

BY ORDER OF THE COMMANDER

SMC Standard SMC-S-012

13 June 2008



Supersedes:
New issue

Air Force Space Command

**SPACE AND MISSILE SYSTEMS CENTER
STANDARD**

**SOFTWARE
DEVELOPMENT
FOR SPACE SYSTEMS**


APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

FOREWORD

1. This standard defines the Government's requirements and expectations for contractor performance in defense system acquisitions and technology developments.
2. This new-issue SMC standard comprises the text of The Aerospace Corporation report number TOR-2004(3909)-3537, Rev B.
3. Beneficial comments (recommendations, changes, additions, deletions, etc.) and any pertinent data that may be of use in improving this standard should be forwarded to the following addressee using the Standardization Document Improvement Proposal appearing at the end of this document or by letter:

Division Chief, SMC/EAE
SPACE AND MISSILE SYSTEMS CENTER
Air Force Space Command
483 N. Aviation Blvd.
El Segundo, CA 90245

4. This standard has been approved for use on all Space and Missile Systems Center/Air Force Program Executive Office - Space development, acquisition, and sustainment contracts.



James Horejsi, Col, USAF
SMC Chief Engineer

Contents

1.	Scope	1
1.1	Purpose	1
1.2	Application	1
1.3	Order of precedence	3
2.	Referenced Documents	5
3.	Definitions	7
3.1	Terms	7
4.	General Requirements	13
4.1	Software development process	13
4.2	General requirements for software development.....	13
5.	Detailed Requirements	17
5.1	Project planning and oversight	17
5.2	Establishing a software development environment	18
5.3	System requirements analysis.....	19
5.4	System design.....	20
5.5	Software requirements analysis.....	21
5.6	Software design	21
5.7	Software implementation and unit testing.....	22
5.8	Unit integration and testing	23
5.9	Software item qualification testing.....	24
5.10	Software/hardware item integration and testing	26
5.10.1	Preparing for software/hardware item integration and testing	26
5.10.2	Performing software/hardware item integration and testing.....	27
5.10.3	Revision and retesting	27
5.10.4	Analyzing and recording software/hardware item integration and test results	27
5.11	System qualification testing	27
5.11.1	Independence in system qualification testing	28
5.11.2	Testing on the target computer system	28
5.11.3	Preparing for system qualification testing	28
5.11.4	Dry run of system qualification testing.....	28
5.11.5	Performing system qualification testing	28
5.11.6	Revision and retesting	28
5.11.7	Analyzing and recording system qualification test results.....	28
5.12	Preparing for software transition to operations	28
5.12.1	Preparing the executable software	29
5.12.2	Preparing version descriptions for user sites.....	29
5.12.3	Preparing user manuals.....	29
5.12.3.1	Software user manuals	29
5.12.3.2	Computer operation manuals	29
5.12.4	Installation at user sites.....	29
5.13	Preparing for software transition to maintenance.....	30
5.13.1	Preparing the executable software	30
5.13.2	Preparing source files.....	30
5.13.3	Preparing version descriptions for the maintenance site.....	30
5.13.4	Preparing the “as built” software item design and related information	30

5.13.5	Updating the system/subsystem design description.....	31
5.13.6	Updating the software requirements	31
5.13.7	Updating the system requirements.....	31
5.13.8	Preparing maintenance manuals.....	31
	5.13.8.1 Computer programming manuals.....	31
	5.13.8.2 Firmware support manuals	32
5.13.9	Transition to the designated maintenance site	32
5.14	Software configuration management.....	32
5.14.1	Configuration identification.....	32
5.14.2	Configuration control.....	32
5.14.3	Configuration status accounting.....	33
5.14.4	Configuration audits.....	33
5.14.5	Packaging, storage, handling, and delivery.....	33
5.15	Software peer reviews and product evaluations	33
5.15.1	Software peer reviews.....	33
	5.15.1.1 Prepare for software peer reviews	33
	5.15.1.2 Conduct peer reviews	34
	5.15.1.3 Analyze peer review data	34
5.15.2	Software product evaluations.....	34
	5.15.2.1 In-process and final software product evaluations	34
	5.15.2.2 Software product evaluation records	34
	5.15.2.3 Independence in software product evaluation	35
5.16	Software quality assurance	35
5.16.1	Software quality assurance evaluations	35
5.16.2	Software quality assurance records.....	35
5.16.3	Independence in software quality assurance	35
5.16.4	Software quality assurance noncompliance issues.....	35
5.17	Corrective action	36
5.17.1	Problem/change reports	36
5.17.2	Corrective action system.....	36
5.18	Joint technical and management reviews	36
5.18.1	Joint technical reviews	37
5.18.2	Joint management reviews	37
5.19	Risk management	37
5.20	Software management indicators.....	37
5.21	Security and privacy	38
5.22	Subcontractor management	38
5.23	Interface with software IV&V agents.....	38
5.24	Coordination with associate developers	38
5.25	Improvement of project processes.....	38
6.	Notes.....	39
6.1	Intended use.....	39
6.2	Data Item Descriptions (DIDs).....	39
6.3	Relationship between standard and CDRL.....	40
6.4	Delivery of tool contents	40
6.5	Tailoring guidance.....	40
6.6	Related standardization documents	40
6.7	Subject term (key word) listing	41
Appendix A.	List of acronyms	A-1
A.1	Scope	A-1
A.2	Acronyms	A-1

Appendix B. Interpreting this standard for incorporation of COTS and reusable software products	B-1
B.1 Scope	B-1
B.2 Evaluating reusable software products	B-1
B.3 Interpreting this standard’s activities for reusable software products	B-3
Appendix C. Category and severity classifications for problem reporting	C-1
C.1 Scope	C-1
C.2 Classification by category	C-1
C.3 Classification by severity	C-3
Appendix D. Software product evaluations.....	D-1
D.1 Scope	D-1
D.2 Required evaluations	D-1
D.3 Criteria definitions.....	D-1
Appendix E. Candidate joint management reviews.....	E-1
E.1 Scope	E-1
E.2 Assumptions	E-1
E.3 Candidate reviews	E-1
E.3.1 Software plan reviews	E-1
E.3.2 Operational concept reviews	E-1
E.3.3 System/Subsystem requirements reviews.....	E-1
E.3.4 System/Subsystem design reviews	E-1
E.3.5 Software requirements reviews	E-2
E.3.6 Software design reviews.....	E-2
E.3.7 Test readiness reviews.....	E-2
E.3.8 Test results reviews	E-2
E.3.9 Software usability reviews	E-2
E.3.10 Software maintenance reviews	E-2
E.3.11 Critical requirements reviews.....	E-2
Appendix F. Candidate management indicators.....	F-1
F.1 Scope	F-1
F.2 Candidate indicators	F-1
Appendix G. Guidance on contracting for delivery of software products.....	G-1
G.1 Scope	G-1
G.2 Contracting for deliverables	G-1
G.3 Scheduling deliverables.....	G-1
G.4 Format of deliverables	G-1
G.5 Tailoring the content requirements for deliverables.....	G-2
Appendix H. Software Development Plan Template	H-1

Figures

Figure C.2-1 Example categories of software products.....	C-1
Figure C.2-2 Example categories of software activities.....	C-2
Figure C.3-1 Example scheme for classifying problems by severity	C-3
Figure D.3-1 Software products and associated evaluation criteria	D-4

1. Scope

1.1 Purpose

The purpose of this standard is to establish uniform requirements for software development activities.

1.2 Application²

This standard applies to the development of systems that contain software (such as hardware-software systems), software-only systems, and stand-alone software products. The application of this standard is intended as follows.

1.2.1 Organizations and agreements

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

1.2.2 Contract-specific application

This standard is invoked by citing it on a contract. It applies to each software product and to each type of software covered by the contract, regardless of storage medium, to the extent specified in the contract. The acquirer is expected to specify the types of software to which the standard applies and to tailor the standard appropriately for each type of software. While this standard is written in terms of software items, it may be applied to software not designated as a software item, with the term “software item” interpreted appropriately. Software installed in firmware is subject to all of the aforementioned provisions. This standard does not apply to the hardware element of firmware.

Unless specified otherwise in the contract, for SMC and NRO programs, this standard shall apply to the prime contractor and all significant software team members. This standard shall apply to the following categories of software: onboard software (e.g., spacecraft, communications, payload); ground operations software (e.g., mission planning; mission processing; mission support; telemetry, tracking, and commanding; infrastructure and services); and other software used in satisfying, verifying, or validating requirements or used in performing or supporting operations or sustainment (e.g., training, simulation, analysis, database support, automatic test equipment, and maintenance). A *software team member* is any internal or external organization that develops, tests, or supports software-related work being performed for this contract and has an agreement (formal or informal) with the prime contractor or any other software team member. These organizations include, but are not limited to, intra-corporation software organizations, in-house service providers, developers, fabrication/manufacturing organizations, laboratories, and subcontractors. Examples of an agreement include a contract, work authorization, memorandum of agreement, or oral agreement. A *significant software team member* is a software team member with significant software development

² This is based on information from Section 1 of J-STD-016-1995 [6].

responsibility. Significant software development responsibility includes responsibility for any category of software described above.

1.2.3 Tailoring

This standard can be tailored for each type of software to which it is applied. While tailoring is the responsibility of the acquirer, prospective and selected developers may provide suggested tailoring. General tailoring guidance can be found in Section 6.5.

1.2.4 Compliance

Compliance with this standard as tailored for a project and recorded in the contract is defined as:

- a) Performing the activities that resulted from tailoring this standard for a specific project,
- b) Recording applicable information resulting from the performance of the activities.

An activity is complete when all actions constituting the activity, as specified in the contract or agreement, have been accomplished and all applicable information has been recorded.

1.2.5 Interpretation of selected terms

The following terms have a special interpretation as used in this standard.

1.2.5.1 Interpretation of “system”

The following interpretations apply:

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

For most SMC and NRO programs, software is an integral part of the combined hardware-software space system. For these programs, this standard and its tailoring shall be interpreted in that context.

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

1.2.5.2 Interpretation of “participate” in system development²

The term “participate” in paragraphs regarding system-level activities is to be interpreted as follows:

- a) If the software covered by this standard is part of a hardware-software system for which this standard covers only the software portion, the term “participate” is to be interpreted as “take part in, as described in the SDP.”
- b) If the software (possibly with its computers) is considered to constitute a system, the term “participate” is to be interpreted as “be responsible for.”
- c) If the software in a hardware-software system is modified, but the hardware is not, the term “participate” may also be interpreted as “be responsible for.”

1.2.5.3 Interpretation of “develop,” “define,” etc.

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

1.2.5.4 Interpretation of “record” (the verb)

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

1.2.5.5 Interpretation of “applicable”

Within this standard, requirements to provide “applicable” information or to perform “applicable” activities are interpreted as meaning to “provide the information or perform the activities that are required for the project in accordance with the methods, tools, and procedures explained in the Software Development Plan.” The term “applicable” suggests that not all development projects will require the information or the activity. The acquirer and the developer jointly agree upon whether providing information or performing activities are applicable.

1.3 Order of precedence

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

2. Referenced Documents

- [1] Abelson, L. A., R. J. Adams, S. Eslinger. *Metrics-based Software Acquisition Management*. Aerospace Report No. TOR-2004(3909)3405. The Aerospace Corporation. May 2004.
- [2] Adams, R. J., S. Eslinger, P. Hantos, K. L. Owens, L. T. Stephenson, R. Weiskopf. *Recommended Software Standards for Space Systems*. Aerospace Report No. TOR-2004(3909)-3406. The Aerospace Corporation. May 2004.
- [3] Defense Acquisition University. *GLOSSARY: Defense Acquisition Acronyms and Terms*. September 2003.
- [4] Department of Defense. *Software Development and Documentation*. MIL-STD-498, December 1994.
- [5] Department of Defense and U.S. Army. *Practical Software and System Measurement*. Version 4.0c, March 2003.
- [6] Electronic Industries Association / Institute of Electrical and Electronic Engineers, Inc. *EIA/IEEE Interim Standard J-STD-016-1995. Standard for Information Technology, Software Life Cycle Processes Software Development Acquirer-Supplier Agreement*. September 1995.
- [7] Institute of Electrical and Electronics Engineers. *IEEE Standard Classification for Software Anomalies*. IEEE 1044-1993. January 1995
- [8] Institute of Electrical and Electronics Engineers. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. September 1990.
- [9] International Organization for Standards / International Electrotechnical Commission (ISO / IEC). *Software Engineering - Software Measurement Process*. International Organization for Standardization. ISO/IEC 15939. 2002.
- [10] McGarry, John, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean and Fred Hall. *Practical Software Measurement: Objective Information for Decision Makers*. Addison Wesley, October 2001.
- [11] Software Engineering Institute. *Capability Maturity Model Integration, Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/PPD/SS, V1.1), Continuous Representation*. Carnegie Mellon University, SEI-2002-TR-011, March 2002.
- [12] Software Engineering Institute. *Capability Maturity Model Integration, Version 1.1, CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing (CMMI-SE/SW/PPD/SS, V1.1), Staged Representation*. Carnegie Mellon University, SEI-2002-TR-012, March 2002.

