between the potential benefits to be gained from having multiple projects use a consistent environment in support against the disadvantages of forcing a developer to use methods/procedures/tools that may differ from well-established company-wide processes or an existing software development environment.

MIL-STD-498's joint technical and management reviews provide for acquirer-developer discussion of the software development environment. For example, joint reviews that include software development environment issues might discuss:

- Open issues regarding the status of the preparations and activities for the transition of the software development environment to the acquirer
- Status of licenses, data rights, etc.
- Modifications or updates to the software development environment needed for subsequent builds, including methods and procedures used

5.49.3 Additional things to think about.

Does the developer have experience with the individual components of the SDE? With the integrated environment?
What is the relationship between the software development environment and the software support environment?
Do the developer's plans for software transition cover all activities relevant to the transition of the deliverable elements of the SDE, including the transfer of software licenses for CASE tools and other COTS software, if applicable?
If SDFs reside in software tools, how will the data residing in the tools be converted if the tools are not deliverable?
Has the developer ensured that the acquirer has or can obtain all non-deliverable SDE software that is needed for the operation and support of the deliverable software?
Is there potential for contention between the SEE and the STE resources?
Is the developer using or planning to use immature or unproven SDE elements?

5.49.4 Related guidebook topics.

Builds
Software configuration management
Software development files
Software development library
Software development methods
Software engineering environment
Software test environment

5.50 Software development files.

5.50.1 Requirements summary. MIL-STD-498 provides requirements for the developer to establish, control, and maintain software development files (SDFs). The purpose of an SDF is to provide a repository for recording information about the development of a particular body of software and to require the developer to implement SDFs appropriate for the software development process and methods used on the project. SDFs are intended to be used by the developer throughout the development process to capture design and test information as it is produced. The standard leaves it up to the developer to decide whether SDFs are maintained as electronic or paper files, or some combination, and to describe the approach in the SDP. The SDP is subject to acquirer review and approval. The standard does not provide requirements for delivery of SDFs nor a DID to specify format and content requirements and the acquirer does not approve or disapprove SDFs, although they may be reviewed at any time. References to ***Software development files*** in MIL-STD-498 are listed in Figure 105, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.34*, 5.2.4, 5.7.2, 5.7.4, 5.7.5, 5.8.1, 5.8.3, 5.8.4, 5.9.4, 5.9.6, 5.10.1, 5.10.3, 5.10.4, 5.11.4, 5.11.6, 5.15, Figure 3, Figure 6 (Item 29), *Figures 9-12*
SDP	5.2.4, 5.7.2, 5.7.4, 5.7.5, 5.8.1, 5.8.3, 5.8.4, 5.9.4, 5.9.6, 5.10.1, 5.10.3, 5.10.4, 5.11.4, 5.11.6, 5.15

FIGURE 105. MIL-STD-498's references to ***Software development files***.

The standard provides requirements for developer evaluation of a sample of SDFs using the criteria in Figure 6 of the standard. Evaluation criteria include:

- Do the SDFs meet the SOW provisions regarding SDFs?
- Are the SDFs understandable by the intended audience (e.g., programmer-to-programmer information need not be understandable by testers)?
- Are the SDFs internally consistent?
- Do the SDFs contain what the SDP states they will contain?
- Are SDF contents current with the ongoing software development effort?
- Are test cases/procedures/data/results adequate for this particular body of software?

Figure 106 lists developer's key activities related to ***Software development files***.

Developer's key activities related to ***Software development files***	References in MIL-STD-498
Describe the approach to be followed for software development files, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software development files.	SDP 5.2.4, 5.7.2, 5.7.4, 5.7.5, 5.8.1, 5.8.3, 5.8.4, 5.9.4, 5.9.6, 5.10.1, 5.10.3, 5.10.4, 5.11.4, 5.11.6, 5.15
Establish, control, and maintain an SDF for each software unit or logically related group of software units, for each CSCI, and, as applicable, for logical groups of CSCIs, for subsystems, and for the overall system. Record information about the development of the software in the appropriate SDFs and maintain the SDFs for the duration of the contract.	5.2.4
Record test information in SDFs (see Figure 107 of this guidebook for details).	5.7.2, 5.7.4, 5.7.5, 5.8.1, 5.8.3, 5.8.4, 5.9.4, 5.9.6, 5.10.1, 5.10.3, 5.10.4, 5.11.4, 5.11.6
Perform software product evaluations on a sampling of SDFs using the criteria listed in Figure 6 of the standard.	5.15, Figure 6 (Item 29)
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 106. Developer's key activities related to ***Software development files***.

5.50.2 Acquirer's responsibilities and considerations. The standard's requirements for SDF contents may be met by SDFs that contain only test information or by SDFs that contain virtually all software product information generated during development. As automation of software development evolves, typical SDFs may include electronic pointers to all applicable information for a project.

The contents of an SDF 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. Despite this description of typical SDF contents, the standard's only actual requirements concerning SDF contents are the requirements to record test information (see Figure 107). Note that all information regarding developer-internal testing and the dry run portion of qualification testing information is recorded in SDFs. Other qualification test information is recorded in the Software Test Plan (STP), Software Test Description (STD), and Software Test Report (STR) for each CSCI, subsystem, and the system.

Test Level	MIL-STD-498 requirement to record test information in SDFs		
	Record test cases, test procedures, and test data in SDFs	Update SDFs based on the results of revision and retesting	Record test results and analysis in SDFs
Unit testing	5.7.2	5.7.4	5.7.5
Unit integration testing	5.8.1	5.8.3	5.8.4
Dry run of CSCI qualification testing	--	5.9.6	5.9.4
CSCI qualification testing	--	5.9.6	--
CSCI/HWCI integration testing	5.10.1	5.10.3	5.10.4
Dry run of system qualification testing	--	5.11.6	5.11.4
System qualification testing	--	5.11.6	--

FIGURE 107. MIL-STD-498's requirements to record test information in software development files.

The standard is flexible regarding how many SDFs are required. Figure 108 illustrates this flexibility. Note that the standard requires the development and recording of planning and engineering information (content of the DIDs) regardless of whether or not a deliverable is required. Unless a deliverable document is required via the CLIN or CDRL, the standard is silent regarding where this information is to be recorded and its format.

Develop SDFs for:	Required
Software unit or logically related group of software units	Yes
CSCI	Yes
Logical groups of CSCIs	As applicable
Subsystems	As applicable
System	As applicable

FIGURE 108. MIL-STD-498's requirements on levels of software development files.

If an acquirer needs deliverable SDFs based on the selected software support concept or other considerations, the acquirer may specify:

- Delivery of SDFs in contractor format. If developer format and content is acceptable, the SDFs can be ordered as a CLIN.

- Delivery of SDFs via a DID. MIL-STD-498 does not have a DID for SDFs. If developer content and format is not acceptable, the acquirer may develop a one-time DID specifying required content and format. One DID that may be suitable for this purpose is DI-MGMT-80614, Project Folders. The acquirer may also want to consider a specific on-line or hardcopy format for the SDFs or require the use of specific tools to generate the information.

When the acquirer does not know whether SDFs are needed, the acquirer may want to put the Data Accession List on CLIN or CDRL and order SDFs (or selected SDFs) at a later date, as needs are identified. (See this guidebook's topic, *Data accession list*, for more information.)

5.50.3 Additional things to think about.

Is the developer's sampling approach suitable for effective SDF evaluation?
Have the software development files been evaluated for potential reuse of test and load routines?
Is the developer recording required information in the SDFs? If SDFs are automated, is information accessible to the acquirer for review?
Is the developer's approach to SDF implementation consistent with the selected program strategy (i.e., Grand Design, Incremental, Evolutionary, other)?

5.50.4 Related guidebook topics.

Access for acquirer review
Data accession list
Joint technical and management reviews
Qualification testing
Software configuration management
Software development environment
Software development planning
Software development process

5.51 Software development library.

5.51.1 Requirements summary. MIL-STD-498 defines the software development library (SDL) as "*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*." References to ***Software development library*** in MIL-STD-498 are listed in Figure 109, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.35*, 5.2.3, Figure 3, *Figures 9-12*
SDP	5.2.3

FIGURE 109. MIL-STD-498's references to ***Software development library***.

The developer is required to describe the methods/procedures/tools (approach) for providing the SDL in the SDP for the project. All applicable contract clauses pertaining to the SDL are to be addressed in this planning. As part of the overall activity of establishing, controlling, and maintaining the project's software development environment (SDE), the developer establishes, controls and maintains the SDL. An SDL is to be employed throughout the duration of the contract. If a system/subsystem or CSCI is developed in multiple builds, the SDL in each build is understood to take place in the context of the software products and controls in place at the start of the build. Figure 110 lists developer's key activities related to ***Software development library***.

Developer's key activities related to ***Software development library***	References in MIL-STD-498
Describe the approach to be followed for the software development library, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software development library.	SDP 5.2.3
Establish, control, and maintain a software development library to facilitate the orderly development and subsequent support of software.	5.2.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 110. Developer's key activities related to ***Software development library***.

5.51.2 Acquirer's responsibilities and considerations. The SDL is an integral part of the SDE, and as such, will probably reflect the sophistication of the SDE. For example, the SDL for a small project of one or two programmers using basic programming tools might consist of a single directory for source code, another directory for object code, and a third directory for executables. The SDL of a large project with a staff of 25 to 100 engineers, however, might consist of a file management system that reserves (e.g., locks) files for use by one author, automatically tracks all versions of files placed in it, and provides sophisticated features for managing sets of files and producing status reports.

There are many ways that SDLs can be used on a project and a project can have more than one SDL. For example, the software engineering group might use an SDL to control files undergoing implementation and unit test, while the software configuration management (SCM) group might use a different SDL to control software builds and delivered documents. At the same time, the software test group might have a separate SDL to control different versions of software undergoing qualification test. The same files wouldn't necessarily reside in all three SDLs. The developer's SDP describes how the project organizes its SDL(s), uses its SDL(s), and who is responsible for controlling their contents.

5.51.3 Additional things to think about.

Are the SDL control procedures and security measures (file access) adequate for the project?
Are there plans for modifying or updating the SDL(s) over time along with other elements of the software development environment?
If the developer is using new automated SDL tools, what backup methods/procedures/tools exist?
Is there an archive or backup mechanism in place for SDL materials?
Is the SDL organized to provide a smooth transition to the support environment?

5.51.4 Related guidebook topics.

Reusable software products
Software configuration management
Software development environment
Software engineering environment
Software test environment

5.52 Software development methods.

5.52.1 Requirements summary. MIL-STD-498 provides requirements for using software methods, but does not provide requirements to use a particular software development method. References to ***Software development methods*** in MIL-STD-498 are listed in Figure 111, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword-5*, 4.2.1
SDP	4.2.1, 5

FIGURE 111. MIL-STD-498's references to ***Software development methods***.

The developer is required to describe or reference the software development methods to be used in the SDP for the project and identify any constraints on the use of those software development methods. The SDP will also describe procedures and tools that support the software development methods for each activity applicable to the project. For each of the standard's activities, the developer will describe, in the project's SDP, the methods/procedures/tools to be applied to: (1) the analysis or other technical tasks involved, (2) the recording of results, and (3) the preparation of associated deliverables, if applicable. The description will identify applicable risks/uncertainties and plans for dealing with software development method risks. If different builds or different software on the project require different planning, these differences are to be noted. Figure 112 lists developer's key activities related to ***Software development methods***.

Developer's key activities related to ***Software development methods***	References in MIL-STD-498
Describe or reference the software development methods to be used. Include descriptions of the manual and automated tools and procedures to be used in support of these methods. Cover all contractual clauses pertaining to software development methods.	SDP 4.2.1, 5
Use systematic, documented methods for all software development activities. Describe these methods in, or reference these methods from, the software development plan for the project.	4.2.1
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 112. Developer's key activities related to ***Software development methods***.

5.52.2 Acquirer's responsibilities and considerations. MIL-STD-498 purposely does not specify software development method(s) to be used on the project. Software development methods are constantly evolving to meet new technology and better ways of doing things. Many "best practices" exist to perform the activities required by the standard. Each developer exerts a significant effort in adapting selected methods/procedures/tools to the people, culture, and projects specific to that developer. What works well for one developer may not work for another.

"Method" is a way, technique, or process for doing something. Software development methods are the ways, techniques, or processes used for developing software. Although the term "software development methods" in the standard applies to all activities in the standard, the term is often used for referring to procedures and tools associated with the "front-end" activities of software design and coding or with an overall approach to the complete process. The standard does not require automated methods, but allows for them to be used. The standard is flexible. It allows a developer to select the methods/procedures/tools and to organize the activities in a way that works for the project.

Software development methods/procedures/tools need to change over time. Retooling of the software development environment is a problem software developers have in common with hardware developers. A hardware manufacturer's assembly line consists of the tools and facilities needed to develop products. Because hardware is not flexible and is hard to change, the tools found in the assembly line are usually upgraded gradually over time, or only when a new line is undertaken. Software development's production line consists of the tools and equipment found in the software development environment. Software, unlike hardware, is flexible and easily changed, and computers, compilers, languages, etc., by which software is developed are also easily changed. Computer manufacturers are constantly improving the speed and memory capacity of the hardware. These factors cause the market to drive a much more rapid pace of software retooling than is usually the case on the hardware production line.

To keep current, to take advantage of productivity enhancing tools in the software engineering and test environments, and to retain market competitiveness, developers need to retool. This can be disruptive on an ongoing project. The acquirer should recognize the developer's need for change over time, allow time for that change in overall software development schedules, recognize the possible need to modify software development methods along with the procedures and tools used, and accept those changes as part of the software project, when beneficial. Similar to hardware production, the best time to modify the "software production line" is at the start of a new project (product line) or when a major revision or new build is about to begin.

A few software development methods, with associated representations/notations, in practice at the time of this guidebook's publication are:

- Box Structured Design
- Clean Room
- Code Inspections
- Data Modeling, e.g., IDEF1X
- Entity-Relationship Diagramming
- Essential Systems Analysis
- Information Engineering (IE)
- Joint Application Development, i.e., JAD
- Object Oriented Techniques
- Petri Nets
- Process Modeling, e.g., IDEF0
- Rapid Application Development, i.e., RAD
- Structured Analysis
- Structured Design

5.52.3 Additional things to think about.

Are acquirer personnel or their authorized representatives familiar with the methods/procedures/tools used for software development on the project? If not, have resources and time been allocated to train acquirer personnel? Have resources and time been allocated in the development schedule to familiarize acquirer personnel with the specific application of the methods/procedures/tools for the project?
Are support agency personnel familiar with the software development methods planned for use on the project? If not, has time been planned to train support agency personnel on use of the methods/procedures/tools for the project, if needed? Have funds been planned for this training?
Does the software development plan describe the software development methods for each activity?
Are any of the methods/procedures/tools used for software development proprietary? How will the acquirer or their authorized representatives gain access to proprietary methods/procedures/tools?
What are the plans for dealing with company-wide enhancements or changes in methods/procedures/tools that affect the project?
Are the developer's personnel familiar with the software development methods planned for the project? Does the developer have experience with the tools needed to implement the methods, if any?
Has time been planned in the schedule that allows for changing software development methods/procedures/tools over time, if needed? Have funds been planned for these changes?
Has time been planned in the schedule to allow for training associated with any changes in the methods/procedures/tools for the project, if needed? Have funds been planned for this training?

5.52.4 Related guidebook topics.

Documentation (Preparing documents)
Documentation (Recording information)
Risk management
Standards for software products

5.53 Software development planning.

5.53.1 Requirements summary. MIL-STD-498 provides requirements for the developer to develop and record plans for conducting the activities of the standard and for other software-related requirements in the contract. The developer's planning is to be consistent with system-level planning and to cover all applicable items in the SDP DID. The SDP describes a developer's plans for conducting a software development effort. The term "software development" is meant to include new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products. The SDP provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods/procedures/tools (approach) to be followed for each activity, and project schedules, organization, and resources. References to the ***Software development planning*** in MIL-STD-498 are listed in Figure 113, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.4.2*, 4.1, 5.1.1, 5.1.6, *5.14.2 Note*, 5.16.1, 5.16.2, 5.17.1, 5.17.2, 5.19.1, 5.19.2, 5.19.7, *6.2*, *Figure 2*, B.3, Figure 4, Figure 6, D.4.7, *E.4.1, Figures 9-13, G.6.3, H.4, Figures 15-17*
SDP	All

FIGURE 113. MIL-STD-498's references to ***Software development planning***.

MIL-STD-498's SDP DID mirrors Sections 4 and 5 of the standard. The developer is required to describe the methods/procedures/tools (approach) in the SDP for the project to be applied to: (1) the analysis or other technical tasks involved, (2) the recording of the results of the activity, (3) the preparation of associated deliverables, if applicable. The discussion is to include identification of applicable risks and uncertainties and plans for dealing with them.

The SDP covers all activities required by MIL-STD-498. Portions of the plan may be bound or maintained separately if this approach enhances the usability or maintenance of the information. For example, the developer may want a separate plan for software configuration management for ease in updating software configuration management procedures when it is known that new tools will be added to automate manual procedures over time. Figure 114 lists developer's key activities related to ***Software development planning***.

Developer's key activities related to ***Software development planning***	References in MIL-STD-498
Describe the approach to be followed for software development planning identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software development planning.	SDP 5.1.1, 5.1.6
Develop and record plans for conducting the activities required by the standard and by other software-related requirements in the contract. Ensure that plans are consistent with system-level planning. Include all applicable items in the SDP DID.	5.1.1 SDP
Obtain acquirer approval of software development plans. Conduct the relevant activities in accordance with the plan. Have developer's management review the software development process at intervals specified in the SDP to assure that the process complies with the contract and adheres to the plans. Obtain acquirer approval for all updates to the SDP (developer-internal scheduling and related staffing information updates need not have acquirer approval.)	5.1.6
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 114. Developer's key activities related to ***Software development planning***.

5.53.2 Acquirer's responsibilities and considerations. The developer's SDP for the project is key to the acquirer's understanding and approval of the software development process proposed for the project. MIL-STD-498 provides a framework for the software development process. MIL-STD-498 is not the software development process for the project. The standard defines what activities are to be performed. MIL-STD-498 does not prescribe how to perform them. The SDP for the project describes the developer's methods/procedures/tools (approach) for performing the activities, that is, the developer's software development process (how) for the project. (See this guidebook's topic, *Software development process*, for more information.)

The developer's SDP describes the approach to be followed, risks/uncertainties and plans for dealing with them. By approving the SDP, the acquirer says to the developer, "This approach appears to be sound and can be used as the basis for work." Approval of the SDP does not shift the responsibility from the developer to meet contractual requirements. For example, if the contract requires testing the software, the developer is required to test the software even though the SDP is silent on the developer's approach to testing. (See this guidebook's topic, *Approval by the acquirer,* for more information.)

MIL-STD-498 also provides requirements to ensure that the developer's software development planning is consistent with system-level planning. When the project involves both hardware and software development, the SDP will address how the software development personnel will

participate in the system-level activities including system requirements analysis, system design, CSCI integration and testing, and system qualification testing. If the development is performed in builds, the SDP will identify the builds, their objectives, and the software development activities to be performed in each build. Planning for each build should include overall planning for the contract, detailed planning for the current build, and planning for future builds covered under the contract to a level of detail compatible with the information available. If the project includes reusable software, the evaluation criteria for the reusable software products are to be included in the SDP.

The SDP relates requirements and constraints for the proposed system to the development process in the areas of:

- The system and software to be developed
- Project documentation
- The position of the project in the system life cycle
- The selected program/acquisition strategy
- Project schedules and resources
- Project security, privacy, safety, and other critical requirements
- Methods, standards, and interdependencies between hardware and software development

The developer may reference the software methods to be used for the activities and reference the standards to be followed for representing requirements, design, code, test cases, test procedures, and test results rather than include these in the project's SDP. If the SDP references, rather than includes the methods and standards, the SDP's Referenced documents section will list the number, title, revision, and date of the document containing the methods and standards, and identify the source for all documents not available through normal government stocking activities. If the SDP references company proprietary methods and standards, the developer and acquirer should agree that the referenced approach is acceptable, and agree upon the procedures to be followed for acquirer or the acquirer's authorized representative to review those methods and standards. (See this guidebook's topic, *Access for acquirer review,* for more information.)

