J-STD-016-1995
30 September 1995

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

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

PDF: ISBN 0-7381-0427-2, SS94377
Print: SH94377

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

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)

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) **ls 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 GrayLewis 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

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 confidenti-

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 hasbeen 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 (IRSs). 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 5.17 (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		
- System too large to do all at once	M	- User prefers all capabilities at first delivery	M	- User prefers all capabilities at first delivery	M
- Rapid changes in 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

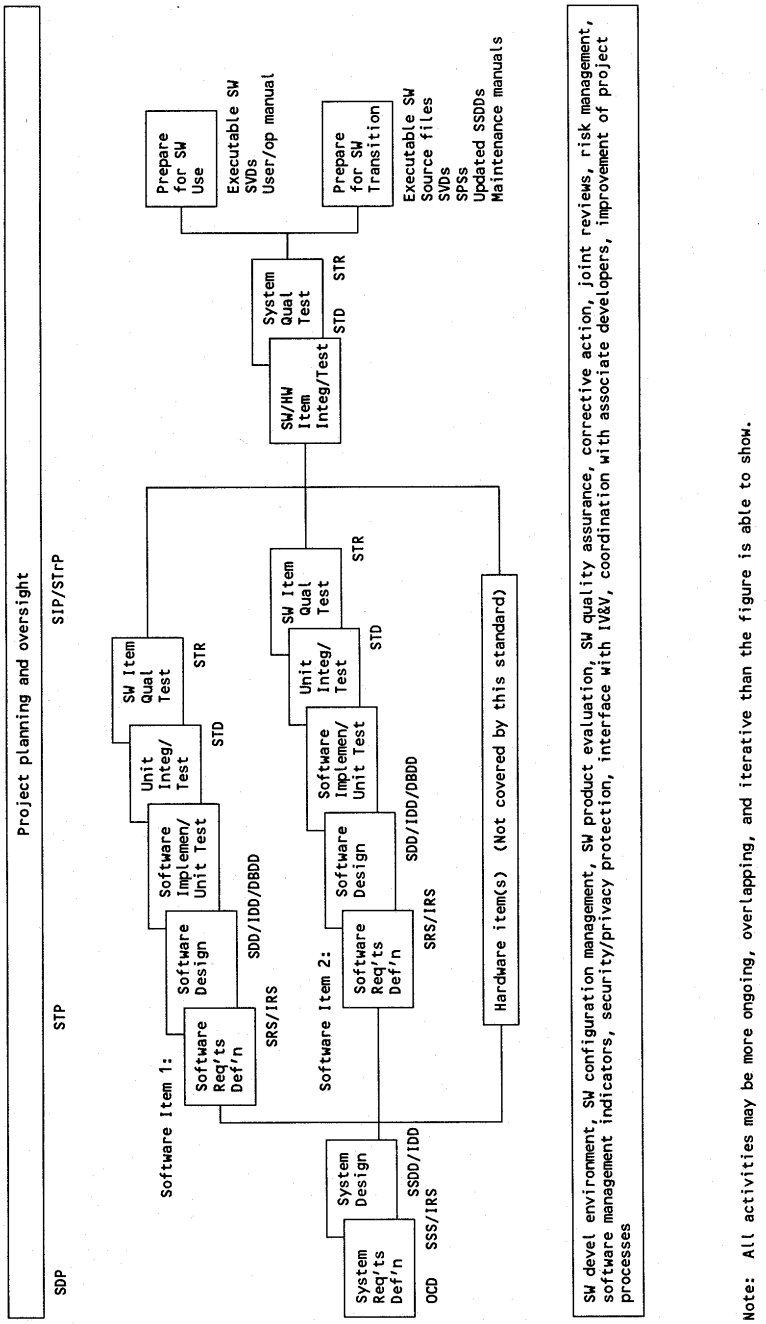

Figure 4 — One possible way of applying this standard to the Once-Through development strategy

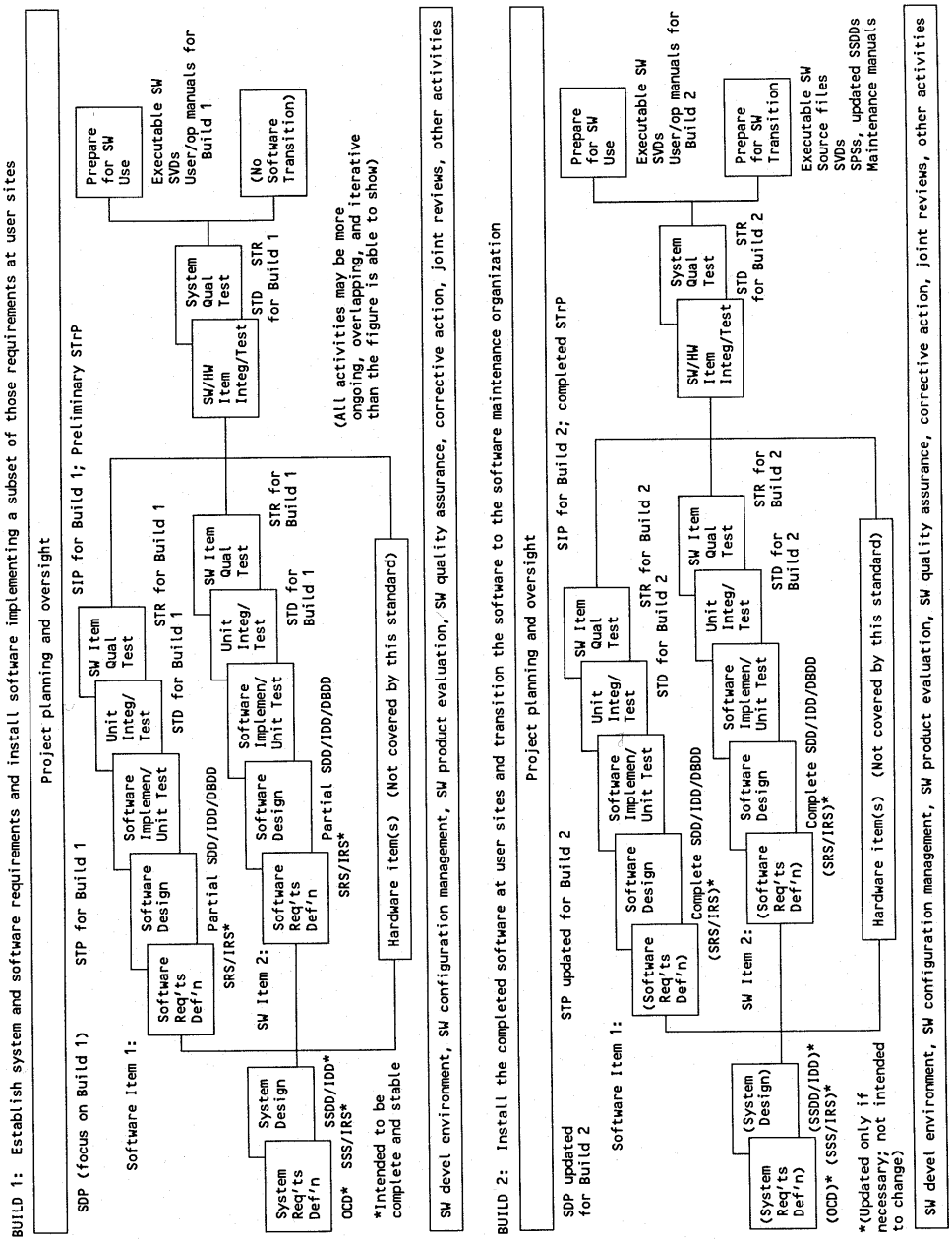

Figure 5 — One possible way of applying this standard to the Incremental development strategy

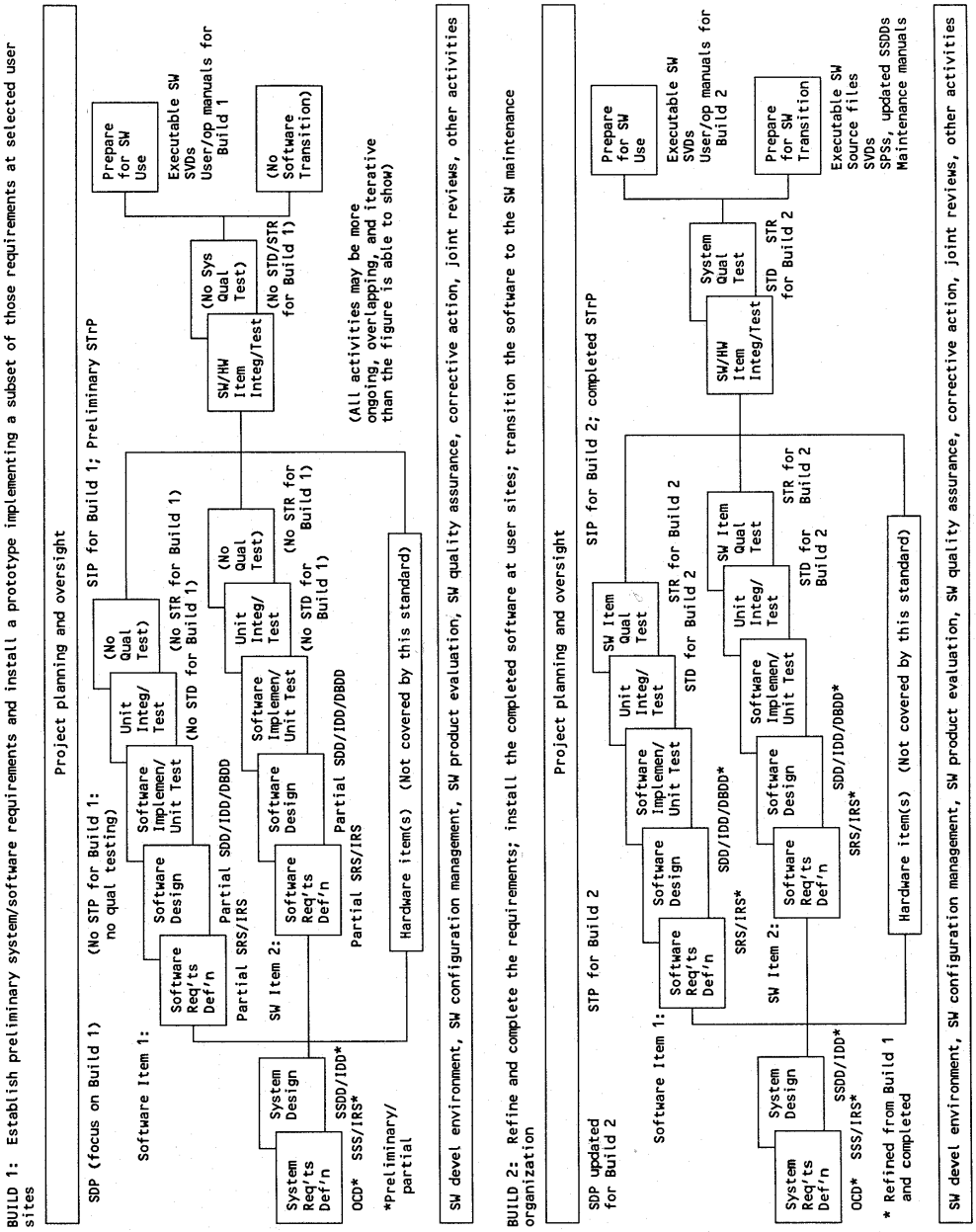

Figure6 — One possible way of applying this standard to the Evolutionary development strategy

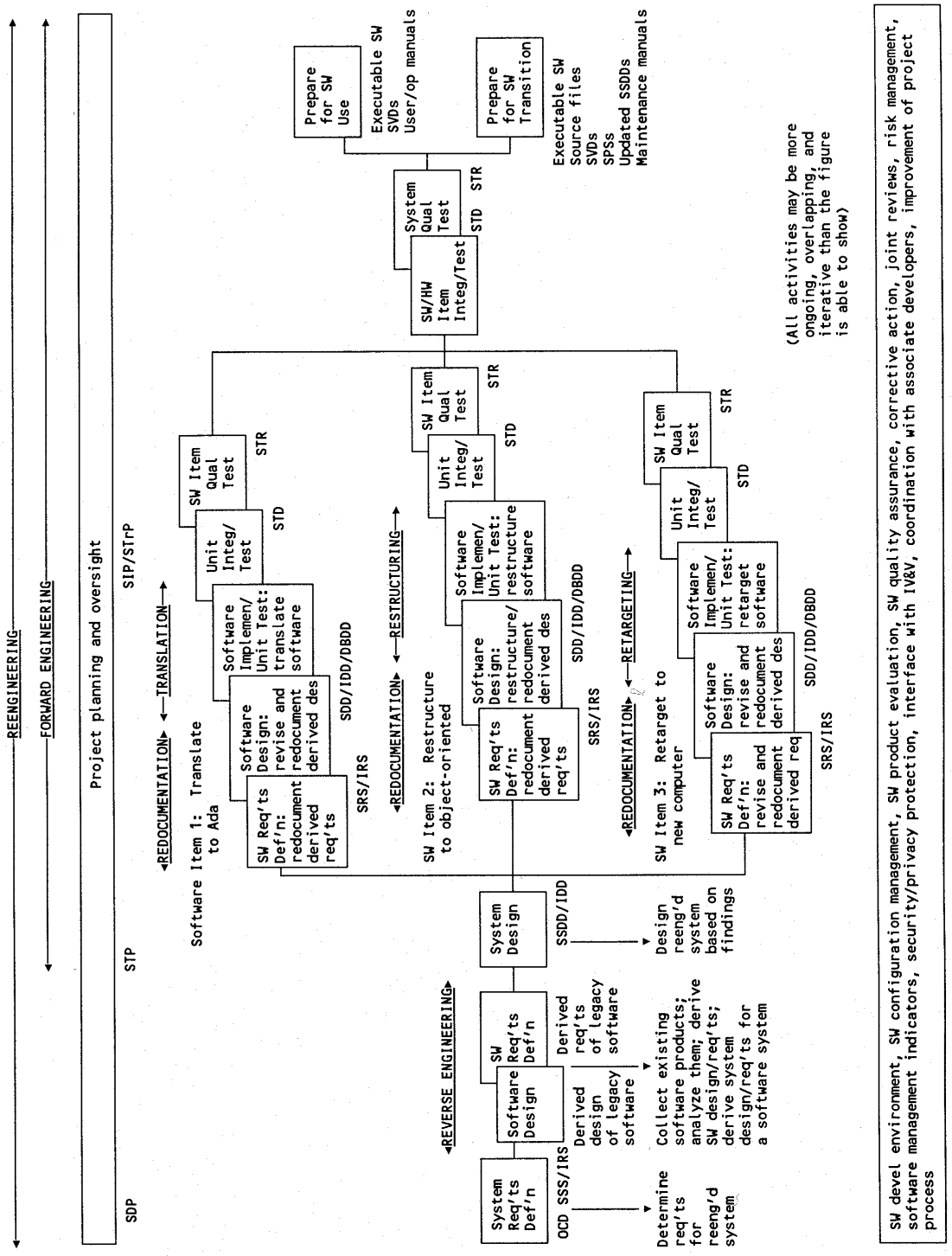

Figure7 — One possible way of applying this standard to a reengineering project

B.5.2 Identifying the activities to be performed in each build

The next step in build planning is identifying which activities apply in each build and determining the extent to which they apply. The activities of figure 8 show the start of this tailoring. Listed on the left are the subclauses of this standard. The worksheet entries indicate in which builds each activity is to be performed and include any notes regarding the nature of each activity in each build. For example, the figure shows that each build will include software development planning (5.1.1), but that the nature of that planning changes in each build. Some activities will not apply at all in a given build, some will apply identically in all builds, and some will apply differently in different builds. Since some aspects of the project, such as number and type of software items, may not have been identified at the time the worksheet is being filled out, completion of the worksheet may itself be incremental. The following guidelines apply:

a) Different decisions will apply to different types of software on a project. These differences can be shown within the entries of one worksheet or by using different worksheets for different types of software on a project.
b) If early builds are devoted to experimentation, developing "throw-away" software to arrive at a system concept or system requirements, it may be appropriate to forgo certain formalities, such as coding standards, that will be imposed later on the "real" software. If the early software will be used later, such formalities may be appropriate from the start. These decisions are project-dependent.

B.5.3 Recording tailoring decisions

Tailoring decisions made by the acquirer before the project begins are specified in the contract. Tailoring proposed by the developer may be communicated via feedback on draft requests for proposal, proposals, the Software Development Plan, joint reviews during the project, or by other means of communication. Refinements to the tailoring decisions may be ongoing as the project proceeds. Those involving contractual changes are handled accordingly.

B.5.4 Scheduling the selected activities in each build

Another important step in build planning is scheduling the activities in each build. As with tailoring, the acquirer may set forth general milestones and have the developer provide specifics or may provide specific schedules. The following guidelines apply:

a) A common mistake is to treat all software items as though they are required to be developed in "lock-step," reaching key milestones at the same time. Allowing software items to be on different schedules can result in more optimum development.
b) A similar mistake is to treat software units as though they are required to be developed in "lock-step," all designed by a certain date, implemented by a certain date, etc. Flexibility in the scheduling of software units can also be effective.
c) The activities in this standard need not be performed sequentially. Several may be taking place at one time, and an activity may be performed continually or intermittently throughout a build or over multiple builds. The activities in each build are to be laid out in the manner that best suits the work to be done.

BUILD PLANNING WORKSHEET		Build			
		1	2	3	4
1. Identify at right the objectives of each build 2. Indicate below which activities are to be accomplished during the development of each build. Add clarifying notes as needed.		Deliver to selected users an operational prototype that meets the following system-level requirements: SSS-1, SSS-5, ..., SSS-1250	Deliver to selected users an operational prototype that meets the requirements of Build 1 plus: SSS-2, SSS-3, SSS-15, ..., SSS-1249	Deliver to all users a tested system that meets the requirements of Builds 1 and 2 plus: SSS-4, SSS-7, SSS-10, ..., SSS-1248	Deliver to all users a tested system that meets all system-level requirements; transition to designated maintenance organization
Para	Activity				
5.1	PROJECT PLANNING AND OVERSIGHT				
5.1.1	Plan the software development effort	Yes: Plan Build 1 in detail; Builds 2-4 in general	Yes: Plan Build 2 in detail; Builds 3-4 in general	Yes: Plan Build 3 in detail; Build 4 in general	Yes: Plan Build 4 in detail
5.1.2	Plan for software item qualification testing	No: No software item qual testing in this build	No: No software item qual testing in this build	Yes: Plan for software item qual testing in this build	Yes: Update for software item qual testing in this build
5.1.3	Plan for system qualification testing	No: No system qual testing in this build	No: No system qual testing in this build	Yes: Plan for system qual testing in this build	Yes: Update for system qual testing in this build
5.1.4	Plan for installing software at user sites	No: Let users install on their own	No: Let users install on their own	Yes: Plan to install at user sites	Yes: Update as needed for installation of Bld 4
5.1.5	Plan for transitioning software to the maintenance organization	Yes: Very preliminary planning only	Yes: Update preliminary plans	Yes: Update preliminary plans	Yes: Finalize transition planning
5.1.6	Follow plans; perform management review	Yes: For those plans that are in effect	Yes: For those plans that are in effect	Yes: For those plans that are in effect	Yes: For those plans that are in effect
5.2	ESTABLISHING A SOFTWARE DEVEL ENVIRONMENT				
5.2.1	Establish a software engineering environment	Yes: As needed for Build 1	Yes: Update as needed for Build 2	Yes: Update as needed for Build 3	Yes: Update as needed for Build 4
5.2.2	Establish a software test environment	Yes: As needed for Build 1 testing	Yes: As needed for Build 2 testing	Yes: Set up fully for Build 3 qual testing	Yes: Update as needed for Build 4 qual testing

Figure 8 — Example of build planning

Annex C

(informative)

Guidance on ordering deliverables

C.1 Scope

This annex provides guidance to the acquirer on the deliverables to be required on a software development project. It is informative only, provided to aid in using this standard.

C.2 Ordering deliverables

This standard has been worded to differentiate between the planning/engineering activities that make up a software development project and the generation of deliverables. A key objective of this wording is to eliminate the notion that the acquirer is required to order a given deliverable in order to have planning or engineering work take place. Under this standard, the planning and engineering work takes place regardless of which deliverables are ordered, unless a given activity is tailored out of the standard. In addition, joint technical reviews have been included to review the results of that work in its natural form, without the generation of deliverables. Deliverables are to be ordered only when there is a genuine need to have planning or engineering information transformed into a deliverable (such as deliverables needed for use or maintenance of the software), recognizing that this transformation requires time and effort that would otherwise be spent on the engineering effort. Figure 10 in annex K provides information helpful in deciding whether the corresponding deliverable is to be ordered.

C.3 Scheduling deliverables

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

C.4 Format of deliverables

Traditional deliverables take the form of paper documents exactly following required formats and structures. While this form works well for some deliverables, it is not the only form, and alternatives are to be considered. One variation from paper documents is word processing files containing those documents. This saves paper, is less expensive to duplicate, transport, and change, but still requires the developer to format and structure the information in a particular way. Another variation is specifying that a paper or word processor

document is to include required contents but may be in the developer's format and structure. Yet another variation is allowing deliverables to take forms that are not traditional documents at all, such as data in computer-aided software engineering (CASE) tools. These variations can be specified in the contract.

C.5 Tailoring the content requirements for deliverables

Tailoring the content requirements for deliverables consists of deleting from the software product contents clauses of annexes E through J any requirements for unneeded information, altering requirements to more explicitly reflect the application to a particular effort, or adding requirements as needed. It can also consist of changes such as combining two or more documents under one cover.

Annex D

(informative)

Interpreting this standard for incorporation of reusable software products

D.1 Scope

This annex interprets this standard when applied to the incorporation of reusable software products. It is informative only, provided to aid in using this standard.

D.2 Evaluating reusable software products

The criteria for use in evaluating reusable software products will be described in the developer's SDP. Examples of candidate criteria that may be used include, but are not limited to:

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

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

The following rules can be applied in interpreting this standard:

a) Any requirement that calls for development of a software product may be met by a reusable software product that fulfills the requirement and meets the criteria established in the Software Development Plan. The reusable software product may be used as-is or modified and may be used to satisfy part or all of the requirement. For example, a requirement may be met by using an existing plan, specification, or design.
b) When the reusable software product to be incorporated is the software itself, some of the requirements in this standard require special interpretation. Figure 9 provides this interpretation. Key issues

are whether the software will be modified, whether unmodified software constitutes an entire software item or only one or more software units, and whether unmodified software has a positive performance record (no firm criteria exist for making this determination). The figure is presented in a conditional manner: if an activity in the left column is required for a given type of software, the figure tells how to interpret the activity for reusable software of that type.