3. Definitions

3.1 Terms

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

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

Approval. Written notification by an authorized representative of the acquirer that a developer's plans, requirements, designs, 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.

Architecture. The organizational structure of a system or software item, identifying its components, their interfaces, and a concept of interaction among them.

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

Availability. This defines the probability that the system, prior to the start of a mission, will be mission capable when a mission is needed. Availability accounts for system faults and the time it takes to restore the system to a mission capable state following failure (maintainability).

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

Build.

- (1) A version of software that meets a specified subset of the requirements that the completed software will meet.
- (2) The period of time during which such a version is developed.

Note: The relationship of the terms "build" and "version" is up to the developer; for example, it may take several versions to reach a build, a build may be released in several parallel versions (such as to different sites), or the terms may be used as synonyms.

Commercial off-the-shelf (COTS) software. See reusable software product.

Computer database. See database.

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

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

Computer software. See software.

Computer software configuration item (CSCI). See software item.

Computer software unit (CSU). See software unit.

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

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

Data item description (DID). The format and content preparation instructions for a data product generated by the specific and discrete task requirement as delineated in the contract. Examples include DIDs for the Software Development Plan (SDP), Software Design Description (SDD), Software Test Report (STR), and Software User Manual (SUM). See Section 6.2 for a list of DIDs applicable to this standard.

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

Dependability. This defines the probability that the system will be able to complete the mission even if there is a failure given the system was available at the start of the mission. Dependability accounts for system faults and the time it takes to restore the system to a mission capable state following failure (maintainability).

Design. Those characteristics of a system or software item 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.

Developer. An organization that develops software products (“develops” may include new development, modification, reuse, reengineering, maintenance, or any other activity that results in software products).

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.

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

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

Hardware item. An aggregation of hardware that satisfies an end use function and is designated for purposes of specification, interfacing, qualification testing, configuration management, or other purposes.

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

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

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

Key performance parameter (KPP)³. Those minimum attributes or characteristics considered most essential for an effective military capability. Failure to meet a KPP threshold can be cause for the system concept to be reevaluated or the program to be reassessed or terminated. KPPs are validated by the Joint Requirements Oversight Council (JROC).

Maintainability. Prior to the start of a mission, this defines the probability that the system can be retained in or returned to a mission capable state within a specified period of time when a failure occurs. Pre-mission maintainability supports availability. During a mission, this defines the probability that the system can be retained in or returned to an operational state within a specified period of time when a failure occurs. Mission maintainability supports dependability.

Maintenance (of software)⁴. See software maintenance.

Maintenance organization. The organization that is responsible for modifying and otherwise sustaining the software after transition from the development organization.

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

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

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

Reengineering. The process of examining and altering an existing system to reconstitute it in a new form. May include reverse engineering (analyzing a system and producing a representation at a higher level of abstraction, such as design from code), restructuring (transforming a system from one

³ This is from the *GLOSSARY: Defense Acquisition Acronyms and Terms*, Eleventh Edition, September 2003 [3].

⁴ This is based on information from Section 3.1 of J-STD-016-1995 [6].

representation to another at the same level of abstraction), redocumentation (analyzing a system and producing user or support documentation), forward engineering (using software products derived from an existing system, together with new requirements, to produce a new system), retargeting (transforming a system to install it on a different target system), and translation (transforming source code from one language to another or from one version of a language to another).

Reliability. This defines the probability that the system will be able to complete the mission without failure given the system was available at the start of the mission. Reliability does not account for returning the system to a mission capable state (maintainability).

Requirement.

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

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

Software⁵. Computer programs, procedures, and data pertaining to the operation of a computer system. Data may include, for example, information in databases, rule bases, configuration data. Procedures may include, for example, interpreted scripts.

Note: Although some definitions of software include documentation, this standard limits the definition to computer programs, procedures, and data.

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

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

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

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

⁵ This is adapted from *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990. September 28, 1990 [8].

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

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

Software item⁴. An aggregation of software that satisfies an end use function and is designated for purposes of specification, interfacing, qualification testing, configuration management, or other purposes. Software items are selected based on tradeoffs among software function, size, host or target computers, developer, support strategies, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors. A software item is comprised of one or more software units. A software item is sometimes called a computer software configuration item (CSCI).

Software maintenance⁴. 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 maintenance includes sustaining support, aid to users, and related activities.

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

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

Software support. See software maintenance.

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

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

Software transition. The set of activities that enables responsibility for software to pass from one organization to another.

Software unit. An element in the design of a software item; for example, a major subdivision of a software item, 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. A software unit is sometimes called a computer software unit (CSU).

Support (of software). See software maintenance.

Supportability. The degree to which system design characteristics and planned logistics resources (including software) meet pre-mission readiness and operational utilization requirements. The primary performance requirements that characterize supportability are availability, dependability/reliability, and maintainability.

Technical performance measure (TPM). A measurement that indicates progress toward meeting critical system characteristics (technical parameters) that are specified in requirements or constrained by system design. Technical parameters that are tracked with TPMs have clearly identifiable and measurable target and threshold values. Examples of technical parameters, which can be tracked using TPMs, are space vehicle weight, space vehicle power, and computer system resource margins (e.g., CPU, I/O, memory margins).

Test. The terms “test” and “testing,” as used in this standard, refer to the activities of verifying that the implementation meets the design (unit and integration testing) and verifying that the requirements are satisfied (qualification testing). These terms should not be confused with the verification method “Test,” which is only one of the four standard methods used for verification (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

Testability. A design characteristic of hardware and software, which allows the status of an item to be confidently determined in a timely fashion.

Transition (of software). See software transition.

User. For the purpose of this standard, users include operators of the system in addition to the end users of the data produced by the system.

Work product⁶. Any artifact produced by a process.

⁶ This is from Capability Maturity Model Integration, Version 1.1, CMU/SEI-2002-TR-012, March 2002, p. 24. [11, 12].

4. General Requirements

4.1 Software development process

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

The framework used to organize the major software activities is called the software development life cycle model. The developer shall select software life cycle model(s) appropriate to the software being developed and shall document the selected software life cycle model(s) in the SDP. A software project may have more than one type of software development life cycle model in use for different types of software being developed for the contract.

- a) Project planning and oversight (Section 5.1)
- b) Establishing a software development environment (Section 5.2)
- c) System requirements analysis (Section 5.3)
- d) System design (Section 5.4)
- e) Software requirements analysis (Section 5.5)
- f) Software design (Section 5.6)
- g) Software implementation and unit testing (Section 5.7)
- h) Unit integration and testing (Section 5.8)
- i) Software item qualification testing (Section 5.9)
- j) Software/hardware item integration and testing (Section 5.10)
- k) System qualification testing (Section 5.11)
- l) Preparing for software transition to operations (Section 5.12)
- m) Preparing for software transition to maintenance (Section 5.13)
- n) Integral processes:
 - 1) Software configuration management (Section 5.14)
 - 2) Software peer reviews and product evaluation (Section 5.15)
 - 3) Software quality assurance (Section 5.16)
 - 4) Corrective action (Section 5.17)
 - 5) Joint technical and management reviews (Section 5.18)
 - 6) Risk management (Section 5.19)
 - 7) Software management indicators (Section 5.20)
 - 8) Security and privacy (Section 5.21)
 - 9) Subcontractor management (Section 5.22)
 - 10) Interface with software IV&V agents (Section 5.23)
 - 11) Coordination with associate developers (Section 5.24)
 - 12) Improvement of project processes (Section 5.25)

4.2 General requirements for software development

The developer shall meet the following general requirements in carrying out the detailed requirements in Section 5 of this standard.

4.2.1 Software development methods

The developer shall use systematic, documented methods for all software development activities. These methods shall be described in the SDP.

4.2.2 Standards for software products

The developer shall develop and apply standards for representing requirements, design, code, test cases, test procedures, and test results. These standards shall be described in the SDP.

4.2.3 Traceability⁷

The developer shall perform bi-directional traceability between levels of requirements, between requirements and design, between design and the software that implements it, between requirements and qualification test information, and between computer hardware resource utilization requirements and measured computer hardware resource utilization. The result shall include the applicable traceability items of the System/Segment Specification DID, Interface Requirements Specification DID, Software Requirements Specification DID, System/Segment Design Description DID, Software Design Description DID, Software Test Plan DID, Software Test Description DID, Software Test Report (STR) DID, and Software Product Specification DID. (See Section 6.2 for the DID numbers.)

When requirements are derived in whole or in part from design decisions, then those requirements shall be traced to the documentation of those design decisions.

For the purposes of satisfying the requirements of this paragraph, the traceability “between design and the software that implements it” shall encompass all forms of software used to implement the design, not just the code. Thus, for example, design traceability to software would include implementation such as interpreted command scripts and rule bases, if they were used to implement the design, not just traceability to the interpreter or inference engine.

Note: The intent of this requirement is to trace both upward and downward to:

- 1) Ensure that all requirements are implemented and tested, and
- 2) Provide information for software maintenance.

4.2.4 Reusable software products

The developer shall identify and evaluate reusable software products (examples include COTS, software products in reuse libraries, and pre-existing developer software products) for use in fulfilling the requirements of the contract. The scope of the search and the criteria to be used for evaluation shall be as described in the SDP. Reusable software products that meet the criteria shall be used where practical. Appendix B provides candidate criteria for evaluating reusable software products. Incorporated software products shall meet the data rights requirements in the contract.

Note 1: In addition, the developer may be required by the contract to develop software products specifically for reuse.

Note 2: Any requirement that calls for the development of a software product may be met by a reusable software product that fulfills the requirement and meets the criteria established in the SDP. The reusable software product may be used as-is or modified and may be used to satisfy part or the entire requirement.

⁷ This is based on information from Section 4.2.3 of J-STD-016-1995 [6].

Note 3: When a reusable software product to be incorporated is the software itself, some of the requirements in this standard may require special interpretation. Such interpretation should be documented in the SDP.

4.2.5 Assurance of critical requirements

If a system relies on a computer subsystem (including software and computer hardware) to satisfy requirements identified as critical, the developer shall identify those computer hardware and/or software items or portions thereof whose failure could lead to violations of those critical requirements; develop a strategy, including both test and analyses, to ensure that the requirements, design, implementation, and operating procedures for the identified computer hardware and/or software minimize or eliminate the potential for such violations; record the strategy in the SDP; implement the strategy; and produce evidence, as part of the required computer hardware and software products, that the assurance strategy has been successfully carried out. The developer shall identify and develop strategies for the following types of critical requirements:

- a) Safety: those software items 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);
- b) Security: those software items or portions thereof whose failure could lead to a breach of system security;
- c) Privacy protection: those software items or portions thereof whose failure could lead to a breach of system privacy protection;
- d) Dependability, reliability, maintainability, and availability;
- e) Other mission-critical requirements (e.g., derived from Key Performance Parameters (KPPs)) as agreed to by the acquirer and developer.

4.2.6 Computer hardware resource utilization

The developer shall analyze requirements and design constraints concerning computer hardware resource utilization (such as maximum allowable use of processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity). The developer shall allocate computer hardware resources, monitor the utilization of these resources, and reallocate or identify the need for additional resources as necessary to meet requirements and design constraints. The allocated resource utilizations shall be managed as Technical Performance Measures (TPMs).

4.2.7 Recording rationale

The developer shall record rationale for key decisions made in specifying, designing, implementing, and testing the software. The rationale shall include trade-offs considered, analysis methods, and criteria used to make the decisions. The rationale shall be recorded in documents, code comments, or other media that will transition to the maintenance organization (contractor or government). The meaning of “key decisions” and the approach for providing the rationale shall be described in the SDP.