Section 4.11 of this guidebook explains different ways for putting MIL-STD-498 on contract. Two ways listed rely on putting the SDP on the contract. A drawback of these methods is that changing the SDP during the project may require a contract modification.

Plans should be living documents that represent how the project work is performed. The SDP has been designed to provide the acquirer insight into, and a tool for monitoring, the processes to be followed for software development. MIL-STD-498's approach to a software development process is to require the developer to plan the work, then to work the plan.

5.53.3 Additional things to think about.

Does the SDP address all activities? All items specified in the contract?
Are the SDP's schedules and activities coordinated with other system and software plans?
Is the mechanism for updating the SDP clear? Is the SDP kept current with decisions agreed upon at joint management reviews?
Are key decisions and rationale being recorded?
Are the methods/procedures/tools to be used for the project systematic and well-documented?
Are the acquirer's project personnel familiar with the methods/procedures/tools to be used for the project? If not, have resources and time been planned for learning how they apply to the project?
Are the developer's project personnel familiar with the methods/procedures/tools to be used for the project? If not, have resources and time been planned for learning how they apply to the project?
Have plans and procedures been defined for access to any proprietary methods/procedures/tools?

5.53.4 Related guidebook topics.

Access for acquirer review
Approval by the acquirer
Software development process
Other activity and product topics, such as Detailed design, Process Improvement, Requirements, etc.

5.54 Software development process.

5.54.1 Requirements summary. MIL-STD-498 defines a "software development process" as "*an organized set of activities performed to translate user needs into software products*." References to ***Software development process*** in MIL-STD-498 are listed in Figure 115, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.36*, 4.1, *Figure 1*, 5.19.1, 5.19.2, 5.19.7, *6.5, Figure 2, Appendix G, Figures 9-12, H.4*
SDP	4.1, 5.19.1, 5.19.2, 5.19.7

FIGURE 115. MIL-STD-498's references to ***Software development process***.

MIL-STD-498 contains activities for establishing a software development process; the standard's software development activities are listed in Figure 116. The standard's activity definitions are basic elements in beginning-to-end software development and require tailoring to meet contractual objectives of a less than complete development or for other considerations. (See the MIL-STD-498 Overview and Tailoring Guidebook for information on tailoring.) MIL-STD-498 does not impose a sequence on the software development activities provided in the standard. The activities may overlap, may be applied iteratively to different elements of software, and may be performed in any order suitable for the project. They cover new software development, modification, reuse, reengineering, maintenance and all other activities resulting in software products.

The developer is required to describe and record the approach to performing the specified activities in the SDP for the project. Following acquirer approval of the SDP, the developer is required to conduct the relevant activities in accordance with the plan, and provide for developer management review of the software development process at intervals specified in the SDP to assure that the process complies with the contract and adheres to the plans. Updates to the SDP, with the exception of developer-internal scheduling and related staffing information, are subject to acquirer approval. (See this guidebook's topic, *Software development planning*, for more information.)

MIL-STD-498 Software Development Activities	
5.1 Project planning and oversight	5.11 System qualification testing
5.2 Establishing a software development environment	5.12 Preparing for software use
5.3 System requirements analysis	5.13 Preparing for software transition
5.4 System design	5.14 Software configuration management
5.5 Software requirements analysis	5.15 Software product evaluation
5.6 Software design	5.16 Software quality assurance
5.7 Software implementation and unit testing	5.17 Corrective action
5.8 Unit integration and testing	5.18 Joint technical and management reviews
5.9 CSCI qualification testing	5.19 Other activities: • Risk management • Software management indicators • Security and privacy • Subcontractor management • Interface with software IV&V agents • Coordination with associate developers • Improvement of project processes
5.10 CSCI/HWCI integration and testing	

FIGURE 116. MIL-STD-498 software development activities.

The intent of the standard's requirements for activities is that a clearly defined software development process be used for software development on the project and to periodically assess that process. When beneficial improvements can be made to the software development process, the standard provides requirements for proposing process improvements through updates to the SDP and implementing them, when approved. Figure 117 lists developer's key activities related to ***Software development process***.

Developer's key activities related to ***Software development process***	References in MIL-STD-498
Describe the software development process to be used. Cover all contractual clauses concerning the software development process, identifying planned builds, their objectives, and the software development activities to be performed in each build.	SDP 4.1
Describe the approach to be followed for improvement of project processes. Cover all contractual clauses regarding process improvement.	SDP 5.19.7
Establish a software development process consistent with contract requirements. It shall include major activities listed in Figure 116, which may overlap, may be applied iteratively, may be applied differently to different elements of software, and need not be performed in the order listed. Describe the process in the SDP.	4.1
Periodically assess the processes used on the project and implement improvements if approved by the acquirer.	5.19.7
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 117. Developer's key activities related to ***Software development process***.

5.54.2 Acquirer's responsibilities and considerations. MIL-STD-498's software development activities are stated as required tasks to be performed. The standard does not specify how to perform those tasks. MIL-STD-498's flexibility allows the developer to use any well-documented software development methods/procedures/tools (approach) to accomplish the required tasks and develop the required information.

Although a somewhat logical, possibly sequential, list of activities is presented in the standard, the developer is responsible for selecting and organizing the activities into a software development process useful for the project. The developer provides details of the approach to be used for software development including methods/procedures/tools to be applied to each activity and product required for the project. The acquirer should ensure that CLIN or CDRL dates for deliverable items do not impose unrealistic, undesirable sequencing of software development activities thereby adversely affecting the developer's ability to develop the software using the software development process best suited to the project. Nor should unrealistic, undesirable schedules for joint technical and management reviews be imposed that would unnecessarily constrain the developer's process. MIL-STD-498's Figures 9-12 provide some possible ways of applying the standard on a project. (See this guidebook's topics, *Contract data requirements list* and *Joint technical and management reviews*, for additional information.)

The standard's activities are supported by three mandatory appendixes and twenty-two software product descriptions in the form of DIDs. Six additional appendixes aid in understanding, interpreting, and applying the standard on a project. Together, the standard, its appendixes, and DIDs form the basis for development of a software development process and for definition of the data to be generated by those processes, as applicable to the project.

Figure 118 shows, simplistically, the process a developer follows to define a software development process for a project.

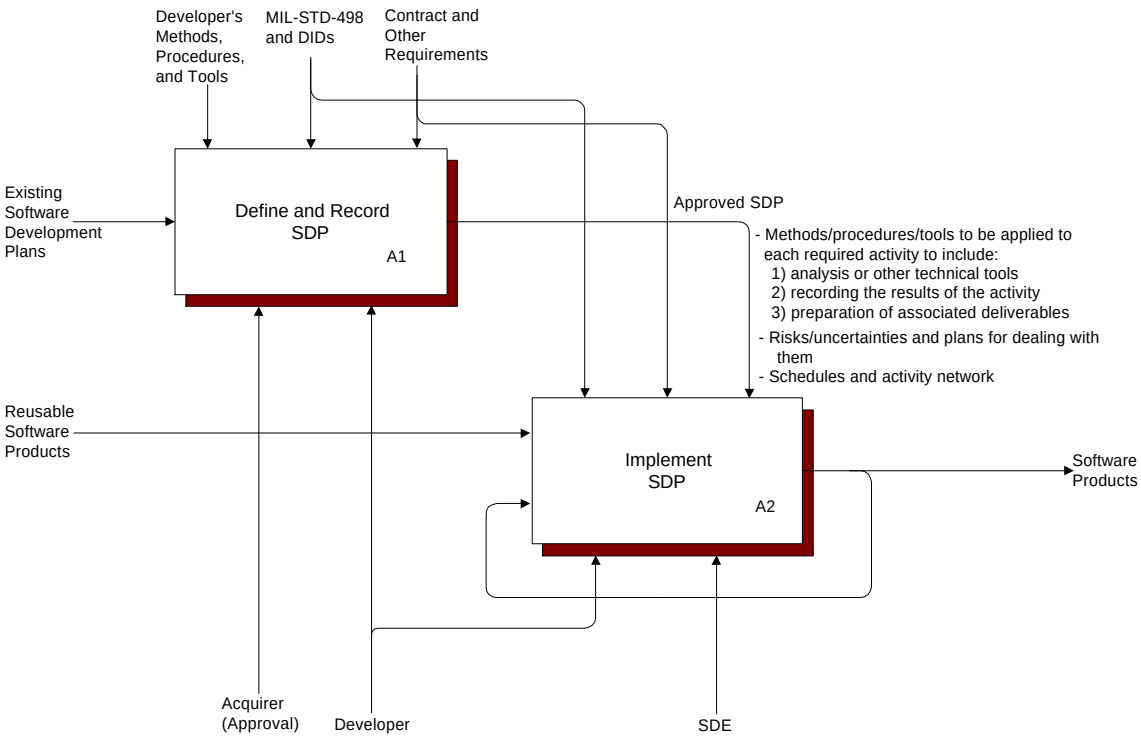


FIGURE 118. IDEF0 model to define a project's software development process.

5.54.3 Additional things to think about.

Is the developer's process assessment approach suitable and in place?
Is sufficient detail provided in the SDP to allow the acquirer to evaluate the suitability of the proposed software development process?
Is the developer's software development process for the project based on well-defined organizational process definitions (e.g., policies, standards, procedures)?

5.54.4 Related guidebook topics.

Contract data requirements list
Joint technical and management reviews
Risk management
Software development methods
Software development planning

5.55 Software engineering environment.

5.55.1 Requirements summary. MIL-STD-498 defines the software engineering environment (SEE) as "*the facilities, hardware, software, firmware, procedures, and documentation needed to perform software engineering.*" References to the ***Software engineering environment*** in MIL-STD-498 are listed in Figure 119, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.2, 3.37, 3.38,* 4.1, 4.2.7, 5.2.1, 5.2.3, *Figures 9-13*
CPM	3
SDP	4.2.7, 5.2.1
STR	3.2

FIGURE 119. MIL-STD-498's references to ***Software engineering environment***.

The SEE is a subset of the software development environment (SDE). The standard makes a distinction between the software test environment (STE), and the software engineering environment (SEE). The STE is that part of the SDE needed to perform qualification, and possibly other, testing of software. The SEE contains the elements needed to perform all activities except for qualification testing. This distinction is made to facilitate tailoring and allow the testing task to be contracted separately from the development task. However, the software engineering and software test environments need not be separated and resources common to both need not be duplicated. Resources that are part of the software test environment may also be part of the software engineering environment. Figure 104 of this guidebook's topic, *Software development environment*, lists examples of the typical components of the SEE and the STE. Figure 120 lists developer's key activities related to ***Software engineering environment***.

Developer's key activities related to ***Software engineering environment***	References in MIL-STD-498
Describe the approach to be followed for the software engineering environment, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the software engineering environment.	SDP 4.2.7, 5.2.1
Establish, control, and maintain a software engineering environment to perform the software engineering effort. Ensure that each element of the SEE performs its intended function.	5.2.1
Provide acquirer access to developer facilities for review of SEE.	4.2.7
Provide an assessment of the manner in which the SEE may be different from the operational environment and the effect of this difference on the qualification test results.	STR 3.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 120. Developer's key activities related to ***Software engineering environment***.

5.55.2 Acquirer's responsibilities and considerations. See this guidebook's topic, *Software development environment*, for more information.

5.55.3 Additional things to think about.

Does the developer have experience with the individual components of the SEE? With the integrated environment?
Do the developer's plans for software transition cover all activities relevant to the transition of the deliverable elements of the SEE including the preparation of support manuals (computer programming manuals and firmware support manuals), if applicable?
Does the designated software support environment include all necessary elements of the SDE?
Has the developer ensured that the acquirer has or can obtain all non-deliverable SEE software that is needed for the operation and support of the deliverable software?
Is there a conflict between the availability of SEE and STE resources?
Does the SEE support the planned software development process for the project?
Are SEE tool acquisition, development, modification, installation, integration, etc., on schedule?

5.55.4 Related guidebook topics.

Software configuration management
Software development environment
Software development library
Software development methods
Software support
Software test environment

5.56 Software implementation and unit testing.

5.56.1 Requirements summary. MIL-STD-498 provides requirements for the developer to perform software implementation and unit testing. "Software" includes both computer programs and computer databases and "implementation" is the act of converting software design into software units, including, as applicable, coding computer instructions and data definitions, building databases, populating databases and other data files with data values, and other activities needed to implement the design. "Unit testing" is the testing of whatever entity has been identified as a unit for the language and methodology used. Since units may consist of other units, some unit integration and testing may take place during testing of the individual units of which the aggregate is comprised.

The methods/procedures/tools (approach) to be followed for software implementation and unit testing are described in the SDP for the project, along with the developer's description of software units for the methods proposed for the project. (See this guidebook's topic, *Software unit*, for more information.) References to ***Software implementation and unit testing*** in MIL-STD-498 are listed in Figure 121, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.1, *Figure 1*, 5.7, 5.9.1, 5.11.1, Figure 3, Figure 6, *Figures 9-12*, *Figure 14*
SDP	5.7
SPS	6

FIGURE 121. MIL-STD-498's references to ***Software implementation and unit testing***.

The term "implemented software" is intended to cover data, data files, databases, or other representations that may not be covered under the terms "executable" and "source."

Software implementation and unit testing activities include:

- Software implementation
- Preparing for unit testing
- Performing unit testing
- Revision and retesting
- Analyzing and recording unit test results

If a CSCI is developed in multiple builds, implementation and unit testing of that CSCI will not be completed until the final build. Implementation and unit testing in each build is interpreted to include those units, or parts of units, needed to meet the CSCI requirements to be implemented in that build.

MIL-STD-498 provides requirements for independence in qualification testing. The standard does not require independence in developer-internal testing, such as unit testing. (See this guidebook's topic, *Testing (Developer-internal*, for more information.) Figure 122 lists developer's key activities related to ***Software implementation and unit testing***.

Developer's key activities related to ***Software implementation and unit testing***	References in MIL-STD-498
Describe the approach to be followed for software implementation and unit testing, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software implementation and unit testing.	SDP 5.7
Develop and apply standards for representing code, test cases, and test procedures. Describe or reference the standards in the SDP.	4.2.2 SDP 4.2.2
Develop and record software corresponding to each software unit in the CSCI design. Obtain acquirer approval to use any programming language not specified in the contract for deliverable software.	5.7.1
Establish test cases in terms of inputs, expected results, and evaluation criteria; test procedures; and test data for testing the software corresponding to each software unit. Cover all aspects of the unit's detailed design and record the information in the appropriate SDFs.	5.7.2
Test the software corresponding to each software unit in accordance with the unit test cases and procedures.	5.7.3
Make all necessary revisions to the software, perform all necessary retesting, and update the SDFs and other software products as needed.	5.7.4
Analyze the results of unit testing and record the test and analysis results in appropriate SDFs.	5.7.5
Trace from each CSCI source file to the software units(s) that it implements. Trace each software unit to the source files that implement it.	SPS 6.a, 6.b
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 122. Developer's key activities related to ***Software implementation and unit testing***.

5.56.2 Acquirer's responsibilities and considerations. MIL-STD-498 is intended to be language and methodology independent; that is, a developer should be able to use the standard and its DIDs regardless of whether the software is to be coded in Ada, C, COBOL, assembly, or any other language, or developed by database, spreadsheet, knowledge-based language, code generator, or other method. The standard is designed to accommodate hierarchical, object-oriented, structured, or any design method needed to develop computer programs and databases. The developer is required to describe the methods/procedures/tools and coding standards to be used in the SDP for the project. Note that the standard requires the developer to obtain acquirer approval to use any programming language not specified in the contract. (See this guidebook's topic, *Programming languages*, for more information.)

A software development process is intended to transform a set of requirements into executable code. MIL-STD-498's traceability requirements support this intention. Requirements are traced between levels and into implemented source code files. The acquirer should not expect to see a clean one-to-one mapping between requirements and software units or between software units and source code files. Requirements may be transformed, divided, combined, retransformed, redivided and recombined many times during the design of a system. Traceability, then, at any one level might reflect a one-to-many, many-to-many, or many-to-one picture that fans in and out over the transformation process.

Software implementation and unit testing usually take place in the software engineering environment for the project. Unit testing may use elements of the software test environment identified for qualification testing. The developer's software engineering environment may be one specifically designed for the project or part of a larger combined-project, department-wide, or company-wide software engineering environment.

Unit testing is usually performed before software is placed under project-level software configuration control, often while the software is under software configuration author-control. Since problem/change reports resulting from evaluations are required for products under project-level or higher configuration control, it would be unusual to find problem/change reports on the software as a result of unit testing. (Problem/change reports might be generated against requirements and/or design under project-level control, however, as a result of implementation and/or unit testing.) If a developer has a process in place that requires problem/change reports for unit testing, that process need not be changed. The intent of the requirement is to limit problem/change reports, which are frequently part of a company-wide process, to those items that had been thoroughly "wrung-out" to avoid clogging the developer's system and to avoid the formality sometimes associated with correcting problems under project-level or higher control.

A project or team may implement a problem/change reporting system for products under less than project-level control, but is not required to do so. Many developers have institutionalized team-level peer reviews and informal evaluations and may use them even if not required to do so on a project. In general, a developer may provide a more disciplined approach, within cost and schedule constraints, than the standard requires, (e.g., formal peer reviews, problems/change reports on unit testing, etc.), but not less.

5.56.3 Additional things to think about.

Is the software engineering environment complete? Do all tools perform as intended? Have implementors been trained on the use of the tools?
Are implementation and unit testing procedures mature and documented?
Has the developer evaluated reusable software units?
Is the developer recording required information in the SDFs? If SDFs are automated, is information accessible to the acquirer for review?
Are key decisions and rationale being recorded?
Are schedules for implementation and unit testing coordinated with integration plans?
Are testing tools ready for unit testing?
If prototype hardware is needed to test units, is the hardware stable (not going to be changed)?
Has the developer identified any need to use programming languages other than those specified in the contract? If so, has approval been provided by the acquirer prior to using those languages?

5.56.4 Related guidebook topics.

Corrective action
Problem/change report
Programming languages
Qualification testing
Reusable software products
Software development files
Software engineering environment
Software test environment
Software unit
Standards for software products
Testing (Developer-internal)
Traceability

5.57 Software life cycle processes.

5.57.1 Requirements summary. In its foreword, MIL-STD-498 states that it implements the development and documentation life cycle processes of the International Standard, *Information technology - Software life cycle processes*, ISO/IEC 12207. The intent of this statement is to designate MIL-STD-498 as the DoD-approved standard for implementing ISO/IEC 12207.

A comprehensive set of software life cycle processes is described in ISO/IEC 12207. Activities that may be performed during the life cycle of software are grouped into five primary processes, eight supporting processes, and four organizational processes. The software processes in ISO/IEC 12207 cover software from conceptualization of ideas through the retirement of the system. Each primary life cycle process view -- acquirer, supplier, developer, operator, maintainer -- specifies the activities applicable to each of those key players.

MIL-STD-498 is the DoD's standard for software development and documentation. MIL-STD-498 defines a set of activities and documentation suitable for the development of both weapon systems and automated information systems. MIL-STD-498 provides requirements for software development activities and software products that result. MIL-STD-498's activities cover the activities and tasks in the primary process, Development, of ISO/IEC 12207 and the supporting and organizational processes needed to support this primary process. It also covers many of the activities and tasks in the other four primary processes to some degree. Neither MIL-STD-498 nor ISO/IEC 12207 dictate how to implement or perform the activities and tasks specified.

5.57.2 Acquirer's responsibilities and considerations. The acquirer is responsible for determining which of MIL-STD-498's activities apply to the project. Two views of the activities applicable are provided by the phases and strategies of DoDI 5000.2 and DoDI 8120.2. MIL-STD-498's Figures 9-12 show how the activities of the standard can be applied to DoD's program strategies. Appendix G of the standard provides guidance on program strategies, tailoring, and build planning. Paragraphs 4.6 and 4.7 of this guidebook describe DoD's phases and strategies.

