

Trial-Use Standard Standard for Information Technology Software Life Cycle Processes Software Development Acquirer-Supplier Agreement

Keywords: Builds/incremental development, Database, Joint Technical/management reviews, Operational concept, Reusable software, Risk management, Security/privacy protection, Software, Software configuration management, Software development, Software documentation, Software implementation, Software management indicators, Software product evaluation, Software quality assurance, Software requirements definition, Software safety, Software maintenance, Software testing, Software unit, Tailoring.

Prepared by a Joint IEEE / EIA Working Group

This is a joint IEEE/EIA Standards Project, subject to change. Permission is hereby granted for IEEE/EIA Standards Committee participants to reproduce this document for purposes of IEEE/EIA standardization activities, including balloting and coordination. If this document is to be submitted to ISO or IEC, notification shall be given to the IEEE/EIA. Permission is also granted for member bodies and technical committees of ISO and IEC to reproduce this document for the purpose of developing an international position. Other entities seeking permission to reproduce portions of this document for these and other uses must contact the IEEE or EIA.

Institute of Electrical and Electronic Engineers, Inc. 345 East 47th Street New York, NY 10017, USA
Electronic Industries Association 2600 Wilson Blvd. Arlington, VA 22201-3834

NOTICE

EIA and IEEE Engineering Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Use of such Standards and Publications is wholly voluntary. Existence of such Standards and Publications shall not in any respect preclude any member or nonmember of EIA and IEEE from manufacturing or selling products not conforming to such Standards and Publications, nor shall the existence of such Standards and Publications preclude their voluntary use by those other than EIA and IEEE members, whether the standard is to be used either domestically or internationally.

Standards and Publications are approved by EIA and IEEE in accordance with the American National Standards Institute (ANSI) patent policy. By such action, EIA and IEEE do not assume any liability to any patent owner, nor do they assume any obligation whatever to parties adopting the Standard or Publication.

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. EIA and IEEE shall not be responsible for identifying all patents for which a license may be required by an EIA and IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

INTERIM (TRIAL-USE) STANDARDS

This EIA/IEEE Interim (Trial-Use) Standard is based upon the major technical content of ISO/IEC 12207 Information Technology Software Life Cycle Process, 22 February 1995, for the software development process. It conforms in all essential respects with this process of the ISO/IEC Standard.

EIA/IEEE Interim (Trial-Use) Standards contain information deemed to be of technical value to the industry, and are published at the request of the originating Committee without necessarily following the rigorous public review and resolution of comments which is a procedural part of the development of an EIA/IEEE Standard.

EIA Interim Standards should be reviewed on an annual basis by the formulating Committee and a decision made on whether to proceed to develop an EIA/IEEE Standard on this subject. EIA Interim Standards must be cancelled by the Committee and removed from the EIA Standards Catalog before the end of their fifth year of existence. In accordance with IEEE policy, trial-use standards are effective for 24 months from the date of publication. Comments for revision will be accepted for 18 months after publication. It is expected that following the 24-month period, this trial-use standard, revised as necessary, shall be submitted to the IEEE Standards Board for approval as a full-use standard.

(This Interim (Trial-Use) Standard was developed by the Joint EIA/IEEE Working Group on Software Development. Members of the Joint Committee serve voluntarily and without compensation. They are not necessarily members of EIA or IEEE.

Comments on standards and requests for interpretation should be addressed to:

Vice President, Standards and Technology
Electronic Industries Association
2500 Wilson Boulevard
Arlington, Virginia 22201-3834

Published by
©ELECTRONIC INDUSTRIES ASSOCIATION 1996

Engineering Department
2500 Wilson Boulevard
Arlington, Virginia 22201

**PRICE: Please refer to the current
Catalog of EIA, JEDEC, and TIA STANDARDS and ENGINEERING
PUBLICATIONS
or call Global Engineering Documents, USA and Canada (1-800-854-7179)
International (303-397-7956)**

All rights reserved
Printed in U.S.A.

CAUTION—PLEASE!

DON'T VIOLATE THE LAW!

This document is copyrighted by the EIA and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies through entering into a license agreement. For information, contact:

Global Engineering Documents 15 Inverness Way East Englewood, CO 80112-5704 or call U.S.A. and Canada 1-800-854-7179, International (303) 397-7956

Foreword

1. This standard defines a set of software development activities and resulting software products. It provides a framework for software development planning and engineering.
2. This standard is written in contractual language. This choice is based on the fact that it is easier for non-contractual users to adapt contractual language to their use than for contractual users to apply non-contractual language.
3. This standard is the first step in the implementation of ISO/IEC 12207, Software Life Cycle Processes. Future versions are intended to implement additional software life cycle processes and supplier roles, such as the operator and maintainer. Those activities of the maintainer that are the same as those of a developer are covered in this version of the standard. For purposes of this implementation, the “supplier” of ISO/IEC 12207 is always the “developer,” and the term “developer” is used throughout this standard.
4. This standard is meant to be tailored to ensure that only necessary and cost-effective requirements are applied. It is not intended to specify or discourage the use of any particular software development method. The developer is responsible for selecting software development methods that support the achievement of contract requirements.
5. This standard emphasizes that information resulting from the activities required by this standard is intrinsic to the software development process, to be recorded regardless of whether a deliverable is required. When information is not required to be delivered, the software products cited in this standard are to be used as checklists of items to be covered, as applicable, in the planning and engineering activities for the project.
6. This standard allows information to be recorded in representations other than traditional documents (for example, computer-aided software engineering (CASE) tools) and provides relaxed requirements for the format and structure of that information for such representations.

Introduction

J-STD-016-1995 is a standard for the software development process. It is applicable throughout the system life cycle. It establishes uniform requirements for acquiring, developing, modifying, and documenting software. It defines standard terminology and establishes activities, tasks, and products for a software development or maintenance project. It can be applied to any type of software, including application software, operating system software, the software portion of firmware, reusable software, and software employed to develop deliverable software.

The activities in the standard form a comprehensive set, sufficient for a large, complex project. The standard is meant to be tailored as needed for each project by specifying in the contract which provisions of the standard apply.

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

- The activities and tasks in the standard tell what to do, not how to do it. For example, the standard requires the developer to perform architectural design, but does not require use of the object-oriented design method.
- The standard does not specify or encourage any software life cycle model (waterfall, incremental, evolutionary, spiral, etc.). Instead, the standard provides the building blocks needed to create a life-cycle model for a software project.
- The standard does not specify or depend on any design or programming language. It is meant to be applicable regardless of the language used.
- The standard provides flexibility regarding documentation:
 - A distinction is made between the task of generating and recording planning or engineering information and the task of generating a deliverable containing that information.
 - Information and deliverables can be in hardcopy form (using contractor or software product format) or in computer-aided software engineering (CASE) tools.
- The standard does not emphasize any particular software quality factor, such as reliability, maintainability, or reusability. This choice is left to each project.
- The standard can be used within an organization or contractually between two parties.
- The standard provides a performance-based distinction between requirements for a product and design of a product. A requirement is a characteristic that a system or software item is required to possess in order to be acceptable to the acquirer while design is a characteristic that the acquirer is willing to leave up to the developer. This emphasis on required performance can allow a developer to provide quality products at reduced costs when unique product solutions are not required.

J-STD-016-1995 specifically removes key problems found in the use of predecessor standards. J-STD-016-1995:

a) **Is usable with any development strategy** . J-STD-016-1995 has been structured to better accommodate incremental, evolutionary, and other development models than the traditional “waterfall” model. It is structured to avoid time-oriented dependencies and implications, provides alternatives to formal reviews (that can force a waterfall development model), and explains how to apply the standard across multiple builds or iterations.

b) **Is usable with any development methods** . To improve compatibility with object-oriented and other methods not using functional decomposition, J-STD-016-1995 allows the developer to define in the software development plan the design and implementation structures to be used, and to use that structure in the documentation.

c) **Is compatible with CASE tools** . J-STD-016-1995 recognizes that much of the significant work on a software development project does not involve writing documents. The standard: 1) separates planning and engineering activities from preparation of deliverables to make it easier to call for one without the other; 2) provides language that permits the recording of project information in forms other than traditional documents, such as CASE tools; 3) acknowledges data in CASE tools as a substitute for traditional documents; and 4) provides guidance to prevent unnecessary deliverables.

d) **Provides requirements for software reuse** . J-STD-016-1995 recognizes that many projects incorporate or are based on reusable software. To provide clearer requirements when this is the case, the standard: 1) identifies criteria for use in evaluating reusable software, and 2) provides guidance on interpreting J-STD-016-1995's activities and deliverables when applied to reusable software.

e) **Supports the use of software management indicators** . J-STD-016-1995 supports the use of software management indicators. It has incorporated a requirement to identify and apply such indicators, and included an annex of candidate indicators that might be used.

f) **Emphasizes software supportability** . J-STD-016-1995 has enhanced software supportability requirements. Examples are requirements to: 1) record the rationale for key decisions made on the project; 2) identify all resources the maintenance organization will need to maintain the software, and 3) demonstrate that those resources are sufficient to maintain the software.

g) **Provides a role for software in the system** . J-STD-016-1995 recognizes that software exists in the context of a system. It provides a role for the software developer both in the case of a hardware-software system and where the software (possibly with the computers on which it will run) is considered to constitute the system.