4.2.8 Access for acquirer review

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

5. Detailed Requirements

The order of the requirements in this section is not intended to specify the order in which they must be carried out. Some activities may not be applicable to the project or contracted effort. Many of the activities may be ongoing at one time; different software products may proceed at different paces; and activities specified in early subsections may depend on input from activities in later subsections. If the software is developed in multiple builds, some activities may be performed in every build, others may be performed only in selected builds, and activities and software products may not be complete until several or all builds are accomplished. Non-mandatory notes throughout Section 5 tell how to interpret each activity on a project involving multiple builds. A project involving a single build will accomplish all required activities in that build.

5.1 Project planning and oversight

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

Note: If a system or software item is developed in multiple builds, planning for each build should be interpreted to include:

- a) Overall planning for the contract,
- b) Detailed planning for the current build, and
- c) Planning for future builds to a level of detail compatible with the information available.

5.1.1 Software development planning

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

The developer shall define, and document in the SDP, the artifacts and other information to be developed and recorded to satisfy the intent and spirit of the contents of the software-related products required by this standard. In addition, the developer shall define and document in the SDP, the artifacts and other information to be developed and recorded to prepare all software-related CDRL items required by the contract. When any product (or portion thereof) described in the DIDs referenced in this standard is required as a CDRL item by the contract, the DID for the CDRL item, as tailored for the contract, shall take precedence over the DID contents referenced in this standard.

Note: The wording here and throughout this standard is designed to:

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

5.1.2 Software item test planning

The developer shall develop and record plans for conducting software item qualification testing. This planning shall include all applicable items in the Software Test Plan (STP) DID, as defined in the SDP (see Section 5.1.1). The software test planning shall address all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)) and levels (e.g., unit, item, subsystem, element, segment, or system) necessary to fully verify the software requirements, including the software interface requirements.

Note: See Section 6.2 for the STP DID number.

5.1.3 System test planning

The developer shall participate in developing and recording plans for conducting system qualification testing. The system test planning shall address all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)) and verification levels (e.g., unit, build, item, subsystem, element, segment, or system) necessary to fully verify the system requirements, including the system interface requirements.

Note 1: System qualification testing in this paragraph is to be interpreted to mean the verification of all levels of requirements higher in the specification tree than the software requirements (e.g., subsystem, element, segment, or system).

5.1.4 Planning for software transition to operations

The developer shall develop and record plans for performing software installation and training at the user sites specified in the contract. This planning shall include all applicable items in the Software Installation Plan (SIP) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SIP DID number.

5.1.5 Planning for software transition to maintenance

The developer shall identify all software development resources that will be needed by the maintenance organization to fulfill the support concept specified in the contract. The developer shall develop and record plans identifying these resources and describing the approach to be followed for transitioning deliverable items to the maintenance organization. This planning shall include all applicable items in the Software Transition Plan (STrP) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the STrP DID number.

5.1.6 Following and updating plans

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

5.2 Establishing a software development environment

The developer shall establish, control, and maintain a software development environment in accordance with the following requirements. The developer shall analyze the adequacy of the planned software engineering and test environments and shall document the results of that analysis in the SDP.

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

5.2.1 Software engineering environment

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

5.2.2 Software integration and test environment

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

5.2.3 Software development library

The developer shall establish, control, and maintain a software development library (SDL) to facilitate the orderly development and subsequent maintenance of software. The SDL may be an integral part of the software engineering and test environments. The developer shall maintain the SDL throughout the contract period of performance.

5.2.4 Software development files

The developer shall establish, control, and maintain a Software Development File (SDF) for each software unit or logically related group of software units, for each software item, for each build (when using iterative life cycle models), and, as applicable, for logical groups of software items, for subsystems, and for the overall system. The developer shall record information about the development of the software in appropriate SDFs and shall maintain the SDFs throughout the contract period of performance.

5.2.5 Non-deliverable software

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

5.3 System requirements analysis

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

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

5.3.1 Analysis of user input⁸

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

Note: This requirement is intended to ensure that all interested parties maintain ongoing communications regarding user needs throughout the development of the system. Interested parties include, but are not limited to, the user, acquirer, developer, and maintenance and test organizations.

5.3.2 Operational concept

The developer shall participate in defining and recording the operational concept for the system. The result shall include all applicable items in the Operational Concept Description (OCD) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the OCD DID number.

5.3.3 System requirements

Based on the analysis of user needs, the operational concepts, and other considerations, the developer shall participate in defining and recording the requirements to be met by the system and the methods to be used to ensure that each requirement has been met. The result shall include all applicable items in the System/Subsystem Specification (SSS) DID, as defined in the SDP (see Section 5.1.1).

Depending on CDRL provisions, requirements concerning system interfaces may be included in the SSS or in the Interface Requirements Specifications (IRSs).

Note 1: If a system consists of segments/subsystems, the activity in 5.3.3 is intended to be performed iteratively with the activities in 5.4 (System design) to define system requirements, design the system and identify its segments/subsystems, define the requirements for those segments/subsystems, design the segments/subsystems and identify their components, and so on.

Note 2: See Section 6.2 for the SSS and IRS DID numbers.

5.4 System design

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

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

5.4.1 System-wide design decisions

The developer shall participate in defining and recording system-wide design decisions (that is, decisions about the system's behavioral design and other decisions affecting the selection and design of system components). The result shall include all applicable items in the system-wide design section of the System/Subsystem Design Description (SSDD) DID, as defined in the SDP (see Section 5.1.1).

Note 1: Design decisions remain at the discretion of the developer unless converted to requirements. The developer is responsible for fulfilling all requirements and demonstrating this fulfillment through qualification testing (see Sections 5.9, 5.11). Design decisions act as developer-internal

⁸ The Note is based on information from Section 5.3.1 of J-STD-016-1995 [6].

“requirements,” to be implemented, imposed on subcontractors, if applicable, and confirmed by developer-internal testing, but their fulfillment need not be demonstrated to the acquirer.

Note 2: See Section 6.2 for the SSDD DID number.

5.4.2 System architectural design⁹

The developer shall participate in defining and recording the architectural design of the system (identifying the components of the system, their interfaces, and a concept of interaction among them) and the traceability between the system components and system requirements. The result shall include all applicable items in the architectural design and traceability sections of the System/Subsystem Design Description (SSDD) DID, as defined in the SDP (see Section 5.1.1).

Note 1: For conciseness, this section is written in terms of organizing a system into components. However, this should be interpreted to cover organizing a system into segments/subsystems, organizing a segment/subsystem into hardware items, software items, and manual operations, or other variations as appropriate.

Note 2: See Section 6.2 for the SSDD DID number.

5.5 Software requirements analysis

Based on the analysis of system requirements, the system design, and other considerations, the developer shall define and record the software requirements to be met by each software item, the methods and levels for verifying each requirement, and the traceability between the software item requirements and their parent requirements. The result shall include all applicable items in the Software Requirements Specification (SRS) DID and Interface Requirements Specification (IRS) DID, as defined in the SDP (see Section 5.1.1).

Note 1: If a software item is developed in multiple builds, its requirements may not be fully defined until the final build. The developer’s planning will identify the subset of each software item’s requirements to be defined in each build and the subset to be implemented in each build. Software requirements analysis for a given build should be interpreted to mean defining the software item requirements so identified for that build.

Note 2: See Section 6.2 for the SRS and IRS DID numbers.

5.6 Software design

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

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

5.6.1 Software item-wide design decisions

The developer shall define and record software item-wide design decisions (that is, decisions about the software item’s behavioral design and other decisions affecting the selection and design of the software units comprising the software item). The result shall include all applicable items in the software item-wide design section of the Software Design Description (SDD) DID, as defined in the SDP (see Section 5.1.1). Design pertaining to interfaces and databases shall be included in Interface

⁹ Note 1 is based on information from Section 5.4.2 of J-STD-016-1995 [6].

Design Descriptions (IDDs) and Database Design Descriptions (DBDDs), as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SDD, IDD, and DBDD DID numbers.

5.6.2 Software item architectural design

The developer shall define and record the architectural design of each software item (identifying the software units comprising the software item, their interfaces, and a concept of interaction among them) and the traceability between the software units and the software item requirements. The result shall include all applicable items in the architectural design and traceability sections of the Software Design Description (SDD) DID, as defined in the SDP (see Section 5.1.1). Designs pertaining to interfaces and databases shall be included in Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs), as defined in the SDP (see Section 5.1.1).

Note 1: Software units may be made up of other software units and may be organized into as many levels as are needed to represent the software item architecture. For example, a software item may be divided into three software units, each of which is divided into additional software units, and so on.

Note 2: See Section 6.2 for the SDD, IDD, and DBDD DID numbers.

5.6.3 Software item detailed design

The developer shall develop and record a description of each software unit. The result shall include all applicable items in the detailed design section of the Software Design Description (SDD) DID, as defined in the SDP (see Section 5.1.1). Designs pertaining to interfaces shall include all items in the Interface Design Description (IDD), as defined in the SDP (see Section 5.1.1). Designs of software units that are databases or are service units that access or manipulate databases shall include all items in the Database Design Description (DBDD), as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SDD, IDD, or DBDD DID numbers.

5.7 Software implementation and unit testing

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

Note 1: If a software item is developed in multiple builds, software implementation and unit testing of that software item might not be completed until the final build. Software implementation and unit testing in each build should be interpreted to include those units, or parts of units, needed to meet the software item requirements to be implemented in that build.

Note 2: The terms “unit test” and “unit testing” refer to the activity of verifying the correctness of the unit implementation. The use of the words “test” or “testing” in this paragraph should not be confused with the verification method of Test. The activity of unit testing may require the use of all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

5.7.1 Software implementation

The developer shall develop and record software corresponding to each software unit in the software item design. This activity shall include, as applicable, coding computer instructions and data definitions, building databases, populating databases and other data files with data values, and other activities needed to implement the design.

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

5.7.2 Preparing for unit testing

The developer shall establish test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data for testing the software corresponding to each software unit. The test cases shall cover the unit's design, including, as a minimum, the correct execution of all statements and branches; all error and exception handling; all software unit interfaces including limits and boundary conditions; start-up, termination, and restart (when applicable); and all algorithms. Legacy reuse software shall be unit tested for all modified reuse software units, for all reuse software units where the track record indicates potential problems (even if the reuse units have not been modified), and all critical reuse software units (even if the reuse units have not been modified). The developer shall record this information in the appropriate SDFs.

5.7.3 Performing unit testing

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

5.7.4 Revision and retesting

Based on the results of unit testing, the developer shall make all necessary revisions to the software, perform all necessary retesting, and update the SDFs and other software products as needed, based on the results of unit testing. Regression testing of affected software unit test cases shall be performed after any modification to previously tested software.

5.7.5 Analyzing and recording unit test results

The developer shall analyze the results of unit testing and shall record the test and analysis results in appropriate SDFs.

5.8 Unit integration and testing

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

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

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

Note 3: As stated in Note 1, the term "integration and test" refers to the activity of verifying the correct functioning of an integrated collection of units. The use of the words "test" or "testing" in this paragraph should not be confused with the verification method of Test. The activity of unit

integration and testing may require the use of all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

5.8.1 Preparing for unit integration and testing

The developer shall establish test cases (in terms of inputs, expected results, and evaluation criteria), test procedures, and test data for conducting unit integration and testing. The test cases shall cover, as a minimum, the correct execution of all interfaces between software units, including limit and boundary conditions; integrated error and exception handling across the software units under test; all end-to-end functional capabilities through the software units under test; all software requirements allocated to the software units under test; performance testing, including operational input and output data rates and timing and accuracy requirements; stress testing, including worst-case scenario(s); start-up, termination, and restart (when applicable); fault detection, isolation, and recovery handling (e.g., fault tolerance, fail over, data capture and reporting); and resource utilization measurement (e.g., CPU, memory, storage, bandwidth). Whenever possible, software unit integration testing shall be performed on the target hardware in a configuration as close as possible to the operational configuration. All reuse software, including modified and unmodified legacy reuse and COTS software, shall undergo software unit integration testing. The developer shall record this information in the appropriate SDFs.

5.8.2 Performing unit integration and testing

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

5.8.3 Revision and retesting

The developer shall make all necessary revisions to the software, perform all necessary retesting, and update the SDFs and other software products as needed, based on the results of unit integration and testing. Regression testing of affected software integration test cases shall be performed after any modification to previously tested software.