5.57.3 Additional things to think about.

Are the development and documentation processes consistent with the related program strategy?
Are acquirer activities scheduled in synchronization with developer activities? User activities? Support activities?

5.57.4 Related guidebook topics.

Software development process

5.58 Software management indicators.

5.58.1 Requirements summary. MIL-STD-498 provides requirements for the developer to use software management indicators to aid in managing the software development and communicating progress to the acquirer. The developer is required to identify and define "a set" of software management indicators, including the data to be collected, the methods to be used to interpret and apply the data, and the planned reporting mechanism. Once plans have been approved, the developer is required to collect, interpret, apply, and report on the set of indicators selected for the project, as described in the developer's SDP. References to ***Software management indicators*** in MIL-STD-498 are listed in Figure 123 with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword-3*, 5.19.2, *Figure 2*, *Appendix F*, *Figures 9-12*
SDP	5.19.2

FIGURE 123. MIL-STD-498's references to ***Software management indicators***.

MIL-STD-498 uses the term "management indicators" rather than "metrics" (terms sometimes used synonymously when referring to software "measurements"), to underline the fact that measurements are to be regarded as indications of the software development effort rather than hard and fast facts.

The standard provides a candidate set of software management indicators for use. Many alternative sets exist. The candidate set of software management indicators is not the required set. The developer is free to propose alternatives. The following candidate indicators are identified in MIL-STD-498 Appendix F:

- Requirements volatility
- Software size
- Software staffing
- Software complexity
- Software progress
- Problem/change report status
- Build release content
- Computer hardware resource utilization
- Milestone performance
- Scrap/rework
- Effect of reuse

Figure 124 lists developer's key activities related to ***Software management indicators***.

Developer's key activities related to ***Software management indicators***	References in MIL-STD-498
Describe the approach to be followed for defining software management indicators, collecting the data, and reporting on them identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software management indicators.	SDP 5.19.2
Identify and define a set of software management indicators, including the data to be collected, the methods to be used to interpret and apply the data, and the planned reporting mechanism. Record the information in the SDP and collect, interpret, apply, and report on those indicators as described in the plan. Consider as candidate management indicators the list in Appendix F.	5.19.2, Appendix F
Meet general requirements and perform integral processes of the standard.	4, 5.14-5,19, Appendix B-G

FIGURE 124. Developer's key activities related to ***Software management indicators***.

5.58.2 Acquirer's responsibilities and considerations. Although much work has been done in the area of metrics and management indicators for software, MIL-STD-498 takes a cautious approach by not requiring "the set" of information to be gathered, but requiring that "a set" be used to aid in managing the software development process and communicating its status to the acquirer. Some reasons for this caution are:

- Lack of correlation of things-that-can-be-measured to resulting software quality or progress on the project (source lines of code and function points can be measured, but what is the relationship of that number to productivity of the development team? Quality of the software developed? Capabilities/functionality provided?)

- Lack of uniformity of scales used to make the measurements (Is a source line of code the same in Ada as in FORTRAN or COBOL? Are Mary's source lines of code the same as James'? Do function points mean the same thing for accounting software as for mission planning or knowledge-based systems? Do results vary over time as people learn how to get the "right answer" for the indicator?)

- Lack of guidance in how the results of measurements should be used and interpreted once made.

Both the acquirer and the developer should be clear about what use is to be made of the management indicator information and how the information is to be interpreted before deciding upon a set of indicators for a project. Measurements calibrated over time by a developer for a specific domain, for a specific methodology, and for a specific language can provide valuable information and help determine the chances for project success. However, it often takes years for a developer to gather this information. Measurement for the sake of measurement can be

costly and ineffective. (Additional information on software management indicators can be found in Practical Software Measure: A Guide to Objective Program Insight.)

There are many things that can be measured. What should be measured is more difficult to determine. Typically an acquirer or developer wants to know three things: (1) will the system give the user what is needed, (2) can the system be developed at a cost the acquirer can afford, and (3) can the system be developed in time. What should be measured to provide these answers? The following two cases illustrate the problem:

Case 1: To answer the question, "Will the system provide the needed capability?" the number of errors in the software is often used to indicate whether the system will perform as required. The number of errors found in code can be measured. But how will the information be used? What do these measurements mean?

If many errors are found in the software, are:

a. Coders doing an inferior job in providing the needed capability?
b. Testers doing a superior job in testing the capabilities?

If few errors are found, are:

a. Coders superb?
b. Testers not testing the right things?

Case 2: Actual project progress over time can be tracked and compared to planned time to answer the question, "Will the system be developed at cost and on schedule or will overruns occur?" But how will progress information be used? What do these measurements mean?

If the project appears to be late, is:

a. The development team not capable?
b. The development team doing "hard" things first and will catch up later?
c. The original estimate low?
d. The scope of the job different from planned?

If the project appears to be on time, is:

a. The development team working unrecorded overtime?
b. The development team doing a shoddy job?
c. The schedule estimate good?
d. The developer doing the "easy" things first to meet schedule?

There is no doubt that "something" is indicated. But caution should be used when interpreting "what" is indicated.

Two goals of management indicators are to aid in (1) managing the software development process and (2) communicating the development status to the acquirer. The software measures to be used should be based on the technical and management objectives and issues of the project. For example, the developer might evaluate a candidate set of indicators based on whether the indicators:

- Are flexible and support the identification and analysis of new issues as the project evolves
- Are consistent with the methods/procedures/tools used for the project
- Will objectively measure the software products and processes
- Can be described so that both the developer and acquirer can understand the measurement approach and software attributes measured
- Can be used consistently across developers, when multiple developers are involved
- Can be mapped to both the software component (i.e., CSCI, software unit) and the development activity (e.g., Software implementation, Performing unit testing, etc.)
- Provide timely information
- Can be applied throughout the project to support planning, development, and support

5.58.3 Additional things to think about.

Has the developer identified what data should be collected in problem/change reports in support of software management indicator activity?
How often should information be provided to the acquirer?
How will the selected management indicators be tracked and reported in an effective manner?
How frequently should the indicators be evaluated for effectiveness?
What automated tools are being used to collect measurement data?
Have any indicators been "standardized" across projects?
Are joint technical and management reviews used to discuss results of analysis of management indicator data?
What will the management indicators be used for?

5.58.4 Related guidebook topics.

Cost estimation
Problem/change report
Process improvement
Risk management

5.59 Software product evaluation.

5.59.1 Requirements summary. MIL-STD-498 provides requirements for the developer to establish a software development process that includes software product evaluation. The developer is required to describe the approach to be followed for software product evaluation in the SDP for the project. The evaluations performed include both in-process and final evaluations for the project. Criteria for performing the evaluation of each product are provided in Figure 6 of the standard. References to ***Software product evaluation*** in MIL-STD-498 are listed in Figure 125, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.39*, 4.1, *Figure 1,* 5.15, 5.16.1, 5.16.3, 5.18.1, *Figure 2*, Figure 3, Appendix D, Figure 6, *Figures 9-12*
SDP	5.15

FIGURE 125. MIL-STD-498's references to ***Software product evaluation***.

The standard provides requirements for the developer to evaluate software products, to prepare records of each software product evaluation, and to maintain those records for the life of the contract. If problems are found in software products under project-level or higher configuration control, the problems are to be handled through the developer's corrective action activity. See this guidebook's topic, *Corrective action*, for a discussion of that activity. For projects that consist of multiple builds, the software products of each build should be evaluated in the context of the objectives of that build.

MIL-STD-498 provides six basic evaluation criteria that apply to nearly all products. These evaluation criteria are listed below in the order listed in Figure 6 of the standard and identified by the applicable letter on the figure. Numbers following each criterion identify the paragraph in the Appendix describing it.

a. Does the software product contain all applicable information in (a specified DID)? (D.4.4)

b. Does the software product fulfill any SOW provisions regarding (the specific software product)? (D.4.10)

c. Does the software product meet the format, markings, etc., specified in the CLIN or CDRL? (D.4.9)

d. Is the content of the software product understandable by the intended audience? (D.4.14)

e. Is the software product internally consistent (i.e., (1) no two statements or representations in a software product contradict one another, (2) a given term, acronym, or abbreviation means the same thing throughout the software product, (3) a given term or concept is referred to by the same name or description throughout the software product)? (D.4.8)

f. Does the software product show evidence of having been developed in accordance with the methods/procedures/tools (approach) described in the software development plan? (D.4.7)

In addition to the basic six criteria, the standard provides eight additional evaluation criteria applicable to selected software products. These additional criteria are:

- Do the descriptions or instructions regarding the final product (manuals, "as built" design, version descriptions) correctly depict the product described? (D.4.1)

- Are test cases, procedures, data, and results adequate? (D.4.2)

- Is the software product consistent with other indicated products (i.e., (1) no statement or representation in one software product contradicts a statement or representation in the other software products, (2) a given term, acronym, or abbreviation means the same thing in all of the software products, (3) a given item or concept is referred to by the same name or description in all of the software products)? (D.4.3)

- Does the software product deal with (i.e., cover) every item in the specified set of items? (D.4.5)

- Based on the knowledge and experience of the evaluator, do the concepts, requirements, design, tests, etc., avoid violating any known principle or lessons learned that would render it impossible to carry out? (D.4.6)

- Based on the knowledge of the evaluator, does the plan represent a reasonable way to carry out the required activities? (D.4.11)

- Do recorded test results show that the item under test either passed all tests the first time or was revised and retested until the tests were passed? (D.4.12)

- Can an objective and feasible test be designed to determine whether each requirement has been met? (D.4.13)

Although the criteria provided in the standard's Appendix D and summarized above are part of a mandatory Appendix, the acquirer may tailor (i.e., add, modify, or delete) these criteria and the developer may use alternate criteria or definitions, if approved by the acquirer. Figure 126 lists developer's key activities related to ***Software product evaluation***.

Developer's key activities related to ***Software product evaluation***	References in MIL-STD-498
Describe the approach to be followed for software product evaluation, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software product evaluation.	SDP 5.15
Perform in-process evaluations of the software products generated. Perform a final evaluation of each deliverable software product before its delivery.	5.15.1, Appendix D
Prepare and maintain records of each software product evaluation. Maintain these records for the life of the contract. Handle problems in software products under project-level or higher configuration control in accordance with paragraph 5.17 of the standard.	5.15.2
Maintain independence in software product evaluation. The persons responsible for evaluating a software product are not to be the same persons who developed the product.	5.15.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 126. Developer's key activities related to ***Software product evaluation***.

5.59.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for both process evaluation and product evaluation. Because of overlaps regarding process and product evaluations in predecessor standards, there has been some confusion about who does them. MIL-STD-498 does not identify who performs any activity; it provides the requirements for the activity. It is up to the developer, when defining the software development process for the project, to determine how the activities will be performed, the sequencing of the activities, when they will be performed, and who will perform them. One example of a possible implementation of MIL-STD-498 requirements for software product evaluation is shown in Figure 127.

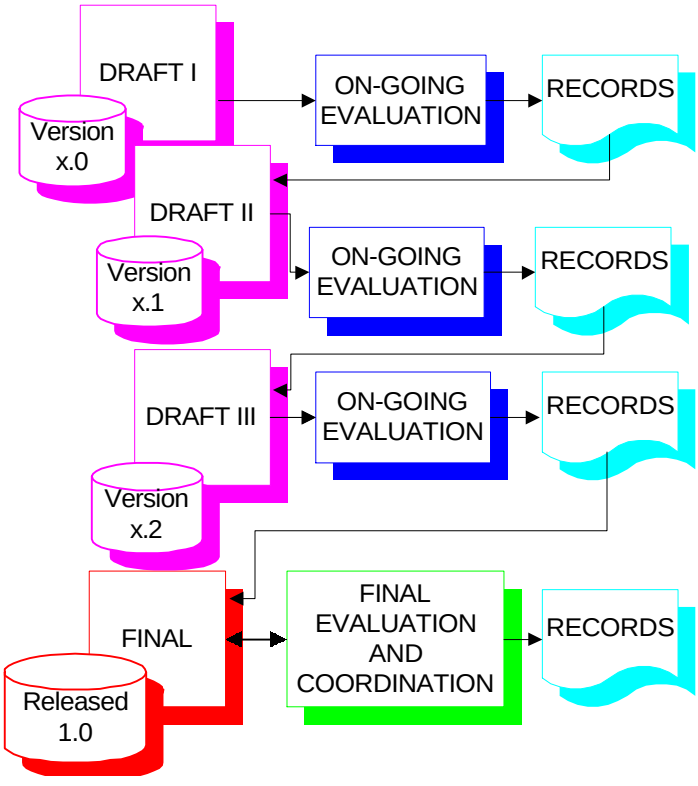


FIGURE 127. Example of one possible implementation of software product evaluation.

MIL-STD-498 carefully words the requirements for software quality assurance to avoid any overlap with software product evaluation. Process evaluation (software quality assurance) is concerned with evaluation of *activities* covering the development of software products. Product evaluation is concerned with evaluation of the *software products* themselves. Software product evaluation covers doing the actual product evaluations, whereas software quality assurance covers assuring that the evaluations have been done and that required software products exist. The standard's requirements for software quality assurance are:

- For *activities* required by the contract or described in the SDP -- the developer is responsible for *assuring* they are being performed in accordance with the contract and with the plan.

- For *software products* required by the contract -- the developer is responsible for *assuring* that they exist and have undergone software product evaluation, testing, and corrective action as required by MIL-STD-498 and the contract.

Software quality assurance records can be examined by an acquirer during the course of the contract for assurance that software product evaluation is being accomplished and software development tasks have been performed in accordance with the contract and software plans. (See this guidebook's topic, *Software quality assurance*, for more information.)

MIL-STD-498 provides requirements to assure that personnel having responsibility for software product evaluation are not those who developed the product. This does not preclude other members of the development team from performing or being responsible for those evaluations. For example, one element of software design might be evaluated by those who developed the software specifications or another element of the design, the code might be evaluated by those who developed the software design or by a coder of a different piece of the software, and software specifications might be evaluated by the project's testers. Alternatively, the developer may select someone from another project or organization, such as software quality assurance organization, to perform these evaluations. MIL-STD-498 neither requires nor precludes a separate "independent" product evaluation organization, but rather allows the developer to select those most qualified to do the job.

MIL-STD-498's software product evaluation requirements are requirements on the developer. The standard's evaluation criteria are also meant for acquirer use if evaluations are performed by the acquirer, his authorized representative, or others, such as IV&V agents. Joint technical and management review discussions can be more productive when evaluation criteria have been agreed upon and used by all. (See this guidebook's topics, *Independent verification and validation* and *Joint technical and management reviews*, for more information.) Figure 30 of this guidebook's topic, *Contract*, provides guidance on consideration of acquirer organizations' participation in software product evaluations.

A significant problem in specifying criteria for software product evaluations is that some of the most important criteria are subjective. For example, a requirements 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. MIL-STD-498 handles this problem by including subjective criteria in Figure 6 and providing definitions for subjective criteria in paragraph D.4. Because of the subjectivity of many of the criteria, the requirement is not to prove that a software product meets the criteria but to use the criteria as a basis for discussion of potential problems. This explicit handling of subjective criteria provides more meaningful software product evaluations and the ability to surface risks associated with those criteria at joint technical and management reviews and to resolve them.

5.59.3 Additional things to think about.

How can the acquirer be assured that the software product is acceptable?
Are evaluations being performed in a timely fashion?
Are records being kept properly and in a timely fashion?
When should the acquirer be concerned about too many problems? When about too few?
Are those performing software product evaluation qualified?

5.59.4 Related guidebook topics.

Contract
Corrective action
Documentation (Recording information)
Independent verification and validation
Joint technical and management reviews
Software development process
Software quality assurance

5.60 Software quality assurance.

5.60.1 Requirements summary. MIL-STD-498 provides requirements for the developer to conduct ongoing evaluations of software development activities and the resulting software products to assure that:

- Each activity required by the contract or described in the SDP is being performed in accordance with the contract and the SDP.
- Each software product required by MIL-STD-498 or by other contract provisions exists and has undergone software product evaluations, testing, and corrective action as required by MIL-STD-498 and by other contract provisions.

This activity is called software quality assurance. References to ***Software quality assurance*** in MIL-STD-498 are listed in Figure 128, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.1, *Figure 1, 5.1.1 Note 3*, 5.16, *Figure 2*, Figure 3, Figure 6, *Figures 9-12*
SDP	5.16

FIGURE 128. MIL-STD-498's references to ***Software quality assurance***.

The developer is required to describe the approach to accomplishing this activity in the SDP for the project and cover:

- Evaluations
- Records of those evaluations
- Independence

Software quality assurance activities are employed throughout the duration of the contract. If a system/subsystem or CSCI is developed in multiple builds, software quality assurance activities in each build are understood to take place in the context of the objectives established for that build. Appendix B of the standard shows how software quality assurance activities should be interpreted for reusable products. Figure 129 lists developer's key activities related to ***Software quality assurance***.

Developer's key activities related to ***Software quality assurance***	References in MIL-STD-498
Describe the approach to be followed for software quality assurance, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software quality assurance.	SDP 5.16
Assure that each activity required by the contract or described in the SDP is being performed in accordance with the contract and the SDP. Assure that each software product required by MIL-STD-498 or by other contract provisions exists and has undergone software product evaluations, testing, and corrective action as required by this standard and by other contract provisions.	5.16.1
Prepare and maintain records of each software quality assurance evaluation for the life of the contract. Handle problems in software products under project-level or higher configuration control and problems in activities as described in MIL-STD-498, paragraph 5.17, Corrective Action.	5.16.2
Identify those responsible for conducting software quality assurance evaluations.	5.16.3
Apply software quality assurance to all software activities performed and products prepared, modified, or used in incorporating reusable software.	Figure 3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 129. Developer's key activities related to ***Software quality assurance***.

MIL-STD-498's requirements for software quality assurance include requirements for independence in process evaluations. Those *responsible* for performing software quality assurance evaluations are not to have developed the product or performed the activity, or have been responsible for the software product or activity. Those who developed the product, performed the activity, or were responsible for the software product or activity may *take part in* these evaluations. The person(s) responsible for assuring compliance with the contract are to be provided the resources, responsibility, authority, and organizational freedom to permit objective software quality assurance evaluations and to initiate and verify corrective actions.

5.60.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for the developer to evaluate both the software development activities for the project and the resulting products to ensure that the methods/procedures/tools used are generating the required information and products. The criteria for evaluation of products are found in Appendix D of MIL-STD-498. (See this guidebook's topic, *Software product evaluation,* for more information.) The standard takes the position that evaluations of the software products under development are performed as one step of the development of that software product and that quality is "built into" software by the developers rather than "added on" at a later date by others.

MIL-STD-498's requirements for process evaluations are found in the standard's software quality assurance activity. Criteria for evaluation of the software development process are that each required activity is being performed in accordance (1) with the contract and (2) with the software development plan. Process evaluators determine whether the required products have been developed, whether information has been recorded, and whether products have undergone software product evaluations, testing, and corrective actions, as required. Process evaluations are intended to determine whether product evaluations have been performed, not necessarily to perform those product evaluations. However, this does not preclude those performing process evaluations from performing product evaluations as well, or from examining products to determine that processes are working as planned.

MIL-STD-498's activity requirements and independence requirements are not intended to be interpreted as organizational requirements. That is, the standard does not require a separate software quality assurance department/organization, nor a configuration management department/organization, nor a testing or coding organization, etc. The developer is free to assign people and organizations in whatever way works for that developer within independence constraints as follows:

- The persons responsible for conducting software quality assurance evaluations are not to be the persons who developed the software product, performed the activity, or were responsible for the software product or activity. That is, the person who developed the code for a particular piece of software cannot be the one responsible for the evaluations of the process used to develop that code, or responsible for the code or coding activity (such as a team leader or project manager). However, the coder could perform evaluations of the coding methods/procedures/tools used, develop problem/change reports, make process improvement recommendations, etc.

- The persons responsible for assuring compliance with the contract are to have the resources, responsibility, authority, and organizational freedom to permit objective software quality assurance evaluations and to initiate and verify corrective actions. That is, those persons given the responsibility for making evaluations of the process are not to be ignored or deprived of resources needed to perform the evaluations and are to have a path open to the appropriate level of management to assure that products developed are in compliance with contractual obligations.