If this activity is required:	Interpret the activity as follows for each type of existing, reusable software:				
	For software items to be used unmodified		For software units to be used unmodified		For software units being modified for/ during project
	Positive performance record	No or poor performance record	Positive performance record	No or poor performance record	
5.1 Project planning and oversight	Include the activities in this figure in project plans				
5.2 Establishing software devel environment	Establish and apply a software test environment, software development library, and software development files as appropriate to perform the activities in this figure				Apply full requirements
5.3 System requirements definition	Consider software's capabilities in defining the operational concept & system requirements				
	Use test/ performance records to confirm ability to meet needs	Test to confirm ability to meet needs	Use test/ performance records to confirm ability to meet needs	Test to confirm ability to meet needs	Use tests or records to determine potential to meet needs
5.4.1 System-wide design	Consider the software's capabilities and characteristics in designing system behavior and in making other system-wide design decisions				
5.4.2 System architectural design	Include the software item in the system architecture; allocate system requirements to it		Consider the unit's capabilities and characteristics in designating software items and allocating system requirements to them		
5.5 Software requirements definition	Specify the project-specific requirements the software item is required to meet; verify via records or retest that the software item can meet them		Consider the unit's capabilities and characteristics in specifying the requirements for the software item of which it is a part		
5.6.1 Software item-wide design	No requirement: the software item-wide design decisions have already been made (recording the "as built" design is under 5.13)		Consider the unit's capabilities and characteristics in designing software item behavior and making other software item-wide design decisions		
5.6.2 Software item architectural design	No requirement: the architectural design is already defined (recording the "as built" design is under 5.13)		Include the unit in the software item architecture and allocate software item requirements to it		
5.6.3 Software item detailed design	No requirement: the detailed design is already defined (recording the "as built" design is under 5.13)		No requirement: the unit is already designed (recording the "as built" design is under 5.13)		Modify the unit's design as needed
5.7.1 Software implementation	No requirement: the software for the software item's units is already implemented		No requirement: the software for the unit is already implemented		Modify the software for the unit
5.7.2-5.7.5 Unit testing	No requirement: the software items's units are already tested	Perform selectively if in question and units are accessible	No requirement: the unit is already tested	Perform this testing	

Figure 9 — Interpreting this standard for incorporation of reusable software

If this activity is required:	Interpret the activity as follows for each type of existing, reusable software:				
	For software items to be used unmodified		For software units to be used unmodified		For software units being modified for/ during project
	Positive performance record	No or poor performance record	Positive performance record	No or poor performance record	
5.8 Unit integration and testing	No requirement: the SW items units are already integrated	Perform selectively if in question and units are accessible	Perform except where integration is already tested/proven	Perform this testing	
5.9 Software item qualification testing	No requirement: software item is already tested & proven	Perform this testing	Include the unit in software item qualification testing		
5.10 Software/ hardware item integration and testing	Perform, except where integration is already tested/proven	Include the software item in SW/HW item integration and testing	Include the unit in software/hardware item integration and testing		
5.11 System qualification testing	Include the software item in system qualification testing		Include the unit in system qualification testing		
5.12 Preparing for software use	Include the software for the software item or unit in the executable software; include in version descriptions; handle any license issues; cover use of the software item or unit, as appropriate, via existing, new, or revised user/operator manuals; install the software item or unit as part of the overall system; include its use, as appropriate, in the training offered				
5.13 Preparing for software transition	Include the software for the software item or unit in the executable software; prepare source files for the software item or unit, if available; include in version descriptions; handle any license issues; prepare or provide "as built" design descriptions for software whose design is known; install the software item or unit at the maintenance site; demonstrate regenerability if source is available; include in the training offered				
5.14 Software configuration management	Apply to all software products prepared, modified, or used in incorporating this software				
5.15 Software product evaluation	Apply to all software products prepared or modified in incorporating this software; for software products used unchanged, apply unless a positive performance record or evidence of past evaluations indicates that such an evaluation would be duplicative				
5.16 Software quality assurance	Apply to all activities performed and all software products prepared, modified, or used in incorporating this software				
5.17 Corrective action	Apply to all activities performed and all software products prepared or modified in incorporating this software				
5.18 Joint reviews	Cover the software products prepared or modified in incorporating this software				
5.19 - 5.25 Other activities	Apply the full requirements of each of these 6 clauses, if applicable				

Figure 9 — Interpreting this standard for incorporation of reusable software -- (continued)

Annex E

(normative)

Planning software product descriptions

E.1 Scope

This annex specifies the contents of software products associated with planning in this standard. It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

E.2 Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in the order they are referenced in this standard and include the following:

a) Software Development Plan (SDP)
b) Software Test Plan (STP)
c) Software Installation Plan (SIP)
d) Software Transition Plan (STrP)

E.2.1 Contents of the Software Development Plan (SDP)

SOFTWARE DEVELOPMENT PLAN (SDP)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
   1.4) Relationship to other plans
2) Referenced documents
3) Overview of required work
4) Plans for performing general software development activities
   4.1) Software development process
   4.2) General plans for software development
      4.2.1)Software development methods
      4.2.2)Standards and practices for software products
      4.2.3)Traceability
      4.2.4)Reusable software products
      4.2.5)Handling of critical requirements
      4.2.6)Computer hardware resource utilization
      4.2.7)Recording rationale
      4.2.8)Access for acquirer review
5) Plans for performing detailed software development activities

5.1) Project planning and oversight
5.1.1)Software development planning (covering updates to this plan)
5.1.2)Software item test planning
5.1.3)System test planning
5.1.4)Software installation planning
5.1.5)Software transition planning
5.1.6)Following and updating plans, including the intervals for management review
5.2) Establishing a software development environment
5.2.1)Software engineering environment
5.2.2)Software test environment
5.2.3)Software development library
5.2.4)Software development files
5.2.5)Non-deliverable software
5.3) System requirements definition
5.3.1)Analysis of user input
5.3.2)Operational concept
5.3.3)System requirements
5.4) System design
5.4.1)System-wide design decisions
5.4.2)System architectural design
5.5) Software requirements definition
5.6) Software design
5.6.1)Software item-wide design decisions
5.6.2)Software item architectural design
5.6.3)Software item detailed design
5.7) Software implementation and unit testing
5.7.1)Software implementation
5.7.2)Preparing for unit testing
5.7.3)Performing unit testing
5.7.4)Revision and retesting
5.7.5)Analyzing and recording unit test results
5.8) Unit integration and testing
5.8.1)Preparing for unit integration and testing
5.8.2)Performing unit integration and testing
5.8.3)Revision and retesting
5.8.4)Analyzing and recording unit integration and test results
5.9) Software item qualification testing
5.9.1)Independence in software item qualification testing
5.9.2)Testing on the target computer system
5.9.3)Preparing for software item qualification testing
5.9.4)Dry run of software item qualification testing
5.9.5)Performing software item qualification testing
5.9.6)Revision and retesting
5.9.7)Analyzing and recording software item qualification test results
5.10) Software/hardware item integration and testing
5.10.1)Preparing for software/hardware item integration and testing
5.10.2)Performing software/hardware item integration and testing
5.10.3)Revision and retesting
5.10.4)Analyzing and recording software/hardware item integration and test results
5.11) System qualification testing
5.11.1)Independence in system qualification testing
5.11.2)Testing on the target computer system
5.11.3)Preparing for system qualification testing
5.11.4)Dry run of system qualification testing

5.11.5)Performing system qualification testing
5.11.6)Revision and retesting
5.11.7)Analyzing and recording system qualification test results
5.12) Preparing for software use
5.12.1)Preparing the executable software
5.12.2)Preparing version descriptions for user sites
5.12.3)Preparing user manuals
5.12.4)Installation at user sites
5.13) Preparing for software transition
5.13.1)Preparing the executable software
5.13.2)Preparing source files
5.13.3)Preparing version descriptions for the maintenance site
5.13.4)Preparing the “as built” software item design and other software maintenance information
5.13.5)Updating the system design description
5.13.6)Updating the software requirements specification
5.13.7)Updating the system/subsystem specification
5.13.8)Preparing maintenance manuals
5.13.9)Transition to the designated maintenance site
5.14) Software configuration management
5.14.1)Configuration identification
5.14.2)Configuration control
5.14.3)Configuration status accounting
5.14.4)Configuration audits
5.14.5)Packaging, storage, handling, and delivery
5.15) Software product evaluation
5.15.1)In-process and final software product evaluations
5.15.2)Software product evaluation records, including items to be recorded
5.15.3)Independence in software product evaluation
5.16) Software quality assurance
5.16.1)Software quality assurance evaluations
5.16.2)Software quality assurance records, including items to be recorded
5.16.3)Independence in software quality assurance
5.17) Corrective action
5.17.1)Problem/change reports, including items to be recorded
5.17.2)Corrective action system
5.18) Joint technical and management reviews
5.18.1)Joint technical reviews, including a proposed set of reviews
5.18.2)Joint management reviews, including a proposed set of reviews
5.19) Risk management
5.20) Software management indicators
5.21) Administrative security and privacy protection
5.22) Managing subcontractors
5.23) Interfacing with software IV&V agents
5.24) Coordinating with associate developers
5.25) Project process improvement
6) Schedules and activity network
7) Project organization and resources
7.1) Project organization
7.2) Project resources
8) Notes
A) Annexes
1) **Scope** . This clause should be divided into the following subclauses.

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

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

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

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

2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Overview of required work** . This clause should be divided into subclauses as needed to establish the context for the planning described in later clauses. It shall include, as applicable, an overview of:

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

4) **Plans for performing general software development activities** . This clause should be divided into the following subclauses. Provisions corresponding to non-required activities may be satisfied by the words “Not applicable.” If different builds or different software on the project require different planning, these differences shall be noted in the subclauses. In addition to the content specified below, each subclause shall identify applicable risks/uncertainties, plans for dealing with those risks/uncertainties, and shall cover all contractual clauses regarding the identified topic.

4.1) **Software development process** . This subclause shall describe the software development process to be used. The planning shall identify planned builds, if applicable, their objectives, and the software development activities to be performed in each build.

4.2) **General plans for software development** . This subclause should be divided into the following.

4.2.1) **Software development methods** . This subclause shall describe or reference the software development methods to be used. Included shall be descriptions of the manual and automated tools and procedures to be used in support of these methods. Reference may be made to other subclauses in this plan if the methods are better described in context with the activities to which they will be applied.

4.2.2) **Standards and practices for software products** . This subclause shall describe or reference the standards and practices to be followed for representing requirements, design, code, test cases, test procedures, and test results. Reference may be made to other subclauses in this plan if the standards are better described in context with the activities to which they will be applied. Standards for code shall be provided for each programming language to be used. They shall include at a minimum:

a) Standards for format (such as indentation, spacing, capitalization, and order of information)
b) Standards for header comments (requiring, for example, name/identifier of the code; version identification; modification history; purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.)
c) Standards for other comments (such as required number and content expectations)

d) Naming conventions for variables, parameters, packages, procedures, files, etc.
e) Restrictions, if any, on the use of programming language constructs or features
f) Restrictions, if any, on the complexity of code aggregates

4.2.3) **Traceability** . This subclause shall describe the approach to be followed for performing upward and downward traceability.

4.2.4) **Reusable software products** . This subclause shall describe the approach to be followed for identifying, evaluating, and incorporating reusable software products, including the scope of the search for such products and the criteria to be used for their evaluation and incorporation. Candidate or selected reusable software products known at the time this plan is prepared or updated shall be identified and described, together with benefits, drawbacks, and restrictions, as applicable, associated with their use.

4.2.5) **Handling of critical requirements** . This subclause shall describe the approach to be followed for identifying, developing strategies for, and handling safety, security, privacy protection, and other critical requirements.

4.2.6) **Computer hardware resource utilization** . This subclause shall describe the approach to be followed for allocating computer hardware resources and monitoring their utilization.

4.2.7) **Recording rationale** . This subclause shall describe the approach to be followed for recording rationale that will be useful to the maintenance organization for key decisions made on the project. It shall interpret the term "key decisions" for the project and state where the rationale are to be recorded.

4.2.8) **Access for acquirer review** . This subclause shall describe the approach to be followed for providing the acquirer or its authorized representative access to developer and subcontractor facilities for review of software products and activities.

5) **Plans for performing detailed software development activities** . This clause should be divided into the following subclauses. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the subclauses. The discussion of each activity shall include the approach (methods/procedures/tools) to be applied to: 1) the analysis or other technical tasks involved, 2) the recording of results, and 3) the preparation of associated deliverables, if applicable. The discussion shall also identify applicable risks/uncertainties, plans for dealing with those risks/uncertainties, and shall cover all contractual clauses regarding the identified topic. Reference may be made to 4.2.1 if applicable methods are described there.

5.1) **Project planning and oversight** . This subclause should be divided into the following to describe the approach to be followed for project planning and oversight.

5.1.1) Software development planning (covering updates to this plan)
5.1.2) Software item test planning
5.1.3) System test planning
5.1.4) Software installation planning
5.1.5) Software transition planning
5.1.6) Following and updating plans, including the intervals for management review

5.2) **Establishing a software development environment** . This subclause should be divided into the following to describe the approach to be followed for establishing, controlling, and maintaining a software development environment.

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

5.3) **System requirements definition** . This subclause should be divided into the following to describe the approach to be followed for participating in system requirements definition.

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

5.4) **System design** . This subclause should be divided into the following to describe the approach to be followed for participating in system design.
5.4.1)System-wide design decisions
5.4.2)System architectural design

5.5) **Software requirements definition** . This subclause shall describe the approach to be followed for software requirements definition.

5.6) **Software design** . This subclause should be divided into the following to describe the approach to be followed for software design.
5.6.1)Software item-wide design decisions
5.6.2)Software item architectural design
5.6.3)Software item detailed design

5.7) **Software implementation and unit testing** . This subclause should be divided into the following to describe the approach to be followed for software implementation and unit testing.
5.7.1)Software implementation
5.7.2)Preparing for unit testing
5.7.3)Performing unit testing
5.7.4)Revision and retesting
5.7.5)Analyzing and recording unit test results

5.8) **Unit integration and testing** . This subclause should be divided into the following to describe the approach to be followed for unit integration and testing.
5.8.1)Preparing for unit integration and testing
5.8.2)Performing unit integration and testing
5.8.3)Revision and retesting
5.8.4)Analyzing and recording unit integration and test results

5.9) **Software item qualification testing** . This subclause should be divided into the following to describe the approach to be followed for software item qualification testing.
5.9.1)Independence in software item qualification testing
5.9.2)Testing on the target computer system
5.9.3)Preparing for software item qualification testing
5.9.4)Dry run of software item qualification testing
5.9.5)Performing software item qualification testing
5.9.6)Revision and retesting
5.9.7)Analyzing and recording software item qualification test results

5.10)**Software/hardware item integration and testing** . This subclause should be divided into the following to describe the approach to be followed for participating in software/hardware item integration and testing.
5.10.1)Preparing for software/hardware item integration and testing
5.10.2)Performing software/hardware item integration and testing
5.10.3)Revision and retesting
5.10.4)Analyzing and recording software/hardware item integration and test results

5.11)**System qualification testing** . This subclause should be divided into the following to describe the approach to be followed for participating in system qualification testing.
5.11.1)Independence in system qualification testing
5.11.2)Testing on the target computer system
5.11.3)Preparing for system qualification testing
5.11.4)Dry run of system qualification testing
5.11.5)Performing system qualification testing
5.11.6)Revision and retesting
5.11.7)Analyzing and recording system qualification test results

5.12)**Preparing for software use** . This subclause should be divided into the following to describe the approach to be followed for preparing for software use.
5.12.1)Preparing the executable software
5.12.2)Preparing version descriptions for user sites
5.12.3)Preparing user manuals

5.12.4)Installation at user sites

5.13)**Preparing for software transition** . This subclause should be divided into the following to describe the approach to be followed for preparing for software transition.

5.13.1)Preparing the executable software

5.13.2)Preparing source files

5.13.3)Preparing version descriptions for the maintenance site

5.13.4)Preparing the “as built” software item design and other software maintenance information

5.13.5)Updating the system design description

5.13.6)Updating the software requirements specification

5.13.7)Updating the system/subsystem specification

5.13.8)Preparing maintenance manuals

5.13.9)Transition to the designated maintenance site

5.14)**Software configuration management** . This subclause should be divided into the following to describe the approach to be followed for software configuration management.

5.14.1)Configuration identification

5.14.2)Configuration control

5.14.3)Configuration status accounting

5.14.4)Configuration audits

5.14.5)Packaging, storage, handling, and delivery

5.15)**Software product evaluation** . This subclause should be divided into the following to describe the approach to be followed for software product evaluation.

5.15.1)In-process and final software product evaluations

5.15.2)Software product evaluation records, including items to be recorded

5.15.3)Independence in software product evaluation

5.16)**Software quality assurance** . This subclause should be divided into the following to describe the approach to be followed for software quality assurance.

5.16.1)Software quality assurance evaluations

5.16.2)Software quality assurance records, including items to be recorded

5.16.3)Independence in software quality assurance

5.17)**Corrective action** . This subclause should be divided into the following to describe the approach to be followed for corrective action.

5.17.1)Problem/change reports, including items to be recorded (candidate items include project name, originator, problem number, problem name, software element or document affected, origination date, category and priority, description, analyst assigned to the problem, date assigned, date completed, analysis time, recommended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, description of solution implemented)

5.17.2)Corrective action system

5.18)**Joint technical and management reviews** . This subclause should be divided into the following to describe the approach to be followed for joint technical and management reviews.

5.18.1)Joint technical reviews, including a proposed set of reviews

5.18.2)Joint management reviews, including a proposed set of reviews

5.19)**Risk management** . This subclause shall describe the approach to be followed for risk management.

5.20)**Software management indicators** . This subclause shall describe the approach to be followed for use of software management indicators.

5.21)**Administrative security and privacy protection** . This subclause shall describe the approach to be followed for administrative security and privacy protection.

5.22)**Managing subcontractors** . This subclause shall describe the approach to be followed for managing subcontractors.

5.23)**Interfacing with software IV&V agents** . This subclause shall describe the approach to be followed for interfacing with IV&V agents or auditors.

5.24) **Coordinating with associate developers** . This subclause shall describe the approach to be followed for coordinating with associate developers.

5.25) **Project process improvement** . This subclause shall describe the approach to be followed for project process improvement.

6) **Schedules and activity network** . This clause shall present:
   a) Schedule(s) identifying the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity
   b) An activity network, depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project

7) **Project organization and resources** . This clause should be divided into the following subclauses to describe the project organization and resources to be applied in each build.

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

   7.2) **Project resources** . This subclause shall describe the resources to be applied to the project. It shall include, as applicable:
      a) Personnel resources, including:
      1) The estimated staff-loading for the project (number of personnel over time)
      2) The breakdown of the staff-loading numbers by responsibility (for example, management, software engineering, software testing, software configuration management, software product evaluation, software quality assurance)
      3) A breakdown of the skill levels, geographic locations, and security clearances of personnel performing each responsibility
      b) Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.
      c) Acquirer-furnished equipment, software, services, documentation, data, and facilities required for the contracted effort. A schedule detailing when these items will be needed shall also be included.
      d) Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

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

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

E.2.2 Contents of the Software Test Plan (STP)

SOFTWARE TEST PLAN (STP)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
   1.4) Relationship to other plans
2) Referenced documents

3) Software test environment
   3.x) (Name of test site(s))
      3.x.1)Software
      3.x.2)Hardware and firmware
      3.x.3)Other materials
      3.x.4)Proprietary nature, acquirer's rights, and licensing
      3.x.5)Installation, testing, and control
      3.x.6)Participating organizations
      3.x.7)Personnel
      3.x.8)Orientation plan
      3.x.9)Tests to be performed
4) Test identification
   4.1) General information
      4.1.1)Test levels
      4.1.2)Test classes
      4.1.3)General test conditions
      4.1.4)Test progression
      4.1.5)Data recording, reduction, and analysis
   4.2) Planned tests
      4.2.x)(Item(s) to be tested)
      4.2.x.y)(Project-unique identifier of a test)
5) Test schedules
6) Requirements traceability
7) Notes
A) Annexes

1) **Scope** . This clause should be divided into the following subclauses.
   1.1) **Identification** . This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** . This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** . This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
   1.4) **Relationship to other plans** . This subclause shall describe the relationship, if any, of the STP to related project management plans.
2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Software test environment** . This clause should be divided into the following subclauses to describe the software test environment at each intended test site. Reference may be made to the Software Development Plan (SDP) for resources that are described there.
   3.x) **(Name of test site(s))** . This subclause shall identify one or more test sites to be used for the testing, and should be divided into the following to describe the software test environment at the site(s). If all tests will be conducted at a single site, this subclause and its subclauses shall be presented only once. If multiple test sites use the same or similar software test environments, they may be discussed together. Duplicative information among test site descriptions may be reduced by referencing earlier descriptions.
      3.x.1)**Software** . This subclause shall identify by name, number, and version, as applicable, the software (e.g., operating systems, compilers, communications software, related applications software, databases, input files, code auditors, dynamic path analyzers, test drivers, preprocessors, test data generators, test control software, other special test software, post-

processors) necessary to perform the planned testing activities at the test site(s). This subclause shall describe the purpose of the software, describe its media (tape, disk, etc.), identify those that are expected to be supplied by the site, and identify any classified processing or other security or privacy protection issues associated with the software.

3.x.2)**Hardware and firmware** . This subclause shall identify by name, number, and version, as applicable, the computer hardware, interfacing equipment, communications equipment, test data reduction equipment, apparatus such as extra peripherals (tape drives, printers, plotters), test message generators, test timing devices, test event records, etc.,and firmware that will be used in the software test environment at the test site(s). This subclause shall describe the purpose of the hardware or firmware, state the period of usage and the number of each needed, identify those that are expected to be supplied by the site, and identify any classified processing or other security or privacy protection issues associated with the hardware or firmware.

3.x.3)**Other materials** . This subclause shall identify and describe any other materials needed for the testing at the test site(s). These materials may include manuals, software listings, media containing the software to be tested, media containing data to be used in the tests, sample listings of output, and other forms or instructions. This subclause shall identify those items that are to be delivered to the site and those that are expected to be supplied by the site. The description shall include the type, layout, and quantity of the materials, as applicable. This subclause shall identify any classified processing or other security or privacy protection issues associated with the items.

3.x.4)**Proprietary nature, acquirer's rights, and licensing** . This subclause shall identify the proprietary nature, acquirer's rights, and licensing issues associated with each element of the software test environment.

3.x.5)**Installation, testing, and control** . This subclause shall identify the developer's plans for performing each of the following, possibly in conjunction with personnel at the test site(s):

a) Acquiring or developing each element of the software test environment
b) Installing and testing each item of the software test environment prior to its use
c) Controlling and maintaining each item of the software test environment

3.x.6)**Participating organizations** . This subclause shall identify the organizations that will participate in the testing at the test sites(s) and the roles and responsibilities of each.

3.x.7)**Personnel** . This subclause shall identify the number, type, and skill level of personnel needed during the test period at the test site(s), the dates and times they will be needed, and any special needs, such as multishift operation and retention of key skills to ensure continuity and consistency in extensive test programs.

3.x.8)**Orientation plan** . This subclause shall describe any orientation and training to be given before and during the testing. This information shall be related to the personnel needs given in 3.x.7. This training may include user instruction, operator instruction, maintenance and control group instruction, and orientation briefings to staff personnel. If extensive training is anticipated, a separate plan may be developed and referenced here.

3.x.9)**Tests to be performed** . This subclause shall identify, by referencing clause 4, the tests to be performed at the test site(s).

4) **Test identification** . This clause should be divided into the following subclauses to identify and describe each test to which this STP applies.

4.1) **General information** . This subclause should be divided into subclauses to present general information applicable to the overall testing to be performed.

4.1.1)**Test levels** . This subclause shall describe the levels at which testing will be performed, for example, software item level or system level.

4.1.2)**Test classes** . This subclause shall describe the types or classes of tests that will be performed (for example, timing tests, erroneous input tests, maximum capacity tests).

4.1.3)**General test conditions** . This subclause shall describe conditions that apply to all of the tests or to a group of tests. For example: "each test shall include nominal, maximum, and minimum values;" "each test of type x shall use live data;" "execution size and time shall be measured for each software item." Included shall be a statement of the extent of testing

to be performed and rationale for the extent selected. The extent of testing shall be expressed as a percentage of some well defined total quantity, such as the number of samples of discrete operating conditions or values, or other sampling approach. Also included shall be the approach to be followed for retesting/regression testing.

4.1.4)**Test progression** . In cases of progressive or cumulative tests, this subclause shall explain the planned sequence or progression of tests.

4.1.5)**Data recording, reduction, and analysis** . This subclause shall identify and describe the data recording, reduction, and analysis procedures to be used during and after the tests identified in this STP. These procedures shall include, as applicable, manual, automatic, and semi-automatic techniques for recording test results, manipulating the raw results into a form suitable for evaluation, and retaining the results of data reduction and analysis.

4.2) **Planned tests** . This subclause should be divided into the following to describe the total scope of the planned testing.

4.2.x)**(Item(s) to be tested)** . This subclause shall identify a software item, subsystem, system, or other entity by name and project-unique identifier, and should be divided into the following to describe the testing planned for the item(s). (Note: the "tests" in this plan are collections of test cases. There is no intent to describe each test case in this document.)

4.2.x.y)**(Project-unique identifier of a test)** . This subclause shall identify a test by project-unique identifier and shall provide the information specified below for the test. Reference may be made as needed to the general information in 4.1.

a) Test objective
b) Test level
c) Test type or class
d) Qualification method(s) as specified in the requirements specification
e) Identifier of the software item requirements and, if applicable, software system requirements addressed by this test. (Alternatively, this information may be provided in clause 6.)
f) Special requirements (for example, 48 hours of continuous facility time, simulation, extent of test, use of a special input or database)
g) Type of data to be recorded
h) Type of data recording/reduction/analysis to be employed
i) Assumptions and constraints, such as anticipated limitations on the test due to system or test conditions—timing, interfaces, equipment, personnel, database, etc.
j) Safety, security, and privacy protection considerations associated with the test

5) **Test schedules** . This clause shall contain or reference the schedules for conducting the tests identified in this plan. It shall include:

a) A listing or chart depicting the sites at which the testing will be scheduled and the time frames during which the testing will be conducted
b) A schedule for each test site depicting the activities and events listed below, as applicable, in chronological order with supporting narrative as necessary:
   1) On-site test period and periods assigned to major portions of the testing
   2) Pretest on-site period needed for setting up the software test environment and other equipment, system debugging, orientation, and familiarization
   3) Collection of database/data file values, input values, and other operational data needed for the testing
   4) Conducting the tests, including planned retesting
   5) Preparation, review, and approval of the Software Test Report (STR)

6) **Requirements traceability** . This subclause shall contain:

a) Traceability from each test identified in this plan to the software item requirements and, if applicable, software system requirements it addresses. (Alternatively, this traceability may be provided in 4.2.x.y and referenced from this subclause.)
b) Traceability from each software item requirement and, if applicable, each software system requirement covered by this test plan to the test(s) that address it. The traceability shall cover the software item requirements in all applicable Software Requirements Specifications (SRSs) and associated Interface Requirements Specifications (IRSs), and, for software systems, the

system requirements in all applicable System/Subsystem Specifications (SSS) and associated system-level IRSs.

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

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

E.2.3 Contents of the Software Installation Plan (SIP)

SOFTWARE INSTALLATION PLAN (SIP)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
   1.4) Relationship to other plans
2) Referenced documents
3) Installation overview
   3.1) Description
   3.2) Contact point
   3.3) Support materials
   3.4) Training
   3.5) Tasks
   3.6) Personnel
   3.7) Security and privacy protection
4) Site-specific information for software center operations staff
   4.x) (Site name)
      4.x.1)Schedule
      4.x.2)Software inventory
      4.x.3)Facilities
      4.x.4)Installation team
      4.x.5)Installation procedures
      4.x.6)Data update procedures
5) Site-specific information for software users
   5.x) (Site name)
      5.x.1)Schedule
      5.x.2)Installation procedures
      5.x.3)Data update procedures
6) Notes

A) Annexes

1) **Scope** . This clause should be divided into the following subclauses.

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

   1.2) **System overview** . This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software;

summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

1.3) **Document overview** . This subclause shall summarize the purpose and contents of this plan and shall describe any security or privacy protection considerations associated with its use.

1.4) **Relationship to other plans** . This subclause shall describe the relationship, if any, of the SIP to other project management plans.

2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Installation overview** . This clause should be divided into the following subclauses to provide an overview of the installation process.

3.1) **Description** . This subclause shall provide a general description of the installation process to provide a frame of reference for the remainder of the document. A list of sites for software installation, the schedule dates, and the method of installation shall be included.

3.2) **Contact point** . This subclause shall provide the organizational name, department/division, and telephone number of a point of contact for questions relating to this installation.

3.3) **Support materials** . This subclause shall list the type, source, and quantity of support materials needed for the installation. Included shall be items such as magnetic tapes, disk packs, computer printer paper, and special forms.

3.4) **Training** . This subclause shall describe the developer's plans for training personnel who will operate and/or use the software installed at user sites. Included shall be the delineation between general orientation, classroom training, and "hands-on" training.

3.5) **Tasks** . This subclause shall list and describe in general terms each task involved in the software installation. Each task description shall identify the organization that will accomplish the task, usually either the user, computer operations, or the developer. The task list shall include such items as:

a) Providing overall planning, coordination, and preparation for installation
b) Providing personnel for the installation team
c) Arranging lodging, transportation, and office facilities for the installation team
d) Ensuring that all manuals applicable to the installation are available when needed
e) Ensuring that all other prerequisites have been fulfilled prior to the installation
f) Planning and conducting training activities
g) Providing students for the training
h) Providing computer support and technical assistance for the installation
i) Providing for conversion from the current system

3.6) **Personnel** . This subclause shall describe the number, type, and skill level of the personnel needed during the installation period, including the need for multishift operation, clerical support, etc.

3.7) **Security and privacy protection** . This subclause shall contain an overview of the security and privacy protection considerations associated with the system.

4) **Site-specific information for software center operations staff** . This clause applies if the software will be installed in computer center(s) or other centralized or networked software installations for users to access via terminals or using batch input/output. If this type of installation does not apply, this clause shall contain the words "Not applicable."