5.8.4 Analyzing and recording unit integration and test results

The developer shall analyze the results of unit integration and testing and shall record the test and analysis results in appropriate SDFs.

5.9 Software item qualification testing

The developer shall perform software item qualification testing in accordance with the following requirements.

Note 1: Software item qualification testing is performed to demonstrate to the acquirer that software item requirements have been met. It covers the software item requirements in Software Requirements Specifications (SRSs) and in associated Interface Requirements Specifications (IRs). This testing contrasts with developer-internal software item testing, performed as the final stage of unit integration and testing.

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

Note 3: As stated in Note 1, the term “software item qualification testing” refers to the activity of verifying that the software requirements have been met, including the software interface requirements. The use of the words “test” or “testing” in this paragraph should not be confused with the verification method of Test. The activity of qualification testing may require the use of all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

5.9.1 Independence in software item qualification testing

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

5.9.2 Testing on the target computer system

Software item qualification testing shall be performed using the target hardware. The target hardware used for software qualification testing shall be as close as possible to the operational target hardware and shall be in a configuration as close as possible to the operational configuration.

5.9.3 Preparing for software item qualification testing

The developer shall define and record the test preparations, test cases, and test procedures to be used for software item qualification testing and the traceability between the test cases, test procedure steps, and the software item and software interface requirements. The result shall include all applicable items in the Software Test Description (STD) DID, as defined in the SDP (see Section 5.1.1). The developer shall prepare the test data needed to carry out the test cases and provide the acquirer advance notice of the time and location of software item qualification testing. The software qualification test cases shall cover, as a minimum, verification of all software requirements under conditions that are as close as possible to those that the software will encounter in the operational environment (e.g., operational data constants, operational input and output data rates, operational scenarios, target hardware configurations); verification of all software interface requirements, using the actual interfaces wherever possible or high-fidelity simulation of the interfaces where not possible; verification of all software specialty engineering requirements (e.g., supportability, testability, dependability/reliability/maintainability/ availability, safety, security, and human systems integration, as applicable), including in particular verification of software reliability requirements and fault detection, isolation, and recovery requirements; stress testing, including worst-case scenarios; and resource utilization measurement (e.g., CPU, memory, storage, bandwidth). All software requirements shall be verified by software qualification testing whether they are satisfied by COTS, reuse (modified or unmodified) or newly developed software.

Note: See Section 6.2 for the STD DID number.

5.9.4 Dry run of software item qualification testing

The developer shall dry run the software item test cases and procedures to ensure that they are complete and accurate and that the software is ready for witnessed testing. The developer shall record the results of this activity in appropriate SDFs and shall update the software item test cases and procedures as appropriate.

5.9.5 Performing software item qualification testing

The developer shall perform software item qualification testing of each software item. The testing shall be in accordance with the software item test cases and procedures. Regression testing of the affected software qualification test cases shall be performed after any modification to previously tested software.

5.9.6 Revision and retesting

The developer shall make necessary revisions to the software, provide the acquirer advance notice of retesting, conduct all necessary retesting, and update the SDFs and other software products as needed, based on the results of software item qualification testing.

5.9.7 Analyzing and recording software item qualification test results

The developer shall analyze and record the results of software item qualification testing. The results shall include all applicable items in the Software Test Report (STR) DID, as defined in the SDP (see Section 5.1.1). The developer shall document and maintain a cumulative record of the verification status (fully verified, partially verified, not verified) of all software requirements, including software interface requirements, for all verification testing events, all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)), and all levels in the verification testing hierarchy (e.g., unit, item, build, item, subsystem, segment, or system).

Note: See Section 6.2 for the STR DID number.

5.10 Software/hardware item integration and testing

The developer shall participate in software/hardware item integration and testing activities in accordance with the following requirements.

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

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

Note 3: As stated in Note 1, the term “software/hardware item integration and testing” refers to the activity of verifying the correct functioning of an integrated collection of hardware and software items or portions thereof. The use of the words “test” or “testing” in this paragraph should not be confused with the verification method of Test. The activity of unit integration and testing may require the use of all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

5.10.1 Preparing for software/hardware item integration and testing

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

The software/hardware integration test cases shall cover, as a minimum, the correct execution of all software-to-software and software-to-hardware interfaces among the software and hardware items under test, including limit and boundary conditions; integrated error and exception handling across the software and hardware items under test; all end-to-end functional capabilities through the software and hardware items under test; all software requirements allocated to the software and hardware items under test; performance testing, including operational input and output data rates and timing and accuracy requirements; stress testing, including worst-case scenario(s); start-up, termination, and restart (when applicable); fault detection, isolation, and recovery handling (e.g., fault tolerance, fail over, data capture and reporting); and resource utilization measurement (e.g., CPU, memory, storage, bandwidth). Software/hardware integration testing shall be performed using the target hardware that is as close as possible to the operational target hardware and is in a configuration as close as possible to the operational configuration. All reuse software, including modified and unmodified legacy reuse and COTS software, shall undergo software/hardware item integration testing.

5.10.2 Performing software/hardware item integration and testing

The developer shall participate in software/hardware item integration and testing. The testing shall be in accordance with the software/hardware integration test cases and procedures.

5.10.3 Revision and retesting

The developer shall make necessary revisions to the software, participate in all necessary retesting, and update the appropriate SDFs and other software products as needed, based on the results of software/hardware item integration and testing. Regression testing of affected software/hardware integration test cases shall be performed after any modification to previously tested software.

5.10.4 Analyzing and recording software/hardware item integration and test results

The developer shall participate in analyzing the results of software/hardware item integration and testing. Software-related analysis and test results shall be recorded in appropriate SDFs.

5.11 System qualification testing

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

Note 1: System qualification testing is performed to demonstrate to the acquirer that system requirements have been met. It covers requirements in the system specification, the segment specifications (if applicable) and all other levels of requirements between the system and segment specifications and the software requirements specifications in the specification tree (e.g., element specifications, subsystem specifications, etc.), including interface requirements at all of these levels. This testing contrasts with developer-internal system testing, performed as the final stage of software/hardware item integration and testing.

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

Note 3: As stated in Note 1, the term “system qualification testing” refers to the activity of verifying that the system requirements have been met, including the system interface requirements. The use of the words “test” or “testing” in this paragraph should not be confused with the verification method of Test. The activity of qualification testing may require the use of all verification methods (i.e., Inspection (I), Analysis (A), Demonstration (D), and Test (T)).

5.11.1 Independence in system qualification testing

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

5.11.2 Testing on the target computer system

System qualification testing shall include testing using the operational target hardware in the operational configuration.

5.11.3 Preparing for system qualification testing

The developer shall participate in developing and recording the test preparations, test cases, and test procedures to be used for system qualification testing and the traceability between the test cases, test procedure steps, and the system requirements. The developer shall participate in preparing the test data needed to carry out the test cases and in providing the acquirer advance notice of the time and location of system qualification testing.

5.11.4 Dry run of system qualification testing

The developer shall participate in dry running the system test cases and procedures to ensure that they are complete and accurate and that the system is ready for witnessed testing. The developer shall record the software-related results of this activity in appropriate SDFs and shall participate in updating the system test cases and procedures as appropriate.

5.11.5 Performing system qualification testing

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

5.11.6 Revision and retesting

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

5.11.7 Analyzing and recording system qualification test results

The developer shall participate in analyzing and recording the results of system qualification testing.

5.12 Preparing for software transition to operations

The developer shall prepare for software use (including transition to operations) in accordance with the following requirements.

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

5.12.1 Preparing the executable software

The developer shall prepare the executable software for each user site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s). The result shall include all applicable items in the executable software section of the Software Product Specification (SPS) DID, as defined in the SDP (see Section 5.1.1).

Note 1: To contractually require only the executable software (delaying delivery of source files and associated support information to a later build), the acquirer can use the SPS DID, tailoring out all but the executable software section of that DID.

Note 2: See Section 6.2 for the SPS DID number.

5.12.2 Preparing version descriptions for user sites

The developer shall identify and record the exact version of software prepared for each user site. The information shall include all applicable items in the Software Version Description (SVD) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SVD DID number.

5.12.3 Preparing user manuals

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

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

5.12.3.1 Software user manuals

The developer shall identify and record information needed by hands-on users of the software (persons who will both operate the software and make use of its results). The information shall include all applicable items in the Software User Manual (SUM) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SUM DID number.

5.12.3.2 Computer operation manuals

The developer shall identify and record information needed to operate the computers on which the software will run. The information shall include all applicable items in the Computer Operation Manual (COM) DID, as defined by the SDP (see Section 5.1.1).

Note 1: This paragraph only applies to computers without commercial operations manuals or to computers whose commercial operations manuals are not adequate. Examples of computer hardware for which this paragraph may apply are the processors in special test equipment or flight testbeds.

Note 2: See Section 6.2 for the COM DID number.

5.12.4 Installation at user sites

The developer shall:

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

- b) Provide training to users as specified in the contract.
- c) Provide other assistance to user sites as specified in the contract.

5.13 Preparing for software transition to maintenance

The developer shall prepare for transition of software to maintenance in accordance with the following requirements.

Note 1: If software is developed in multiple builds, the developer's planning should identify what software, if any, is to be transitioned to the maintenance organization in each build. Preparing for software transition in each build is interpreted to include those activities necessary to carry out the transition plans for that build.

Note 2: Software maintenance may be performed by the same organization that developed the software or by a different organization (e.g., a government maintenance organization, another development contractor, or another organization within the company that developed the software).

5.13.1 Preparing the executable software

The developer shall prepare the executable software to be transitioned to the maintenance site, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s). The result shall include all applicable items in the executable software section of the Software Product Specification (SPS) DID, as defined by the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SPS DID number.

5.13.2 Preparing source files

The developer shall prepare the source files to be transitioned to the maintenance site, including any batch files, command files, data files, or other files needed to regenerate the executable software. The result shall include all applicable items in the source file section of the Software Product Specification (SPS) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SPS DID number.

5.13.3 Preparing version descriptions for the maintenance site

The developer shall identify and record the exact version of software prepared for the maintenance site. The information shall include all applicable items in the Software Version Description (SVD) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SVD DID number.

5.13.4 Preparing the "as built" software item design and related information

The developer shall update the design description of each software item to match the approved "as built" software and shall define and record: the methods to be used to verify copies of the software, the measured computer hardware resource utilization for the software item, other information needed to maintain the software, and traceability between the software item's source files and software units and between the computer hardware resource utilization measurements and the software item requirements concerning them. The result shall include all applicable items in the qualification, software maintenance, and traceability sections of the Software Product Specification (SPS) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SPS DID number.

5.13.5 Updating the system/subsystem design description

The developer shall participate in updating the system design description to match the approved “as built” system. The result shall include all applicable items in the System/Segment Design Description (SSDD) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SSDD DID number.

5.13.6 Updating the software requirements¹⁰

The developer shall participate in updating the software and interface requirements for each software item to match the approved “as built” system. The result shall include all applicable items in the Software Requirements Specification (SRS) DID and in the Interface Requirements Specification (IRS) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SRS and IRS DID numbers.

5.13.7 Updating the system requirements¹¹

The developer shall participate in updating the system and interface requirements to match the approved “as built” system. The result shall include all applicable items in the System/Segment Specification (SSS) DID and in the Interface Requirements Specification (IRS) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the SSS and IRS DID numbers.

5.13.8 Preparing maintenance manuals

The developer shall prepare maintenance manuals in accordance with the following requirements.

Note: Not all systems will need the manuals in this section. The intent is for the acquirer, maintainer, and developer to determine which manuals are appropriate for a given system and to require the development of only those manuals. The manuals specified in this standard can be substituted with commercial or other manuals that contain the required information. The manuals in this section supplement the System/Segment Design Description (SSDD) and the Software Product Specifications (SPSs), which serve as the primary sources of information for software maintenance. The user manuals cited in Section 5.12.3 are also useful to maintenance personnel.

5.13.8.1 Computer programming manuals

The developer shall identify and record information needed to program the computers on which the software was developed or on which it will run. The information shall include all applicable items in the Computer Programming Manual (CPM) DID, as defined in the SDP (see Section 5.1.1).

Note 1: This paragraph applies only to newly developed computer hardware, which does not have commercial programming manuals available, or to computer hardware for which the commercial programming manuals are not adequate.