    - For example, those who developed the detailed design for a CSCI might perform evaluations of the methods/procedures/tools used for testing that CSCI. These evaluations might reveal that the testing process did not include development of developer-internal tests for behavioral design. The designers performing the evaluation could report that process deficiency (problem) through the corrective action process. Those responsible for assuring compliance with the contract would monitor the corrective action process to determine whether the problem was acted upon and the testing process corrected. If the problem was not corrected, those responsible for assuring compliance could elevate the problem to a level within the project or the

developer's organization that could ensure that the problem was corrected. Although the question could be asked, "What if the developer doesn't want to comply with the contract?" the standard assumes that the developer wants to comply. The standard is written as a two party contract; both parties are expected to comply.

Software quality assurance records are intended to provide evidence that evaluations have been performed. Software quality assurance records can be reviewed by the acquirer. Problems found in processes and products may be one subject for discussion at joint technical and management reviews.

Software quality assurance should not be confused with software testing or IV&V. Although a Software Quality Assurance Organization may exist within a company that is tasked with performing those additional activities, the software quality assurance activity of MIL-STD-498 covers process evaluations only.

5.60.3 Additional things to think about.

Are developer software quality assurance activities adequately funded?
Are process evaluations performed in a timely fashion?
Are software quality assurance records properly maintained?
Are software quality assurance records available to the acquirer or the acquirer's authorized representative(s)?
Does the developer's SDP show how independence is maintained where needed?
Are the software quality assurance processes evaluated? By whom?

5.60.4 Related guidebook topics.

Access for acquirer review
Corrective action
Independent verification and validation
Joint technical and management reviews
Oversight
Software product evaluation
Testing (Developer-internal)

5.61 Software support.

5.61.1 Requirements summary. MIL-STD-498 provides requirements regarding planning and preparing for software support. The standard defines software support as "*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*." The standard's requirements for planning and preparing for software transition are intended to provide the information and products needed by the support agency to plan for software support, and to modify, enhance, correct and otherwise support the deliverable software once transition has occurred. References to ***Software support*** in MIL-STD-498 are listed in Figure 130, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 3, 1.2.1, 1.2.4.3, 3.12, 3.33, 3.35, 3.41*, *3.44, 3.46*, 4.2.6, 5.1.5, 5.2.3, 5.2.5, 5.13, *6.2, E.4.10*, *Figure 2*, Figure 3, *G.6.1, Figures 9-12, Figure 13, Figure 14*
All DIDs	1.2
OCD	3.5, 5.5, 7
SDP	4.2.6, 5.1.5, 5.2.3, 5.2.5, 5.13
SPS	5
SRS	3.13, 3.15
SSS	3.13, 3.15
STrP	All

FIGURE 130. MIL-STD-498's references to ***Software support***.

All of the standard's activities and software products are intended for use during support of the software when modifications, enhancements, and corrections, or any changes (such as reengineering or redocumentation) are needed. When the software changes, the software products that describe that software may also need to be updated, revised, or prepared anew. Figure 131 lists developer's key activities related to ***Software support***.

Developer's key activities related to ***Software support***.	References in MIL-STD-498
Describe the approach to be followed for software support, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the description of software support.	SDP 4.2.6, 5.1.5, 5.2.3, 5.2.5, 5.13
Plan for transition to the specified software support agency.	5.1.5 STrP
Record key decision rationale that will be useful for the support agency and describe the approach for recording this information. Define what is meant by a "key decision" (what types of decisions are to be included, level of detail, etc.). Record rationale that will be useful to the support agency for key decisions made in specifying, designing, implementing, and testing the software in documents, code comments, or other media that will transition to the support agency. Include trade-offs considered, analysis methods, and criteria used to make the decisions.	4.2.6
Prepare the executable software, source files, and version descriptions to be transitioned to the support site.	5.13.1, 5.13.2, 5.13.3 SPS 3.1, 3.2, 4 SVD 3
Prepare the "as built" CSCI design and related information, including methods to be used to demonstrate that a given body of software is a valid copy, compilation/build procedures, modification procedures, computer hardware resource utilization, and traceability between CSCI source file and software unit. Update the system design description and prepare required support manuals.	5.13.4, 5.13.5, 5.13.6 CPM DBDD FSM IDD SDD SPS 5.1, 5.2, 5.3 5.4, 6 SSDD
Transition the software to the support site: install and check out the deliverable software in the support environment, demonstrate to the acquirer that the deliverable software can be regenerated and maintained, provide training to the support agency as specified in the contract, provide other assistance to the support agency as specified in the contract.	5.13.7 SPS 5.1, 5.2, 5.3, 5.4
Provide overviews of the support concept for both the current system and new or modified system, including, as applicable, support agency(ies); facilities; equipment; support software; repair/replacement criteria; maintenance levels and cycles; and storage, distribution, and supply methods. Provide anticipated operational impacts, organizational impacts, and developmental impacts on the support agency(ies) brought about by the development of the new or modified system.	OCD 3.5, 5.5, 7

FIGURE 131. Developer's key activities related to ***Software support***.

Developer's key activities related to ***Software support***.	References in MIL-STD-498
Specify the system/subsystem and CSCI requirements, if any, concerned with logistics considerations, such as personnel requirements and training, system maintenance, software support, system transportation modes, supply-system requirements, impact on existing facilities, and impact on existing equipment.	SRS 3.13, 3.15 SSS 3.13, 3.15
Ensure acquirer or support agency has or can obtain any non-deliverable software necessary for support of deliverable software.	5.2.5
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 131. Developer's key activities related to ***Software support*** - *(continued)*.

5.61.2 Acquirer's responsibilities and considerations. A key decision that the acquirer should make early is the support concept for the software. This decision involves determining who should be responsible for modifying, upgrading, correcting and otherwise changing the software. One option is for the developer to continue to support the software throughout the life of the system; another is to transition support to a different contractor or a government support agency; yet another is to determine there will be no support. To facilitate development decisions, the standard calls for this information to be defined early and recorded in the Operational Concept Description (OCD) for the system. (Note that the OCD may be prepared either before or after the System/Subsystem Specification (SSS) for the project. See this guidebook's topic, *Operational concept,* for more information.)

When the acquirer determines that the subsequent support of the system/software will be handled by a government support agency, the developer can consider the designated support agency's existing facilities, equipment, and support software when selecting the processes and tools for the Software Development Environment (SDE). The developer may also factor in repair/replacement criteria, maintenance levels and cycles, storage, distribution, and supply methods to be used to support that system/software.

Information in the OCD, SDP, and Software Transition Plan (STrP) can provide valuable information that may influence the acquirer's Computer Resources Life Cycle Management Plan (CRLCMP) for the system/software as well as Integrated Logistic Support Plans (ILSPs), the Software Installation Plan (SIP), and other fielding plans for the system. Plans should be prepared as early as possible in the project, reviewed and validated often, and updated as necessary to reflect changes in requirements, schedule, or approach. (See this guidebook's topic, *Software development planning*, for more information.) When support and user personnel are aware of the planned changes, they can prepare and reduce the impacts of the

changes. Planning can help reduce impacts that might occur to operations from changes in such things as:

- Interfaces between computer operating centers
- Operational procedures
- Use of new data sources
- Quantity, type, and timing of data to be input to the system
- Data retention requirements
- New modes of operation based on peacetime, alert, wartime, or emergency conditions

In addition to information in the software products listed above, software support information can be found in the software products developed according to the standard's DIDs. For example, the Software Product Specification (SPS) is designed to provide a summary of the history of the maintenance of the software. The SPS is the principal source of support information. It contains source and executable software, packaging and marking requirements, validation methods, "as built" design, compilation/build procedures, modification procedures, descriptions of computer hardware resource utilization, and traceability information. (See this guidebook's topics, *Software support manuals* and *Software transition*, for more information). Another DID, the OCD requires overviews of the support concept for both the current (old) system and the new or modified system, a description of impacts on the support agency(ies), and an overview of the system. Rationale for key decisions that would be of value to the support agency is called for as a general requirement affecting all information generated during the development phase.

Other key sources of information needed for support are:

- Software Version Description (SVD) -- identifies and describes a software version including materials released, inventory of software contents, changes installed, adaptation data, related documents, installation instructions, and possible problems and known errors
- Software User Manual (SUM) (and possibly other user's manuals) -- tells a hands-on user how to install and use the software
- System/Subsystem Design Description (SSDD) -- describes the system/subsystem-wide design and architectural design of a system or subsystem

5.61.3 Additional things to think about.

Have software support and other logistic support requirements been specified? If a CRLCMP already exists, have decisions regarding support been communicated to the developer?
Are the support agencies aware of the role planned for them in support of the system/software? Are they participating in planning activities? Is support planning covered in the CRLCMP?
Have resources needed for support been planned in long-range funding profiles and been included in affordability assessments?
Does the support agency have the necessary subject matter expertise as well as the software expertise to make changes, corrections, or enhancements to the software? To test the software?
If the software development processes and tools require the support team to learn new technologies or methodologies, has adequate time been allowed for the support team to come up to speed?
Have those who will have responsibility for software support been involved with defining acceptance criteria? Reviewing designs? Determining if requirements have been met?
Have cost models been prepared for projected life cycle costs?
If the support agency is other than the developer, what plans are there to transition open problem reports?
Is there any advice or are there lessons learned regarding software configuration management that the acquirer should pass on to the software support activity?
Have arrangements been made with the developer to provide ongoing, as needed, or emergency support to the support agency, if required, after transition?

5.61.4 Related guidebook topics.

Operational concept
Rationale/key decisions
Software development library
Software development planning
Software support manuals
Software transition

5.62 Software support manuals.

5.62.1 Requirements summary. MIL-STD-498 provides requirements for development of two manuals as part of preparing for software transition, the Computer Programming Manual (CPM) and the Firmware Support Manual (FSM). References to ***Software support manuals*** in MIL-STD-498 are listed in Figure 132, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.39*, 5.13.6, *6.2*, Figure 4, Figure 6, *Figures 9-12, Figure 16, Figure 17, E.4.10*
CPM	All
FSM	All
SDP	5.13.6

FIGURE 132. MIL-STD-498's references to ***Software support manuals***.

The detailed content and format for these manuals are described in the CPM DID and the FSM DID. The two support manuals are intended to cover the following:

- *Computer Programming Manual*. The CPM provides information needed by a programmer to understand how to program a given computer. It focuses on the computer itself, not on the particular software that will run on the computer. It is intended for newly developed computers, special-purpose computers, or other computers for which commercial or other operation manuals are not readily available.

- *Firmware Support Manual*. The FSM provides the information needed to program and reprogram the firmware devices of a system. It applies to read-only memories (ROMs), programmable ROMs (PROMs), erasable PROMs (EPROMs), and other firmware devices. The FSM 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.

The developer is required to describe the approach to be followed in development of the manuals and the methods to be used to evaluate and control the manuals in the SDP for the project. Figure 133 lists developer's key activities related to ***Software support manuals***.

Developer's key activities related to ***Software support manuals***	References in MIL-STD-498
Describe the approach to be followed for developing software support manuals, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to preparing software support manuals.	SDP 5.13.6
Identify and record information needed to program the computers on which the software was developed or on which it will run.	5.13.6.1 CPM
Identify and record information needed to program and reprogram any firmware devices in which the software will be installed.	5.13.6.2 FSM
Identify documentation needed to support hardware and software.	STrP 3.2, 3.3, 3.4
Perform corrective action. Include "manuals" as one of the categories used in classifying problems in software products.	Figure 4
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 133. Developer's key activities related to ***Software support manuals***.

5.62.2 Acquirer's responsibilities and considerations. Not all systems will need software support manuals. The intent is for the acquirer, with input from the developer, to determine which manuals are appropriate for a given system and require only those manuals. These two manuals are frequently developed commercially and usually need to be prepared "from scratch" only when custom computers or firmware devices are developed for the project. When the CPM and FSM do need to be delivered, the acquirer should list the CPM and the FSM DID each on a CLIN or CDRL. (See this guidebook's topic, *Contract data requirements list*, for a sample CDRL, DD Form 1423.)

The support manuals listed here, the CPM and the FSM, are specialized support manuals. The document containing the principal information required by the support agency for supporting the software is the Software Product Specification (SPS). The SPS provides for:

- The executable software
- The source files
- Packaging and marking requirements
- Methods to be used to demonstrate that the software is a valid copy
- The "as built" CSCI design and related information
- Compilation/build procedures

- Modification procedures
- Computer hardware resource utilization
- Traceability between CSCI source file and software unit, and between computer hardware resource utilization measurement and CSCI utilization requirements

Note that the Software Transition Plan (STrP) requires the identification of manuals associated with the hardware and software needed to support the deliverable software. These manuals may be commercial manuals or ones developed for the project or by the developer. The acquirer will want support agency personnel to review the information in software support manuals to be certain that it is clearly presented and understandable by those who will be responsible for updating, modifying, correcting, and otherwise supporting the software after transition to the support agency. (See this guidebook's topic, *Software support*, for more information.)

5.62.3 Additional things to think about.

Will manuals developed for custom-made computers be available in time to support developer's programming and firmware development activities?
If commercial manuals are provided, do they contain all needed information? If not, where will the supplemental information be recorded?
Are costs for commercially provided manuals appropriate?
If commercial manuals are provided, are there data rights limitations on copying and distributing the manuals? If so, can additional manuals be purchased? Can a license for multiple copies be obtained?
Are there commercial manuals that can be substituted for support manuals?
Have user manuals and other information about the software and hardware in the deliverable software support environment been identified?
Who should review the support manuals?

5.62.4 Related guidebook topics.

Contract data requirements list
Installation (Support environment)
Software support
Software transition
Software user manuals

5.63 Software test environment.

5.63.1 Requirements summary. MIL-STD-498 provides requirements for establishing a software test environment and defines the software test environment (STE) as the facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification, and possibly other, testing of software. References to ***Software test environment*** in MIL-STD-498 are listed in Figure 134, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.2, 3.37, 3.43*, 4.2.7, 5.2.2, 5.2.3, *E.4.7*, Figure 3, *Figure 13*
SDP	4.2.7, 5.2.2
STP	3
STR	3.2

FIGURE 134. MIL-STD-498's references to ***Software test environment***.

The STE is a subset of the software development environment (SDE). The standard makes a distinction between the STE and the software engineering environment (SEE). The SEE is the part of the SDE that contains the elements needed to perform all activities except for qualification testing. The distinction between the two is made to facilitate tailoring to allow the acquirer to task someone other than the developer to perform qualification testing. Figure 104 of this guidebook's topic, *Software development environment*, lists examples of the typical components of the SEE and the STE.

Elements of the STE may be unique to the STE or be a part of both the STE and the SEE. The SDL, which is used to facilitate the orderly development and subsequent support of software, may be an integral part of both environments. The deliverable software in the STE is subject to the requirements of MIL-STD-498 to the extent specified in the contract.

The developer is required to describe the approach to be followed for this activity in the SDP for the project. Figure 2 of the standard lists other standardization documents related to software testing. Figure 135 lists developer's key activities related to ***Software test environment***.

Developer's key activities related to ***Software test environment***	References in MIL-STD-498
Describe the approach to be followed for the software test environment, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the software test environment.	SDP 4.2.7, 5.2.2
Describe the software test environment at each intended test site.	STP 3
Establish, control, and maintain a software test environment to perform qualification, and possibly other, testing of software. Ensure that each element of the STE performs its intended function.	5.2.2
If there are open issues regarding the status of the STE, consider holding, as a joint management review, a test readiness review.	E.4.7
Provide the acquirer access to developer facilities for review of the STE.	4.2.7
Provide an assessment of the manner in which the test environment may be different from the operational environment and the effect of this difference on the qualification test results.	STR 3.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 135. Developer's key activities related to ***Software test environment***.

5.63.2 Acquirer's responsibilities and considerations. See this guidebook's topics, *Software development environment, Software engineering environment, Qualification testing*, and *Testing (Developer-internal)*, for more information.

5.63.3 Additional things to think about.

Does the designated software support environment include all necessary elements of the STE?
Are STE tool acquisition, modification, development, integration, and demonstration events on schedule? Have dry runs been performed to assure readiness and availability for qualification testing?
Has the developer ensured that the acquirer has or can obtain all non-deliverable STE software that is needed for the support of the deliverable software?
Is there a conflict between the availability of SEE and STE resources?

5.63.4 Related guidebook topics.

Qualification testing
Software configuration management
Software development environment
Software development library
Software engineering environment
Software support
Testing (Developer-internal)

5.64 Software transition.

5.64.1 Requirements summary. MIL-STD-498 provides requirements regarding software transition. The standard defines software transition as "*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*." MIL-STD-498 identifies "preparing for software transition" as one of the activities that a developer may be required to include as part of the defined software development process for a project. References to ***Software transition*** in MIL-STD-498 are listed in Figure 136, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.44, 3.47*, 4.1, 4.2.6, *Figure 1*, 5.1.5, 5.13, 5.14.5, *6.2*, Figure 3, D.3, Figure 6, *E.4.1*, *Figures 9-12*, *Figures 15-17*
CPM	All
FSM	All
SDP	5.1.5, 5.13
SPS	All
STrP	All
SVD	All

FIGURE 136. MIL-STD-498's references to ***Software transition***.

The developer is required to describe the approach to be used to plan for software transition in the SDP for the project and identify risks/uncertainties associated with software transition and plans to handle them. When software is to transition to a support environment, the developer is required to develop and record information regarding that transition in the Software Transition Plan (STrP). (See this guidebook's topic, *Software support*, for more information.) The STrP DID requires the developer to:

- Identify all resources that will be needed for the support of the deliverable software including the facilities, hardware, software, personnel, and any other resources needed and the relationships among those resources
- Describe any procedures, advice, and lessons learned that the developer may wish to recommend to the support agency for supporting the deliverables
- Describe the developer's plans for training support personnel
- Describe anticipated areas of change to the deliverable software
- Describe the plans for transitioning the deliverable software

Figure 137 lists developer's key activities related to ***Software transition***.

Developer's key activities related to ***Software transition***	References in MIL-STD-498
Describe the approach to be followed for software transition, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses regarding software transition.	SDP 5.1.5, 5.13
Identify all software development resources that will be needed by the support agency to fulfill the support concept specified in the contract. Develop and record plans identifying these resources. Describe the approach to be followed for transitioning deliverable items.	5.1.5 STrP
Establish and implement procedures for the packaging, storage, handling, and delivery of deliverable products and specify packaging and marking requirements in the SPS.	5.14.5 SPS 3.3
Prepare the executable software, source files, and version descriptions to be transitioned to the support site.	5.13.1, 5.13.2, 5.13.3 SPS 3.1, 3.2, 4, SVD 3
Prepare the "as built" CSCI design and related information, including methods to be used to demonstrate that a given body of software is a valid copy, compilation/build procedures, modification procedures, computer hardware resource utilization, and traceability between CSCI source file and software unit. Update the system design description and prepare required support manuals.	5.13.4, 5.13.5, 5.13.6 CPM DBDD FSM IDD SDD SPS 5.1, 5.2, 5.3, 5.4, 6 SSDD
Transition the software to the support site: install and check out the deliverable software in the support environment, demonstrate to the acquirer that the deliverable software can be regenerated and maintained, provide training to the support agency as specified in the contract, provide other assistance to the support agency as specified in the contract.	5.13.7 SPS 5.1, 5.2, 5.3, 5.4
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 137. Developer's key activities related to ***Software transition***.

5.64.2 Acquirer's responsibilities and considerations. A key decision that the acquirer should make early is the support concept for the software. This decision involves determining who should be responsible for modifying, upgrading, correcting and otherwise supporting the software. One option is for the developer to continue to support the software throughout the life of the system; another is to transition to a third party who might be a contractor or a government support agency; another is to determine that no support will be provided.

Although some of the support activities apply regardless of whether or not support transfers from the original team (e.g., update the design descriptions), many do not. For example, installation in a new support environment would probably not occur if support remains with the original developer.