4.x) **(Site name)** . This subclause shall identify a site or set of sites and should be divided into the following to discuss those sites. Multiple sites may be discussed together when the information for those sites is generally the same.

4.x.1)**Schedule** . This subclause shall present a schedule of tasks to be accomplished during installation. It shall depict the tasks in chronological order with beginning and ending dates of each task and supporting narrative as necessary.

4.x.2)**Software inventory** . This subclause shall provide an inventory of the software needed to support the installation. The software shall be identified by name, identification number, version number, release number, configuration, and security classification, as applicable. This subclause shall indicate whether the software is expected to be on site or will be

delivered for the installation and shall identify any software to be used only to facilitate the installation process.

4.x.3)**Facilities** . This subclause shall detail the physical facilities and accommodations needed during the installation period. This description shall include the following, as applicable:

a) Classroom, work space, and training aids needed, specifying hours per day, number of days, and shifts
b) Hardware that is required to be operational and available
c) Transportation and lodging for the installation team

4.x.4)**Installation team** . This subclause shall describe the composition of the installation team. Each team member's tasks shall be defined.

4.x.5)**Installation procedures** . This subclause shall provide step-by-step procedures for accomplishing the installation. References may be made to other documents, such as operator manuals. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable. The procedures shall include the following, as applicable:

a) Installing the software
b) Checking out the software once installed
c) Initializing databases and other software with site-specific data
d) Conversion from the current system, possibly involving running in parallel
e) Dry run of the procedures in operator and user manuals

4.x.6)**Data update procedures** . This subclause shall present the data update procedures to be followed during the installation period. When the data update procedures are the same as normal updating or processing procedures, reference may be made to other documents, such as operator manuals.

5) **Site-specific information for software users** . This clause shall provide installation planning pertinent to users of the software. When more than one type of user is involved, for example, users at different positions, performing different functions, or in different organizations, a separate clause (clauses 5 through n) may be written for each type of user and the clause titles modified to reflect each user.

5.x) **(Site name)** . This subclause shall identify a site or set of sites and should be divided into the following to discuss those sites. Multiple sites may be discussed together when the information for those sites is generally the same.

5.x.1)**Schedule** . This subclause shall present a schedule of tasks to be accomplished by the user during installation. It shall depict the tasks in chronological order including beginning and ending dates for each task and supporting narrative as necessary.

5.x.2)**Installation procedures** . This subclause shall provide step-by-step procedures for accomplishing the installation. Reference may be made to other documents, such as user manuals. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable. The procedures shall include the following, as applicable:

a) Performing the tasks under 4.x.5 if not performed by operations staff
b) Initializing user-specific data
c) Setting up queries and other user inputs
d) Performing sample processing
e) Generating sample reports
f) Conversion from the current system, possibly involving running in parallel
g) Dry run of procedures in user manuals

5.x.3)**Data update procedures** . This subclause should be divided into subclauses to present the user's data update procedures to be followed during the installation period. When update procedures are the same as normal processing, reference may be made to other documents, such as user manuals, and to clause 4 of this document

6) **Notes** . This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document. If clause 5 has been expanded into clause(s) 6, ...n, this clause shall be numbered as the next clause following clause n.A.

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

E.2.4 Contents of the Software Transition Plan (STrP)

SOFTWARE TRANSITION PLAN (STrP)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
   1.4) Relationship to other plans
2) Referenced documents
3) Software maintenance resources
   3.1) Facilities
   3.2) Hardware
   3.3) Software
   3.4) Other documentation
   3.5) Personnel
   3.6) Other resources
   3.7) Interrelationship of components
4) Recommended procedures
5) Training
6) Anticipated areas of change
7) Transition planning
8) Notes

A) Annexes

1) **Scope** This clause should be divided into the following subclauses.
   1.1) **Scope Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
   1.4) **Relationship to other plans** .This subclause shall describe the relationship, if any, of the STrP to other project management plans.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Software maintenance resources** .This clause should be divided into subclauses to identify and describe the resources needed to maintain the deliverable software. These resources shall include items needed to control, copy, and distribute the software and its documentation, and to specify, design, implement, document, test, evaluate, control, copy, and distribute modifications to the software.

3.1) **Facilities** .This subclause shall describe the facilities needed to maintain the deliverable software. These facilities may include special buildings, rooms, mock-ups, building features such as raised flooring or cabling; building features to support security and privacy protection requirements (electromagnetic shielding, vaults, etc.), building features to support safety requirements (smoke alarms, safety glass, etc.), special power requirements, and so on. The purpose of each item shall be described. Diagrams may be included as applicable.

3.2) **Hardware** .This subclause shall identify and describe the hardware and associated documentation needed to maintain the deliverable software. This hardware may include computers, peripheral equipment, hardware simulators, stimulators, emulators, diagnostic equipment, and non-computer equipment. The description shall include:

a) Specific models, versions, and configurations
b) Rationale for the selected hardware
c) Reference to user/operator manuals or instructions for each item, as applicable
d) Identification of each hardware item and document as acquirer-furnished, an item that will be delivered to the maintenance organization, an item the maintenance organization is known to have, an item the maintenance organization is required to acquire, or other description of status
e) If items are required to be acquired, information about a current source of supply, including whether the item is currently available and whether it is expected to be available at the time of delivery
f) Information about manufacturer support, licensing, and usage and ownership rights, including whether the item is currently supported by the manufacturer, whether it is expected to be supported at the time of delivery, whether licenses will be assigned to the maintenance organization, and the terms of such licenses
g) Security and privacy protection considerations, limitations, or other items of interest

3.3) **Software** .This subclause shall identify and describe the software and associated documentation needed to maintain the deliverable software. This software may include computer-aided software engineering (CASE) tools, data in these tools, compilers, test tools, test data, simulations, emulations, utilities, configuration management tools, databases and data files, and other software. The description shall include:

a) Specific names, identification numbers, version numbers, release numbers, and configurations, as applicable
b) Rationale for the selected software
c) Reference to user/operator manuals or instructions for each item, as applicable
d) Identification of each item of software and document as acquirer-furnished, an item that will be delivered to the maintenance organization, an item the maintenance organization is known to have, an item the maintenance organization is required to acquire, or other description of status
e) If items are required to be acquired, information about a current source of supply, including whether the item is currently available and whether it is expected to be available at the time of delivery
f) Information about vendor support, licensing, and usage and ownership rights, including whether the item is currently supported by the vendor, whether it is expected to be supported at the time of delivery, whether licenses will be assigned to the maintenance organization, and the terms of such licenses
g) Security and privacy protection considerations, limitations, or other items of interest

3.4) **Other documentation** .This subclause shall identify any other documentation needed to maintain the deliverable software. The list will include, for example, plans, reports, studies, specifications, design descriptions, test cases/procedures, test reports, user/operator manuals, and maintenance manuals for the deliverable software. This subclause shall provide:

a) Names, identification numbers, version numbers, and release numbers, as applicable
b) Rationale for including each document in the list

c) Identification of each document as acquirer-furnished, an item that will be delivered to the maintenance organization, an item the maintenance organization is known to have, an item the maintenance organization is required to acquire, or other description of status
d) If a document is required to be acquired, information about where to acquire it
e) Information about licensing and usage and ownership rights
f) Security and privacy protection considerations, limitations, or other items of interest

3.5) **Personnel** .This subclause shall describe the personnel needed to maintain the deliverable software, including anticipated number of personnel, types and levels of skills and expertise, and security clearances. This subclause shall cite, as applicable, actual staffing on the development project as a basis for the staffing needs cited.

3.6) **Other resources** .This subclause shall identify any other resources needed to maintain the deliverable software. Included may be consumable such as magnetic tapes and diskettes, together with an estimate of the type and number that should be acquired.

3.7) **Interrelationship of components** .This subclause shall identify the interrelationships of the components identified in the preceding subclauses. A figure may be used to show the interrelationships.

4) **Recommended procedures** .This clause should be divided into subclauses as needed to describe any procedures, including advice and lessons learned, that the developer may wish to recommend to the maintenance organization for maintaining the deliverable software and associated maintenance environment.

5) **Training** .This clause should be divided into subclauses as appropriate to describe the developer's plans for training maintenance personnel to maintain the deliverable software. This clause shall include:
a) The schedule, duration, and location for the training
b) The delineation between classroom training and "hands-on" training
c) Provision (either directly or by reference) for:
1) Familiarization with the operational software and target computer(s)
2) Familiarization with the maintenance software and host system

6) **Anticipated areas of change** .This clause shall describe anticipated areas of change to the deliverable software.

7) **Transition planning.** This clause should be divided into subclauses as needed to describe the developer's plans for transitioning the deliverable software to the maintenance organization. This clause shall address the following:
a) All activities to be performed to transition the deliverable software to the maintenance organization. These activities may include planning/coordination meetings; preparation of items to be delivered to the maintenance organization; packaging, shipment, installation, and checkout of the software maintenance environment; packaging, shipment, installation, and checkout of the operational software; and training of maintenance personnel.
b) Roles and responsibilities for each activity
c) The resources needed to carry out the transition activities and the source from which each resource will be provided
d) Schedules and milestones for conducting the transition activities. These schedules and milestones shall be compatible with the contract master schedule.
e) Procedures for installation and checkout of deliverable items in the maintenance environment

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

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

Annex F

(normative)

Concept and requirements software product descriptions

F.1 Scope

This annex specifies the contents of software products associated with concept description and requirements specification in this standard. It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

F.2 Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in the order they are referenced in this standard and include the following:

a) Operational Concept Description (OCD)
b) System/Subsystem Specification (SSS)
c) Interface Requirements Specification (IRS)
d) Software Requirements Specification (SRS)

F.2.1 Contents of the Operational Concept Description (OCD)

OPERATIONAL CONCEPT DESCRIPTION (OCD)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Current system or situation
   3.1) Background, objectives, and scope
   3.2) Operational policies and constraints
   3.3) Description of current system or situation
   3.4) Users or involved personnel
   3.5) Support strategy
4) Justification for and nature of changes
   4.1) Justification for change
   4.2) Description of needed changes
   4.3) Priorities among the changes
   4.4) Changes considered but not included
   4.5) Assumptions and constraints
5) Concept for a new or modified system

5.1) Background, objectives, and scope
5.2) Operational policies and constraints
5.3) Description of the new or modified system
5.4) Users/affected personnel
5.5) Support strategy
6) Operational scenarios
7) Summary of impacts
7.1) Operational impacts
7.2) Organizational impacts
7.3) Impacts during development
8) Analysis of the proposed system
8.1) Summary of advantages
8.2) Summary of disadvantages/limitations
8.3) Alternatives and trade-offs considered
9) Notes
A) Annexes
1) **Scope** . This clause should be divided into the following subclauses.
1.1) **Identification** . This subclause shall contain a full identification of the system to which this document applies, including, as applicable, identification number(s), title(s), abbreviations), version number(s), and release number(s).
1.2) **System overview** . This subclause shall briefly state the purpose of the system to which this document applies. It shall describe the general nature of the system; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
1.3) **Document overview** . This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Current system or situation** . This clause should be divided into the following subclauses to describe the system or situation as it currently exists.
3.1) **Back-ground, objectives, and scope** . This subclause shall describe the background, mission or objectives, and scope of the current system or situation.
3.2) **Operational policies and constraints** . This subclause shall describe any operational policies and constraints that apply to the current system or situation.
3.3) **Description of current system or situation** . This subclause shall provide a description of the current system or situation, identifying differences associated with different states or modes of operation (for example, regular, maintenance, training, degraded, emergency, alternative-site, wartime, peacetime). The distinction between states and modes is arbitrary. A system may be described in terms of states only, modes only, states within modes, modes within states, or any other scheme that is useful. If the system operates without states or modes, this subclause shall so state, without the need to create artificial distinctions. The description shall include, as applicable:
a) The operational environment and its characteristics
b) Major system components and the interconnections among these components
c) Interfaces to external systems or procedures
d) Capabilities/functions of the current system
e) Charts and accompanying descriptions depicting input, output, data flow, and manual and automated processes sufficient to understand the current system or situation from the user's point of view
f) Performance characteristics, such as speed, throughput, volume, frequency
g) Quality attributes, such as reliability, maintainability, availability, flexibility, portability, usability, efficiency

h) Provisions for safety, security, privacy protection, and continuity of operations in emergencies

3.4) **Users or involved personnel** . This subclause shall describe the types of users of the system, or personnel involved in the current situation, including, as applicable, organizational structures, training/skills, responsibilities, activities, and interactions with one another.

3.5) **Support strategy** . This subclause shall provide an overview of the support strategy for the current system, including, as applicable to this document, maintenance organization(s); facilities; equipment; maintenance software; repair/replacement criteria; maintenance levels and cycles; and storage, distribution, and supply methods.

4) **Justification for and nature of changes** . This clause should be divided into the following subclauses.

4.1) **Justification for change** . This subclause shall:

a) Describe new or modified aspects of user needs, threats, missions, objectives, environments, interfaces, personnel or other factors that require a new or modified system

b) Summarize deficiencies or limitations in the current system or situation that make it unable to respond to these factors

4.2) **Description of needed changes** . This subclause shall summarize new or modified capabilities/functions, processes, interfaces, or other changes needed to respond to the factors identified in 4.1.

4.3) **Priorities among the changes** . This subclause shall identify priorities among the needed changes. It shall, for example, identify each change as essential, desirable, or optional, and prioritize the desirable and optional changes.

4.4) **Changes considered but not included** . This subclause shall identify changes considered but not included in 4.2, and rationale for not including them.

4.5) **Assumptions and constraints** . This subclause shall identify any assumptions and constraints applicable to the changes identified in this clause.

5) **Concept for a new or modified system** . This clause should be divided into the following subclauses to describe a new or modified system.

5.1) **Background, objectives, and scope** . This subclause shall describe the background, mission or objectives, and scope of the new or modified system.

5.2) **Operational policies and constraints** . This subclause shall describe any operational policies and constraints that apply to the new or modified system.

5.3) **Description of the new or modified system** . This subclause shall provide a description of the new or modified system, identifying differences associated with different states or modes of operation (for example, regular, maintenance, training, degraded, emergency, alternative-site, wartime, peacetime). The distinction between states and modes is arbitrary. A system may be described in terms of states only, modes only, states within modes, modes within states, or any other scheme that is useful. If the system operates without states or modes, this subclause shall so state, without the need to create artificial distinctions. The description shall include, as applicable:

a) The operational environment and its characteristics

b) Major system components and the interconnections among these components

c) Interfaces to external systems or procedures

d) Capabilities/functions of the new or modified system

e) Charts and accompanying descriptions depicting input, output, data flow, and manual and automated processes sufficient to understand the new or modified system or situation from the user's point of view

f) Performance characteristics, such as speed, throughput, volume, frequency

g) Quality attributes, such as reliability, maintainability, availability, flexibility, portability, usability, efficiency

h) Provisions for safety, security, privacy protection, and continuity of operations in emergencies

5.4) **Users/affected personnel** . This subclause shall describe the types of users of the new or modified system, including, as applicable, organizational structures, training/skills, responsibilities, and interactions with one another.

5.5) **Support strategy** . This subclause shall provide an overview of the support strategy for the new or modified system, including, as applicable, maintenance organizations); facilities; equipment; maintenance software; repair/replacement criteria; maintenance levels and cycles; and storage, distribution, and supply methods.

6) **Operational scenarios** . This clause shall describe one or more operational scenarios that illustrate the role of the new or modified system, its interaction with users, its interface to other systems, and all states or modes identified for the system. The scenarios shall include events, actions, stimuli, information, interactions, etc., as applicable. Reference may be made to other media, such as videos, to provide part or all of this information.

7) **Summary of impacts** . This clause should be divided into the following subclauses.

7.1) **Operational impacts** . This subclause shall describe anticipated operational impacts on the user, acquirer, developer, and maintenance organizations. These impacts may include changes in interfaces with computer operating centers; change in procedures; use of new data sources; changes in quantity, type, and timing of data to be input to the system; changes in data retention requirements; and new modes of operation based on peacetime, alert, wartime, or emergency conditions.

7.2) **Organizational impacts** . This subclause shall describe anticipated organizational impacts on the user, acquirer, developer, and maintenance organizations. These impacts may include modification of responsibilities; addition or elimination of responsibilities or positions; need for training or retraining; and changes in number, skill levels, position identifiers, or location of personnel in various modes of operation.

7.3) **Impacts during development** . This subclause shall describe anticipated impacts on the user, acquirer, developer, and maintenance organizations during the development effort. These impacts may include meetings/discussions regarding the new system; development or modification of databases; training; parallel operation of the new and existing systems; impacts during testing of the new system; and other activities needed to aid or monitor development.

8) **Analysis of the proposed system** . This clause should be divided into the following subclauses.

8.1) **Summary of advantages** . This subclause shall provide a qualitative and quantitative summary of the advantages to be obtained from the new or modified system. This summary shall include new capabilities, enhanced capabilities, and improved performance, as applicable, and their relationship to deficiencies identified in 4.1.

8.2) **Summary of disadvantages/limitations** . This subclause shall provide a qualitative and quantitative summary of disadvantages or limitations of the new or modified system. These disadvantages and limitations shall include, as applicable, degraded or missing capabilities, degraded or less-than-desired performance, greater-than-desired use of computer hardware resources, undesirable operational impacts, conflicts with user assumptions, and other constraints.

8.3) **Alternatives and trade-offs considered** . This subclause shall identify and describe major alternatives considered to the system or its characteristics, the trade-offs among them, and rationale for the decisions reached.

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

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

F.2.2 Contents of the System/Subsystem Specification (SSS)

SYSTEM/SUBSYSTEM SPECIFICATION (SSS)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Requirements
   3.1) Required states and modes
   3.2) System capability requirements
      3.2.x)(System capability)
   3.3) System external interface requirements
      3.3.1)Interface identification and diagrams
      3.3.2)(Project-unique identifier of interface)
   3.4) System internal interface requirements
   3.5) System internal data requirements
   3.6) Adaptation requirements
   3.7) Safety requirements
   3.8) Security and privacy protection requirements
   3.9) System environment requirements
   3.10)Computer resource requirements
      3.10.1)Computer hardware requirements
      3.10.2)Computer hardware resource utilization requirements
      3.10.3)Computer software requirements
      3.10.4)Computer communications requirements
   3.11)System quality factors
   3.12)Design and construction constraints
   3.13)Personnel-related requirements
   3.14)Training-related requirements
   3.15)Logistics-related requirements
   3.16)Other requirements
   3.17)Packaging requirements
   3.18)Precedence and criticality of requirements
4) Qualification provisions
5) Requirements traceability
6) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system to which this document applies. It shall describe the general nature of the system; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Requirements** .This clause should be divided into the following subclauses to specify the system requirements, that is, those characteristics of the system that are conditions for its acceptance. Each requirement shall be assigned a project-unique identifier to support testing and traceability and shall be stated in such a way that an objective test can be defined for it. Each requirement shall be annotated with associated qualification method(s) (see clause 4) and, for subsystems, traceability to system requirements (see 5.a), if not provided in those clauses. The degree of detail to be provided shall be guided by the following rule: include those characteristics of the system that are conditions for system acceptance; defer to design descriptions those characteristics that the acquirer is willing to leave up to the developer. If there are no requirements in a given subclause, the subclause shall so state. If a given requirement fits into more than one subclause, it may be stated once and referenced from the other subclauses. Conventions needed to understand the requirements shall be presented or referenced.

   3.1) **Required states and modes** .If the system is required to operate in more than one state or mode having requirements distinct from other states or modes, this subclause shall identify and define each state and mode. Examples of states and modes include: idle, ready, active, post-use analysis, training, degraded, emergency, backup, wartime, peacetime. The distinction between states and modes is arbitrary. A system may be described in terms of states only, modes only, states within modes, modes within states, or any other scheme that is useful. If no states or modes are required, this subclause shall so state, without the need to create artificial distinctions. If states and/or modes are required, each requirement or group of requirements in this specification shall be correlated to the states and modes. The correlation may be indicated by a table or other method in this subclause, in an annex referenced from this subclause, or by annotation of the requirements in the subclauses where they appear.

   3.2) **System capability requirements** .This subclause should be divided into subclauses to itemize the requirements associated with each capability of the system. A "capability" is defined as a group of related requirements. The word "capability" may be replaced with "function," "subject," "object," or other term useful for presenting the requirements.

      3.2.x)**(System capability)** .This subclause shall identify a required system capability and shall itemize the requirements associated with the capability. If the capability can be more clearly specified by dividing it into constituent capabilities, the constituent capabilities shall be specified in subclauses. The requirements shall specify required behavior of the system and shall include applicable parameters, such as response times, throughput times, other timing constraints, sequencing, accuracy, capacities (how much/how many), priorities, continuous operation requirements, and allowable deviations based on operating conditions. The requirements shall include, as applicable, required behavior under unexpected, unallowed, or "out of bounds" conditions, requirements for error handling, and any provisions to be incorporated into the system to provide continuity of operations in the event of emergencies. Subclause 3.3.x of the SSS provides a list of topics to be considered when specifying requirements regarding input the system is required to accept and output it is required to produce.

   3.3) **System external interface requirements** .This subclause should be divided into subclauses to specify the requirements, if any, for the system's external interfaces. This subclause may reference one or more Interface Requirements Specifications (IRSs) or other documents containing these requirements.

      3.3.1)**Interface identification and diagrams** .This subclause shall identify the required external interfaces of the system. The identification of each interface shall include a project-unique identifier and shall designate the interfacing entities (systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided to depict the interfaces.

3.3.2)(**Project-unique identifier of interface)** .This subclause (beginning with 3.3.2) shall identify a system external interface by project-unique identifier, shall briefly identify the interfacing entities, and should be divided into subclauses as needed to state the requirements imposed on the system to achieve the interface. Interface characteristics of the other entities involved in the interface shall be stated as assumptions or as "When [the entity not covered] does this, the system shall ...," not as requirements on the other entities. This subclause may reference other documents (such as data dictionaries, standards for communication protocols, and standards for user interfaces) in place of stating the information here. The requirements shall include the following, as applicable, presented in any order suited to the requirements, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority that the system is required to assign the interface
b) Requirements on the type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Required characteristics of individual data elements that the system is required to provide, store, send, access, receive, etc., such as:

1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Standard data element name
d) Technical name (e.g., variable or field name in code or database)
e) Abbreviation or synonymous names
2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0-99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)

d) Required characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the system is required to provide, store, send, access, receive, etc., such as:

1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)

e) Required characteristics of communication methods that the system is required to use for the interface, such as:

1) Project-unique identifiers)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting

4) Flow control (such as sequence numbering and buffer allocation)
5) Data transfer rate, whether periodic/periodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing

f) Required characteristics of protocols the system is required to use for the interface, such as:

1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features

g) Other required characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

3.4) **System internal interface requirements** .This subclause shall specify the requirements, if any, imposed on interfaces internal to the system. If all internal interfaces are left to the design or to requirement specifications for system components, this fact shall be so stated. If such requirements are to be imposed, subclause 3.3 of the SSS provides a list of topics to be considered.

3.5) **System internal data requirements** .This subclause shall specify the requirements, if any, imposed on data internal to the system. Included shall be requirements, if any, on databases and data files to be included in the system. If all decisions about internal data are left to the design or to requirements specifications for system components, this fact shall be so stated. If such requirements are to be imposed, subclauses 3.3.x.c and 3.3.x.d of the SSS provide a list of topics to be considered.

3.6) **Adaptation requirements** .This subclause shall specify the requirements, if any, concerning installation-dependent data that the system is required to provide (such as site-dependent latitude and longitude or site-dependent state tax codes) and parameters that the system is required to use that may vary according to operational needs (such as operation-dependent targeting constants or site-dependent month-end dates).

3.7) **Safety requirements** .This subclause shall specify the system requirements, if any, concerned with preventing or minimizing unintended hazards to personnel, property, and the physical environment. Examples include restricting the use of dangerous materials; classifying explosives for purposes of shipping, handling, and storing; abort/escape provisions from enclosures; gas detection and warning devices; grounding of electrical systems; decontamination; and explosion proofing. This subclause shall include the system requirements, if any, for nuclear components, including, as applicable, requirements for component design, prevention of inadvertent detonation, and compliance with nuclear safety rules.

3.8) **Security and privacy protection requirements** .This subclause shall specify the system requirements, if any, concerned with maintaining security and privacy protection. The requirements shall include, as applicable, the security/privacy protection environment in which the system is required to operate, the type and degree of security or privacy protection to be provided, the security/privacy protection risks the system is required to withstand, required safeguards to reduce those risks, the security/privacy protection policy that is required to be met, the security/privacy protection accountability the system is required to provide, and the criteria that are required to be met for security/privacy protection certification/accreditation.

3.9) **System environment requirements** .This subclause shall specify the requirements, if any, regarding the environment in which the system is required to operate. Examples for a software system are the computer hardware and operating system on which the software is required to run. (Additional requirements concerning computer resources are given in the next subclause). Examples for a hardware-software system include the environmental conditions that the system is required to withstand during transportation, storage, and operation, such as conditions in the natural environment (wind, rain, temperature, geographic location), the induced environment

(motion, shock, noise, electromagnetic radiation), and environments due to hostile action (explosions, radiation).

3.10) **Computer resource requirements** .This subclause should be divided into the following. Depending upon the nature of the system, the computer resources covered in these subclauses may constitute the environment of the system (as for a software system) or components of the system (as for a hardware-software system).

3.10.1) **Computer hardware requirements** .This subclause shall specify the requirements, if any, regarding computer hardware that is required to be used by, or incorporated into, the system. The requirements shall include, as applicable, number of each type of equipment, type, size, capacity, and other required characteristics of processors, memory, input/output devices, auxiliary storage, communications/network equipment, and other required equipment.

3.10.2) **Computer hardware resource utilization requirements** .This subclause shall specify the requirements, if any, on the system's 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 requirements (stated, for example, as percentages of the capacity of each computer hardware resource) shall include the conditions, if any, under which the resource utilization is to be measured.

3.10.3) **Computer software requirements** .This subclause shall specify the requirements, if any, regarding computer software that is required to be used by, or incorporated into, the system. Examples include operating systems, database management systems, communications/ network software, utility software, input and equipment simulators, test software, and manufacturing software. The correct nomenclature, version, and documentation references of each such item of software shall be provided.

3.10.4) **Computer communications requirements** .This subclause shall specify the additional requirements, if any, concerning the computer communications that are required to be used by, or incorporated into, the system. Examples include geographic locations to be linked; configuration and network topology; transmission techniques; data transfer rates; gateways; required system use times; type and volume of data to be transmitted/received; time boundaries for transmission/reception/response; peak volumes of data; and diagnostic features.