Note 2: See Section 6.2 for the CPM DID number.

¹⁰ This is based on information from Section 5.13.6 of J-STD-016-1995 [6].

¹¹ This is based on information from Section 5.13.7 of J-STD-016-1995 [6].

5.13.8.2 Firmware support manuals

The developer shall identify and record information needed to program and reprogram any firmware devices in which the software will be installed. The information shall include all applicable items in the Firmware Support Manual (FSM) DID, as defined in the SDP (see Section 5.1.1).

Note: See Section 6.2 for the FSM DID number.

5.13.9 Transition to the designated maintenance site

The developer shall:

- a) Install and check out the deliverable software in the maintenance environment designated in the contract.
- b) Demonstrate to the acquirer that the deliverable software can be regenerated (compiled/linked/loaded into an executable product) and maintained using commercially available, acquirer-owned, or contractually deliverable software and hardware designated in the contract or approved by the acquirer.
- c) Provide training to the maintenance organization as specified in the contract.
- d) Provide other assistance to the maintenance organization as specified in the contract.

5.14 Software configuration management

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

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

5.14.1 Configuration identification

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

5.14.2 Configuration control

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

Note: The levels of control to be implemented are dependent upon the entity to be placed under configuration control. Thus, an SRS generally passes through author level control, software configuration management level control, and program level control. Software code, on the other hand, generally has more levels of control, for example: individual developer, software integration (team leader level), software build integration, software qualification, and segment/system control. The SDP defines the levels of control to be used for each entity to be placed under configuration control, in addition to the roles, responsibilities and procedures for each level.

5.14.3 Configuration status accounting

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

Note: The degree of formality of the configuration status accounting may differ at different levels of configuration control.

5.14.4 Configuration audits¹²

The developer shall periodically conduct audits of all entities that have been placed under any level of configuration control above the individual author/developer level. These audits shall ensure that each entity only incorporates all approved changes (and no other changes) scheduled for inclusion at the time of the audit.

Note: The degree of formality of the configuration audits may differ at different levels of configuration control.

5.14.5 Packaging, storage, handling, and delivery

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

5.15 Software peer reviews and product evaluations

The developer shall perform software peer reviews and product evaluation in accordance with the following requirements.

5.15.1 Software peer reviews

The developer shall perform peer reviews of software work products in accordance with the following requirements.

Note: Planning for peer reviews of software work products is part of software development planning (see Section 5.1.1) and is documented in the SDP.

5.15.1.1 Prepare for software peer reviews

The developer shall perform peer reviews on the software work products as specified in the SDP. For each peer review:

¹² This is based on information from Section 5.14.4 of J-STD-016-1995 [6].

- a) The developer shall determine what type of peer review will be conducted on the software work product.
- b) The developer shall identify key reviewers who must participate in the peer review.
- c) The developer shall ensure that the software work product satisfies the peer review entry criteria prior to distribution of peer review materials.
- d) Each reviewer shall prepare by reviewing the software work product prior to conducting the peer review.

5.15.1.2 Conduct peer reviews

- a) The developer shall identify and document defects and other issues in the software work product.
- b) The developer shall record and collect the results of the peer review, including action items.
- c) The developer shall ensure that the exit criteria for the peer review are satisfied.

5.15.1.3 Analyze peer review data

The developer shall analyze data about preparation, conduct, and results of the peer reviews.

5.15.2 Software product evaluations

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

Note 1: If a system or software item is developed in multiple builds, the software products of each build are 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.

Note 2: Planning for software product evaluations is part of software development planning (see Section 5.1.1) and is documented in the SDP.

Note 3: It is not the intention of Sections 5.15.1 and 5.15.2 to require two separate processes, one for peer reviews and one for software product evaluations, although the developer may choose to satisfy the requirements of the standard in this manner. Software products evaluations may be accomplished via the peer review process as long as the requirements for product evaluations specified in the following subparagraphs are satisfied.

5.15.2.1 In-process and final software product evaluations

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

5.15.2.2 Software product evaluation records

The developer shall prepare and maintain records of each software product evaluation. These records shall be maintained for the duration of the contract. Problems in software products under any level of configuration control above the individual author/developer level shall be handled as described in Section 5.17 (Corrective action).

5.15.2.3 Independence in software product evaluation

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

5.16 Software quality assurance

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

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

5.16.1 Software quality assurance evaluations

The developer shall conduct ongoing evaluations of software development processes, software products and work products, and software services to ensure:

- a) Adherence of the designated, performed processes to the applicable process descriptions, standards, and procedures in accordance with the contract and with the SDP;
- b) Adherence of the designated work products and services to the applicable process descriptions, standards, and procedures in accordance with the contract and with the SDP; and
- c) That each software product required by this standard or by other contract provisions exists and has undergone software product evaluations and peer reviews, testing (for those products where testing is applicable), and corrective action (for all identified problems), as required by this standard and by other contract provisions.

5.16.2 Software quality assurance records

The developer shall establish and maintain records of each software quality assurance activity. These records shall be maintained for the duration of the contract.

5.16.3 Independence in software quality assurance

The persons responsible for conducting software quality assurance evaluations shall not be the persons who developed the software product or work product, performed the process or service, or are responsible for the software product, work product, or process. The persons responsible for assuring compliance with the contract shall have the resources, responsibility, authority, and organizational freedom to permit objective software quality assurance evaluations and to initiate and verify corrective actions.

5.16.4 Software quality assurance noncompliance issues

The developer shall:

- a) Communicate quality issues and ensure resolution of noncompliance issues with the staff and managers;
- b) Use an established escalation mechanism to ensure that the appropriate level of management can resolve the issues; and

- c) Track noncompliance issues to resolution. Noncompliance issues in software products or work products under any level of configuration control above the individual author/developer level shall be handled as described in Section 5.17 (Corrective action system).

5.17 Corrective action

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

5.17.1 Problem/change reports

The developer shall prepare a problem/change report to describe each problem detected in software entities under any level of configuration control above the individual author/developer level and each problem in activities required by the contract or described in the SDP. The problem/change report shall describe the problem, the corrective action needed, and the actions taken to date. These reports shall serve as input to the corrective action system.

Note: The degree of formality of the problem/change reports may differ at different levels of configuration control.

5.17.2 Corrective action system

The developer shall implement a corrective action system for handling each problem detected in software entities under any level of configuration control above the individual author/developer level and each problem in activities required by the contract or described in the SDP. The system shall meet the following requirements:

- a) Inputs to the system shall consist of problem/change reports.
- b) The system shall be closed-loop, ensuring that all detected problems are promptly reported and entered into the system, action is initiated on them, resolution is achieved, status is tracked, and records of the problems are maintained for the duration of the contract.
- c) Each problem shall be classified by category and severity, as a minimum. Guidance is provided in Appendix C.
- d) Analysis shall be performed to detect trends in the problems reported.
- e) Corrective actions shall be evaluated to determine whether problems have been resolved, adverse trends have been reversed, and changes have been correctly implemented without introducing additional problems.

Note: The degree of formality of the problem/change reports may differ at different levels of configuration control.

5.18 Joint technical and management reviews

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

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

5.18.1 Joint technical reviews

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

- a) Review evolving software products, using as minimum criteria the software product evaluation criteria in Appendix D; review and demonstrate proposed technical solutions; provide insight and obtain feedback on the technical effort; surface and resolve technical issues.
- b) Review project status; surface near- and long-term risks regarding technical, cost, and schedule issues.
- c) Arrive at agreed-upon mitigation strategies for identified risks, within the authority of those present.
- d) Identify risks and issues to be raised at joint management reviews.
- e) Ensure ongoing communication between acquirer and developer technical personnel.

5.18.2 Joint management reviews

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

- a) Keep management informed about project status, directions being taken, technical agreements reached, and overall status of evolving software products.
- b) Resolve issues that could not be resolved at joint technical reviews.
- c) Arrive at agreed-upon mitigation strategies for near- and long-term risks that could not be resolved at joint technical reviews.
- d) Identify and resolve management-level issues and risks not raised at joint technical reviews.
- e) Obtain commitments and acquirer approvals needed for timely accomplishment of the project.

5.19 Risk management

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

5.20 Software management indicators

The developer shall perform software measurement throughout the software life cycle in accordance with ISO 15939 [9]. The developer shall plan the measurement process by identifying and defining 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 reporting mechanism. The developer shall record this information in the SDP or a Software Metrics Plan. The developer shall perform the measurement

process by collecting, interpreting, applying, and reporting on those indicators at regular intervals, as described in the SDP or Software Metrics Plan. Management indicators selected for use in managing the software development process are at the discretion of the contractor. Candidate indicators are given in Appendix F. Additional guidance is provided in the book “*Practical Software Measurement Objective Information for Decision Makers*” by John McGarry et al, Addison Wesley, 2001 [10].

If the acquirer requires specific software measures to be collected and reported, the developer shall incorporate these measures into their selected set of management indicators as part of their software measurement process.

5.21 Security and privacy

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

5.22 Subcontractor management

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

5.23 Interface with software IV&V agents

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

5.24 Coordination with associate developers

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

5.25 Improvement of project processes

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

6. Notes

6.1 Intended use

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

6.2 Data Item Descriptions (DIDs)

The following DIDs must be listed, as applicable, on the Contract Data Requirements List (CDRL) in order to have the product delivered under the contract.

<u>DID Title</u>	<u>DID Number</u>	<u>Reference Section</u>
Software Development Plan (SDP)	Use Appendix H	5.1.1, 5.1.2, 5.1.4, 5.1.5, 5.1.6, 5.2, 5.3.2, 5.3.3, 5.4.1, 5.4.2, 5.5, 5.6.1, 5.6.2, 5.6.3, 5.9.3, 5.9.7, 5.12.1, 5.12.2, 5.12.3.1, 5.12.3.2, 5.13.1, 5.13.2, 5.13.3, 5.13.4, 5.13.5, 5.13.6, 5.13.8.1, 5.13.8.2, 5.14.2, 5.15.1, 5.15.1.1, 5.15.2, 5.16, 5.16.1, 5.17.1, 5.17.2, 5.19, 5.20, 5.25
Software Test Plan (STP)	DI-IPSC-81438A	5.1.2, 5.1.3
Software Installation Plan (SIP)	DI-IPSC-81428A	5.1.4
Software Transition Plan (STrP)	DI-IPSC-81429A	5.1.5
Operational Concept Description (OCD)	DI-IPSC-81430A	5.3.2
System/Segment Specification (SSS)	DI-IPSC-81431A	5.3.3, 5.13.7
Interface Requirements Specification (IRS)	DI-IPSC-81434A	5.3.3, 5.5, 5.9, 5.13.6, 5.13.7
System/Segment Design Description (SSDD)	DI-IPSC-81432A	5.4.1, 5.4.2, 5.13.5, 5.13.8
Interface Design Description (IDD)	DI-IPSC-81436A	5.6.1, 5.6.2, 5.6.3
Software Requirements Specification (SRS)	DI-IPSC-81433A	5.5, 5.9, 5.11, 5.13.6, 5.14.2
Software Design Description (SDD)	DI-IPSC-81435A	5.6.1, 5.6.2, 5.6.3
Database Design Description (DBDD)	DI-IPSC-81437A	5.6.1, 5.6.2, 5.6.3
Software Test Description (STD)	DI-IPSC-81439A	5.9.3
Software Test Report (STR)	DI-IPSC-81440A	5.9.7, 5.11.7
Software Product Specification (SPS)	DI-IPSC-81441A	5.12.1, 5.13.1, 5.13.2, 5.13.4, 5.13.8
Software Version Description (SVD)	DI-IPSC-81442A	5.12.2, 5.13.3
Software User Manual (SUM)	DI-IPSC-81443A	5.12.3.1
Computer Operation Manual (COM)	DI-IPSC-81446A	5.12.3.2
Computer Programming Manual (CPM)	DI-IPSC-81447A	5.13.8.1
Firmware Support Manual (FSM)	DI-IPSC-81448A	5.13.8.2

6.3 Relationship between standard and CDRL

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

6.4 Delivery of tool contents

Depending on contract provisions, the developer may be permitted to satisfy CDRL requirements by delivering:

- 1) A repository or database containing the information specified in the cited DID;
- 2) A means of accessing that repository or database, such as a CASE tool, if not already available to the recipients designated on the CDRL; and
- 3) A hard-copy or electronically stored table of contents, specifying how and where to access the information required in each paragraph of the DID.