Participants

At the time this trial use standard was completed, the Joint Industry Working Group on Software Development had the following membership:

Perry DeWeese, (EIA Co-Chair)
Raghu Singh, (IEEE Co-Chair)

Stuart Campbell
John Chihorek
Richard Evans
Lewis Gray
Lewis Gray
Robert Hegland
Jim Heil

Helmut Hummel
Tim Janes
Capt Jonathan Liles
Marv Lubofsky
David Maibor

James Moore
Myrna Olson
Al Peschel
Alex Polack
Norma Stopyra
Leonard Tripp

Other individuals who have contributed review and comments are the following:

Dennis Ahern
Mordechai Ben-Menachim
Dave Benson
Ron Berlack
William J. Boll
John Bowers
Don Calvert
Andrew Campbell
Jaya R. Carl
Theo Clarke
Jack Cooper
Paul Cooley
Rita Costello
Kenneth Costello
Geoff Cozens
Gregory Daich
Neil Davis
Chris Denham
Bostjan K. Derganc
Dave Dikel
Treasure Diehl
Peter Eirich
Bob Elston
SueEllen Eslinger
Eva Freund
William Gess
Gregg Glesler
Lawrence Gunther
Ranwa Haddad
John Hamlin
Nancy Hannon
John Harauz

Robert Harley
Rick Hefner
Carl Hirshfield
John Horch
Stephen Huffman
George Jackelen
Allan Jaworski
Diana Kang
Ron Kenett
Judy Kerner
John Kerr
Larry Klos
Rick Kreke
Dorothy Kuckuck
Jerome Lake
Dennis Lawrence
Jim Longbucco
Joan Lovelace
Stan Magee
John Marshall
Judy McCloskey
P. D. McCray
Jack McGarry
Archibald McKinley
Celia Modell
Dennis Nickle
Bart Nigro
James Orr
Gerald L. Ourada
Mark Paulk
Sherry Paquin
John G. Phippen

Peter Poon
Lawrence Przybylski
Ken Ptack
Jane Radatz
Larry Reed
Dennis Rilling
Keith Shewbridge
Katsutoshi Shintani
Carl A. Singer
Melford E. Smyre
Don Sova
Richard Storch
Richard Summers
David J. Szymanski
Booker T. Thomas
Patricia Trellue
Ann Turner
Dennis Turner
Theodore Urbanowicz
Howard Vern
Delores Wallace
Steven Wang
Scott Whitmire
Charles Wilson
Audrey Winston
Paul Wolfgang
Paul Work
Grady Wright
Gene Yancy
Natalie Yopconka
Geraldine Zimmerman
Peter Zoll

CONTENTS

1. Scope.....	1
1.1 Purpose	1
1.2 Application	1
2. Referenced documents	4
3. Definitions.....	4
3.1. Terms	4
4. General requirements	8
4.1 Software development process	8
4.2 General requirements for software development.....	8
5. Detailed requirements	10
5.1 Project planning and oversight	10
5.2 Establishing a software development environment	13
5.3 System requirements definition	14
5.4 System design	15
5.5 Software requirements definition	15
5.6 Software design	15
5.7 Software implementation and unit testing	16
5.8 Unit integration and testing	17
5.9 Software item qualification testing	18
5.10 Software/hardware item integration and testing	19
5.11 System qualification testing.....	20
5.12 Preparing for software use	21
5.13 Preparing for software transition	22
5.14 Software configuration management	24
5.15 Software product evaluation	25
5.16 Software quality assurance	25
5.17 Corrective action.....	26
5.18 Joint technical and management reviews	26
5.19 Risk management	27
5.20 Software management indicators	27
5.21 Administrative security and privacy protection.....	28
5.22 Managing subcontractors.....	28
5.23 Interfacing with software IV&V agents	28
5.24 Coordinating with associate developers	28
5.25 Project process improvement.....	28
6. Notes	28
6.1 Cross reference of standard subclauses to annex subclauses	28
6.2 Delivery of tool contents	29
Annex A (normative) Tailoring this standard for a project	30
A.1 Scope.....	30
A.2 Tailoring process.....	30
A.3 Identifying the project environment	30
A.4 Soliciting input.....	30
A.5 Selecting processes, activities, and tasks	30
A.6 Specifying tailoring decisions.....	31
Annex B (informative)Guidance on development strategies, tailoring, and build planning	32
B.1 Scope	32
B.2 Candidate development strategies.....	32
B.3 Selecting an appropriate development strategy	33

B.4 Relationship of this standard to development strategies	33
B.5 Planning software builds and tailoring this standard.....	33
Annex C (informative) Guidance on ordering deliverables	41
C.1 Scope	41
C.2 Ordering deliverables	41
C.3 Scheduling deliverables.....	41
C.4 Format of deliverables.....	41
C.5 Tailoring the content requirements for deliverables	42
Annex D (informative) Interpreting this standard for incorporation of reusable software products	43
D.1 Scope	43
D.2 Evaluating reusable software products.....	43
D.3 Interpreting this standard's activities for reusable software products.....	43
Annex E (normative) Planning software product descriptions	46
E.1 Scope	46
E.2 Software products covered	46
Annex F (normative) Concept and requirements software product descriptions	63
F.1 Scope.....	63
F.2 Software products covered.....	63
Annex G (normative) Design software product descriptions.....	83
G.1 Scope.....	83
G.2 Software products covered.....	83
Annex H (normative) Qualification testing software product descriptions	100
H.1 Scope	100
H.2 Contents of the Software Test Description (STD)	100
Annex I (normative) Maintenance software product descriptions	106
I.1 Scope.....	106
I.2 Software products covered	106
Annex J (normative) User/operator software product descriptions	115
J.1 Scope	115
J.2 Software products covered	115
Annex K (informative) Software product description format and structure	129
K.1 Scope.....	129
K.2 Software products covered.....	129
K.3 Format and structure for all software products	129
K.3.3 Title page or identifier.....	133
K.3.4 Table of contents and index	133
K.3.5 Page numbering/labeling.....	133
K.3.6 Response to tailoring instructions	133
K.3.7 Multiple clauses and their subordinates	133
K.3.8 Alternative presentation styles	133
K.3.9 Substitution of existing documents	133
K.3.10 Standard data descriptions.....	134
K.3.11 Applicability of required contents	134
Annex L (normative) Software product evaluations	135

L.1 Scope	135
L.2 Required evaluations	135
L.3 Criteria definitions.....	135
Annex M (normative) Category and priority classifications for problem reporting.....	142
M.1 Scope	142
M.2 Classification by category	142
M.3 Classification by priority	142
Annex N (informative) Candidate joint management reviews	144
N.1 Scope.....	144
N.2 Assumptions.....	144
N.3 Candidate reviews.....	144
Annex O (informative) Candidate management indicators	147
O.1 Scope	147
O.2 Candidate indicators	147
Annex P (informative) Questionnaire on use of this standard	148

Trial-Use Standard Standard for Information Technology Software Life Cycle Processes Software Development Acquirer-Supplier Agreement

1. Scope

1.1 Purpose

This standard establishes uniform requirements for software development activities and resulting software products. It is also intended to merge commercial and Government software development requirements within the framework of the software life cycle process requirements of the Electronic Industries Association (EIA), Institute of Electrical and Electronics Engineers (IEEE), and International Organization for Standardization (ISO). The term “software development” is used as an inclusive term encompassing new development, modification, reuse, reengineering, maintenance, and all other processes or activities resulting in software products.

1.2 Application

The requirements in this standard are intended for application to the development of systems that contain software (such as software embedded in hardware-software systems), to systems consisting of software alone, and to stand-alone software products. This standard is intended to be applied as follows.