3.11) **System quality factors** .This subclause shall specify the requirements, if any, pertaining to system quality factors. Examples include quantitative requirements concerning system functionality (the ability to perform all required functions), reliability (the ability to perform with correct, consistent results — such as mean time between failure for equipment), maintainability (the ability to be easily serviced, repaired, or corrected), availability (the ability to be accessed and operated when needed), flexibility (the ability to be easily adapted to changing requirements), portability of software (the ability to be easily modified for a new environment), reusability (the ability to be used in multiple applications), testability (the ability to be easily and thoroughly tested), usability (the ability to be easily learned and used), and other attributes.

3.12) **Design and construction constraints** .This subclause shall specify the requirements, if any, that constrain the design and construction of the system. For hardware-software systems, this subclause shall include the physical requirements imposed on the system. These requirements may be specified by reference to appropriate commercial or government standards and specifications. Examples include requirements concerning:

a) Use of a particular system architecture or requirements on the architecture, such as required subsystems; use of standard, military, or existing components; or use of acquirer-furnished property (equipment, information, or software)

b) Use of particular design or construction standards; use of particular data standards; use of a particular programming language; workmanship requirements and production techniques

c) Physical characteristics of the system (such as weight limits, dimensional limits, color, protective coatings); interchangeability of parts; ability to be transported from one location to another; ability to be carried or set up by one, or a given number of, persons
d) Materials that can and cannot be used; requirements on the handling of toxic materials; limits on the electromagnetic radiation that the system is permitted to generate
e) Use of nameplates, part marking, serial and lot number marking, and other identifying markings
f) Flexibility and expandability that are required to be provided to support anticipated areas of growth or changes in technology, threat, or mission

3.13) **Personnel-related requirements** .This subclause shall specify the system requirements, if any, included to accommodate the number, skill levels, duty cycles, training needs, or other information about the personnel who will use or support the system. Examples include requirements for the number of work stations to be provided and for built-in help and training features. Also included shall be the human factors engineering requirements, if any, imposed on the system. These requirements shall include, as applicable, considerations for the capabilities and limitations of humans, foreseeable human errors under both normal and extreme conditions, and specific areas where the effects of human error would be particularly serious. Examples include requirements for adjustable-height work stations, color and duration of error messages, physical placement of critical indicators or buttons, and use of auditory signals.

3.14) **Training-related requirements** .This subclause shall specify the system requirements, if any, pertaining to training. Examples include training devices and training materials to be included in the system.

3.15) **Logistics-related requirements** .This subclause shall specify the system requirements, if any, concerned with logistics considerations. These considerations may include: system maintenance, software maintenance, system transportation modes, supply-system requirements, impact on existing facilities, and impact on existing equipment.

3.16) **Other requirements** .This subclause shall specify additional system requirements, if any, not covered in the previous subclauses. Examples include requirements for system documentation, such as specifications, drawings, technical manuals, test plans and procedures, and installation instruction data, if not covered in other contractual documents.

3.17) **Packaging requirements** .This clause shall specify the requirements, if any, for packaging, labeling, and handling the system and its components for delivery. Applicable commercial or government specifications and standards may be referenced if appropriate.

3.18) **Precedence and criticality of requirements** .This subclause shall specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements in this specification. Examples include identifying those requirements deemed critical to safety, to security, or to privacy protection for purposes of singling them out for special treatment. If all requirements have equal weight, this subclause shall so state.

4) **Qualification provisions** .This clause shall define a set of qualification methods and shall specify for each requirement in clause 3 the method(s) to be used to ensure that the requirement has been met. A table may be used to present this information, or each requirement in clause 3 may be annotated with the method(s) to be used. Qualification methods may include:

a) Demonstration: The operation of the system, or a part of the system, that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
b) Test: The operation of the system, or a part of the system, using instrumentation or other special test equipment to collect data for later analysis.
c) Analysis: The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpolation, or extrapolation of test results.
d) Inspection: The visual examination of system components, documentation, etc.
e) Special qualification methods: Any special qualification methods for the system, such as special tools, techniques, procedures, facilities, acceptance limits, use of standard samples, preprocessing or periodic production samples, pilot models, or pilot lots.

5) **Requirements traceability** .For system-level specifications, this subclause does not apply. For subsystem-level specifications, this subclause shall contain:
   a) Traceability from each subsystem requirement in this specification to the system requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in clause 3.)Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates two subsystems may result in requirements about how the subsystems will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.
   b) Traceability from each system requirement that has been allocated to the subsystem covered by this specification to the subsystem requirements that address it. All system requirements allocated to the subsystem shall be accounted for. Those that trace to subsystem requirements contained in IRSs shall reference those IRSs.

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

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

F.2.3 Contents of the Interface Requirements Specification (IRS)

INTERFACE REQUIREMENTS SPECIFICATION (IRS)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Requirements
   3.1) Interface identification and diagrams
   3.2) (Project-unique identifier of interface)
   3.3) Precedence and criticality of requirements
4) Qualification provisions
5) Requirements traceability
6) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the interfacing entities (systems, subsystems, hardware items, software items, manual operations, or other system components) and the interfaces to which this document applies, including, as applicable, identification number(s), title(s), abbreviations), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system(s) and software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

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

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Requirements** .This clause should be divided into the following subclauses to specify the requirements imposed on one or more systems, subsystems, hardware items, software items, manual operations, or other system components to achieve one or more interfaces among these entities. Each requirement shall be assigned a project-unique identifier to support testing and traceability and shall be stated in such a way that an objective test can be defined for it. Each requirement shall be annotated with associated qualification method(s) (see clause 4) and traceability to system (or subsystem, if applicable) requirements (see 5.a) if not provided in those clauses. The degree of detail to be provided shall be guided by the following rule: include those characteristics of the interfacing entities that are conditions for their acceptance; defer to design descriptions those characteristics that the acquirer is willing to leave up to the developer. If a given requirement fits into more than one subclause, it may be stated once and referenced from the other subclauses. If an interfacing entity included in this specification will operate in states and/or modes having interface requirements different from other states and modes, each requirement or group of requirements for that entity shall be correlated to the states and modes. The correlation may be indicated by a table or other method in this subclause, in an annex referenced from this subclause, or by annotation of the requirements in the subclauses where they appear. Conventions needed to understand the requirements shall be presented or referenced.

3.1) **Interface identification and diagrams** .For each interface identified in 1.1, this subclause shall include a project-unique identifier and shall designate the interfacing entities (systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided to depict the interfaces.

3.x) **(Project-unique identifier of interface)** .This subclause (beginning with 3.2) shall identify an interface by project-unique identifier, shall briefly identify the interfacing entities, and should be divided as needed to state the requirements imposed on one or more of the interfacing entities to achieve the interface. If the interface characteristics of an entity are not covered by this IRS but need to be mentioned to specify the requirements for entities that are, those characteristics shall be stated as assumptions or as "When [the entity not covered] does this, the [entity being specified] shall ...," rather than as requirements on the entities not covered by this IRS. This subclause may reference other documents (such as data dictionaries, standards for communication protocols, and standards for user interfaces) in place of stating the information here. The requirements shall include the following, as applicable, presented in any order suited to the requirements, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority that the interfacing entity(ies) is required to assign the interface

b) Requirements on the type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented

c) Required characteristics of individual data elements that the interfacing entity(ies) is required to provide, store, send, access, receive, etc., such as:

1) Names/identifiers

a) Project-unique identifier

b) Non-technical (natural language) name

c) Standard data element name

d) Technical name (e.g., variable or field name in code or database)

e) Abbreviation or synonymous names

2) Data type (alphanumeric, integer, etc.)

3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0–99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)

d) Required characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) is required to provide, store, send, access, receive, etc., such as:

1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)

e) Required characteristics of communication methods that the interfacing entity(ies) is required to use for the interface, such as:

1) Project-unique identifier(s)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting
4) Flow control (such as sequence numbering and buffer allocation)
5) Data transfer rate, whether periodic/aperiodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing

f) Required characteristics of protocols the interfacing entity(ies) is required to use for the interface, such as:

1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features

g) Other required characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

3.y) **Precedence and criticality of requirements** .This subclause shall be numbered as the last subclause in clause 3 and shall specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements in this specification. Examples include identifying those requirements deemed critical to safety, to security, or to privacy protection for purposes of singling them out for special treatment. If all requirements have equal weight, this subclause shall so state.

4) **Qualification provisions** .This clause shall define a set of qualification methods and shall specify, for each requirement in clause 3, the qualification method(s) to be used to ensure that the require-

ment has been met. A table may be used to present this information, or each requirement in clause 3 may be annotated with the method(s) to be used. Qualification methods may include:

a) Demonstration: The operation of interfacing entities that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
b) Test: The operation of interfacing entities using instrumentation or special test equipment to collect data for later analysis.
c) Analysis: The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpretation, or extrapolation of test results.
d) Inspection: The visual examination of interfacing entities, documentation, etc.
e) Special qualification methods: Any special qualification methods for the interfacing entities, such as special tools, techniques, procedures, facilities, and acceptance limits.

5) **Requirements traceability** .For system-level interfacing entities, this subclause does not apply. For each subsystem- or lower-level interfacing entity covered by this IRS, this subclause shall contain:
   a) Traceability from each requirement imposed on the entity in this specification to the system (or subsystem, if applicable) requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in clause 3.)Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates multiple software items may result in requirements about how the software items will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.
   b) Traceability from each system (or subsystem, if applicable) requirement that has been allocated to the interfacing entity and that affects an interface covered in this specification to the requirements in this specification that address it.
6) **Notes** .This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

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

F.2.4 Contents of the Software Requirements Specification (SRS)

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Requirements
   3.1) Required states and modes
   3.2) Software item capability requirements
      3.2.x)Software item capability)
   3.3) Software item external interface requirements
      3.3.1)Interface identification and diagrams
      3.3.x)Project-unique identifier of interface)
   3.4) Software item internal interface requirements

3.5) Software item internal data requirements
3.6) Adaptation requirements
3.7) Safety requirements
3.8) Security and privacy protection requirements
3.9) Software item environment requirements
3.10)Computer resource requirements
3.10.1)Computer hardware requirements
3.10.2)Computer hardware resource utilization requirements
3.10.3)Computer software requirements
3.10.4)Computer communications requirements
3.11)Software quality factors
3.12)Design and implementation constraints
3.13)Personnel-related requirements
3.14)Training-related requirements
3.15)Logistics-related requirements
3.16)Other requirements
3.17)Packaging requirements
3.18)Precedence and criticality of requirements

4) Qualification provisions
5) Requirements traceability
6) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviations), version number(s), and release number(s).
1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Requirements** .This clause should be divided into the following subclauses to specify the software item requirements, that is, those characteristics of the software item that are conditions for its acceptance. Software item requirements are software requirements generated to satisfy the system requirements allocated to this software item. Each requirement shall be assigned a project-unique identifier to support testing and traceability and shall be stated in such a way that an objective test can be defined for it. Each requirement shall be annotated with associated qualification method(s) (see clause 4) and traceability to system (or subsystem, if applicable) requirements (see 5.a) if not provided in those clauses. The degree of detail to be provided shall be guided by the following rule: include those characteristics of the software item that are conditions for software item acceptance; defer to design descriptions those characteristics that the acquirer is willing to leave up to the developer. If there are no requirements in a given subclause, the subclause shall so state. If a given requirement fits into more than one subclause, it may be stated once and referenced from the other subclauses. Conventions needed to understand the requirements shall be presented or referenced.
3.1) **Required states and modes** .If the software item is required to operate in more than one state or mode having requirements distinct from other states or modes, this subclause shall identify and define each state and mode. Examples of states and modes include: idle, ready, active, post-use analysis, training, degraded, emergency, backup, wartime, peacetime. The distinction between states and modes is arbitrary. A software item may be described in terms of states

only, modes only, states within modes, modes within states, or any other scheme that is useful. If no states or modes are required, this subclause shall so state, without the need to create artificial distinctions. If states and/or modes are required, each requirement or group of requirements in this specification shall be correlated to the states and modes. The correlation may be indicated by a table or other method in this subclause, in an annex referenced from this subclause, or by annotation of the requirements in the subclauses where they appear.

3.2) **Software item capability requirements** .This subclause should be divided into subclauses to itemize the requirements associated with each capability of the software item. A "capability" is defined as a group of related requirements. The word "capability" may be replaced with "function," "subject," "object," or other term useful for presenting the requirements.

3.2.x)**(Software item capability)** .This subclause shall identify a required software item capability and shall itemize the requirements associated with the capability. If the capability can be more clearly specified by dividing it into constituent capabilities, the constituent capabilities shall be specified in subclauses. The requirements shall specify required behavior of the software item and shall include applicable parameters, such as response times, throughput times, other timing constraints, sequencing, accuracy, capacities (how much/how many), priorities, continuous operation requirements, and allowable deviations based on operating conditions. The requirements shall include, as applicable, required behavior under unexpected, unallowed, or "out of bounds" conditions, requirements for error handling, and any provisions to be incorporated into the software item to provide continuity of operations in the event of emergencies. Subclause 3.3.x of the SRS provides a list of topics to be considered when specifying requirements regarding input the software item is required to accept and output it is required to produce.

3.3) **Software item external interface requirements** .This subclause should be divided into subclauses to specify the requirements, if any, for the software item's external interfaces. This subclause may reference one or more Interface Requirements Specifications (IRSs) or other documents containing these requirements.

3.3.1)**Interface identification and diagrams** .This subclause shall identify the required external interfaces of the software item (that is, relationships with other entities that involve sharing, providing or exchanging data). The identification of each interface shall include a project-unique identifier and shall designate the interfacing entities (systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided to depict the interfaces.

3.3.x)**(Project-unique identifier of interface)** .This subclause (beginning with 3.3.2) shall identify a software item external interface by project-unique identifier, shall briefly identify the interfacing entities, and should be divided into subclauses as needed to state the requirements imposed on the software item to achieve the interface. Interface characteristics of the other entities involved in the interface shall be stated as assumptions or as "When [the entity not covered] does this, the software item shall...," not as requirements on the other entities. This subclause may reference other documents (such as data dictionaries, standards for communication protocols, and standards for user interfaces) in place of stating the information here. The requirements shall include the following, as applicable, presented in any order suited to the requirements, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority that the software item is required to assign the interface
b) Requirements on the type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Required characteristics of individual data elements that the software item is required to provide, store, send, access, receive, etc., such as:

1) Names/identifiers

a) Project-unique identifier
b) Non-technical (natural language) name
c) Standard data element name
d) Technical name (e.g., variable or field name in code or database)
e) Abbreviation or synonymous names
2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0–99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)
d) Required characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the software item is required to provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)
e) Required characteristics of communication methods that the software item is required to use for the interface, such as:
1) Project-unique identifiers)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting
4) Flow control (such as sequence numbering and buffer allocation)
5) Data transfer rate, whether periodic/aperiodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing
f) Required characteristics of protocols the software item is required to use for the interface, such as:
1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features
g) Other required characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

3.4) **Software item internal interface requirements** .This subclause shall specify the requirements, if any, imposed on interfaces internal to the software item. If all internal interfaces are

left to the design, this fact shall be so stated. If such requirements are to be imposed, subclause 3.3 of the SRS provides a list of topics to be considered.

3.5) **Software item internal data requirements** .This subclause shall specify the requirements, if any, imposed on data internal to the software item. Included shall be requirements, if any, on databases and data files to be included in the software item. If all decisions about internal data are left to the design, this fact shall be so stated. If such requirements are to be imposed, subclauses 3.3.x.c and 3.3.x.d of the SRS provide a list of topics to be considered.

3.6) **Adaptation requirements** .This subclause shall specify the requirements, if any, concerning installation-dependent data to be provided by the software item (such as site-dependent latitude and longitude or site-dependent state tax codes) and parameters that the software item is required to use that may vary according to operational needs (such as operation-dependent targeting constants or site-dependent month-end dates).

3.7) **Safety requirements** .This subclause shall specify the software item requirements, if any, concerned with preventing or minimizing unintended hazards to personnel, property, and the physical environment. Examples include safeguards the software item is required to provide to prevent inadvertent actions (such as accidentally issuing an "auto pilot off" command) and non-actions (such as failure to issue an intended "auto pilot off" command). This subclause shall include the software item requirements, if any, regarding nuclear components of the system, including, as applicable, prevention of inadvertent detonation and compliance with nuclear safety rules.

3.8) **Security and privacy protection requirements** .This subclause shall specify the software item requirements, if any, concerned with maintaining security and privacy protection. These requirements shall include, as applicable, the security/privacy protection environment in which the software item is required to operate, the type and degree of security or privacy protection to be provided, the security/privacy protection risks the software item is required to withstand, required safeguards to reduce those risks, the security/privacy protection policy that is required to be met, the security/privacy protection accountability the software item is required to provide, and the criteria that are required to be met for security/privacy protection certification/ accreditation.

3.9) **Software item environment requirements** .This subclause shall specify the requirements, if any, regarding the environment in which the software item is required to operate. Examples include the computer hardware and operating system on which the software item is required to run. (Additional requirements concerning computer resources are given in the next subclause.)

3.10) **Computer resource requirements** .This subclause should be divided into the following.

3.10.1) **Computer hardware requirements** .This subclause shall specify the requirements, if any, regarding computer hardware that is required to be used by the software item. The requirements shall include, as applicable, number of each type of equipment, type, size, capacity, and other required characteristics of processors, memory, input/output devices, auxiliary storage, communications/network equipment, and other required equipment.

3.10.2) **Computer hardware resource utilization requirements** .This subclause shall specify the requirements, if any, on the software item's 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 requirements (stated, for example, as percentages of the capacity of each computer hardware resource) shall include the conditions, if any, under which the resource utilization is to be measured.

3.10.3) **Computer software requirements** .This subclause shall specify the requirements, if any, regarding computer software that is required to be used by, or incorporated into, the software item. Examples include operating systems, database management systems, communications/ network software, utility software, input and equipment simulators, test software, and manufacturing software. The correct nomenclature, version, and documentation references of each such item of software shall be provided.

3.10.4) **Computer communications requirements** .This subclause shall specify the additional requirements, if any, concerning the computer communications that are required to be

used by the software item. Examples include geographic locations to be linked; configuration and network topology; transmission techniques; data transfer rates; gateways; required system use times; type and volume of data to be transmitted/received; time boundaries for transmission/reception/response; peak volumes of data; and diagnostic features.

3.11) **Software quality factors** .This subclause shall specify the software item requirements, if any, concerned with software quality factors identified in the contract or derived from a higher level specification. Examples include quantitative requirements regarding software item functionality (the ability to perform all required functions), reliability (the ability to perform with correct, consistent results), maintainability (the ability to be easily corrected), availability (the ability to be accessed and operated when needed), flexibility (the ability to be easily adapted to changing requirements), portability (the ability to be easily modified for a new environment), reusability (the ability to be used in multiple applications), testability (the ability to be easily and thoroughly tested), usability (the ability to be easily learned and used), and other attributes.

3.12) **Design and implementation constraints** .This subclause shall specify the requirements, if any, that constrain the design and implementation of the software item. These requirements may be specified by reference to appropriate commercial or government standards and specifications. Examples include requirements concerning:

a) Use of a particular software item architecture or requirements on the architecture, such as required databases or other software units; use of standard, military, or existing components; or use of acquirer-furnished property (equipment, information, or software)

b) Use of particular design or implementation standards; use of particular data standards; use of a particular programming language

c) Flexibility and expandability that are required to be provided to support anticipated areas of growth or changes in technology, threat, or mission

3.13) **Personnel-related requirements** .This subclause shall specify the software item requirements, if any, included to accommodate the number, skill levels, duty cycles, training needs, or other information about the personnel who will use or maintain the software item. Examples include requirements for number of simultaneous users and for built-in help or training features. Also included shall be the human factors engineering requirements, if any, imposed on the software item. These requirements shall include, as applicable, considerations for the capabilities and limitations of humans; foreseeable human errors under both normal and extreme conditions; and specific areas where the effects of human error would be particularly serious. Examples include requirements for color and duration of error messages, physical placement of critical indicators or keys, and use of auditory signals.

3.14) **Training-related requirements** .This subclause shall specify the software item requirements, if any, pertaining to training. Examples include training software to be included in the software item.

3.15) **Logistics-related requirements** .This subclause shall specify the software item requirements, if any, concerned with logistics considerations. These considerations may include: system maintenance, software maintenance, system transportation modes, supply-system requirements, impact on existing facilities, and impact on existing equipment.

3.16) **Other requirements** .This subclause shall specify additional software item requirements, if any, not covered in the previous subclauses.

3.17) **Packaging requirements** .This clause shall specify the requirements, if any, for packaging, labeling, and handling the software item for delivery (for example, delivery on 8 track magnetic tape labelled and packaged in a certain way). Applicable commercial or government specifications and standards may be referenced if appropriate.

3.18) **Precedence and criticality of requirements** .This subclause shall specify, if applicable, the order of precedence, criticality, or assigned weights indicating the relative importance of the requirements in this specification. Examples include identifying those requirements deemed critical to safety, to security, or to privacy protection for purposes of singling them out for special treatment. If all requirements have equal weight, this subclause shall so state.

4) **Qualification provisions** .This clause shall define a set of qualification methods and shall specify for each requirement in clause 3 the method(s) to be used to ensure that the requirement has been met. A table may be used to present this information, or each requirement in clause 3 may be annotated with the method(s) to be used. Qualification methods may include:
   a) Demonstration: The operation of the software item, or a part of the software item, that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
   b) Test: The operation of the software item, or a part of the software item, using instrumentation or other special test equipment to collect data for later analysis.
   c) Analysis: The processing of accumulated data obtained from other qualification methods. Examples are reduction, interpolation, or extrapolation of test results.
   d) Inspection: The visual examination of software item code, documentation, etc.
   e) Special qualification methods: Any special qualification methods for the software item, such as special tools, techniques, procedures, facilities, and acceptance limits.
5) **Requirements traceability** .This subclause shall contain:
   a) Traceability from each software item requirement in this specification to the system (or subsystem, if applicable) requirements it addresses. (Alternatively, this traceability may be provided by annotating each requirement in clause 3.)Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates multiple software items may result in requirements about how the software items will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as “system implementation” or to the system design decisions that resulted in their generation.
   b) Traceability from each system (or subsystem, if applicable) requirement allocated to this software item to the software item requirements that address it. All system (subsystem) requirements allocated to this software item shall be accounted for. Those that trace to software item requirements contained in IRSs shall reference those IRSs.
6) **Notes** .This clause shall contain any general information that aids in understanding this specification (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

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

Annex G

(normative)

Design software product descriptions

G.1 Scope

This annex specifies the contents of software products associated with design in this standard. It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

G.2 Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in the order they are referenced in this standard and include the following:

a)) System/Subsystem Design Description (SSDD)
b)) Interface Design Description (IDD)
c)) Database Design Description (DBDD)
d)) Software Design Description (SDD)

G.2.1 Contents of the System/Subsystem Design Description (SSDD)

SYSTEM/SUBSYSTEM DESIGN DESCRIPTION (SSDD)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) System-wide design decisions
4) System architectural design
   4.1) System components
   4.2) Concept of execution
   4.3) Interface design
      4.3.1)Interface identification and diagrams
      4.3.x)(Project-unique identifier of interface)
5) Requirements traceability
6) Notes

A) Annexes

1) **Scope** . This clause should be divided into the following subclauses.

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

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

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

2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **System-wide design decisions** . This clause should be divided into subclauses as needed to present system-wide design decisions, that is, decisions about the system's behavioral design (how it will behave, from a user's point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting the selection and design of system components. If all such decisions are explicit in the requirements or are deferred to the design of the system components, this clause shall so state. Design decisions that respond to requirements designated critical, such as those for safety, security, or privacy protection, shall be placed in separate subclauses. If a design decision depends upon system states or modes, this dependency shall be indicated. Design conventions needed to understand the design shall be presented or referenced. Examples of system-wide design decisions are the following:

a) Design decisions regarding input the system will accept and output it will produce, including interfaces with other systems, hardware items, software items, and users (4.3.x of the SSDD identifies topics to be considered in this description). If part or all of this information is given in Interface Design Descriptions (IDDs), they may be referenced.

b) Design decisions on system behavior in response to each input or condition, including actions the system will perform, response times and other performance characteristics, description of physical systems modeled, selected equations/algorithms/rules, and handling of unallowed inputs or conditions.

c) Design decisions on how system databases/data files will appear to the user (4.3.x of the SSDD identifies topics to be considered in this description). If part or all of this information is given in Database Design Descriptions (DBDDs), they may be referenced.

d) Selected approach to meeting safety, security, and privacy protection requirements.

e) Design and construction choices for hardware or hardware-software systems, such as physical size, color, shape, weight, materials, and markings.

f) Other system-wide design decisions made in response to requirements, such as selected approach to providing required flexibility, availability, and maintainability.

4) **System architectural design** . This clause should be divided into the following subclauses to describe the system architectural design. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one subclause, it may be presented once and referenced from the other subclauses. Design conventions needed to understand the design shall be presented or referenced.Note: For brevity, this clause is written in terms of organizing a system directly into hardware items, software items, and manual operations, but is interpreted to cover organizing a system into subsystems, organizing a subsystem into hardware items, software items, and manual operations, or other variations as appropriate.

4.1) **System components** . This subclause shall:

a) Identify the components of the system (hardware items, software items, and manual operations). Each component shall be assigned a project-unique identifier. Note: A database may be treated as a software item or as part of a software item.

b) Show the static (such as "consists of") relationship(s) of the components. Multiple relationships may be presented, depending on the selected design methodology.

c) State the purpose of each component and identify the system requirements and system-wide design decisions allocated to it. (Alternatively, the allocation of requirements may be provided in 5.a.)

d) Identify each component's development type, if known (such as new development, existing component to be reused as is, existing design to be reused as is, existing design or component to be reengineered, component to be developed for reuse, component planned for Build N, etc.) For existing design or components, the description shall provide identifying information, such as name, version, documentation references, location, etc.

e) For each computer system or other aggregate of computer hardware resources identified for use in the system, describe its computer hardware resources (such as processors, memory, input/output devices, auxiliary storage, and communications/network equipment). Each description shall, as applicable, identify the hardware and software items that will use the resource, describe the allocation of resource utilization to each software item that will use the resource (for example, 20% of the resource's capacity allocated to software item 1, 30% to software item 2), describe the conditions under which utilization will be measured, and describe the characteristics of the resource:

1) Descriptions of computer processors shall include, as applicable, manufacturer name and model number, processor speed/capacity, identification of instruction set architecture, applicable compiler(s), word size (number of bits in each computer word), character set standard (such as ASCII, EBCDIC), and interrupt capabilities.