The STrP should serve as a working tool for communicating project activities and up-to-date schedules to individuals that will be involved with or possibly impacted by the transition. The STrP can provide valuable information for use in preparing or updating the acquirer's Computer Resources Life Cycle Management Plan (CRLCMP). Software transition planning should take place as early as possible in the project, reviewed and validated often, and updated as necessary to reflect changes in requirements, schedule, or approach.

Once the support concept has been established and other program strategies developed, the acquirer decisions that depend upon both can be made. For example, when both the support concept and the program strategy are determined, the acquirer can decide what software, if any, is to be transitioned to the support agency in each build and plan for that transition. Another decision that can be made is to identify the information that should be prepared for delivery to the support agency and whether that information should be in the form of traditional documents or another form, such as CASE tools or databases.

The acquirer and the developer along with the designated support agency, if any, will decide which support manuals should be prepared. Note that not all support manuals apply to all projects. (See this guidebook's topic, *Software support manuals*, for more information.)

5.64.3 Additional things to think about.

What is the support concept for this system?
Is the support agency for the system different from the support agency for the software? If so, how will the two (or more) agencies interface?
Will the support agency require contractor support?
Will the support agency have the necessary qualified personnel and facilities available on a timely basis to support transition as it is scheduled?
Has the development team assessed the risk of including COTS software in the deliverable software? If unknown problems develop, can the support agency rely on the vendor to fix them?
Have COTS products been modified? What licensing/warranty provisions apply to modified COTS (or portions thereof)? To unmodified COTS?
Have users, support agency and other key personnel (such as security and safety personnel) been kept involved and informed regarding transition activities?
What are the criteria for successful transition to the support agency?
What information regarding the user site(s) does the support agency need?
Are there commercial manuals that can be substituted for support manuals?
Have user manuals and other information about the software and hardware in the deliverable software support environment been identified?
Have all software and hardware components that are to transition to the support agency been identified and controlled?

5.64.4 Related guidebook topics.

Commercial-off-the-shelf software products
Installation (Support environment)
Risk management
Schedules
Software development planning
Software support
Software support manuals

5.65 Software unit.

5.65.1 Requirements summary. MIL-STD-498 provides requirements for designing, implementing and testing software units. The standard defines software unit as:

> "*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*."

This term, software unit, purposely overloaded in the standard to meet the varied needs of the wide-range of design and implementation methods and languages, serves as a "placeholder" for design method or language-unique terms the developer selects to represent the design and implementation for the project. "Software unit" needs to be defined by the developer in the context of the methods/procedures/tools used on the project. The developer is required to describe or reference these methods/procedures/tools and explain the meaning of the standard's software units in the terms of the appropriate methods, conventions, standards, and notations selected for use in the SDP for the project. References to ***Software unit*** in MIL-STD-498 are listed in Figure 138, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.24, 3.45*, 4.2.3.1, *Figure 1*, 5.2.4, 5.6, 5.7, 5.8, 5.13.4, *Figure 2,* B.4, Figure 3, Figure 6, *F.3, Figures 9-12, G.6.4, Figures 14, 15, and 17*
DBDD	3, 4, 4.x, 5, 5.x, 6
IDD	3, 3.1, 3.x, 4
SDD	3, 4, 4.1, 4.2, 4.3, 4.3.1, 4.3.x, 5, 5.x, 6
SDP	4.2.2, 4.2.3.1, 5.2.4, 5.6, 5.7, 5.8, 5.13.4
SPS	5.1, 5.4, 6
SRS	3.12

FIGURE 138. MIL-STD-498's references to ***Software unit***.

Figure 139 lists developer's key activities related to ***Software unit***.

Developer's key activities related to ***Software unit***	References in MIL-STD-498
Describe the approach to be followed for all activities involving software units, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software unit.	SDP 4.2.2, 5.2.4, 5.6, 5.7, 5.8, 5.13.4
Identify and evaluate reusable software products for use in fulfilling the requirements of the contract.	4.2.3.1
Establish, control, and maintain an SDF for each software unit or logically related group of software units, for each CSCI, and, as applicable, for logical groups of CSCIs, for subsystems, and for the overall system. Record information about the development of the software in the appropriate SDFs and maintain the SDFs for the duration of the contract.	5.2.4
Define and record CSCI-wide design decisions affecting the selection and design of the software units comprising the CSCI, the architectural design of each CSCI (identifying the software units comprising the CSCI, their interfaces, and a concept of execution among them) and traceability between the software units and the CSCI requirements. Include all applicable items in the SDD, IDD, and DBDD.	5.6 DBDD 3, 4, 4.x, 6 IDD 3, 3.1, 3.x, 4 SDD 3, 4, 4.1, 4.2, 4.3, 4.3.1, 4.3.x, 6
Develop, record and test software corresponding to each software unit in the CSCI design. Include coding computer instructions and data definitions, building databases, populating databases, and other data files with data values, and other activities needed to implement the design. Obtain acquirer approval for any programming language not specified in the contract. Establish test cases, test procedures, and test data for testing the software corresponding to each software unit. Cover all aspects of the unit's detailed design and record in the appropriate SDFs. Include all applicable items in the SDD, IDD, and DBDD.	5.7 DBDD 5, 5.x IDD 3, 3.1, 3.x, 4 SDD 5, 5.x
Perform unit integration and testing. Integrate the software corresponding to two or more software units, testing the resulting software to ensure that it works, until all software in each CSCI is integrated and tested. Establish test cases, test procedures, and test data for conducting unit integration and testing. Cover all aspects of the CSCI-wide and CSCI architectural design and record in the appropriate SDFs. Revise and retest as needed, updating other software products. Analyze and record the results.	5.8
Update the "as built" design and define and record the information needed to support the software. Define and record the traceability between the CSCI's source files and software units.	5.13.4 SPS 5.1, 6
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 139. Developer's key activities related to ***Software unit***.

MIL-STD-498 provides requirements for the design and implementation of software and for the evaluation of that design and implementation, along with evaluations of the tests of the software units. Evaluation criteria is provided in MIL-STD-498 Figure 6. These criteria include:

- Do the design, implementation, and tests meet the SOW provisions regarding SDF's?
- Are the design, implementation, and tests of the software understandable by the intended audience (e.g., programmer-to-programmer information need not be understandable by testers)?
- Are the design and implementation internally consistent?
- Do the design, implementation and tests follow the SDP?
- Is the design consistent with CSCI requirements?
- Is the design feasible?
- Is detailed design consistent with CSCI-wide design decisions?
- Does the implemented software cover the CSCI detailed design?
- Are the software units and the files in which they reside identified?
- Do the source files cover all units in the CSCI design?
- Are the versions that passed testing exactly those of the source files?
- Are the contents of the SDFs containing information about the software units up-to-date?
- Are the unit and unit integration test cases/procedures/data/results adequate?

5.65.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for the developer to present the various components of the software architecture according to the terminology of the selected design method and implementation language. Software units may be made up of other software units. Software units may be organized into as many levels or views as needed. To describe the software's design and implementation, the standard allows the developer to substitute the terminology appropriate to the project's methods/procedures/tools and standards. The developer describes the design method(s) or language-unique terms used that represent "software units" in the SDP for the project.

MIL-STD-498's terminology for describing the software entities or components (i.e., software unit) was selected to accommodate object oriented methods and Ada for which the Computer Software Component (CSC)/Computer Software Unit (CSU) scheme of DOD-STD-2167A did

not work well. Developers whose methods do work with a CSC/CSU structure can continue to use them. Note that software units in the design do not have to have a one-to-one correspondence with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

5.65.3 Additional things to think about.

Do the developer's methods/procedures/tools and standards clearly identify software units?
Is the developer's approach for selecting and describing software units provided in the SDP?
Should COTS software be defined as a separate "unit?" As a separate CSCI? As a subsystem?
Is the partitioning of software units compatible with the development approach?

5.65.4 Related guidebook topics.

Architectural design
Builds
Database design
Detailed design
Interfaces
Software development environment
Software development files
Software engineering environment
Source files
Testing (Developer-internal)
Traceability

5.66 Software user manuals.

5.66.1 Requirements summary. MIL-STD-498 provides requirements for the development of four manuals as part of preparing for software use: the Computer Operation Manual (COM), Software Center Operator Manual (SCOM), Software Input/Output Manual (SIOM), and Software User Manual (SUM). References to ***Software user manuals*** in MIL-STD-498 are listed in Figure 140, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.1.4, 5.12.3, *6.2*, *Figure 2*, Figure 3, Figure 4, Figure 6, *E.4.9, Figures 9-12, Figures 15-17*
COM	All
SCOM	All
SDP	5.12.3
SIOM	All
SIP	3.5, 4.x.6, 5.x.2, 5.x.3
STrP	3.2, 3.3, 3.4
SUM	All

FIGURE 140. MIL-STD-498's references to ***Software user manuals***.

The detailed content and format for these manuals are described in the DIDs for those manuals. The four user manuals are intended to cover different groups of users.

- *Software User Manual*. Explains to a hands-on software user how to install and use a CSCI, a group of related CSCIs, or a software system or subsystem. It may also cover a particular aspect of software operation, such as instructions for a particular user position or task. The SUM is 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.

- *Software Input/Output Manual*. Explains to 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. The SIOM is 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. This manual is often used with the SCOM.

- *Software Center Operator Manual*. This manual provides information to personnel in a computer center or other centralized or networked software installation on how to install and operate a software system. The SCOM is 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. This manual is often used with the SIOM.

- *Computer Operation Manual*. This manual provides information needed to operate a given computer and its peripheral equipment. The COM focuses on the computer itself, not on a particular software program that will run on the computer. It is intended for newly developed computers, special-purpose computers, or other computers for which commercial or other operation manuals are not readily available. The COM includes information on power-on and power-off, initiation, and shut down; input/output media and procedures needed for read/write; problem-handling; and diagnostics.

The developer is required to describe the approach to be followed in development of the manuals as well as the methods to be used to evaluate and control the manuals in the SDP for the project. Figure 141 lists developer's key activities related to ***Software user manuals***.

Developer's key activities related to ***Software user manuals***	References in MIL-STD-498
Describe the approach to be followed for developing software user manuals, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to software user manuals.	SDP 5.12.3
Identify and record information needed by hands-on users (persons who will both operate the software and make use of its results).	5.12.3.1 SUM
Identify and record information needed by persons who will submit inputs to, and receive outputs from, the software, relying on others to operate the software in a computer center or other centralized or networked software installation.	5.12.3.2 SIOM
Identify and record information needed by persons who will operate the software in a computer center or other centralized or networked software installation, so that it can be used by others.	5.12.3.3 SCOM
Identify and record information needed to operate the computers on which the software will run.	5.12.3.4 COM
Identify documentation needed to support hardware and software.	STrP 3.2, 3.3, 3.4
Perform corrective action. Include "manuals" as one of the categories used in classifying problems in software products.	Figure 4
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 141. Developer's key activities related to ***Software user manuals***.

5.66.2 Acquirer's responsibilities and considerations. Not all systems will need all software user manuals. The intent is for the acquirer, with input from the developer, users, and the support agency, to determine which manuals are appropriate for a given system and require only those manuals. For example, the SIOM is usually developed when users must access software that runs remotely, such as on a mainframe in a computer center or other centralized or networked software installation, with those users accessing the system via terminals or personal computers or submitting and receiving inputs and outputs in batch mode. The SCOM is the user manual for those in the computer center who will install and operate that software. The two often go together and may be prepared instead of a SUM. A software system usually would not require all three.

For some systems, especially when COTS software products have been used, it may be appropriate for the developer to provide both the user manual that was prepared by the vendor of the COTS product (such as word processing, spreadsheet, or database user manuals) as well as user manuals specifically prepared to support the application for which the COTS has been used. For example, financial applications are often "built" on some combination of interfaced COTS spreadsheet and database software products. The COTS software provides the basic capability. A developer might create customized screens for user input, tailored queries, and special reports. A user might, therefore, need the vendor's user manuals to provide information on specific basic capabilities as well as developer generated user manuals explaining how to input customer data, query the software for customer specific information, and generate reports in customer format. The developer can determine whether documentation requirements can be satisfied wholly or in part by commercially available manuals and provide or supplement those manuals as needed. Other information commonly needed when developer-customized systems are built on COTS is that regarding utility programs that might be needed to update, delete, or modify data, generate routine reports, update transactions, or do routine "housekeeping" on the system.

User manuals are often routinely identified as a needed deliverable. However, when help files or manuals can be accessed on-line and printed if needed, it may not be necessary to order hardcopy documents, especially for easy-to-install software in a commonly used environment (such as software running under Windows on a PC). Such on-line help can be specified as one of the capabilities the software is to provide. In that case, a CLIN or CDRL will not be needed, but the capability will need to be specified as a requirement in the System/Subsystem Specification (SSS) or Software Requirements Specification (SRS), as appropriate. If the acquirers and users want hardcopy or softcopy documents delivered, however, a CLIN or CDRL should be prepared for each and included in the contract. (See this guidebook's topic, *Contract data requirements list,* for more information.)

Users, support agency, specifiers, and testers of the system/software should all be involved in the evaluation of user manuals, especially of commercial documentation used as a substitute for a developer's own manuals, to ensure that information needed to operate the software is useful and understandable. It is often worthwhile for someone other than the developer of the system, the software, or the hardware on which it runs, to try to operate the system/software using only the manuals provided. Important, obvious information, such as how to terminate a job or exit an application, can be inadvertently overlooked by those familiar with the system.

User manual information is frequently some of the earliest information prepared in development of a system. When the specifiers and designers create information for system/subsystem-wide and CSCI-wide design of the system/software capabilities, describing from the user perspective how the system/software will appear to behave, much of the information necessary to prepare user manuals is available and can be provided to users for evaluation. (Such information is often used by testers to prepare test cases and procedures to validate the software capabilities, as well.)

When software is part of a larger hardware-software system to be operated in the field, it is not unusual for a DoD service or government agency to have specific format, structure, and other requirements for technical manuals (i.e., CALS compliant, part of Technical Order xxx, etc.) that are part of the overall contract requirements for the system. The acquirer should determine whether the information called for in the software user manuals is duplicative to that called for by other user documentation identified or whether the information is to be provided as input to those documents rather than used as stand-alone manuals.

5.66.3 Additional things to think about.

Do the selected user manual(s) fulfill a user's need?
Will the manuals be ready when they are needed? For use? For training? For testing?
If commercial manuals are provided, do they contain all needed information? If not, where will the supplemental information be recorded?
If commercial manuals are provided, are there data rights limitations on copying and distributing the manuals? If so, can additional manuals be purchased? Can a license for multiple copies be obtained?
Should the user manuals be on-line rather than or in addition to paper copies? Are there alternative presentations, such as checklist command strips, that may be attached to the computer or given to the operator?
Is user information "positional" (i.e., different operators have different responsibilities) so that a user manual for one user position should be different from another user position?
Are there users or others in the acquirer's group who can review preliminary drafts of the manuals?

5.66.4 Related guidebook topics.

Contract data requirements list
Installation (User site(s))
Software development environment
Software product evaluation
Software support manuals

5.67 Source files.

5.67.1 Requirements summary. MIL-STD-498 provides requirements for the development, packaging, and delivery of source files. MIL-STD-498 requires the developer to implement each software unit's design and test the executable software generated by that implementation. Software implementation may include activities other than coding computer instructions and data definitions to create source code, such as building databases and populating databases and other data files. The software implementation activity of the standard also includes tracing between each CSCI source file to the software unit(s) that it implements and from each software unit to the source files that implement it. Note that 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. Additional information regarding the development of source code is discussed in this guidebook's topic, *Software implementation and unit testing*. References to ***Source files*** in MIL-STD-498 are listed in Figure 142, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.29, 3.39*, 4.2.2, 4.2.3.1, 4.2.6, 5.7.1, 5.12.1, 5.13.2, 5.13.3, 5.13.4, B.3, Figure 3, Figure 4, Figure 6 (Item 24), *F.3.b*
SDP	4.2.2, 4.2.3.1, 5.7.1, 5.13.2, 5.13.3, 5.13.4
SPS	3.2, 4, 5.1, 5.2, 5.3, 6.a, 6.b
SVD	3.2

FIGURE 142. MIL-STD-498's references to ***Source files***.

The developer is also required to describe the standards for representing code during software implementation in the SDP for the project. At a minimum, coding standards include: standards for format; standards for header and other comments; naming conventions for variables, parameters, packages, procedures, etc.; restrictions, if any, on the use of programming language constructs or features; and restrictions, if any, on the complexity of code aggregates. The developer must obtain acquirer approval to use any programming language not specified in the contract. Figure 143 lists developer's key activities related to ***Source files***.

Developer's key activities related to ***Source files***	References in MIL-STD-498
Describe the approach to be followed for source files, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to source files.	SDP 5.7.1, 5.13.2, 5.13.3
Develop and apply standards for representing code. Describe or reference the standards to be followed in the SDP.	4.2.2 SDP 4.2.2
Obtain acquirer approval to use any programming language not specified in the contract.	5.7.1
Perform corrective action. Include "code" as one of the categories used for classifying problems in software products.	Figure 4
Apply software management indicators. Consider using software size as a management indicator.	F.3.b
Provide references in the SPS, if applicable, to the "as built" design information provided in source file headers, comments, and code. Identify all compilation/build and modification procedures, software, hardware, data, etc. needed. Define and record the traceability between source files and software units.	5.13.4 SPS 5.1, 5.2, 5.3, 6.a, 6.b
Prepare the source files to be transitioned to the support site. State the method to be used to demonstrate that the software is a valid copy of the CSCI. Deliver source files for the CSCI on electronic media.	5.13.2 SPS 3.2, 4
Identify and record the exact version of the software prepared for the support site. Provide an inventory of software contents for the software version being released. List all source files by identifying numbers, titles, abbreviations, dates, version numbers, and release numbers, as applicable.	5.13.3 SVD 3.2
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 143. Developer's key activities related to ***Source files***.

Source code may be one of the products reused on a project. The standard requires the developer to evaluate reusable software products for use in fulfilling contract requirements, possibly taking into consideration the availability and quality of source files. If source code is reused, Figure 3 of the standard interprets the applicability of the standard's activities to the reused code. The interpretation makes allowances to reduce the amount of work to be performed. For example, the levels of testing (unit, CSCI, etc.) necessary for reusable software depend on whether the software will be used without modification and whether there is a positive performance record for the software. (See this guidebook's topic, *Reusable software products*, for more information.) The intent of these requirements is to ensure that the source code, whether developed or reused, is prepared, controlled, evaluated, and delivered following a well-defined process.

5.67.2 Acquirer's responsibilities and considerations. An acquirer has three primary decisions to make regarding source code:

- What programming language should be required?
- What contractual mechanism should be used for providing delivery of source files?
- What media should the source files be delivered on?

Regarding the first question, MIL-STD-498 has no requirements specifying the programming language in which source code is to be developed. MIL-STD-498 is usable with any programming language. MIL-STD-498 provides requirements for the developer to obtain approval to use any language not specified in the contract. Current DoD policy mandates the Ada programming language for new starts and requires Ada for upgrades if more than 30% of the code is to be modified.

The specification of a source language can be a difficult choice for system enhancements, reengineering efforts, and major system upgrades where existing code is other than Ada. Different DoD services or government agencies may have different policy/guidelines regarding this issue. The acquirer and developer may need to evaluate trade-offs such as:

- Programming new source code using the same language as existing software to maintain consistency of languages versus implementing new source code in Ada, thereby moving to the mandated language but creating a "mixed bag" of languages
- Programming new source code in Ada and translating existing code to Ada avoiding the "mixed bag" versus the costs and risks of reengineering software that works "as is"

MIL-STD-498 provides for two possible options regarding the second question. There are two conventional ways to require delivery of source files: (1) via a CLIN, and (2) via the CDRL. When ordering source files via a CLIN, the contracting office will create a CLIN that states that source files are to be delivered and specify the media on which they are to be delivered. When ordering source files via the CDRL, the contracting office will create a CDRL item listing the Software Product Specification (SPS) as a deliverable item citing the SPS DID for content and format. Note that the SPS calls for delivery of executable software, source files, and other pertinent information needed by the support organization to modify, maintain, and otherwise support the software. The CDRL should also state what media the source files are to be delivered on. Source code listings (paper printouts) are <u>not</u> required unless specified by the acquirer. The developer is required to identify the software, hardware, compilation/build and modification procedures needed to read the source files and to make modifications.

