IEEE Standard for Software and System Test Documentation

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

829™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

18 July 2008

IEEE Std 829™-2008
(Revision of
IEEE Std 829-1998)

IEEE Std 829™-2008
(Revision of
IEEE Std 829-1998)

IEEE Standard for Software and System Test Documentation

Sponsor

Software & Systems Engineering Standards Committee

of the

IEEE Computer Society

Approved 27 March 2008

IEEE-SA Standards Board

Abstract: Test processes determine whether the development products of a given activity conform to the requirements of that activity and whether the system and/or software satisfies its intended use and user needs. Testing process tasks are specified for different integrity levels. These process tasks determine the appropriate breadth and depth of test documentation. The documentation elements for each type of test documentation can then be selected. The scope of testing encompasses software-based systems, computer software, hardware, and their interfaces. This standard applies to software-based systems being developed, maintained, or reused (legacy, commercial off-the-shelf, Non-Developmental Items). The term “software” also includes firmware, microcode, and documentation. Test processes can include inspection, analysis, demonstration, verification, and validation of software and software-based system products.

Keywords: integrity level, life cycle, test documentation, testing

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-7381-5746-7 STD95799
Print: ISBN 978-0-7381-5747-4 STDPD95799

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.
The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied "**AS IS**."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board

445 Hoes Lane

Piscataway, NJ 08854

USA

Introduction

This introduction is not part of IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation.

Software and software-based systems testing is a technical discipline of systems engineering. The purpose of software and software-based systems testing is to help the development organization build quality into the software and system during the life cycle processes and to validate that the quality was achieved. The test process determines whether the products of a given life cycle activity conform to the requirements of that activity, and whether the product satisfies its intended use and user needs. This determination can include inspection, demonstration, analysis, and testing of software and software-based system products. Test activities are performed in parallel with software and system development, not just at the conclusion of the development effort.

The test activities provide objective data and conclusions about software and system quality. This feedback can include anomaly identification, performance measurement, and identification of potential quality improvements for expected operating conditions across the full spectrum of the software-based systems and their interfaces. Early feedback allows the development organization to modify the products in a timely fashion and thereby reduce overall project and schedule impacts. Without a proactive approach, anomalies and associated changes are typically delayed to later in the schedule, resulting in greater costs and schedule delays.

This revision of the standard makes significant changes from the prior version. The following is a summary of the changes:

— Changed focus from being document-focused to being process-focused in keeping with IEEE/EIA Std 12207.0™-1996[a] while retaining information on test documentation.

— Added the concept of an integrity level to assist organizations in determining a recommended minimum set of testing tasks and concurrent selection of test documentation needed to support the tasks.

— Identified minimum recommended tasks for the sample integrity level scheme.

— Added an activity for choosing appropriate documents and contents.

— Added a Master Test Plan (MTP) for documenting the actual management of the total test effort.

— Added a Level Interim Test Status Report to be issued during the test execution activity.

— Added a Master Test Report for when there are multiple Level Test Reports that need consolidation. The Master Test Report may also summarize the results of the tasks identified in the Master Test Plan.

— Identified sample metrics in Annex E.

— Added the concept of independence in Annex F.

The following key concepts are emphasized in this standard:

— *Integrity levels.* Defines four integrity levels (ranging from high integrity to low integrity) to describe the importance of the software and the software-based system to the user.

[a] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards/ieee.org/).

— *Recommended minimum testing tasks for each integrity level.* Defines the recommended minimum testing tasks required for each of the four integrity levels. Includes a table of optional testing tasks for tailoring the test effort to meet project needs and application specific characteristics.

— *Intensity and rigor applied to testing tasks.* Introduces the notion that the intensity and rigor applied to the testing tasks vary according to the integrity level. Higher integrity levels require the application of greater intensity and rigor to the testing tasks. Intensity includes greater scope of testing across all normal and abnormal system operating conditions. Rigor includes more formal techniques and recording procedures.

— *Detailed criteria for testing tasks*. Defines specific criteria for each testing task, including recommended minimum criteria for correctness, consistency, completeness, accuracy, readability, and testability. The testing task descriptions include a list of the required task inputs and outputs.

— *Systems viewpoint*. Includes recommended minimum testing tasks to respond to system issues.

— *Selection of test documentation*. Both the types of test documentation and the content topics within each documentation type need to be selected based on the testing tasks associated with the identified integrity level.

Compliance with International and IEEE Standards. Defines the test processes to be compliant with life cycle process standards such as ISO/IEC 12207:1995,[b] IEEE Std 1074™-2006 and IEEE/EIA Std 12207.0-1996 as well as the entire family of IEEE software and systems engineering standards. This standard supports the full software life cycle processes, including acquisition, supply, development, operation, and maintenance. The standard is compatible with all life cycle models; however, not all life cycle models use all of the life cycle processes described in this standard.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

[b] ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association Web site at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Web site at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/.

Patents

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. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses. Other Essential Patent Claims may exist for which a statement of assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions are reasonable or non-discriminatory. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was completed, the Working Group for the Revision of IEEE Std 829-1998 had the following membership:

Claire Lohr, *Chair*
Eva Freund, *Co-Chair*
Sue Carroll, *Editor*
Gretchen Henrick, *Registrar*

***In Memoriam**, Dr. William Stephen Turner III*

Ted Ahmanson
Steve Allott
F. Scot Anderson
Steve Arbuckle
John Franklin Arce
Mark Ashford
James Bach
Mary Baggett
William Bail
Richard Bender
Robyn Brilliant
Charlotte Burns
Rick Craig
Scott P. Duncan
Elfriede Dustin
Butch Ewing
Denise Flynn
Dorothy Graham
Robin Goldsmith
Sam Guckenheimer
Hans-Ludwig Hausen
Sherry Heinze
Gary Hild
Bernard Homès
Manish Ingle
Heather King
Ed Kit
Danuta Kuc
Pamela Langford
Don Mills
Dan Misch
Robert Moeller
Carol Nibblett
Joseph B. Nutt
Ursula T. Parker
Dale Perry
Erik Petersen
Hans Schaefer
Arthur Skuja
Kathy Spurr
Keith Stobie
Teri Stokes
Ray Stone
Danielle Teague
William Stephen Turner III*
Nicole Wenzler
Jennifer Yohe

*Deceased

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

T. Scott Ankrum
Chris Bagge
Bakul Banerjee
Juris Borzovs
Pieter Botman
Juan Carreon
Norbert N. Carte
Lawrence Catchpole
Keith Chow
Raul Colcher
Paul Croll
William Cuthbertson
Geoffrey Darnton
Thomas Dineen
Antonio Doria
Scott P. Duncan
Sourav Dutta
Kenneth D. Echeberry
Carla Ewart
Harriet Feldman
Andrew Fieldsend
Eva Freund
David Friscia
Gregg Giesler
Lewis Gray
Randall Groves
John Harauz
Michelle R. Herman
Rutger A. Heunks
Werner Hoelzl
Robert Holibaugh
Bernard Homes
Peter Hung
Mark Jaeger
James St. Clair
Michael C. Jett
Cem Kaner
Robert B. Kelsey
Mark Knight
Dwayne Knirk
Thomas Kurihara
George Kyle
Marc Lacroix
David J. Leciston
Claire Lohr
Carol Long
William Lumpkins
Yury Makedonov
Edward Mccall
Keith Middleton
James Moore
Michael S. Newman
Miroslav Pavlovic
Robert Peterson
William Petit
Ulrich Pohl
Cam Posani
Robert Robinson
Randall Safier
James Sanders
Helmut H. Sandmayr
Robert Schaaf
Hans Schaefer
Richard Schrenker
David J. Schultz
Carl Singer
James Sivak
Luca Spotorno
Thomas Starai
Walter Struppler
K Subrahmanyam
R. Thayer
Thomas Tullia
Vincent J. Tume
John Walz
Oren Yuen
Janusz Zalewski
Alexandru Zamfirescu

When the IEEE-SA Standards Board approved this trial-use standard on 27 March 2008, it had the following membership:

Robert M. Grow, *Chair*
Thomas A. Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Victor Berman
Richard DeBlasio
Andrew Drozd
Mark Epstein
Alex Gelman
William R. Goldbach
Arnold M. Greenspan
Kenneth S. Hanus
James Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons
Ronald C. Petersen
Narayanan Ramachandran
Jon Rosdahl
Anne-Marie Sahazizian
Malcolm V. Thaden
Howard L. Wolfman
Don Wright

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Michael H. Kelley, *NIST Representative*

Michelle D. Turner
IEEE Standards Program Manager, Document Development

Malia Zaman
IEEE Standards Program Manager, Technical Program Development

CONTENTS

IEEE Standard for Software and System Test Documentation

1. Overview

This standard supports all software life cycle processes, including acquisition, supply, development, operation, and maintenance. This standard is compatible with all life cycle models. Not all life cycle models use all of the life cycle processes listed in this standard.

System and software test processes determine whether the outcomes of a given activity conform to the requirements of that activity and whether the development product satisfies its intended use and user needs. The determination may include analysis, demonstration, inspection, and testing of software products.

The user of this standard may invoke those software life cycle processes and the associated test processes that apply to the project. A description of the software life cycle processes may be found in ISO/IEC 12207:1995 [B24],[1] IEEE Std 1074™-2006 [B17], and IEEE/EIA Std 12207.0™-1996 [B21].

1.1 Scope

This standard applies to all software-based systems. It applies to systems and software being developed, acquired, operated, maintained, and/or reused [e.g., legacy, modified, Commercial-Off-the-Shelf (COTS), Government-Off-the-Shelf (GOTS), or Non-Developmental Items (NDIs)]. When conducting the test process, it is important to examine the software in its interactions with the other parts of the system. This standard identifies the system considerations that test processes and tasks address in determining system and software correctness and other attributes (e.g., completeness, accuracy, consistency, and testability), and the applicable resultant test documentation.

[1]The numbers in brackets correspond to those of the bibliography in Annex A.

1.2 Purpose

The purpose of this standard is to:

— Establish a common framework for test processes, activities, and tasks in support of all software life cycle processes, including acquisition, supply, development, operation, and maintenance processes

— Define the test tasks, required inputs, and required outputs

— Identify the recommended minimum test tasks corresponding to integrity levels for a four-level integrity scheme (see the used example in 4.1)

— Define the use and contents of the Master Test Plan and the Level Test Plan(s) (e.g., for component, integration, system, and acceptance test)

— Define the use and contents of related test documentation (Test Design, Test Case, Test Procedure, Anomaly Report, Test Log, Level Test Report, Interim Test Report, and Master Test Report)

1.3 Test objectives

Software-based systems are composed of software, hardware, interfaces, and operators/users. Testing processes include the consideration of interactions with all other system components, such as:

— *Environment*: Determine that the solution represented in the software-based system correctly accounts for all conditions, such as natural phenomena, physical laws of nature, business rules, physical properties, and the full ranges of the system operating environment (as applicable).

— *Operators/users*: Determine that the software communicates the proper status/condition of the software-based system to the operator/user and correctly processes all operator/user inputs to produce the required results. For incorrect operator/user inputs, ensure that the system is protected from entering into a dangerous or uncontrolled state. Validate that operator/user policies and procedures (e.g., security, interface protocols, and system assumptions) are consistently applied and used across each component interface. Validate that all forms of user documentation are complete and correct (e.g., error messages, help files, online and offline system support, user guides, and user training materials).

— *Hardware*: Determine that the software correctly interacts with each hardware interface and provides a controlled system response (i.e., graceful degradation) for hardware faults.

— *Other software*: Determine that the software interfaces correctly with other software components of the system or other systems in accordance with the requirements, and that errors are not propagated among software components.

The testing process provides objective evidence that the software-based system and its associated products:

a) Satisfy the allocated system requirements

b) Solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions)

c) Satisfy its intended use and user needs

Development of a reasonable body of evidence requires a tradeoff between the amount of time spent and a finite set of system conditions and assumptions against which to perform the testing tasks. Each project should define criteria for a reasonable body of evidence (i.e., selecting an integrity level establishes one of the basic parameters), time schedule, and scope of the testing tasks (i.e., range of system conditions and assumptions), and should select the documents and documentation content topics that are most appropriate based on this information.

Testing processes provide an objective assessment of software-based system products throughout each project's life cycle in addition to providing an objective assessment of the system at the completion of each development iteration, completion of all development, and throughout the operations phases of the life cycle. This assessment demonstrates whether software and system requirements are satisfied (e.g., at the completion of development, at installation and go-live, during operation and maintenance, and retirement). Other objectives of performing testing activities are to:

- Validate that the system satisfied the requirements for its intended use and user needs
- Validate that the right problem is solved (e.g., implement business rules and use the proper system assumptions)
- Establish responsibility and accountability for the testing processes
- Facilitate early detection and correction of software and system anomalies
- Provide an early assessment of software and system performance
- Provide factual information to management for determining the business risk of releasing the product in its current state
- Identify areas of anomaly clusters in functionality, etc.

Testing supports the primary life cycle processes. Test activities are most effective when conducted in parallel with the software development processes, not just at the completion of development.

1.4 Organization of the standard

This standard is organized into clauses (Clause 1 though Clause 17), figures, tables, and annexes (Annex A through Annex H).

- Clause 1 contains material useful in understanding and using this standard.
- Clause 2 lists normative references.
- Clause 3 provides definitions of terms, abbreviations, and conventions.
- Clause 4 explains the concept of using software integrity levels for determining the scope and rigor of test processes.
- Clause 5 describes each primary software life cycle process (using a life cycle chosen for illustrative purposes) and lists the test activities and tasks associated with the life cycle process.
- Clause 6 defines the process for choosing the test documentation contents.
- Clause 7 defines the verb "address" and requires that each possible documentation content topic be considered for inclusion in the test documentation.
- Clause 8 defines the recommended contents of a Master Test Plan.
- Clause 9 defines the recommended contents of a Level Test Plan(s).
- Clause 10 defines the recommended contents of a Level Test Design.
- Clause 11 defines the recommended contents of a Level Test Case.
- Clause 12 defines the recommended contents of a Level Test Procedure.
- Clause 13 defines the recommended contents of a Level Test Log.
- Clause 14 defines the recommended contents of an Anomaly Report.
- Clause 15 defines the recommended contents of a Level Interim Test Status Report
- Clause 16 defines the recommended contents of a Level Test Report.
- Clause 17 defines the recommended contents of a Master Test Report.

Figure 1 illustrates a possible flow for fully using this standard. It starts with the development or the use of an existing integrity level scheme (4.1 and Annex B). Then the integrity level for software-based system that requires test documentation is selected. The higher levels of integrity require more rigorous and extensive documentation. Next, there is the creation of an inventory of all testing tasks (Clause 5). The final step of that inventory is the identification of the inputs and outputs for each task (Annex C). This will include non-test-specific project documentation as well as all test documents. The integrity level definitions steps of the process (define the integrity level scheme, identify the tasks for each integrity level, select the test documents needed for each task) may be done once for an organization, and then all each system has to do is select its particular integrity level.

The identified inputs and outputs per test task (Annex C) may then be compiled into a list of documents that are needed.

For each type of document, there is a list of its possible content items (Clause 8 through Clause 17). The next step is to consider each candidate content item and to accept it as useful or reject it (Clause 7). Once the list of contents per document is completed, then the final step is to identify all content items that are already documented elsewhere (Clause 6).

Figure 2 illustrates a possible flow for partial use of this standard. Some users are not responsible for an entire test program, and they may want to consider only the test documentation most relevant to their responsibilities. Other users of this standard may not yet have or may not want full test documentation. Figure 2 can then serve as an index into the descriptions of the possible content items for the different types of test documentation. The planning documents are in Clause 8 (Master Test Plan) and Clause 9 (Level Test Plan). The test design document is described in Clause 10 (Level Test Design). The test case description is in Clause 11 (Level Test Case), and the procedures for execution of the test cases are in Clause 12 (Level Test Procedure). The documents for test results are in Clause 13 (Level Test Log), Clause 14 (Anomaly Report), Clause 15 (Level Interim Test Status Report), Clause 16 (Level Test Report), and Clause 17 (Master Test Report).

The tables contain detailed summary information:

- Table 1 provides an example of a consequence-based integrity level scheme.
- Table 2 lists the test documentation mapped to integrity levels for development.
- Table 3 provides testing task descriptions, inputs, and outputs for each life cycle process.
- Table 4 provides an example of a task description in the Master Test Plan.
- Table B.1 provides a description of integrity levels.
- Table B.2 lists definitions of consequences of failures.
- Table B.3 provides a risk assessment scheme.
- Table C.1 lists recommended minimum testing tasks required for different integrity levels.
- Table D.1 lists optional testing tasks and suggested applications.
- Table F.1 provides alternatives for test organizations.
- Table H.1 through Table H.10 provide summaries of and coverage for IEEE standards.

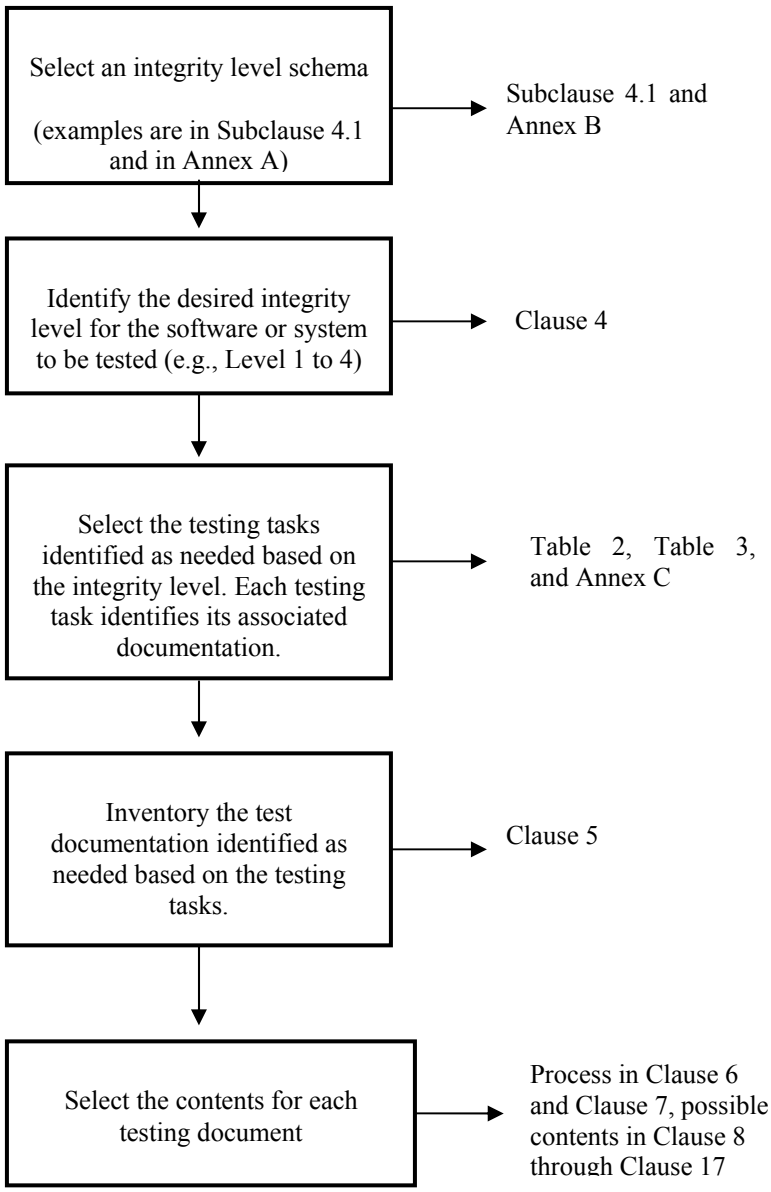

Figure 1—Full use of this standard

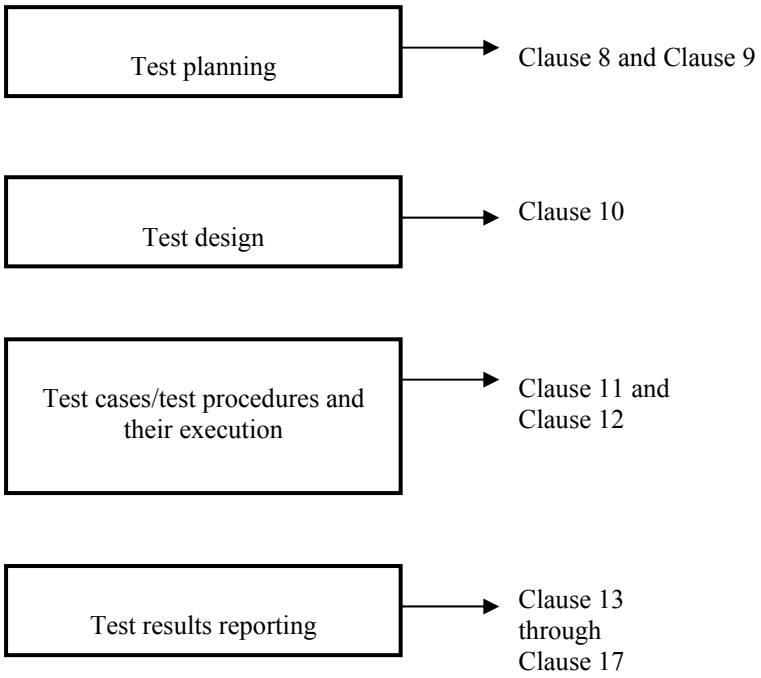

Figure 2—Partial use of this standard

The annexes contain information useful to implementing the requirements of this standard:

- Annex A provides a bibliography of standards referenced in this standard. References to documents listed in the bibliography at Annex A are denoted as [Bn].
- Annex B provides an example of a risk-based (as opposed to a consequence-based) integrity scheme consistent with Table 1.
- Annex C provides the recommended minimum test activities and tasks (including their inputs and outputs) supporting the above processes.
- Annex D provides a description of optional testing tasks.
- Annex E describes test metrics.
- Annex F provides a detailed definition of independence.
- Annex G provides examples for tailoring the documentation content topics.
- Annex H provides the mapping for compliance with IEEE/EIA 12207.1™-1997 [B22].

1.5 Audience

The audience for this standard includes software-based system suppliers, acquirers, developers, testers, and managers in both the supplier and acquirer organizations. This standard may also be of interest to software-based system users, maintenance personnel, quality assurance personnel and auditors, and participants in the legal system.

1.6 Conformance

The word *shall* identifies mandatory requirements to claim conformance with this standard. Any other words, such as *should* or *may,* indicate optional tasks that are not required to claim conformance with this standard.

Any software integrity level scheme may be used with this standard, or if the stakeholders agree, the decision can be made to not use an integrity level scheme (and simply to choose the test documents desired). The software integrity level scheme used in this standard is not mandatory, but rather, it establishes an example of the recommended minimum testing tasks for the referenced integrity scheme. To demonstrate conformance to this standard whenever different integrity schemes are used, the user should map the project-specific integrity scheme to the integrity scheme used in this standard. This mapping then establishes the minimum testing tasks that should be assigned to the project. This mapping and the associated minimum testing tasks should be documented in the highest level test plan.

Not all test efforts are initiated at the start of the life cycle acquisition process and continued through the maintenance process. If a project uses only selected life cycle processes, then conformance with this standard is achieved if the minimum testing tasks are implemented for the associated life cycle processes selected for the project. If used, the integrity level assigned to the system and software defines the minimum testing tasks. For life cycle processes that are not used by the project, the test requirements and tasks for those life cycle processes are optional testing tasks invoked as needed. Specific software development methods and technologies may eliminate development steps or combine several development steps into one. Therefore, a corresponding adaptation of the minimum testing tasks is recommended.

When this standard is invoked for existing software and the required information for the testing task is not available, then use other available sources or reconstruct the needed information to achieve conformance with this standard.

This standard provides examples for a process that leads to an effective set of test documents and their contents. It does not mandate using a particular (or any) integrity level scheme, using all of the test documents described herein, or using all of the possible content items listed for each test document. It is a suggested process for reaching appropriate conclusions for an individual organization’s needs.

1.7 Disclaimer

This standard establishes recommended minimum criteria for test processes, activities, and tasks. The implementation of these criteria does not, however, automatically ensure compliance to system or mission objectives, or prevent adverse consequences (e.g., loss of life, mission failure, and loss of system safety or security, financial or social loss). Conformance with this standard does not absolve any party from any social, moral, financial, or legal obligations.

1.8 Limitations

There are no limitations.

2. Normative references

This standard does not require the use of any normative references. Other standards considered useful in the implementation and interpretation of this standard are listed in Annex A, Bibliography.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B2] and IEEE Std 610.12™-1990 [B3] should be referenced for terms not defined in this clause.

3.1.1 acceptance testing: (A) Testing conducted to establish whether a system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system. **(B)** Formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component. This is analogous to qualification testing in IEEE/EIA Std 12207.0-1996 [B21]. Another commonly used synonym is validation testing.

3.1.2 activity: An element of work performed during the implementation of a process. An activity normally has an expected duration, cost, and resource requirements. Activities are often subdivided into tasks.

3.1.3 address: To deal with, to take into consideration; (specifically) to decide whether and when a defined documentation topic is to be included, either directly or by reference to another document. Make a decision as to whether an item is to be recorded prior to the test execution (in a tool or not in a tool), recorded during the test execution, recorded post-test execution, not recorded (addressed by the process), or excluded.

3.1.4 anomaly: Anything observed in the documentation or operation of software or system that deviates from expectations based on previously verified software products, reference documents, or other sources of indicative behavior. (adopted from IEEE Std 610.12-1990 [B3])

3.1.5 checkout: Testing conducted in the operational or support environment to ensure that a software product performs as required after installation. (adopted from IEEE Std 610.12-1990 [B3])

3.1.6 component: One part that makes up a system. A component may be hardware or software and may be subdivided into other components. (adopted from IEEE Std 610.12-1990 [B3])

NOTE—The terms “module,” “component,” and “unit” are often used interchangeably or defined to be subelements of one another in different ways depending on the context. The relationship of these terms is not yet standardized.

3.1.7 component integration testing: Testing of groups of related components.

3.1.8 component testing: Testing of individual hardware or software components. (adopted from IEEE Std 610.12-1990 [B3])

3.1.9 criticality: The degree of impact that a requirement, module, error, fault, failure, or other characteristic has on the development or operation of a system. (adopted from IEEE Std 610.12-1990 [B3])

3.1.10 development testing: Testing conducted to establish whether a new software product or software-based system (or components of it) satisfies its criteria. The criteria will vary based on the level of test being performed.