2) Descriptions of memory shall include, as applicable, manufacturer name and model number and memory size, type, speed, and configuration (such as 256K cache memory, 16MB RAM (4MB x 4)).

3) Descriptions of input/output devices shall include, as applicable, manufacturer name and model number, type of device, and device speed/capacity.

4) Descriptions of auxiliary storage shall include, as applicable, manufacturer name and model number, type of storage, amount of installed storage, and storage speed.

5) Descriptions of communications/network equipment, such as modems, network interface cards, hubs, gateways, cabling, high speed data lines, or aggregates of these or other components, shall include, as applicable, manufacturer name and model number, data transfer rates/capacities, network topologies, transmission techniques, and protocols used.

6) Each description shall also include, as applicable, growth capabilities, diagnostic capabilities, and any additional hardware capabilities relevant to the description.

f) Present a specification tree for the system, that is, a diagram that identifies and shows the relationships among the planned specifications for the system components.

4.2) **Concept of execution** . This subclause shall describe the concept of execution among the system components. It shall include diagrams and descriptions showing the dynamic relationship of the components, that is, how they will interact during system operation, including, as applicable, flow of execution control, data flow, dynamically controlled sequencing, state transition diagrams, timing diagrams, priorities among components, handling of interrupts, timing/sequencing relationships, exception handling, concurrent execution, dynamic allocation/deallocation, dynamic creation/deletion of objects, processes, tasks, and other aspects of dynamic behavior.

4.3) **Interface design** . This subclause should be divided into the following to describe the interface characteristics of the system components. It shall include both interfaces among the components and their interfaces with external entities such as other systems, hardware items, software items, and users. Note: There is no requirement for these interfaces to be completely designed at this level; this subclause is provided to allow the recording of interface design decisions made as part of system architectural design. If part or all of this information is contained in Interface Design Descriptions (IDDs) or elsewhere, these sources may be referenced.

4.3.1)**Interface identification and diagrams** .This subclause shall state the project-unique identifier assigned to each interface and shall identify the interfacing entities (systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface

characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided, as appropriate, to depict the interfaces.

4.3.x)**(Project-unique identifier of interfaces)** .This subclause (beginning with 4.3.2) shall identify an interface by project-unique identifier, shall briefly identify the interfacing entities, and should be divided into subclauses as needed to describe the interface characteristics of one or both of the interfacing entities. If a given interfacing entity is not covered by this SSDD (for example, an external system) but its interface characteristics need to be mentioned to describe interfacing entities that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the entity that is covered] will ... ." This subclause may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority assigned to the interface by the interfacing entity(ies)
b) Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Standard data element name
d) Technical name (e.g., variable or field name in code or database)
e) Abbreviation or synonymous names
2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0-99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)
d) Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier to be used for traceability
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)

e) Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
1) Project-unique identifier(s)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting
4) Flow control (such as sequence numbering and buffer allocation)
5) Data transfer rate, whether periodic/aperiodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing
f) Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:
1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features
g) Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

5) **Requirements traceability** .This subclause shall contain:
a) Traceability from each system component identified in this SSDD to the system requirements allocated to it. (Alternatively, this traceability may be provided in 4.1.)Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates two subsystems may result in requirements about how the subsystems will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.
b) Traceability from each system requirement to the system components to which it is allocated.

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

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

G.2.2 Contents of the Interface Design Description (IDD)

INTERFACE DESIGN DESCRIPTION (IDD)

Contents

1) Scope
1.1) Identification
1.2) System overview
1.3) Document overview
2) Referenced documents
3) Interface design
3.1) Interface identification and diagrams

3.x) (Project-unique identifier of interface)
4) Requirements traceability
5) Notes
A) Annexes
1) **Scope** . This clause should be divided into the following subclauses.
1.1) **Identification** . This subclause shall contain a full identification of the interfacing entities (systems, subsystems, hardware items, software items, manual operations, or other system components) and interfaces to which this document applies, including, as applicable, identification number(s), title(s), abbreviations), version number(s), and release number(s).
1.2) **System overview** . This subclause shall briefly state the purpose of the system(s) and software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
1.3) **Document overview** . This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** . This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Interface design** . This clause should be divided into the following subclauses to describe the interface characteristics of one or more systems, subsystems, hardware items, software items, manual operations, or other system components. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one subclause, it may be presented once and referenced from the other subclauses. If part or all of this information is documented elsewhere, it may be referenced. Design conventions needed to understand the design shall be presented or referenced.
3.1) **Interface identification and diagrams** . For each interface identified in 1.1, this subclause shall state the project-unique identifier assigned to the interface and shall identify the interfacing entities (systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided, as appropriate, to depict the interfaces.
3.x) **(Project-unique identifier of interface)** . This subclause (beginning with 3.2) shall identify an interface by project-unique identifier, shall briefly identify the interfacing entities, and should be divided as needed to describe the interface characteristics of one or both of the interfacing entities. If a given interfacing entity is not covered by this IDD (for example, an external system) but its interface characteristics need to be mentioned to describe interfacing entities that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the entity that is covered] will... ." This subclause may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):
a) Priority assigned to the interface by the interfacing entity(ies)
b) Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name

c) Standard data element name
d) Technical name (e.g., variable or field name in code or database)
e) Abbreviation or synonymous names
2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0-99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)
d) Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)
e) Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
1) Project-unique identifier(s)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting
4) Flow control (such as sequence numbering and buffer allocation)
5) Data transfer rate, whether periodic/aperiodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing
f) Characteristics of protocols the interfacing entity(ies) will use for the interface, such as:
1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features
g) Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

4) **Requirements traceability** .This subclause shall contain:
a) Traceability from each interfacing entity covered by this IDD to the system or software item requirements addressed by the entity's interface design.
Note: Each level of system or software item refinement may result in requirements not directly traceable to higher-level requirements. For example, a system or software architectural design

that creates two subsystems or two software units may result in requirements about how the subsystems or software units will interface, even though these interfaces are not covered in system or software item requirements. Such requirements may be traced to a general requirement such as “system implementation” or “software item” or to the system or software design decisions (e.g., in the SDD) that resulted in their generation.

b) Traceability from each system or software item requirement that affects an interface covered in this IDD to the interfacing entities that address it.

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

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

G.2.3 Contents of the Database Design Description (DBDD)

DATABASE DESIGN DESCRIPTION (DBDD)

Contents

1) Scope
   1.1) Identification
   1.2) Database overview
   1.3) Document overview
2) Referenced documents
3) Database-wide design decisions
4) Design of the database
   4.x) (Name of database design level)
5) Detailed design of software units used for database access or manipulation
   5.x) (Project-unique identifier of a software unit, or designator of a group of software units)
6) Requirements traceability
7) Notes

A) Annexes

1) **Scope** . This clause should be divided into the following subclauses.

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

   1.2) **Database overview** .This subclause shall briefly state the purpose of the database to which this document applies. It shall describe the general nature of the database; summarize the history of its development, use, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

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

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Database-wide design decisions** .This clause should be divided into subclauses as needed to present database-wide design decisions, that is, decisions about the database’s behavioral design (how it will behave, from a user’s point of view, in meeting its requirements, ignoring internal

implementation) and other decisions affecting further design of the database. If all such decisions are explicit in the system or software item requirements, this clause shall so state. Design decisions that respond to requirements designated critical, such as those for safety, security, or privacy protection, shall be placed in separate subclauses. If a design decision depends upon system states or modes, this dependency shall be indicated. If some or all of the design decisions are described in the documentation of a custom or commercial database management system (DBMS), they may be referenced from this clause. Design conventions needed to understand the design shall be presented or referenced. Examples of database-wide design decisions are the following:

a) Design decisions regarding queries or other input the database will accept and output (displays, reports, messages, responses, etc.) it will produce, including interfaces with other systems, hardware items, software items, and users (5.x.d of the DBDD identifies topics to be considered in this description). If part or all of this information is given in Interface Design Descriptions (IDDs), they may be referenced.
b) Design decisions on database behavior in response to each input or query, including actions, response times and other performance characteristics, selected equations/algorithms/rules, disposition, and handling of unallowed inputs
c) Design decisions on how databases/data files will appear to the user (4.x of the DBDD identifies topics to be considered in this description)
d) Design decisions on the database management system to be used (including name, version/ release) and the type of flexibility to be built into the database for adapting to changing requirements
e) Design decisions on the levels and types of availability, security, privacy protection, and continuity of operations to be offered by the database
f) Design decisions on database distribution (such as client/server), master database file updates and maintenance, including maintaining consistency, establishing/ re-establishing and maintaining synchronization, enforcing integrity and business rules
g) Design decisions on backup and restoration including data and process distribution strategies, permissible actions during backup and restoration, and special considerations for new or non-standard technologies such as video and sound
h) Design decisions on repacking, sorting, indexing, synchronization, and consistency including automated disk management and space reclamation considerations, optimizing strategies and considerations, storage and size considerations, and population of the database and capture of legacy data

4) **Design of the database** .This clause should be divided into subclauses as needed to describe the design of the database. The number of levels of design and the names of those levels shall be based on the design methodology used. Examples of database design levels include conceptual, internal, logical, and physical. If part or all of the design depends upon system states or modes, this dependency shall be indicated. Design conventions needed to understand the design shall be presented or referenced.

Note: The term “data element assembly” is used to mean any entity, relation, schema, field, table, array, etc., that has structure (number/order/grouping of data elements) at a given design level (e.g., conceptual, internal, logical, physical) and the term “data element” to mean any relation, attribute, field, cell, data element, etc. that does not have structure at that level.

4.x) **(Name of database design level)** .This subclause shall identify a database design level and shall describe the data elements and data element assemblies of the database in the terminology of the selected design method. The information shall include the following, as applicable, presented in any order suited to the information to be provided:

a) Characteristics of individual data elements in the database design, such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Standard data element name
d) Technical name (e.g., field name in the database)
e) Abbreviation or synonymous names

2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0–99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)

b) Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) in the database design, such as:

1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)

5) **Detailed design of software units used for database access or manipulation** .This clause should be divided into the following subclauses to describe each software unit used for database access or manipulation. If part or all of this information is provided elsewhere, such as in a Software Design Description (SDD), the SDD for a customized DBMS, or the user manual of a commercial DBMS, that information may be referenced rather than repeated here. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one subclause, it may be presented once and referenced from the other subclauses. Design conventions needed to understand the design shall be presented or referenced.

5.x) **(Project-unique identifier of a software unit, or designator of a group of software units)** .This subclause shall identify a software unit by project-unique identifier and shall describe the unit. The description shall include the following information, as applicable. Alternatively, this subclause may designate a group of software units and identify and describe the software units in subordinate subclauses. Software units that contain other software units may reference the descriptions of those units rather than repeating information.

a) Unit design decisions, if any, such as algorithms to be used, if not previously selected.
b) Any constraints, limitations, or unusual features in the design of the software unit
c) The programming language to be used and rationale for its use if other than the specified language for the software item
d) If the software unit consists of or contains procedural commands (such as menu selections in a DBMS for defining forms and reports, on-line DBMS queries for database access and manipulation, input to a graphical user interface (GUI) builder for automated code generation, commands to the operating system, or shell scripts), a list of the procedural commands and a reference to user manuals or other documents that explain them
e) If the software unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements and data element assemblies, as applicable. Data local to the software unit shall be described separately from data input to or output from the software unit. Interface characteristics may be provided here or by referencing Interface Design Description(s). If a given interfacing entity is not covered by this DBDD (for example, an external system) but its interface characteristics need to be mentioned to describe soft-

ware units that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the software unit] will... ." This subclause may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

1) Project-unique identifier for the interface
2) Identification of the interfacing entities (software units, software items, hardware items, users, etc.) by name, number, version, and documentation references, as applicable
3) Priority assigned to the interface by the interfacing entity(ies)
4) Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
5) Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc. Subclause 4.x.a of the DBDD identifies topics to be covered in this description.
6) Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc. Subclause 4.x.b of the DBDD identifies topics to be covered in this description.
7) Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:

a) Project-unique identifiers)
b) Communication links/bands/frequencies/media and their characteristics
c) Message formatting
d) Flow control (such as sequence numbering and buffer allocation)
e) Data transfer rate, whether periodic/aperiodic, and interval between transfers
f) Routing, addressing, and naming conventions
g) Transmission services, including priority and grade
h) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing

8) Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:

a) Project-unique identifier(s)
b) Priority/layer of the protocol
c) Packeting, including fragmentation and reassembly, routing, and addressing
d) Legality checks, error control, and recovery procedures
e) Synchronization, including connection establishment, maintenance, termination
f) Status, identification, and any other reporting features

9) Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

f) If the software unit contains logic, the logic to be used by the software unit, including, as applicable:

1) Conditions in effect within the software unit when its execution is initiated
2) Conditions under which control is passed to other software units
3) Response and response time to each input, including data conversion, renaming, and data transfer operations
4) Sequence of operations and dynamically controlled sequencing during the software unit's operation, including:

a) The method for sequence control
b) The logic and input conditions of that method, such as timing variations, priority assignments
c) Data transfer in and out of memory

d) The sensing of discrete input signals, and timing relationships between interrupt operations within the software unit
5) Exception and error handling

6) **Requirements traceability** . This clause shall contain:
a) Traceability from each database or other software unit covered by this DBDD to the system or software item requirements it addresses.
Note: Each level of system refinement may result in requirements not directly traceable to higher-level requirements. For example, a system architectural design that creates two subsystems may result in requirements about how the subsystems will interface, even though these interfaces are not covered in system requirements. Such requirements may be traced to a general requirement such as "system implementation" or to the system design decisions that resulted in their generation.
b) Traceability from each system or software item requirement that has been allocated to a database or other software unit covered in this DBDD to the database or other software units that address it.

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

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

G.2.4 Contents of the Software Design Description (SDD)

SOFTWARE DESIGN DESCRIPTION (SDD)

Contents

1) Scope
1.1) Identification
1.2) System overview
1.3) Document overview
2) Referenced documents
3) System item-wide design decisions
4) Software item architectural design
4.1) Software item components
4.2) Concept of execution
4.3) Interface design
4.3.1)Interface identification and diagrams
4.3.x)(Project-unique identifier of interface)
5) Software item detailed design
5.x) (Project-unique identifier of a software unit, or designator of a group of software units)
6) Requirements traceability
7) Notes
A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

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

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

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Software item-wide design decisions** .This clause should be divided into subclauses as needed to present software item-wide design decisions, that is, decisions about the software item's behavioral design (how it will behave, from a user's point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting the selection and design of the software units that make up the software item. If all such decisions are explicit in the software item requirements or are deferred to the design of the software item's software units, this clause shall so state. Design decisions that respond to requirements designated critical, such as those for safety, security, or privacy protection, shall be placed in separate subclauses. If a design decision depends upon system states or modes, this dependency shall be indicated. Design conventions needed to understand the design shall be presented or referenced. Examples of software item-wide design decisions are the following:

   a) Design decisions regarding input the software item will accept and output it will produce, including interfaces with other systems, hardware items, software items, and users (4.3.x of the SDD identifies topics to be considered in this description). If part or all of this information is given in Interface Design Descriptions (IDDs), they may be referenced.
   b) Design decisions on software item behavior in response to each input or condition, including actions the software item will perform, response times and other performance characteristics, description of physical systems modeled, selected equations/algorithms/rules, and handling of unallowed inputs or conditions.
   c) Design decisions on how databases/data files will appear to the user (4.3.x of the SDD identifies topics to be considered in this description). If part or all of this information is given in Database Design Descriptions (DBDDs), they may be referenced.
   d) Selected approach to meeting safety, security, and privacy protection requirements.
   e) Other software item-wide design decisions made in response to requirements, such as selected approach to providing required flexibility, availability, and maintainability.

4) **Software item architectural design** .This clause should be divided into the following subclauses to describe the software item architectural design. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one subclause, it may be presented once and referenced from the other subclauses. Design conventions needed to understand the design shall be presented or referenced.

   4.1) **Software item components** .This subclause shall:

      a) Identify the software units that make up the software item. Each software unit shall be assigned a project-unique identifier.

         Note: A software unit is an element in the design of a software item; for example, a major subdivision of a software item, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities. A database may be treated as a software item or as a software unit. The SDD may refer to software units by any name(s) consistent with the design methodology being used.
      b) Show the static (such as "consists of") relationships) of the software units. Multiple relationships may be presented, depending on the selected software design methodology (for

example, in an object-oriented design, this subclause may present the class and object structures as well as the module and process architectures of the software item).

c) State the purpose of each software unit and identify the software item requirements and software item-wide design decisions allocated to it. (Alternatively, the allocation of requirements may be provided in 6.a.)

d) Identify each software unit's development type (such as new development, existing design or software to be reused as is, existing design or software to be reengineered, software to be developed for reuse, software planned for Build N, etc.) For existing design or software, the description shall provide identifying information, such as name, version, documentation references, library, etc.

e) Describe the software item's (and as applicable, each software unit's) planned utilization of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage capacity, and communications/network equipment capacity). The description shall cover all computer hardware resources included in resource utilization requirements for the software item, in system-level resource allocations affecting the softwareitem, and in resource utilization measurement planning in the Software Development Plan. If all utilization data for a given computer hardware resource are presented in a single location, such as in one SDD, this subclause may reference that source. Included for each computer hardware resource shall be:

1) The software item requirements or system-level resource allocations being satisfied
2) The assumptions and conditions on which the utilization data are based (for example, typical usage, worst-case usage, assumption of certain events)
3) Any special considerations affecting the utilization (such as use of virtual memory, overlays, or multiprocessors or the impacts of operating system overhead, library software, or other implementation overhead)
4) The units of measure used (such as percentage of processor capacity, cycles per second, bytes of memory, kilobytes per second)
5) The level(s) at which the estimates or measures will be made (such as software unit, software item, or executable program)

f) Identify the program library in which the software that implements each software unit is to be placed

4.2) **Concept of execution** .This subclause shall describe the concept of execution among the software units. It shall include diagrams and descriptions showing the dynamic relationship of the software units, that is, how they will interact during software item operation, including, as applicable, flow of execution control, data flow, dynamically controlled sequencing, state transition diagrams, timing diagrams, priorities among units, handling of interrupts, timing/sequencing relationships, exception handling, concurrent execution, dynamic allocation/deallocation, dynamic creation/deletion of objects, processes, tasks, and other aspects of dynamic behavior.

4.3) **Interface design** .This subclause should be divided as follows to describe the interface characteristics of the software units. It shall include both interfaces among the software units and their interfaces with external entities such as systems, hardware items, software items, and users. If part or all of this information is contained in Interface Design Descriptions (IDDs), in clause 5 of the SDD, or elsewhere, these sources may be referenced.

4.3.1)**Interface identification and diagrams** .This subclause shall state the project-unique identifier assigned to each interface and shall identify the interfacing entities (software units, systems, hardware items, software items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided, as appropriate, to depict the interfaces.

4.3.x)**(Project-unique identifier of interface)** .This subclause (beginning with 4.3.x) shall identify an interface by project-unique identifier, shall briefly identify the interfacing enti-

ties, and should be divided as needed to describe the interface characteristics of one or both of the interfacing entities. If a given interfacing entity is not covered by this SDD (for example, an external system) but its interface characteristics need to be mentioned to describe interfacing entities that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the entity that is covered] will ... ." This subclause may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

a) Priority assigned to the interface by the interfacing entity(ies)
b) Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
c) Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Standard data element name
d) Technical name (e.g., variable or field name in code or database)
e) Abbreviation or synonymous names
2) Data type (alphanumeric, integer, etc.)
3) Size and format (such as length and punctuation of a character string)
4) Units of measurement (such as meters, dollars, nanoseconds)
5) Range or enumeration of possible values (such as 0-99)
6) Accuracy (how correct) and precision (number of significant digits)
7) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
8) Security and privacy protection constraints
9) Sources (setting/sending entities) and recipients (using/receiving entities)
d) Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
1) Names/identifiers
a) Project-unique identifier
b) Non-technical (natural language) name
c) Technical name (e.g., record or data structure name in code or database)
d) Abbreviations or synonymous names
2) Data elements in the assembly and their structure (number, order, grouping)
3) Medium (such as disk) and structure of data elements/assemblies on the medium
4) Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
5) Relationships among assemblies, such as sorting/access characteristics
6) Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
7) Security and privacy protection constraints
8) Sources (setting/sending entities) and recipients (using/receiving entities)
e) Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
1) Project-unique identifier(s)
2) Communication links/bands/frequencies/media and their characteristics
3) Message formatting
4) Flow control (such as sequence numbering and buffer allocation)

5) Data transfer rate, whether periodic/aperiodic, and interval between transfers
6) Routing, addressing, and naming conventions
7) Transmission services, including priority and grade
8) Safety/security/privacy protection considerations, such as encryption, user authentication, compartmentalization, and auditing

f) Characteristics of protocols that the interfacing entity(ies) will use for the interface, such as:

1) Project-unique identifier(s)
2) Priority/layer of the protocol
3) Packeting, including fragmentation and reassembly, routing, and addressing
4) Legality checks, error control, and recovery procedures
5) Synchronization, including connection establishment, maintenance, termination
6) Status, identification, and any other reporting features

g) Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

5) **Software item detailed design** .This clause should be divided into the following subclauses to describe each software unit of the software item. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one subclause, it may be presented once and referenced from the other subclauses. Design conventions needed to understand the design shall be presented or referenced. Interface characteristics of software units may be described here, in clause 4, or in Interface Design Descriptions (IDDs). Software units that are databases, or that are used to access or manipulate databases, may be described here or in Database Design Descriptions (DBDDs).

5.x) **(Project-unique identifier of a software unit, or designator of a group of software units)** .This subclause shall identify a software unit by project-unique identifier and shall describe the unit. The description shall include the following information, as applicable. Alternatively, this subclause may designate a group of software units and identify and describe the software units in subordinate subclauses. Software units that contain other software units may reference the descriptions of those units rather than repeating information.

a) Unit design decisions, if any, such as algorithms to be used, if not previously selected
b) Any constraints, limitations, or unusual features in the design of the software unit
c) The programming language to be used and rationale for its use if other than the specified software item language
d) If the software unit consists of or contains procedural commands (such as menu selections in a database management system (DBMS) for defining forms and reports, on-line DBMS queries for database access and manipulation, input to a graphical user interface (GUI) builder for automated code generation, commands to the operating system, or shell scripts), a list of the procedural commands and reference to user manuals or other documents that explain them
e) If the software unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements and data element assemblies, as applicable. Subclause 4.3.x of the SDD provides a list of topics to be covered, as applicable. Data local to the software unit may be described separately from data input to or output from the software unit. If the software unit is a database, a corresponding Database Design Description (DBDD) may be referenced; interface characteristics may be provided here or by referencing clause 4 or the corresponding Interface Design Description(s).
f) If the software unit contains logic, the logic to be used by the software unit, including, as applicable:

1) Conditions in effect within the software unit when its execution is initiated
2) Conditions under which control is passed to other software units
3) Response and response time to each input, including data conversion, renaming, and data transfer operations
4) Sequence of operations and dynamically controlled sequencing during the software unit's operation, including:

a) The method for sequence control
b) The logic and input conditions of that method, such as timing variations, priority assignments
c) Data transfer in and out of memory
d) The sensing of discrete input signals, and timing relationships between interrupt operations within the software unit
5) Exception and error handling

6) **Requirements traceability** . This clause shall contain:
a) Traceability from each software unit identified in this SDD to the software item requirements allocated to it. (Alternatively, this traceability may be provided in 4.1.)
Note: Each level of software item refinement may result in requirements not directly traceable to higher-level requirements. For example, a software architectural design that creates two software units may result in requirements about how the software units will interface, even though these interfaces are not covered in software item requirements. Such requirements may be traced to a general requirement such as “software item” or to the software design decisions that resulted in their generation.
b) Traceability from each software item requirement to the software units to which it is allocated.

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

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

Annex H

(normative)

Annex H Qualification testing software product descriptions

H.1)Scope

This annex specifies the contents of software products associated with qualification testing in this standard (the Software Test Plan is found together with other plans in annex E). It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

H.2)Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in the order they are referenced in this standard and include the following:

a)) Software Test Description (STD)
b)) Software Test Report (STR)

H.2.1 Contents of the Software Test Description (STD)

SOFTWARE TEST DESCRIPTION (STD)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Test preparations
   3.x) (project-unique identifier of a test)
      3.x.1)Hardware preparation
      3.x.2)Software preparation
      3.x.3)Other pre-test preparations
4) Test descriptions
   4.x) (Project-unique identifier of a test)
      4.x.y)(Project-unique identifier of a test case)
      4.x.y.1)Requirements addressed
      4.x.y.2)Prerequisite conditions
      4.x.y.3)Test input
      4.x.y.4)Expected test results
      4.x.y.5)Criteria for evaluating results
      4.x.y.6)Test procedure
      4.x.y.7)Assumptions and constraints

5) Requirements traceability
6) Notes
A) Annexes
1) **Scope** . This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Test preparations** . This clause should be divided into the following subclauses. Safety precautions, marked by WARNING or CAUTION, and security and privacy protection considerations shall be included as applicable.
   3.x) **(Project-unique identifier of a test)** . This subclause shall identify a test by project-unique identifier, shall provide a brief description, and should be divided into the following. When the information required duplicates information previously specified for another test, that information may be referenced rather than repeated.
      3.x.1)**Hardware preparation** . This subclause shall describe the procedures necessary to prepare the hardware for the test. Reference may be made to published operating manuals for these procedures. The following shall be provided, as applicable:
      a) The specific hardware to be used, identified by name and, if applicable, number
      b) Any switch settings and cabling necessary to connect the hardware
      c) One or more diagrams to show hardware, interconnecting control, and data paths
      d) Step-by-step instructions for placing the hardware in a state of readiness
      3.x.2)**Software preparation** . This subclause shall describe the procedures necessary to prepare the item(s) under test and any related software, including data, for the test. Reference may be made to published software manuals for these procedures. The following information shall be provided, as applicable:
      a) The specific software to be used in the test
      b) The storage medium of the item(s) under test (e.g., magnetic tape, diskette)
      c) The storage medium of any related software (e.g., simulators, test drivers, databases)
      d) Instructions for loading the software, including required sequence
      e) Instructions for software initialization common to more than one test case
      3.x.3)**Other pre-test preparations** .This subclause shall describe any other pre-test personnel actions, preparations, or procedures necessary to perform the test.
4) **Test descriptions** . This clause should be divided into the following subclauses. Safety precautions, marked by WARNING or CAUTION, and security and privacy protection considerations shall be included as applicable.
   4.x) **(Project-unique identifier of a test)** .This subclause shall identify a test by project-unique identifier and should be divided into the following. When the required information duplicates information previously provided, that information may be referenced rather than repeated.
      4.x.y)**(Project-unique identifier of a test case)** .This subclause shall identify a test case by project-unique identifier, state its purpose, and provide a brief description. The following shall provide a detailed description of the test case.
      4.x.y.1)**Requirements addressed** .This subclause shall identify the software item or system requirements addressed by the test case. (Alternatively, this information may be provided in 5.a.)