6.5 Tailoring guidance

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

6.6 Related standardization documents

Other standardization documents may be imposed or quoted in the SOW or Compliance Documents section of the contract to supplement the requirements in this standard. The acquirer should use caution to ensure that supplemental standards are appropriate to the project and that any conflicts among them or with this standard are identified and resolved. See *Recommended Software Standards for Space Systems*, Aerospace Report No. TOR-2004(3909)-3406 [2] for information on other software standards.

6.7 Subject term (key word) listing

The following list of key words may be used to catalog or characterize key topics in this standard.

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

Appendix A. List of acronyms

A.1 Scope

This appendix provides a list of acronyms used in this standard. This appendix is not a mandatory part of the standard. The information provided is intended for guidance only.

A.2 Acronyms

CASE	Computer-Aided Software Engineering
CDRL	Contract Data Requirements List
COM	Computer Operation Manual
COTS	Commercial Off-The-Shelf
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
CPM	Computer Programming Manual
DBDD	Database Design Description
DID	Data Item Description
DoD	Department of Defense
FSM	Firmware Support Manual
HW	Hardware
IDD	Interface Design Description
IPT	Integrated Product Team
IRS	Interface Requirements Specification
IV&V	Independent Verification and Validation
JROC	Joint Requirements Oversight Council
KPP	Key Performance Parameter
OCD	Operational Concept Description
SDD	Software Design Description
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SIP	Software Installation Plan
SOW	Statement of Work
SPS	Software Product Specification
SRS	Software Requirements Specification
SSDD	System/Segment Design Description
SSS	System/Segment Specification
STD	Software Test Description
STP	Software Test Plan
STR	Software Test Report
STrP	Software Transition Plan
SUM	Software User Manual
SVD	Software Version Description
SW	Software
TPM	Technical Performance Measure
WBS	Work Breakdown Structure

Appendix B. Interpreting this standard for incorporation of COTS and reusable software products

B.1 Scope

This appendix interprets this standard when applied to the incorporation of reusable software products. This appendix is an informative part of the standard. It is provided as an aid in using this standard.

B.2 Evaluating reusable software products

Examples of criteria for evaluating reusable software products are:

- a) Ability to provide required capabilities and meet required constraints
 - 1) Ability to satisfy requirements
 - 2) Ability to achieve necessary performance, especially with realistic operational workloads
 - 3) Appropriateness of algorithms in the COTS/reusable software product for use in the new system
 - 4) As evidenced by characterization/stress testing within the system context to determine capabilities and performance
- b) Ability to provide required protection (safety, security, and privacy)
 - 1) Provided inherently in the COTS or reusable software product, or
 - 2) Able to be provided around the COTS/reusable software product by system design and implementation
- c) Reliability/maturity
 - 1) As evidenced by established track record
 - 2) As evidenced by prototype evaluation within the system context
- d) Testability
 - 1) As evidenced by the ability to identify and isolate faults
- e) Operability
 - 1) Suitability of the COTS/reusable software product's implied operations concept to the operations concept of the new system
- f) COTS or reusable software product supplier viability
 - 1) Compatibility of COTS or reusable software product supplier's future direction with program needs (including both software and platform emphasis)
 - 2) Supplier long-term commitment to COTS or reusable software product
 - 3) Supplier long-term business prospects
 - 4) Type of supplier support available
 - 5) Quality of supplier support available
- g) Suitability for incorporation into the new system architecture
 - 1) Compatible software architecture and design features
 - 2) Absence of obsolete technologies
 - 3) Need for re-engineering and/or additional code development (e.g., wraps, "glue" code)
 - 4) Compatibility among the set of reusable software products
 - 5) As evidenced by prototyping within the system context (e.g., to determine compatibility, wraps, "glue" code)
- h) Ability to remove or disable features/capabilities not required in the new system
 - 1) Impact if those features cannot be removed/disabled or are not removed/disabled

- 2) As evidenced by prototyping within the system context
- i) Interoperability with other system and system-external elements
 - 1) Compatibility with system interfaces
 - 2) Adherence to standards (e.g., open systems interface standards)
 - 3) Ability to interface with legacy systems
- j) Availability of personnel knowledgeable about the reusable software product
 - 1) Training required
 - 2) Hiring required
 - 3) Vendor or third-party support required
- k) Availability and quality of documentation and source files
 - 1) Completeness
 - 2) Accuracy
- l) Acceptability of reusable software product licensing and data rights
 - 1) Restrictions on copying/distributing the reusable software product or documentation
 - 2) License or other fees applicable to each copy
 - 3) Acquirer's usage and ownership rights, especially to the source code
 - a) Ability to place source code in escrow against the possibility of the vendor/developer going out of business
 - 4) Warranties available
 - 5) Absence of unacceptable restrictions in standard license (e.g., export restrictions, expiring keys)
- m) Supportability
 - 1) Suitability of the COTS/reusable software product's support paradigm (e.g., distribution, installation) to the support concept of the new system, especially for mobile or remote sites
- n) Ability to make changes, including:
 - 1) Likelihood the software product will need to be changed
 - 2) Feasibility/difficulty of accomplishing that change, when changes are to be made by the program reusing the software product
 - a) Quality of design, code and documentation
 - b) Need for re-engineering and/or restructuring
 - 3) Feasibility/difficulty of accomplishing that change, when changes must be made by the vendor or product developer (e.g., for COTS or proprietary software)
 - a) Priority of changes required by this program versus other changes being made
 - b) Likelihood that the changed version will continue to be maintained by the vendor/developer
 - c) Likelihood of being able to modify future versions to include changes
 - d) Impact on life cycle costs
 - e) Impact if the current version is not maintained by the vendor/developer or if changes are not able to be incorporated into future versions
- o) Impacts of upgrades to COTS or reusable software products
 - 1) Frequency of COTS/reusable software product upgrades/modifications being made by the vendor/developer (i.e., a completely new version or upgrade patches to the existing version being released) after a particular version has been incorporated into the system
 - 2) Feasibility/difficulty of incorporating the new version of the COTS/reusable software product into the system
 - 3) Impact if the new version is not incorporated

- 4) Ability of the new architecture to support the evolution of COTS/reusable software products
- p) Compatibility of planned upgrades of COTS or reusable software products with software development plans and schedules
 - 1) Compatibility of planned upgrades with build content and schedules
 - 2) Impact on development costs and schedule to incorporated upgrades
 - 3) Dependencies among COTS and reusable software products
 - a) Potential for an incompatible set of COTS and/or reusable software products
 - b) Potential for schedule delays until all dependent COTS and/or reusable software products are upgraded
- q) Criticality of the functionality provided by the COTS or reusable software product
 - 1) Availability of alternate source(s) for the functionality
- r) Short- and long-term cost impacts of using the COTS or reusable software product
 - 1) Amount of management reserve needed to handle uncertainties
 - a) For example, less COTS/reusable software product usable; more newly developed software required; COTS/reusable software product limitations identified
- s) Technical, cost, and schedule risks and tradeoffs in using the COTS or reusable software product
 - 1) Ability to tolerate COTS or reusable software product problems beyond the program's control at any point in the system life cycle
 - 2) Ability to incorporate continuous evolution of COTS or reusable software products during development and maintenance

B.3 Interpreting this standard's activities for reusable software products

The following rules apply in interpreting this standard:

- a) Any requirement that calls for the development of a software product may be met by a reusable software product that fulfills the requirement and meets the criteria established in the SDP. The reusable software product may be used as-is or modified and may be used to satisfy part or the entire requirement.
- b) When a reusable software product to be incorporated is the software itself, some of the requirements in this standard may require special interpretation. Such interpretation should be documented in the SDP.

Appendix C. Category and severity classifications for problem reporting

C.1 Scope

This appendix contains guidance for a category and severity classification scheme to be applied to each problem submitted to the corrective action system. This appendix is an informative part of the standard. The developer may use alternate category and severity schemes, as described in the SDP. Additional guidance may be found in IEEE Std 1044, *IEEE Standard Classification for Software Anomalies* [7].

C.2 Classification by category

It is recommended the software problems be categorized by the products in which they occur and the activities in which the problems are injected and discovered.

- a) Figure C.2-1 provides examples of categories for classifying problems found in software products.
- b) Figure C.2-2 provides examples of categories for classifying problems by the software activity in which the problem is injected and the activity in which the problem is discovered.

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

Figure C.2-1 Example categories of software products

Section	Activity Categories
5.1	Project planning and oversight
5.2	Establishing a software development environment
5.3	System requirements analysis
5.4	System design
5.5	Software requirements analysis
5.6	Software design
5.7	Software implementation and unit testing
5.8	Unit integration and testing
5.9	Software item qualification testing
5.10	Software/hardware item integration and testing
5.11	System qualification testing
5.12	Preparing for software transition to operations
5.13	Preparing for software transition to maintenance
5.14	Software configuration management
5.15	Software peer reviews and product evaluation
5.16	Software quality assurance
5.17	Corrective action
5.18	Joint technical and management reviews
5.19	Risk management
5.20	Software management indicators
5.21	Security and privacy
5.22	Subcontractor management
5.23	Interface with software IV&V agents
5.24	Coordination with associate developers
5.25	Improvement of project processes

Figure C.2-2 Example categories of software activities

C.3 Classification by severity

Figure C.3-1 provides an example of a scheme for classifying problems by severity.

Severity	Applies if a problem could:
1	<ul style="list-style-type: none">a. Prevent the accomplishment of an operational or mission essential capabilityb. Jeopardize safety, security, or other requirement designated “critical”
2	<ul style="list-style-type: none">a. Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is knownb. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known
3	<ul style="list-style-type: none">a. Adversely affect the accomplishment of an operational or mission essential capability but a work-around solution is knownb. Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known
4	<ul style="list-style-type: none">a. Result in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capabilityb. Result in inconvenience or annoyance for development or maintenance personnel, but does not prevent the accomplishment of those responsibilities
5	Any other effect

Figure C.3-1 Example scheme for classifying problems by severity

Appendix D. Software product evaluations

D.1 Scope

This appendix identifies the software products that are to undergo software product evaluations, identifies the criteria to be used for each evaluation, and contains a default set of definitions for the evaluation criteria. This appendix is a mandatory part of the standard, subject to the following conditions:

- 1) These requirements may be tailored by the acquirer;
- 2) The developer may use alternate criteria or definitions if approved by the acquirer; and
- 3) If the development of a given software product has been tailored out of the standard, the requirement to evaluate that product does not apply.

D.2 Required evaluations

Figure D.3-1 identifies the software products that are to undergo software product evaluations and states the criteria to be applied to each one. Each software product and criterion is labeled for purposes of identification and tailoring. For convenience, they may be treated as subsections of this section (referring to the first criterion, for example, as D.2.1.a). The software products are expressed in lower case letters to convey that they are generic products, not necessarily in the form of hard-copy documents. Evaluations of system-level products are to be interpreted as participation in these evaluations. Some of the criteria are subjective. Because of this, there is no requirement to prove that the criteria have been met; the requirement is to perform the evaluations using these criteria and to identify possible problems for discussion and resolution.

D.3 Criteria definitions

The following subsections provide definitions for the criteria in Figure D.3-1 that may not be self-explanatory. The criteria are listed in alphabetical order, matching as closely as possible the wording used in Figure D.3-1.

D.3.1 Accurately describes (an item)

This criterion, applied to user/operator/programmer instructions and to the “as built” design and version descriptions, means that the instructions or descriptions are correct depictions of the software or other item described.

D.3.2 Adequate test cases, procedures, data, results

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

D.3.3 Consistent with indicated product(s)

This criterion means that:

- 1) No statement or representation in one software product contradicts a statement or representation in the other software products;
- 2) A given term, acronym, or abbreviation means the same thing in all of the software products; and
- 3) A given item or concept is referred to by the same name or description in all of the software products.

D.3.4 Contains all applicable information

This criterion uses the software product contents described in the SDP for the required contents of the software products that are not deliverable. This criterion uses the DIDs, as tailored for the contract, to specify the required content of deliverable software products. The formatting specified in the DID (required structure and numbering) is not relevant to this evaluation. The SDP describes the artifacts and other information to be developed and recorded for each product required by this standard (see Section 5.1.1).