3.1.11 document: (A) A medium, and the information recorded on it, that generally has permanence and can be read by a person or a machine. Examples in software engineering include project plans, specifications, test plans, and user manuals. **(B)** To create a document as in (A). (adopted from IEEE Std 610.12-1990 [B3])

3.1.12 documentation: (A) A collection of documents on a given subject. **(B)** Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results. **(C)** The process of generating or revising a document. **(D)** The management of documents, including identification, acquisition, processing, storage, and dissemination. (adopted from IEEE Std 610.12-1990 [B3])

3.1.13 feature: A distinguishing characteristic of a system item (includes both functional and nonfunctional attributes such as performance and reusability).

3.1.14 integration testing: Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction among them. This term is commonly used for both the integration of components and the integration of entire systems. (adopted from IEEE Std 610.12-1990 [B3])

3.1.15 integrity level: (A) The degree to which software complies or must comply with a set of stakeholder-selected software and/or software-based system characteristics (e.g., software complexity, risk assessment, safety level, security level, desired performance, reliability, or cost), defined to reflect the importance of the software to its stakeholders. **(B)** A symbolic value representing this degree of compliance within an integrity level scheme.

3.1.16 integrity level scheme: A set of system characteristics (such as complexity, risk, safety level, security level, desired performance, reliability, and/or cost) selected as important to stakeholders, and arranged into discrete levels of performance or compliance (integrity levels), to help define the level of quality control to be applied in developing and/or delivering the software.

3.1.17 Interface Requirements Specification (IRS): Documentation that specifies requirements for interfaces between or among systems or components. These requirements include constraints on formats and timing. This may be included as a part of the Software Requirements Specification. (adopted from IEEE Std 610.12-1990 [B3] and IEEE Std 1012™-2004 [B10])

3.1.18 life cycle processes: A set of interrelated activities that result in the development or assessment of software products. Each activity consists of tasks. The life cycle processes may overlap one another.

3.1.19 minimum tasks: Those tasks required for the integrity level assigned to the software to be tested.

3.1.20 operational: (A) Pertaining to a system or component that is ready for use in its intended environment. **(B)** Pertaining to a system or component that is installed in its intended environment. **(C)** Pertaining to the environment in which a system or component is intended to be used. (adopted from IEEE Std 610.12-1990 [B3])

3.1.21 operational testing: Testing conducted to evaluate a system or component in its operational environment. (adopted from IEEE Std 610.12-1990 [B3])

3.1.22 optional tasks: Those tasks that may be added to the minimum testing tasks to address specific requirements. (adopted from *The Authoritative Dictionary of IEEE Standards Terms* [B2])

3.1.23 process: A set of interrelated activities, which transform inputs into outputs.

3.1.24 qualification testing: Conducted to determine whether a system or component is suitable for operational use. *See also:* **acceptance testing**; **development testing**; **operational testing**.

3.1.25 quality: (A) The degree to which a system, component, or process meets specified requirements. **(B)** The degree to which a system, component, or process meets customer or user needs or expectations. (adopted from IEEE Std 610.12-1990 [B3])

3.1.26 Request for Proposal (RFP): A document used by the acquirer as the means to announce its intention to potential bidders to acquire a specified system, software product, or software service. (adopted from IEEE Std 1074-2006 [B17])

3.1.27 required inputs: The set of items necessary to perform the minimum testing tasks mandated within any life cycle activity. (adopted from *The Authoritative Dictionary of IEEE Standards Terms* [B2])

3.1.28 required outputs: The set of items produced as a result of performing the minimum testing tasks mandated within any life cycle activity. c

3.1.29 reusable product: A 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 (COTS) products, acquirer-furnished products, products in reuse libraries, and preexisting developer products. Each use may include all or part of the product and may involve its modification. This term can be applied to any software or system product (for example, requirements or architectures), not just to software or system itself. (adopted from *The Authoritative Dictionary of IEEE Standards Terms* [B2])

3.1.30 risk: (A) The combination of the probability of occurrence and the consequences of a given future undesirable event. Risk can be associated with software and/or systems. **(B)** The combination of the probability of an abnormal event or failure and the consequence(s) of that event or failure to a system's components, operators, users, or environment. (adopted from *The Authoritative Dictionary of IEEE Standards Terms* [B2])

3.1.31 scenario: (A) A description of a series of events that may occur concurrently or sequentially. **(B)** An account or synopsis of a projected course of events or actions. (adopted from IEEE Std 1362™-1998 [B20]) **(C)** Commonly used for groups of test cases; synonyms are script, set, or suite.

3.1.32 software: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. (adopted from IEEE Std 610.12-1990 [B3])

3.1.33 software-based systems: Computer systems that are controlled by software.

3.1.34 software design description (SDD): A representation of software created to facilitate analysis, planning, implementation, and decision making. The software design description is used as a medium for communicating software design information, and it may be thought of as a blueprint or model of the system. (adopted from *The Authoritative Dictionary of IEEE Standards Terms* [B2])

3.1.35 Software Requirements Specification (SRS): Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces. (adopted from IEEE Std 610.12-1990 [B3])

3.1.36 systems integration testing: Testing conducted on multiple complete, integrated systems to evaluate their ability to communicate successfully with each other and to meet the overall integrated systems' specified requirements.

3.1.37 system testing: Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. (adopted from IEEE Std 610.12-1990 [B3])

3.1.38 task: The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project members. Related tasks are usually grouped to form activities. (adopted from IEEE Std 1074-2006 [B17])

3.1.39 test: (A) A set of one or more test cases. **(B)** A set of one or more test procedures. **(C)** A set of one or more test cases and procedures. (adopted from IEEE Std 610.12-1990 [B3] **(D)** The activity of executing (A), (B), and/or (C).

3.1.40 test approach: A particular method that will be employed to pick the particular test case values. This may vary in specificity from very general (e.g., black box or white box) to very specific (e.g., minimum and maximum boundary values).

3.1.41 test case: (A) A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. **(B)** Documentation specifying inputs, predicted results, and a set of execution conditions for a test item. (adopted from IEEE Std 610.12-1990 [B2])

3.1.42 test class: A designated grouping of test cases.

3.1.43 test design: Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests (commonly including the organization of the tests into groups). (adopted from IEEE Std 610.12-1990 [B2])

3.1.44 test effort: The activity of performing one or more testing tasks.

3.1.45 test level: A separate test effort that has its own documentation and resources (e.g., component, component integration, system, and acceptance).

3.1.46 testing: (A) An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. **(B)** To conduct an activity as in (A).

3.1.47 testing task iteration: A task that is re-performed during maintenance after having been originally performed during development.

3.1.48 test item: A software or system item that is an object of testing.

3.1.49 test plan: (A) A document describing the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. **(B)** A document that describes the technical and management approach to be followed for testing a system or component. Typical contents identify the items to be tested, tasks to be performed, responsibilities, schedules, and required resources for the testing activity. (adopted from IEEE Std 610.12-1990 [B2]) The document may be a Master Test Plan or a Level Test Plan.

3.1.50 test procedure: (A) Detailed instructions for the setup, execution, and evaluation of results for a given test case. **(B)** A document containing a set of associated instructions as in (A). **(C)** Documentation that specifies a sequence of actions for the execution of a test. (adopted from IEEE Std 982.1™-2005 [B7])

3.1.51 testware: All products produced by the testing effort, e.g., documentation and data.

3.1.52 user documentation: All documentation specifically written for users of a system, such as online help text and error messages, compact disc or hard copy system description, technical support manual, user manual, all system training materials, and release notes for patches and updates.

3.1.53 validation: (A) The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. (adopted from IEEE Std 610.12-1990 [B3]) **(B)** The process of providing evidence that the software and its associated products satisfy system requirements allocated to software at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, or use the proper system assumptions), and satisfy intended use and user needs.

3.1.54 verification: (A) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (adopted from IEEE Std 610.12-1990 [B3]) **(B)** The process of providing objective evidence that the software and its

associated products comply with requirements (e.g., for correctness, completeness, consistency, and accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance), satisfy standards, practices, and conventions during life cycle processes, and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities (e.g., building the software correctly).

3.2 Acronyms and abbreviations

ANSI	American National Standards Institute
AR	Anomaly Report
COTS	Commercial-Off-the-Shelf
GOTS	Government-Off-the-shelf
IEC	International Electrotechnical Commission
IRS	Interface Requirements Specification
ISO	International Organization for Standardization
LITSR	Level Interim Test Status Report
LTC	Level Test Case
LTD	Level Test Design
LTL	Level Test Log
LTP	Level Test Plan
LTPr	Level Test Procedure
MTP	Master Test Plan
LTR	Level Test Report
MTR	Master Test Report
NDI	Non-Developmental Item
RFP	Request for Proposal
RTM	Requirements Traceability Matrix
SRS	System Requirements Specification

4. Software and system integrity levels

In addition to a similar role in all other life cycle processes, a scheme of integrity levels provides the structured means for setting the breadth and depth of testing. A high integrity level requires additional test tasks than does a low integrity level. A high integrity level requires testing that is more in-depth than does a low integrity level. Without a requirement for a certain integrity level, the tester will not know which functions, requirements, or products require only a cursory effort and which require an intense effort.

This standard uses integrity levels to determine the testing tasks to be performed. Integrity levels may be applied to requirements, functions, groups of functions, components, and subsystems. Some of these may not require the assignment of an integrity level because their failure would impart no negative consequence on the intended system operation. The integrity scheme may be based on functionality, performance, security, or some other system or software characteristic.

Whether an integrity level scheme is mandatory is dependent on the needs of the stakeholders for the system. The user may follow the four-level schema provided as an example in this standard or may use a different schema. However, if a different schema is used, a mapping should be made between the user's schema and the example. Some software elements and components may not require the assignment of an integrity level (i.e., not applicable) because the failure would impart no consequences on the intended system operations. The user may want to add a Level 0 to Table 1. Level 0 would cover failures that would cause no consequences or are not applicable.

There shall be a documented definition of the integrity levels or the decision to not use an integrity level scheme. The integrity level (or the decision to not use an integrity level scheme) shall be assigned as a result of agreements among all specified parties [or their designated representative(s)], such as the acquirer, supplier, developer, and independent assurance authorities (e.g., a regulatory body or responsible agency).

The terms “software” and “system” are used interchangeably in this standard to mean an identified component or collection of components that include but are not restricted to software (e.g., other system components may be hardware, facilities, or people) that comprise a software-based system. The processes described in this standard are used when selecting an appropriate set of testing tasks and their associated documentation for the identified software or software-based system. The recommended testing tasks for each of the integrity levels are found in Table 2 and Table 3.

4.1 Integrity levels

An integrity level is an indication of the relative importance of software (or of a software characteristic, component, or test level) to its stakeholders, as established by a set of selected attributes such as complexity, risk assessment, safety level, security level, data integrity, desired performance, reliability, quality, cost, size, and/or other unique characteristics of the software. The characteristics used to determine the integrity level vary depending on the system’s intended application and use. The set of characteristics selected as important to stakeholders, together with its arrangement into discrete levels of performance or compliance, constitutes an integrity level scheme. An organization may have an integrity level scheme that is applicable for all projects; in which case, a project can simply reference the level in that scheme that is applicable to this system.

Unique identifiers (such as integer numbers) denote individual integrity levels within an integrity level scheme. Implicitly or explicitly, software with a relatively “higher” integrity level will require more testing effort, and more test documentation, than software with a relatively “lower” integrity level. The characteristics selected for inclusion in an integrity level scheme, and the number of integrity levels provided, will vary depending on the stakeholders’ needs and the software’s intended application and use. An assigned integrity level may change as the system evolves. Design, coding, procedural, and technology features implemented by the development organization may raise or lower the assigned integrity level. Annex B contains three tables that provide additional information. Table B.1 describes the assignment of integrity levels. Table B.2 provides definitions of consequences of failure. Finally, Table B.3 describes a risk-based scheme, including the possible integrity levels.

This standard uses the concept of integrity levels to determine the recommended minimum testing tasks to be performed. The inputs and outputs of the indicated testing tasks identify the test documentation needed. High integrity software requires a larger set of test processes, a more rigorous application of testing tasks, and as a result, more test documentation. Integrity levels can be assigned to software requirements, functions, groups of functions, software components, subsystems, systems, or groups of systems (e.g., a product line). The test processes and resultant test documentation should be tailored to specific system requirements and applications through the selection of an integrity level (with its corresponding minimum testing tasks and addition of optional testing tasks). The addition of optional testing tasks allows the test effort to include application-specific characteristics of the software.

This standard provides as an example the four-level software integrity level scheme shown below. The categories of the software integrity level scheme shown below are defined based on the seriousness of the *consequence(s)* (of incorrect behavior during execution) and on the potential for mitigation (taking steps to lessen risk by lowering the probability of a risk event’s occurrence or reducing its effect should it occur).

Each level may have an associated descriptive word, e.g.:

Level 4—Catastrophic

Level 3—Critical

Level 2—Marginal

Level 1—Negligible

Table 1— Consequence-based integrity level scheme

Description	Level
Software must execute correctly or grave consequences (loss of life, loss of system, environmental damage, economic or social loss) will occur. No mitigation is possible.	4
Software must execute correctly or the intended use (mission) of system/software will not be realized causing serious consequences (permanent injury, major system degradation, environmental damage, economic or social impact). Partial-to-complete mitigation is possible.	3
Software must execute correctly or an intended function will not be realized causing minor consequences. Complete mitigation possible.	2
Software must execute correctly or intended function will not be realized causing negligible consequences. Mitigation not required.	1

An alternative example integrity level scheme, based on *risk analysis,* is described in Annex B.

This standard does not mandate the use of the integrity level schemes. The user of this standard may use one of the two example integrity level schemes provided here, develop another scheme, or not use any integrity level scheme at all (choose test documentation based on another criteria). The use of an integrity level scheme is a recommended best practice that facilitates the selection of the most appropriate activities and tasks and, as a result, the needed test documentation.

Integrity levels are assigned to software components as part of the integrity level identification task. The test effort identifies an integrity level scheme if one is not already defined. The integrity level assigned to reused software components is in accordance with the integrity level scheme adopted for the software component(s) where they are reused, and the reused software component(s) will be evaluated for use in the context of their application. Tools that insert or translate code (e.g., optimizing compilers or auto-code generators) will be assigned the same integrity level as the integrity level assigned to the software element that the tool affects. The software-based system will be assigned the same integrity level as the highest level assigned to any individual element. The integrity level assignment will be continually reviewed and updated by conducting the integrity level identification task throughout the software development or maintenance process.

Table 3 at the end of Clause 5 identifies an example of recommended minimum testing tasks that are to be performed for each integrity level. To identify the minimum testing tasks that apply to a different selected integrity level scheme, the users of this standard may map this standard's integrity level scheme and associated minimum testing tasks to their selected integrity level scheme. The mapping of the integrity level scheme and the associated minimum testing tasks are documented in the Master Test Plan (MTP) or the highest Level Test Plan (LTP) if there is no MTP (or the highest level document selected if there is no LTP).

5. Test processes

This standard follows the structure defined in IEEE/EIA Std 12207.0-1996 [B21] for describing processes as shown in Figure 3. Each process has one or more supporting activities that are in turn executed by one or more tasks. This is a top-down method for determining which tasks are required. Once each task has been identified, its needed inputs and resultant outputs can be identified. In the case of this standard, the inputs and outputs determine which test documentation are needed.

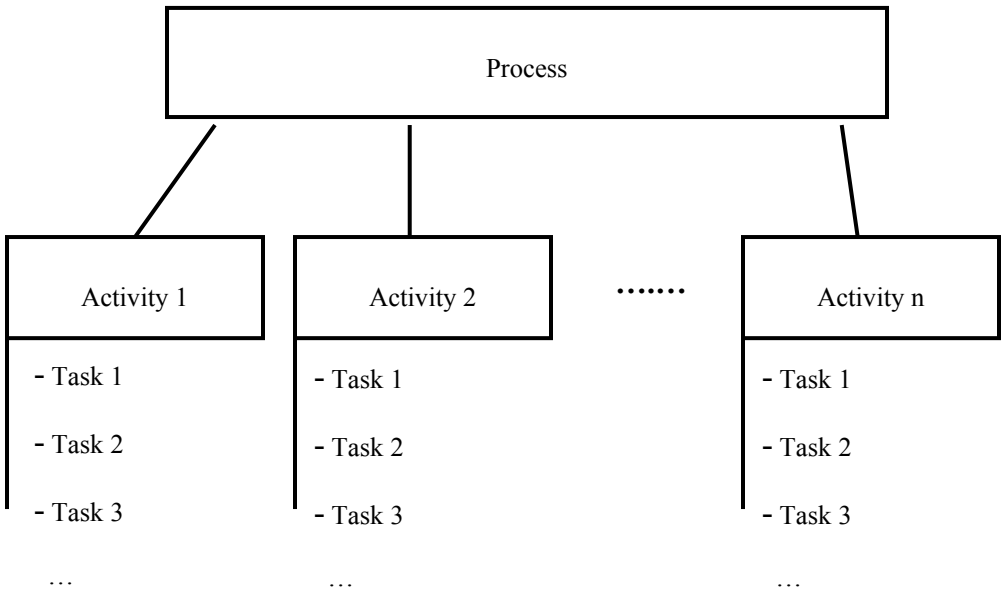

Figure 3—IEEE/EIA Std 12207.0-1996 [B21] Process Definition Infrastructure

Testing needs to support the following IEEE/EIA Std 12207.0-1996 [B21] processes:

- Management (5.1)
- Acquisition (5.2)
- Supply (5.3)
- Development (5.4)
- Operation (5.5)
- Maintenance process (5.6)

The recommended minimum test activities and tasks supporting the above processes are identified in the following subclauses and described in detail (including their inputs and outputs) in Table C.1 in Annex C. The inputs and outputs specify recommended test documentation (contents are defined in Clause 7 through Clause 16). The test effort will perform the minimum testing tasks recommended for the assigned system and software integrity level, as specified in Table 3. Table 3 contains the subset of Table C.1 that is relevant for a given integrity level. The inputs and outputs for the tasks identified by the selected integrity level list recommended test documentation.

Not all software projects include each of the lifecycle processes listed above. To be in conformance with this standard, the test processes need include only those life cycle processes used by the project.

The degree of intensity and rigor in performing and documenting the task are commensurate with the integrity level. As the integrity level decreases, so does the required scope, intensity, and degree of rigor associated with the testing tasks and documentation (see Clause 6). If a user of this standard has selected a different software integrity level scheme, then the mapping of that integrity level scheme to Table 3 will define the recommended minimum testing tasks for each of the software integrity levels.

Users of this standard should document their test processes in the highest level test plan (the MTP, if there is one); define the information and facilities necessary to manage and perform their processes, activities and tasks; and coordinate these test processes with other related aspects of the project. If no MTP is generated, these test processes should be documented in the highest level selected test documentation (Clause 9 through Clause 17). Table 2 summarizes the test documentation selected for each integrity level in this example integrity level scheme. The integrity level identifies the tasks that in turn identify the associated documents. During maintenance, use either the prior documentation selections (for changes and added new functionality) or invoke this standard as if it were for new development.

Table 2—Test documentation mapped to integrity levels for development

Integrity level	Example test documentation
4 Catastrophic	Master Test Plan
	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Test Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
	Master Test Report
3 Critical	Master Test Plan
	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Test Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
	Master Test Report
2 Marginal	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Tests Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
1 Negligible	Level Test Plan (Component Integration, System)
	Level Test Design (Component Integration, System)
	Level Test Case (Component Integration, System)
	Level Test Procedure (Component Integration, System,)
	Level Test Log (Component Integration, System)
	Anomaly Report (Component Integration, System)
	Level Test Report (Component Integration, System)

5.1 Process—management

The test management process includes the preparation of the plans for execution of the test processes.

After the initiation of the implementation of the plans, the following activities occur:

a) Monitoring of the plan execution

b) Analysis of anomalies discovered during execution of the plan

c) Reporting on progress of the test processes

d) Assessing the test results for conformance to expectations

e) Determining whether a testing task is complete

f) Checking of the test results for completeness

5.1.1 Activity—test management

The test management activity monitors and evaluates all test results. The test management activity is performed in all life cycle processes and activities. This activity continuously reviews the testing, generates the MTP if required by the integrity level, and revises the MTP as necessary. Test management generates Level Test Plans and revises them as necessary based on updated project schedules and development status. Test management coordinates the activities related to the test results with the developer and other supporting processes such as quality assurance, configuration management, and reviews and audits. Through the use of test metrics and other qualitative and quantitative measures, test management develops test trend data and identifies possible risk issues that are provided to the affected organizations, such as development and integration, to effect timely notification and resolution. At key milestones (e.g., requirements review, design review, and test readiness review), the manager of testing consolidates the test results to establish supporting evidence as to whether to proceed to the next set of system and software development activities. Whenever necessary, the manager of testing identifies lessons learned and determines whether a testing task needs to be iterated for any reason.

Test management performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Generates the Master Test Plan (MTP)

b) Performs management review of test effort

c) Performs management review and technical review support

d) Interfaces with organizational and supporting processes

e) Identifies process improvement opportunities, including lessons learned, in the conduct of test

5.2 Process—acquisition

The acquisition process begins with the definition of the need to acquire a system, software product, or software service. The process continues with the preparation and issue of an RFP and with selection of a supplier. It ends with the management of the acquisition process through to the acceptance of the system, software product, or software service. Test tasks are executed throughout the acquisition process in order to support it. These tasks range from the identification of supplier required test documentation in the RFP and the contract to the execution of acceptance testing.

5.2.1 Activity—acquisition support

At the beginning of the acquisition process, which is described above, the test activity executes several tasks that support project initiation, especially the request for proposal and contract preparation. Test tasks that support the balance of the acquisition process are described in 5.3 through 5.6.

The test effort performs, as appropriate for the selected software integrity level, these recommended minimum test tasks:

a) Scope test effort (preliminary)

b) Plan interface between the test effort and supplier (preliminary)

c) Assess System Requirements

d) Establish contract criteria for supplier/vendor testing

5.3 Process—supply

The supply process is initiated by either a decision to prepare a proposal to answer an acquirer's RFP or by signing and entering into a contract with the acquirer to provide the software-based system, software product, or service. The process continues with the determination of procedures and resources needed to manage the project, including preparation of project plans and execution of the plans through delivery of the software-based system, software product, or service to the acquirer. The test effort uses the supply process to continue determining the scope of the test effort. The test planning activity uses the contract requirements, including the overall schedules, to revise and update the test interface planning between the supplier and the acquirer.

5.3.1 Activity—test planning

The participants in the test planning activity may participate in the initiation of a request for proposal, preparation of response, contract planning, execution and control, review and evaluation, and delivery and completion activities.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Plan interface between the test effort and the supplier (continuing)

b) Scope test effort (continuing)

c) Identify metrics

d) Identify integrity level

5.4 Process—development

The development process consists of the activities and tasks of the development group. The development process covers the development activities of requirements analysis, design, coding, component integration, testing, installation, and acceptance, related to software or the software-based system product. The test activities verify the products of these development activities. The test activities described in this standard are organized into the life cycle activities of concept, requirements, design, implementation, test, and installation/checkout. Users of this standard would need to tailor these activities to reflect the practices of their organization and/or to trace their activities to those defined in this standard.

5.4.1 Activity—concept

The concept activity represents the identification of a specific implementation to solve the stakeholders' problem or fill the stakeholders' need. A preliminary system architecture may be identified and a system requirements analysis may be started during the concept activity. During the concept activity, the system architecture is selected, and system requirements are allocated to hardware, software, and user interface components. The objective of the test effort is to review the concept and requirements documents to understand better the user needs.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Review concept documentation for tester knowledge of user needs

b) Review System Requirements Document

c) Generate Test Traceability Matrix (preliminary)

d) Identify integrity level

5.4.2 Activity—requirements

The requirements activity defines the functional and nonfunctional (e.g., performance requirements usability, security, and portability) requirements, followed by interfaces external to the software, safety and security requirements, data definitions, user documentation for the system and software, installation and acceptance requirements, user operations and execution requirements, and user maintenance requirements. During the requirements activity, the testing activity participates in software requirements verification. The objective of the test effort during requirements is to verify the testability of the allocated requirements, as well as helping identify requirement ambiguity, lack of clarity, or missed requirements.

During the requirements activity, the test planning that spans several development activities begins. The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Generate Acceptance Test Plan

b) Generate System Test Plan

c) Review Software Requirements and Interface Requirements for testability

d) Identify integrity level (update)

e) Generate Test Traceability Matrix (update)

f) Identify risk(s) (test)

g) Identify security issues (test)

5.4.3 Activity—design

In the design activity, system requirements are transformed into an architecture and detailed design for each software component. The design includes databases and interfaces (external to the software and system, among the software and system components, and among the software units). The design activity creates the software and system architectural design and software and system detailed design. The objective of the test effort during the design activity is to continue the test planning.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Generate Acceptance Test Design

b) Generate System Test Design

c) Generate Component Integration Test Plan

d) Generate Component Integration Test Design

e) Generate Component Test Plan

f) Generate Component Test Design

g) Identify integrity level (update)

h) Generate Test Traceability Matrix (update)

i) Identify risk(s) (test)

j) Identify security issues (test)

5.4.4 Activity—implementation

The implementation activity implements the design into code, database structures, and related machine-executable representations. Alternatively, the design may be implemented in customization or configuration of already coded components or systems. The implementation activity includes software creation and/or modification and testing. The objective of the test effort is to verify that these implementations are correct, accurate, and complete.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Generate Acceptance Test Cases

b) Generate Acceptance Test Procedures

c) Generate System Test Cases

d) Generate System Test Procedures

e) Generate Component Integration Test Cases

f) Generate Component Integration Test Procedures

g) Generate Component Test Cases

h) Generate Component Test Procedures

i) Execute Component Test

j) Evaluate Component Test results

k) Prepare Component Test Report

l) Generate Traceability Matrix (update)

m) Perform Test Readiness Review

n) Identify integrity level (update)

o) Identify risk(s) (test)

p) Identify security issues (test)

5.4.5 Activity—test

The test activity covers software component integration, software acceptance testing, system integration, and system acceptance testing. The objective of the test effort is to verify that the software requirements and system requirements are satisfied by execution of component integration, system, and acceptance tests. Versions or releases of software are tested, for progression (testing new and corrected features) and regression (testing that no unintended changes have occurred).