1.2.1 Candidate uses

This standard can be:

- a) Included or referenced in contracts or similar agreements, with the parties (called the acquirer and the developer) agreeing that the developer will perform software development in accordance with the standard
- b) Adopted as an in-house standard by a project or organization that decides to perform software development in accordance with the standard
- c) Used as the basis for an organization’s software development process

1.2.2 Organizations and agreements

This standard can be applied to contractors, subcontractors, or in-house organizations performing software development. For uniformity, the term “acquirer” is used for the organization requiring the technical effort, “developer” for the organization performing the technical effort, “contract” for the agreement between these parties, “Statement of Work” (SOW) for the list of tasks to be performed by the developer, “subcontractor” for any organization tasked by the developer to perform part of the required effort, “user” for the persons or

organizations that will operate and/or use the software, and “maintenance organization” for the organization that will have responsibility for modifying and otherwise maintaining the software after transition from the development organization.

1.2.3 Contract-specific application

When this standard is invoked in a contract, it applies to each software product and to each type of software covered by the contract, regardless of storage medium, to the extent specified in the contract. Examples of types of software include deliverable versus non-deliverable, software designed to meet user needs versus software in the engineering and test environments, and software designed to meet one user need versus software designed to meet another. The acquirer is expected to specify the types of software to which the standard applies. Software installed in firmware is subject to all of the aforementioned provisions. This standard does not apply to the hardware element of firmware.

1.2.4 In-house application

When this standard is used to assist an organization in defining and establishing its software development process, it is intended for reference as a source of recommended software development activities and products on which to build specific organizational practices and procedures. When the standard is applied on an in-house development project, it is intended to be used as a source for selecting software development activities and products to satisfy specific project needs.

1.2.5 Tailoring

This standard is meant to be tailored for each type of software to which it is applied. The standard is to be tailored to the specific requirements of a particular program, program phase, or contractual structure. Tailoring takes the form of deletion of unnecessary requirements, alteration of requirements to more explicitly reflect the application to a particular effort, or addition of requirements as needed. Care is to be taken to eliminate tasks and data that add unnecessary costs or that do not add value to the activity or the product for the project. The tailoring requirements of this standard are normative (that is, binding) to be performed in accordance with this standard. Tailoring requirements, specified in annex A, are not subject to tailoring. All other requirements of this standard are considered tailorable. While final tailoring decisions are the responsibility of the acquirer, tailoring may be performed jointly with prospective or selected developers. Tailoring guidance can be found in annexes B and C.

1.2.6 Compliance

Compliance with this standard means “compliance with the standard as tailored for the project and recorded in the contract.”

NOTE—Regulatory and legal requirements are invoked separately through the contract.

1.2.6.1 Compliance with the “as tailored” standard

Compliance with the “as tailored” standard is defined as:

- a) Performing the activities that resulted from tailoring this standard for a specific project as recorded in the contract
- b) Recording applicable information resulting from the performance of the selected activities

1.2.6.2 Activity completion criteria

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

1.2.6.3 Compliance stipulation

Any entity (such as, person, company, Military Service, industrial association, organization) imposing this standard as a condition of trade is responsible for identifying and making public the minimum set of activities and resulting software products that is required to be fulfilled by a developer.

1.2.7 Interpretation of selected terms

The following terms have a special interpretation as used in this standard. These are not “definitions” of the terms, but clarifications of their use.

1.2.7.1 Interpretation of “system”

The following interpretations apply:

- a) The term “system,” as used in this standard, may mean: (1) a hardware-software system (for example, a radar system) for which this standard covers only the software portion, or (2) a software system (for example, a payroll system) for which this standard governs overall development.
- b) If a system consists of subsystems, all requirements in this standard concerning systems apply to the subsystems as well. If a contract is based on alternatives to systems and subsystems, the requirements in this standard concerning the system and its specification apply to these alternatives and their specifications.

1.2.7.2 Interpretation of “participate” in system development

The term “participate” in subclauses regarding system-level activities is to be interpreted as follows: if the software covered by this standard is part of a hardware-software system, the term “participate” is to be interpreted as “take part in, as described in the Software Development Plan.” If the software (possibly with its computers) is considered to constitute a system, the term “participate” is to be interpreted as “be responsible for.” If the software in a hardware-software system is changed, but the hardware is not, the term “participate” may also be interpreted as “be responsible for.”

1.2.7.3 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.

1.2.7.4 Interpretation of “record”

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

1.2.7.5 Interpretation of “applicable”

Throughout this standard, requirements to provide “applicable” information and perform “applicable” activities are to be interpreted to mean “provide the information and perform the activities that are natural by-products created by performing the activities required for the project in accordance with the tools, methods, and procedures described in the software development plan.” The term “applicable” conveys the meaning that not all projects will generate the information or require the activity. “Applicability” is to be agreed upon jointly by the acquirer and developer.

2. Referenced documents

This standard shall be used in conjunction with the following publications.

FIPS PUB 39, Glossary for Computer Systems Security

IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology/(ANSI)

3. Definitions

NOTE—In addition to the definitions and acronyms provided here, clause 1 describes this standard's interpretation or special usage of the following terms: acquirer, applicable, contract, define, develop, developer, establish, identify, participate, record, software development, Statement of Work, subcontractor, subsystem, maintenance organization, system, and user.

3.1. Terms

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

3.1.2 acquirer: An organization that procures software products for itself or another organization.

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

3.1.4 architecture: The organizational structure of a system or a software item, identifying its components, their interfaces, and a concept of execution among them.

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

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

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

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

3.1.8 computer database: *See* : database.

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

3.1.10 computer program: A combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions. (Source IEEE Std 610.12-1990).

3.1.11 computer software: *See:* software.

3.1.12 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. (Adapted from IEEE Std 610.12-1990).

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

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

3.1.15 design: Those characteristics of a system or a 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.

3.1.16 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, and includes the testing, quality assurance, configuration management, and other activities applied to these products).

3.1.17 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. (Adapted from IEEE Std 610.12-1990).

3.1.18 evaluation: The process of determining whether an item or activity meets specified criteria.

3.1.19 firmware: The combination of a hardware device and computer instructions and/or computer data that reside as read-only software on the hardware device. (Adapted from IEEE Std 610.12-1990).

3.1.20 hardware item: An aggregation of hardware that is designated for purposes of specification, testing, interfacing, configuration management, or other purposes.

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

3.1.22 interface: In software development, a relationship among two or more entities (such as software item - software item, software item - hardware item, software item - user, or software unit - software unit) 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.

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

3.1.24 maintenance (of software): *See:* software maintenance.

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

3.1.26 privacy protection: The establishment of appropriate administrative, technical, and physical safeguards to ensure the security and confidentiality of data records and to protect both security and confidentiali-

ality against any anticipated threats or hazards that could result in substantial harm, embarrassment, inconvenience or unfairness to any individual about whom such information is maintained. (Source FIPS PUB 39)

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

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

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

3.1.30 requirement: (1) A characteristic that a system or a software item is required to possess in order to be acceptable to the acquirer. (2) A binding statement in this standard or another portion of the contract. Requirements are expressed using the word "shall."

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

3.1.32 software: Computer programs and computer databases.

NOTE—Although some definitions of software include documentation, this standard limits the definition to computer programs and computer databases.

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

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

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

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

3.1.37 software item: An aggregation of software, such as a computer program or database, that satisfies an end use function and is designated for purposes of specification, qualification testing, interfacing, configuration management, or other purposes. Software items are selected based on trade-offs among software func-

tion, size, host or target computers, developer, support strategy, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors. A software item is made up of one or more software units.

3.1.38 software maintenance: 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 improvements, aid to users, and related activities.

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

3.1.40 software quality: The ability of software to satisfy its specified requirements.

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

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

3.1.43 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.

3.1.44 transition (of software): *See:* software transition.

3.2 Abbreviations and acronyms

CASE	Computer-Aided Software Engineering
COM	Computer Operation Manual
CPM	Computer Programming Manual
DBDD	Database Design Description
FSM	Firmware Support Manual
HW	Hardware
IDD	Interface Design Description
IRS	Interface Requirements Specification
IV&V	Independent Verification and Validation
OCD	Operational Concept Description
SCOM	Software Center Operator Manual
SDD	Software Design Description
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan
SIOM	Software Input/Output Manual
SIP	Software Installation Plan
SOW	Statement of Work
SPS	Software Product Specification
SRS	Software Requirements Specification
SSDD	System/Subsystem Design Description
SSS	System/Subsystem Specification
STD	Software Test Description
STP	Software Test Plan

STR	Software Test Report
STrP	Software Transition Plan
SUM	Software User Manual
SVD	Software Version Description
SW	Software

4. General requirements

4.1 Software development process

The developer shall establish a software development process consistent with contract requirements. The software development process shall include the following major activities, which may overlap, may be applied iteratively, may be applied differently to different elements of software, and need not be performed in the order listed below. The software process established may use or reference an organization's existing process for performing these activities. Annex B provides examples. The developer's software development process shall be described in the Software Development Plan.