The third question usually depends on the support concept selected. If the acquirer knows that the support hardware on which the source files will be maintained is different from the development hardware, the acquirer should state what media will be required to prevent delays in transition at the support site. The standard's "shell" requirement in the paragraph regarding transition to the designated support site alerts the acquirer to identify the software and hardware on which the software is to be maintained. Without such information, the selected support hardware may have drives different from the developer's with incompatible formats, or drives may be of incompatible size or density. Unless the acquirer specifies the media to provide, the developer will select the media for delivery.

5.67.3 Additional things to think about.

If the source code is not available for any COTS and/or NDI that will be used in the system, have provisions, such as escrow accounts, been provided to ensure supportability?
If source code is not available for any COTS/NDI in the software engineering and test environments (e.g., compilers, CASE tools, etc.) that will be needed for support, have provisions (such as escrow accounts) been provided to ensure access to the source code so that changes/corrections can be made if the vendor will no longer support the COTS/NDI?
Can the executable version that passed qualification testing be regenerated from the version of source code that was delivered?
Are adequate controls in place to ensure the integrity between the source files, build procedures, and identification/accounting methods?
Has population of databases been addressed adequately in the SDP?
If reusable software is to be used, has the developer obtained the appropriate licenses and data rights?
If redevelopment is required, has the support agency been contacted regarding their possible requirements for coding standards or development tools?
Has the developer adequately planned for the delivery of source files, licenses, and data rights to the support agency?

5.67.4 Related guidebook topics.

Contract data requirements list
Executable software
Programming languages
Reusable software products
Risk management
Software implementation and unit testing
Software transition
Standards for software products

5.68 Standards for software products.

5.68.1 Requirements summary. MIL-STD-498 provides requirements for the developer to develop and apply standards for representing requirements, design, code, test cases, test procedures, and test results, and to present or reference any conventions needed for understanding them. The standards are described in the SDP for the project and cover all contractual clauses concerning standards for software products. The SDP allows the developer flexibility in selecting "where" to record these standards. Reference may be made to: (1) other paragraphs in the SDP if the standards are better described in context with the activities to which they will be applied, or (2) to other documents containing descriptions of the standards, such as company or organizational policies and procedures. References to ***Standards for software products*** in MIL-STD-498 are listed in Figure 144, with non-mandatory references indicated with italicized text.

MIL-STD-498	4.2.2, 5.1.5, D.4.7, *G.6.2.b*
DBDD	3, 4, 5
IDD	3
SDD	3, 4, 5
SDP	4.2.2
SPS	5.3
SSDD	3, 4

FIGURE 144. MIL-STD-498's references to ***Standards for software products***.

MIL-STD-498 does not specify the information to be included for representing requirements, design, test cases, test procedures, and test results, but does specify requirements for describing the standards used for code. Standards for representing code are required for each programming language to be used. Coding standards are to describe, as a minimum:

- Standards for code format (such as indentation, spacing, capitalization, and order of information)
- Standards for header comments (requiring, for example, name or identifier of code; version identification; modification history; purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.)
- Standards for other comments (such as required number and content expectations)

- Naming conventions for variables, parameters, packages, procedures, files, etc.
- Restrictions, if any, on the use of programming language constructs or features
- Restrictions, if any, on the complexity of code aggregates

The developer is required to use the standards described in the SDP during software development. The standards will also be used during software product evaluations. MIL-STD-498's Appendix D, D.4.7, Follows software development plan, states: "*This criterion means that the software product shows evidence of having been developed in accordance with the approach described in the software development plan. Examples include following design and coding standards described in the plan*."

The developer is also required to include or reference, as part of the modification procedures described in the Software Product Specification (SPS) DID, the design, coding, and other conventions to be followed to modify the software. Figure 145 lists developer's key activities related to ***Standards for software products***.

Developer's key activities related to ***Standards for software products***	References in MIL-STD-498
Describe the approach to be followed for standards for software products, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to standards for software products.	SDP 4.2.2
Develop and apply standards for representing requirements, design, code, test cases, test procedures, and test results.	4.2.2
Present or reference design conventions needed to understand the design.	DBDD 3, 4, 5 IDD 3 SDD 3, 4, 5 SSDD 3, 4
Describe the procedures needed to modify the CSCI, including the design, coding, and other conventions to be followed.	SPS 5.3
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 145. Developer's key activities related to ***Standards for software products***.

5.68.2 Acquirer's responsibilities and considerations. The developer's standards for representing requirements, design, code, and test information affect the development costs and supportability of the software products that are produced. Vague, unclear, or conflicting standards can result in software that is difficult to develop and evaluate, and can increase the cost of software support as well.

The developer's standards for representing requirements, design, code, and test information should be understandable by the intended audience, have sufficient detail, and be consistent with the methods/procedures/tools used. If CASE or other automated tools are used, the format for the representation of the information may be determined by those automated tools. The developer's SDP will describe the tools and standards used so the acquirer can determine whether the code that will be generated using those tools and standards meets the acquirer and/or support agency needs. Note that for programs with multiple associate contractors, it may be beneficial for the acquirer to specify the standards to be used for the project in Requests For Proposal (RFP). The acquirer's trade-off considerations for specifying standards across developers includes whether it is more beneficial to force developer(s) to use a uniform set of standards, possibly requiring a developer to change the methods/procedures/tools familiar to personnel versus having a consistent "look and feel" to the software products to aid reviewers and support personnel.

5.68.3 Additional things to think about.

Did the support agency provide input to the requirements for standards for software products?
For programs involving many associate contractors, is it beneficial to have program-wide standards? If so, how will the standards be selected?
If the developer references internal company standards in the SDP, can timely access to review and evaluate those standards be provided?
Are the developer's standards for software products consistent with the development processes and methods/procedures/tools described in the SDP?
Should legacy code be revised to conform to the project's coding conventions?

5.68.4 Related guidebook topics.

Programming languages
Software development environment
Software engineering environment
Software test environment
Source files

5.69 Statement of work.

5.69.1 Requirements summary. MIL-STD-498 does not provide requirements regarding how a Statement of Work (SOW) is to look or be prepared. The standard is intended to be used as part of or referenced from the SOW to provide requirements for developing software and documents associated with that development. The standard describes the SOW as the list of tasks to be performed by the developer. The SOW is a key part of a contract. This guidebook's topics, *Contract* and *Requirements of the standard*, provide additional information. The SOW is that portion of the contract that specifies the tasks the developer is to perform to fulfill the contract's requirements. Figure 6 of this guidebook places the SOW in relationship to players and agreements assumed by MIL-STD-498. References to the ***Statement of work*** in MIL-STD-498 are listed in Figure 146, with non-mandatory references indicated with italicized text.

MIL-STD-498	*1.2.1, 6.5, 6.7,* Figure 6, D.4.5, D.4.10, *G.6.3*

FIGURE 146. MIL-STD-498's references to ***Statement of work***.

The SOW describes the contractual requirements with which the developer's services and products must comply. The developer is to define, plan and establish a software development process that meets the requirements in the SOW. If the developer has subcontracted for software development, the applicable provisions of the prime contract's SOW are to be "flowed down," as necessary, to the subcontract SOW.

5.69.2 Acquirer's responsibilities and considerations. An acquirer can use the SOW to put MIL-STD-498 on contract. Other options for applying MIL-STD-498 are described in section 4.11 of this guidebook. Prior to citing the standard in the SOW, the acquirer will select the appropriate activities to be performed for the project (i.e., tailor the standard) and the software products that are to be delivered (i.e., tailor the DIDs). (See the MIL-STD-498 Overview and Tailoring Guidebook for detailed information on tailoring.) Tailoring decisions regarding the activities (tasks) to be performed are recorded in the SOW and those regarding the software products on the CLIN or CDRL.

MIL-STD-498 does not require that a SOW be used to require tasks to be performed and has no requirements regarding how the SOW is to look or what other items are to be attached to a contract. The SOW is one of several instruments that may be used by an organization to describe the tasks that are to be performed by another. When the developer is a contractor, the contract containing a SOW is the usual instrument. When the developer is another DoD service or government agency or another organization within a specific DoD service or

government agency, such as a support center or lab, the instrument may have another name such as Memorandum of Agreement, Memorandum of Understanding, etc., with an attached Task List or other such description of the work to be performed. The term, Statement of Work, is the term used in the standard to refer to all such lists of tasks, but is intended to be translated appropriately when other instruments are used.

5.69.3 Additional things to think about.

Has the standard been tailored for the project prior to citing in the SOW?
Are other standards applicable to this project (see Figure 2 of MIL-STD-498) that should be invoked via the SOW?
Does the SOW clearly identify the tasks to be performed?
How can the acquirer ensure that developer input is considered prior to developing the SOW tasking?
Have assumptions been documented and validated with the developer?
Where, besides the SOW, should the acquirer record the rationale for tailoring decisions that may be of benefit to future contracts?

5.69.4 Related guidebook topics.

Contract
Contract data requirements list
Requirements of the standard
Subcontractor management

5.70 Subcontractor management.

5.70.1 Requirements summary. MIL-STD-498 provides requirements regarding subcontractor management. The standard can be applied to contractors, subcontractors, or government in-house agencies performing software development. When MIL-STD-498 is invoked in a two-party agreement between an acquirer and a developer, it is usually cited on a contract. MIL-STD-498 defines "contract" as an agreement between the acquirer and developer. When the developer in turn becomes an acquirer of another developer's products for a project, the contract between them is referred to as a "subcontract" and the second developer becomes a subcontractor to the first developer (sometimes called the "prime" contractor to more clearly distinguish between them). References to ***Subcontractor management*** in MIL-STD-498 are listed in Figure 147, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 4,* 1.2.1, *3.5*, 4.2.7, *5.4.1 Note*, 5.19.4
SDP	4.2.7, 5.19.4

FIGURE 147. MIL-STD-498's references to ***Subcontractor management***.

Prime contractors are to "flow down" to their subcontractors the applicable MIL-STD-498 requirements necessary to ensure that software products are developed in accordance with prime contract requirements. In addition, the developer is to provide the acquirer or its authorized representative access to developer and subcontractor facilities, including the software engineering and test environments, for review of software products and activities required by the contract. (See this guidebook's topic, *Access for acquirer review*, for more information.) Figure 148 lists developer's key activities related to ***Subcontractor management***.

Developer's key activities related to ***Subcontractor management***	References in MIL-STD-498
Describe the approach to be followed for subcontractor management. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses regarding subcontractor management.	SDP 4.2.7, 5.19.4
Provide the acquirer or its authorized representative access to developer and subcontractor facilities, including the software engineering and test environments, for review of software products and activities required by the contract.	4.2.7
Include in subcontracts all contractual requirements necessary to ensure that software products are developed in accordance with prime contract requirements.	5.19.4

FIGURE 148. Developer's key activities related to ***Subcontractor management***.

5.70.2 Acquirer's responsibilities and considerations. MIL-STD-498 provides requirements for the prime contractor to "flow down" *all necessary* requirements to the subcontractor to meet the prime's contract with an acquirer. The standard's wording is intended to: (1) allow the prime contractor to task someone else to develop specific capabilities, and (2) ensure that all software development products (specifications, designs, etc.), and all processes applicable to the development of those products are included in the subcontract(s). The prime may impose more requirements on the subcontractor than those in the standard or fewer requirements when a subcontractor is performing only a subset of the work to be done. For example a prime may "flow down":

- More requirements -- A prime may impose as requirements on a subcontractor the prime's designs. For example, behavioral design in the prime's System/Subsystem Design Description (SSDD) may be specified as a requirement in a Software Requirements Specification (SRS) provided to a subcontractor. Although these are a prime's developer-internal design decisions and remain so in regard to the acquirer, they are "requirements" on the subcontractor. In this case, the subcontract might require the subcontractor to test and qualify the "design." However, the prime need not include qualification of these "subcontractor requirements" in qualification tests to the acquirer.

- Fewer requirements -- A prime's contract may be to develop an aircraft. A subcontractor may be tasked to code only the software for one "black box" that will be aboard that aircraft with the requirements and design for the software provided by the prime. The prime need not impose its coding standards upon the subcontractor (although it may), as long as the subcontractor's software development procedures cover the requirements for coding standards and, of course, any other requirements (such as language) that may be part of the prime's contract.

The acquirer should review the prime contractor's (developer's) SDP section on subcontractor management to determine the approach to be used for subcontractor management. For example, the SDP will describe the approach used to "flow down" applicable requirements and how acquirer access to subcontractors is provided. Prior to review of subcontractor products in the subcontractor facility, the acquirer or its authorized representatives should coordinate with the prime contractor to ensure that such visits will not adversely impact subcontractor schedules and that the reviewers are fully aware of the scope of the contract and products the subcontractor is developing.

5.70.3 Additional things to think about.

Is the SDP clear regarding the approach to be used to "flow down" requirements to subcontractors? Will all, some, or none be "flowed down"?
Have data rights and licensing issues been addressed in subcontracts?
Is it clear how access to subcontractor facilities will be coordinated? When products will be available for review?

5.70.4 Related guidebook topics.

Acceptance by the acquirer
Access for acquirer review
Contract
Oversight
Requirements
Statement of work

5.71 System/subsystem.

5.71.1 Requirements summary. MIL-STD-498 provides requirements relating to systems/subsystems. If a system consists of subsystems, all requirements in the standard concerning systems apply to the subsystems as well. If a contract is based on alternatives to systems and subsystems, such as a hardware-software item less than a subsystem, the requirements and design activities in the standard concerning the system also apply to those alternatives. System requirements and system design activities are intended to be performed iteratively. Requirement specification and design may be top-down, bottom-up, middle-out, or other method such as structured or object-oriented. The task is to perform system requirements analysis and system design until all HWCIs, CSCIs, and manual operations within the system have been defined. References to ***System/subsystem*** in MIL-STD-498 are listed in Figure 149, with non-mandatory references indicated with italicized text.

MIL-STD-498	*Foreword 3, 1.2.4.1, 1.2.4.2, 3.17, 3.28,* 4.1, 5.1.3, 5.2.4, 5.3, 5.4, 5.5, 5.10, 5.11, 5.13.5, *5.13.6 Note*, *6.2, Figure 2,* Figure 3, Figure 4, Figure 6, *E.4.2-4.4, E.4.7-4.8, Figures 9-12, Figures 15-17*
DBDD	3, 4, 6
IDD	All
IRS	All
SDP	5.1.3, 5.3, 5.4, 5.5, 5.10, 5.11, 5.13.5
SRS	5
SSDD	All
SSS	All
STD	All
STP	All
STR	All

FIGURE 149. MIL-STD-498's references to ***System/subsystem***.

The standard covers four aspects of system development: system requirements analysis (including development of the operational concept), system design, system integration and testing, and system qualification testing. The term "system," as used in the standard, may mean: (1) a hardware-software system for which the standard covers only the software portion; or (2) a software system for which the standard governs overall development. The standard provides requirements for the developer to "participate in" those activities. For systems involving hardware and software development, "participate in" is interpreted to mean *"take part in, as described in the software development plan."* For software systems,

"participate in" means *"be responsible for."* Figure 150 lists developer's key activities related to ***System/subsystem***.

Developer's key activities related to ***System/subsystem***	References in MIL-STD-498
Describe the approach to be followed for system requirements analysis, system design, CSCI/HWCI integration and testing, and system qualification testing, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to system/subsystem.	SDP 5.1.3, 5.3, 5.4, 5.5, 5.10, 5.11
Participate in system requirements analysis.	5.3 IRS SSS
Participate in system design.	5.4, 5.13.5 DBDD IDD SSDD
Define and record the traceability between the CSCI requirements and the system requirements.	5.5 DBDD 6 IDD 4 SRS 5 SSS 5
Participate in CSCI/HWCI integration and testing.	5.10
Participate in developing and recording plans for conducting system qualification testing. Participate in system qualification testing.	5.1.3, 5.11 STD STP STR
Consider as candidate joint management reviews: operational concept reviews, system/subsystem requirements reviews, system/subsystem design reviews, test readiness reviews, test results reviews, and critical requirement reviews.	E.4.2, E.4.3, E.4.4, E.4.7, E.4.8, E.4.10
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 150. Developer's key activities related to ***System/subsystem***.

MIL-STD-498 provides requirements for defining and recording information about the system/subsystem under development. Requirement, design, and traceability information is defined in accordance with the Operational Concept Description (OCD), System/Subsystem Specification (SSS), and System/Subsystem Design Description (SSDD) DIDs. Information to be defined and recorded about system/subsystem qualification testing is in accordance with the Software Test Plan (STP), the Software Test Description (STD), and the Software Test Report (STR) DIDs. CSCI/HWCI integration and testing information is defined and recorded in the appropriate software development files for the level of testing performed. The developer is required to describe the methods/procedures/tools to be followed and used for the

system/subsystem activities, identifying risks/uncertainties and plans for dealing with them in the SDP for the project. Software plans are to be coordinated with system level plans, such as a developer's System Engineering Management Plan (SEMP), and to cover all contract clauses pertaining to system/subsystem for software.

The standard does not provide requirements for the development of hardware, for requirements associated with specialty engineering disciplines, such as electronics and chemical engineering, logistics, or for other engineering aspects that may be necessary for development and support of a system. The standard does not cover the manual operations associated with system operational aspects other than those associated with the system's software. The standard assumes that other system's aspects are covered by other standards or provided as requirements elsewhere in the contract.

5.71.2 Acquirer's responsibilities and considerations. System level activities are included in MIL-STD-498 for two reasons: 1) when a software-only system is developed, system activities are performed by software developers, and 2) when a hardware-software system is developed, the software developer may need to "participate in" system level activities.

For hardware-software systems, the level of software "participation in" system activities can be a problem. Frequently, software developers have little involvement in system level requirement and design activities. For simple systems where software performs discrete capabilities easily isolated within the system, this may work. For complex systems where software performs capabilities throughout the system, ignoring software doesn't work well. In today's complex systems, software is everywhere. Like pipes carrying water through a plumbing system, software routes signals, messages, and data from one part of a system to another. Failing to take software into account is like failing to take into account how water will get from the street to the faucet in a house. MIL-STD-498's requirements for the software developer to "participate in" the systems activities is intended to ensure that software considerations are addressed when systems are specified and designed to prevent costly rework and other inefficiencies. The provisions for requiring the software developer to participate in system testing activities are to ensure that the software developer is still around when systems/subsystems are integrated and tested, to assist or perform that integration, and fix problems that might arise.

The acquirer is responsible for ensuring that system and software requirements are established and used as a basis for planning, development and qualification testing. The acquirer is also responsible for fleshing out the system/subsystem-related "shell" requirements so that whoever is responsible for preparing the system/subsystem requirements and design has the information needed. Many of MIL-STD-498's "shell" requirements are needed to complete system requirements analysis and design activities for the subsystem(s) and

CSCI(s). Figure 151 lists MIL-STD-498 shell requirements and system level activities affected by those requirements.

Specify "Shell" requirement for:	Needed to:
Data rights	Avoid unnecessary costs that might be incurred if a developer evaluates reusable software whose data rights are not compliant with an acquirer's unspecified needs.
Critical requirements	Specify or design special architectural needs concerning those requirements and plan for coordination of testing activities with special groups, if needed.
Computer hardware resource utilization	Avoid unnecessary costs that might be incurred if a developer incorporates computer resources that are not capable of meeting utilization requirements.
User sites	Specify or design any interfaces among user sites.
User input	Prepare for analysis and coordination activities.
Programming languages	Ensure a complete software development environment and personnel qualified to develop the code.
Witnessing CSCI qualification tests	Determine whether CSCI dry run testing will be needed.
Witnessing system qualification tests	Determine whether system/subsystem dry run testing will be needed.
Support sites	Specify or design any interfaces that may be required among support sites and plan for transition to those sites.
Support concept	Assess amount/level of detail of information needed regarding development; identify necessary hardware/software needed by the support agency; and prepare installation plans, including long-lead items, such as facilities, that should be budgeted for early.
Security and privacy	Plan the needed controls for the facilities, such as TEMPEST shielding.

FIGURE 151. Shell requirements needed for system activities.