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Execute Component Integration Tests

b) Evaluate Component Integration Test results

c) Prepare Component Integration Test Report

d) Execute System Test

e) Evaluate System Test results

f) Prepare System Test Report

g) Execute Acceptance Test

h) Evaluate Acceptance Test results

i) Prepare Acceptance Test Report

j) Identify risk(s)

k) Identify security issues

5.4.6 Activity—installation/checkout

The installation/checkout activity encompasses the installation of the software-based system, software product, or service in the target environment and the acquirer's acceptance review and testing of the product. The installation/checkout activity consists of the test effort supporting installation and acceptance of the installed system. The objective of the test effort is to verify the correctness of the installation in the target environment.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Support Installation Configuration Audits (physical and functional; IEEE Std 828™-2008 [B5])

b) Perform installation/checkout

c) Evaluate installation/checkout

a) Prepare Installation/Checkout Report

b) Prepare Master Test Report (if required)

c) Identify risk(s) (test)

d) Identify security issues (test)

5.5 Process—operation

The operation process covers the operation of the software product and operational support to users. The operation activity performs operational testing, system operation, and user support.

5.5.1 Activity—operational test

The operational test activity is the use of the software-based system, software product, or service by the end user in an operational environment. The operational test activity performs operational testing, system operation, and user support. The objectives of the test effort are to validate that the operational software-based system, software product, or service satisfies user requirements and meets the business needs of the organization.

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Evaluate operating procedures

b) Execute Operational Test

c) Evaluate Operational Test results

d) Prepare Operational Test Report

e) Identify risk(s) (test)

f) Identify security issues (test)

5.6 Process—maintenance

The maintenance process is activated when the software-based system or software product undergoes modifications to code and associated documentation caused by a problem, a need for improvement, or adaptation. The maintenance activity consists of modifications, migration, or retirement of the software-based system or software product during the operational process.

Modifications of the software-based system or software product will be treated as development processes and will be verified as described in 5.1 (management process) and 5.4 (development process) of this standard. Integrity levels are assessed during the maintenance process. The integrity level assignments may be revised as appropriate to reflect the requirements of the maintenance process. These modifications may be derived from requirements specified to correct software errors, to adapt to a changed operating environment, or to respond to additional user requests or enhancements.

5.6.1 Activity—maintenance test

The maintenance activity covers modifications (to code, configuration, environment, etc), migration, and retirement of the software-based system or software product. Migration of software is the movement of software to another operational environment. For migrating software, the test effort verifies that the migrated system and software meets the requirements of 5.4 through 5.5 of this standard. The retirement of software is the withdrawal of active support by the operation and maintenance organization, partial or total replacement by a new system, or installation of an upgraded system.

If the software-based system or software product was developed under this standard, the standard should continue to be followed in the maintenance process. If the software-based system or software product was not developed under this standard and appropriate documentation is neither available nor adequate, the test effort may recommend that the missing or incomplete documentation be generated. In making a determination of whether to generate missing documentation, the minimum test requirements of the assigned integrity level need to be taken into consideration.

The maintenance activity performs problem and modification analysis, modification implementation, maintenance review and acceptance, migration, and system and software retirement. The objectives of the test effort are to verify and validate modification, migration or retirement requirements, and repeat testing tasks as appropriate. Versions or releases of software are tested, for progression (testing new and corrected features) and regression (testing that no unintended changes have occurred).

The test effort performs, as appropriate for the selected integrity level, these recommended minimum test tasks:

a) Revise all affected test documentation

b) Perform anomaly evaluation

c) Test task iteration

Table 3 is a summary of the correlations between the defined test activities and where they appear during the life cycle processes (by integrity level). The test activities have been sorted in alphabetical order for convenient reference.

Table 3—Minimum testing tasks assigned to each integrity level during each activity

Test Activities	Life Cycle Processes																																							
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																				Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Acceptance Interim Test Status Report generation																									X	X	X															
Acceptance Test Case generation																					X	X	X																		
Acceptance Test Design generation																	X	X	X																						
Acceptance test execution																									X	X	X															
Acceptance Test Log generation																									X	X	X															
Acceptance Test Plan generation													X	X	X																										
Acceptance Test Procedure generation																									X	X	X															

Test Activities	Life Cycle Processes																																	
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																				Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Acceptance Test Report generation																									X	X	X															
Anomaly (operational) evaluation																																					X	X	X	
Anomaly Report generation																					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Component Interim Test Status Report generation																									X	X	X															
Component Test Case generation																					X	X	X																			
Component Test Design generation																	X	X	X																							
Component test execution																					X	X	X																			
Component Test Log generation																					X	X	X																			
Component Test Plan generation																	X	X	X																							

Test Activities	Life Cycle Processes																																			
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Component Test Procedure generation																					X	X	X																	
Component Test Report generation																					X	X	X																	
Concept documentation review for test purposes									X	X																														
Criticality analysis									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X													X	X	X	X
Master Test Report preparation																													X	X	X									
Identify improvement opportunities in the conduct of test	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Identify security issues (test)													X	X			X	X			X	X			X	X			X	X			X	X						
Identify risks													X	X			X	X			X	X			X	X			X	X			X	X						
Interface requirements review for testability													X	X	X																										

Test Activities	Life Cycle Processes																																			
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Installation/checkout																													X	X										
Installation configuration audits (functional, physical)																													X	X										
Installation/checkout report																													X	X	X									
Component Integration Interim Test Status Report generation																									X	X	X													
Interface with organizational and supporting processes													X	X			X	X			X	X			X	X			X	X			X	X						
Component Integration Test Case generation																					X	X	X	X																
Component Integration Test Design generation																	X	X	X	X																				
Component Integration test execution																									X	X	X	X												
Component Integration Test Log generation																									X	X	X	X												

Test Activities	Life Cycle Processes																																	
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Component Integration Test Plan generation																	X	X	X	X																				
Component Integration Test Procedure generation																					X	X	X	X																
Component Integration Test Report generation																									X	X	X	X												
Identify Integrity Level					X	X	X	X	X	X	X		X	X	X		X	X	X																				
Management of test effort	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Management and technical review support	X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X			X	X		
Master Test Plan generation	X	X																																						
Master Test Plan revision					X	X			X	X																											X	X		
Metrics identification					X	X	X	X																																
Operational test execution																													X	X										

Test Activities	Life Cycle Processes																															
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
Operational Test Report generation																																	X	X						
Planning the interface between the test effort and supplier	X	X	X	X	X	X	X	X																																
Scoping the test effort	X	X	X	X	X	X	X	X																																
System Interim Test Status Report generation																									X	X	X													
Software requirements evaluation for test purposes													X	X	X	X																								
System requirements review for testability	X	X	X	X																																				
System Test Case generation																					X	X	X	X																
System Test Design generation																	X	X	X	X																				
System test execution																									X	X	X	X												
System Test Log generation																									X	X	X	X												

Test Activities	Life Cycle Processes																																			
	Acquisition (5.2)				Supply (5.3)				Development (5.4)																Operation (5.5)				Maintenance (5.6)			
	Acquisition Support Test (5.2.1)				Planning (5.3.1)				Concept (5.4.1)				Requirements (5.4.2)				Design (5.4.3)				Implement-ation (5.4.4)				Test (5.4.5)				Installation/ Checkout (5.4.6)				Operation Test (5.5.1)				Maintenance Test (5.6.1)			
	Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level				Integrity Level			
	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1	4	3	2	1
System Test Plan generation													X	X	X	X																								
System Test Procedure generation																					X	X	X	X																
System Test Report generation																									X	X	X	X												
System test execution																									X	X	X	X												
Task iteration																																					X	X	X	X
Test Readiness Review participation																					X	X			X	X														
Master Test Report generation																													X	X	X	X					X	X		
Test Traceability Matrix generation									X	X	X		X	X	X		X	X	X		X	X	X		X	X	X													

6. Test documentation content selection process

Clause 8 through Clause 17 provide documentation content topics to consider for inclusion in the different testing documents. The following lists are designed to include as many commonly recognized test documentation content topics as possible, providing a comprehensive set. The full contents for each document and the definition of the document abbreviations are in Clause 8 through Clause 17:

- Clause 8 is the Master Test Plan (MTP)
- Clause 9 is the Level Test Plan (LTP)
- Clause 10 is the Level Test Design (LTD)
- Clause 11 is the Level Test Case (LTC)
- Clause 12 is the Level Test Procedure (LTPr)
- Clause 13 is the Level Test Log (LTL)
- Clause 14 is the Anomaly Report (AR)
- Clause 15 is the Level Interim Test Status Report (LITSR)
- Clause 16 is the Level Test Report (LTR)
- Clause 17 is the Master Test Report (MTR).

Users of this standard may choose to add, combine, or eliminate whole documents and/or documentation content topics based on the needs (and integrity level) of their individual systems. Annex G provides specific examples of possible alternatives using the approaches suggested in this clause. Users of this standard may also choose to require all of the possible content topics for the documents they have selected.

When the documentation content is covered elsewhere (in non-test documentation, such as a Project Plan), the outline of the document may be kept the same by referencing the other non-test document in the appropriate place. Another alternative is to modify the outline to eliminate unused content topics. It may be desirable for modified outlines to provide a list of the references that cover the eliminated content topics. The following is a suggested approach for tailoring the provided information to a specific circumstance. The suggestions include the consideration of:

a) Providing a reference to information documented elsewhere (6.1)
b) Elimination of content topics covered by the process (6.2)
c) Elimination of content topics covered by automated tools (6.3)
d) Combination or elimination of documents (6.4)
e) Combination or elimination of documentation content topics (6.5)

6.1 Provide a reference to information documented elsewhere

All of the contents are useful, but some of them may already be documented outside of the test documentation. Standard practices for testing and project activities, a project-specific glossary of terms, and system descriptions in specifications and manuals can be documented at the project management level (e.g., in a program management plan or a project management plan) and just referenced by test documentation. There is no need to copy the content of approved work instructions and system manuals into test documentation. Relevant sections of such material can be referenced in test documents and then copied for use during test activities as needed.

The other location(s) of the test documentation contents may be referenced in the testing documents. For example, if a content topic such as References is identical for all project documents and is contained in the Project Management Plan, then it would be appropriate to state in all stand-alone testing documents that References are found in the Project Management Plan.

Content topics that are applicable across multiple contexts may be published in centralized documents, perhaps on the organization's Intranet instead of being placed in individual documents. In this case, the individual document should provide a reference to the Intranet (or whatever other central) location.

6.2 Eliminate content topics covered by the process

Some organizations are using agile methods and exploratory testing techniques that may de-emphasize written documentation. For testing software with agile methods, they can choose as little as a general approach test plan, and then no detailed documentation of anything except the Anomaly Reports. Some organizations use a blended approach combining agile testing with parts of the test documentation detailed in this standard. It is up to the individual organization to ensure that there is adequate documentation to meet the identified integrity level.

6.3 Eliminate content topics covered by automated tools

There is no need to repeat test information that is completely managed by an automated tool. However, the document should reference the automated tool and the location of the information.

6.4 Choose to combine or eliminate documents

Organizations should evaluate their need to combine or eliminate some of the testing documents. If testing documents are combined (e.g., a test procedure may be incorporated by multiple test cases) and issued as a stand-alone document, then those contents that would otherwise be duplicated need only be included one time in that stand-alone document.

The integrity level scheme can be used to determine which testing documents may be eliminated. The test documentation recommended by the example integrity levels in this standard is as follows.

Integrity Level	Selected Documents
4	MTP, LTP, LTD, LTC, LTPr, LTL, AR, LITSR, LTR, MTR
3	MTP, LTP, LTD, LTC, LTPr, LTL, AR, LITSR, LTR, MTR
2	LTP, LTD, LTC, LTPr, LTR, LTL, AR, LITSR, LTR
1	LTP, LTD, LTC, LTPr, LTL, AR, LTR

The following factors would make it less likely that testing documents would be combined or eliminated:

- Longer life cycle systems (duration over time; total life cycle or only development life cycle)
- A need for a high level of integrity
- An organization with an established culture of documentation
- Large, complex systems with many interfacing elements

- Many levels of test
- The organization maintaining the system consists of personnel different from those of the development organization (the change in personnel may necessitate documenting more information than would be needed if there were no change in personnel)
- Geographically dispersed organizations
- Outsourcing
- Distributed development, especially multiple geographic regions and time zones
- Regulated industry (e.g., medical or aerospace).

The following factors would make it more likely that combination or elimination of testing documents is desirable:

- Fast turnaround from inception to delivery (short life cycle)
- A lower level of integrity is required
- Simpler systems
- Continuity of personnel

Possible situations for eliminating or combining testing documents are as follows:

- It may be useful to skip the Level Test Plan and go straight to the Test Design for systems where the Level Test Planning issues are covered by a Project Plan or similar document.
- The Level Test Design can be omitted if the needed information was already included in the Level Test Plan or other documentation.
- The Level Test Cases, Level Test Procedures, and Level Test Logs can be combined.
- Many organizations combine everything but the Anomaly Reports into one document called a "Test Plan." It is not all written at one time, but it is only one document when completed.

When documents are combined, the sections that are common to the documents being combined (e.g., the introductory sections and the glossary) need appear only once.

6.5 Choose to combine or eliminate documentation content topics

It may or may not be desirable for an organization to combine or eliminate some of the test documentation content topics within each testing document. The factors listed in 6.4 would make it more or less likely that test documentation content topics would be combined or eliminated. Although the integrity level prescribes those activities undertaken and therefore the documents needed, it is also a contributing factor in the decision regarding the breadth and depth of each document's contents. The intent of this standard is to provide an inventory of documentation contents that could be useful, rather than to impose a specific outline. Many users would prefer fewer content topics within each document to make the document more useable, understandable, and therefore useful to the organization.

7. Test documentation content topics to be addressed

"Addressed" is defined as making a decision as to whether a documentation content topic is to be documented prior to the test execution (in a tool or not in a tool), documented post-test execution, not

documented (addressed by the process), or not included. “Included” means that either the information is present or there is a reference to where it exists. All test documentation content topics listed in Clause 8 through Clause 17 shall be addressed. Entire documents may be combined or eliminated as described in Clause 6. The test documentation content decisions and rationale shall be documented in the highest level testing document and be signed-off on by all designated stakeholders.

For the documentation and topics in this standard that will be included, their content shall be arranged according to the needs of their authors. The arrangement may follow the section numbers in this standard, but it is not necessary. The section numbers in this standard are, for example, only intended for clarification and illustration for the user of this standard. For each document, the content topics have been grouped into introductory, detailed, and general topics. The purpose of this grouping is to make this standard more understandable, and it does not imply that the same grouping or order is required when the standard is used. Additional sections may be included where deemed necessary. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the documentation or made accessible to users of the document.

Figure 4 provides an overview of the types of test documentation delineated by this standard (using four levels of test as an example). It contains each document type and its primary focus. There are two types of planning documents: Master and Level (e.g., for component, component integration, system, and acceptance). There can be one MTP for a project or a series of projects. The MTP identifies how many levels of test are required. There can be one LTP for each level identified in the MTP, or multiple Level Test Plans for some levels (e.g., one LTP for each of multiple Components, one for each multiple integration steps).

Each LTP may lead to a series of documents containing more detail: The LTC(s) and Level Test Procedure(s) (LTPr) provide enough detailed information to support reproducible test execution. During test execution, there can be multiple instances of issuing an Interim Level Test Status Report. Subsequent to test execution, there can be four types of reporting documents: the LTL, AR, one Level Test Report (LTR) for each level of test, and one MTR rolling up the results and drawing final conclusions for all required levels of test.

Figure 4 also includes the clause of this standard where a recommended list of documentation content topics is provided.

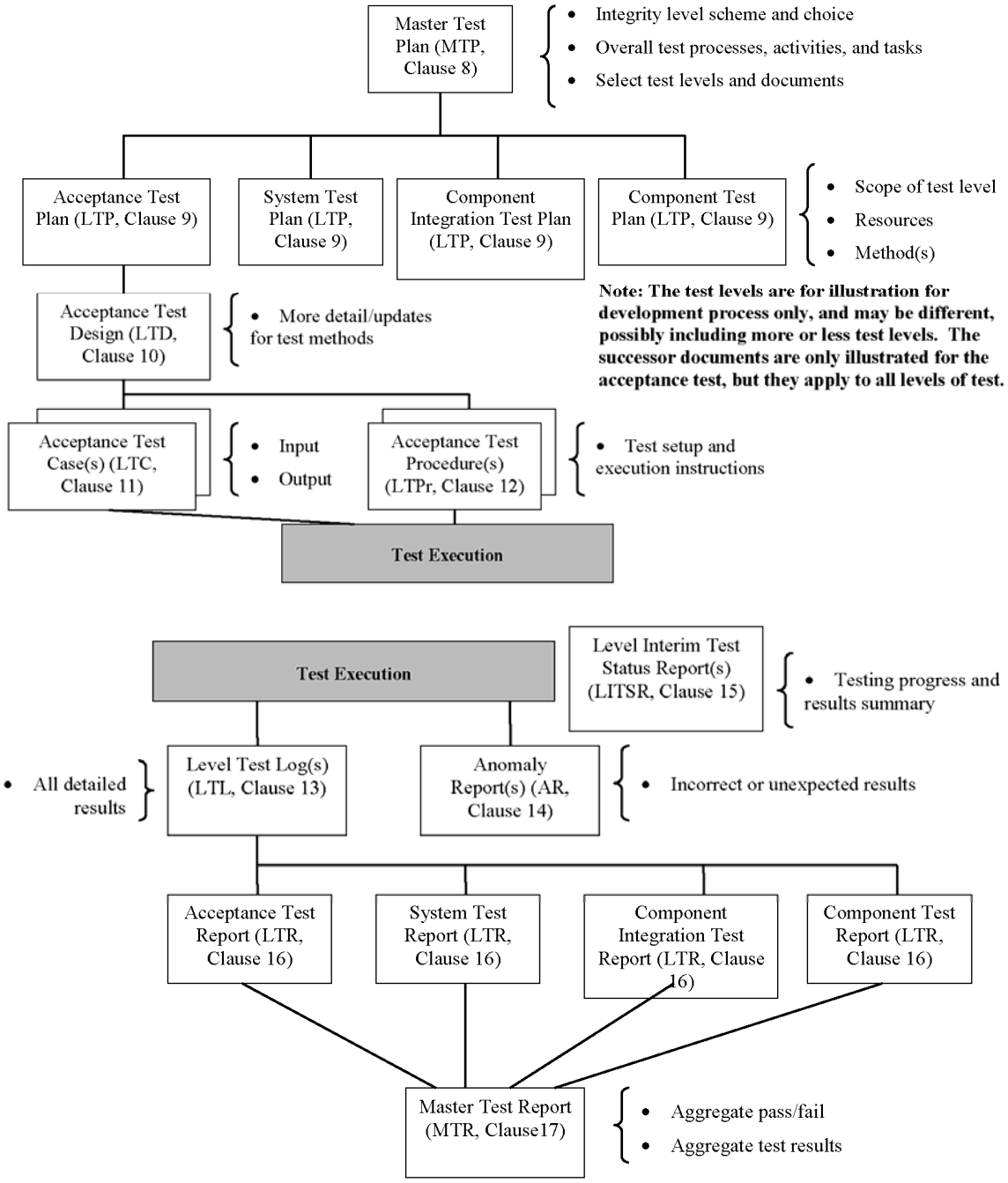

Figure 4—Test documentation overview

8. Master Test Plan (MTP)

The purpose of the Master Test Plan (MTP) is to provide an overall test planning and test management document for multiple levels of test (either within one project or across multiple projects). In view of the software requirements and the project's (umbrella) quality assurance planning, master test planning as an activity comprises selecting the constituent parts of the project's test effort; setting the objectives for each part; setting the division of labor (time, resources) and the interrelationships between the parts; identifying the risks, assumptions, and standards of workmanship to be considered and accounted for by the parts; defining the test effort's controls; and confirming the applicable objectives set by quality assurance planning. It identifies the integrity level schema and the integrity level selected, the number of levels of test, the overall tasks to be performed, and the documentation requirements.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 8.1 through 8.3. A full example of an MTP outline is shown in the boxed text.

Master Test Plan Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

1.4. System overview and key features

1.5. Test overview

1.5.1 Organization

1.5.2 Master test schedule

1.5.3 Integrity level schema

1.5.4 Resources summary

1.5.5 Responsibilities

1.5.6 Tools, techniques, methods, and metrics

2. Details of the Master Test Plan

2.1. Test processes including definition of test levels

2.1.1 Process: Management

2.1.1.1 Activity: Management of test effort

2.1.2 Process: Acquisition

2.1.2.1: Activity: Acquisition support test

2.1.3 Process: Supply

2.1.3.1 Activity: Planning test

2.1.4 Process: Development

2.1.4.1 Activity: Concept

2.1.4.2 Activity: Requirements

2.1.4.3 Activity: Design

2.1.4.4 Activity: Implementation

2.1.4.5 Activity: Test

2.1.4.6 Activity: Installation/checkout

2.1.5 Process: Operation

2.1.5.1 Activity: Operational test

2.1.6 Process: Maintenance

2.1.6.1 Activity: Maintenance test

2.2. Test documentation requirements

2.3. Test administration requirements

2.4. Test reporting requirements

3. General

3.1. Glossary

3.2. Document change procedures and history

8.1 (MTP Section 1) Introduction

Introduce the following subordinate sections. This section identifies the document and places it in context of the project-specific lifecycle. It is in this section that the entire test effort is described, including the test organization, the test schedule, and the integrity schema. A summary of required resources, responsibilities, and tools and techniques may also be included in this section.

8.1.1 (MTP Section 1.1) Document identifier

Uniquely identify a version of the document by including information such as the date of issue, the issuing organization, the author(s), the approval signatures (possibly electronic), and the status/version (e.g., draft, reviewed, corrected, or final). Identifying information may also include the reviewers and pertinent managers. This information is commonly put on an early page in the document, such as the cover page or the pages immediately following it. Some organizations put this information at the end of the document. This information may also be kept in a place other than in the text of the document (e.g., in the configuration management system or in the header or footer of the document).

8.1.2 (MTP Section 1.2) Scope

Describe the purpose, goals, and scope of the system/software test effort. Include a description of any tailoring of this standard that has been implemented. Identify the project(s) for which the Plan is being written and the specific processes and products covered by the test effort. Describe the inclusions, exclusions, and assumptions/limitations. It is important to define clearly the limits of the test effort for any test plan. This is most clearly done by specifying what is being included (inclusions) and equally important, what is being excluded (exclusions) from the test effort. For example, only the current new version of a product might be included and prior versions might be excluded from a specific test effort. In addition, there may be gray areas for the test effort (assumptions and/or limitations) where management discretion or technical assumptions are being used to direct or influence the test effort. For example, system subcomponents purchased from other suppliers might be assumed to have been tested by their originators, and thus, their testing in this effort would be limited to only test the features used as subcomponents in the new system.

It is implied that the test tasks will reflect the overall test approach and the development methodology. If the development is based on a "waterfall" methodology, then each level of the test will be executed only one time. However, if the development is based on an iterative methodology, then there will be multiple iterations of each level of test. For example, component testing may be taking place on the most recent iteration at the same time that acceptance testing is taking place on products that were developed during an earlier iteration.

The test approach identifies what will be tested and in what order for the entire gamut of testing levels (component, component integration, system, and acceptance). The test approach identifies the rationale for testing or not testing, and it identifies the rationale for the selected order of testing. The test approach describes the relationship to the development methodology. The test approach may identify the types of testing done at the different levels. For example, "thread testing" may be executed at a system level, whereas "requirements testing" may take place at the component integration as well as at a systems integration level.

The documentation (LTP, LTD, LTC, LTPr, LTR, and LITSR) required is dependent on the selection of the test approach(es).

8.1.3 (MTP Section 1.3) References

List all of the applicable reference documents. The references are separated into "external" references that are imposed external to the project and "internal" references that are imposed from within to the project. This may also be at the end of the document.

8.1.3.1 (MTP Section 1.3.1) External references

List references to the relevant policies or laws that give rise to the need for this plan, e.g.:

a) Laws

b) Government regulations

c) Standards (e.g., governmental and/or consensus)

d) Policies

The reference to this standard includes how and if it has been tailored for this project, an overview of the level(s) of documentation expected, and their contents (or a reference to an organizational standard or document that delineates the expected test documentation details).

8.1.3.2 (MTP Section 1.3.2) Internal references

List references to documents such as other plans or task descriptions that supplement this plan, e.g.:

a) Project authorization

b) Project plan (or project management plan)

c) Quality assurance plan

d) Configuration management plan

8.1.4 (MTP Section 1.4) System overview and key features

Describe the mission or business purpose of the system or software product under test (or reference where the information can be found, e.g., in a system definition document, such as a Concept of Operations). Describe the key features of the system or software under test [or reference where the information can be found, e.g., in a requirements document or COTS documentation].

8.1.5 (MTP Section 1.5) Test overview

Describe the test organization, test schedule, integrity level scheme, test resources, responsibilities, tools, techniques, and methods necessary to perform the testing.

8.1.5.1 (MTP Section 1.5.1) Organization

Describe the relationship of the test processes to other processes such as development, project management, quality assurance, and configuration management. Include the lines of communication within the testing organization(s), the authority for resolving issues raised by the testing tasks, and the authority for approving test products and processes. This may include (but should not be limited to) a visual representation, e.g., an organization chart.

8.1.5.2 (MTP Section 1.5.2) Master test schedule

Describe the test activities within the project life cycle and milestones. Summarize the overall schedule of the testing tasks, identifying where task results feed back to the development, organizational, and supporting processes (e.g., quality assurance and configuration management). Describe the task iteration policy for the re-execution of test tasks and any dependencies.

8.1.5.3 (MTP Section 1.5.3) Integrity level scheme

Describe the identified integrity level scheme for the software-based system or software product, and the mapping of the selected scheme to the integrity level scheme used in this standard. If the selected integrity level scheme is the example presented in this standard, it may be referenced and does not need to be repeated in the MTP. The MTP documents the assignment of integrity levels to individual components (e.g., requirements, functions, software modules, subsystems, non-functional characteristics, or other partitions), where there are differing integrity levels assigned within the system. At the beginning of each process, the assignment of integrity levels is reassessed with respect to changes that may need to be made in the integrity levels as a result of architecture selection, design choices, code construction, or other development activities.