4.x.y.2)**Prerequisite conditions** . This subclause shall identify any prerequisite conditions that are required to be established prior to performing the test case. The following considerations shall be discussed, as applicable:

a) Hardware and software configuration
b) Flags, initial breakpoints, pointers, control parameters, or initial data to be set/reset prior to test commencement
c) Preset hardware conditions or electrical states necessary to run the test case
d) Initial conditions to be used in making timing measurements
e) Conditioning of the simulated environment
f) Other special conditions peculiar to the test case

4.x.y.3)**Test input** . This subclause shall describe the test input necessary for the test case. The following shall be provided, as applicable:

a) Name, purpose, and description (e.g., range of values, accuracy) of each test input
b) Source of the test input and the method to be used for selecting the test input
c) Whether the test input is real or simulated
d) Time or event sequence of test input
e) The manner in which the input data will be controlled to:
1) Test the item(s) with a minimum/reasonable number of data types and values
2) Exercise the item(s) with a range of valid data types and values that test for overload, saturation, and other "worst case" effects
3) Exercise the item(s) with invalid data types and values to test for appropriate handling of irregular inputs
4) Permit retesting, if necessary

4.x.y.4)**Expected test results** .This subclause shall identify all expected test results for the test case. Both intermediate and final test results shall be provided, as applicable.

4.x.y.5)**Criteria for evaluating results** .This subclause shall identify the criteria to be used for evaluating the intermediate and final results of the test case. For each test result, the following information shall be provided, as applicable:

a) The range or accuracy over which an output can vary and still be acceptable
b) Minimum number of combinations or alternatives of input and output conditions that constitute an acceptable test result
c) Maximum/minimum allowable test duration, in terms of time or number of events
d) Maximum number of interrupts, halts, or other system breaks that may occur
e) Allowable severity of processing errors
f) Conditions under which the result is inconclusive and re-testing is to be performed
g) Conditions under which the output is to be interpreted as indicating irregularities in input test data, in the test database/data files, or in test procedures
h) Allowable indications of the control, status, and results of the test and the readiness for the next test case (may be output of auxiliary test software)
i) Additional criteria not mentioned above.

4.x.y.6)**Test procedure** .This subclause shall define the test procedure for the test case. The test procedure shall be defined as a series of individually numbered steps listed sequentially in the order in which the steps are to be performed. For convenience in document maintenance, the test procedures may be included as an annex and referenced in this subclause. The appropriate level of detail in each test procedure depends on the type of software being tested. For some software, each keystroke may be a separate test procedure step; for most software, each step may include a logically related series of keystrokes or other actions. The appropriate level of detail is the level at which it is useful to specify expected results and compare them to actual results. The following shall be provided for each test procedure, as applicable:

a) Test operator actions and equipment operation required for each step, including commands, as applicable, to:
1) Initiate the test case and apply test input
2) Inspect test conditions

3) Perform interim evaluations of test results
4) Record data
5) Halt or interrupt the test case
6) Request data dumps or other aids, if needed
7) Modify the database/data files
8) Repeat the test case if unsuccessful
9) Apply alternate modes as required by the test case
10) Terminate the test case

b) Expected result and evaluation criteria for each step

c) If the test case addresses multiple requirements, identification of which test procedure step(s) address which requirements. (Alternatively, this information may be provided in 5.)

d) Actions to follow in the event of a program stop or indicated error, such as:

1) Recording of critical data from indicators for reference purposes
2) Halting or pausing time-sensitive test-support software and test apparatus
3) Collection of system and operator records of test results

e) Procedures to be used to reduce and analyze test results to accomplish the following, as applicable:

1) Detect whether an output has been produced
2) Identify media and location of data produced by the test case
3) Evaluate output as a basis for continuation of test sequence
4) Evaluate test output against required output

4.x.y.7) **Assumptions and constraints** .This subclause shall identify any assumptions made and constraints or limitations imposed in the description of the test case due to system or test conditions, such as limitations on timing, interfaces, equipment, personnel, and database/ data files. If waivers or exceptions to specified limits and parameters are approved, they shall be identified and this subclause shall address their effects and impacts upon the test case.

5) **Requirements traceability** .This subclause shall contain:

a) Traceability from each test case in this STD to the system or software item requirements it addresses. If a test case addresses multiple requirements, traceability from each set of test procedure steps to the requirements addressed. (Alternatively, this traceability may be provided in 4.x.y.1.)

b) Traceability from each system or software item requirement covered by this STD to the test case(s) that address it. For software item testing, traceability from each software item requirement in the software item's Software Requirements Specification (SRS) and associated Interface Requirements Specifications (IRSs). For system testing, traceability from each system requirement in the system's System/Subsystem Specification (SSS) and associated IRSs. If a test case addresses multiple requirements, the traceability shall indicate the particular test procedure steps that address each requirement.

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

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

H.2.2 Contents of the Software Test Report (STR)

SOFTWARE TEST REPORT (STR)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Overview of test results
   3.1) Overall assessment of the software tested
   3.2) Impact of test environment
   3.3) Recommended improvements
4) Detailed test results
   4.x) (Project-unique identifier of a test)
      4.x.1)Summary of test results
      4.x.2)Problems encountered
      4.x.2.y)(Project-unique identifier of a test case)
      4.x.3)Deviations from test cases/procedures
      4.x.3.y)(Project-unique identifier of a test case)
5) Test log
6) Notes

A) Annexes

1) **Scope** . This clause should be divided into the following subclauses.
   1.1) **Identification** . This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Overview of test results** . This clause should be divided into the following subclauses to provide an overview of test results.
   3.1) **Overall assessment of the software tested** .This subclause shall:
      a) Provide an overall assessment of the software as demonstrated by the test results in this report
      b) Identify any remaining deficiencies, limitations, or constraints that were detected by the testing performed. Problem/change reports may be used to provide deficiency information.
      c) For each remaining deficiency, limitation, or constraint, describe:
      1) Its impact on software and system performance, including identification of requirements not met
      2) The impact on software and system design to correct it
      3) A recommended solution/approach for correcting it
   3.2) **Impact of test environment** .This subclause shall provide an assessment of the manner in which the test environment may be different from the operational environment and the effect of this difference on the test results.
   3.3) **Recommended improvements** .This subclause shall provide any recommended improvements in the design, operation, or testing of the software tested. A discussion of each recommenda-

tion and its impact on the software may be provided. If no recommended improvements are provided, this subclause shall state "None."

4) **Detailed test results** .This clause should be divided into the following subclauses to describe the detailed results for each test. Note: The word "test" means a related collection of test cases.

4.x) **(Project-unique identifier of a test)** .This subclause shall identify a test by project-unique identifier and should be divided into the following to describe the test results.

4.x.1)**Summary of test results** .This subclause shall summarize the results of the test. The summary shall include, possibly in a table, the completion status of each test case associated with the test (for example, "all results as expected," "problems encountered," "deviations required"). When the completion status is not "as expected," this subclause shall reference the following subclauses for details.

4.x.2)**Problems encountered** .This subclause should be divided into subclauses that identify each test case in which one or more problems occurred.

4.x.2.y)**(Project-unique identifier of a test case)** .This subclause shall identify by project-unique identifier a test case in which one or more problems occurred, and shall provide:

a) A brief description of the problem(s) that occurred
b) Identification of the test procedure step(s) in which they occurred
c) Reference(s) to the associated problem/change report(s) and backup data, as applicable
d) The number of times the procedure or step was repeated in attempting to correct the problem(s) and the outcome of each attempt
e) Back-up points or test steps where tests were resumed for retesting

4.x.3)**Deviations from test cases/procedures** .This subclause should be divided into subclauses that identify each test case in which deviations from test case/test procedures occurred.

4.x.3.y)**(Project-unique identifier of a test case)** .This subclause shall identify by project-unique identifier a test case in which one or more deviations occurred, and shall provide:

a) A description of the deviations) (for example, test case run in which the deviation occurred and nature of the deviation, such as substitution of required equipment, procedural steps not followed, schedule deviations). (Red-lined test procedures may be used to show the deviations)
b) The rationale for the deviation(s)
c) An assessment of the deviations' impact on the validity of the test case

5) **Test log .** This clause shall present, possibly in a figure or annex, a chronological record of the test events covered by this report. This test log shall include:

a) The date(s), time(s), and location(s) of the tests performed
b) The hardware and software configurations used for each test including, as applicable, part/model/serial number, manufacturer, revision level, and calibration date of all hardware, and version number and name for the software components used
c) The date and time of each test-related activity, the identity of the individual(s) who performed the activity, and the identities of witnesses, as applicable

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

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

Annex I

(normative)

Maintenance software product descriptions

I.1 Scope

This annex specifies the contents of software products associated with maintenance in this standard. It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

I.2 Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in the order they are referenced in this standard and include the following:

a)) Software Product Specification (SPS)
b)) Software Version Description (SVD)
c)) Computer Programming Manual (CPM)
d)) Firmware Support Manual (FSM)

I.2.1 Contents of the Software Product Specification (SPS)

SOFTWARE PRODUCT SPECIFICATION (SPS)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Requirements
   3.1) Executable software
   3.2) Source files
   3.3) Packaging requirements
4) Qualification provisions
5) Software maintenance information
   5.1) “As built” software design
   5.2) Compilation/build procedures
   5.3) Modification procedures
   5.4) Computer hardware resource utilization
6) Requirements traceability
7) Notes
A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Requirements** .This clause should be divided into the following subclauses to achieve delivery of the software and to establish the requirements that another body of software is required to meet to be considered a valid copy of the software item.Note: In past versions of the SPS, clause 3 required a presentation of the software design describing the "as built" software. That approach was modeled on hardware development, in which the final product is a design to which hardware items are required to be manufactured. For software, however, this approach does not apply. Software "manufacturing" consists of electronic duplication of the software itself, not recreation from design, and the validity of a "manufactured" copy is determined by comparison to the software itself, not to a design description. This clause therefore establishes the software itself as the criterion that is required to be matched for a body of software to be considered a valid copy of the software item. The updated software design has been placed in clause 5 below, not as a requirement, but as information to be used to modify, enhance, or otherwise maintain the software. Clause 3 is the only portion of this specification to be placed under acquirer configuration control. It is the software itself that establishes the product baseline, not a description of the software's design.
   3.1) **Executable software** .This subclause shall provide, by reference to enclosed or otherwise provided electronic media, the executable software for the software item, including any batch files, command files, data files, or other software files needed to install and operate the software on its target computer(s). In order for a body of software to be considered a valid copy of the software item's executable software, it is required to be shown to match these files exactly.
   3.2) **Source files** .This subclause shall provide, by reference to enclosed or otherwise provided electronic media, the source files for the software item, including any batch files, command files, data files, or other files needed to regenerate the executable software for the software item. In order for a body of software to be considered a valid copy of the software item's source files, it is required to be shown to match these files exactly.
   3.3) **Packaging requirements** .This subclause shall state the requirements, if any, for packaging and marking copies of the software item.
4) **Qualification provisions** .This subclause shall state the method(s) to be used to demonstrate that a given body of software is a valid copy of the software item. For example, the method for executable files might be to establish that each executable file referenced in 3.1 has an identically-named counterpart in the software in question and that each such counterpart can be shown, via bit-for-bit comparison, check sum, or other method, to be identical to the corresponding executable file. The method for source files might be comparable, using the source files referenced in 3.2.
5) **Software maintenance information** .This clause should be divided into the following subclauses to provide information needed to maintain the software item.
   5.1) **"As built" software design** .This subclause shall contain, or reference an annex or other deliverable document that contains, information describing the design of the "as built" software item. The information shall be the same as that required in a Software Design Description (SDD), Interface Design Description (IDD), and Database Design Description (DBDD), as applicable. If these documents or their equivalents are to be delivered for the "as built" software item, this subclause shall reference them. If not, the information shall be provided in this

document. Information provided in the headers, comments, and code of the source code listings may be referenced and need not be repeated in this clause. If the SDD, IDD, or DBDD is included in an annex, the subclause numbers and page numbers need not be changed.

5.2) **Compilation/build procedures** .This subclause shall describe, or reference an annex that describes, the compilation/build process to be used to create the executable files from the source files and to prepare the executable files to be loaded into firmware or other distribution media. It shall specify the compiler(s)/assembler(s) to be used, including version numbers; other hardware and software needed, including version numbers; any settings, options, or conventions to be used; and procedures for compiling/assembling, linking, and building the software item and the software system/subsystem containing the software item, including variations for different sites, configurations, versions, etc. Build procedures above the software item level may be presented in one SPS and referenced from the others.

5.3) **Modification procedures** .This subclause shall describe procedures that are required to be followed to modify the software item. It shall include or reference information on the following, as applicable:

a) Maintenance facilities, equipment, and software, and procedures for their use
b) Databases/data files used by the software item and procedures for using and modifying them
c) Design, coding, and other conventions to be followed
d) Compilation/build procedures if different from those above
e) Integration and testing procedures to be followed

5.4) **Computer hardware resource utilization** .This subclause shall describe the “as built” software item’s measured utilization of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage capacity, and communications/network equipment capacity). It shall cover all computer hardware resources included in utilization requirements for the software item, in system-level resource allocations affecting the software item, or in the Software Development Plan. If all utilization data for a given computer hardware resource is presented in a single location, such as in one SPS, this subclause may reference that source. Included for each computer hardware resource shall be:

a) The software item requirements or system-level resource allocations being satisfied. (Alternatively, the traceability to software item requirements may be provided in 6.c.)
b) The assumptions and conditions on which the utilization data are based (for example, typical usage, worst-case usage, assumption of certain events)
c) Any special considerations affecting the utilization (such as use of virtual memory, overlays, or multiprocessors or the impacts of operating system overhead, library software, or other implementation overhead)
d) The units of measure used (such as percentage of processor capacity, cycles per second, bytes of memory, kilobytes per second)
e) The level(s) at which the estimates or measures have been made (such as software unit, software item, or executable program)

6) **Requirements traceability** .This clause shall provide:

a) Traceability from each software item source file to the software unit(s) that it implements.
b) Traceability from each software unit to the source files that implement it.
c) Traceability from each computer hardware resource utilization measurement given in 5.4 to the software item requirements it addresses. (Alternatively, this traceability may be provided in 5.4.)
d) Traceability from each software item requirement regarding computer hardware resource utilization to the utilization measurements given in 5.4.

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

A) **Annexes** . Annexes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each annex shall be referenced

in the main body of the document where the data would normally have been provided. Annexes may be bound as separate documents for ease in handling. Annexes shall be lettered alphabetically (A, B, etc.).

I.2.2 Contents of the Software Version Description (SVD)

SOFTWARE VERSION DESCRIPTION (SVD)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Version description
   3.1) Inventory of materials released
   3.2) Inventory of software contents
   3.3) Changes installed
   3.4) Adaptation data
   3.5) Related documents
   3.6) Installation instructions
   3.7) Possible problems and known errors
4) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s). It shall also identify the intended recipients of the SVD to the extent that this identification affects the contents of the software released (for example, source code may not be released to all recipients.)
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this document and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Version description** .This clause should be divided into the following subclauses.
   3.1) **Inventory of materials released** .This subclause shall list by identifying numbers, titles, abbreviations, dates, version numbers, and release numbers, as applicable, all physical media (for example, listings, tapes, disks) and associated documentation that make up the software version being released. It shall include applicable security and privacy protection considerations for these items, safeguards for handling them, such as concerns for static and magnetic fields, and instructions and restrictions regarding duplication and license provisions.
   3.2) **Inventory of software contents** .This subclause shall list by identifying numbers, titles, abbreviations, dates, version numbers, and release numbers, as applicable, all computer files that make up the software version being released. Any applicable security and privacy protection considerations shall be included.

3.3) **Changes installed** .This subclause shall contain a list of all changes incorporated into the software version since the previous version. If classes or categories of changes have been used, the changes shall be separated into these classes or categories. This subclause shall identify, as applicable, the problem reports, change proposals, and change notices associated with each change and the effects, if any, of each change on system operation and on interfaces with other hardware and software. This subclause does not apply to the initial software version.

3.4) **Adaptation data** .This subclause shall identify or reference all unique-to-site data contained in the software version. For software versions after the first, this subclause shall describe changes made to the adaptation data.

3.5) **Related documents** .This subclause shall list by identifying numbers, titles, abbreviations, dates, version numbers, and release numbers, as applicable, all documents pertinent to the software version being released but not included in the release.

3.6) **Installation instructions** .This subclause shall provide or reference the following information, as applicable:

a) Instructions for installing the software version
b) Identification of other changes that have to be installed for this version to be used, including site-unique adaptation data not included in the software version
c) Security, privacy protection, or safety precautions relevant to the installation
d) Procedures for determining whether the version has been installed properly
e) A point of contact to be consulted if there are problems or questions with the installation

3.7) **Possible problems and known errors** .This subclause shall identify any possible problems or known errors with the software version at the time of release, any steps being taken to resolve the problems or errors, and instructions (either directly or by reference) for recognizing, avoiding, correcting, or otherwise handling each one. The information presented shall be appropriate to the intended recipient of the SVD (for example, a user organization may need advice on avoiding errors, a maintenance organization on correcting them).

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

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

I.2.3 Contents of the Computer Programming Manual (CPM)

COMPUTER PROGRAMMING MANUAL (CPM)

Contents

1) Scope
   1.1) Identification
   1.2) Computer system overview
   1.3) Document overview
2) Referenced documents
3) Programming environment
4) Programming information
5) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.

1.1) **Identification** .This subclause shall contain the manufacturer's name, model number, and any other identifying information for the computer system to which this document applies.

1.2) **Computer system overview** .This subclause shall briefly state the purpose of the computer system to which this document applies.

1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Programming environment** .This clause should be divided into subclauses as appropriate to provide the following information.

a) The components and configuration of the computer system

b) Operating characteristics, capabilities, and limitations, including, as applicable:

1) Machine cycle time
2) Word length
3) Memory capacity and characteristics
4) Instruction set characteristics
5) Interrupt capabilities
6) Modes of operation (e.g., batch, interactive, privileged, non-privileged)
7) Operational registers
8) Error indicators
9) Input/output characteristics
10) Special features

c) Description of the equipment (e.g., tapes, disks, other peripheral equipment) necessary to perform compilations and assemblies on the computer system. Identify (as applicable) by name and version number the editor, linker, link-editor, compiler, assembler, cross-compilers, cross-assemblers, and other utilities used, and reference appropriate manuals describing their use. Highlight any special flags or instructions necessary for loading, executing, or recording the results.

4) **Programming information** .This clause should be divided into subclauses as appropriate to provide the following information.

a) Description of the programming features of the computer's instruction set architecture, including, as applicable:

1) Data representation (e.g., byte, word, integer, floating-point, double precision)
2) Instruction formats and addressing modes
3) Special registers and words (e.g., stack pointer, program counter)
4) Control instructions (e.g., branch, jump, subroutine and procedure call instructions, privileged instructions, and the modes they operate in)
5) Subroutines and procedures (e.g., non-reentrant, reentrant, macrocode routines, argument lists, parameter passing conventions)
6) Interrupt processing
7) Timers and clocks
8) Memory protection features (e.g., read-only memory)
9) Additional features, such as instruction or data cache architecture

b) Description of each instruction, including, as applicable:

1) Use
2) Syntax
3) Condition codes set
4) Execution time
5) Machine-code format
6) Mnemonic conventions
7) Other characteristics

c) Description of input and output control programming, including, as applicable:

1) Initial loading and verification of computer memory
2) Serial and parallel data channels

3) Discrete input and output
4) Interface components
5) Device numbers, operational codes, and memory locations for peripheral equipment

d) Additional, restricted, or special programming techniques associated with the computer system (e.g., a concise description of the microprogram control clause)
e) Examples that demonstrate the programming features described above, including examples of the proper use of all categories of instructions on the computer system
f) Error detection and diagnostic features associated with the computer system, including condition codes, overflow and addressing exception interrupts, and input and output error status indicators

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

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

I.2.4 Contents of the Firmware Support Manual (FSM)

FIRMWARE SUPPORT MANUAL (FSM)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Firmware programming instructions
   3.x) (identifier of programmed firmware device)
      3.x.1)Description of pre-programmed device
      3.x.2)Software to be programmed into the device
      3.x.3)Programming equipment
      3.x.4)Programming software
      3.x.5)Programming procedures
      3.x.6)Installation and repair procedures
      3.x.7)Vendor information
4) Notes
A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system, software, and firmware devices to which this document applies, including, as applicable, identification number(s), title(s), abbreviations), version number(s), and release number(s) of the system and software and manufacturer's name and model number for each firmware device.
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.

1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Firmware programming instructions** .This clause should be divided into the following subclauses.

3.x) **(Identifier of programmed firmware device)** .This subclause shall state the project-unique identifier of a programmed firmware device to be used in the system and should be divided into the following.

3.x.1)**Description of pre-programmed device** .This subclause shall:

a) Identify by manufacturer's name and model number the firmware device to be programmed

b) Provide a complete physical description of the firmware device, including the following, as applicable:

1) Memory size, type, speed, and configuration (such as 64Kx1, 8Kx8)
2) Operating characteristics (such as access time, power requirements, logic levels)
3) Pin functional descriptions
4) Logical interfaces (such as addressing scheme, chip selection)
5) Internal and external identification scheme used
6) Timing diagrams

c) Describe the operational and environmental limits to which the firmware device may be subjected and still maintain satisfactory operation

3.x.2)**Software to be programmed into the device** .This subclause shall identify by project-unique identifier(s) the software to be programmed into the firmware device.

3.x.3)**Programming equipment** .This subclause shall describe the equipment to be used for programming and reprogramming the firmware device. It shall include computer equipment, general purpose equipment, and special equipment to be used for device erasure, loading, verification, and marking, as applicable. Each piece of equipment shall be identified by manufacturer's name, model number, and any other information that is necessary to uniquely identify that piece of equipment. A description of each piece of equipment shall be provided, including its purpose, usage, and major capabilities.

3.x.4)**Programming software** .This subclause shall describe the software to be used for programming and reprogramming the firmware device. It shall include software to be used for device erasure, loading, verification, and marking, as applicable. Each item of software shall be identified by vendor's name, software name, number, version/release, and any other information necessary to uniquely identify the item of software. A description of each item of software shall be provided, including its purpose, usage, and major capabilities.

3.x.5)**Programming procedures** .This subclause shall describe the procedures to be used for programming and reprogramming the firmware device. It shall include procedures to be used for device erasure, loading, verification, and marking, as applicable. All equipment and software necessary for each procedure shall be identified, together with any security and privacy protection measures to be applied.

3.x.6)**Installation and repair procedures** .This subclause shall contain the installation, replacement, and repair procedures for the firmware device. This subclause shall also include remove-and-replace procedures, device addressing scheme and implementation, description of the host board layout, and any procedures for ensuring continuity of operations in the event of emergencies. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

3.x.7)**Vendor information** .This clause shall include or reference any relevant information supplied by the vendor(s) of the firmware device, programming equipment, or programming software.

4) **Notes** .This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing

of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document.

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

Annex J

(normative)

User/operator software product descriptions

J.1 Scope

This annex specifies the contents of software products associated with user and operator manuals in this standard. It applies regardless of whether the software products are deliverable. This annex is a normative (that is, binding) part of this standard, subject to tailoring.

J.2 Software products covered

This following subclauses describe the contents of the software products covered by this annex. The products are listed in order they are referenced in this standard and include the following:

a)) Software User Manual (SUM)
b)) Software Input/Output Manual (SIOM)
c)) Software Center Operator Manual (SCOM)
d)) Computer Operation Manual (COM)

J.2.1 Contents of the Software User Manual (SUM)

SOFTWARE USER MANUAL (SUM)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Software summary
   3.1) Software application
   3.2) Software inventory
   3.3) Software environment
   3.4) Software organization and overview of operation
   3.5) Contingencies and alternate states and modes of operation
   3.6) Security and privacy protection
   3.7) Assistance and problem reporting
4) Access to the software
   4.1) First-time user of the software
      4.1.1)Equipment familiarization
      4.1.2)Access control
      4.1.3)Installation and setup

4.2) Initiating a session
4.3) Stopping and suspending work

5) Processing reference guide
5.1) Capabilities
5.2) Conventions
5.3) Processing procedures
5.3.x)(Aspect of software use)
5.4) Related processing
5.5) Data backup
5.6) Recovery from errors, malfunctions, and emergencies
5.7) Messages
5.8) Quick-reference guide

6) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
1.1) **Identification** .This subclause shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Software summary** .This clause should be divided into the following subclauses.
3.1) **Software application.** This subclause shall provide a brief description of the intended uses of the software. Capabilities, operating improvements, and benefits expected from its use shall be described.
3.2) **Software inventory** .This subclause shall identify all software files, including databases and data files, that are required to be installed for the software to operate. The identification shall include security and privacy protection considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.
3.3) **Software environment** .This subclause shall identify the hardware, software, manual operations, and other resources needed for a user to install and run the software. Included, as applicable, shall be identification of:
a) Computer equipment that is required to be present, including amount of memory needed, amount of auxiliary storage needed, and peripheral equipment such as printers and other input/output devices
b) Communications equipment that is required to be present
c) Other software that is required to be present, such as operating systems, databases, data files, utilities, and other supporting systems
d) Forms, procedures, or other manual operations that are required to be present
e) Other facilities, equipment, or resources that are required to be present
3.4) **Software organization and overview of operation** .This subclause shall provide a brief description of the organization and operation of the software from the user's point of view. The description shall include, as applicable:
a) Logical components of the software, from the user's point of view, and an overview of the purpose/operation of each component
b) Performance characteristics that can be expected by the user, such as:
1) Type, volume, rate of input accepted
2) Type, volume, accuracy, rate of output the software can produce

3) Typical response time and factors that affect it
4) Typical processing time and factors that affect it
5) Limitations, such as number of events that can be tracked
6) Error rate that can be expected
7) Reliability that can be expected

c) Relationship of the functions performed by the software with interfacing systems, organizations, or positions
d) Supervisory controls that can be implemented (such as passwords) to manage the software

3.5) **Contingencies and alternate states and modes of operation** .This subclause shall explain differences in what the user will be able to do with the software at times of emergency and in various states and modes of operation, if applicable.