MIL-STD-498's system requirements, design, and qualification testing DIDs are intended to support both the developer and acquirer in coming to agreements on the requirements for the system. These DIDs provide a checklist of information to be recorded, as applicable, regarding system/subsystem requirements, design, and testing. Not all of the information in these DIDs is applicable to every system. The developer is responsible for providing all of the information applicable to the project unless the acquirer has specifically tailored something out, in essence saying, "Don't do this."

During system requirements definition, the acquirer will determine what characteristics of the system are conditions of acceptance. The acquirer may develop a system specification, may task a systems engineering contractor to develop specifications, or may task the developer to

define and record the requirements. When the acquirer or party other than the developer defines requirements, requirements often are included in the Request For Proposal (RFP) and become part of a contract with a developer. When a developer defines the requirements as part of an overall development project, the acquirer usually will want the developer to provide a copy(ies) of the specification for the acquirer to review and agree upon. An acquirer often takes configuration control of this specification and requires a formal request for change to any of the specified characteristics.

The SSS lists eighteen categories of requirements to be addressed. (See this guidebook's topic, *Requirements,* for additional information.) During system/subsystem design, the developer will define and record the system-wide design decisions, that is, decisions about how the system will behave, from a user's point of view, in meeting its requirements and other discussions affecting the selections and design of system components. System design also includes architectural design, that is, the structure of the system, identifying and describing its components, their interfaces, and a concept of execution among them. (See this guidebook's topics, *Architectural design, Behavioral design, Interfaces*, and *System/subsystem-wide and CSCI-wide design*, for more information.)

MIL-STD-498's suite of test DIDs, (STP, STD, and STR) are meant to be used at any level within the system/subsystem for qualification testing and for CSCI qualification testing. Test cases covering requirements tested at a lower level, such as CSCI, may be reused at the subsystem and system levels as appropriate to qualify a requirement.

The acquirer may need to form or participate in various cross-functional working groups such as the Computer Resource Working Group (CRWG), the Test Planning Working Group (TPWG), the Interface Control Working Group (ICWG), or other specialty engineering working groups. Plans such as the Computer Resource Life Cycle Management Plan (CRLCMP), acquisition plans, Test and Evaluation Master Plan (TEMP), SEMP, SDP, and STP all need to be coordinated and consistent for the project and, for maximum efficiency, these plans and schedules should be coordinated with plans of various cross-functional working groups. Joint technical and management reviews can provide a forum to discuss issues applicable to the system, resolve problems, evaluate project status, and resolve issues relevant to other working groups.

5.71.3 Additional things to think about.

Are roles and responsibilities clearly defined in system analysis and design activities? Is there adequate systems engineering support for software requirements analysis and design? Is there adequate software engineering support for systems requirements analysis and design?
Have software size, processing throughput, and system/subsystem timing impacts on selected system hardware computer resources been assessed and addressed?
Has the software's approach to controlling system states and modes and achieving the needed levels of reliability and maintainability been accounted for?
Have system-wide standardization concerns, such as software languages, standard operating systems, computers, compilers, instruction set architectures, etc., been addressed?
Are system design decision rationale recorded?
Are joint technical and management reviews scheduled to address system/subsystem issues?
Are software reviews coordinated with system/subsystem reviews to ensure timeliness and consistency in decision-making? Are system/subsystem specifiers/designers included in software reviews? Are software specifiers/designers included in system/subsystem reviews?
Are software specifiers/designers participating in system test activities?
Are CSCI and HWCI/CSCI integration activities consistent with system/subsystem testing plans? Have system/subsystem testing plans and activities been coordinated with acquirer test plans and testing activities?
Are computer hardware resource utilization issues being adequately addressed in system requirements reviews and system design reviews?
Is there sufficient input into systems and computer resource engineering concerning software issues from the user and support agent(s)? From interoperability, security, safety and other certification agents?
Are all needed interests and disciplines represented in the acquirer's staff and the cross-functional working groups for the project (such as the CRWG, TPWG, ICWG, etc.)?

5.71.4 Related guidebook topics.

Architectural design
Behavioral design
Contract
Interfaces
Joint technical and management reviews
Operational concept
Qualification testing
Requirements
System/subsystem-wide and CSCI-wide design
Testing (Developer-internal)
Traceability

5.72 System/subsystem-wide and CSCI-wide design.

5.72.1 Requirements summary. MIL-STD-498 provides requirements regarding design decisions that are made at the system, subsystem, and CSCI level and that ignore internal implementation details of the system or CSCI. System/subsystem-wide design decisions are recorded in System/Subsystem Design Descriptions (SSDDs), CSCI-wide design decisions are recorded in Software Design Descriptions (SDDs), and database-wide design decisions are recorded in Database Design Descriptions (DBDDs). (A database may be a system/subsystem, a CSCI, or a software unit.) References to ***System/subsystem-wide and CSCI-wide design*** are listed in Figure 152, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.6*, 4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.8.1, 5.10.1, 5.13.4, 5.13.5, Figure 3, Figure 6 (Items 7, 10, 25, 26), *E.4.4, E.4.6, Figures 9-12*
DBDD	3
IDD	3
OCD	3.3, 5.3, 6
SDD	3, 4.1.c, 4.3
SDP	4.2.2, 4.2.3.1, 5.4.1, 5.6.1, 5.13.4, 5.13.5
SPS	5.1
SRS	3.12
SSDD	3, 4.1.c, 4.3
SSS	3.12

FIGURE 152. MIL-STD-498's references to ***System/subsystem-wide and CSCI-wide design***.

The three basic elements of design in MIL-STD-498 are: architectural, system/subsystem-wide and CSCI-wide, and detailed design. Behavioral design is one aspect of system/subsystem-wide and CSCI-wide design. (See this guidebook's topics, *Architectural design, Behavioral design*, and *Detailed design*, for more information.) Examples of system/subsystem-wide and CSCI-wide design are: inputs the system, subsystem, or CSCI will accept and outputs it will produce, including interfaces with other systems, configuration items, and users; how the system, subsystem, or CSCI will behave in response to each input and output; how databases/data files will appear to the users; approach to meeting safety, security, and privacy requirements; design and construction choices for hardware or hardware-software systems, such as physical size, color, shape, weight, materials, and markings; and other system/subsystem-wide and CSCI-wide design decisions made in response to requirements, such as the approach to providing flexibility, availability, and

maintainability. The developer is required to describe the methods/procedures/tools, standards, techniques, and conventions to be used in system/subsystem-wide and CSCI-wide design in the SDP for the project. Figure 153 lists developer's key activities related to ***System/subsystem-wide and CSCI-wide design***.

Developer's key activities related to ***System/subsystem-wide and CSCI-wide design***	References in MIL-STD-498
Describe the approach for defining and recording the system/subsystem-wide and CSCI-wide design decisions. Identify risks/uncertainties and plans for dealing with them. Cover all contractual clauses for system/subsystem-wide and CSCI-wide design.	SDP 4.2.2, 5.4.1, 5.6.1, 5.13.4, 5.13.5
Develop and apply standards for representing design. Describe or reference the standards to be followed in the SDP.	4.2.2
Identify, evaluate, and include applicable reusable software products in the software design. Allocate requirements to candidate reusable CSCIs and software units.	4.2.3.1, Figure 3
Describe current and new or modified system or situation, identifying differences associated with different states or modes of operation (regular, maintenance, training, degraded, emergency, alternative-site, wartime, peacetime). Describe one or more operational scenarios that illustrate the role of the new or modified system, its interaction with users, its interface to other systems, and all states or modes identified for the system.	OCD 3.3, 5.3, 6
Participate in defining and recording system/subsystem-wide and database-wide design decisions. Incorporate design constraints specified in the SSS and IRS.	5.4.1 DBDD 3 SSDD 3, 4.1.c, 4.3 SSS 3.12
Define and record CSCI-wide and database-wide design decisions and the traceability between software units and (1) the CSCI requirements and (2) CSCI-wide design decisions. Incorporate design constraints specified in SRSs and IRSs.	5.6.1 DBDD 3, 4 IDD 3 SDD 3, 4.1.c, 4.3 SRS 3.12
Establish test cases, test procedures, and test data for conducting unit integration and testing, covering all aspects of the CSCI-wide design.	5.8.1
Participate in developing and recording test cases, test procedures, and test data for conducting CSCI/HWCI integration and testing, covering all aspects of the system-wide and system architectural design. Record software-related information in SDFs.	5.10.1
Update the design description of each CSCI to match the "as built" software and participate in updating the system design description to match the "as built" system.	5.13.4, 5.13.5 SPS 5.1 SSDD
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 153. Developer's key activities related to ***System/subsystem-wide and CSCI-wide design***.

5.72.2 Acquirer's responsibilities and considerations. System/subsystem-wide and CSCI-wide design does not need to be concerned with internal architectural constraints and does not require the designer to have knowledge of the inner workings of a component. System/subsystem-wide and CSCI-wide design, together with architectural design and, at the CSCI-level, detailed design, provide the design decisions necessary for those responsible for the implementation to do their jobs. Figure 154 shows system/subsystem-wide and CSCI-wide design in relationship to other design elements and requirements.

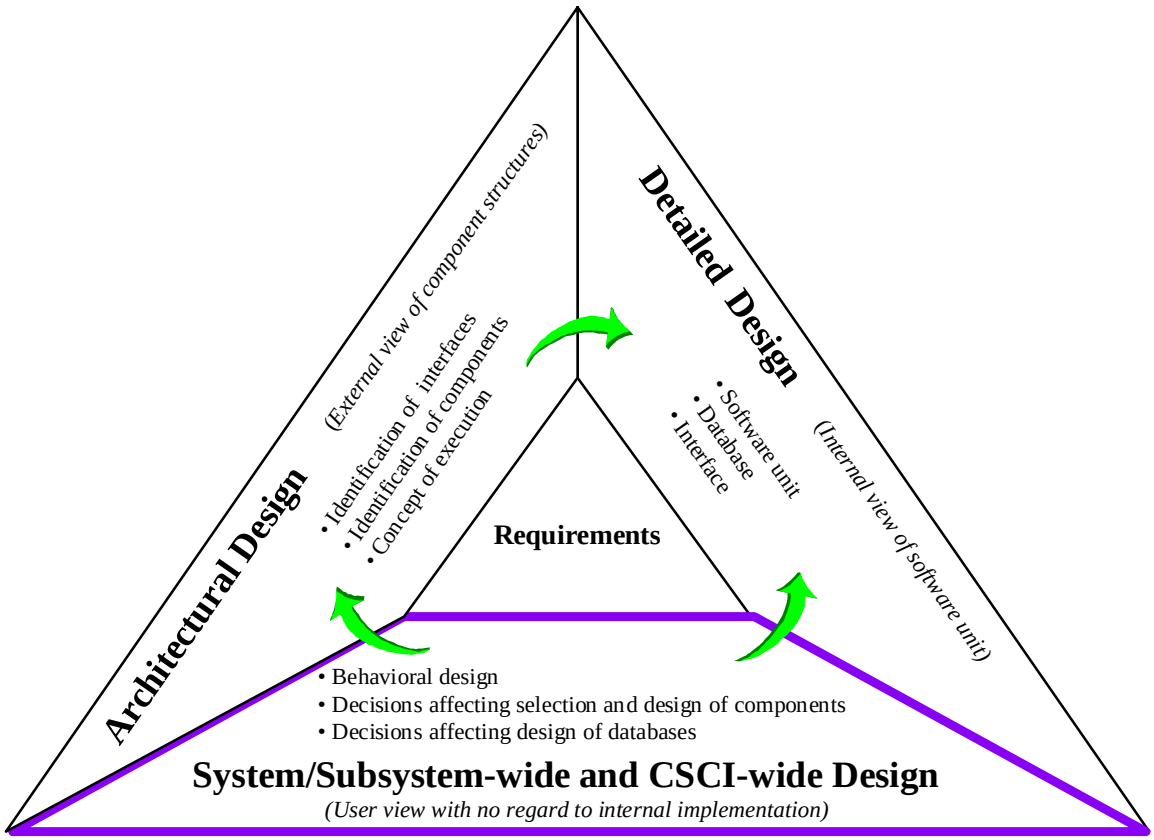


FIGURE 154. System/subsystem-wide and CSCI-wide design in relationship to other design elements.

System/subsystem-wide and CSCI-wide design includes behavioral design. System/subsystem-wide and CSCI-wide design, however, goes beyond how the system/subsystem or CSCI appears to behave from a user's point of view to include other system/subsystem-wide and CSCI-wide design decisions for meeting requirements, such as how a component will meet reliability, availability, flexibility, or safety, security and privacy requirements.

5.72.3 Additional things to think about.

Are system/subsystem-wide and CSCI-wide design included in appropriate joint technical reviews?
Has the acquirer coordinated personnel (operators, maintainers, trainers, security, safety, etc.) and resources to review the system/subsystem-wide and CSCI-wide design?
Are there design standardization issues to be surfaced and resolved?
Have key decisions regarding system/subsystem-wide and CSCI-wide design been recorded?
Have the "as built" system/subsystem-wide and CSCI-wide designs been updated?
Do system/subsystem/CSCI specifications impose unnecessary constraints on the system/subsystem-wide and CSCI-wide design?

5.72.4 Related guidebook topics.

Architectural design
Behavioral design
Detailed design
Interfaces
Joint technical and management reviews

5.73 Testing (Developer-internal).

5.73.1 Requirements summary. MIL-STD-498 provides requirements for the developer to perform both developer-internal and qualification testing. Developer-internal testing is all software testing other than qualification testing. Qualification testing is that testing needed to demonstrate to the acquirer that requirements have been met. (See this guidebook's topic, *Qualification testing,* for more information.) Developer-internal testing, performed at all levels (unit, CSCI, subsystem, and system) is performed to test all aspects of the software, including, but not limited to, system/subsystem-wide and CSCI-wide design, and architectural design. Developer-internal testing includes all types of testing, such as component, integration, black-box, white-box, thread, stress, boundary, performance, path, etc. and includes preparing and conducting tests and retesting, as necessary. References to ***Testing (Developer-internal)*** in MIL-STD-498 are listed in Figure 155, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.34, 3.39, 3.43*, 4.1, *Figure 1*, 5.2.2, 5.2.4, *5.4.1 Note*, 5.7, 5.8, 5.10, 5.16.1, *Figure 2*, Figure 3, Figure 4, Figure 6, *Figures 9-12*
SDP	5.7, 5.8, 5.10

FIGURE 155. MIL-STD-498's references to ***Testing (Developer-internal)***.

To cover systems containing both hardware and software (such as radar systems) and software-only systems (such as payroll systems), the standard includes systems-level activities for developer-internal testing and requires the developer to "participate in" them. For systems involving hardware and software development, "participate in" is interpreted to mean "*take part in, as described in the software development plan*." For software systems, "participate in" means "*be responsible for*."

The planned approach for all levels of developer-internal testing and retesting is described in the SDP for the project. Information regarding those tests is included in SDFs for the project, that is, unit tests in the SDF for that unit, unit integration tests in the SDF for the CSCI (or other identified aggregate) and so on. At each level defined for the project, SDFs are a repository for material pertinent to the development of that body of software. (See this guidebook's topic, *Software development files,* for more information.) Figure 156 lists developer's key activities related to ***Testing (Developer-internal)***.

Developer's key activities related to ***Testing (Developer-internal)***	References in MIL-STD-498
Describe the approach to be followed for developer-internal testing, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to developer-internal testing.	SDP 5.7, 5.8, 5.10
Establish, control, and maintain a software development file (SDF) for each software unit or logically related group of software units, for each CSCI, and, as applicable, for logical groups of CSCIs for subsystems, and for the overall system. Record information about the development of the software in appropriate SDFs and maintain them for the duration of the contract.	5.2.4
Establish unit test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data covering all aspects of the unit's detailed design. Record this information in the appropriate SDFs.	5.7.2
Test the software corresponding to each software unit in accordance with the unit test cases and procedures. Make all necessary revisions, perform all necessary retesting, and update SDFs and other software products as needed, based on the results. Analyze the results of testing and retesting. Record the test and analysis results in appropriate SDFs.	5.7.3, 5.7.4, 5.7.5
Establish unit integration test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data covering all aspects of the CSCI-wide and CSCI architectural design. Record this information in the appropriate SDFs.	5.8.1
Perform unit integration and testing in accordance with the unit integration test cases and procedures. Make all necessary revisions, perform all necessary retesting, and update SDFs and other software products as needed, based on the results. Analyze the results of testing and retesting. Record the test and analysis results in appropriate SDFs.	5.8.2, 5.8.3, 5.8.4
Participate in developing and recording CSCI/HWCI integration test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data. Cover all aspects of the system-wide and system architectural design. Record software-related information in appropriate SDFs.	5.10.1
Participate in CSCI/HWCI testing in accordance with the CSCI/HWCI integration test cases and results. Make all necessary revisions, perform all necessary retesting, and update SDFs and other software products as needed, based on the results. Analyze the results of testing. Record the test and analysis results in appropriate SDFs.	5.10.2, 5.10.3, 5.10.4
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 156. Developer's key activities related to ***Testing (Developer-internal)***.

MIL-STD-498 provides requirements for a developer to evaluate the developer-internal test process. Problems in the developer's internal testing process found as a result of these evaluations or through other means, such as the results of testing and product evaluations, are to be handled in accordance with procedures for the corrective action system described in the SDP. When problems found during evaluations or during developer-internal testing are not with the software itself but with test plans, descriptions, and reports, a special problem category labeled "test information" has been provided for classifying those problems (see Figure 4 of the standard). Software test cases, procedures, and results in SDFs are to be evaluated in accordance with the criteria in Figure 6, Item 29, of MIL-STD-498.

If a system contains reusable software, the standard provides requirements in Figure 3, interpreting the standard's testing requirements for any reusable software incorporated into the system. Appendix B of the standard provides interpretations of activities for reusable software products based on whether the software is being used "as is" or modified, whether the software is a CSCI or software unit, and whether the software has a positive performance record or no or poor performance record.

5.73.2 Acquirer's responsibilities and considerations. Developer-internal testing includes testing the component itself (units, CSCIs, subsystems, or their equivalents) and the aggregates of those components. Integration and testing means aggregating the components, testing those aggregates, and continuing to add components until complete (e.g., CSCI, subsystem, system). Integration testing includes testing the interfaces between each component. Since units may consist of other units, some unit integration and testing may take place during testing of individual units. The last step of integration and testing is developer-internal component testing.

At each level (unit, CSCI, subsystem, system), the software corresponding to two or more components is integrated and tested to ensure that the resulting software works together as intended. Developer-internal testing, concentrating on all aspects of the software to reveal errors and validate correctness, is not to be confused with CSCI qualification testing, whose emphasis is on demonstrating to the acquirer that requirements have been met. Figure 157 shows developer-internal integration and testing activities.