8.1.5.4 (MTP Section 1.5.4) Resources summary

Summarize the test resources, including staffing, facilities, tools, and special procedural requirements (e.g., security, access rights, and documentation control).

8.1.5.5 (MTP Section 1.5.5) Responsibilities

Provide an overview of the organizational content topic(s) and responsibilities for testing tasks. Identify organizational components and their primary (they are the task leader) and secondary (they are not the leader, but providing support) test-related responsibilities.

8.1.5.6 (MTP Section 1.5.6) Tools, techniques, methods, and metrics

Describe documents, hardware and software, test tools, techniques, methods, and test environment to be used in the test process. Describe the techniques that will be used to identify and capture reusable testware. Include information regarding acquisition, training, support, and qualification for each tool, technology, and method.

Document the metrics to be used by the test effort, and describe how these metrics support the test objectives. Metrics appropriate to the Level Test Plans (e.g., component, component integration, system, and acceptance) may be included in those documents (see Annex E).

8.2 (MTP Section 2) Details of the Master Test Plan

Introduce the following subordinate sections. This section describes the test processes, test documentation requirements, and test reporting requirements for the entire test effort.

8.2.1 (MTP Section 2.1) Test processes including definition of test levels

Identify test activities and tasks to be performed for each of the test processes described in Clause 5 of this standard (or the alternative test processes defined by the user of this standard), and document those test activities and tasks. Provide an overview of the test activities and tasks for all development life cycle processes. Identify the number and sequence of levels of test. There may be a different number of levels than the example used in this standard (component, component integration, system, and acceptance). Integration is often accomplished through a series of test levels, for both component integration and systems integration. Examples of possible additional test levels include security, usability, performance, stress, recovery, and regression. Small systems may have fewer levels of test, e.g., combining system and acceptance. If the test processes are already defined by an organization's standards, a reference to those standards could be substituted for the contents of this subclause.

8.2.1.1 (MTP Sections 2.1.1 through 2.1.6) "Life cycle"[4] processes

Describe how all requirements of the standard are satisfied (e.g., by cross referencing to this standard) if the life cycle used in the MTP differs from the life cycle model in this standard. Testing requires advance planning that spans several development activities. An example of test documentation and its occurrence during the life cycle is shown in Figure 4. Include sections 2.1.1 through 2.1.6 (or sections for each life cycle, if different from the example used in this standard) for test activities and tasks as shown in the MTP Outline (Clause 8).

Address the following eight topics for each test activity (as in the example in Table 4).

a) *Test tasks:* Identify the test tasks to be performed. Table 3 provides example minimum test tasks, task criteria, and required inputs and outputs. Table C.1 provides example minimum test tasks that will be performed for each system/software integrity level. Optional test tasks may also be performed to augment the test effort to satisfy project needs. Some possible optional tasks are described in Annex D. The standard allows for optional test tasks to be used as appropriate, and/or additional test tasks not identified by this standard.
   Some test tasks are applicable to more than one integrity level. The degree of intensity and rigor in performing and documenting the task should be commensurate with the integrity level. As the integrity level increases or decreases, so do the required scope, intensity, and degree of rigor associated with the test task.

b) *Methods:* Describe the methods and procedures for each test task, including tools. Define the criteria for evaluating the test task results.

c) *Inputs:* Identify the required inputs for the test task. Specify the source of each input. For any test activity and task, any of the inputs or outputs of the preceding activities and tasks may be used.

d) *Outputs:* Identify the required outputs from the test task. The outputs of the management of test and of the test tasks will become inputs to subsequent processes and activities, as appropriate.

---

[4] "Life Cycle" sections are 6.1 Process: Management, 6.2 Process: Acquisition, 6.3 Process: Supply, 6.4 Process: Development, 6.5 Process: Operation, and 6.6 Process: Maintenance; see Clause 5.

e) *Schedule:* Describe the schedule for the test tasks. Establish specific milestones for initiating and completing each task, for the receipt of each input, and for the delivery of each output.

f) *Resources:* Identify the resources for the performance of the test tasks. Specify resources by category (e.g., staffing, tools, equipment, facilities, travel budget, and training).

g) *Risks and Assumptions:* Identify the risk(s) (e.g., schedule, resources, technical approach, or for going into production) and assumptions associated with the test tasks. Provide recommendations to eliminate, reduce, or mitigate risk(s).

h) *Roles and responsibilities:* Identify for each test task the organizational elements that have the primary and secondary responsibilities for the execution of the tas k, and the nature of the roles they will play.

Table 4—Example task description (for one task)

Task	**Generate system test design**
Methods	Ensure that test design correctly emanates from the system test plan and conforms to IEEE Std 829-2008 regarding purpose, format, and content.
Inputs	System Test Plan, IEEE Std 829-2008
Outputs	System Test Design, provide input to Master Test Report
Schedule	Initiate (with all inputs received) 30 days after the start of the project. Must be completed and approved 120 days after start of project.
Resources	Refer to MTP clause 1.5.4.
Risk(s) and assumptions	Risk: adequacy and timeliness of the test plans Assumption: Timeliness is a primary concern because the team writing the test cases is dependent on the receipt of this the test plans
Roles and responsibilities	Refer to MTP clause 1.5.5.

8.2.2 (MTP Section 2.2) Test documentation requirements

Define the purpose, format, and content of all other testing documents that are to be used (in addition to those that are defined in MTP Section 2.4). A description of these documents may be found in Clause 9 through Clause 16. If the test effort uses test documentation or test levels different from those in this standard (i.e., component, component integration, system, and acceptance), this section needs to map the documentation and process requirements to the test documentation contents defined in this standard.

8.2.3 (MTP Section 2.3) Test administration requirements

Describe the anomaly resolution and reporting processes, task iteration policy, deviation policy, control procedures and standards, practices, and conventions. These activities are needed to administer the tests during execution.

8.2.3.1 (MTP Section 2.3.1) Anomaly resolution and reporting

Describe the method of reporting and resolving anomalies, including the standards for reporting an anomaly, the Anomaly Report distribution list, and the authority and time line for resolving anomalies. This section of the plan defines the anomaly criticality levels. Classification for software anomalies may be found in IEEE Std 1044™-1993 [B13].

8.2.3.2 (MTP Section 2.3.2) Task iteration policy

Describe the criteria used to determine the extent to which a testing task is repeated when its input is changed or task procedure is changed (e.g., reexecuting tests after anomalies have been fixed). These criteria may include assessments of change, integrity level, and effects on budget, schedule, and quality.

8.2.3.3 (MTP Section 2.3.3) Deviation policy

Describe the procedures and criteria used to deviate from the MTP and level test documentation after they are developed. The information required for deviations includes task identification, rationale, and effect on system/software quality. Identify the authorities responsible for approving deviations.

8.2.3.4 (MTP Section 2.3.4) Control procedures

Identify control procedures applied to the test activities. These procedures describe how the software-based system and software products and test results will be configured, protected, and stored.

These procedures may describe quality assurance, configuration management, data management, or other activities if they are not addressed by other efforts. Describe how the test activities comply with existing security provisions and how the test results are to be protected from unauthorized alterations.

8.2.3.5 (MTP Section 2.3.5) Standards, practices, and conventions

Identify the standards, practices, and conventions that govern the performance of testing tasks including, but not limited to, internal organizational standards, practices, and policies.

8.2.4 (MTP Section 2.4) Test reporting requirements

Specify the purpose, content, format, recipients, and timing of all test reports. Test reporting consists of Test Logs (Clause 13), Anomaly Reports (Clause 14), Level Interim Test Status Report(s) (Clause 15), Level Test Report(s) (Clause 16), and the Master Test Report (Clause 17). Test reporting may also include optional reports defined by the user of this standard. The format and grouping of the optional reports are user defined and will vary according to subject matter.

8.3 (MTP Section 3) General

Introduce the following subordinate sections. This section includes the glossary of terms and acronyms. It also describes the frequency and the process by which the MTP is changed and baselined. It may also contain a change-page containing the history of the changes (date, reason for change, and who initiated the change).

8.3.1 (MTP section 3.1) Glossary

Provide an alphabetical list of terms that may require definition for the users of the MTP with their corresponding definitions. This includes acronyms. There may also be a reference to a project glossary, possibly posted online.

8.3.2 (MTP section 3.2) Document change procedures and history

Specify the means for identifying, approving, implementing, and recording changes to the MTP. This may be recorded in an overall configuration management system that is documented in a Configuration Management Plan that is referenced here. The change procedures need to include a log of all of the changes that have occurred since the inception of the MTP. This may include a Document ID (every testing document should have a unique ID connected to the system project), version number (sequential starting with first approved version), description of document changes, reason for changes (e.g., audit comments, team review, system changes), name of person making changes, and role of person to document (e.g., document author, project manager, system owner). This information is commonly put on an early page in the document (after the title page and before Section 1). Some organizations put this information at the end of the document.

9. Level Test Plan(s)

Specify for each LTP the scope, approach, resources, and schedule of the testing activities for its specified level of testing. Identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the associated risk(s). In the title of the plan, the word "Level" is replaced by the organization's name for the particular level being documented by the plan (e.g., Component Test Plan, Component Integration Test Plan, System Test Plan, and Acceptance Test Plan).

In most projects, there are different test levels requiring different resources, methods, and environments. As a result, each level is best described in a separate plan. Different Level Test Plans may require different usage of the documentation content topics listed below. Some examples of test levels for the development activity to undertake (from IEEE/EIA Std 12207.0-1996 [B21]) are as follows:

— Each software unit (IEEE Std 1008™-1987 [B9]) and database.

— Integrated units (IEEE Std 1008-1987 [B9]) and components.

— Tests for each software requirement.

— Software qualification testing for all requirements.

— Systems integration: aggregates of other software configuration items, hardware, manual operations, and other systems. It is not unusual for large systems to have multiple levels of integration testing.

— System qualification testing for system requirements.

Other possible examples of levels include operations, installation, maintenance, regression, and nonfunctional levels such as security, usability, performance, stress, and recovery. Any one of the example levels may be more than one level for an organization; e.g., Acceptance testing may be two levels: Supplier's System and User's Acceptance test levels.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 9.1 through 9.4. A full example of an LTP outline is shown in the boxed text.

Level Test Plan Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

1.4. Level in the overall sequence

1.5. Test classes and overall test conditions

2. Details for this level of test plan

2.1 Test items and their identifiers

2.2 Test Traceability Matrix

2.3 Features to be tested

2.4 Features not to be tested

2.5 Approach

2.6 Item pass/fail criteria

2.7 Suspension criteria and resumption requirements

2.8 Test deliverables

3. Test management

3.1 Planned activities and tasks; test progression

3.2 Environment/infrastructure

3.3 Responsibilities and authority

3.4 Interfaces among the parties involved

3.5 Resources and their allocation

3.6 Training

3.7 Schedules, estimates, and costs

3.8 Risk(s) and contingency(s)

4. General

4.1 Quality assurance procedures

4.2 Metrics

4.3 Test coverage

4.4 Glossary

4.5 Document change procedures and history

9.1 (LTP Section 1) Introduction

Introduce the following subordinate sections. This section identifies the document and puts it in context of the test effort and the project-specific lifecycle. This section also identifies the types of tests and test conditions for the specific level of testing.

9.1.1 (LTP Section 1.1) Document identifier

See 8.1.1.

9.1.2 (LTP Section 1.2) Scope

Summarize the software product or system items and features to be tested by this particular level of test. The need for each item and its history may be included. This section may be a reference to a portion of the MTP, be an addition to the MTP, or reflect changes from the MTP.

9.1.3 (LTP Section 1.3) References

See 8.1.3.

9.1.4 (LTP Section 1.4) Level in the overall sequence

Show this level's context in the overall test hierarchy or sequence. This is best supported by an illustration. It may be combined with LTP Section 1.2, Scope (see 9.1.2).

9.1.5 (LTP Section 1.5) Test classes and overall test conditions

Summarize the unique nature of this particular level of test. This is additional detail within the basic scope defined in Section 9.1.2. For example, provide descriptions for the particular level such as follows:

a) Component test would focus on the desired attributes (e.g., logic) of each (one at a time, in groups, or for all) component

b) Each level of integration test would have an inventory of interfaces to exercise

c) System test would focus on meeting the system's requirements

d) Acceptance test would focus on the attributes of fitness for use

Some examples of possible classes within one or more levels are as follows:

— Positive (or valid) testing of input values that should be processed successfully

— Negative (or invalid) values that should NOT process but do provide an appropriate error processing, such as an error notification message to a user

— All boundary values, including those just above, just below, and just on each limit

— Normal values based on usage profiles

— Exploratory tests based on compatibility requirements with prior version

9.2 (LTP Section 2) Details for this level of test plan

Introduce the following subordinate sections. This section describes the specific items to be tested at the designated level and provides a Test Traceability Matrix that links the items to be tested with the requirements. It is in this section that the approach is described along with the pass/fail criteria and suspension/resumption criteria, and test deliverables are identified.

9.2.1 (LTP Section 2.1) Test items and their identifiers

Identify the test items (software or system) that are the object of testing, e.g., specific attributes of the software, the installation instructions, the user instructions, interfacing hardware, database conversion software that is not a part of the operational system) including their version/revision level. Also identify any procedures for their transfer from other environments to the test environment.

Supply references to the test item documentation relevant to an individual level of test, if it exists, such as follows:

- Requirements
- Design
- User's guide
- Operations guide
- Installation guide

Reference any Anomaly Reports relating to the test items.

Identify any items that are to be specifically excluded from testing.

9.2.2 (LTP Section 2.2) Test Traceability Matrix

Provide a list of the requirements (software and/or system; may be a table or a database) that are being exercised by this level of test and show the corresponding test cases or procedures. The requirements may be software product or software-based system functions or nonfunctional requirements for the higher levels of test, or design or coding standards for the lower levels of test. This matrix may be part of a larger Requirements Traceability Matrix (RTM) referenced by this plan that includes requirements for all levels of test and traces to multiple levels of life cycle documentation products. It may include both forward and backward tracing. The RTM may be referenced by Section 4.3 coverage.

9.2.3 (LTP Section 2.3) Features to be tested

Identify all software product or software-based system features and combinations of software or system features to be tested.

9.2.4 (LTP Section 2.4) Features not to be tested

Identify all features and known significant combinations of features that will not be tested and the rationale for exclusion.

9.2.5 (LTP Section 2.5) Approach

Describe the overall approach for the level of testing. For each major feature or group of features, specify the approach that will ensure that they are adequately tested. The approach may be described in sufficient detail to permit identification of the major testing tasks and estimation of the time required to do each one.

Features to be tested (LTP Section 2.3), features not to be tested (LTP Section 2.4), and approaches (LTP Section 2.5) are commonly combined in a table called a Test Matrix. It contains a unique identifier for each requirement for the test (e.g., system and/or software requirements, design, or code), an indication of the source of the requirement (e.g., a paragraph number in the source document), and a summary of the requirement and an identification of one or more generic method(s) of test. Some examples of possible methods are as follows:

— Black box: The test inputs can be generated and the outputs captured and completely evaluated from the outside of a test item; i.e., test cases are developed from the test item specification, only without looking at the code or design.

— White box: Considers the internal structure of the software (e.g., attempts to reach all of the code). Commonly requires some kind of test support software.

— Analysis: Just viewing the outputs cannot confirm that the test executed successfully; some kind of additional computations, simulations, studies, and so on will be required.

— Inspection: This is a static test; the code or documentation is read and examined without being executed.

The Test Matrix may be combined with the Test Traceability Matrix. The Test Traceability Matrix links each requirement with one or more test cases. The test coverage requirements (LTP Section 4.4) may reference or be combined with this section.

9.2.6 (LTP Section 2.6) Item pass/fail criteria

Specify the criteria to be used to determine whether each test item has passed or failed testing. This is commonly based on the number of anomalies found in specific severity category(s). For example, require that there are no category 1 or 2 anomalies remaining.

9.2.7 (LTP Section 2.7) Suspension criteria and resumption requirements

Specify the criteria used to suspend all or a portion of the testing activity on the test items associated with this plan. Specify the testing activities that must be repeated when testing is resumed.

9.2.8 (LTP Section 2.8) Test deliverables

Identify all information that is to be delivered by the test activity (documents, data, etc.). The following documents may be included:

- Level Test Plan(s)
- Level Test Design(s)
- Level Test Cases
- Level Test Procedures
- Level Test Logs
- Anomaly Reports
- Level Interim Test Status Report(s)
- Level Test Report(s)
- Master Test Report

Test input data and test output data may be identified as deliverables. Test tools may also be included. If documents have been combined or eliminated, then this list will be modified accordingly.

Describe the process of delivering the completed information to the individuals (preferably by position, not name) and organizational entities that will need it. This may be a reference to a Configuration Management Plan. This delivery process description is not required if it is covered by the MTP and there are no changes.

9.3 (LTP Section 3) Test management

Introduce the following subordinate sections. This section describes the test activities and tasks for the specified level and the progression of these. It is here that the infrastructure, responsibilities and authority, organizational interfaces, resources, training, schedules, and risk(s) are identified if they are not identified or described in a higher level document such as the MTP.

9.3.1 (LTP Section 3.1) Planned activities and tasks; test progression

Identify the set of tasks necessary to prepare for and perform testing. Identify all inter-task dependencies. Identify any significant constraints such as test item availability, testing resource availability, and deadlines. It may be desirable to combine all of the documentation content topics about resources (LTP Sections 3.1 through 3.8) into one section. This content topic and the next one (LTP Sections 3.1 and 3.2) could be combined in a chart showing entry criteria, person responsible, task, and exit criteria.

9.3.2 (LTP Section 3.2) Environment/infrastructure

Specify both the necessary and the desired properties of the test environment and any relevant test data. This may include the physical characteristics of the facilities, including the hardware, the off-the-shelf software, the test support tools and databases, personnel (identifying their organizations as appropriate), and anything else needed to support the test. It includes the environment for setup before the testing, execution during the testing (including data capture), and any post-testing activities (e.g., data reduction and analysis). Also specify the level of security that must be provided for, and any safety issues related to, the testing facilities, software, and any proprietary components. It may include externally provided content topics (possibly provided by third parties) including systems and/or subsystems. Identify the source(s) for all of these needs.

9.3.3 (LTP Section 3.3) Responsibilities and authority

Identify the individuals or groups responsible for managing, designing, preparing, executing, witnessing, and checking results of this level of testing, and for resolving the anomalies found. In addition, identify those persons responsible for providing the test items identified in LTP Section 2 and the environmental needs identified in LTP Section 3.2.

The responsible parties may include the developers, testers, operations staff, user representatives, technical support staff, data administration staff, and quality support staff. They may be participating either full or part time. They may have primary or secondary responsibilities.

9.3.4 (LTP Section 3.4) Interfaces among the parties involved

Describe the means and the contents of communication between the individuals and groups identified in LTP Section 3.5. A figure that illustrates the flow of information and data may be included.

9.3.5 (LTP Section 3.5) Resources and their allocation

Delineate any additional required resources that are not already documented by other parts of the plan (test environment needs are in LTP Section 3.2, personnel resources are in LTP Section 3.5, and schedule needs are in LTP Section 3.7). This includes both internal and external resources (such as outside test resources, e.g., test labs, outsourcing, etc.).

9.3.6 (LTP Section 3.6) Training

Specify test training needs by skill level. Identify training options for providing necessary skills. Training can be varied, including options such as traditional classroom training, self-paced computer-based training, training over the Internet, visiting the future user site, and mentoring by more knowledgeable staff members.

9.3.7 (LTP Section 3.7) Schedules, estimates, and costs

Include test milestones identified in the software or system project schedule as well as all test item transmittal events.

Define any additional test milestones needed. Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing resource (i.e., facilities, tools, and staff), specify its periods of use.

9.3.8 (LTP Section 3.8) Risk(s) and contingency(s)

Identify the risk issues that may adversely impact successful completion of the planned level testing activities. Specify potential impact(s) of each risk, with contingency plan(s) for mitigating or avoiding the risk. The risk(s) and contingency(s) that are current at the time of signoff of the first document release may change as the project continues, and then the risk(s) and contingency(s) can be tracked in a separate document (risk register) that is not under signoff control.

LTP Section 3.8 may be a reference to a master project plan, or it may supplement it with a greater level of detail. LTP Section 3.8 may be combined with LTP Section 3.1, planned activities, and tasks.

9.4 (LTP Section 4) General

Introduce the following subordinate sections. This section describes the QA procedures and metrics. It also contains the glossary and a description of the frequency and process by which the document is revised and re-baselined. It may also contain a change-page containing the history of the changes (date, reason for change, and who initiated the change).

9.4.1 (LTP Section 4.1) Quality assurance procedures

Identify the means by which the quality of testing processes and products will be assured. Include or reference anomaly tracking and resolution procedures. The quality assurance information may be described in a Quality Assurance Plan or Standard Procedure that can be referenced.

9.4.2 (LTP Section 4.2) Metrics

Identify the specific measures that will be collected, analyzed, and reported. The metrics specified here are those that only apply to this particular test level (the global metrics are described in MTP Section 1.5.6). This may be a reference to where it is documented in a Quality Assurance Plan or as a part of documentation in an overall measurement program.

9.4.3 (LTP Section 4.3) Test coverage

Specify the requirement(s) for test coverage. Test coverage is an indication of the degree to which the test item has been reached or “covered” by the test cases, including both the breadth and depth. The type of coverage that is relevant varies by the level of test. For example, unit (IEEE Std 1008-1987 [B9]) test coverage is often expressed in terms of percentage of code tested, and software or system validation test coverage can be a percentage of requirements tested. There is a need for specification of coverage or some other method for ensuring sufficiency of testing.

9.4.4 (LTP Section 4.4) Glossary

See 8.3.1.

9.4.5 (LTP Section 4.5) Document change procedures and history

See 8.3.2.

10. Level Test Design

The purpose of the LTD is to specify any refinements of the test approach (LTP Section 2.5) and to identify the features to be tested by this design and its associated tests.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 10.1 through 10.3. A full example of an LTD outline is shown in the boxed text.

Level Test Design Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details of the Level Test Design

2.1. Features to be tested

2.2. Approach refinements

2.3. Test identification

2.4. Feature pass/fail criteria

2.5 Test deliverables

3. General

3.1. Glossary

3.2. Document change procedures and history

10.1 (LTD Section 1) Introduction

Introduce the following subordinate sections. This section identifies the issuing organization and the details of issuance. It includes required approvals and status (DRAFT/FINAL) of the document. It is here that the scope is described and references identified.

10.1.1 (LTD Section 1.1) Document identifier

See 8.1.1.

10.1.2 (LTD Section 1.2) Scope

See 9.1.2.

10.1.3 (LTD Section 1.3) References

See 8.1.3.

10.2 (LTD Section 2) Details of the Level Test Design

Introduce the following subordinate sections. This section describes the features to be tested and any refinements to the test approach as required for the level. It also identifies the sets of test cases (highest level test cases) or scenarios along with the pass/fail criteria. It may also include the test deliverables.

10.2.1 (LTD Section 2.1) Features to be tested

Identify the test items and describe the features and combinations of features that are the object of this LTD. Other features that may be exercised but that are not the specific object of this LTD need not be identified (e.g., a database management system that is supporting the reports that are being tested). The LTD provides more detailed information than the Level Test Plan. For example, identify an overall test architecture of all test scenarios, the individual scenarios, and the detailed test objectives within each scenario.

For each feature or feature combination, a reference to its associated requirements in the item requirement and/or design description may be included. This may be documented in a Test Traceability Matrix (LTP Section 2.2).

10.2.2 (LTD Section 2.2) Approach refinements

Specify refinements to the approach described in the corresponding Level Test Plan (if there is one; otherwise specify the entire approach). Include specific test techniques to be used. The method of analyzing test results should be identified (e.g., comparator tools, visual inspection, etc.).

Summarize the common attributes of any test cases. This may include input constraints that must be true for every input in a set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies. Sets of associated test cases may be identified as scenarios (also commonly called scripts or suites). Test scenarios should be designed to be as reusable as possible for regression testing, revalidation testing for changes, and training new employees who must either use or support the system over time.

10.2.3 (LTD Section 2.3) Test identification

List the identifier and a brief description of each test case (or set of related test cases) in scenarios for this design. A particular test case, scenario, or procedure may be identified in more than one LTD. List the identifier and a brief description of each procedure associated with this LTD.

10.2.4 (LTD Section 2.4) Feature pass/fail criteria

Specify the criteria to be used to determine whether the feature or feature combination has passed or failed. This is commonly based on the number of anomalies found in each severity category(s). This section is not needed if it is covered by an MTP and there have been no subsequent changes to the criteria.

10.2.5 (LTD Section 2.5) Test deliverables

See 9.2.8.

10.3 (LTD Section 3) General

Introduce the following subordinate sections. This section includes the Glossary and document change procedures.

10.3.1 (LTD Section 3.1) Glossary

See 8.3.1.

10.3.2 (LTD Section 3.2) Document change procedures and history

See 8.3.2.

11. Level Test Case

The purpose of the LTC is to define (to an appropriate level of detail) the information needed as it pertains to inputs to and outputs from the software or software-based system being tested. The LTC includes all test case(s) identified by the associated segment of the LTD (if there is one).

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 11.1 through 11.3. A full example of an LTC outline is shown in the boxed text. Section 1 and 3 are specified once per document. The Test Case details in Sections 2 are documented once per Test Case.

Level Test Case Outline (full example)
1. Introduction (once per document)
1.1. Document identifier
1.2. Scope
1.3. References
1.4. Context
1.5. Notation for description
2. Details (once per test case)
2.1. Test case identifier
2.2. Objective
2.3. Inputs
2.4. Outcome(s)
2.5. Environmental needs
2.6. Special procedural requirements
2.7. Intercase dependencies
3. Global (once per document)
3.1. Glossary
3.2. Document change procedures and history

Since a test case may be referenced by several Level Test Designs used by different groups over a long time period, enough specific information must be included in the test case to permit reuse.

11.1 (LTC Section 1) Introduction

See 10.1.

11.1.1 (LTC Section 1.1) Document identifier

See 8.1.1.