3.6) **Security and privacy protection** .This subclause shall contain an overview of the security and privacy protection considerations associated with the software. A warning shall be included regarding making unauthorized copies of software or documents, if applicable.

3.7) **Assistance and problem reporting** .This subclause shall identify points of contact and procedures to be followed to obtain assistance and report problems encountered in using the software.

4) **Access to the software** .This clause shall contain step-by-step procedures oriented to the first time/occasional user. Enough detail shall be presented so that the user can reliably access the software before learning the details of its functional capabilities. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

4.1) **First-time user of the software** .This subclause should be divided into the following.

4.1.1)**Equipment familiarization** .This subclause shall describe the following as appropriate:

a) Procedures for turning on power and making adjustments
b) Dimensions and capabilities of the visual display screen
c) Appearance of the cursor, how to identify an active cursor if more than one cursor can appear, how to position a cursor, and how to use a cursor
d) Keyboard layout and role of different types of keys and pointing devices
e) Procedures for turning power off if special sequencing of operations is needed

4.1.2)**Access control** .This subclause shall present an overview of the access and security features of the software that are visible to the user. The following items shall be included, as applicable:

a) How and from whom to obtain a password
b) How to add, delete, or change passwords under user control
c) Security and privacy protection considerations pertaining to the storage and marking of output reports and other media that the user will generate

4.1.3)**Installation and setup** .This subclause shall describe any procedures that the user is required to perform to be identified or authorized to access or install software on the equipment, to perform the installation, to configure the software, to delete or overwrite former files or data, and to enter parameters for software operation.

4.2) **Initiating a session** .This subclause shall provide step-by-step procedures for beginning work, including any options available. A checklist for problem determination shall be included in case difficulties are encountered.

4.3) **Stopping and suspending work** .This subclause shall describe how the user can cease or interrupt use of the software and how to determine whether normal termination or cessation has occurred.

5) **Processing reference guide** .This clause shall provide the user with procedures for using the software. If procedures are complicated or extensive, additional clauses 6, 7, ... may be added in the same subclause structure as this clause and with titles meaningful to the clauses selected. The organization of the document will depend on the characteristics of the software being documented. For example, one approach is to base the clauses on the organizations in which users work, their assigned positions, their work sites, or the tasks they are required to perform. For other software, it may be more appropriate to have clause 5 be a guide to menus, clause 6 be a guide to the command

language used, and clause 7 be a guide to functions. Detailed procedures are intended to be presented in subclauses of subclause 5.3. Depending on the design of the software, the subclauses might be organized on a function-by-function, menu-by-menu, transaction-by-transaction, or other basis. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

5.1) **Capabilities** .This subclause shall briefly describe the interrelationships of the transactions, menus, functions, or other processes in order to provide an overview of the use of the software.

5.2) **Conventions** .This subclause shall describe any conventions used by the software, such as the use of colors in displays, the use of audible alarms, the use of abbreviated vocabulary, and the use of rules for assigning names or codes.

5.3) **Processing procedures** .This subclause shall explain the organization of subsequent subclauses, e.g., by function, by menu, by screen. Any necessary order in which procedures are required to be accomplished shall be described.

5.3.x)**(Aspect of software use)** .The title of this subclause shall identify the function, menu, transaction, or other process being described. This subclause shall describe and give options and examples, as applicable, of menus, graphical icons, data entry forms, user inputs, inputs from other software or hardware that may affect the software's interface with the user, outputs, diagnostic or error messages or alarms, and help facilities that can provide on-line descriptive or tutorial information. The format for presenting this information can be adapted to the particular characteristics of the software, but a consistent style of presentation shall be used, i.e., the descriptions of menus shall be consistent, the descriptions of transactions shall be consistent among themselves.

5.4) **Related processing** .This subclause shall identify and describe any related batch, offline, or background processing performed by the software that is not invoked directly by the user and is not described in subclause 5.3. Any user responsibilities to support this processing shall be specified.

5.5) **Data backup** .This subclause shall describe procedures for creating and retaining backup data that can be used to replace primary copies of data in event of errors, defects, malfunctions, or accidents.

5.6) **Recovery from errors, malfunctions, and emergencies** .This subclause shall present detailed procedures for restart or recovery from errors or malfunctions occurring during processing and for ensuring continuity of operations in the event of emergencies.

5.7) **Messages** .This subclause shall list, or refer to an annex that lists, all error messages, diagnostic messages, and information messages that can occur while accomplishing any of the user's functions. The meaning of each message and the action that should be taken after each such message shall be identified and described.

5.8) **Quick-reference guide** .If appropriate to the software, this subclause shall provide or reference a quick-reference card or page for using the software. This quick-reference guide shall summarize, as applicable, frequently-used function keys, control sequences, formats, commands, or other aspects of software use.

6) **Notes** .This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document. If clause 5 has been expanded into clause(s) 6, ..., this clause shall be numbered as the next clause following clause n.

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

J.2.2 Contents of the Software Input/Output Manual (SIOM)

SOFTWARE INPUT/OUTPUT MANUAL (SIOM)

Contents

1) **Scope** .This clause should be divided into the following subclauses.

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

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

1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.

2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.

3) **Software summary** .This clause should be divided into the following subclauses.

3.1) **Software application** .This subclause shall provide a brief description of the intended uses of the software. Capabilities, operating improvements, and benefits expected from its use shall be described.

3.2) **Software inventory** .This subclause shall identify the software files, if any, including databases and data files, that the user is responsible for requesting in order to access the software described in this manual. The identification shall include security and privacy protection considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.

3.3) **Software environment** .This subclause shall identify the hardware, software, manual operations, and other resources needed to access and use the software. This subclause shall be based on the assumption that the software is installed in a computer center or other centralized or networked environment and shall focus on the resources that a user is required to have to access and use the software in that environment. Included, as applicable, shall be identification of:

a) Computer equipment that is required to be present, such as terminals, printers, or other input/output devices
b) Communications equipment that is required to be present
c) Other software that is required to be present, such as networking software
d) Forms, procedures, or other manual operations that are required to be present
e) Other facilities, equipment, or resources that are required to be present

3.4) **Software organization and overview of operation** .This subclause shall provide a brief description of the organization and operation of the software from the user's point of view. The description shall include, as applicable:

a) Logical components of the software, from the user's point of view, including databases and data files the user can access, Database Management Systems (DBMSs), and communications paths, and an overview of the purpose/operation of each component
b) Performance characteristics that can be expected by the user, such as:
1) Type, volume, rate of input accepted
2) Type, volume, accuracy, rate of output the software can produce
3) Typical response time and factors that affect it
4) Typical processing time and factors that affect it
5) Limitations, e.g, restrictions on what data may be queried and from what location
6) Error rate that can be expected
7) Reliability that can be expected
c) Relationships of the functions performed by the software with interfacing systems and with the organizations or stations that are sources of input or recipients of output
d) Supervisory controls that can be implemented (such as passwords) to manage the software

3.5) **Contingencies and alternate states and modes of operation** .This subclause shall explain the differences in what the user will be able to do with the software at times of emergency and in various states and modes of operation, if applicable.

3.6) **Security and privacy protection** .This subclause shall contain an overview of the security and privacy protection considerations associated with the software. A warning shall be included regarding making unauthorized copies of software or documents, if applicable.

3.7) **Assistance and problem reporting** .This subclause shall identify points of contact and procedures to be followed to obtain assistance and report problems encountered in using the software.

4) **Using the software** .This clause should be divided into the following subclauses to describe how to prepare input to, and interpret output from, the software. If the software has a query capability, this subclause shall reference clause 5 for a description of this capability. If the software can be accessed

via terminal, this subclause shall reference clauses 6 through n to describe terminal processing procedures. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

4.1) **Initiation procedures** .This subclause shall contain the procedures that are required to be followed to initiate use of the software. Included may be information such as sample job request forms or sample control statements.

4.2) **Description of input** .This subclause should be divided into the following.

4.2.1)**Input conditions** .This subclause shall describe the conditions to be observed in preparing each type or class of input to the software. The conditions shall include the following, as applicable:

a) Reason for input, such as normal status report, need to update data
b) Frequency of input, such as monthly, on demand
c) Origin of input, such as the organization or station authorized to generate the input
d) Medium of input, such as magnetic tape
e) Related input that is required to be entered at the same time as this input
f) Other applicable information, such as priority; security and privacy protection considerations

4.2.2)**Input formats** .This subclause shall illustrate the layout formats to be used in the preparation of input to the software and shall explain the information that may be entered in the various clauses and lines of each format.

4.2.3)**Composition rules** .This subclause shall describe any rules and conventions that are required to be observed to prepare input. The rules of syntax, usage of punctuation, etc., shall be explained. The rules shall include the following, as applicable:

a) Input transaction length, such as 100 characters maximum
b) Format conventions, such as all input items are required to be left-justified
c) Labeling, such as usage of identifiers to denote major data sets to the software
d) Sequencing, such as order and placement of items in the input
e) Punctuation, such as spacing and use of symbols (virgule, asterisk, character combinations, etc.) to denote start and end of input, of data groups, and of fields
f) Restrictions, such as rules forbidding use of particular characters or parameter sets

4.2.4)**Input vocabulary** .This subclause shall explain the legal character combinations or codes that are required to be used to prepare input. An annex may be provided containing an ordered listing of these codes.

4.2.5)**Sample input** .This subclause shall provide examples that illustrate and explain each type or class of input acceptable by the software. Included shall be information on the following types of input, as applicable:

a) Headers denoting the start of input
b) Text or body of the input
c) Trailers denoting the end of input
d) Portions of the input that may be omitted
e) Portions of the input that may be repeated

4.3) **Description of output** .This subclause should be divided into the following.

4.3.1)**General description** .This subclause shall provide the following information, as applicable, for each type or class of output:

a) Reasons why the output is generated
b) Frequency of the output, such as monthly, on demand
c) Any modifications or variations of the basic output that are available
d) Media, such as printout, display screen, tape
e) Location where the output will appear, such as in the computer area or remotely
f) Any additional characteristics, such as priority, security and privacy protection considerations, associated output that complements the information in this output

4.3.2)**Output formats** .This subclause shall illustrate and explain the layout of each type or class of output from the software. The following aspects shall be explained, as applicable:

a) Security and privacy protection markings
b) Data that may appear in headers
c) Information that may appear in the body or text of the output, including column headings and subsets or clauses in the output format
d) Data that may appear in trailers
e) Additional characteristics, such as the meaning of special symbols

4.3.3) **Sample output** .This subclause shall provide illustrations of each type or class of output from the software. A description of each sample shall be provided, including, as applicable:

a) Meaning and use of each column, entry, etc.
b) Source, such as extracted from database, calculated
c) Characteristics, such as when omitted, range of values, unit of measure

4.3.4) **Output vocabulary** .This subclause shall describe any codes or abbreviations that appear in the output that differ from those used in the input described in subclause 4.2.4.

4.4) **Use of output** .This subclause shall explain the use of the output by the operational area or activity that receives it.

4.5) **Recovery and error correction procedures** .This subclause shall list the error codes generated by the software, give their meanings, and describe the corrective actions to be taken by the user. Also included shall be the procedures to be followed by the user with respect to restart, recovery, and continuity of operations in the event of emergencies.

4.6) **Communications diagnostics** .This subclause shall describe the diagnostic procedures available to the user for validating communications and for identifying and classifying problems.

5) **Query procedures** .This clause shall be prepared for software with a query capability. It should be divided into the following subclauses.

5.1) **Database/data file format** .This subclause shall provide a user's view of the format and content of each database and data file that can be queried. Information such as the following shall be provided for each data element, as applicable:

a) Data element name
b) Synonymous names
c) Definition
d) Format
e) Range and enumeration of values
f) Unit of measurement
g) Data item names, abbreviations, and codes

5.2) **Query capabilities** .This subclause shall identify and describe the preprogrammed and ad hoc query capabilities provided by the software.

5.3) **Query preparation** .This subclause shall provide instructions for preparing queries.

5.4) **Control instructions** .This subclause shall provide instructions for the sequencing of runs and other actions necessary to extract responses to query requests. These instructions shall include control statements that may be required by the computer system or software.

6) **User terminal processing procedures** .This clause should be divided into the following subclauses to provide the user with information on the use of terminals to accomplish processing. If the procedures are complicated or extensive, clauses 7 through n may be added in the same subclause structure as this clause and with titles meaningful to the clauses selected. The organization of the document will depend on the characteristics of the software being documented. For example, clauses might be based on the organizations in which users work, their assigned positions, work sites, or the tasks they are required to perform. For other software, it may be more appropriate to have clause 6 be a guide to menus, clause 7 be a guide to the command language, and clause 8 be a guide to functions. Detailed procedures are intended to be presented in subclauses 6.2 through 6.5. Depending on the design of the software, the subclauses might be organized on a function-by-function, menu-by-menu, transaction-by-transaction, or other basis. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

6.1) **Available capabilities** .This subclause shall describe in general terms the capabilities for retrieval, display, and update of data through terminal operations.

6.2) **Access procedures** .This subclause shall present the sequence of steps and any applicable rules pertaining to accessing the software to initiate software operations.

6.3) **Display, updates, and retrieval procedures** .This subclause should be divided into subclauses to provide the step-by-step procedures necessary to produce the displays, updates, and retrievals that are available through the use of a terminal. Each procedure shall include the name of the operation, input formats, and sample responses, as applicable.

6.4) **Recovery and error correction procedures** .This subclause shall identify error messages that may be displayed and shall indicate their meanings and any corrective actions that should be taken. Also included shall be any procedures to be followed by the user with respect to restart, recovery, and continuity of operations in the event of emergencies.

6.5) **Termination procedures** .This subclause shall present the sequence of steps necessary to terminate the processing.

7) **Notes** .This clause shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This clause shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of terms and definitions needed to understand this document. If clause 6 has been expanded into clause(s) 7, ..., this clause shall be numbered as the next clause following clause n.

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

J.2.3 Contents of the Software Center Operator Manual (SCOM)

SOFTWARE CENTER OPERATOR MANUAL (SCOM)

Contents

1) Scope
   1.1) Identification
   1.2) System overview
   1.3) Document overview
2) Referenced documents
3) Software summary
   3.1) Software application
   3.2) Software inventory
   3.3) Software environment
   3.4) Software organization and overview of operation
   3.5) Contingencies and alternate states and modes of operation
   3.6) Security and privacy protection
   3.7) Assistance and problem reporting
4) Installation and setup
5) Description of runs
   5.1) Run inventory
   5.2) Phasing
   5.3) Diagnostic procedures
   5.4) Error messages
   5.5) Description of each run
      5.5.x)Run description for (run name or identifier)
      5.5.x.1)Control input
      5.5.x.2)Run management information
      5.5.x.3)Input-Output files

5.5.x.4)Output reports
5.5.x.5)Reproduced output reports
5.5.x.6)Procedures for restart/recovery and continuity of operations

6) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain a full identification of the system and software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).
   1.2) **System overview** .This subclause shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and maintenance organizations; identify current and planned operating sites; and list other relevant documents.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Software summary** .This clause should be divided into the following subclauses.
   3.1) **Software application** .This subclause shall provide a brief description of the intended uses of the software. Capabilities, operating improvements, and benefits expected from its use shall be described.
   3.2) **Software inventory** .This subclause shall identify all software files, including databases and data files, that are required to be installed for the software to operate. The identification shall include security and privacy protection considerations for each file and identification of the software necessary to continue or resume operation in case of an emergency.
   3.3) **Software environment** .This subclause shall identify the hardware, software, manual operations, and other resources needed to install and operate the software. Included, as applicable, shall be identification of:
      a) Computer equipment that is required to be present, including amount of memory needed, amount of auxiliary storage needed, and peripheral equipment such as terminals, printers, and other input/output devices
      b) Communications equipment that is required to be present
      c) Other software that is required to be present, such as networking software, operating systems, databases, data files, utilities, permanent files that are referenced, created, or updated by the software; and databases/data files necessary to resume operation in the event of emergencies
      d) Forms, procedures, or other manual operations that are required to be present
      e) Other facilities, equipment, or resources that are required to be present
   3.4) **Software organization and overview of operation** .This subclause shall provide a brief description of the organization and operation of the software from the operator's point of view. The description shall include, as applicable:
      a) Logical components of the software, from the operator's point of view, and an overview of the purpose/operation of each component
      b) Types of input/access that can be made to the software and the software's response to each type
      c) The reports and other output that are produced by the software, including security and privacy protection considerations for each
      d) Typical run times and factors that affect them
      e) Organization of software operation into runs. This description shall use a chart, if applicable, showing how the different operations are interrelated. If sets of runs are grouped by time periods or cycles, each set of integrated operations required on a daily, weekly, etc., basis shall be presented. If runs may be grouped logically by organizational level, the

groups of runs that can be performed by each organizational level such as headquarters or central office processing, field office processing, etc., shall be presented.

f) Any system restrictions, waivers of operational standards, information oriented toward specific support areas (for example, library, small computer and teleprocessing support, interfaces with other systems), or other special aspects of processing

g) General description of the communications functions and processes of the software, including, as applicable, a diagram of the communications network used in the system

3.5) **Contingencies and alternate states and modes of operation** .This subclause shall explain the differences in software operation at times of emergency and in various states and modes of operation, if applicable.

3.6) **Security and privacy protection** .This subclause shall contain an overview of the security and privacy protection considerations associated with the software. A warning shall be included regarding making unauthorized copies of software or documents, if applicable.

3.7) **Assistance and problem reporting** .This subclause shall identify points of contact and procedures to be followed to obtain assistance and report problems encountered in operating the software.

4) **Installation and setup** .This subclause shall describe any procedures that the operator is required to perform to install the software on the equipment, to configure the software, to delete or overwrite former files or data, and to enter parameters for software operation. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

5) **Description of runs** .This clause should be divided into the following subclauses to provide a description of the runs to be performed. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.

5.1) **Run inventory** .This subclause shall provide a list of the runs to be performed, identifying the software and the jobs that make up each run. It shall include a brief summary of the purpose of each run and shall relate the list to the run descriptions included in the remainder of this clause.

5.2) **Phasing** .This subclause shall describe acceptable phasing of the software into a logical series of operations. A run may be phased to permit manual or semiautomatic checking of intermediate results, to provide the user with intermediate results for other purposes, or to permit a logical break if higher priority jobs are submitted. An example of the minimum division for most systems would be edit, file update, and report preparation.

5.3) **Diagnostic procedures** .This subclause shall provide the setup and execution procedures for any software diagnostics. Included shall be procedures for validation and trouble shooting. All parameters (both input and output), codes, and range values for diagnostic software shall be explained.

5.4) **Error messages** .This subclause shall list all error messages output by the software, along with the meaning and corresponding correction procedure for each message.

5.5) **Description of each run** .This subclause should be divided into the following.

5.5.x)**Run description for (run name or identifier)** .This subclause shall identify a run and should be divided into the following to describe the run.

5.5.x.1)**Control input** .This subclause shall provide a listing of the run stream of job control statements needed to initiate the run.

5.5.x.2)**Run management information** .This subclause shall provide the information needed to manage the run including, as applicable:

a) Peripheral and resource requirements
b) Security and privacy protection considerations
c) Method of initiation, such as on request, after another run, or at a predetermined time
d) Estimated run time
e) Required turnaround time
f) Messages and responses
g) Procedures for taking check points
h) Waivers from operational standards

5.5.x.3)**Input-output files** .This subclause shall provide information about the files and databases that serve as input to or that are created or updated by the run. Included for each

shall be information such as name, security and privacy protection, recording medium, retention schedule, and disposition.

5.5.x.4)**Output reports** .This subclause shall provide information about the reports that are produced during the run. Included for each report shall be the following information, as applicable: report identifier, product control number, report control symbol, title, security and privacy protection, media (e.g., hard copy, magnetic tape), volume of report, number of copies, and distribution of copies.

5.5.x.5)**Reproduced output reports** .This subclause shall provide information about computer-generated reports that are subsequently reproduced by other means. Included for each report shall be information such as report identification, security and privacy protection, reproduction technique, paper size, binding method, number of copies, and distribution of copies.

5.5.x.6)**Procedures for restart/recovery and continuity of operations** .This subclause shall provide procedures to be followed by operator personnel concerning restart/recovery in the event of a system failure and for continuity of operations in the event of emergencies.

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

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

J.2.4 Contents of the Computer Operation Manual (COM)

COMPUTER OPERATION MANUAL (COM)

Contents

1) Scope
   1.1) Identification
   1.2) Computer system overview
   1.3) Document overview
2) Referenced documents
3) Computer system operation
   3.1) Computer system preparation and shutdown
      3.1.1)Power on and off
      3.1.2)Initiation
      3.1.3)Shutdown
   3.2) Operating procedures
      3.2.1)Input and output procedures
      3.2.2)Monitoring procedures
      3.2.3)Off-line procedures
      3.2.4)Other procedures
   3.3) Problem-handling procedures
4) Diagnostic features
   4.1) Diagnostic features summary
   4.2) Diagnostic procedures
   4.3) Diagnostic tools
5) Notes

A) Annexes

1) **Scope** .This clause should be divided into the following subclauses.
   1.1) **Identification** .This subclause shall contain the manufacturer's name, model number, and any other identifying information for the computer system to which this COM applies.
   1.2) **Computer system overview** .This subclause shall briefly state the purpose of the computer system to which this COM applies.
   1.3) **Document overview** .This subclause shall summarize the purpose and contents of this manual and shall describe any security or privacy protection considerations associated with its use.
2) **Referenced documents** .This clause shall list the number, title, revision, date, and source of all documents referenced in this manual.
3) **Computer system operation** .This clause should be divided into the following subclauses. Safety precautions, marked by WARNING or CAUTION, shall be included where applicable.
   3.1) **Computer system preparation and shutdown** .This subclause should be divided into the following.
      3.1.1)**Power on and off** .This subclause shall contain the procedures necessary to power-on and power-off the computer system.
      3.1.2)**Initiation** .This subclause shall contain the procedures necessary to initiate operation of the computer system, including, as applicable, equipment setup, pre-operation, bootstrapping, and commands typically used during computer system initiation.
      3.1.3)**Shutdown** .This subclause shall contain the procedures necessary to terminate computer system operation.
   3.2) **Operating procedures** .This subclause should be divided into the following. If more than one mode of operation is available, instructions for each mode shall be provided.
      3.2.1)**Input and output procedures** .This subclause shall describe the input and output media (e.g., magnetic disk, tape) relevant to the computer system, state the procedures to read and write on these media, briefly describe the operating system control language, and list procedures for interactive messages and replies (e.g, terminals to use, passwords, keys).
      3.2.2)**Monitoring procedures** .This subclause shall contain the procedures to be followed for monitoring the computer system in operation. It shall describe available indicators, interpretation of those indicators, and routine and special monitoring procedures to be followed.
      3.2.3)**Off-line procedures** .This subclause shall contain the procedures necessary to operate all relevant off-line equipment of the computer system.
      3.2.4)**Other procedures** .This subclause shall contain any additional procedures to be followed by the operator (e.g., computer system alarms, computer system security or privacy protection considerations, switch over to a redundant computer system, or other measures to ensure continuity of operations in the event of emergencies).
   3.3) **Problem-handling procedures** .This subclause shall identify problems that may occur in any step of operation described in the preceding subclauses in clause 3. It shall state the error messages or other indications accompanying those problems and shall describe the automatic and manual procedures to be followed for each occurrence, including, as applicable, evaluation techniques, conditions requiring computer system shutdown, procedures for on-line intervention or abort, steps to be taken to restart computer system operation after an abort or interruption of operation, and procedures for recording information concerning a malfunction.
4) **Diagnostic features** .This clause should be divided into the following subclauses to describe diagnostics that may be performed to identify and troubleshoot malfunctions in the computer system.
   4.1) **Diagnostic features summary** .This subclause shall summarize the diagnostic features of the computer system, including error message syntax and hierarchy for fault isolation. This subclause shall describe the purpose of each diagnostic feature.
   4.2) **Diagnostic procedures** .This subclause should be divided as needed to describe the diagnostic procedures to be followed for the computer system, including:
      a) Identification of hardware, software, or firmware necessary for executing each procedure
      b) Step-by-step instructions for executing each procedure
      c) Diagnostic messages and the corresponding required action

4.3) **Diagnostic tools** .This subclause should be divided as needed to describe the diagnostic tools available for the computer system. These tools may be hardware, software, or firmware. This subclause shall identify each tool by name and number and shall describe the tool and its application.

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

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

Annex K

(informative)

Software product description format and structure

K.1 Scope

This annex provides guidance on the format and structure of deliverable software products. It is informative only, provided to aid in using this standard. Content requirements for software products associated with this standard are found in annexes E through J.

K.2 Software products covered

Figure 10 identifies the software products covered by this standard and provides guidance on the use of each product.

K.3 Format and structure for all software products

The following guidelines apply to all of the software products in annexes E through J.

K.3.1 Structure

Annexes E through J describe software products in terms of clauses and subclauses for use in developing traditional documents. When other forms of presenting information are allowed and where the intent is that a traditional document need not be produced, such as information in CASE tools or "in contractor format", this structure does not apply. In all cases, the content requirements do apply, as applicable to the project. When two or more software products are to be combined into a single deliverable, the acquirer should decide which one document should be the "parent" and which should be an annex(es) allowing the annexed document(s) to be attached without renumbering paragraphs or pages.

K.3.2 Format

The term "document" is to be interpreted to mean a collection of information regardless of the medium in which it is stored. The format for delivery of software products is project dependent, as described in annex H. Examples of format choices are: 1) whether the information is to be delivered on paper or electronic media; 2) whether electronic deliverables are required to be compatible with a specified standard, word processor, or other maintenance software; and 3) whether the information may reside in a CASE or other automated tool rather than in the form of a traditional document.