D.3.5 Covers (a given set of items)

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

D.3.6 Feasible

This criterion means that, based upon the knowledge and experience of the evaluator, a given concept, set of requirements, designs, tests, etc. violates no known principles or lessons learned that would render it impossible to carry out.

D.3.7 Follows SDP

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

D.3.8 Internally consistent

This criterion means that:

- 1) No two statements or representations in a software product contradict one another;
- 2) A given term, acronym, or abbreviation means the same thing throughout the software product; and
- 3) A given item or concept is referred to by the same name or description throughout the software product.

D.3.9 Meets CDRL requirements, if applicable

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

D.3.10 Meets SOW, if applicable

This criterion means that the software product fulfills any SOW provisions regarding it. For example, the SOW may place constraints on the operational concept or the design.

D.3.11 Presents a sound approach

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

D.3.12 Shows evidence that an item under test meets its requirements

This criterion means that recorded test results show that the item under test either passed all tests the first time or was revised and retested until the tests were passed.

D.3.13 Testable

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

D.3.14 Understandable

This criterion means “understandable by the intended audience.” For example, software products intended for programmer-to-programmer communication need not be understandable by non-programmers. A product that correctly identifies its audience and is considered understandable to that audience meets this criterion.

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets contract, if applic.	Understandable	Intern. consistent	Follows SDP	Additional Criteria
Software Development Plan (5.1.1)	a. SDP DID*	b.	c.	d.	e.	f. (Updates)	g. Covers all activities/deliverables in SOW and CDRL h. Consistent with other project plans i. Presents a sound approach to the development
Software Test Plan (5.1.2, 5.1.3)	a. STP DID*	b.	c.	d.	e.	f.	g. Covers all software-related qualification activities in the SOW h. Covers all requirements for the items under test i. Consistent with other project plans j. Presents a sound approach to the testing
Software Installation Plan (5.1.4)	a. SIP DID*	b.	c.	d.	e.	f.	g. Covers all user site installation activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the installation
Software Transition Plan (5.1.5)	a. STrP DID*	b.	c.	d.	e.	f.	g. Covers all transition-related activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the transition
Operational Concept (5.3.2)	a. OCD DID*	b.	c.	d.	e.	f.	g. Feasible
System/Segment Requirements (5.3.3, 5.13.7)	a. SSS IRS DIDs*	b.	c.	d.	e.	f.	g. Covers the operational concept h. Feasible i. Testable
System-Wide Design Decisions (5.4.1)	a. SSDD IDD DBDD DIDs*	b.	c.	d.	e.	f.	g. Consistent with system requirements h. Feasible
System Architectural Design (5.4.2)	a. SSDD IDD DIDs*	b.	c.	d.	e.	f.	g. Covers the system requirements h. Consistent with the system-wide design decisions i. Feasible

Figure D.3-1 Software products and associated evaluation criteria

* As described in the SDP

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets contract, if applic.	Understandable	Intern. consistent	Follows SDP	Additional Criteria
Software Requirements (5.5)	a. SRS IRS DIDs*	b.	c.	d.	e.	f.	g. Covers system requirements allocated to the software item h. Feasible i. Testable
Software Item-Wide Design Decisions (5.6.1)	a. SDD IDD DBDD DIDs*	b.	c.	d.	e.	f.	g. Consistent with software item requirements h. Feasible
Software Item Architectural Design (5.6.2)	a. SDD IDD DIDs*	b.	c.	d.	e.	f.	g. Covers software item requirements h. Consistent with software item-wide design decisions i. Feasible
Software Item Detailed Design (5.6.3)	a. SDD IDD DBDD DIDs*	b.	c.	d.	e.	f.	g. Covers software item requirements allocated to each unit h. Consistent with software item-wide design decisions
Software Implementation (5.7.1)	a. N/A	b.	c.	d.	e.	f.	g. Covers the software item detailed design
Software Item Qualification Test Descriptions (5.9.3)	a. STD DID*	b.	c.	d.	e.	f.	g. Covers all software item requirements
Software Item Qualification Test Results (5.9.7)	a. STR DID*	b.	c.	d.	e.	f.	g. Covers all planned software item qualification test cases h. Shows evidence that the software item meets its requirements
Software Item Qualification Test Descriptions (5.11.3)	a. STD DID*	b.	c.	d.	e.	f.	g. Covers all system requirements
Systems Qualification Test Results (5.11.7)	a. STR DID*	b.	c.	d.	e.	f.	g. Covers all planned system qualification test cases h. Shows evidence the system meets its requirements
Executable Software (5.12.1, 5.13.1)	a. N/A	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All software necessary for execution is present i. Version exactly matches version that passed testing j. Deliverable media accurately labeled
Software Version Descriptions (5.12.2, 5.13.3)	a. SVD DID*	b.	c.	d.	e.	f.	g. Accurately identifies the version of each software component (file, unit, software item, etc.) delivered h. Accurately identifies the changes incorporated

Figure D.3-1 Software products and associated evaluation criteria – continued

* As described in the SDP

Software Product	Evaluation Criteria						
	Contains all applic. info in:	Meets SOW, if applic.	Meets contract, if applic.	Understandable	Intern. consistent	Follows SDP	Additional Criteria
Software User Manuals (5.12.3.1)	a. SUM DID*	b.	c.	d.	e.	f.	g. Accurately describes software installation and use to the intended audience of this manual
Computer Operation Manuals (5.12.3.4)	a. COM DID*	b.	c.	d.	e.	f.	g. Accurately describes the operational characteristics of the computer
Source Files (5.13.2)	a. SPS DID*	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All required software is present i. Version exactly matches version that passed testing j. Deliverable media accurately labeled
“As built” Software Item Design and Related Information (5.13.4)	a. SPS DID*	b.	c.	d.	e.	f.	g. Accurately describes the “as built” design of the software item h. Accurately describes compilation/build procedures i. Accurately describes modification procedures j. Source files cover all units in the software item design k. Measured resource utilization meets software item requirements
“As built” System/Segment Design (5.13.5)	a. SSDD DID*	b.	c.	d.	e.	f.	g. Accurately describes the “as built” system design
Computer Programming Manuals (5.13.8.1)	a. CPM DID*	b.	c.	d.	e.	f.	g. Accurately describes the programming features of the computer
Firmware Support Manuals (5.13.8.2)	a. FSM DID*	b.	c.	d.	e.	f.	g. Accurately describes firmware programming features
Sampling of Software Development Files (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)	a. N/A	b.	N/A	d.	e.	f.	g. Contents are current with the ongoing effort h. Adequate unit test cases/procedures/data/results i. Adequate unit integration test cases/procedures/data/results j. Adequate software item qualification dry run results k. Adequate software/hardware item integration test cases/procedures/data/results l. Adequate system qualification dry run results

Figure D.3-1 Software products and associated evaluation criteria - continued

* As described in the SDP

Appendix E. Candidate joint management reviews

E.1 Scope

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

E.2 Assumptions

This appendix makes the following assumptions:

- a) The acquirer has reviewed the subject products in advance, and one or more joint technical reviews have been held to resolve issues, leaving the joint management review as a forum to resolve open issues and reach agreement as to the acceptability of each product.
- b) Any of the reviews may be conducted incrementally, dealing at each review with a subset of the listed items or a subset of the system (e.g., segment, element, subsystem, as applicable) or software item(s) being reviewed.

E.3 Candidate reviews

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

E.3.1 Software plan reviews

These reviews are held to resolve open issues regarding one or more of the following:

- a) Software Development Plan;
- b) Software Test Plan;
- c) Software Installation Plan;
- d) Software Transition Plan.

E.3.2 Operational concept reviews

These reviews are held to resolve open issues regarding the operational concept for the system.

E.3.3 System/Subsystem requirements reviews

These reviews are held to resolve open issues regarding the specified requirements for the system, segment, element, subsystem, etc.

E.3.4 System/Subsystem design reviews

These reviews are held to resolve open issues regarding one or more of the following:

- a) The system-wide/segment-wide/element-wide/subsystem-wide/etc. design decisions;
- b) The architectural design of the system/segment/element/subsystem/etc.

E.3.5 Software requirements reviews

These reviews are held to resolve open issues regarding the specified requirements for a software item.

E.3.6 Software design reviews

These reviews are held to resolve open issues regarding one or more of the following:

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

E.3.7 Test readiness reviews

These reviews are held to resolve open issues regarding one or more of the following:

- a) The status of the system/segment/element/subsystem/etc. test environment;
- b) The test cases and test procedures to be used for software item qualification testing or system/segment/element/subsystem/etc. qualification testing;
- c) The status of the system/segment/element/subsystem/etc. to be tested.

E.3.8 Test results reviews

These reviews are held to resolve open issues regarding the results of software item qualification testing or system/segment/element/subsystem/etc. qualification testing.

E.3.9 Software usability reviews

These reviews are held to resolve open issues regarding one or more of the following:

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

E.3.10 Software maintenance reviews

These reviews are held to resolve open issues regarding one or more of the following:

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

E.3.11 Critical requirements reviews

These reviews are held to resolve open issues regarding the handling of critical requirements, such as those for safety, security, and privacy. (See Section 4.2.5.)

Appendix F. Candidate management indicators

F.1 Scope

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

F.2 Candidate indicators

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

- a) Requirements volatility: total number of requirements and requirement changes over time.
- b) Software size: planned and actual number of units, lines of code, or other size measurement over time.
- c) Software staffing: planned and actual staffing levels over time.
- d) Software complexity: complexity of each software unit.
- e) Software progress: planned and actual number of software units designed, implemented, unit tested, and integrated over time.
- f) Problem/change report status: total number, number closed, number opened in the current reporting period, age, severity.
- g) Build release content: planned and actual number of software units released in each build.
- h) Build release volatility¹³: planned and actual number of software requirements implemented in each build.
- i) Computer hardware resource utilization: planned and actual use of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity) over time.
- j) Milestone performance: planned and actual dates of key project milestones.
- k) Scrap/rework: amount of resources expended to replace or revise software products after they are placed under any level of configuration control above the individual author/developer level.
- l) Effect of reuse: a breakout of each of the indicators above for reused versus new software products.
- m) Cost performance¹³: identifies how efficiently the project team has turned costs into progress to date.
- n) Budgeted cost of work performed¹³: identifies the cumulative work that has been delivered to date.

¹³ This is from *Metrics-based Software Acquisition Management*, Aerospace Report No. TOR-2004(3909)-3405 [1].

Additional information can be found in the *Practical Software Measurement: Objective Information for Decision Makers*, Addison-Wesley, 2001 [10]; *Practical Software and System Measurement*, Version 4.0c, March 2003 [5]; and *Metrics-based Software Acquisition Management*, Aerospace Report No. TOR-2004(3909)-3405 [1].

Appendix G. Guidance on contracting for delivery of software products

G.1 Scope

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

G.2 Contracting for deliverables

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

G.3 Scheduling deliverables

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

G.4 Format of deliverables

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

G.5 Tailoring the content requirements for deliverables

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

Appendix H. Software Development Plan Template

This appendix is a mandatory part of the standard. The numbering shown below in sections 4 and 5 is consistent with the paragraphs in this standard.

Referenced information cited in Paragraphs 4, 5, 6, and 7 shall be provided as attachments to the plan.

1. Scope. This section shall be divided into the following paragraphs.

1.1 Identification. This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

1.2 System overview. This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents.

1.3 Document overview. This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

1.4 Relationship to other plans. This paragraph shall describe the relationship, if any, of the SDP to other project management plans.

2. Referenced documents. This section shall list the number, title, revision, and date of all documents referenced in this plan. This section shall also identify the source for all documents not available through normal Government stocking activities.

3. Overview of required work. This section shall be divided into paragraphs as needed to establish the context for the planning described in later sections. It shall include, as applicable, an overview of:

- a. Requirements and constraints on the system and software to be developed
- b. Requirements and constraints on project documentation
- c. Position of the project in the system life cycle
- d. The selected program/acquisition strategy or any requirements or constraints on it
- e. Requirements and constraints on project schedules and resources
- f. Other requirements and constraints, such as on project security, privacy, methods, standards, interdependencies in hardware and software development, etc.

4. General requirements. This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. In addition to the content specified below, each paragraph shall identify applicable risks/uncertainties and plans for dealing with them.

4.1 Software development process. This paragraph shall describe the software development process to be used. The planning shall cover all contractual clauses concerning this topic and identification of the software developmental life cycle model(s) to be used, including: planned builds, if applicable, their build objectives, and the software development activities to be performed in each build.