11.1.2 (LTC Section 1.2) Scope

See 9.1.2.

11.1.3 (LTC Section 1.3) References

See 9.1.3.

11.1.4 (LTC Section 1.4) Context

Provide any required context that is not already covered by other sections of this document (e.g., third-party testing via the Internet).

11.1.5 (LTC Section 1.5) Notation for description

Define any numbering schemes, e.g., for scenarios and test cases. The intent of this section is to explain any such schema.

11.2 (LTC Section 2) Details of the Level Test Case

Introduce the following subordinate sections. Describe each test case and its unique identifier, objectives, outcomes, environmental needs, and special procedures.

11.2.1 (LTC Section 2.1) Test case identifier

Describe the unique identifier needed by each test case so that it can be distinguished from all other test cases. An automated tool may control the generation of the identifiers.

11.2.2 (LTC Section 2.2) Objective

Identify and briefly describe the special focus or objective for the test case or a series of test cases. This is much more detailed than the Test Items in the Level Test Plan. It may include the risk or priority for this particular test case or series of test cases.

11.2.3 (LTC Section 2.3) Inputs

Specify each input required to execute each test case. Some inputs will be specified by value (with tolerances where appropriate), whereas others, such as constant tables or transaction files, will be specified by name. Identify all appropriate databases, files, terminal messages, memory resident areas, and values passed by the operating system.

Specify all required relationships between inputs (e.g., timing).

11.2.4 (LTC Section 2.4) Outcome(s)

Specify all outputs and the expected behavior (e.g., response time) required of the test items. Provide the exact value(s) (with tolerances where appropriate) for each required output and expected behavior. This section is not required for self-validating tests.

11.2.5 (LTC Section 2.5) Environmental needs

Describe the test environment needed for test setup, execution, and results recording. This section is commonly documented per scenario or group of scenarios. It may be illustrated with one or more figures showing all of the components and where they interact. This section is only needed if it provides more information than the Level Test Plan or if there have been changes since the Level Test Plan was developed.

11.2.5.1 Hardware

Specify the characteristics and configuration(s) of the hardware required to execute this test case.

11.2.5.2 Software

Specify all software configuration(s) required to execute this test case. This may include system software such as operating systems, compilers, simulators, and test tools. In addition, the test item may interact with application software.

11.2.5.3 Other

Specify any other requirements not yet included [e.g., unique facility needs, specially trained personnel, and third-party provided environment(s)] if there are any.

11.2.6 (LTC Section 2.6) Special procedural requirements

Describe any special constraints on the Level Test Procedures that execute this test case such as pre- and post-conditions and/or processing. This section may reference the use of automated test tools. This section provides exceptions and/or additions to the Level Test Procedures, not a repeat of any of the information contained in the procedures.

11.2.7 (LTC Section 2.7) Intercase dependencies

List the identifiers of test cases that must be executed prior to this test case. Summarize the nature of the dependencies. If test cases are documented (in a tool or otherwise) in the order in which they need to be executed, the Intercase Dependencies for most or all of the cases may not be needed.

11.3 (LTC Section 3) General

See 10.3.

11.3.1 (LTC Section 3.1) Glossary

See 8.3.1.

11.3.2 (LTC Section 3.2) Document change procedures and history

See 8.3.2.

12. Level Test Procedure

The purpose of an LTPr is to specify the steps for executing a set of test cases or, more generally, the steps used to exercise a software product or software-based system item in order to evaluate a set of features.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 12.1 through 12.3. A full example of the LTPr outline is shown in the boxed text.

Level Test Procedure Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

1.4. Relationship to other procedures

2. Details

2.1. Inputs, outputs, and special requirements

2.2. Ordered description of the steps to be taken to execute the test cases

3. General

3.1. Glossary

3.2. Document change procedures and history

12.1 (LTPr Section 1) Introduction

See 10.1.

12.1.1 (LTPr Section 1.1) Document identifier

See 8.1.1.

12.1.2 (LTPr Section 1.2) Scope

Describe the scope of this procedure in terms of both its particular focus and its role supporting the implementation of the LTC and/or other test documentation (e.g., LTD, LTP, or MTP).

12.1.3 (LTPr Section 1.3) References

See 8.1.3 of this standard. Include references to relevant sections of any applicable test item documentation (e.g., references to usage procedures). If this procedure executes test case(s), provide reference(s) for all of them.

12.1.4 (LTPr Section 1.4) Relationship to other procedures

Describe any requirements this procedure may have for other procedures. Some examples of requirements for other test procedures include that they execute:

— Before this one

— Concurrently with this one

— Subsequent to this one

12.2 (LTPr Section 2) Details of the Level Test Procedure

Introduce the following subordinate sections. This section includes the inputs and outputs as well as the ordered description of the test steps required to execute each test case.

12.2.1 (LTPr Section 2.1) Inputs, outputs, and special requirements

Identify all that is needed to execute the tests, including but not limited to test cases, databases, automated tools, and external and/or third-party systems.

Identify any special requirements that are necessary for the execution of this procedure. These may include prerequisite procedures, special skill requirements, and special environmental requirements.

12.2.2 (LTPr Section 2.2) Ordered description of the steps to be taken by each participant

Include the activities below (as applicable) for each procedure; there may be one or multiple procedures in one Level Test Procedure document. Include the degree to which the procedure steps can be varied and the process for determining the allowable degree of variation (if variance is allowed).

- Log: List any tools or methods for logging (the results of test execution, any anomalies observed, and any other events pertinent to the test).
- Setup: Provide the sequence of actions necessary to prepare for execution of the procedure.
- Start: Provide the actions necessary to begin execution of the procedure.
- Proceed: Provide any actions necessary during execution of the procedure.
- Measurement: Describe how the test measurements will be made.
- Shut down: Describe the actions necessary to temporarily suspend testing, when unscheduled events dictate.
- Restart: Describe any procedural restart points and the actions necessary to restart the procedure at each of these points.
- Stop: Provide the actions necessary to bring execution to an orderly halt.
- Wrap-up: Provide the actions necessary after the execution of the procedure has been completed (including termination of logging).
- Contingencies: Provide the actions necessary to deal with anomalies that may occur during execution.

12.3 (LTPr Section 3) General

See 10.3.

12.3.1 (LTPr Section 3.1) Glossary

See 8.3.1.

12.3.2 (LTPr Section 3.2) Document change procedures and history

See 8.3.2.

13. Level Test Log

The purpose of the LTL is to provide a chronological record of relevant details about the execution of tests. An automated tool may capture all or part of this information.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 13.1 through 13.3. A full example of an LTL outline is shown in the boxed text.

Level Test Log Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details

2.1. Description

2.2. Activity and event entries

3. General

3.1. Glossary

13.1 (LTL Section 1) Introduction

See 10.1.

13.1.1 (LTL Section 1.1) Document identifier

See 8.1.1. The log ID needs to correspond with the procedure ID in order to clarify which tests created the results logged.

13.1.2 Scope (LTL Section 1.2)

Describe the scope of this log both in terms of its particular focus and its role supporting the implementation of the LTPr and/or other test documentation (e.g., LTC, LTD, LTP, or MTP).

13.1.3 (LTL Section 1.3) References

See 8.1.3.

13.2 (LTL Section 2) Details of the Level Test Log

Introduce the following subordinate sections. This section includes the essence of the test log, especially the information regarding each entry.

13.2.1 (LTL Section 2.1) Description

Provide any general information that applies to all entries in the log (exceptions can be specifically noted in a log entry). The following information may be considered:

- Identify the items being tested including their version/revision levels
- Identify any changes from the prior testing documents to the attributes of the environments in which the testing is conducted
- Date and time of start and stop
- Name of the individual running the test
- Any issue that causes testing to halt

13.2.2 (LTL Section 2.2) Activity and event entries

Record activities/events for each relevant detail, including the beginning and end of activities, and the occurrence date and time along with the identity of the author.

13.2.2.1 Execution description

Record the identifier of the Level Test Procedure being executed. Record all personnel participating in the execution including testers, support personnel, and observers. Also indicate the role of each individual.

13.2.2.2 Procedure results

For each execution, create a record of the results (manually or automated by a tool). Record the success or failure of each test case.

13.2.2.3 Environmental information

Record any changes in the test environment that are deviations from the plans.

13.2.2.4 Anomalous events

Record what happened before and after an unexpected event occurred. Record all circumstances surrounding the inability to begin execution of a Test Procedure or failure to complete a Level Test Procedure. If this information is recorded in an Anomaly Report, it is not also recorded here.

13.2.2.5 Anomaly Report identifiers

Record the identifier of each test Anomaly Report, whenever one is opened.

13.3 (LTL Section 3) General

See 10.3.

13.3.1 (LTL Section 6) Glossary

See 8.3.1.

14. Anomaly Report

The purpose of the AR is to document any event that occurs during the testing process that requires investigation. This may be called a problem, test incident, defect, trouble, issue, anomaly, or error report.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 14.1 through 14.3. A full example of an AR outline is shown in the boxed text.

Anomaly Report Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details

2.1. Summary

2.2. Date anomaly discovered

2.3. Context

2.4. Description of anomaly

2.5. Impact

2.6. Originator's assessment of urgency (see IEEE 1044-1993 [B13])

2.7. Description of the corrective action

2.8. Status of the anomaly

2.9. Conclusions and recommendations

3. General

3.1 Document change procedures and history

14.1 (AR Section 1) Introduction

See 10.1. This may include a title for the AR, if one is desired.

14.1.1 (AR Section 1.1) Document identifier

See 8.1.1.

14.1.2 (AR Section 1.2) Scope

Briefly describe any contextual information not covered elsewhere in the AR that is needed to make this AR understandable.

14.1.3 (AR Section 1.3) References

See 8.1.3.

14.2 (AR Section 2) Details of the Anomaly Report

Introduce the following subordinate sections. This section identifies the items contained in the AR including its status and corrective actions taken.

14.2.1 (AR Section 2.1) Summary

Summarize the anomaly.

14.2.2 (AR Section 2.2) Date anomaly discovered

Record the date (and possibly also the time) that the anomaly was first identified.

14.2.3 (AR Section 2.3) Context

Identify the software or system item (including any version numbers), or software or system configuration item, and/or the software or system life cycle process in which the anomaly was observed. Identify the test items involved indicating their version/revision level. References to the appropriate Test Procedure, Test Case, and Test Log may be supplied. Identify the level of test.

14.2.4 (AR Section 2.4) Description of the anomaly

Provide a description of the anomaly. Indicate whether the anomaly is reproducible, and provide enough information to make it reproducible if it is. This description may include (if not already covered by the referenced information) the following items:

- Inputs
- Expected results
- Actual results
- Unexpected outcomes
- Procedure step
- Environment

— Attempts to repeat

— Testers

— Observers

Related activities and observations that may help to isolate and correct the cause of the anomaly may be included (e.g., describe any test case executions that might have a bearing on this particular anomaly and any variations from the published Test Procedure).

14.2.5 (AR Section 2.5) Impact

Indicate (if known) the depth and breath of the impact this anomaly will have on technical and business issues (e.g., test documentation, development documentation, user's ability to perform tasks, and system operations). Identify the existence of any known workarounds. This may include an estimate of the time, effort, and risk to fix the defect (completed by the development organization after the Anomaly Report is established).

14.2.6 (AR Section 2.6) Originator's assessment of urgency

Provide an evaluation of the need for an immediate repair. See IEEE Std 1044-1993 [B13] for suggested categories. Most organizations have from three to five categories, where the most serious category means that the product is unusable, and the least serious is a cosmetic anomaly. Include any relevant risk analysis and conclusions about risk.

14.2.7 (AR Section 2.7) Description of the corrective action

Summarize the activities during the corrective action taken to resolve the reported anomaly. It may include the time, effort, and risk required for the fix(es), with the actual time and effort added after the fix is completed. The corrective action may be deferral or retirement of a duplicate.

14.2.8 (AR Section 2.8) Status of the anomaly

Identify the current status of the anomaly. A common sequence might be as follows: open, approved for resolution, assigned for resolution, fixed, and retested with the fix confirmed.

14.2.9 (AR Section 2.9) Conclusions and recommendations

Specify any recommendations for changes to the development and/or testing processes and documentation that would help to prevent this kind of anomaly in the future. This may include identification of the source or injection point of the anomaly.

14.3 (AR Section 3) General

See 10.3.

14.3.1 (AR Section 3.1) Document change procedures and history

See 8.3.2.

15. Level Interim Test Status Report

The purpose of the LITSR is to summarize the results of the designated testing activities and optionally to provide evaluations and recommendations based on these results. It is customary to replace the word "Level" in the title of the document with the organization's name for the particular test level, e.g., Acceptance Interim Test Status Report. There is one defined format for the LITSR for each test level identified by the organization. They may vary greatly in the level of detail.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 15.1 through 15.3. A full example of an LITSR outline is shown in the boxed text.

Level Interim Test Status Report Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details

2.1. Test status summary

2.2. Changes from plans

2.3. Test status metrics

3. General

3.1. Document change procedures and history

15.1 (LITSR Section 1) Introduction

See 10.1.

15.1.1 (LITSR Section 1.1) Document identifier

See 8.1.1.

15.1.2 (LITSR Section 1.2) Scope

Specify the contents and organization of this document. Include references to any information captured in automated tools and not contained in this document.

15.1.3 (LITSR Section 1.3) References

See 8.1.3.

15.2 (LITSR Section 2) Details of the Level Interim Test Status Report

Introduce the following subordinate sections. This section describes the test status summary, changes from the test plan, and test status metrics.

15.2.1 (LITSR Section 2.1) Test status summary

Summarize the testing status results compared to the planned tests to be run. Summarize the actual versus the desired status of the Anomaly Reports for this level of test, e.g., opened, in the process of being resolved, and resolved.

15.2.2 (LITSR Section 2.2) Changes from plans

Summarize tests that were planned to be executed (as of the date of the LITSR) that were not yet executed, and why. Provide any changes to the remaining tests, e.g., schedule changes, tests that will no longer be executed, and tests that will be re-executed.

15.2.3 (LITSR Section 2.3) Test status metrics

Provide the interim status metrics as delineated in the Level Test Plan. Describe any discrepancies from stated goals.

15.3 (LITSR Section 3) General

See 10.3.

15.3.1 (LITSR Section 3.1) Document change procedures and history

See 8.3.2.

16. Level Test Report (LTR)

The purpose of the Level Test Report (LTR) is to summarize the results of the designated testing activities and to provide evaluations and recommendations based on these results. It is customary to replace the word “Level” in the title of the document with the organization’s name for the particular test level, e.g., Acceptance Test Report. There is one LTR for each test level defined by the organization or project. Small projects may merge reports for multiple levels. They may vary greatly in the level of detail of documentation (e.g., a Unit Test Report may simply be a statement that it passed or failed, whereas an Acceptance Test Report may be much more detailed).

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 16.1 through 16.3. A full example of an LTR outline is shown in the boxed text.

Level Test Report Outline (full example)
1. Introduction
1.1. Document identifier
1.2. Scope
1.3. References
2. Details
2.1. Overview of test results
2.2. Detailed test results
2.3. Rationale for decisions
2.4. Conclusions and recommendations
3. General
3.1. Glossary
3.2. Document change procedures and history

16.1 (LTR Section 1) Introduction

See 10.1.

16.1.1 (LTR Section 1.1) Document identifier

See 8.1.1.

16.1.2 (LTR Section 1.2) Scope

Specify the contents and organization of this document. Include references to any information captured in automated tools and not contained in this document.

16.1.3 (LTR Section 1.3) References

See 8.3.1. For each test item, supply references to the following documents if they exist: Master Test Plan, Level Test Plan, Level Test Design, Level Test Cases, Level Test Procedures, Level Test Logs, and Anomaly Reports.

16.2 (LTR Section 2) Details of the Level Test Report

Introduce the following subordinate sections. This section provides an overview of the test results, all of the detailed test results, rationale for all decisions, and the final conclusions and recommendations.

16.2.1 (LTR Section 2.1) Overview of test results

Summarize the evaluation of the test items. Identify the items tested, indicating their version/revision level. Indicate the environment in which the testing activities took place, and its impact (if any).

16.2.2 (LTR Section 2.2) Detailed test results

Summarize the results of testing. Identify all resolved anomalies and summarize their resolutions (or reference where that information is available). Identify all unresolved anomalies. If anomalies are deferred, explain (or reference) the process for handling deferrals.

Summarize the major testing activities and events. Summarize the relevant metrics collected.

Report any variances of the test items from their specifications. Indicate any variances from the test documentation (e.g., test changes or tests not executed). Specify the reason for each variance (or specified group(s) of variances), or reference where it is recorded.

Evaluate the comprehensiveness of the testing process (e.g., coverage metrics, if specified) against the comprehensiveness criteria specified in the Level Test Plan, if the plan exists.

16.2.3 (LTR Section 2.3) Rationale for decisions

Specify the issues that were considered for any decisions and the reason(s) for the selection of the conclusion(s).

16.2.4 (LTR Section 2.4) Conclusions and recommendations

Specify an overall evaluation of each test item, including its limitations. This evaluation will be based on the test results and on the item level pass/fail criteria. An estimate of failure risk may be included. Recommend its status relative to availability for production use, and under what circumstances (e.g., immediately, with a specified subset of anomalies resolved, or never). This may include identification of anomaly clusters in functionality and/or anomaly root cause analysis.

16.3 (LTR Section 3) General

See 10.3.

16.3.1 (LTR Section 3.1) Glossary

See 8.3.1.

16.3.2 (LTR Section 3.2) Document change procedures and history

See 8.3.2.

17. Master Test Report

The purpose of the MTR is to summarize the results of the levels of the designated testing activities and to provide evaluations based on these results. This report may be used by any organization using the MTP. Whenever an MTP is generated and implemented, there needs to be a corresponding MTR that describes the results of the MTP implementation.

Clause 7 specifies how to address the content topics shown in the outline example below. Details on the content for each topic are contained in 17.1 through 17.3. A full example of an MTR outline is shown in the boxed text.

Master Test Report Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details of the Master Test Report

2.1. Overview of all aggregate test results

2.2. Rationale for decisions

2.3. Conclusions and recommendations

3. General

3.1. Glossary

3.2. Document change procedures and history

17.1 (MTR Section 1) Introduction

See 10.1.

17.1.1 (MTR Section 1.1) Document identifier

See 8.1.1.

17.1.2 (MTR Section 1.2) Scope

Specify the contents and organization of this document. Include references to any information captured in automated tools and not contained in this document.

17.1.3 (MTR Section 1.3) References

See 8.1.3. For each test level included, provide references to the Level Test Reports.

17.2 (MTR Section 2) Details of the Master Test Report

Introduce the following subordinate sections. This section describes the overview of all aggregate test results, rational for any decisions, and the final conclusions and recommendations.

17.2.1 (MTR Section 2.1) Overview of all aggregate test results

Summarize an evaluation of the test levels, including the following information:

- Summary of testing activities: Provide an executive-level summary of all test activities performed in support of this release, increment, or version. The activities identified in this section of the report should correspond to the test activities described in the MTP.
- Summary of testing task results: Provide an executive-level summary of all testing tasks performed in support of this release, increment, or version. The tasks identified in this section of the report should correspond to the test tasks described in the MTP.
- Summary of anomalies and resolutions: Provide a categorized summary of anomalies discovered during testing performed in support of this release, increment, or version. Provide separate summaries for those anomalies that are resolved and those that remain unresolved.
- Assessment of release/increment quality: Provide an overall assessment of the quality of the products delivered as part of this release, increment, or version. Include justification for the assessment.
- Summaries of final metrics collected.

17.2.2 (MTR Section 2.2) Rationale for decisions

Specify the reasons for the pass/fail/conditional-pass conclusions for the software or software-based system. A conditional-pass is a pass qualified by some kind of restrictions (e.g., limits on the use or range of operating conditions) or based on the resolution of a specified subset of anomalies.

17.2.3 (MTR Section 2.3) Conclusions and recommendations

Specify an overall evaluation of the software product or software-based system. An estimate of failure risk may be included. Recommend status of the software or software-based system relative to suitability for production use, and under what circumstances (e.g., immediately, with a specified subset of anomalies resolved, or never). Offer conclusions/recommendations concerning acceptance of the product delivered as in this release, increment, or version. Conclusions include a summary of the effort to ascertain the readiness of the product for release. Recommendations include an indication as to the belief that the product is ready for release, ready for release pending minor modification, or not ready for release.

Describe any lessons learned and resulting changes in practice that were discovered during the conduct of this test. This section of the report may include process improvements that were identified during the implementation of the MTP and incorporated into the management of the test effort. This may include identification of anomaly clusters in functionality or anomaly root cause analysis.

If anomalies are deferred, explain (or reference) the process for handling deferrals.

17.3 (MTR Section 3) General

See 10.3.

17.3.1 (MTR Section 3.1) Glossary

See 8.3.1.

17.3.2 (MTR Section 3.2) Document change procedures and history

See 8.3.2.

Annex A

(informative)

Bibliography

[B1] EIA/IEEE J-STD 016™-1995, EIA/IEEE Trial-Use Standard for Information Technology Software Life Cycle Processes Software Development Acquirer-Supplier Agreement (withdrawn).[5]

[B2] IEEE 100™, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition. New York, Institute of Electrical and Electronics Engineers, Inc.[6, 7]

[B3] IEEE Std 610.12™-1990, IEEE Standard Glossary of Software Engineering Terminology.

[B4] IEEE Std 730™-2002, IEEE Standard for Software Quality Assurance Plans.

[B5] IEEE Std 828™-2005, IEEE Standard for Software Configuration Management Plans.

[B6] IEEE Std 830™-1998, IEEE Recommended Practice for Software Requirements Specifications.

[B7] IEEE Std 982.1™-2005, IEEE Standard Dictionary of Measures of the Software Aspects of Dependability.

[B8] IEEE Std 982.2™-1988, IEEE Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software.

[B9] IEEE Std 1008™-1987, IEEE Standard for Software Unit Testing.

[B10] IEEE Std 1012™-2004, IEEE Standard for Software Verification and Validation.

[B11] IEEE Std 1016™-1998, IEEE Recommended Practice for Software Design Descriptions.

[B12] IEEE Std 1028™-1997, IEEE Standard for Software Reviews.

[B13] IEEE Std 1044™-1993, IEEE Standard for Classification for Software Anomalies.

[B14] IEEE Std 1058™-1998, IEEE Standard for Software Project Management Plans.

[5] EIA/IEEE Std J-STD-016-1995 has been withdrawn; however, copies can be obtained from Global Engineering, 15 Inverness Way East, Englewood, CO 80112-5704, USA, tel. (303) 792-2181 (http://global.ihs.com/).
[6] IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards/ieee.org/).
[7] The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

[B15] IEEE Std 1061™-1998, IEEE Standard for a Software Quality Metrics Methodology.

[B16] IEEE Std 1062™-1998, IEEE Recommended Practice for Software Acquisition.

[B17] IEEE Std 1074™-2006, IEEE Standard for Developing a Software Project Life Cycle Process.

[B18] IEEE Std 1219™-1998, IEEE Standard for Software Maintenance.

[B19] IEEE Std 1233™-1998, IEEE Guide for Developing System Requirements Specifications.

[B20] IEEE Std 1362™-1998, IEEE Guide for Information Technology—System Definition—Concept of Operations (ConOps) Document.

[B21] IEEE/EIA Std 12207.0™-1996, IEEE and EIA Joint Standard Development, Industry Implementation of International Standard ISO/IEC 12207-1995, Standard for Information Technology —Software Life Cycle Processes.

[B22] IEEE/EIA Std 12207.1™-1997, Industry implementation of International Standard ISO/IEC 12207:1995. (ISO/IEC 12207) Standard for Information Technology—Software Life Cycle Processes—Life Cycle Data.

[B23] IEEE/EIA Std 12207.2™-1997, Information Technology—Implementation Considerations.[8]

[B24] ISO/IEC 12207:1995, Information Technology—Software Life Cycle Processes.[9]

[B25] ISO/IEC 12119:1994, Information technology—Software packages—Quality requirements and testing.

[B26] ISO/IEC DIS 15026:1996, Information technology—System and software integrity levels.

[8]This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

[9]ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (http://www.iso.ch/). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (http://global.ihs.com/). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

Annex B

(informative)

Example integrity level scheme

Table B.1 defines four integrity levels used as an illustration by this standard. Table B.2 describes the consequences of failures for each of the four software integrity levels. Table B.3 illustrates how the concepts of integrity and risk are used to identify an integrity level for a software-based system, software product, or service. Table B.3 shows overlaps among the levels to allow for individual interpretations of acceptable risk. An integrity level 0 (zero) may be assigned if there are no consequences associated with a failure that may occur in the software-based system, software product, or service.

Table B.1—Description of integrity levels

Integrity level	Description
4	A failure in a function or system feature causes **catastrophic** consequences to the system (including consequences to users, the environment, etc.) with reasonable, probable, or occasional likelihood of occurrence of an operating state that contributes to the error.
3	A failure in a function or system feature causes **critical** consequences with reasonable, probable, or occasional likelihood of occurrence of an operating state that contributes to the error.
2	A failure in a function or system feature causes **marginal** consequences with reasonable, probable, or occasional likelihood of occurrence of an operating state that contributes to the error.
1	A failure in a function or system feature causes **negligible** consequences with reasonable, probable, occasional, or infrequent likelihood of occurrence of an operating state that contributes to the error.

Table B.2—Definitions of consequences of failures

Consequence	Definitions
Catastrophic	Loss of human life, complete mission failure, loss of system security and safety, or extensive financial or social loss.
Critical	Major and permanent injury, partial loss of mission, major system damage, or major financial or social loss.
Marginal	Moderate injury or illness, degradation of secondary mission, or moderate financial or social loss.
Negligible	Minor injury or illness, minor impact on system performance, or operator inconvenience.

Table B.3 illustrates the risk-based scheme shown in Table B.1 and Table B.2. Each cell in the table describes a software integrity level based on the combination of an error consequence and the likelihood of failure of an operating state that contributes to the error. Some table cells reflect more than one software integrity level indicating that the final assignment of the software integrity level can be selected in response to the system application and risk mitigation recommendations. For some industry applications, the definition of likelihood of occurrence categories may be expressed as probability figures derived by analysis or from system requirements.