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

4.2 General requirements for software development

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

4.2.1 Software development methods

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

4.2.2 Standards and practices for software products

The developer shall establish and apply standards and practices for representing requirements, design, code, test cases, test procedures, and test results. These standards and practices shall be described in, or referenced from, the Software Development Plan.

4.2.3 Traceability

The developer shall perform 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 all applicable traceability items of the System/Subsystem Specification (see F.2.2), Interface Requirements Specification (see F.2.3), Software Requirements Specification (see F.2.4), System/Subsystem Design Description (see G.2.1), Interface Design Description (see G.2.2), Database Design Description (see G.2.3), Software Design Description (see G.2.4), Software Test Plan (see E.2.2), Software Test Description (see H.2.1), and Software Product Specification (see I.2.1).

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 (e.g., commercial off-the-shelf, acquirer-furnished, software products in reuse libraries, and pre-existing developer software products), that meet specified requirements and are cost-effective over the life of the system, for use in fulfilling the requirements of the contract. The scope of the search and the criteria to be used for evaluation and incorporation shall be as described in the Software Development Plan. Annex D provides candidate evaluation criteria and interprets this standard for incorporation of reusable software products.

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

4.2.5 Assurance of safety, security, privacy protection, or other critical requirements

If a system relies on software to satisfy requirements deemed critical by the contract or by system specifications, the developer shall identify those software items or portions thereof whose failure could lead to violation of those critical requirements; develop a strategy, including both test and analyses, to assure that the requirements, design, implementation, and operating procedures for the identified software minimize or eliminate the potential for such violations; record the strategy in the Software Development Plan; implement the strategy; and produce evidence, as part of the required software products, that the assurance strategy has been carried out. The developer shall identify by type, and develop strategies for, the following types of critical requirements:

- a) Safety-critical: 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-critical: those software items or portions thereof whose failure could lead to a breach of system security
- c) Privacy protection-critical: those software items or portions thereof whose failure could lead to a breach of system privacy protection

NOTE—For purposes of this standard, privacy protection is concerned with preventing unauthorized disclosure of confidential data such as salary information; security is concerned with any accidental or malicious modification, destruction, or disclosure of computer software or computer data.

4.2.6 Computer hardware resource utilization.

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

4.2.7 Recording rationale

The developer shall record rationale that will be useful to the maintenance organization 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. The meaning of “key decisions” and the approach for providing the rationale shall be described in the Software Development Plan.

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 required by the contract.

5. Detailed requirements

The order of the requirements in this clause is not intended to specify the order in which they are required to be carried out. Some activities may not apply to the project or the 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 subclauses may depend on input from activities in later subclauses. If the software is developed in multiple builds, some activities may be performed in every build, others may be performed only in selected builds, and activities and software products may not be complete until several or all builds are accomplished. Figure 1 provides an example of how each activity may be applied in one or more builds. Non-binding notes throughout clause 5 explain 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. Annex B provides guidance for planning builds, determining which activities apply to each build, and scheduling these activities.

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 is interpreted to include: a) overall planning for the contract, b) detailed planning for the current build, and c) planning for future builds covered under the contract to a level of detail compatible with the information available.

5.1.1 Software development planning

The developer shall develop and record plans for conducting the activities required by this standard and by other software-related requirements in the contract. This planning shall be consistent with system-level planning and shall include all applicable items in the Software Development Plan (SDP) (see E.2.1).

NOTES:

1—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 software products in annexes E through J as checklists of items to be covered in the planning and engineering activities; and 3) Permit representations other than traditional documents for recording the information (for example, computer-aided software engineering (CASE) tools).

2—If the contract specifies delivery of the information generated by this or any other subclause, the developer is required to format, assemble, mark, copy, and distribute the deliverable in accordance with the contract. This task is recognized to be separate from the task of generating and recording the required information and to require additional time and effort on the part of the developer. Annex K provides information on format and structure of deliverable information.

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

4—The requirement to develop and record software development planning information may be satisfied by referencing an organization's existing practices and procedures that comply with the requirements of the contract.

Activity	Builds			
	Build 1	Build 2	Build 3	Build 4
5.1 Project planning and oversight	x	x	x	x
5.2 Establishing a software development environment	x	x	x	x
5.3 System requirements definition	x	x		
5.4 System design	x	x	x	
5.5 Software requirements definition	x	x	x	x
5.6 Software design	x	x	x	x
5.7 Software implementation and unit testing	x	x	x	x
5.8 Unit integration and testing	x	x	x	x
5.9 Software item qualification testing		x	x	x
5.10 Software/hardware item integration and testing		x	x	x
5.11 System qualification testing			x	x
5.12 Preparing for software use	x	x	x	x
5.13 Preparing for software transition				x
Integral processes:				
5.14 Software configuration management	x	x	x	x
5.15 Software product evaluation	x	x	x	x
5.16 Software quality assurance	x	x	x	x
5.17 Corrective action	x	x	x	x
5.18 Joint technical and management reviews	x	x	x	x
5.19 Risk management	x	x	x	x
5.20 Software management indicators			x	x
5.21 Administrative security and privacy protection				
5.22 Managing subcontractors	x	x		
5.23 Interfacing with software IV&V agents				
5.24 Coordinating with associate developers				
5.25 Project process improvement	x	x	x	

Figure 1 — One possible mapping of this standard's activities to multiple builds

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) (see E.2.2).

5.1.3 System test planning

The developer shall participate in developing and recording plans for conducting system qualification testing. For software systems, this planning shall include all applicable items in the Software Test Plan (STP) (see E.2.2). (The intent for software systems is a single Software Test Plan covering both software item and system qualification testing.)

5.1.4 Software installation planning

The developer shall develop and record plans for performing software installation and training at the user sites specified in the contract. This planning shall include all applicable items in the Software Installation Plan (SIP) (see E.2.3).

5.1.5 Software transition planning

The developer shall identify all software development resources that will be needed by the maintenance organization to fulfill the support strategy 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) (see E.2.4).

5.1.6 Following and updating plans

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

5.2 Establishing a software development environment

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

NOTE—If a system or 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 (e.g., the facilities, hardware, software, firmware, procedures, and documentation, such as computer-aided software engineering (CASE) tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and databasemanagement systems) to perform the software engineering effort. The developer shall ensure that each element of the environment performs its intended functions prior to its use on the project.

5.2.2 Software test environment

The developer shall establish, control, and maintain a software test environment (e.g., the facilities, hardware, software, firmware, procedures, and documentation, such as simulators, code analyzers, test case generators, and path analyzers, and possibly other items of the software engineering environment) to perform qualification, and possibly other, testing of software. The developer shall ensure that each element of the environment performs its intended functions prior to its use on the project.

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 for the duration specified in the contract.

5.2.4 Software development files

The developer shall establish, control, and maintain a software development file (SDF) for each software unit or logically related group of software units, for each software item, 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 for the duration specified in the contract.

5.2.5 Non-deliverable software

The developer may use non-deliverable software in the development of deliverable software provided: 1) the operation and maintenance of the deliverable software after delivery to the acquirer do not depend on the non-deliverable software, or 2) the acquirer has or can obtain the same software. The developer shall ensure that all non-deliverable software used on the project performs its intended functions prior to its use on the project.

5.3 System requirements definition

The developer shall participate in system requirements definition 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 definition for a given build is interpreted to mean defining the system requirements so identified for that build.

5.3.1 Analysis of user input

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

NOTE—The intent of this requirement is to ensure that all interested parties, (such as user, acquirer, developer, and maintenance and test organizations), maintain on-going communications regarding user needs throughout the development of the system.

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) (see F.2.1).

5.3.3 System requirements

Based on analysis of user needs, 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) (see F.2.2). Depending on contract provisions, requirements concerning system interfaces may be included in the SSS or in Interface Requirements Specifications (IRSs).

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

5.4 System design

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

NOTE—If a system is developed in multiple builds, its design may not be fully defined until the final build. The developer's planning 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 clause of the System/Subsystem Design Description (SSDD) (see G.2.1). Depending on contract provisions, design pertaining to interfaces may be included in the SSDD or in Interface Design Descriptions (IDDs), and design pertaining to databases may be included in the SSDD or in Database Design Descriptions (DBDDs).