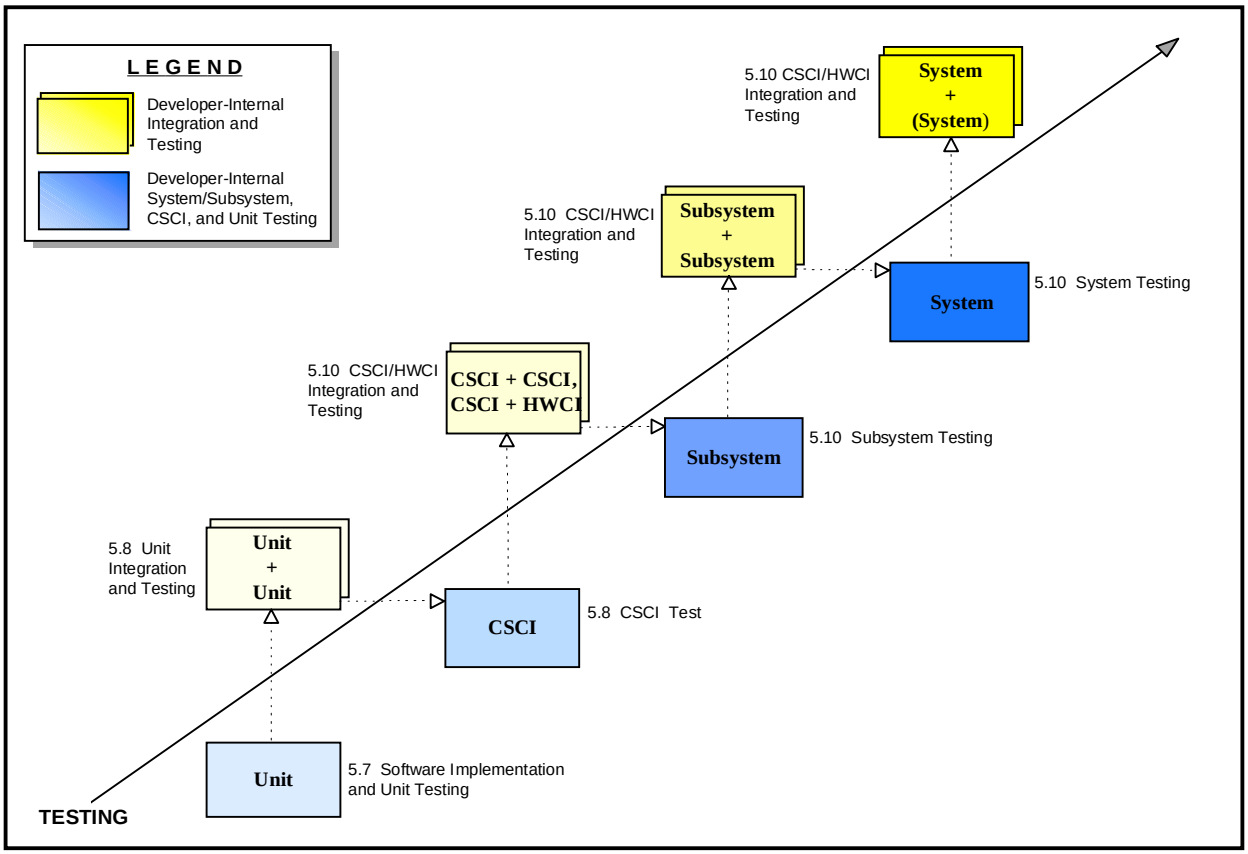


FIGURE 157. Developer-internal integration and testing activities.

Since the primary objective of developer-internal testing is to discover and correct errors, it is highly likely that revisions will be required and retesting performed. The developer's retesting approach will describe the scope and degree of retesting and regression testing to be performed at each level. The approach will also include aspects of retesting, such as how errors found during higher levels of integration will be analyzed and corrected, how the extent of retesting and regression testing is to be determined, and variations allowable in the retesting based on the criticality of the software.

An acquirer may gain insight into the developer-internal testing activities through review of test cases and procedures and information resulting from those tests in SDFs, records of evaluations performed on those SDFs, software management indicators or other measures of the software, and trends. Software management indicators, progress reporting at joint reviews, and process and product evaluation records are all intended to provide the acquirer (and developer) confidence that the project is on track.

5.73.3 Additional things to think about.

Is planning for developer-internal testing and retesting commensurate with software criticality?
Are the standards and procedures for conducting developer-internal testing identified?
Is planning for developer-internal testing and retesting supportive of qualification testing schedules?
Have developer-internal testing schedules included the needed development time for test tools and other items needed to perform developer-internal testing?
Are schedules for developer-internal testing consistent with overall system development schedules, such as hardware development, computer development/acquisition, and vendor-developed operating systems, so that CSCI/HWCI integration testing can occur when needed?
Has acquirer-furnished equipment been identified and plans made to provide that equipment in time for developer-internal preparations for developer-internal testing to be performed?
Does the SDP provide information regarding the types and levels of developer-internal testing to be performed?
If software implements critical requirements, are special developer-internal test cases, procedures, and precautions needed? Are they planned for? If a higher degree of rigor is needed for this type of software, what special considerations and activities are planned?
Does the SDP describe acquirer access to SDFs for review?
Is the degree of rigor in developer-internal testing commensurate with the critical requirements of the system?

5.73.4 Related guidebook topics.

Qualification testing
Software development files
Software development planning
Software implementation and unit testing
Software management indicators
Software quality assurance

5.74 Traceability.

5.74.1 Requirements summary. MIL-STD-498 provides requirements for the following types of traceability:

- Traceability between levels of requirements
- Traceability between requirements and design
- Traceability between design of the software and implementation of the software
- Traceability between requirements and qualification test information
- Traceability between computer hardware resource utilization requirements and computer hardware resource utilization measurements.

References to ***Traceability*** in MIL-STD-498 are listed in Figure 158, with non-mandatory references indicated with italicized text.

MIL-STD-498	5.4.2, 5.5, 5.6.2, 5.9.3, 5.11.3, 5.13.4
DBDD	6
IDD	4
IRS	3, 5
SDD	6
SDP	5.4.2, 5.5, 5.6.2, 5.9.3, 5.11.3, 5.13.4
SPS	5.4, 6
SRS	3, 5
SSDD	5
SSS	3, 5
STD	4.x.y.1, 5
STP	4.2.x.y, 6

FIGURE 158. MIL-STD-498's references to ***Traceability***.

Traceability requirements are intended to promote disciplined software development practices that will result in software that provides all specified requirements. Figure 159 depicts the types of traceability and the source of the relevant requirements within the standard.

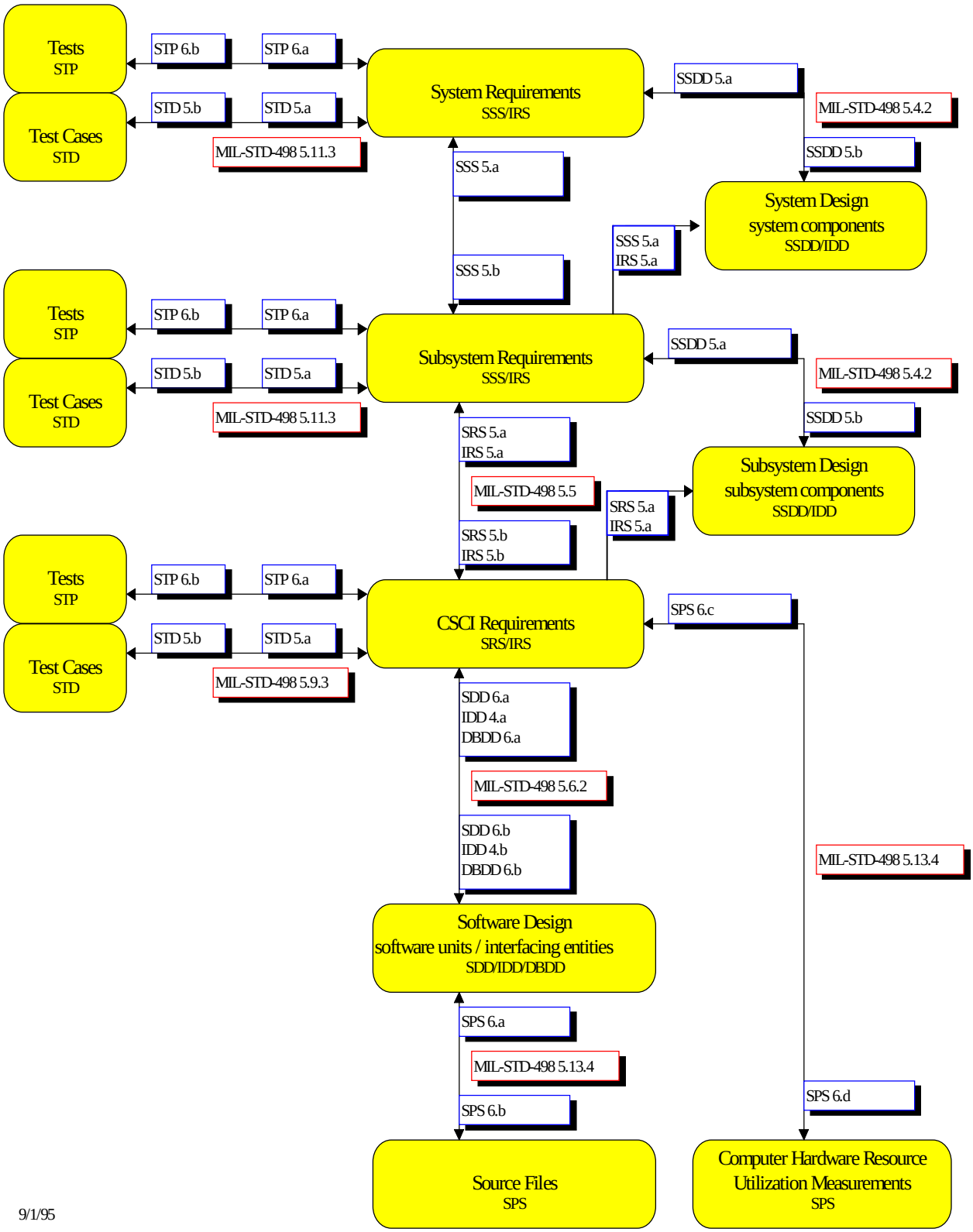


FIGURE 159. MIL-STD-498 traceability requirements.

5.74.2 Acquirer responsibilities and considerations. MIL-STD-498's traceability requirements specify that the contents of a given software product (such as requirements, design components, or test cases) are to be traced to and from higher-level contents of other software products, where a system is higher-level than subsystems, subsystems than CSCIs, requirements than design, design than software, requirements than tests, and tests than test cases. Upward traceability (e.g., design to requirements) covers the link from each lower-level item to the upper-level item(s) that it addresses. Downward traceability (e.g., requirements to design) covers the link from each upper-level item allocated to the lower level, to the lower-level item(s) that address it.

The intent of traceability requirements is to ensure implementation of a disciplined software development process that verifies that all system, subsystem (or their alternatives), and software requirements have been implemented and tested. The standard's requirements to provide a project-unique identifier (number or other tag) for each requirement, rather than each paragraph or sentence, and to provide the method to be used to test that requirement can aid in tracing.

MIL-STD-498's software product descriptions allow a developer the flexibility to select how and where the traceability information is recorded by allowing the information to be recorded in-line with the description of the requirement, or separately. When software product description information resides in other than traditional documents, an acquirer might find all traceability information in one central repository, in on-line software development files, or provided by other automated means. The developer's approach to meeting the traceability requirements is described in the SDP.

Traceability records can also provide information that may be useful to the developer or support personnel in determining the impact that proposed changes will have on the software. By tracing requirement changes through the design to implementation in the code, the dispersion of the requirement throughout the software can be determined. This allows a more accurate estimate to be made of the amount of recoding and retesting required to implement the change.

5.74.3 Additional things to think about.

Has the developer considered using an automated traceability tool?
What is the "lowest-level" software unit in the design to which to trace? Is the software unit too large to provide a meaningful trace? Is the software unit too small to provide a meaningful trace?
Is it clear from the SDP how the traceability between the software unit in the design and the implementation of that design in actual files will be accomplished?
If the software will be transitioned to a support agency, does the support agency have a need to specify a traceability tool or on-line database format to facilitate subsequent software support efforts?
Has each requirement been identified (i.e., each requirement versus each sentence or paragraph) so that traceability is clear? Is interface requirements traceability consistent for multi-contractor developments?

5.74.4 Related guidebook topics.

Critical requirements
Documentation (Recording information)
Independent verification and validation
Qualification testing
Requirements
Safety
Security and privacy
Software configuration management
Software development process
Testing (Developer-internal)

5.75 Version/revision/release.

5.75.1 Requirements summary. MIL-STD-498 provides requirements for identifying the entities that comprise the software. The standard uses the terms "version," "revision," and "release" interchangeably without making a clear distinction among them. The terms are often used inconsistently within the software development industry. MIL-STD-498 leaves the relationship(s) of the terms version/revision/release up to the developer. For example, it may take several versions to reach a build, a build may be released in several parallel versions (to different sites), a build may be a revision of a previous version, etc. References to ***Version/revision/release*** in MIL-STD-498 are listed in Figure 160, with non-mandatory references indicated with italicized text.

MIL-STD-498	*3.7*, 5.12.2, 5.13.3, 5.13.7, 5.14.1, 5.14.2, 5.14.3, B.3, Figure 3, Figure 6, *E.4.9, E.4.10, Figures 9-12*
All DIDs	10.1.c
DBDD	3.d, 5.x.e.2
FSM	3.x.4
IDD	1.1, 3.1
IRS	1.1, 3.1
SDD	4.1, 4.3.1
SDP	4.2.2.b, 5.12.2, 5.13.3, 5.13.7, 5.14.1, 5.14.2, 5.14.3, 5.17.1
SIP	4.x.2
SPS	5.2
SRS	3.3.1, 3.10.3
SSDD	4.1, 4.3.1
SSS	3.3.1, 3.10.3
STR	5.b
STrP	3.2, 3.3, 3.4
SVD	3

FIGURE 160. MIL-STD-498's references to ***Version/revision/release***.

Version/revision/release

Version/revision/release are terms used when keeping track of the software under development and the information regarding that software. MIL-STD-498 provides requirements for the developer to identify entities with a project-unique identifier and to keep track of the entities so that a version/revision/release can be created, tested, and used.

MIL-STD-498's Software Version Description (SVD) DID uses the term "version." A version may be an initial release, a build, a revision, a re-release, etc. The information in an SVD identifies and describes software consisting of one or more CSCIs. The SVD is used to release, track, and control software versions. The SVD is divided into seven parts:

- Inventory of physical media (such as listings, tapes, and disks) and associated documentation that make up the software version being released
- Inventory of all computer files that make up the software version being released
- List of changes incorporated into the software version since the previous version
- All unique-to-site data contained in the software version
- List of all pertinent documents not included in the release
- Installation instructions including: identification of other changes that have to be installed for this version to be used; site-unique adaptation data not included in this version; security, privacy, and safety precautions relevant to the installation; procedures for determining proper installation; and a point of contact for questions or problems
- List identifying possible problems and known errors with steps being taken to resolve the problems or errors and instructions for recognizing, avoiding, correcting, or otherwise handling each one

The SVD DID is intended to be used for the software at whatever level it is released, i.e., system, subsystem, CSCI, or some group of CSCIs. There may be multiple SVDs containing differing information regarding a particular version. For example, the SVD that accompanies the software to a user site may list executable software files and user manuals only. The user's SVD may differ substantially from that delivered to the support site and different user sites may have different SVDs. For example, the support site's SVD may list source files, requirements and design documentation, etc., in addition to the executable software files and user manuals. Figure 161 lists developer's key activities related to ***Version/revision/release***.

Developer's key activities related to ***Version/revision/release***	References in MIL-STD-498
Describe the approach to be followed for defining and recording the version/revision/release, identifying risks/uncertainties and plans for dealing with them. Cover all contractual clauses pertaining to the version/revision/release.	SDP 5.13.3, 5.14.1, 5.14.2, 5.14.3
Include the version/revision indicator on the title page of each document or, if information resides in a database or other alternative form, include on external and internal labels or by equivalent identification methods.	All DIDs 10.1.c
Identify the entities to be placed under configuration control including the software products to be developed or used under the contract and the elements of the software development environment. Include the version/revision/release status of each entity in the identification scheme.	5.14.1
Record information regarding the compilation/build process to be used to create the executable files including compilers/assemblers, hardware, software, etc., and their versions in the SPS. Record information regarding the software needed to support installation in the SIP. Record information regarding the software, hardware, and documentation needed to support the deliverable software in the STrP.	SIP 4.x.2 SPS 5.2 STrP 3.2, 3.3, 3.4
Establish and implement procedures designating the levels of control each identified entity must pass through and the steps to be followed to request authorization for changes, process change requests, track changes, distribute changes and maintain past versions.	5.14.2
Prepare and maintain records of the configuration status of all entities that have been placed under project-level or higher configuration control. Maintain these records for the life of the contract. Include, as applicable, the current version/revision/release of each entity, a record of changes to the entity since being placed under project-level or higher configuration control, and the status of problem/change reports affecting the entity.	5.14.3
If database development uses a database management system (DBMS), record the name and version/release of that DBMS. For any software unit, configuration item, user, etc., with which the software unit or group of software units used for database access or manipulation interfaces, record the identification of the interfacing entities, including the version.	DBDD 3.d, 5.x.e.2
If software is to reside in firmware, record the version/release of the software to be used for programming and reprogramming the firmware device.	FSM 3.x.4
Identify each component of the system/subsystem and each software unit that makes up the CSCI and assign each software unit a project-unique identifier. Provide identifying information for existing design, components, or software, including the version.	SDD 4.1.a, 4.1.d SSDD 4.1.a, 4.1.d

FIGURE 161. Developer's key activities related to ***Version/revision/release***.

Developer's key activities related to ***Version/revision/release***	References in MIL-STD-498
Identify each interface of the system/subsystem and software unit and assign each interface a project-unique identifier. Provide identifying information for existing design, component, or software, including the version.	SDD 4.3.1 SSDD 4.3.1
Identify each interface of the CSCI, subsystem, or system and assign each interface a project-unique identifier. Provide identifying information for existing design or software, including the version.	IDD 1.1, 3.1 IRS 1.1, 3.1 SRS 3.3.1 SSS 3.3.1
Provide the version of computer software that must be used by, or incorporated into, the CSCI/subsystem/system such as operating systems, DBMSs, communications/network software, utility software, input and equipment simulators, test software, and manufacturing software.	SRS 3.10.3 SSS 3.10.3
Identify the hardware and software configurations used for each test including revision level and version number. Identify each test and each test case by project-unique identifier.	STR 5.b
Demonstrate to the acquirer that the deliverable software can be regenerated (compiled/linked/loaded into an executable product) and maintained using commercially available, acquirer-owned, or contractually deliverable software and hardware designated in the contract or approved by the acquirer.	5.13.7
Identify and record the version description for each user site. Prepare the version/revision/release for the support site. Include all applicable items in the SVD DID.	5.12.2, 5.13.3 SVD
Meet general requirements and perform integral processes of the standard.	4, 5.14-5.19, Appendix B-G

FIGURE 161. Developer's key activities related to ***Version/revision/release*** - *(continued)*.

The developer's software development plan will describe the plans for:

- Keeping track of the software
- Identifying, tracking, and controlling each version/revision/release
- Keeping track of the information regarding the software, such as requirements, designs, and tests

5.75.2 Acquirer's responsibilities and considerations. Losing track of the pieces that comprise a CSCI, subsystem, or system is a high risk factor and is a constant concern for both the developer and the acquirer. Software development consists of thousands, sometimes millions of pieces of software and software related information. Keeping track of it all, knowing what pieces go with one another, and how they all relate, can be a major challenge.

To assist the development team to keep track of all of the pieces, software tools have been created to help identify and control the project's work products; marking schemes have been devised; configuration management organizations have been instituted; and company resources have been invested company-wide to develop disciplined processes and procedures. One benefit gained from CASE tools and other integrated software development tools is that identification of the software products is often automated, lessening the risk of losing any of the pieces.

The objectives of all this "keeping track" are to assure that when all is finished: (1) the acquirer will receive a complete product with up-to-date information regarding that product (not some earlier version), (2) the developer will be able to deliver to the user, acquirer, and support agency the exact version that passed testing, and (3) the support agency will be able to regenerate the executable software.

5.75.3 Additional things to think about.

Does the developer's identification scheme provide for project-unique identifiers, version/revision/release identifiers, and (possibly) other information that helps keep track of the software and the intrinsic information regarding its development?
Is responsibility identified for each work product, or portion of work product, i.e., author, team, project, etc.? Does the developer's plan for control clearly define the responsibilities and procedures/practices to be used to pass work products from one control level to the next? What criteria must a product possess that makes it ready for the next level of control?
Is the relationship clear between the software engineering environment (SEE) and the software development library (SDL)? How are work products in the SEE related to those in the SDL?
What provisions have been made to maintain, backup, and otherwise archive versions/revisions/releases of the software and documentation?
Will the current version of software need to be supported in the future?
What are the plans and procedures for archive or disposal of the outdated software after the new software is delivered and installed?
Are the master copies of all delivered software products (including documents or other forms of information) being maintained for the duration of the contract?

5.75.4 Related guidebook topics.

Software configuration management
Software development library
Software support
Software transition