Table B.3—Risk assessment scheme

	Likelihood of occurrence of an operating state that contributes to error			
Consequence	**Likely**	**Probable**	**Occasional**	**Unlikely**
Catastrophic	4	4	4 or 3	3
Critical	4	4 or 3	3	2 or 1
Marginal	3	3 or 2	2 or 1	1
Negligible	2	2 or 1	1	1

Annex C

(informative)

Testing tasks

Table C.1 provides the recommended minimum test activities and tasks supporting the above processes (including their inputs and outputs). The inputs and outputs include the recommended corresponding test documentation (contents are defined in Clause 8 through Clause 17). The activity numbers in the title contain the section of 5.1 where the details of those test activities are provided. The Inputs and Outputs that are in bold are documents that are defined in this standard.

Table C.1—Testing tasks, inputs, and outputs

5.1.1 Testing tasks during the management process: Test management activity		
Testing tasks	**Inputs**[10]	**Outputs**
(1) Generate Master Test Plan (MTP) a) Generate MTP for all life cycle processes. b) Establish a baseline MTP prior to the Requirements Activity. c) Identify project milestones in the MTP. d) Schedule testing tasks. The MTP will be updated throughout the development life cycle.	IEEE Std 829 **Master Test Plan** (with any previous updates, if there is one already) Identified integrity levels [5.3.1, Task (4)] Master schedule Concept documents (e.g., Statement of Need, System Requirements, and Business Rules)	**Master Test Plan** and its updates
(2) Perform Management Review of Test Effort a) Review and summarize the test effort to define changes to testing tasks or to redirect the test effort. b) Recommend whether to proceed to the next set of testing tasks and provide Anomaly Reports and Level Test Reports. c) Verify that all testing tasks comply with task requirements defined in the MTP. d) Verify that testing task results have a basis of evidence supporting the results.	**Master Test Plan** **Anomaly Reports** Development plans and schedules Developer products	Risk and **Anomaly Reports** Input to **Master Test Report** Updated **Master Test Plan** **Level Test Reports**

[10]Other inputs may be used. For any test activity and task, all of the required inputs and outputs from preceding activities and tasks may be used, but for conciseness, only the primary inputs are listed.

5.1.1 Testing tasks during the management process: Test management activity		
Testing tasks	**Inputs[10]**	**Outputs**
(3) Provide Management Review and Technical Review Support a) Support project management reviews and technical reviews (e.g., Preliminary Design Review and Critical Design Review) by assessing the review materials, attending the reviews, and providing Level Test Reports and Anomaly Reports. b) Assess all test results and provide recommendations for acceptance of test results, and use test results as input to the Master Test Report. c) Use results of review to note any process improvement opportunities in the conduct of test (to be used by Task (5) below). The management and technical review support may use any review methodology such as provided in IEEE Std 1028™-1997 [B12].	Development documents Product deliverables Guidelines for reviews and audits **Level Test Reports** **Anomaly Reports**	Review results reports **Anomaly Reports** Input to **Master Test Report**
(4) Interface with Organizational and Supporting Processes a) Coordinate the test effort with organizational and supporting processes. b) Identify the test data to be exchanged with these processes. c) Document the data exchange requirements in the Master Test Plan.	**Master Test Plan** Data identified from organizational and supporting processes	Updates to **Master Test Plan** Data exchange requirements
(**5) Identify Process Improvement Opportunities in the Conduct of Test** a) Gather and analyze the lessons learned. b) Gather and analyze the risk(s) identified. c) Gather and analyze the test metrics. d) Identify and analyze the deficiencies in the test process. e) Determine and implement corrective actions (e.g., repeat testing tasks or conduct a new testing task to implement the corrective action or use a different method/technique for executing a testing task). f) Monitor the efficacy of the corrective actions. g) Document findings in Master Test Report.	Tasks identified in the **Master Test Plan or other high -evel test plan** Test metrics Results of analysis **Level Test Reports** and **Master Test Report**	Changes to test processes Modification to **Master Test Plan or other high-level test plan** Input to **Master Test Report**

5.2.1 Testing tasks during the acquisition process: Acquisition support activity		
Testing tasks	**Inputs**	**Outputs**
(1) Scope Test Effort (Preliminary) a) Determine the minimum testing tasks for the identified integrity level using Table 3 and the selected integrity level scheme (see Section 5.3.1 Task 4, Integrity Level). b) Augment the minimum testing tasks with optional tasks, as necessary. c) Establish the scope of the test effort from the description of testing tasks, inputs, and outputs as defined in this table. d) Provide an estimate of the test budget, including test facilities and tools as required.	Concept documents Identified integrity level	Draft **Master Test Plan** Input to **Master Test Report**
(2) Plan Interface Between the Test Effort and Supplier (Preliminary) a) Plan the test schedule for each testing task. Identify the preliminary list of products to be verified and validated by the test process. b) Describe test access rights to proprietary and classified information. c) Incorporate the identified integrity level into the planning process.	Request for Proposal Identified integrity level Project schedule	Draft **Master Test Plan** Input to **Master Test Report**
(**3) Assess System Requirements** a) Review the system requirements to verify whether objective information that can be demonstrated by testing is provided in the requirements. b) Review other requirements such as deliverable definitions, listing of appropriate compliance standards, regulations, and user needs for testing resource and testability issues.	System requirements Statement of need User needs	**Anomaly Reports** Input to **Master Test Report**
(4) Establish Contract Criteria for Supplier/Vendor Testing (a) Identify the standards for reviews, audits, and test documentation. (b) Describe the role of the acquirer in the supplier/vendor test process. (c) Identify the minimum required test documents. (d) Identify the minimum criteria for the acceptance of the supplier's/vendor's products (software, documentation). (e) Establish schedule for supplier/vendor deliverables. (f) Establish schedule for acquirer acceptance of supplier/vendor deliverables.	Project approved Standards	Contract Criteria Input to **Master Test Report**

5.3.1 Testing tasks during the supply process: Test planning activity		
Testing tasks	**Inputs**	**Outputs**
(1) Plan Interface Between the Test Effort and the Supplier (continuing) a) Review the supplier development plans and schedules to coordinate the test effort with development activities. b) Establish procedures to exchange test data and test results with the development effort. c) Incorporate the identified integrity level into the planning process.	Master schedule Developer schedule Developer plans **Master Test Plan** Technical portion of Contract Identified Integrity Level Project Plan	Revised **Master Test Plan** Input to **Master Test Report** Procedures to exchange test data and test results with the development effort
(2) Scope Test Effort (continuing) a) Adopt the integrity scheme assigned. If no integrity level scheme exists, then one is selected. b) Determine the minimum testing tasks for the software-based system or software product integrity level using Table 3 and the selected integrity level scheme. c) Augment the minimum testing tasks with optional tasks (see Table D.1), as necessary. d) Establish the scope of the test effort from the description of testing tasks, inputs, and outputs as defined in this table. e) Provide an estimate of the test budget, including test facilities and tools as required.	Concept documents Identified Integrity Level	Draft **Master Test Plan** Input to **Master Test Report**
(3) Identify Metrics a) Determine the metrics that are to be collected. b) Define the metric models and counting criteria.		Input to **Master Test Plan** and **Level Test Plans** Identified metrics Input to **Master Test Report**
(4) Identify integrity level a) Determine the system or software characteristics (e.g., complexity, criticality, risk, safety level, security level, desired performance, reliability, or other unique characteristics) that define the importance of the software to the user. b) Adopt the system integrity scheme assigned. If no integrity level scheme exists, then one is selected using a reference such as described in Clause 4 and Annex B of this standard. c) Assign an integrity level to the software-based system or the software product.	Integrity level schema	Identified integrity level Input to **Master Test Report**

5.4.1 Testing tasks during the development process: Concept activity		
Testing tasks	**Inputs**	**Outputs**
(1) Review Concept Documentation a) Gain familiarity with organizational needs. b) Gain familiarity with organizational expectations regarding software-based system, software product, or service to be acquired.	Concept documentation	**Anomaly Report** Input to **Master Test Report**
(2) Review System Requirements Document a) Verify the testability of the requirements if not verified previously or if there are new system requirements.	System Requirements	**Anomaly Report** Input to **Master Test Report**
(3) Generate Test Traceability Matrix (Preliminary) a) Verify that all requirements are traceable to test cases. b) Verify that all test cases are traceable to requirements. c) Verify that there is a valid relationship among the Test Plan, Test Design, Test Cases, and Test Procedures. The Test Traceability Matrix will be updated throughout the life cycle development process.	System Requirements Software, hardware, database, network, and interface requirements, as known	Test Traceability Matrix for the **Level Test Plan(s)** (component, component integration, system, acceptance) Input to **Master Test Report**
(4) Identify integrity level a) If not identified, identify integrity level. b) Determine the system or software characteristics (e.g., complexity, criticality, risk, safety level, security level, desired performance, reliability, or other unique characteristics) that define the importance of the software to the user. c) Adopt the system integrity scheme assigned. If no integrity level scheme exists, then one is selected using a reference such as described in Clause 4 and Annex B of this standard. d) Assign an integrity level to the software-based system or the software product.	Integrity level schema	Identified integrity level Input to **Master Test Report**

5.4.2 Testing tasks during the development process: Requirements activity		
Testing tasks	**Inputs**	**Outputs**
(1) Generate Acceptance Test Plan a) Verify that the software-based system correctly enables the user to perform their tasks in an operational environment. b) Verify that the acceptance test plan complies with this standard in purpose, format, and content.	Concept documents Test Traceability Matrix	**Acceptance Test Plan** Input to **Master Test Report**

5.4.2 Testing tasks during the development process: Requirements activity		
Testing tasks	**Inputs**	**Outputs**
(2) Generate System Test Plan a) Verify that the system-based software or the software product correctly implements system requirements. c) Begin tracing of allocated requirements to test cases, procedures and results. d) Verify that the system test plan complies with the standard in purpose, format, and content.	System requirements Test Traceability Matrix	**System Test Plan** Input to **Master Test Report**
(3) Review Software Requirements and Interface Requirements for Testability. Evaluate the requirements of the software requirements and the interface requirements for testability. a) Verify that the implementation of the requirements can be successfully validated. b) Begin tracing of requirements to test cases, procedures, and results in the Test Traceability Matrix. c) Verify that there are objective criteria for validating the testability of the System Requirements and the Interface Requirements. The software requirements and interface requirements may be evaluated by using such guidance as IEEE Std 1233™-1998 [B19].	System and/or software requirements and interface requirements (SRS and IRS) **Master Test Plan**	**Anomaly Reports** Input to **Master Test Report**
(4) Identify Integrity Level (update) a) If not identified, identify integrity level. b) Determine whether software integrity levels are established for requirements, detailed functions, system components, subsystem, or other partitions as appropriate. c) Verify that the assigned integrity levels are correct. If software integrity levels are not assigned, then assign integrity levels to the system requirements. d) Document the integrity level assigned to individual software components.	Prior identified integrity level	Modified integrity level Modified **Master Test Plan** Input to **Master Test Report**
(5) Generate Test Traceability Matrix (update) b) Verify that all requirements are traceable to test cases. c) Verify that all test cases are traceable to requirements c) Verify that there is a valid relationship among the Test Plan, Test Design, Test Cases, and Test Procedures.	Test Traceability Matrix	Modified Test Traceability Matrix Input to **Master Test Report**

5.4.2 Testing tasks during the development process: Requirements activity		
Testing tasks	**Inputs**	**Outputs**
(6) Identify Risk(s) a) Identify technical and management test risk(s). b) Provide recommendations to eliminate, reduce, or mitigate the testing's identified risk(s).	System requirements Software requirements Interface requirements Supplier development plans	**Anomaly Reports** Input to **Master Test Report**
(7) Identify Security Issues a) Identify system and software security issues. b) Provide recommendations to eliminate or reduce test-identified security issues.	System requirements Software requirements Interface requirements **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**

5.4.3 Testing tasks during the development process: Design activity		
Testing tasks	**Inputs**	**Outputs**
(1) Generate Acceptance Test Design a) Continue tracing required by the acceptance test plan. b) Verify that the test design complies with this standard in purpose, format, and content.	**Acceptance Test Plan**	**Acceptance Test Design** Input to **Master Test Report**
(2) Generate System Test Design a) Continue tracing required by the system test plan. b) Verify that the test design complies with this standard in purpose, format, and content.	**System Test Plan**	**System Test Design** Input to **Master Test Report**
(3) Generate Component Integration Test Plan a) Verify that the software correctly implements the software requirements and design as the components are integrated in accordance with the system design. b) Validate that the component integration test plan complies with this standard in purpose, format, and content.	Subsystem and component requirements System design descriptions	**Component Integration Test Plan** Input to **Master Test Report**
(4) Generate Component Integration Test Design a) Continue tracing required by the component integration test plan. b) Verify that the test design complies with this standard in purpose, format, and content.	**Component Integration Test Plan**	**Component Integration Test Design** Input to **Master Test Report**

5.4.3 Testing tasks during the development process: Design activity		
Testing tasks	**Inputs**	**Outputs**
(5) Generate Component Test Plan a) Verify that software components correctly implement component requirements. b) Continue tracing of requirements from design to test cases, procedures, and results. c) Verify that the component test plan complies with this standard in purpose, format, and content.	Component requirements Detailed design description	**Component Test Plan** Input to **Master Test Report**
(6) Generate Component Test Design a) Verify that the test design implements the test plan. b) Verify that the test design complies with the defined testing document in purpose, format, and content.	**Component Test Plan**	**Component Test Design** Input to **Master Test Report**
(7) Identify Integrity Level (update) a) If not identified, identify integrity level. b) Determine whether software integrity levels are established for requirements, detailed functions, software modules, subsystem, or other software partitions. c) Verify that the assigned integrity levels are correct. If software integrity levels are not assigned, then assign integrity levels to the system requirements. d) Document the integrity level assigned to individual software components.	Prior identified integrity level	Modified integrity level Modified **Master Test Plan** Input to **Master Test Report**
(8) Generate Test Traceability Matrix (update) a) Verify that all requirements are traceable to test cases. b) Verify that all test cases are traceable to requirements. c) Verify that there is a valid relationship among the Test Plan, Test Design, Test Cases, and Test Procedures.	Test Traceability Matrix	Modified Test Traceability Matrix Input to **Master Test Report**
(9) Identify Risk(s) a) Review and update risk analysis using prior risk Anomaly Reports. b) Provide recommendations to eliminate, reduce, or mitigate the test-identified risk(s).	Software Design Interface Design Supplier development plans and schedules **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**
(10) Identify Security Issues (test) a) Identify system and software security issues. b) Provide recommendations to eliminate or reduce test-identified security issues.	Design documents Prior test-identified security issues **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**

5.4.4 Testing tasks during the development process: Implementation activity		
Testing tasks	**Inputs**	**Outputs**
(1) Generate Acceptance Test Cases a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify test cases conform to this standard in purpose, format, and content.	User documentation; use cases **Acceptance Test Plan** **Acceptance Test Design**	**Acceptance Test Cases** Input to **Master Test Report**
(2) Generate Acceptance Test Procedures a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that test procedures conform to this standard in purpose, format, and content.	User documentation **Acceptance Test Cases**	**Acceptance Test Procedures** Input to **Master Test Report**
(3) Generate System Test Cases a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that system test cases conform to this standard in purpose, format, and content.	System requirements **System Test Plan** **System Test Design**	**System Test Cases** Input to **Master Test Report**
(4) Generate System Test Procedures b) Continue tracing of requirements to test design, test cases, test procedures, and test results. c) Verify that system test procedures conform to this standard in purpose, format, and content.	**System Test Cases**	**System Test Procedures** Input to **Master Test Report**
(5) Generate Component Integration Test Cases a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that component integration test cases conform to this standard in purpose, format, and content.	Interface requirements Interface design Component Integration Test Plan **Component Integration Test Design**	**Component Integration Test Cases** Input to **Master Test Report**
(6) Generate Component Integration Test Procedures a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that component integration test procedures conform to this standard in purpose, format, and content.	**Component Integration Test Cases**	**Component Integration Test Procedures** Input to **Master Test Report**
(7) Generate Component Test Cases a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that component test cases conform to this standard in purpose, format, and content.	Software requirements **Component Test Plan** **Component Test Design**	**Component Test Cases** Input to **Master Test Report**

5.4.4 Testing tasks during the development process: Implementation activity		
Testing tasks	**Inputs**	**Outputs**
(8) Generate Component Test Procedures a) Continue tracing of requirements to test design, test cases, test procedures, and test results. b) Verify that component test procedures conform to this standard in purpose, format, and content.	**Component Test Cases**	**Component Test Procedures** Input to **Master Test Report**
(9) Execute Component Test b) Verify the test results trace to test criteria established in the test planning documents. c) Document discrepancies between actual and expected results while executing the test.	**Component Test Plan** **Component Test Design** **Component Test Cases** **Component Test Procedures** **Component test data**	Component test results **Component Interim Test Status Report** **Anomaly Reports** **Component Test Log** Input to **Component Test Report** Input to **Master Test Report**
(10) Evaluate Component Test Results a) Analyze test results to verify the software correctly implements the design and the requirements.	**Component Test Logs** Component test results **Anomaly Reports**	**Anomaly Reports** Component test results analysis Input to **Component Test Report** Input to **Master Test Report**
(11) Prepare Component Test Report a) Document the results as required by the component test plan. b) Verify the Component Test Report conforms to this standard in purpose, format, and content.	Component test results analysis **Component Test Logs** **Anomaly Reports**	**Component Test Report**
(12) Generate Test Traceability Matrix (update) a) Verify that all requirements are traceable to test cases. b) Verify that all test cases are traceable to requirements. c) Verify that there is a valid relationship among the Test Plan, Test Design, Test Cases, and Test Procedures.	Test Traceability Matrix	Modified Test Traceability Matrix Input to **Master Test Report**
(13) Perform Test Readiness Review a) Review all test status. b) Decide if ready to proceed.	All test documentation	Decision to proceed

5.4.4 Testing tasks during the development process: Implementation activity		
Testing tasks	**Inputs**	**Outputs**
(14) Identify Integrity Level (update) a) If not identified, identify integrity level. b) Determine whether software integrity levels are established for requirements, detailed functions, software modules, subsystem, or other software partitions. c) Verify that the assigned integrity levels are correct. If software integrity levels are not assigned, then assign integrity levels to the system requirements. d) Document the integrity level assigned to individual software components.	Prior identified integrity level	Modified integrity level Modified **Master Test Plan** Input to **Master Test Report**
(15) Identify Risk(s) a) Review and update test risk analysis using prior risk Anomaly Reports. b) Provide recommendations to eliminate, reduce, or mitigate the test-identified risk(s).	Component test results Supplier development plans and schedules **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**
(16) Identify Security Issues a) Identify system and software security test issues. b) Provide recommendations to eliminate or reduce test-identified security issues.	Component test results **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**

5.4.5 Testing tasks during the development process: Test activity		
Testing tasks	**Inputs**	**Outputs**
(1) Execute Component Integration Tests a) Verify the test results trace to test criteria established in the test planning documents. b) Document discrepancies between actual and expected results.	**Component Integration Test Plan** **Component Integration Test Design** **Component Integration Test Cases** **Component Integration Test Procedures** Test data	**Anomaly Reports** **Component Integration Test Log** Component Integration test results **Component Integration Interim Test Status Report** Input to **Master Test Report**
(2) Evaluate Component Integration Test Results a) Analyze test results to verify the software components are integrated correctly.	**Component Integration Test Plan** **Component Integration Test Cases** Component Integration test results **Component Integration Test Log**	**Anomaly Reports** Component Integration test results analysis Input to **Component Integration Test Report** Input to **Master Test Report**

5.4.5 Testing tasks during the development process: Test activity		
Testing tasks	**Inputs**	**Outputs**
	Anomaly Reports	
(3) Prepare Component Integration Test Report a) Prepare Component Integration Test Report. b) Document the results as required by the test planning documents. c) Verify the test report conforms to this standard in purpose, format, and content.	Component Integration test results analysis **Anomaly Reports** **Component Integration Test Log**	**Component Integration Test Report** Input to **Master Test Report**
(4) Execute System Test a) Validate test results satisfy system requirements. b) Verify test results trace to test criteria established by test traceability in the test planning documents. c) Document discrepancies between actual and expected results.	**System Test Plan** **System Test Design** **System Test Cases** **System Test Procedures** Test data	**Anomaly Reports** **System Test Log** System test results **System Interim Test Status Report** Input to **Master Test Report**
(5) Evaluate System Test Results a) Validate system requirements are satisfied. b) Verify test results satisfy test criteria established in the System Test Plan.	**System Test Plan** **System Test Logs** System test results **Anomaly Reports**	**Anomaly Reports** System test results analysis Input to **Master Test Report**
(6) Prepare System Test Report a) Prepare System Test Report. b) Document test results as required by the test planning documents.	System test results analysis **Anomaly Reports** **System Test Logs**	**System Test Report** Input to **Master Test Report**
(7) Execute Acceptance Test a) Validate User-driven acceptance test, if appropriate. b) Validate test results satisfy User Needs. c) Verify test results trace to test criteria established by test traceability in the test planning documents. Document discrepancies between actual and expected results.	**Acceptance Test Plan** **Acceptance Test Design** **Acceptance Test Cases** **Acceptance Test Procedures** User data User needs	**Anomaly Reports** **Acceptance Test Log** Acceptance test results **Acceptance Interim Test Status Report** Input to **Acceptance Test Report** Input to **Master Test Report**

5.4.5 Testing tasks during the development process: Test activity		
Testing tasks	**Inputs**	**Outputs**
8) Evaluate Acceptance Test Results a) Validate test results satisfy user needs. b) Verify test results trace to test criteria established by the test documentation. c) Verify traceability in the planning documents. d) Validate the results satisfy the acceptance criteria. e) Document discrepancies between actual and expected results.	Acceptance requirements Acceptance test results Acceptance criteria	**Anomaly Reports** Acceptance test results analysis Input to **Master Test Report**
9) Prepare Acceptance Test Report a) Document test results as required by the test planning documentation. b) Verify that test report conforms to this standard in purpose, format, and content**.**	Acceptance test results analysis **Anomaly Reports** **Acceptance Test Logs**	**Acceptance Test Report** Input to **Master Test Report**
10) Identify Risk(s) a) Review and update risk analysis using prior risk Anomaly Reports. b) Provide recommendations to eliminate, reduce, or mitigate testing identified risk(s).	Development plans and schedules **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**
11) Identify Security Issues a) Identify software-based system and software product security test issues. b) Provide recommendations to eliminate or reduce test-identified security issues.	Security requirements Anomaly Reports	**Anomaly Reports** Input to **Master Test Report**

5.4.6 Testing tasks during the development process: Installation/checkout activity		
Testing tasks	**Inputs**	**Outputs**
(1) Support Installation Configuration Audits a) Verify all products required to correctly install and operate the system or software are present in the installation package. b) Validate all site-specific parameters or conditions to verify supplied values are correct. Functional Configuration Audits and Physical Configuration Audits may be based on IEEE Std 1028-1997 [B12]	Installation Package Site dependent parameters or conditions Supplied values for all components	**Anomaly Reports** Input to **Master Test Report**
(2) Perform Installation/Checkout a) Conduct analysis or tests to verify that the installed system or software product	Installation package User documentation	**Anomaly Reports** Installation test results

5.4.6 Testing tasks during the development process: Installation/checkout activity		
Testing tasks	**Inputs**	**Outputs**
corresponds to the software-based system or software product that satisfied the acceptance test. b) Validate the software code and databases initialize, execute, and terminate as specified. c) In the transition from one version of software to the next, validate (test) that the software can be removed from the system without affecting the functionality of the remaining system components. d) Validate (test) the requirements for continuous operation and service (including user notification) during transition are satisfied.	Subset of **Acceptance Test Cases** and **Acceptance Test Procedures**	Installation Test Logs
(3) Evaluate Installation/Checkout a) Verify results trace to test criteria established by the test documentation. b) Analyze results to verify the installed software-based system or software product is the same as the one that passed the Acceptance Test. c) Document discrepancies between actual and expected results.	Installation/checkout results **Anomaly Reports** **Acceptance Test Plan** **Acceptance Test Design** **Acceptance Test Cases** **Acceptance Test Procedures** **Installation Test Logs**	Input to **Master Test Report** Installation/checkout results analysis
(4) Prepare Installation/Checkout Report a) Document results as required by the test planning documents. b) Verify that the report conforms to this standard in purpose, format, and content.	Installation/checkout results analysis **Anomaly Reports** **Installation Test Logs**	**Installation/Checkout Report** Input to **Master Test Report**
(5) Prepare Master Test Report (if required) a) Document the results of all levels of testing for the software-based system or software product as required by the Master Test Plan or other test planning document. b) Verify that the test report conforms to this standard in purpose, format, and content.	**Level Test Reports** (component, component integration, system, acceptance) **Anomaly Reports** (component, component integration, system, acceptance) **Test Logs** (component, component integration, system, acceptance)	**Master Test Report**
(6) Identify Risk(s) (Test) **a)** Review and update test risk analysis using prior risk Anomaly Reports. **b)** Provide recommendations to eliminate, reduce, or mitigate testing identified risk(s).	**Anomaly Reports** **Level Test Reports** (component, component integration, system, acceptance)	**Anomaly Reports** Input to **Master Test Report**

5.4.6 Testing tasks during the development process: Installation/checkout activity		
Testing tasks	**Inputs**	**Outputs**
(7) Identify Security Issues (Test) a) Identify software-based system and software product security issues. b) Provide recommendations to eliminate or reduce test-identified security issues.	**Level Test Reports** (component, component integration, system, acceptance) **Anomaly Reports**	**Anomaly Reports** Input to **Master Test Report**

5.5.1 Testing tasks during the operation process: Operational test activity		
Testing tasks	**Inputs**	**Outputs**
(1) Evaluate Operating Procedures a) Verify the operating procedures are consistent with the user documentation and conform to the system requirements.	Operating procedure documentation User documentation	**Anomaly Reports**
(2) Execute Operational Tests a) Validate that the system or software performs as expected in the operational environment. b) Document discrepancies between actual and expected results.	Software-based system or software product User data User documentation Selected **Acceptance Test Cases / Procedures**	**Anomaly Reports** **Operational Interim Test Status Report**
(3) Evaluate Operational Test Results a) Analyze test results to verify the system continues to meet the needs of organization. b) Validate the test results satisfy business needs specified in the concept documentation.	Concept documentation Operational test results	**Anomaly Reports** Operational test results analysis Input to **Master Test Report**
(4) Prepare Operational Test Report a) Prepare Operational Test Report. b) Document test results as required by the test planning documents.	Operational test results analysis **Anomaly Reports** **Operational Test Logs**	**Operational Test Report** Input to **Master Test Report**
(5) Identify Risk(s) a) Review and update test risk analysis using prior Risk anomalies. b) Provide recommendations to eliminate reduce or mitigate test-identified risk(s).	Proposed changes to software-based system, software product or service Supplier development plans and schedules Operational problem reports Prior **Anomaly Reports**	**Anomaly Reports**
(6) Identify Security Issues a) Identify software-based system or software product security problems. b) Provide recommendations to eliminate or reduce security test problems.	Prior **Anomaly Reports** Operational test results	**Anomaly Reports** Input to **Master Test Report**

5.6.1 Testing Tasks during the Maintenance Process: Maintenance Test Activity		
Testing tasks	**Inputs**	**Outputs**
(1) Revise all affected test documentation Revise test planning documentation to comply with approved changes to the software-based system, the software product, or the services.	**All Test Documentation** Approved changes	Updated **affected documentation**
(2) Perform anomaly evaluation a) Evaluate Operational anomalies.	**Anomaly Reports**	Anomaly evaluation results
(3) Testing task iteration a) Verify that planned changes are implemented correctly. b) Verify that documentation is complete and current. c) Verify that changes do not cause unacceptable or unintended system behaviors.	Approved changes to software-based system, software product, or services. Updated **Test Plans** (component, component integration, system, acceptance) Updated **Master Test Plan**	**Anomaly Reports**

Annex D

(informative)

Optional testing tasks

This annex provides examples of some optional additional testing tasks that can be considered for inclusion. It is not intended to include all possible optional activities.