NOTE—Design decisions act as developer-internal "requirements," to be implemented, imposed on subcontractors, if applicable, and confirmed by developer-internal testing, but their fulfillment need not be demonstrated to the acquirer.

5.4.2 System architectural design

The developer shall participate in defining and recording the architectural design of the system (identifying the components of the system, their interfaces, and a concept of execution among them) and the traceability between the system components and system requirements. The result shall include all applicable items in the architectural design and traceability clauses of the System/Subsystem Design Description (SSDD) (see G.2.1). Depending on contract provisions, design pertaining to interfaces may be included in the SSDD or in Interface Design Descriptions (IDDs), and design pertaining to databases may be included in the SSDD or in Database Design Descriptions (DBDDs).

NOTE—For brevity, this clause is written in terms of organizing a system into components. However, this should be interpreted to cover organizing a system into subsystems, organizing a subsystem into hardware items, software items, and manual operations, or other variations as appropriate.

5.5 Software requirements definition

Based on analysis of system requirements, system design, and other considerations, the developer shall define, and record the software requirements to be met by each software item, the methods to be used to ensure that each requirement has been met, and the traceability between the software item requirements and system requirements. The result shall include all applicable items in the Software Requirements Specification (SRS) (see F.2.4). Depending on contract provisions, requirements concerning software item interfaces may be included in SRSs or in Interface Requirements Specifications IRSs).

NOTE—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 definition for a given build is interpreted to mean defining the software item requirements so identified for that build.

5.6 Software design

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

NOTE—If a 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 clause of the Software Design Description (SDD) (see G.2.4). Depending on contract provisions, design pertaining to interfaces may be included in SDDs or in Interface Design Descriptions (IDDs), and design pertaining to databases may be included in SDDs or in Database Design Descriptions (DBDDs).

5.6.2 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 execution 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 clauses of the Software Design Description (SDD) (see G.2.4). Depending on contract provisions, design pertaining to interfaces may be included in SDDs or in Interface Design Descriptions (IDDs), and design pertaining to databases may be included in SDDs or in Database Design Descriptions (DBDDs).

NOTE—Software units may be made up of other software units and may be organized into as many levels as are needed to represent the 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.

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 clause of the Software Design Description (SDD) (see G.2.4). Depending on contract provisions, design pertaining to interfaces may be included in SDDs or in Interface Design Descriptions (IDDs), and design of software units that are databases or that access or manipulate databases may be included in SDDs or in Database Design Descriptions (DBDDs).

5.7 Software implementation and unit testing

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

NOTE—The term “software” includes both computer programs and computer databases. The term “implementation” means converting software design into computer programs and computer databases. If a software item is developed in multiple builds, software implementation and unit testing of that software item is complete when the final build is complete. Software implementation and unit testing in each build is interpreted to include those units, or parts of units, needed to meet the software item requirements to be implemented in that build.

5.7.1 Software implementation

The developer shall develop and record software corresponding to each software unit in the 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. For deliverable software, the developer shall use the programming language specified in the contract.

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

5.7.2 Preparing for unit testing

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

5.7.3 Performing unit testing

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

5.7.4 Revision and retesting

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

5.7.5 Analyzing and recording unit test results

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

5.8 Unit integration and testing

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

NOTES:

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 clause are not meant to duplicate those activities.

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.

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 the software item-wide and software item architectural design. The developer shall record this information in the appropriate software development files (SDFs).

5.8.2 Performing unit integration and testing

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

5.8.3 Revision and retesting

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

5.8.4 Analyzing and recording unit integration and test results

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

5.9 Software item qualification testing

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

NOTES:

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 (IRSs). This testing contrasts with developer-internal software item testing, performed as the final stage of unit integration and testing.

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.

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 include testing on the target computer system or an alternative system approved by the acquirer.

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 and the software item requirements. The result shall include all applicable items in the Software Test Description (STD) (see H.2.1). 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.

5.9.4 Dry run of software item qualification testing

If software item qualification testing is to be witnessed by the acquirer, 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 software development files (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 qualification testing of each software item. The testing shall be in accordance with the software item test cases and procedures.

5.9.6 Revision and retesting

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

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) (see H.2.2).

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.

NOTES:

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.

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.

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 the system-wide and system architectural design. The developer shall record software-related information in appropriate software development files (SDFs).

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 item integration test cases and procedures.

5.10.3 Revision and retesting

Based on the results of software/hardware item integration and testing, the developer shall make necessary revisions to the software, participate in necessary retesting, and update the appropriate software development files (SDFs) and other software products as needed.

5.10.4 Analyzing and recording 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 software development files (SDFs).

5.11 System qualification testing

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

NOTES:

1—System qualification testing is performed to demonstrate to the acquirer that system requirements have been met. It covers the system requirements in the System/Subsystem Specifications (SSE) and in associated Interface Requirements Specifications (IRSSs). This testing contrasts with developer-internal system testing, performed as the final stage of software/hardware item integration and testing.

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.

5.11.1 Independence in system qualification testing

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

5.11.2 Testing on the target computer system

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

5.11.3 Preparing for system qualification testing

The developer shall participate in developing and recording the test preparations, test cases, and test procedures to be used for system qualification testing and the traceability between the test cases and the system requirements. For software systems, the results shall include all applicable items in the Software Test Description (STD) (see H.2.1). The developer shall participate in preparing the test data needed to carry out the test cases and in providing the acquirer advance notice of the time and location of system qualification testing.

5.11.4 Dry run of system qualification testing

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

5.11.5 Performing system qualification testing

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

5.11.6 Revision and retesting

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

5.11.7 Analyzing and recording system qualification test results

The developer shall participate in analyzing and recording the results of system qualification testing. For software systems, the result shall include all applicable items in the Software Test Report (STR) (see H.2.2).

5.12 Preparing for software use

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

NOTE—41 software is developed in multiple builds, the developer's planning identifies what software, if any, is to be distributed to users in each build and the extent of distribution (for example, full distribution or distribution to selected evaluators only). Preparing for software use in each build is interpreted to include those activities necessary to carry out the distribution 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 clause of the Software Product Specification (SPS) (see I.2.1).

NOTE—To order only the executable software (delaying delivery of source files and associated maintenance information to a later build), the acquirer can order an SPS, tailoring out all but the executable software clause of the SPS.

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) (see I.2.2).

5.12.3 Preparing user manuals

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

NOTE—Few, if any, systems will need all of the manuals in this clause. 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. Commercial or other manuals that contain the required information can be substituted for the manuals specified in this standard. The manuals in this clause 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) (see J.2.1).

5.12.3.2 Software input/output manuals

The developer shall identify and record information needed by persons who will submit input to, and receive output from, the software, relying on others to operate the software in a computer center or other centralized or networks software installation. The information shall include all applicable items in the Software Input/Output Manual (SIOM) (see J.2.2).

5.12.3.3 Software center operator manuals

The developer shall identify and record information needed by persons who will operate the software in a computer center or other centralized or networked software installation, so that it can be used by others. The information shall include all applicable items in the Software Center Operator Manual (SCOM) (see J.2.3).

5.12.3.4 Computer operation manuals

The developer shall identify and record information needed to operate the computers on which the software will run. The information shall include all applicable items in the Computer Operation Manual (COM) (see J.2.4).

5.12.4 Installation at user sites

The developer shall:

- a) Install and check out the executable software at the user sites specified in the contract.
- b) Provide training to users as specified in the contract.
- c) Provide other assistance to user sites as specified in the contract.

5.13 Preparing for software transition

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

NOTE—If software is developed in multiple builds, the developer's planning identifies what software, if any, is to be transitional 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.

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 clause of the Software Product Specification (SPS) (see I.2.1).

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 clause of the Software Product Specification (SPS) (see I.2.1).

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) (see I.2.2).

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 “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 measured computer hardware resource utilization and the software item requirements concerning them. The result shall include

all applicable items in the qualification, software maintenance, and traceability clauses of the Software Product Specification (SPS) (see I.2.1).

NOTE—In hardware development, the final product is an approved design from which hardware items can be manufactured. In software development, by contrast, the final product is the software, not its design, and “manufacturing” consists of electronically duplicating the software, not recreating it from the design. The “as built” design is included in the Software Product Specification not as the product but as information that may help the maintenance organization understand the software in order to modify, enhance, and otherwise maintain it.