4.2 General requirements for software development. This paragraph shall be divided into the following subparagraphs.

4.2.1 Software development methods. This paragraph shall describe or reference the software development methods to be used. Included shall be descriptions of the manual and automated tools and procedures to be used in support of these methods. The methods shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the methods are better described in context with the activities to which they will be applied.

4.2.2 Standards for software products. This paragraph shall describe or reference the standards to be followed for representing requirements, design, code, test cases, test procedures, and test results. The standards shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the standards are better described in context with the activities to which they will be applied. Standards for code shall be provided for each programming language to be used. They shall include at a minimum:

- a. Standards for format (such as indentation, spacing, capitalization, and order of information)
- b. Standards for header comments (requiring, for example, name/identifier of the 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.)
- c. Standards for other comments (such as required number and content expectations)
- d. Naming conventions for variables, parameters, packages, procedures, files, etc.
- e. Restrictions, if any, on the use of programming language constructs or features
- f. Restrictions, if any, on the complexity of code aggregates

4.2.3 Traceability. This paragraph shall describe the approach to be followed for establishing and maintaining bi-directional traceability between levels of requirements, between requirements and design, between design and the software that implements it, between requirements and qualification test information, and between computer hardware resource utilization requirements and measured computer hardware resource utilization.

4.2.4 Reusable software products. This paragraph shall be divided into the following subparagraphs.

4.2.4.1 Incorporating reusable software products. This paragraph shall describe the approach to be followed for identifying, evaluating, and incorporating reusable software products, including the scope of the search for such products and the criteria to be used for their evaluation. It shall cover all contractual clauses concerning this topic. Candidate or selected reusable software products known at the time this plan is prepared or updated shall be

identified and described, together with benefits, drawbacks, and restrictions, as applicable, associated with their use.

4.2.4.2 Developing reusable software products. This paragraph shall describe the approach to be followed for identifying, evaluating, and reporting opportunities for developing reusable software products. It shall cover all contractual clauses concerning this topic.

4.2.5 Assurance of critical requirements. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for handling requirements designated critical. The planning in each subparagraph shall cover all contractual clauses concerning the identified topic.

4.2.5.1 Safety

4.2.5.2 Security

4.2.5.3 Privacy protection

4.2.5.4 Dependability, reliability, maintainability, and availability

4.2.5.5 Assurance of other mission-critical requirements as agreed to by the acquirer and developer

4.2.6 Computer hardware resource utilization. This paragraph shall describe the approach to be followed for allocating computer hardware resources and monitoring their utilization. It shall cover all contractual clauses concerning this topic.

4.2.7 Recording rationale. This paragraph shall describe the approach to be followed for recording rationale that will be useful to the support agency for key decisions made on the project. It shall interpret the term "key decisions" for the project and state where the rationale are to be recorded. It shall cover all contractual clauses concerning this topic.

4.2.8 Access for acquirer review. This paragraph shall describe the approach to be followed for providing the acquirer or its authorized representative access to developer and subcontractor facilities for review of software products and activities. It shall cover all contractual clauses concerning this topic.

5. Plans for performing detailed software development activities. This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. The discussion of each activity shall include the approach (plans, processes, methods, procedures, tools, roles and responsibilities) 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. For each activity, include entrance criteria, inputs, tasks to be accomplished and products to be produced, verifications to be used (to ensure tasks are performed according to their defined processes and products meet their requirements), outputs, and exit criteria. The discussion shall also identify applicable risks/uncertainties and plans for dealing with them. Reference may be made to 4.2.1 if applicable methods are described there.

5.1 Project planning and oversight. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for project planning and oversight. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.1.1 Software development planning

5.1.2 Software item test planning

- 5.1.3 System test planning
- 5.1.4 Planning for software transition to operations
- 5.1.5 Planning for software transition to maintenance
- 5.1.6 Following and updating plans

5.2 Establishing a software development environment. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for establishing, controlling, and maintaining a software development environment. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.2.1 Software engineering environment
- 5.2.2 Software integration and test environment
- 5.2.3 Software development library
- 5.2.4 Software development files
- 5.2.5 Non-deliverable software

5.3 System requirements analysis. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system requirements analysis. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.3.1 Analysis of user input
- 5.3.2 Operational concept
- 5.3.3 System requirements

5.4 System design. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.4.1 System-wide design decisions
- 5.4.2 System architectural design

5.5 Software requirements analysis. This paragraph shall describe the approach to be followed for software requirements analysis. The approach shall cover all contractual clauses concerning this topic.

5.6 Software design. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.6.1 Software item-wide design decisions
- 5.6.2 Software item architectural design
- 5.6.3 Software item detailed design

5.7 Software implementation and unit testing. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software implementation and unit testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.7.1 Software implementation
- 5.7.2 Preparing for unit testing
- 5.7.3 Performing unit testing
- 5.7.4 Revision and retesting
- 5.7.5 Analyzing and recording unit test results

5.8 Unit integration and testing. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for unit integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.8.1 Preparing for unit integration and testing
- 5.8.2 Performing unit integration and testing
- 5.8.3 Revision and retesting
- 5.8.4 Analyzing and recording unit integration and test results

5.9 Software item qualification testing. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software item qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.9.1 Independence in software item qualification testing
- 5.9.2 Testing on the target computer system
- 5.9.3 Preparing for software item qualification testing
- 5.9.4 Dry run of software item qualification testing
- 5.9.5 Performing software item qualification testing
- 5.9.6 Revision and retesting
- 5.9.7 Analyzing and recording software item qualification test results

5.10 Software/hardware item integration and testing. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in software/hardware item integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.10.1 Preparing for software/hardware item integration and testing
- 5.10.2 Performing software/hardware item integration and testing
- 5.10.3 Revision and retesting
- 5.10.4 Analyzing and recording software/hardware item integration and test results

5.11 System qualification testing. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.11.1 Independence in system qualification testing
- 5.11.2 Testing on the target computer system
- 5.11.3 Preparing for system qualification testing
- 5.11.4 Dry run of system qualification testing
- 5.11.5 Performing system qualification testing
- 5.11.6 Revision and retesting
- 5.11.7 Analyzing and recording system qualification test results

5.12 Preparing for software transition to operations. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software use. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.12.1 Preparing the executable software
- 5.12.2 Preparing version descriptions for user sites
- 5.12.3 Preparing user manuals
 - 5.12.3.1 Software user manuals
 - 5.12.3.2 Computer operations manuals

5.12.4 Installation at user sites

5.13 Preparing for software transition to maintenance. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software transition to maintenance. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.13.1 Preparing the executable software
- 5.13.2 Preparing source files
- 5.13.3 Preparing version descriptions for the maintenance site
- 5.13.4 Preparing the "as built" software item design and other related information
- 5.13.5 Updating the system/subsystem design description
- 5.13.6 Updating the software requirements
- 5.13.7 Updating the system requirements
- 5.13.8 Preparing maintenance manuals
 - 5.13.8.1 Computer programming manuals
 - 5.13.8.2 Firmware support manuals
- 5.13.9 Transition to the designated maintenance site

5.14 Software configuration management. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software configuration management. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.14.1 Configuration identification
- 5.14.2 Configuration control
- 5.14.3 Configuration status accounting
- 5.14.4 Configuration audits
- 5.14.5 Packaging, storage, handling, and delivery

5.15 Software peer reviews and product evaluations. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software peer reviews and product evaluations. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.15.1 Software peer reviews
 - 5.15.1.1 Prepare for software peer reviews
 - 5.15.1.2 Conduct peer reviews
 - 5.15.1.3 Analyze peer review data
- 5.15.2 Software product evaluations
 - 5.15.2.1 In-process and final software product evaluations
 - 5.15.2.2 Software product evaluation records
 - 5.15.2.3 Independence in software product evaluations

5.16 Software quality assurance. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software quality assurance. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.16.1 Software quality assurance evaluations
- 5.16.2 Software quality assurance records
- 5.16.3 Independence in software quality assurance
- 5.16.4 Software quality assurance noncompliance issues

5.17 Corrective action. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for corrective action. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.17.1 Problem/change reports

These reports shall include items to be recorded such as: project name, originator, problem number, problem name, software element or document affected, origination date, category and severity, description, analyst assigned to the problem, date assigned, date completed, analysis time, recommended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, problem type (e.g., using Orthogonal Defect Classification), version where corrected, correction time, and description of solution implemented.

5.17.2 Corrective action system

5.18 Joint technical and management reviews. This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for joint technical and management reviews. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.18.1 Joint technical reviews

5.18.2 Joint management reviews

5.19 Risk management. This paragraph shall describe the approach for performing risk management. The planning shall cover all contractual clauses regarding the identified topic.

5.20 Software management indicators. This paragraph shall describe the approach to be used for software measurement throughout the life cycle. This paragraph shall also include the specific software measurements to be used (that is, collected, analyzed, reported, and used for decision making, corrective actions and reporting to the customer), including which measurements will be reported by lifecycle activity (e.g., requirements, design, code, integration, test). For each measurement the elements of the measurement construct as defined Figure A-1 of Practical Software Measurement: Objective Information for Decision Makers [10] shall be completed. The blank Elements of Measurement Construct form is available at the Practical Software and Systems Measurement web site.

5.21 Security and privacy. This paragraph shall describe the approach for meeting the security and privacy requirements. The planning shall cover all contractual clauses regarding the identified topic.

5.22 Subcontractor management. This paragraph shall describe the approach for performing subcontractor management. The planning shall cover all contractual clauses regarding the identified topic.

5.23 Interface with software independent verification and validation (IV&V) agents. This paragraph shall describe the approach for interfacing with the software IV&V agents. The planning shall cover all contractual clauses regarding the identified topic.

5.24 Coordination with associate developers. This paragraph shall describe the approach for performing the coordination with associate developers, working groups, and interface groups. The planning shall cover all contractual clauses regarding the identified topic.

5.25 Improvement of project processes. This paragraph shall describe the approach for performing the improvement of project processes. The planning shall cover all contractual clauses regarding the identified topic.

6. Schedules and activity network. This section shall present:

- a. 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
- b. An activity network, depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project

7. Project organization and resources. This section shall be divided into the following paragraphs to describe the project organization and resources to be applied in each build.

7.1 Project organization. This paragraph shall describe the organizational structure to be used on the project, including the organizations involved, their relationships to one another, and the authority and responsibility of each organization for carrying out required activities.

7.2 Project resources. This paragraph shall describe the resources to be applied to the project. It shall include, as applicable:

- a. Personnel resources, including:
 - 1) The estimated staff-loading for the project (number of personnel over time)
 - 2) The breakdown of the staff-loading numbers by responsibility (for example, management, software engineering, software testing, software configuration management, software product evaluation, software quality assurance)
 - 3) A breakdown of the skill levels, geographic locations, and security clearances of personnel performing each responsibility
 - 4) The rationale for effort, staff loading and schedule estimates, including software cost and schedule estimation techniques, the input to those techniques (e.g., software size and software cost driver parameters), and any assumptions made
- b. Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.
- c. Acquirer-furnished equipment, software, services, documentation, data, and facilities required for the contracted effort. A schedule detailing when these items will be needed shall also be included.
- d. Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

8. Notes. This section shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

A. Appendixes. Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling. Appendixes shall be lettered alphabetically (A, B, etc.).

SMC Standard Improvement Proposal

INSTRUCTIONS

1. Complete blocks 1 through 7. All blocks must be completed.
2. Send to the Preparing Activity specified in block 8.

NOTE: Do not be used to request copies of documents, or to request waivers, or clarification of requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document(s) or to amend contractual requirements. Comments submitted on this form do not constitute a commitment by the Preparing Activity to implement the suggestion; the Preparing Authority will coordinate a review of the comment and provide disposition to the comment submitter specified in Block 6.

SMC STANDARD CHANGE RECOMMENDATION:	1. Document Number	2. Document Date
3. Document Title		
4. Nature of Change (Identify paragraph number; include proposed revision language and supporting data. Attach extra sheets as needed.)		
5. Reason for Recommendation		
6. Submitter Information		
a. Name	b. Organization	
c. Address	d. Telephone	
e. E-mail address	7. Date Submitted	
8. Preparing Activity	Space and Missile Systems Center AIR FORCE SPACE COMMAND 483 N. Aviation Blvd. El Segundo, CA 91245 Attention: SMC/EAE	