Table D.1—Optional Testing Tasks and suggested applications

Optional testing tasks	Life cycle phases										
	Management	Acquisition	Planning	Concept	Requirements	Design	Implementation	Test	Install./ Chkout	Operation	Maintenance
Audit performance					●	●	●	●	●		●
Audit support	●				●	●	●	●	●		●
Branch coverage testing							●				
Database analysis					●	●	●	●			●
Data flow analysis					●	●	●				●
Defect history evaluation								●		●	●
Disaster recovery plan assessment	●			●	●	●	●			●	●
Independent risk assessment	●	●	●	●	●	●	●	●	●	●	●
Operational evaluation										●	
Peer reviews											
Concept				●							●
Requirements					●						●
Design						●					●
Source code							●				●
Test Plan (Master and Level)					●	●	●		●		●
Level Test Design						●	●		●		●

Optional testing tasks	Life cycle phases										
	Management	Acquisition	Planning	Concept	Requirements	Design	Implementation	Test	Install./ Chkout	Operation	Maintenance
Level Test Case						●	●	●	●		●
Level Test Procedure						●	●	●	●		●
Performance Monitoring				●	●	●	●	●	●	●	●
Post Installation Validation									●	●	●
Proposal Evaluation Support		●									
Qualification Testing								●	●		●
Regression Analysis and Testing					●	●	●	●	●		●
Sizing and Timing Analysis					●	●	●	●			●
Test Certification								●	●	●	●
Test Documentation Review (Supplier's)							●	●			
Test Evaluation					●	●	●	●	●	●	●
Test Tool Plan Generation	●	●	●								●
Test Witnessing (Supplier's)								●	●	●	●
Training Document Evaluation					●	●	●	●	●	●	●
User Documentation Evaluation	●			●	●	●	●	●	●	●	●
User Training	●							●	●	●	●
Volume Testing								●	●	●	●

Audit performance. Provide an independent assessment of whether a software process and its products conform to applicable regulations, standards, plans, procedures, specifications, and guidelines. Audits may be applied to any software process or product at any development stage. The supplier may initiate audits, as well as the acquirer, the developer, or other involved party such as a regulatory agency. The initiator of the audit selects the audit team and determines the degree of independence required. The initiator of the audit and the audit team leader establish the purpose, scope, plan, and reporting requirements for the audit.

The auditors collect sufficient evidence to decide whether the software processes and products meet the evaluation criteria. They identify major deviations, assess risk to quality, schedule, and cost, and

then they report their findings. Examples of processes that could be audited include configuration management practices, use of software tools, and degree of integration of the various software engineering disciplines particularly in developing an architecture, security issues, training, and project management.

Audit support. Provide technical expertise to the auditors on request. The audit support resources may represent the acquirer at audit proceedings and may assist in the test of remedial activities identified by the audit.

Branch coverage testing. Testing designed to execute each outcome of each decision point in a computer program (IEEE Std 828-1998 [B5]).

Database analysis. Evaluation of database design as part of a design review process could include:

- Physical limitations analysis. Identify the physical limitations of the database such as maximum number of records, maximum record length, largest numeric value, smallest numeric value, and maximum array length in a data structure and compare them with designed values.
- Index versus storage analysis. Analyze the use of multiple indexes compared with the volume of stored data to determine whether the proposed approach meets the requirements for data retrieval performance and size constraints.
- Data structures analysis. Some database management systems have specific data structures within a record such as arrays, tables, and date formats. Review the use of these structures for potential impact on requirements for data storage and retrieval.
- Backup and disaster recovery analysis. Review the methods employed for backup against the requirements for data recovery and system disaster recovery and identify deficiencies.

Data flow analysis. Evaluation of data flow diagrams as part of a design review process could include:

- Symbology consistency check. The various methods used to depict data flow diagrams employ very specific symbology to represent the actions performed. Verify that each symbol is used consistently.
- Flow balancing. Compare the output data from each process to the data inputs and the data derived within the process to ensure the data is available when required. This process does not specifically examine timing or sequence considerations.
- Confirmation of derived data. Examine the data derived within a process for correctness and format. Data designed to be entered into a process by operator action should be confirmed to ensure availability.
- Keys to index comparison. Compare the data keys used to retrieve data from data stores within a process to the database index design to confirm that no invalid keys have been used and the uniqueness properties are consistent.

Defect history evaluation. Evaluate all the anomalies identified in the system at each selected development stage (e.g., component, component integration, system, and acceptance). Evaluation may include examining defects prior to acceptance as part of input to the test readiness review and prior to operations as part of the operational readiness review. Evaluations of the defect history provide the testing process with an account of critical areas of the process and product where problems reside as well as areas that are troublesome and/or routinely cause inconsistent test results. Defect history also may be reviewed during the preparation of the Master Test Report.

Disaster recovery plan assessment. Verify that the disaster recovery plan is adequate to restore critical operation of the system in the case of an extended system outage. The disaster recovery plan should include:

- Identification of the disaster recovery team and a contact list
- Recovery operation procedures
- Procedure for establishing an alternative site, including voice and data communications, mail, and support equipment
- Plans for replacement of computer equipment
- Establishment of a system backup schedule
- Procedures for off-site storage and retrieval of software, data, documentation, and vital records
- Logistics of moving staff, data, documentation, etc.

Independent risk assessment. An objective evaluator assesses the current status of identified risk conditions.

Operational evaluation. Assess the deployment readiness and operational readiness of the software. Operational evaluation may include examining the results of operational tests, audit reviews, and Anomaly Reports. This evaluation verifies that the software is as follows:

- At a suitable point of correctness for mass production of that software
- Valid and correct for site-specific configurations

Peer review. Review the software products to detect defects in the specified work products at each selected development activity to assure the quality of the emerging software. The peer review process may consist of multiple steps such as follows:

- Peer review planning
- Product overview
- Peer review preparation
- Examination meeting
- Defect rework
- Resolution follow-up

A peer review is performed by a small team of peers (not their management) and includes, but usually is not led by, the author. They are commonly implemented as inspections or walkthroughs (see IEEE Std 1028-1997 [B12]). The peer review team usually consists of three to six persons, and in some cases, it includes personnel from the test group, quality assurance, or verification and validation. The participants assume specific roles to find, classify, report, and analyze defects in the product. Each type of peer review is specifically defined by its intended purpose, required entry criteria, defect classification, checklists, exit criteria, designated participants, and its preparation and examination procedures. Peer reviews do not debate engineering judgments, suggest corrections, or educate project members; they report apparent anomalies and verify their resolution by the author. Peer reviews, particularly walkthroughs, are a great source of education and cross-training.

Peer review (concept). Verify that the system architecture and requirements satisfy customer needs. Verify that the system requirements are complete and correct, and that anomalies such as omissions, defects, and ambiguities in the requirements are detected.

Peer review (requirements). Verify that the requirements satisfy customer needs, can be implemented, and are complete, traceable, testable, and consistent and that anomalies such as omissions, defects, and ambiguities in the requirements are detected and corrected.

Peer review (design). Verify that the design can be implemented, is traceable to the requirements, and that all interface and procedural logic is complete and correct, and that anomalies such as omissions, defects, and ambiguities in the design are detected and corrected.

Peer review—test plan (component, component integration, system, acceptance). Verify that the scope, strategy, resources, and schedule of the (component, component integration, system, acceptance) testing process have been completely and accurately specified, that all items to be tested and all required tasks to be performed have been defined, and ensure that all personnel and resources necessary to perform the testing have been identified.

Peer review—test design (component, component integration, system, acceptance). Verify that the (component, component integration, system, and acceptance) test design is consistent with the test plan, and that the test design is correct, complete, and readable.

Peer review (source code). Verify that the source code implementation is traceable to the design, and that all interfaces and procedural logic are complete and correct, and that anomalies such as omissions, defects, and ambiguities in the source code are detected and corrected.

Peer review—test cases (component, component integration, system, acceptance). Verify that the (component, component integration, system, and acceptance) test cases are consistent with the test plan, that the set of test cases is complete, and that all test cases are correct.

Peer review—test procedures (component, component integration, system, acceptance). Verify that the standard was implemented with complete, correct, and consistent documentation.

Performance monitoring. Collect information on the performance of software under operational conditions. Determine whether system and software performance requirements are satisfied. Performance monitoring is a continual process and includes evaluation of items such as follows:

- Database transaction rates to determine the need to reorganize or re-index the database
- CPU performance monitoring for load balancing
- Direct access storage utilization
- Network traffic to ensure adequate bandwidth
- Critical outputs of a system (e.g., scheduled frequency, expected range of values, scheduled system reports, and reports of events)

Performance testing. Verify the behavior of a system under predetermined load. Load can be expressed in terms such as follows:

- Number of concurrent connections
- Number of concurrent requests
- Number of processed transactions per time interval

Post-installation validation. Execute a reference benchmark or periodic test for critical software when reliability is crucial or there is a possibility of software corruption. By automatically or manually comparing results with the established benchmark results, the system can be validated prior to each execution of the software. When pre-use benchmark testing is impractical, such as for real-time, process control, and emergency-use software, a periodic test, conducted at a predetermined interval, can be used to ensure continued reliability

Proposal evaluation support. Examine all appropriate proposal response submissions for conformance to both the stated testing requirements and the intent behind the test-related requirements.

Qualification testing. Verify that all software requirements are tested according to qualification testing requirements demonstrating the feasibility of the software for operation and maintenance. Conduct as necessary any tests to verify and validate the correctness, accuracy, and completeness of the qualification testing results. Document the qualification test results together with the expected qualification test results**.** Planning for qualification testing may begin during the requirements activity.

Regression analysis and testing. Determine the extent of analyses and tests that must be repeated when changes are made to any previously examined software products. Assess the nature of the change to determine potential ripple or side effects and impacts on other aspects of the system. Rerun test cases based on changes, error corrections, and impact assessment to detect errors spawned by software modifications.

Sizing and timing analysis. Collect and analyze data about the software functions and resource utilization to determine whether system and software requirements for speed and capacity are satisfied. The types of software functions and resource utilization issues include, but are not limited to:

- CPU load
- Random access memory and secondary storage (e.g., disk and tape) utilization
- Network speed and capacity
- Input and output speed

Sizing and timing analysis is started during requirements and iterated through maintenance.

Test certification. Certify the test results by verifying that the tests were conducted using baselined requirements, a configuration control process, and repeatable tests, as well as by witnessing the tests. Certification may be accomplished at a software configuration item level or at a system level.

Test documentation review. Evaluate Level Test Plans, Test Designs, Test Cases, Test Procedures, and Test Reports (e.g., component, component integration, and system) for requirements coverage and completeness. Verify that traceability is satisfied. This may be done as a part of the peer reviews.

Test evaluation. Evaluate the tests for requirements coverage and test completeness. Assess coverage by assessing the extent of the software exercised. Assess test completeness by determining whether the set of inputs used during the test are a fair representative sample from the set of all possible inputs to the software. Assess whether test inputs include boundary condition inputs, rarely encountered inputs, and invalid inputs. For some software, it may be necessary to have a set of sequential or simultaneous inputs on one or several processors to test the software adequately. This may be done as a part of the peer reviews.

Test tool plan generation. Prepare a plan that describes the tools needed to support the test effort**.** The plan includes a description of each tool's performance, required inputs and associated tools, outputs generated, need date, and cost of tool purchase or development. The tool plan should also describe test

facilities and integration and system test laboratories supporting the test effort. The scope and rigor of the test effort as defined by the selected software integrity level should be considered in defining the performance required of each tool.

Test witnessing. Monitor the fidelity of test execution to the specified test procedures and witness the recording of test results. When a test failure occurs, the testing process can be continued by (1) implementing a "work around" to the failure; (2) inserting a temporary repair to the source of the failure (e.g., a code patch or a repaired test case); or (3) halting the testing process and implementing a complete repair. In all cases, assess the test continuation process for test process breakage (e.g., some software is not tested or a patch is left in place permanently), adverse impact on other tests, and loss of configuration control. Regression analysis and testing should be done for all the software affected by the test failure.

Training document evaluation. Evaluate the training materials and procedures for completeness, correctness, readability, and effectiveness. This may be done as a part of the peer reviews.

User documentation evaluation. Evaluate the user documentation for its completeness, correctness, and consistency with respect to requirements for user interface and for any functionality that can be invoked by the user. The review of the user documentation for its readability and effectiveness should include representative end users who are unfamiliar with the software. Employ the user documentation in planning an acceptance test that is representative of the operational environment. This may be done as a part of the peer reviews.

User training. Ensure that the user training includes rules that are specific to the administrative, operational and application aspects, and industry standards for that system. This training should be based on the technical documentation for users and procedures provided by the manufacturer of the system. The organization responsible for the use of the system should be responsible for providing appropriate user training.

Volume testing. Subject the software to large amounts of data to determine whether limits are reached that cause the software to fail. Volume testing also identifies the continuous maximum load or volume the system can handle for a given period of time. For example, if the software were processing a set of database records to generate a report, a Volume Test would use a large test database and check that the software behaved normally and produced the correct report.

Annex E

(informative)

Metrics From a test management perspective

Metrics related to the Master Test Plan (MTP) are those that support test management. The metrics provide information at a management-detail level. That is, the information is high level and summative. This comprehensive level of information is necessary for easily communicating timely knowledge and facts to management to support decision making associated with test planning and management.

The test management metrics, which assist in the management of the test process, are used to monitor and control test activities. The goal of the metrics is to measure and assess attributes such as resources, task completion, cost, and incidents/anomalies.

Additionally the user may need to provide storage (e.g., location of repository or data source) for the metrics, data resources (e.g., who collects, enters, validates, and provides data), and tools for collection/extraction (e.g., software) for each type of metric.

The metrics are used to generate a metrics report. The metrics report describes how often data are collected for the metric; how often the report is generated; outline metric delivery (e.g., hard copy, on-line electronic); identify who receives the report or has access to the metric; and specify any restrictions on access to the metric and the approval process for additions and deletions to this access and to the standard distribution.

Clause 7 specifies how to address the content topics shown in the outline example below. The user of this standard may adopt any format and section numbering system for the metrics report.

Master Test Plan Metrics Report Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

2. Details of the MTP Metrics Report

2.1. Introduction

2.2. Metric

2.3. Threshold (if applicable)

2.4. Pictorial representation

2.5. Notes

4. General

3.1 Glossary

3.2 Document change procedure and history

E.1 (Metrics Report Section 1) Introduction

Introduce the initial topics in this document.

E.1.1 (MTP Metrics Report Section 1.1) Document identification

See 8.1.1.

E.1.2 (MTP Metrics Report Section 1.2) Scope

Describe the purpose and goal of the report. Specify the contents and organization of the report.

E.1.3 (MTP Metrics Report Section 1.3) References

See 8.1.3.

E.2 (MTP Metrics Report Section 2) Details of the Master Test Plan

Introduce the details section of this document.

E.2.1 (MTP Metrics Report Section 2.1) Overview

List and summarize the purpose of each tracked metric described within the report.

E.2.2 (MTP Metrics Report Section 2.2) Metrics

Specify the metric and identify the reporting period.

E.2.3 (MTP Metrics Report Section 2.3) Threshold (if applicable)

For each measurement, provide the acceptable performance values.

E.2.4 (MTP Metrics Report Section 2.4) Pictorial representation

Provide a bar graph or chart for each tracked metric.

E.2.5 (MTP Metrics Report Section 2.5) Notes

Specify any supporting details that may add explanation or clarity.

E.3. (MTP metrics Report Section 3) General

Introduce the general sections of this document.

E.3.1 (MTP Metrics Report Section 3.1) Glossary

See 8.3.1.

E.3.2 (MTP Metrics Report Section 3.2) Document change procedures and history

See 8.3.2.

Annex F

(informative)

Independence

Test groups experience differing levels of independence depending on their organization. In some organizations, the same individuals who created the software or system may perform testing. In other organizations, an internal test group may perform testing. In yet another organization, a group that is external to the organization may perform testing. Occasionally, the desired degree of independence may be determined by the identified integrity level (refer to Clause 4 and Annex B).

Independence is defined by three parameters: technical independence, managerial independence, and financial independence.

Table F.1—Alternatives for test organizations

Alternative	Technical	Management	Financial
Classical	R	R	R
Modified	R	C	R
Integrated	C	R	R
Internal	C	C	C
Embedded	M	M	M
NOTE— R = Rigorous independence; C = Conditional independence; and M = Minimal independence.			

Annex G

(informative)

Examples of tailoring documentation contents

This annex provides some examples of implementation of the suggestions for tailoring contained in Clause 6.

Content topics for various testing documents that may be covered by the configuration management system for documentation are as follows:

Document	Section #	Contents
MTP	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history
LTP	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	4.6	Document change procedures and history
LTD	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history
LTC	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	1.5	Notation for description
	3.2	Document change procedures and history
LTPr	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history
LTL	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
AR	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	2.8	Status of the anomaly
	3.1	Document change procedures and history
LITSR	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history
LTR	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history
MTR	1.1	Date of issue, issuing organization, author, approvals, and status
	1.3	References
	3.2	Document change procedures and history

Below is an example of a content topic that is applicable across multiple contexts that may be published in a centralized document, perhaps on the organization's Intranet instead of being placed in individual documents (e.g., a project or organization-based glossary available in aggregate online); the section numbers for it in the testing documents are as follows:

Document	Section #	Contents
MTP	3.1	Glossary
LTP	3.1	Glossary
LTD	3.1	Glossary
LTC	3.1	Glossary
LTPr	3.1	Glossary
LTL	3.1	Glossary
LTR	3.1	Glossary
MTR	3.1	Glossary

Content topics that may be covered by the project plan include:

Document	Section #	Contents
MTP	1.5	Test overview
	2.1.1	Process: management
	1.5.2	Master test schedule
	1.5.4	Resources summary
	2.1	Test processes
LTP	2.8	Test deliverables
	3.1	Planned activities and tasks
	3.2	Responsibilities and authority
	3.4	Interfaces among the parties involved
	3.5	Resources and their allocation
	3.6	Training
	3.7	Schedules, estimates, and costs
	3.8	Risk(s) and contingency(s)
LTD	2.5	Test deliverables

Content topics that may be covered by the quality assurance documentation are as follows:

Document	Section #	Contents
LTP	4.2	Quality control procedures
	4.3	Metrics
	4.4	Test coverage

Content topics that may be covered by the development documentation include:

Document	Section #	Contents
MTP	1.4	System overview and key features
LTP	2.3	Features to be tested
LTD	2.1	Features to be tested

Some possible candidates for elimination (full or partial; information may be identified in the tool, but not fully managed) on the basis of the use of automated tools are as follows:

Document	Section #	Contents
LTP	2.2	Test Traceability Matrix
	2.3	Features to be tested
	2.4	Features not to be tested
LTD	2.1	Features to be tested
	2.3	Test identification
LTC	1.4	Context
	1.5	Notation for description
	2.1	Test case identifier
	2.2	Test items
	2.3	Inputs
	2.4	Outcome(s)
	2.5	Environmental needs
	2.6	Special procedural requirements
	2.7	Intercase dependencies
LTPr	All	
LTL	All	
AR	All	
LTR	2.2	Detailed test results

Annex H

(informative)

Guidelines for compliance with IEEE/EIA Std 12207.1-1997 [B22]

H.1 Overview

The Software and Systems Engineering Standards Committee (S2ESC) of the IEEE Computer Society has endorsed the policy of adopting international standards. In 1995, the international standard, ISO/IEC 12207:1995 [B24] was completed. The standard establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry.

In 1995, the SESC evaluated ISO/IEC 12207:1995 [B24] and decided that the standard should be adopted and serve as the basis for life cycle processes within the IEEE Software Engineering Collection. The IEEE adaptation of ISO/IEC 12207:1995 is IEEE/EIA Std 12207.0-1996 [B21]. It contains ISO/IEC 12207:1995 and the following additions: improved compliance approach, life cycle process objectives, life cycle data objectives, and errata.

The implementation of ISO/IEC 12207:1995 within the IEEE also includes the following:

— IEEE/EIA Std 12207.1-1997 [B22]

— IEEE/EIA Std 12207.2-1997 [B23]

— Additions to 11 S2ESC standards (i.e., IEEE Std 730™-2002 [B4], IEEE Std 828-1998 [B5], this standard, IEEE Std 830™-1998 [B6], IEEE Std 1012-2004 [B10], IEEE Std 1016™-1998 [B11], IEEE Std 1058™-1998 [B14], IEEE Std 1062™-1998 [B16], IEEE Std 1219™-1998 [B18], IEEE Std 1233-1998 [B19], and IEEE Std 1362-1998 [B20]) to define the correlation between the data produced by existing S2ESC standards and the data produced by the application of IEEE/EIA Std 12207.1-1997 [B22]

NOTE—Although IEEE/EIA 12207.1-1997 [B22] is a guide, it also contains provisions for application as a standard with specific compliance requirements. This annex treats IEEE/EIA Std 12207.1-1997 as a standard.

In order to achieve compliance with both this standard and IEEE/EIA Std 12207.1-1997 [B22], it is essential that the user review and satisfy the data requirements for both standards.

When this standard is directly referenced, the precedence for conformance is based on this standard alone. When this standard is referenced with the IEEE/EIA 12207.x standard series, the precedence for conformance is based on the directly referenced IEEE/EIA 12207.x standard, unless there is a statement that this standard has precedence.

H.1.1 Scope and purpose

Both this standard and IEEE/EIA Std 12207.1-1997 [B22] place requirements on test plans, test procedures, and test reports for software. The purpose of this annex is to explain the relationship between the two sets of requirements so that users producing documents intended to comply with both standards may do so.

This annex is organized as follows: The context for test plans, test procedures, and test reports is provided in H.2. Document compliance guidance for test plans is provided in H.3. Document compliance guidance for test procedures is provided in H.4. Document compliance guidance for test reports is provided in H.5.

H.2 Correlation

This clause explains the relationship between this standard and IEEE/EIA Std 12207.0-1996 [B21] in the following areas: terminology, process, and life cycle data.

H.2.1 Terminology correlation

The two standards use similar terms in similar ways. This standard discusses test plans (a Master Test Plan for all of testing and Level Test Plans—for each individual level of test), whereas IEEE/EIA Std 12207.0-1996 [B21] uses a broader term, test, or validation plan. This standard discusses Level Test Procedures, whereas IEEE/EIA Std 12207.0-1996 uses a broader term, test, or validation procedure. This standard discusses test reports (Interim Level Test Status Reports, Level Test Reports, and Master Test Report), whereas IEEE/ EIA Std 12207.0-1996 uses a broader term, test, or validation report. This standard uses the term "group" in a similar way that class is used in IEEE/EIA Std 12207.1-1997 [B22]. This standard uses the concept of level to organize multilevel plans.

H.2.2 Process correlation

IEEE/EIA Std 12207.1-1997 [B22] is based on the life cycle view of IEEE/EIA Std 12207.0-1996 [B21]. It has a strong process bias. It is particularly focused toward acquisition and has detailed process requirements. In contrast, this standard requires a process for selecting documentation content items. However, it does work within and supplement process sequences and organization of IEEE/EIA Std 12207.1-1997.

H.2.3 Life cycle data correlation—test plan

The information required in the test plans by this standard and the information required in a test plan by IEEE/EIA Std 12207.1-1997 [B22] are similar. It is reasonable to expect that the documents could comply with both standards. Details are provided in H.3 of this standard.

H.2.4 Life cycle data correlation—test procedure

The equivalent of the test procedure in IEEE/EIA Std 12207.1-1997 [B22] is accomplished by the combination of the Level Test Design, Level Test Case, and Level Test Procedures from this standard. For convenience the three will be denoted as a test procedure in this annex. The test procedure specified by this standard and the information required in a test procedure by IEEE/EIA Std 12207.1-1997 are similar. It is reasonable to expect that an equivalent collection of information could comply with both standards. Details are provided in H.4 of this standard.

H.2.5 Life cycle data correlation—test report

The equivalent of the test report in IEEE/EIA Std 12207.1-1997 [B22] is accomplished by the combination of the Level Test Logs, Anomaly Report, Interim Level Test Status Reports, Level Test Reports, and Master Test Report from this standard. For convenience the three will be denoted as a test report. The test report specified by this standard and the information required in a test report by IEEE/EIA Std 12207.1-1997 are similar. It is reasonable to expect that an equivalent collection of information could comply with both standards. Details are provided in H.5 of this standard.

H.2.6 Life cycle data correlation between other data in IEEE/EIA Std 12207.1-1997 [B22] and this standard

This clause correlates the life cycle data other than test plans, test procedures, or test reports between this standard and IEEE/EIA Std 12207.1-1997 [B22]. It provides information to users of both standards. The other life cycle data is summarized in Table H.1.