5.13.5 Updating the system design description

The developer shall participate in updating the system design description to match the “as built” system. The result shall include all applicable items in the System/Subsystem Design Description (SSDD) (see G.2.1).

5.13.6 Updating the software requirements

The developer shall update the software and interface requirements for each software item to match the approved “as built” software. The result shall include all applicable items in the Software Requirements Specification (SRS) (see F.2.4) and in the Interface Requirements Specification (IRS) (see F.2.3).

5.13.7 Updating the system requirements

The developer shall participate in updating the system requirements and associated interface requirements to match the approved “as built” system. The result shall include all applicable items in the System/Subsystem Specification (SSS) (see F.2.2) and in the Interface Requirements Specification (IRS) (see F.2.3).

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 clause. 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. Commercial or other manuals that contain the required information can be substituted for the manuals specified in this standard. The manuals in this clause supplement the System/Subsystem Design Description (SSDD) and the Software Product Specifications (SPS), which serve as the primary sources of information for software maintenance. The user manuals cited in 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) (see I.2.3).

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) (see I.2.4).

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

NOTE—A number of requirements in this standard refer to “project-level or higher configuration control.” If “project-level” is not a level of control selected for the project, the Software Development Plan states how these requirements map to the selected levels.

5.14.3 Configuration status accounting

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

5.14.4 Configuration audits

The developer shall perform periodic audits of all entities that have been placed under project-level or higher configuration control. These audits shall ensure that each entity incorporates all approved changes, and no other changes, scheduled for inclusion at the time of the audit.

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. Annex K provides guidance on the structure and format of deliverable software products. The developer shall maintain master copies of delivered software products for the duration specified in the contract.

5.15 Software product evaluation

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

NOTE—If a system or 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.

5.15.1 In-process and final software product evaluations

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

5.15.2 Software product evaluation records

The developer shall prepare and maintain records of each software product evaluation. These records shall be maintained for the duration specified in the contract. Problems in software products under project-level or higher configuration control shall be handled as described in 5.1 7 (Corrective action).

5.15.3 Independence in software product evaluation

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

5.16 Software quality assurance

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

NOTE—If a system or 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 5.1.1).

5.16.1 Software quality assurance evaluations

The developer shall conduct on-going evaluations of software development activities and the resulting software products to:

- a) Assure that each activity required by the contract or described in the Software Development Plan is being performed in accordance with the contract and with the Software Development Plan.
- b) Assure that each software product required by this standard or by other contract provisions exists and has undergone software product evaluations, testing, and corrective action as required by this standard and by other contract provisions.

5.16.2 Software quality assurance records

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

5.16.3 Independence in software quality assurance

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

5.17 Corrective action

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

5.17.1 Problem/change reports

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

5.17.2 Corrective action system

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

- a) Input 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 specified in the contract.
- c) Each problem shall be classified by category and priority, using the categories and priorities in annex m or acquirer approved alternatives.
- d) Analysis shall be performed to detect trends in the problems reported.
- e) Corrective actions shall be evaluated to determine whether problems have been resolved, adverse trends have been reversed, and changes have been correctly implemented without introducing additional problems.

5.18 Joint technical and management reviews

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

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

5.18.2 Joint management reviews

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

- 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 in the Software Development Plan; and implement the strategies in accordance with the plan.

5.20 Software management indicators

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

5.21 Administrative security and privacy protection

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

5.22 Managing subcontractors

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 Interfacing with software IV&V agents

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

5.24 Coordinating with associate developers

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

5.25 Project process improvement

The developer shall periodically assess the processes used on the project to determine their suitability and effectiveness. Based on these assessments, the developer shall identify any necessary and beneficial improvements to the project-specific processes, shall identify these improvements to the acquirer in the form of proposed updates to the Software Development Plan and, if approved, shall implement the improvements on the project.

6. Notes

(The contents of this clause are for information only.)

6.1 Cross reference of standard subclauses to annex subclauses

Given below is a cross reference from the subclauses of the standard to subclauses of the annexes that specify the software products cited in each subclause of the standard.

Standard	Annex	Title of Software Product
5.1.1 , 5.16.2, 5.17.1, 5.17.2, 5.19, 5.20, 5.25	E.2.1	Software Development Plan (SDP)
5.1.2 , 5.1.3	E.2.2	Software Test Plan (STP)
5.1.4	E.2.3	Software Installation Plan (SIP)
5.1.5	E.2.4	Software Transition Plan (STrP)
5.3.2	F.2.1	Operational Concept Description (OCD)
5.3.3 , 5.13.7	F.2.2	System/Subsystem Specification (SSS)
5.3.3 , 5.5, 5.13.6, 5.13.7	F.2.3	Interface Requirements Specification (IRS)
5.4.1 , 5.4.2 , 5.13.5	G.2.1	System/Subsystem Design Description (SSDD)
5.4.1 , 5.4.2 , 5.6.1 , 5.6.2 , 5.6.3	G.2.2	Interface Design Description (IDD)
5.4.1 , 5.4.2 , 5.6.1 , 5.6.2 , 5.6.3	G.2.3	Database Design Description (DBDD)
5.5, 5.13.6	F.2.4	Software Requirements Specification (SRS)
5.6.1 , 5.6.2 , 5.6.3	G.2.4	Software Design Description (SDD)
5.9.3 , 5.11.3	H.2.1	Software Test Description (STD)
5.9.7 , 5.11.7	H.2.2	Software Test Report (STR)
5.12.1, 5.13.1, 5.13.2, 5.13.4	I.2.1	Software Product Specification (SPS)
5.12.2, 5.13.3	I.2.2	Software Version Description (SVD)
5.12.3.1	J.2.1	Software User Manual (SUM)
5.12.3.2	J.2.2	Software Input/Output Manual (SIOM)
5.12.3.3	J.2.3	Software Center Operator Manual (SCOM)
5.12.3.4	J.2.4	Computer Operation Manual (COM)
5.13.8.1	I.2.3	Computer Programming Manual (CPM)
5.13.8.2	I.2.4	Firmware Support Manual (FSM)

6.2 Delivery of tool contents

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

Annex A

(normative)

Tailoring this standard for a project

A.1 Scope

This annex contains requirements for tailoring this standard for a project. This annex is normative (that is, binding), and is not subject to tailoring by either the acquirer or developer.

A.2 Tailoring process

The tailoring process consists of the following activities:

- a) Identifying the project environment
- b) Soliciting input
- c) Selecting activities and products
- d) Specifying tailoring decisions

A.3 Identifying the project environment

Characteristics of the project environment that influence tailoring decisions shall be identified. Examples of characteristics include: objectives of this phase of the development (such as exploring concepts, demonstrating capabilities, manufacturing); system and software requirements; organizational policies, procedures, and strategies; size, criticality, and type of system; existing software products; and number of personnel and organizations involved.

A.4 Soliciting input

Input from the organizations affected by tailoring decisions shall be solicited. User, maintenance, contracts, and potential (or selected) development organizations should be involved in tailoring.

A.5 Selecting processes, activities, and tasks

The processes, activities and products applicable to the project shall be identified. The following guidance should be applied when selecting applicable activities and products for the project.

- a) Evaluate the requirements in J-STD-016-1995 to determine whether each requirement is applicable based on the project environment characteristics. Annexes B and C provide guidance on tailoring this standard.
- b) Decide which requirements in this standard are applicable and cost-effective over the life of the project.
- c) Modify, if necessary, requirements in this standard to reflect unique or special needs of the project.

- d) Decide which additional requirements not found in this standard, if any, should be specified.
- e) Evaluate comments on tailoring decisions solicited from affected organizations.
- f) Determine applicable requirements.
- g) Record tailoring decisions and their rationale.

A.6 Specifying tailoring decisions

Tailoring decisions for the project shall be specified in the contract.

Annex B

(informative)

Guidance on development strategies, tailoring, and build planning

B.1 Scope

This annex describes three candidate development strategies and shows how this standard can be applied under each of these strategies and on a project involving reengineering. It is informative only, provided to aid in using this standard.

B.2 Candidate development strategies

There are many different development strategies that can be applied to software development. Three of these strategies are summarized below and in figure 2.

- a) Once-through. The “once-through” strategy consists of performing the development process a single time. Simplistically: determine user needs, define requirements, design the system, implement the system, test, fix, and deliver.
- b) Incremental. The “incremental” strategy determines user needs and defines the system requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities, the next build adds more capabilities, and so on, until the system is complete.
- c) Evolutionary. The “evolutionary” strategy also develops a system in builds, but differs from the incremental strategy in acknowledging that the user need is not fully understood and all requirements cannot be defined up front. In this strategy, user needs and system requirements are partially defined up front, then are refined in each succeeding build.