Software Product	Annex	Overview
Computer Operation Manual (COM)	J.2.4	Provides information needed to operate a given computer and its peripheral equipment. Focuses on the computer itself, not on particular software that will run on the computer. Intended for newly developed computers, special-purpose computers, or other computers for which commercial or other operation manuals are not readily available.
Computer Programming Manual (CPM)	I.2.3	Provides information needed by a programmer to understand how to program a given computer. Focuses on the computer itself, not on particular software that will run on the computer. Intended for newly developed computers, special-purpose computers, or other computers for which commercial or other programming manuals are not readily available.
Database Design Description (DBDD)	G.2.3	Describes the design of a database, that is, a collection of related data stored in one or more computerized files in a manner that can be accessed by users or computer programs via a database management system (DBMS). Can also describe the software units used to access or manipulate the data. Used as the basis for implementing the database and related software units. Provides the acquirer visibility into the design and provides information needed for software maintenance.
Firmware Support Manual (FSM)	I.2.4	Provides the information needed to program and reprogram the firmware devices of a system. Applies to read only memories (ROMs), Programmable ROMs (PROMs), Erasable PROMs (EPROMs), and other firmware devices. Describes the firmware devices and the equipment, software, and procedures needed to erase firmware devices, load software into the firmware devices, verify the load process, and mark the loaded firmware devices.
Interface Design Description (IDD)	G.2.2	Describes the interface characteristics of one or more systems, subsystems, hardware items, software items, manual operations, or other system components. May describe any number of interfaces. Can be used to supplement the System/Subsystem Design Description (SSDD), Software Design Description (SDD), and Database Design Description (DBDD). With its companion Interface Requirements Specification (IRS), serves to communicate and control interface design decisions.
Interface Requirements Specification (IRS)	F.2.3	Specifies the requirements imposed on one or more systems, subsystems, hardware items, software items, manual operations, or other system components to achieve one or more interfaces among these entities. Can cover any number of interfaces. Can be used to supplement the System/Subsystem Specification (SSS) and Software Requirements Specification (SRS) as the basis for design and qualification testing of systems and software items.
Operational Concept Description (OCD)	F.2.1	Describes a proposed system in terms of the user needs it will fulfill, its relationship to existing systems or procedures, and the ways it will be used. Used to obtain consensus among the acquirer, developer, maintenance, and user organizations on the operational concept of a proposed system. Depending on its use, an OCD may focus on communicating the user's needs to the developer or the developer's ideas to the user and other interested parties.

Figure10 — Overview of software products

Software Product	Annex	Overview
Software Center Operator Manual (SCOM)	J.2.3	Provides personnel in a computer center or other centralized or networked software installation information on how to install and operate a software system. Developed for software systems that will be installed in a computer center or other centralized or networked software installation, with users accessing the system via terminals or personal computers or submitting and receiving input and output in batch or interactive mode. Often used with the Software Input/Output Manual (SIOM). This pair of manuals is an alternative to the Software User Manual (SUM).
Software Design Description (SDD)	G.2.4	Describes the design of a software item. Describes the software item-wide design decisions, the software item architectural design, and the detailed design needed to implement the software. The SDD may be supplemented by Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs). With its associated IDDs and DBDDs, is used as the basis for implementing the software. Provides the acquirer visibility into the design and provides information needed for software maintenance.
Software Development Plan (SDP)	E.2.1	Describes a developer's plans for conducting a software development effort. Covers new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products. Provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources. Portions of the plan may be bound separately if this approach enhances their usability. Examples include plans for software configuration management and software quality assurance.
Software Input/Output Manual (SIOM)	J.2.2	Tells a user how to access, submit input to, and interpret output from, a batch or interactive software system that is run by personnel in a computer center or other centralized or networked software installation. Developed for software systems that will be installed in a computer center or other centralized or networked software installation, with users accessing the system via terminals or personal computers or submitting and receiving input and output in batch mode. Often used with the Software Center Operator Manual (SCOM). This pair of manuals is an alternative to the Software User Manual (SUM).
Software Installation Plan (SIP)	E.2.3	A plan for installing software at user sites, including preparations, user training, and conversion from existing systems. Prepared when the developer will be involved in the installation of software at user sites and when the installation process will be sufficiently complex to require a documented plan. For software embedded in a hardware-software system, an installation plan for the hardware-software system may make a separate SIP unnecessary.
Software Product Specification (SPS)	I.2.1	Contains or references the executable software, source files, and software maintenance information, including "as built" design information and compilation, build, and modification procedures, for a software item. Can be used to order the executable software and/or source files for a software item and is the primary software maintenance document for the software item.
Software Requirements Specification (SRS)	F.2.4	Specifies the requirements for a software item and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the software item's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs) referenced from the SRS. The SRS, possibly supplemented by IRSs, is used as the basis for design and qualification testing of a software item.

Figure10 — Overview of software products (continued)

Software Product	Annex	Overview
System/Subsystem Design Description (SSDD)	G.2.1	Describes the system- or subsystem-wide design and the architectural design of a system or subsystem. May be supplemented by Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs). With its associated IDDs and DBDDs, is used as the basis for further system development. May be prepared as a System Design Description or Subsystem Design Description (each using the acronym SSDD).
System/Subsystem Specification (SSS)	F.2.2	Specifies the requirements for a system or subsystem and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the system or subsystem external interfaces may be presented in the SSS or in one or more Interface Requirements Specifications (IRSs) referenced from the SSS. The SSS, possibly supplemented by IRSs, is used as the basis for design and qualification testing of a system or subsystem. May be prepared as a System Specification or Subsystem Specification (each using the acronym SSS).
Software Test Description (STD)	H.2.1	Describes the test preparations, test cases, and test procedures to be used to perform qualification testing of a software item or a software system or subsystem. Enables the acquirer to assess the adequacy of the qualification testing to be performed.
Software Test Plan (STP)	E.2.2	Describes plans for qualification testing of software items and software systems. Describes the software test environment to be used for the testing, identifies the tests to be performed, and provides schedules for test activities. There is usually a single STP for a project. Enables the acquirer to assess the adequacy of planning for software item and, if applicable, software system qualification testing.
Software Test Report (STR)	H.2.2	Provides a record of the qualification testing performed on a software item, a software system or subsystem, or other software-related item. Enables the acquirer to assess the testing and its results.
Software Transition Plan (STrP)	E.2.4	Identifies the hardware, software, and other resources needed for life cycle support of deliverable software and describes the developer's plans for transitioning deliverable items to the maintenance organization. Developed if the software support strategy calls for transition of responsibility from the developer to a separate maintenance organization.
Software User Manual (SUM)	J.2.1	Tells a hands-on software user how to install and use a software item, a group of related software items, or a software system or subsystem. May also cover a particular aspect of software operation, such as instructions for a particular position or task. Developed for software that is run by the user and has a user interface requiring on-line user input or interpretation of displayed output. If the software is embedded in a hardware-software system, user manuals or operating procedures for that system may make separate SUMs unnecessary. An alternative to the Software Center Operator Manual (SCOM) and Software Input/Output Manual (SIOM).
Software Version Description (SVD)	I.2.2	Identifies and describes a software version consisting of one or more software items. Used to release, track, and control software versions. The term "version" may be applied to the initial release of the software, to a subsequent release of that software, or to one of multiple forms of the software released at approximately the same time (for example, to different sites).

Figure10 — Overview of software products (continued)

K.3.3 Title page or identifier

If a software product is in the form of a traditional document, it is to include a title page containing, as applicable: document number; volume number; version/revision indicator; security markings or other restrictions on the handling of the document; date; document title; name, abbreviation, and any other identifier for the system, subsystem, or item to which the document applies; contract number; the software product's identification number in the contract; organization for which the document has been prepared; name and address of the preparing organization; and distribution restrictions. If the software product is in an alternative form, such as a database, this information is to be included on external and internal labels or by equivalent identification methods.

K.3.4 Table of contents and index

If a software product is in the form of a traditional document, it is to contain a table of contents providing the number, title, and page number of each titled subclause, figure, table, and annex. Manuals are to also contain an index providing an alphabetic listing of key terms and concepts covered and the pages or subclauses in which they are covered. If the software product is in an alternative form, such as a database, there is to be an internal or external table of contents containing pointers to, or instructions for accessing, information corresponding to each subclause, figure, table, and annex, and, for manuals, an external or internal index containing pointers to, or instructions for accessing, information relevant to key terms or concepts.

K.3.5 Page numbering/labeling

If a software product is in the form of a traditional document, each page is to contain a unique page number and display the document number, including version, volume, and date, as applicable. If the software product is in an alternative form, such as a database, then files, screens, or other entities are to be assigned names or numbers in such a way that desired data can be indexed and accessed.

K.3.6 Response to tailoring instructions

If a software product is in the form of a traditional document and a subclause has been tailored out of the corresponding software product description in an annex, the resulting document is to contain the corresponding subclause number and title, followed by "This subclause has been tailored out." If the software product is in an alternative form, such as a database, this representation need occur only in the table of contents or equivalent.

K.3.7 Multiple clauses and their subordinates

Any clause or subclause may be written as multiple clauses or subclauses to enhance readability.

K.3.8 Alternative presentation styles

Diagrams, tables, matrices, and other presentation styles are acceptable substitutes for text when required information can be made more readable using these styles.

K.3.9 Substitution of existing documents

Commercial or other existing documents may be substituted for all or part of a software product if they contain the required data.

K.3.10 Standard data descriptions

If a data description required by this annex has been published in a standard data element dictionary specified in the contract, reference to an entry in that dictionary is preferred over including the description itself.

K.3.11 Applicability of required contents

All content requirements in annexes E through J are "as applicable." If the contents of a given subclause do not apply to a particular project, the developer may satisfy that subclause (or a portion thereof) with "Not applicable." If there is a possibility of disagreement with the acquirer regarding which content requirements are applicable, it is advised that the developer use the Software Development Plan and joint technical and management reviews as avenues to propose and reach agreement on the applicability of required content.

Annex L

(normative)

Software product evaluations

L.1 Scope

This annex identifies the software products that are to undergo software product evaluations, identifies the criteria to be used for each evaluation, and contains a default set of definitions for the evaluation criteria. This annex is normative (that is, binding), subject to the following conditions: 1) these requirements may be tailored, 2) the developer may use alternate criteria or definitions if approved by the acquirer, and 3) if the development of a given software product has been tailored out of the standard, the requirement to evaluate that product does not apply.

L.2 Required evaluations

Figure 11 identifies the software products that are to undergo software product evaluations and states the criteria to be applied to each one. Each software product and criterion is labelled for purposes of identification and tailoring. For convenience, they may be treated as subordinate clauses of this subclause (referring to the first criterion, for example, as L.2.1.a). The software products are expressed in lower case letters to emphasize that they are generic products, not necessarily in the form of hard-copy documents. Evaluations of system-level products are to be interpreted as participation in these evaluations. Some of the criteria are subjective. Because of this, there is no requirement to prove that the criteria have been met; the requirement is to perform the evaluations using these criteria and to identify possible problems for discussion and resolution.

NOTE—The evaluation criteria for each software product has been presented in tabular form in figure 11 using the applicable criteria definitions in L.3.1-L.3.14. Note that the lower case letters in the figure are for criterion identification and tailoring purposes only. By substituting the appropriate evaluation criteria for the verb phrases in figure 11 a full set of evaluation criteria may be determined. For example, the criteria to be used for evaluation of a software test plan (item 2) are as follows: a) contains all applicable information in E.2.2 (STP) (see L.3-4); b) meets SOW, if applicable (see L.3.10); c) meets delivery requirements, if applicable (see L.3.9); d) understandable (see L.3.14);e) internally consistent (see L.3.8);f) follows Software Development Plan (see L.3.7); g) covers all software-related qualification activities in the SOW (see L.3.5); h) covers all requirements for the items under test (see L.3.5); i) consistent with other project plans (see L.3.3); and j) presents a sound approach to the testing (see L.3.11).

L.3 Criteria definitions

The following subclauses provide definitions for the criteria in figure 11 that may not be self-explanatory. The criteria are listed in alphabetical order, matching as closely as possible the wording used in figure 11.

L.3.1 Accurately describes (an item)

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

Software Product	Evaluation Criteria							
	Contains all applic. info in:	Meets SOW, if applic.	Meets deliv reqts, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
1. Software development plan (5.1.1)	a. E.2.1 (SDP)	b.	c.	d.	e.	f. (Up-dates)	g. Covers all activities/deliverables required by the contract h. Consistent with other project plans i. Presents a sound approach to the development
2. Software test plan (5.1.2, 5.1.3)	a. E.2.2 (STP)	b.	c.	d.	e.	f.	g. Covers all software-related qualification activities in the SOW h. Covers all requirements for the items under test i. Consistent with other project plans j. Presents a sound approach to the testing
3. Software installation plan (5.1.4)	a. E.2.3 (SIP)	b.	c.	d.	e.	f.	g. Covers all user site installation activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the installation
4. Software transition plan (5.1.5)	a. E.2.4 (STrP)	b.	c.	d.	e.	f.	g. Covers all transition-related activities in the SOW h. Consistent with other project plans i. Presents a sound approach to the transition
5. Operational concept (5.3.2)	a. F.2.1 (OCD)	b.	c.	d.	e.	f.	g. Feasible
6. System requirements (5.3.3)	a. F.2.2, F.2.3 (SSS, IRS)	b.	c.	d.	e.	f.	g. Covers the operational concept h. Feasible i. Testable
7. System-wide design decisions (5.4.1)	a. G.2.1, G.2.2, G.2.3 (SSDD, IDD, DBDD)	b.	c.	d.	e.	f.	g. Consistent with system requirements h. Feasible
8. System architectural design (5.4.2)	a. G.2.1, G.2.2 (SSDD, IDD)	b.	c.	d.	e.	f.	g. Covers the system requirements h. Consistent with the system-wide design decisions i. Feasible

Figure11 — Software products and associated evaluation criteria

Software Product	Evaluation Criteria						Additional Criteria
	Contains all applic. info in:	Meets SOW, if applic.	Meets deliv reqts, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	
9. Software item requirements (5.5)	a. F.2.4, F.2.3 (SRS, IRS)	b.	c.	d.	e.	f.	g. Covers system requirements allocated to the software item h. Feasible i. Testable
10. Software item-wide design decisions (5.6.1)	a. G.2.4, G.2.2, G.2.3 (SDD, IDD, DBDD)	b.	c.	d.	e.	f.	g. Consistent with software item requirements h. Feasible
11. Software item architectural design (5.6.2)	a. G.2.4, G.2.2 (SDD, IDD)	b.	c.	d.	e.	f.	g. Covers software item requirements h. Consistent with software item-wide design decisions i. Feasible
12. Software item detailed design (5.6.3)	a. G.2.4, G.2.2, G.2.3 (SDD, IDD, DBDD)	b.	c.	d.	e.	f.	g. Covers software item requirements allocated to each unit h. Consistent with software item-wide design decisions
13. Implemented software (5.7.1)	N/A	b.	c.	d.	e.	f.	g. Covers the software item detailed design
14. Software item qualification test descriptions (5.9.3)	a. H.2.1 (STD)	b.	c.	d.	e.	f.	g. Covers all software item requirements
15. Software item qualification test results (5.9.7)	a. H.2.2 (STR)	b.	c.	d.	e.	f.	g. Covers all planned software item qualification test cases h. Shows evidence that the software item meets its requirements

Figure11 — Software products and associated evaluation criteria (continued)

Software Product	Evaluation Criteria							
	Contains all applic. info in:	Meets SOW, if applic.	Meets deliv reqts, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan	Additional Criteria
16. System qualification test descriptions (5.11.3)	a. H.2.1 (STD)	b.	c.	d.	e.	f.	g. Covers all system requirements
17. System qualification test results (5.11.7)	a. H.2.2 (STR)	b.	c.	d.	e.	f.	g. Covers all planned system qualification test cases h. Shows evidence the system meets its requirements
18. Executable software (5.12.1, 5.13.1)	N/A	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All software necessary for execution is present i. Version exactly matches version that passed testing j. Deliverable media accurately labelled
19. Software version descriptions (5.12.2, 5.13.3)	a. I.2.2 (SVD)	b.	c.	d.	e.	f.	g. Accurately identifies the version of each software component (file, unit, software item, etc.) delivered h. Accurately identifies the changes incorporated
20. Software user manuals (5.12.3.1)	a. J.2.1 (SUM)	b.	c.	d.	e.	f.	g. Accurately describes software installation and use to the intended audience of this manual
21. Software input/ output manuals (5.12.3.2)	a. J.2.2 (SIOM)	b.	c.	d.	e.	f.	g. Accurately describes software input/output to the intended audience of this manual
22. Software center operator manuals (5.12.3.3)	a. J.2.3 (SCOM)	b.	c.	d.	e.	f.	g. Accurately describes software installation and operation to the intended audience of this manual
23. Computer operation manuals (5.12.3.4)	a. J.2.4 (COM)	b.	c.	d.	e.	f.	g. Accurately describes the operational characteristics of the computer
24. Source files (5.13.2)	a. I.2.1 (SPS)	b.	c.	d.	e.	f.	g. Meets delivery requirements h. All required software is present i. Version exactly matches version that passed testing j. Deliverable media accurately labelled

Figure11 — Software products and associated evaluation criteria (continued)

Software Product	Evaluation Criteria							Additional Criteria
	Contains all applic. info in:	Meets SOW, if applic.	Meets deliv reqts, if applic.	Under-stand-able	Intern. consis-tent	Follows SW dev plan		
25. "As built" software item design and related information (5.13.4)	a. I.2.1 (SPS)	b.	c.	d.	e.	f.	g. h. i. j. k.	Accurately describes the "as built" design of the software item Accurately describes compilation/build procedures Accurately describes modification procedures Source files cover all units in the software item design Measured resource utilization meets software item requirements
26. "As built" system design (5.13.5)	a. G.2.1 (SSDD)	b.	c.	d.	e.	f.	g.	Accurately describes the "as built" system design
27. Computer programming manuals (5.13.6.1)	a. I.2.3 (CPM)	b.	c.	d.	e.	f.	g.	Accurately describes the programming features of the computer
28. Firmware support manuals (5.13.6.2)	a. I.2.4 (FSM)	b.	c.	d.	e.	f.	g.	Accurately describes firmware programming features
29. Sampling of software development files (5.7.2, 5.7.3, 5.8.1, 5.8.4, 5.9.4, 5.10.1, 5.10.4, 5.11.4)	N/A	b.	N/A	d.	e.	f.	g. h. i. j. k. l.	Contents are current with the ongoing effort Adequate unit test cases/procedures/data/results Adequate unit integration test cases/procedures/ data/results Adequate software item qualification dry run results Adequate software/hardware item integration test cases/ procedures/data/results Adequate system qualification dry run results

Figure11 — Software products and associated evaluation criteria (continued)

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

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

L.3.3 Consistent with indicated product(s)

This criterion means that: (1) no statement or representation in one software product contradicts a statement or representation in the other software products, (2) a given term, acronym, or abbreviation means the same thing in all of the software products, and (3) a given item or concept is referred to by the same name or description in all of the software products.

L.3.4 Contains all applicable information in (a specified annex)

This criterion uses annexes E through J to specify the required content of software products, regardless of whether a deliverable document has been ordered. Allowances are to be made for the applicability of each topic in the annex. The formatting specified in the annex (required subclause structure and numbering) are not relevant to this evaluation.

L.3.5 Covers (a given set of items)

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

L.3.6 Feasible

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

L.3.7 Follows Software Development Plan

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

L.3.8 Internally consistent

This criterion means that: (1) no two statements or representations in a software product contradict one another, (2) a given term, acronym, or abbreviation means the same thing throughout the software product, and (3) a given item or concept is referred to by the same name or description throughout the software product.

L.3.9 Meets delivery requirements, if applicable

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

L.3.10 Meets SOW, if applicable

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

L.3.11 Presents a sound approach

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

L.3.12 Shows evidence that (an item under test) meets its requirements

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

L.3.13 Testable

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

L.3.14 Understandable

This criterion means “understandable by the intended audience.” For example, software products intended for programmer-to-programmer communication need not be understandable by non-programmers. A product that correctly identifies its audience (based on information in figure 10 in annex K) and is considered understandable to that audience meets this criterion.

Annex M

(normative)

Category and priority classifications for problem reporting

M.1 Scope

This annex contains requirements for a category and priority classification scheme to be applied to each problem submitted to the corrective action system. This annex is normative (that is, binding), subject to the following conditions: 1) these requirements may be tailored, and 2) the developer may use alternate category and priority schemes if approved by the acquirer.

M.2 Classification by category

The developer shall:

a)) Assign each problem in software products to one or more of the categories in figure 12.
b)) Assign each problem in activities to one or more of the activities in figure 1 (shown at the start of clause 5).

M.3 Classification by priority

The developer shall assign each problem in software products or activities to one of the priorities in figure 13.

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

Figure12 — Categories to be used for classifying problems in software products

Priority	Applies if a problem could:
1	a) Prevent the accomplishment of an essential capability b) Jeopardize safety, security, or other requirement designated "critical"
2	a) Adversely affect the accomplishment of an essential capability and no work-around solution is known b) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around solution is known
3	a) Adversely affect the accomplishment of an essential capability but a work-around solution is known b) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around solution is known
4	a) Result in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capability b) Result in inconvenience or annoyance for development or maintenance personnel, but does not prevent the accomplishment of those responsibilities
5	Any other effect

Figure13 — Priorities to be used for classifying problems

Annex N

(informative)

Candidate joint management reviews

N.1 Scope

This annex describes a candidate set of joint management reviews that might be held during a software development project. It is informative only, provided to aid in using this standard.

N.2 Assumptions

This annex makes the following assumptions:

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

N.3 Candidate reviews

Given below is a set of candidate joint management reviews that might be held during a software development project. There is no intent to require these reviews or to preclude alternatives or combinations of these reviews. The objectives supplement those given in 5.1 8.2. Software products are expressed in lower case letters to emphasize that they are generic products, not necessarily in the form of hard-copy documents.

N.3.1 Software plan reviews

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

a)) The software development plan
b)) The software test plan
c)) The software installation plan
d)) The software transition plan

N.3.2 Operational concept reviews

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

N.3.3 System/subsystem requirements reviews

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

N.3.4 System/subsystem design reviews

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

a)) The system- or subsystem-wide design decisions
b)) The architectural design of a software system or subsystem

N.3.5 Software requirements reviews

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

N.3.6 Software design reviews

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

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

N.3.7 Test readiness reviews

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

a)) The status of the software test environment
b)) The test cases and test procedures to be used for software item qualification testing or system qualification testing
c)) The status of the software to be tested

N.3.8 Test results reviews

These reviews are held to resolve open issues regarding the results of software item qualification testing or system qualification testing.

N.3.9 Software usability reviews

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

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

N.3.10 Software maintenance reviews

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

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

N.3.11 Critical requirement reviews

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

Annex O

(informative)

Candidate management indicators

O.1 Scope

This annex identifies a set of management indicators that might be used on a software development project. It is informative only, provided to aid in using this standard.

O.2 Candidate indicators

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

a)) Requirements volatility: total number of requirements and requirement changes over time.
b)) Software size: planned and actual number of units, lines of code, or other size measurement over time.
c)) Software staffing: planned and actual staffing levels over time.
d)) Software complexity: complexity of each software unit.
e)) Software progress: planned and actual number of software units designed, implemented, unit tested, and integrated over time.
f)) Problem/change report status: total number, number closed, number opened in the current reporting period, age, priority.
g)) Build release content: planned and actual number of software units released in each build.
h)) Computer hardware resource utilization: planned and actual use of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity) over time.
i)) Milestone performance: planned and actual dates of key project milestones.
j)) Scrap/rework: amount of resources expended to replace or revise software products after they are placed under project-level or higher configuration control.
k)) Effect of reuse: a breakout of each of the indicators above for reused versus new software products.

Annex P

(informative)

Questionnaire on use of this standard

This publication of J-STD-016-1995 is as a standard for Trial Use during the period of September 1995 through September 1996. During this period reactions and comments regarding this standard are solicited. Below is a questionnaire that can help the joint IEEE/ElA working group assess the usefulness and applicability of the standard. Please feel free to copy this questionnaire for others to use. Send responses by 30 June 1996 to:

Perry DeWeese

Lockheed-Martin Aeronautical Systems

D/73-F9 MZ0685 86 South Cobb Drive

Marietta, GA 30063

FAX 404/494-1661

email: pdweese@lasc.lockheed.com

(Circle as many as apply)

Question	Response		
What are the characteristics of your project?	Small	Medium	Large
	Safety-critical	Security / Privacy-critical	Communications system
	Weapons system	Information system	State / county / city
	DoD / Federal Agency	Multinational	In-house
Did you plan to apply the standard on international projects?	Yes	No	Already apply
If you used the standard to develop non-government commercial products, what type of software do you develop?	Financial Database Medical Development tools	Client-server Artificial Intel Operating sys Word processor	Games Other (list):
If you have used the standard on client-server project applications, did the standard meet your needs?	Yes	No	Somewhat
Was the standard helpful in supporting business process reengineering activities?	Yes	No	Somewhat
Was the standard useful in facilitating the acquirer-supplier agreement/negotiation process?	Yes	No	N/A
Does the standard conflict with established company organizational-level software standards/procedures?	Yes	No	Somewhat
How does the standard affect your software time-to-market?	Improve	Delay	None/don't know
Have you found the standard to be supportive of innovative procedures in your company?	Yes	No	Somewhat
Was the standard useful for non-traditional life-cycle models?	Incremental	Evolutionary	Other
	Yes / No	Yes / No	Yes / No
Is the concept of joint reviews useful?	Yes	No	Don't know
How are you using the software product descriptions?	Checklist of information	Information in tools or database	Traditional document
Has the standard helped improve the quality of your software products?	Yes	No	Somewhat

Question	Response					
Were you able to draw a meaningful distinction between requirements and design?	Yes		No		Don't know	
Were you able to draw a meaningful distinction between software architecture design and detailed design?	Yes		No		Don't know	
Should the standard provide requirements on the interface between developer and the acquirer's CM control of requirements?	Yes		No		Don't care	
Should the standard require that specific products be baselined at predetermined points in the development?	Yes		No		Don't care	
Should the standard provide requirements on the interface between the developer and the acquirer's configuration audits?	Yes		No		Don't care	
The standard has incorporated activities sometimes treated as non-software activities. Does this support your development process?	Software QA		Sys Rqmt Def		Sys Test	
	Yes	No	Yes	No	Yes	No
The standard does not include activities sometimes used in software development. Should this standard include requirements on:	Domain analysis		Business Process Improvement		Data Standardization	
	Yes	No	Yes	No	Yes	No
Which Military standards for software development and documentation do you currently use?	DOD-STD-2167A/2167		DOD-STD-7935A/7935		DOD-STD-1703	
Which commercial standards for software development and documentation do you currently use?	IEEE Standards		EIA Standards		In-house standards	

Please provide any comments that you feel will help improve this document. Attach additional sheets if necessary.

Contact information (optional):

Name: _____
Address: _____
_____
_____
Phone: _____
e-mail: _____