Table H.1—Summary of requirements between other data in IEEE/EIA Std 12207.1-1997 [B22] and this standard

Information item	IEEE/EIA Std 12207.0-1996 [B21] subclauses	Kind of documentation	IEEE/EIA Std 12207.1-1997 [B22] subclauses	IEEE Std 829-2008 clauses
Software integration plan	5.3.8.1, 5.3.8.5	Plan	6.18	8
System qualification test audit results record	5.3.11.3	Record		12, 13, 14, 15, 16, 17
System test and evaluation criteria record	item a) in 5.5.3.2	Record		12, 13, 14, 15, 16, 17

H.3 Document compliance—test plan

This clause provides details bearing on a claim that a test plan complying with this standard would also achieve document compliance with the test plan described in IEEE/EIA Std 12207.1-1997 [B22]. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA Std 12207.1-1997. That row is reproduced in Table H.2 of this standard.

Table H.2—Summary of requirements for a test plan excerpted from Table 1 of IEEE/EIA Std 12207.1-1997 [B22]

Information item	IEEE/EIA 12207.0-1996 [B21] subclauses	Kind of documentation	IEEE/EIA Std 12207.1-1997 [B22] subclause	References
Test or validation plan	5.3.5.5, 5.3.6.5, 5.3.6.6, 5.3.7.4, 5.3.7.5, and 6.5	Plan	6.27	IEEE Std 829-2008 EIA/IEEE J-STD 016-1995, F.2.4 [B1] ISO/IEC 12119:1994 [B25]

The requirements for document compliance are discussed in the following subclauses:

— H.3.1 discusses compliance with the information requirements noted in column 2 of Table H.2 as prescribed by 5.3.5.5, 5.3.6.5, 5.3.6.6, 5.3.7.4, 5.3.7.5, and 6.5 of IEEE/EIA Std 12207.0-1996 [B21].

— H.3.2 discusses compliance with the generic content guideline (the kind of document) noted in column 3 of Table H.2 as a plan. The generic content guidelines for a plan appear in 5.2 of IEEE/EIA Std 12207.1-1997 [B22].

— H.3.3 discusses compliance with the specific requirements for a test plan noted in column 4 of Table H.2 as prescribed by 6.27 of IEEE/EIA Std 12207.1-1997 [B22].

— H.3.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA Std 12207.0-1996 [B21] as described in 4.2 of IEEE/EIA Std 12207.1-1997 [B22].

H.3.1 Compliance with information requirements of IEEE/EIA Std 12207.0-1996 [B21]

The information requirements for a test plan are those prescribed by 5.3.5.5, 5.3.6.5, 5.3.6.6, 5.3.7.4, and 5.3.7.5 of IEEE/EIA Std 12207.0-1996. The requirements are substantively identical to those considered in H.3.3 of this standard.

H.3.2 Compliance with generic content guidelines of IEEE/EIA Std 12207.1-1997 [B22]

The generic content guidelines for a plan in IEEE/EIA Std 12207.1-1997 are prescribed by 5.2 of IEEE/EIA Std 12207.1-1997. A complying plan achieves the purpose stated in 5.2.1 and includes the information listed in 5.2.2 of that standard.

The purpose of a *plan* is as follows:

> IEEE/EIA Std 12207.1-1997, subclause 5.2.1: Purpose: Define when, how, and by whom specific activities are to be performed, including options and alternatives, as required.

A test plan complying with this standard would achieve the stated purpose.

Any *plan* complying with IEEE/EIA Std 12207.1-1997 satisfies the generic content requirements provided in 5.2.2 of that standard. Table H.3 of this standard lists the generic content items and, where appropriate, references the clause of this standard that requires the same information.

Table H.3—Coverage of generic plan requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
a) Date of issue and status	8.1.1 Date of issue, issuing organization, author, approvals, and status 10.1.1 Date of issue, issuing organization, author, approvals, and status
b) Scope	9.1.2 Scope 10.1.2 Scope
c) Issuing organization	8.1.1 Date of issue, issuing organization, author, approvals, and status 10.1.1 Date of issue, issuing organization, author, approvals, and status

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
d) References	9.1.3 References 10.1.3 References
e) Approval authority	8.1.1 Date of issue, issuing organization, author, approvals, and status 10.1.1 Date of issue, issuing organization, author, approvals, and status
f) Planned activities and tasks	8.2.1 Test processes 9.3.1 Planned activities and tasks; test progression
g) Macro references (policies or laws that give rise to the need for this plan)	9.1.3.1 External References 10.1.3.1 External References
h) Micro references (other plans or task descriptions that elaborate details of this plan)	9.1.3.2 Internal References 10.1.3 Internal References
i) Schedules	8.1.5.3 Schedule 9.3.7 Schedules, estimates, and costs
j) Estimates	8.1.5.4 Resources summary 9.3.7 Schedules, estimates, and costs
k) Resources and their allocation	8.1.5.4 Resources summary 9.3.2 Environment/infrastructure 9.3.7 Schedules, estimates, and costs
l) Responsibilities and authority	8.1.5.5 Responsibilities 9.3.3 Responsibilities and authority
m) Risk(s)	8.1.5.3 Integrity level scheme 9.4.1 Risk(s) and contingency(s)
n) Quality control measures (NOTE—This includes quality control of the test plan itself.)	8.2.1 Test processes 9.4.2 Quality assurance procedures
o) Cost	8.1.5.4 Resources summary 9.3.7 Schedules, estimates, and costs
p) Interfaces among parties involved	8.1.5.5 Responsibilities 9.3.4 Interfaces among parties involved
q) Environment/infrastructure (including safety needs)	8.1.5.4 Resources summary 9.3.2 Environment/infrastructure
r) Training	8.1.5.4 Resources summary 9.3.6 Training
s) Glossary	8.3.1 Glossary 9.4.5 Glossary

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
t) Change procedures and history	8.3.2 Change procedures and history 9.4.6 Change procedures and history

H.3.3 Compliance with specific content requirements of IEEE/EIA Std 12207.1-1997 [B22]

The specific content requirements for a test plan in IEEE/EIA Std 12207.1-1997 are prescribed by 6.27 of IEEE/EIA Std 12207.1-1997. A complying test plan achieves the purpose stated in 6.27.1 and includes the information listed in 6.27.3 of that standard.

The purpose of the *test plan* is as follows:

> IEEE/EIA 12207.1-1997, subclause 6.27.1: Purpose: Describe plans for qualification testing of software items and software systems. Describe the software test environment to be used for the testing, identify the tests to be performed, and provide schedules for test activities.

A *test plan* complying with IEEE/EIA Std 12207.1-1997 satisfies the specific content requirements provided in 6.27.3 of that standard. Table C.1 of this standard lists the specific content items and, where appropriate, references the clause of this standard that requires the same information.

H.3.4 Compliance with life cycle data objectives

In addition to the content requirements, life cycle data are managed in accordance with the objectives provided in Annex H of IEEE/EIA Std 12207.0-1996 [B21].

NOTE—The information items covered by this standard include plans and provisions for creating software life cycle data related to the basic type test data in H.4 of IEEE/EIA Std 12207.0-1996. It provides for the following test data: test strategy and criteria, cases (what to test), procedures (how to carry out tests), test results, and key decision rationale.

H.3.5 Conclusion

The analysis suggests that any test plan complying with this standard and the additions shown in Table H.3 and Table H.4 also complies with the requirements of a test or validation plan in IEEE/EIA Std 12207.1-1997 [B22]. In addition, to comply with IEEE/EIA Std 12207.1-1997, a test plan supports the life cycle data objectives in Annex H of IEEE/EIA Std 12207.0-1996 [B21].

Table H.4—Coverage of specific plan requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] specific content	Corresponding subclauses of IEEE Std 829-2008
b) Test levels	8.2.1 Test processes 9.4.1 Level of this test in the overall sequence
c) Test classes	8.2.1 Test processes 9.1.5 Test classes and overall test conditions 9.4.1 Level of this test in the overall sequence
d) General test conditions	8.2.1 Test processes 9.2.5 Approach
e) Test progression	8.2.1 Test processes 9.3.1 Planned activities and tasks; test progression
f) Data recording, reduction and analysis	8.2.1 Test processes 9.3.1 Planned activities and tasks; test progression
g) Test coverage (breadth and depth) or other methods for assuring sufficiency of testing	8.1.5.6 Tools, techniques, methods, and metrics 9.4.4 Test coverage
h) Planned tests, including items and their identifiers	8.2.1 Test processes 9.2.1 Test items and their identifiers
i) Test schedules	8.1.5.2 Master test schedule 9.3.7 Schedules, estimates, and costs
j) Requirements traceability	8.1.5.6 Tools, techniques, methods, and metrics 9.2.2 Test Traceability Matrix
k) Qualification testing environment,	8.1.5.4 Resources summary 9.3.2 Environment/infrastructure
site,	8.1.5.4 Resources summary 9.3.2 Environment/infrastructure
personnel,	8.1.5.5 Responsibilities 9.3.3 Responsibilities and authority
and participating organizations	8.1.5.5 Responsibilities 9.3.3 Responsibilities and authority

H.4 Document compliance—test procedure

This clause provides details bearing on a claim that a test procedure complying with this standard would also achieve document compliance with the test procedure described in IEEE/EIA Std 12207.1-1997 [B22]. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA 12207.1-1997. That row is reproduced in Table H.5 of this standard.

Table H.5—Summary of requirements for a test procedure excerpted from Table 1 of IEEE/EIA Std 12207.1-1997 [B22]

Information item	IEEE/EIA Std 12207.0-1996 [B21] subclauses	Kind of documentation	IEEE/EIA Std 12207.1-1997 [B22] subclause	References
Test or validation procedures	5.1.5.1, 5.3.7.1, 5.3.8.1, 5.3.8.4, 5.3.10.2, and 6.5	Procedure	6.28	IEEE Std 829-2008 EIA/IEEE J-STD-016-1995, H.2.1 [B1] ISO/IEC 12119:1994 [B25]

The requirements for document compliance are discussed in the following subclauses:

- H4.1 discusses compliance with the information requirements noted in column 2 of Table H.5 as prescribed by 5.1.5.1, 5.3.7.1, 5.3.8.1, 5.3.8.4, 5.3.10.2, and 6.5 of IEEE/EIA Std 12207.0-1996 [B21].
- H.4.2 discusses compliance with the generic content guideline (the kind of document) noted in column 3 of Table H.5 as a procedure. The generic content guidelines for a procedure appear in 5.3 of IEEE/EIA Std 12207.1-1997 [B22].
- H.4.3 discusses compliance with the specific requirements for a test procedure noted in column 4 of Table H.5 as prescribed by 6.28 of IEEE/EIA Std 12207.1-1997 [B22].
- H.4.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA Std 12207.0-1996 [B21] as described in 4.2 of IEEE/EIA Std 12207.1-1997 [B22].

H.4.1 Compliance with information requirements of IEEE/EIA Std 12207.0-1996 [B21]

The information requirements for a test procedure are those prescribed by 5.1.5.1, 5.3.7.1, 5.3.8.1, 5.3.8.4, 5.3.10.2, and 6.5 of IEEE/EIA Std 12207.0-1996. The requirements are substantively identical to those considered in H.4.3 of this standard.

H.4.2 Compliance with generic content guidelines of IEEE/EIA Std 12207.1-1997 [B22]

The generic content guidelines for a procedure in IEEE/EIA 12207.1-1997 are prescribed by 5.3 of IEEE/ EIA Std 12207.1-1997. A complying procedure achieves the purpose stated in 5.3.1 and includes the information listed in 5.3.2 of that standard.

The purpose of a *procedure* is as follows:

> IEEE/EIA 12207.1-1997, subclause 5.3.1: Purpose: Define in detail when and how to perform certain jobs, including needed tools.

A test procedure complying with this standard would achieve the stated purpose.

Any *procedure* complying with IEEE/EIA 12207.1-1997 satisfies the generic content requirements provided in 5.3.2 of that standard. Table H.6 of this standard lists the generic content items and, where appropriate, references the clause of this standard that requires the same information.

Table H.6—Coverage of generic procedure requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
a) Date of issue and status	10.1.1 Date of issue, issuing organization, author, approvals, and status 11.1.1 Date of issue, issuing organization, author, approvals, and status 12.1.1 Date of issue, issuing organization, author, approvals, and status
b) Scope	10.1.2 Scope 11.1.2 Scope 12.1.2 Scope
c) Issuing organization	10.1.1 Date of issue, issuing organization, author, approvals, and status 11.1.1 Date of issue, issuing organization, author, approvals, and status 12.1.1 Date of issue, issuing organization, author, approvals, and status
d) References	10.1.3 References 11.1.3 References 12.1.3 References
e) Approving authority	10.1.1 Date of issue, issuing organization, author, approvals, and status 11.1.1 Date of issue, issuing organization, author, approvals, and status 12.1.1 Date of issue, issuing organization, author, approvals, and status
f) Relationship to other procedures	10.1.2 Scope 11.1.2 Scope 12.1.2 Scope
g) Macro references (policies or laws that give rise to the need for this procedure)	10.1.3.1 External references 11.1.3.1 External references 12.1.3.1 External references
h) Micro references (other plans or task descriptions that elaborate details of this procedure)	10.3.2 Internal references 11.3.2 Internal references 12.3.2 Internal references
i) Inputs and outputs	11.2.3 Inputs 11.2.4 Outcome(s)
j) Ordered description of the steps to be taken by each participant	12.2.2 Ordered description of the steps to be taken to execute the test cases
k) Glossary	10.3.1 Glossary 11.3.1 Glossary 12.3.1 Glossary
l) Change history	10.3.2 Document change procedures and history 11.3.2 Document change procedures and history 12.3.2 Document change procedures and history

H.4.3 Compliance with specific content requirements of IEEE/EIA Std 12207.1-1997 [B22]

The specific content requirements for a test procedure in IEEE/EIA Std 12207.1-1997 are prescribed by 6.28 of IEEE/EIA Std 12207.1-1997. A complying test procedure achieves the purpose stated in 6.28.1 and includes the information listed in 6.28.3 of that standard.

The purpose of the *test procedure* is as follows:

> IEEE/EIA 12207.1-1997, subclause 6.28.1: Purpose: Describe 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. Enable the acquirer to assess the adequacy of the qualification testing to be performed.

A *test procedure* complying with IEEE/EIA Std 12207.1-1997 satisfies the specific content requirements provided in 6.28.3 of that standard. Table H.7 of this standard lists the specific content items and, where appropriate, references the clause of this standard that requires the same information.

Table H.7—Coverage of specific test procedure requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] specific content	Corresponding subclauses of IEEE Std 829-2008
a) Generic procedure information	See Table H.5
b) Identification of test author	10.1.1 Date of issue, issuing organization, author, approvals, and status 11.1.1 Date of issue, issuing organization, author, approvals, and status 12.1.1 Date of issue, issuing organization, author, approvals, and status
c) Identification of test configuration	11.2.5 Environmental needs
d) Test objectives,	11.2.2 Objective
test requirements...	10.2.1 Features to be tested
and test rationale	10.2.2 Approach refinements
e) Test preparations (hardware, software, other) for each test	11.2.5 Environmental needs 11.2.6 Special procedural requirements
f) Test descriptions	11.2.2 Objective
(i) Test identifier	11.2.1 Test case identifier
(ii) Requirements addressed	10.2.1 Features to be tested
(iii) Prerequisite conditions	11.2.6 Special procedural requirements 11.2.7 Intercase dependencies
(iv) Test input	11.2.3 Inputs
(v) Expected test results	11.2.4 Outcome(s)
(vi) Criteria for evaluating results	10.2.4 Feature pass/fail criteria 11.2.4 Outcome(s)
(vii) Instructions for conducting procedure	12.2.2 Ordered description of the steps to be taken to execute the test cases
g) Requirements traceability	10.2.1 Features to be tested
h) Rationale for decisions	10.2.2 Approach refinements

H.4.4 Compliance with life cycle data objectives

In addition to the content requirements, life cycle data are to be managed in accordance with the objectives provided in Annex H of IEEE/EIA Std 12207.0-1996 [B21].

NOTE—The information items covered by this standard include plans and provisions for creating software life cycle data related to the basic type test data in H.4 of IEEE/EIA Std 12207.0-1996. It provides for the following test data: test strategy and criteria, cases (what to test), procedures (how to carry out tests), test results, and key decision rationale.

H.4.5 Conclusion

The analysis suggests that any test procedure complying with this standard and the additions shown in Table H.6 and Table H.7 also complies with the requirements of a test or validation procedure in IEEE/EIA Std 12207.1-1997 [B22]. In addition, to comply with IEEE/EIA Std 12207.1-1997, a test procedure supports the life cycle data objectives in Annex H of IEEE/EIA Std 12207.0-1996 [B21].

H.5 Document Compliance Test report

This clause provides details bearing on a claim that a test report complying with this standard would also achieve document compliance with the test report described in IEEE/EIA Std 12207.1-1997 [B22]. The requirements for document compliance are summarized in a single row of Table 1 of IEEE/EIA Std 12207.1-1997. That row is reproduced in Table H.8 of this standard.

Table H.8—Summary of requirements for a test report excerpted from Table 1 of IEEE/EIA Std 12207.1-1997 [B22]

Information item	IEEE/EIA 12207.0-1996 [B21] subclauses	Kind of documentation	IEEE/EIA 12207.1-1997 [B22] subclause	References
Test or validation results report	5.3.7.2, 5.3.8.2, 5.3.9.1, 5.3.10.1, 5.3.11.1, 5.3.13.1, 6.5	Report	6.29	IEEE Std 829-2008 EIA/IEEE J-STD-016-1995, H.2.2 [B1] ISO/IEC 12119:1994 [B25]

The requirements for document compliance are discussed in the following subclauses:

- H.5.1 discusses compliance with the information requirements noted in column 2 of Table H.8 as prescribed by 5.3.7.2, 5.3.8.2, 5.3.9.1, 5.3.10.1, 5.3.11.1, 5.3.13.1, and 6.5 of IEEE/EIA Std 12207.0-1996 [B21].
- H.5.2 discusses compliance with the generic content guideline (the kind of document) noted in column 3 of Table H.8 as a report. The generic content guidelines for a report appear in 5.5 of IEEE/EIA Std 12207.1-1997 [B22].
- H.5.3 discusses compliance with the specific requirements for a test report noted in column 4 of Table H.8 as prescribed by 6.29 of IEEE/EIA Std 12207.1-1997 [B22].
- H.5.4 discusses compliance with the life cycle data objectives of Annex H of IEEE/EIA Std 12207.0-1996 [B21] as described in 4.2 of IEEE/EIA Std 12207.1-1997 [B22].

H.5.1 Compliance with information requirements of IEEE/EIA Std 12207.0-1996 [B21]

The information requirements for a test report are those prescribed by 5.3.7.2, 5.3.8.2, 5.3.8.5, 5.3.9.1, 5.3.10.1, 5.3.11.1, 5.3.13.1, and 6.5 of IEEE/EIA Std 12207.0-1996. The requirements are substantively identical to those considered in H.5.3 of this standard.

H.5.2 Compliance with generic content guidelines of IEEE/EIA Std 12207.1-1997 [B22]

The generic content guidelines for a report in IEEE/EIA Std 12207.1-1997 are prescribed by 5.5 of IEEE/EIA Std 12207.1-1997. A complying report achieves the purpose stated in 5.5.1 and includes the information listed in 5.5.2 of that standard.

The purpose of a *report* is as follows:

> IEEE/EIA 12207.1-1997, subclause 5.5.1: Purpose: Describe the results of activities such as investigations, assessments, and tests.

Any *report* complying with IEEE/EIA Std 12207.1-1997 satisfies the generic content requirements provided in 5.5.2 of that standard. Table H.9 of this standard lists the generic content items and, where appropriate, references the clause of this standard that requires the same information.

Table H.9—Coverage of generic report requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
a) Date of issue and status	13.1.1 Date of issue, issuing organization, author, approvals, and status 14.1.1 Date of issue, issuing organization, author, approvals, and status 15.1.1 Date of issue, issuing organization, author, approvals, and status 16.1.1 Date of issue, issuing organization, author, approvals, and status 17.1.1 Date of issue, issuing organization, author, approvals, and status
b) Scope	13.1.2 Scope 14.1.2 Scope 15.1.2 Scope 16.1.2 Scope 17.1.2 Scope
c) Issuing organization	13.1.1 Date of issue, issuing organization, author, approvals, and status 14.1.1 Date of issue, issuing organization, author, approvals, and status 15.1.1 Date of issue, issuing organization, author, approvals, and status 16.1.1 Date of issue, issuing organization, author, approvals, and status 17.1.1 Date of issue, issuing organization, author, approvals, and status

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
d) References	13.1.3 References 14.1.3 References 15.1.3 References 16.1.3 References 17.1.3 References
e) Approval authority	13.1.1 Date of issue, issuing organization, author, approvals, and status 14.1.1 Date of issue, issuing organization, author, approvals, and status 15.1.1 Date of issue, issuing organization, author, approvals, and status 16.1.1 Date of issue, issuing organization, author, approvals, and status 17.1.1 Date of issue, issuing organization, author, approvals, and status
f) Introduction	13.1 Introduction 14.1 Introduction 15.1 Introduction 16.1 Introduction 17.1 Introduction
g) Context	13.2.1 Description 14.2.3 Context 15.1.2 Scope 16.1.2 Scope 17.1.2 Scope
h) Message	13.2.1 Description 14.2.3 Context 15.1.2 Scope 16.1.2 Scope 17.1.2 Scope
i) Contributors	13.1.1 Date of issue, issuing organization, author, approvals, and status 14.1.1 Date of issue, issuing organization, author, approvals, and status 15.1.1 Date of issue, issuing organization, author, approvals, and status 16.1.1 Date of issue, issuing organization, author, approvals, and status 17.1.1 Date of issue, issuing organization, author, approvals, and status
j) Body	13.2.1 Description 13.2.2 Activity and event entries 14.2.1 Summary 14.2.2 Date anomaly discovered 14.2.3 Context

IEEE/EIA Std 12207.1-1997 [B22] generic content	Corresponding subclauses of IEEE Std 829-2008
	14.2.4 Description of anomaly 14.2.5 Impact 14.2.6 Originator's assessment of urgency 14.2.7 Description of the corrective action 14.2.8 Status of the anomaly 15.2.1 Test status summary 15.2.2 Changes from plans 15.2.3 Test status metrics 16.2.1 Overview of test results 16.2.2 Detailed test results 16.2.3 Rationale for decisions 17.2.1 Overview of all aggregate test results 17.2.2 Rationale for decisions
k) Conclusions and recommendations	14.2.9 Conclusions and recommendations 15.2.4 Conclusions and recommendations 16.2.4 Conclusions and recommendations 17.2.3 Conclusions and recommendations
l) Bibliography	13.1.3 References 14.1.3 References 15.1.3 References 16.1.3 References 17.1.3 References
m) Glossary	13.3.1 Glossary 15.3.1 Glossary 16.3.1 Glossary 17.3.1 Glossary
n) Change history	14.3.1 Document change procedures and history 15.3.2 Document change procedures and history 16.3.2 Document change procedures and history 17.3.2 Document change procedures and history

H.5.3 Compliance with specific content requirements of IEEE/EIA Std 12207.1-1997 [B22]

The specific content requirements for a test report in IEEE/EIA Std 12207.1-1997 are prescribed by 6.29 of IEEE/EIA Std 12207.1-1997. A complying test report achieves the purpose stated in 6.29.1 and includes the information listed in 6.29.3 of that standard.

The purpose of the *test report* is as follows:

> IEEE/EIA 12207.1-1997, subclause 6.29.1: Purpose: Provide a record of the qualification testing performed on a software item, a software system or subsystem, or other software-related item. Enable the acquirer to assess the testing and its results.

A *test report* complying with IEEE/EIA Std 12207.1-1997 satisfies the specific content requirements provided in 6.29.3 of that standard. Table H.10 of this standard lists the specific content items and,

where appropriate, references the clause of this standard that requires the same information. The third column lists information that is to be added in order to comply with the specific content requirements.

Table H.10—Coverage of specific test report requirements by this standard

IEEE/EIA Std 12207.1-1997 [B22] specific content	Corresponding subclauses of IEEE Std 829-2008	Additions to requirements of IEEE Std 829-2008
a) Generic report information	See Table H.9.	
b) System identification and overview	13.2.1 Description 14.2.3 Context 15.1.2 Scope 16.1.2 Scope 17.1.2 Scope	
c) Overview of test results including	14.2.2 Summary 15.2.1 Test status summary 16.2.1 Overview of test results 17.2.1 Overview of aggregate test results	
c) (i) Overall assessment of the software tested	17.2.3 Conclusions and recommendations	
c) (ii) Impact of test environment	16.2.2 Detailed test results	
d) Detailed test results including (i) Test identifier	16.2.2 Detailed test results	
d) (ii) Test summary	14.2.2 Summary 15.2.1 Test status summary 16.2.1 Overview of test results 17.2.1 Overview of aggregate test results	
d) (iii) Problems encountered	13.2.2.4 Anomalous events 14.2.4 Description of the anomaly	
d) (iv) Deviations from test cases/ procedures	13.2.3. Environmental information 13.2.2.4 Anomalous events 16.2.2 Detailed test results	
e) Test log	13 Level test log	
f) Rationale for decisions	16.2.3 Rationale for decisions 17.2.2 Rationale for decisions	

H.5.4 Compliance with life cycle data objectives

In addition to the content requirements, life cycle data are to be managed in accordance with the objectives provided in Annex H of IEEE/EIA Std 12207.0-1996 [B21].

NOTE—The information items covered by this standard include plans and provisions for creating software life cycle data related to the basic type test data in H.4 of IEEE/EIA Std 12207.0-1996. It provides for the following test data: test strategy and criteria, cases (what to test), procedures (how to carry out tests), test results, and key decision rationale.

H.5.5 Conclusion

The analysis suggests that any test report complying with this standard and the additions shown in Table H.9 and Table H.10 also complies with the requirements of a test or validation report in IEEE/EIA Std 12207.1-1997 [B22]. In addition, to comply with IEEE/EIA Std 12207.1-1997, a test report supports the life cycle data objectives in Annex H of IEEE/EIA Std 12207.0-1996 [B21].