Development strategy	Define All Requirements First?	Multiple Development Cycles?	Distribute Interim Software?
Once-Through	Yes	No	No
Incremental (Preplanned Product Improvement)	Yes	Yes	Maybe
Evolutionary	No	Yes	Yes

Figure 2 — Key features of three development strategies

B.3 Selecting an appropriate development strategy

The development strategy is selected by the acquirer, but may be proposed by prospective or selected developers. Figure 3 illustrates a risk analysis approach for selecting an appropriate strategy. The approach consists of listing risk items (negatives) and opportunity items (positives) for each strategy; assigning each item a risk or opportunity level of High, Medium, or Low; and making a decision on which strategy to use based on a trade-off among the risks and opportunities. The fill-ins shown are sample considerations only. An actual analysis may use others. The “DECISION” entry on the bottom line shows which strategy was selected.

B.4 Relationship of this standard to development strategies

The development strategy usually applies to the overall system. The software within the system may be acquired under the same strategy or under a different one, such as requiring that all software be finalized in the first build of the system. Figures 4, 5, and 6 show how this standard might be applied under each of the development strategies identified in B.2. Figure 7 shows how this standard might be applied on a reengineering project. All four figures are, by necessity, simplified. For example, they show the standard’s activities in sequence when they might actually be ongoing, overlapping, or iterative, and they show each software product as a single entity, without depicting early drafts or updates.

B.5 Planning software builds and tailoring this standard

Planning the software builds on a project for each build may be accomplished in several ways. The acquirer might, for example, select an overall development strategy, leaving it to the developer to lay out the software builds. Alternatively, the acquirer might lay out the software builds as part of the contract. The approach selected will be project-dependent. The subclauses below provide guidelines for planning the builds and tailoring the standard without attempting to identify who performs each of the activities.

B.5.1 Identifying builds and their objectives

The first step in software build planning is to lay out a series of one or more builds and to identify the objectives of each build. The top part of figure 8 illustrates such planning. In the example, the System/Subsystem Specification (SSS) already exists and fulfillment of its requirements is divided into four builds, two of which will be prototypes distributed to a selected set of users, and two of which will be distributed to all users. A further objective of Build 4 is transitioning the software to the designated maintenance organization. An actual project would expand on these objectives.

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

Figure 3 — Sample risk analysis for determining the appropriate development strategy

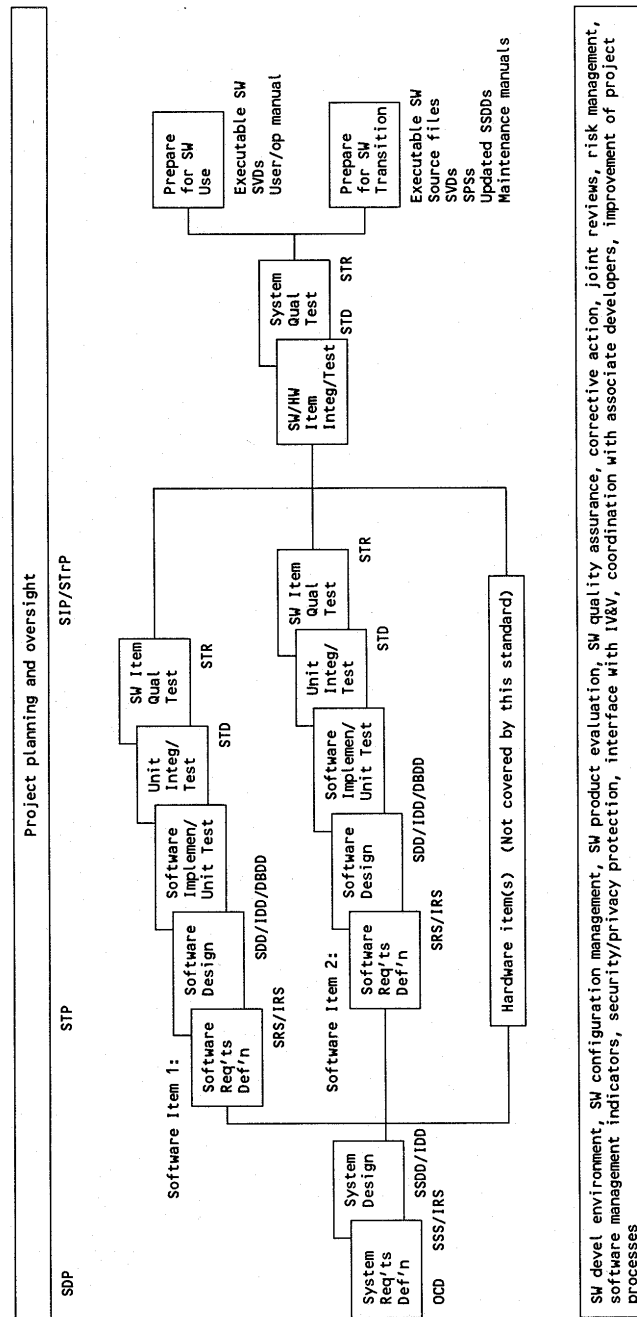


Figure 4 — One possible way of applying this standard to the Once-Through development strategy

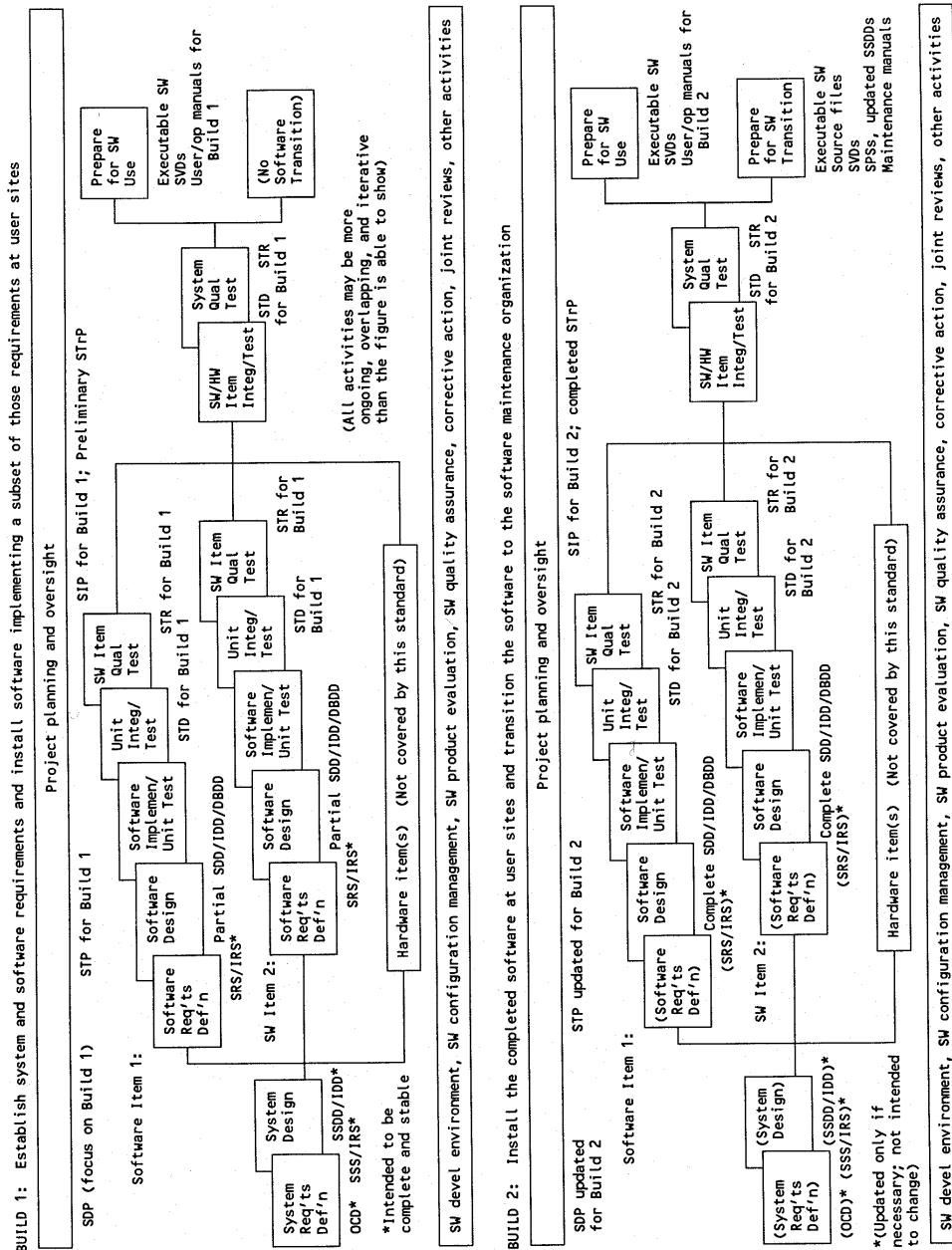


Figure 5 — One possible way of applying this standard to the Incremental development strategy

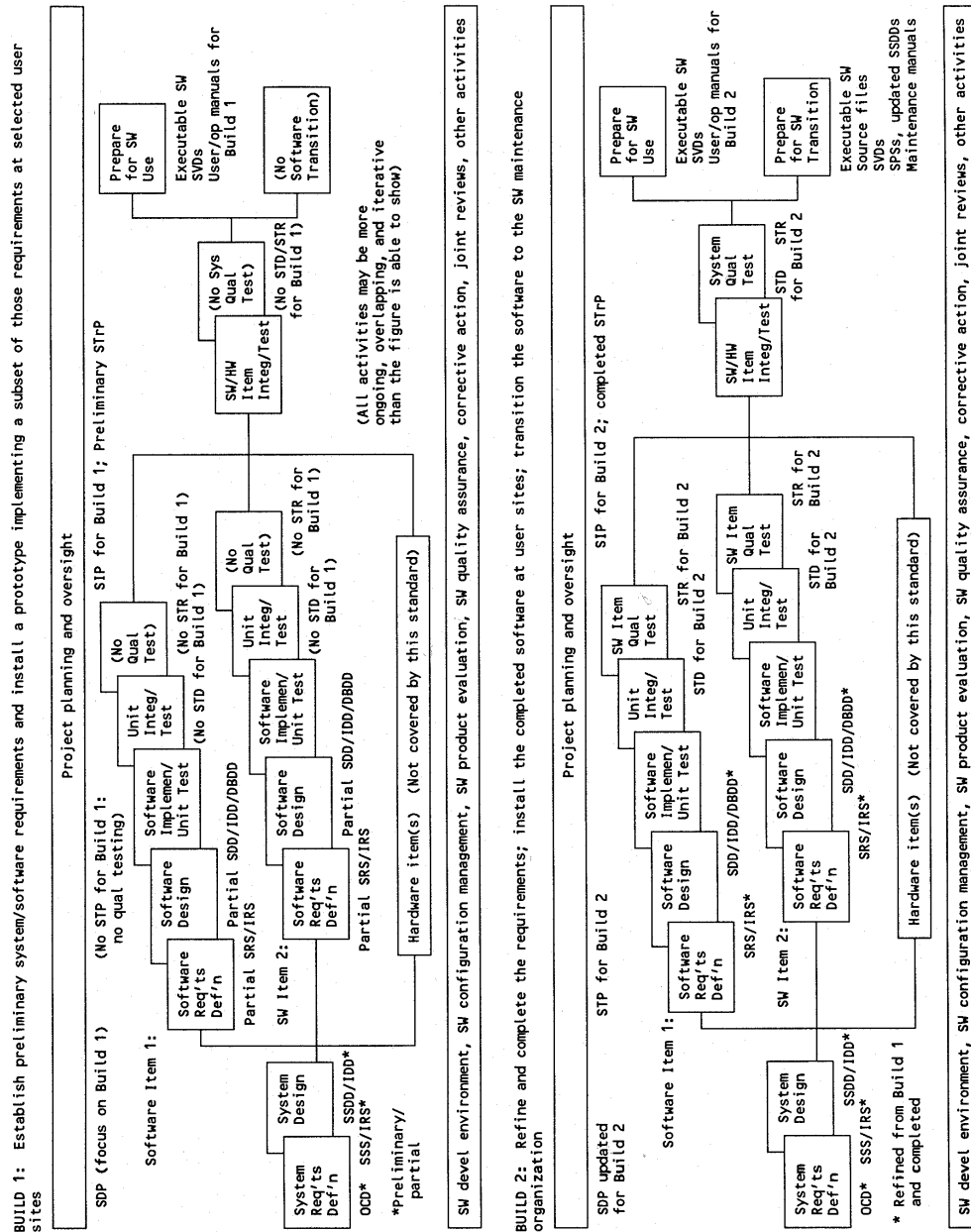


Figure6 — One possible way of applying this standard to the Evolutionary development strategy

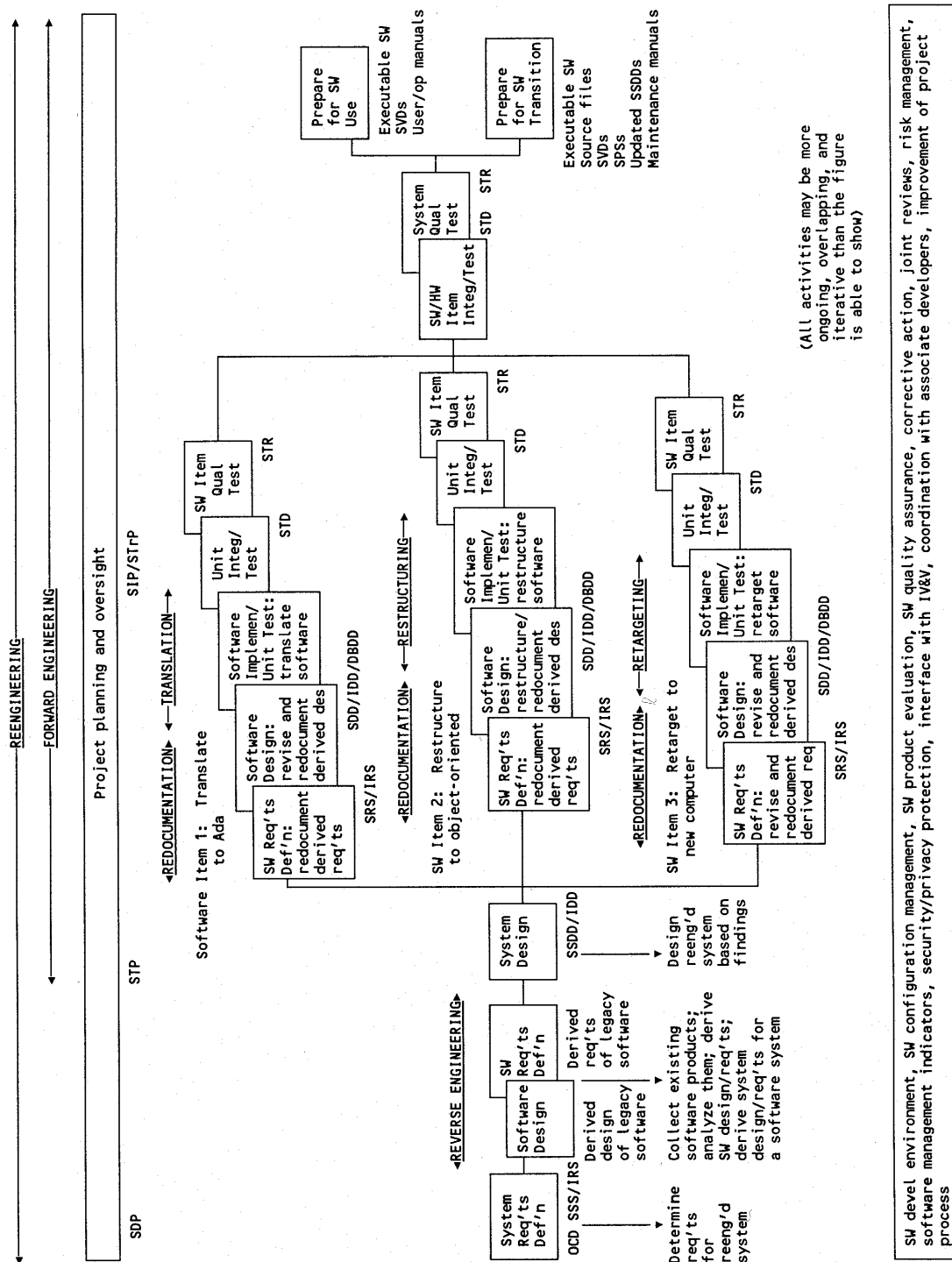


Figure7 — One possible way of applying this standard to a reengineering project